Robotics

en.wikibooks.org

August 9, 2015

Contents

1. Introduction **3**
1.1. Defining Robots 3
1.2. Student Questions 4

2. Contributors **5**

I. Design Basics **7**

3. What you should know **9**
3.1. Mechanics 9
3.2. Electronics 10
3.3. Programming 10
3.4. Solid Mechanics 11
3.5. A.I. 11
3.6. Math 12
3.7. See also 12

4. Physical Design **13**
4.1. General Design Considerations 13
4.2. Platforms 14
4.3. Electronics 17
4.4. Display 19
4.5. Mechanical Design 19

5. Design software **21**
5.1. 2D CAD 21
5.2. Solid Modeling 22
5.3. Pneumatic & Hydraulic Simulation 23
5.4. Schematic Capture & PCB 23
5.5. µControllers 25
5.6. Various tools 27
5.7. Further reading 27

6. Tools and Equipment **29**
6.1. Mechanical Tools 29
6.2. Electronic Tools 30
6.3. Breadboard 31
6.4. Electronic Equipment 32
6.5. Connectors 33
6.6. further reading 35

7. Electronic Components **37**
7.1. Basic Electronics 37
7.2. Special Electronic Components 37

8. Mechanical Components **39**
8.1. Gears 39
8.2. Internal gear 43
8.3. Chains & Belts 43

9. Building Materials **45**
9.1. Wood 45
9.2. Metal 45
9.3. Synthetic Materials 47
9.4. Composite materials 47
9.5. Other Materials 48

10. Basic Programming **49**
10.1. Programmable devices 49
10.2. Programming languages 52
10.3. Writing your software 52
10.4. Further reading 53

II. Physical Construction **55**
10.5. The Platform 57

11. Construction Techniques **59**
11.1. Working with Metals 59
11.2. Working with Plastics 60
11.3. Further reading 60
11.4. Resourcefulness 60

III. Components **63**

12. Power Sources **65**
12.1. Photo Voltaic Cells 65
12.2. Batteries 65
12.3. Fuel Cells 68
12.4. Mechanical 68
12.5. Air Pressure 68
12.6. Chemical Fuel 69

13. Actuation Devices **71**
13.1. Motors 71
13.2. Stepper Motors 73
13.3. Shape Memory Alloys 74
13.4. Air muscle 74
13.5. Linear Electromagnetic 83

13.6. Piezoelectric Actuators 84
13.7. Pneumatics Hydraulics 87
13.8. Miniature internal combustion engines 89

14. Grippers 91
14.1. Kind of grippers 91
14.2. Scoops 91
14.3. Tongs 92
14.4. 'Hands' 92
14.5. Degrees of Freedom 92
14.6. References 93

15. Audio 95
15.1. The Theoretical Edge 95
15.2. The Different Components 96

16. Video 99
16.1. Camera, transmission, display 99
16.2. Computer Vision 99
16.3. Demosaicing 99
16.4. Image Registration 100
16.5. Image Differencing 100
16.6. Image Recognition 100
16.7. Image Segmentation 100

IV. Computer Control 103

17. Control Architectures 105
17.1. Reactive Systems 105
17.2. Sense-Plan-Act 105
17.3. Brooks' Subsumption Architecture 106
17.4. Hybrid Systems 106
17.5. Swarm Robotics 106

18. The Interface 111
18.1. Personal Computers 111
18.2. SBC and multichip modules 116
18.3. Microcontrollers 119
18.4. Remote Control 121
18.5. Networks 123

V. Sensors 127

19. Environment sensors 129
19.1. Object Sensors 129
19.2. Media Sensors 129

20. Thermal Sensors 131
20.1. What is temperature? 131
20.2. Units and their conversions 131
20.3. Methods of measuring temperatures 132
20.4. Further reading 135

21. Pressure Sensors 137
21.1. Measuring Pressure 137

22. Ranging Sensors 141
22.1. Description 141
22.2. Sonic Sensors 142
22.3. Visible or Infrared Light-Based 145

23. Touch Sensors 147
23.1. Construction 147
23.2. Advantages 149
23.3. Disadvantages 149

24. Feedback Sensors 151
24.1. further reading 151

25. Communication Sensors 153
25.1. Data transmission channels 153
25.2. Using IR 153
25.3. Further reading 154

26. Real World Sensors 155
26.1. Light Based Sensors 155
26.2. Sound Based Sensors 157
26.3. Ultrasonic Distance sensors 158
26.4. Magnetism Based Sensors 159
26.5. Other distance sensors 160
26.6. Further reading 160

27. Digital image Acquisition 161
27.1. Image Sensors 161
27.2. Color Images 162
27.3. Image Transfer 162
27.4. References 162

28. Navigation 165
28.1. What is Navigation? 165
28.2. Localization 165
28.3. Current Location 166
28.4. Dead Reckoning 166
28.5. Least Mean Squares 166
28.6. GPS Global Positioning System 169
28.7. GPS-like Systems 169

28.8. Line following 169
28.9. Landmarks 170
28.10. Current Heading 170

29. Collision Avoidance 171
29.1. Basic Idea 171
29.2. Example Algorithm 171
29.3. Major Problem 172
29.4. Detection vs. Avoidance 172
29.5. Collision Avoidance Algorithms 173
29.6. References 173
29.7. Exploration 173
29.8. Mapping 174
29.9. Trajectory Planning 174
29.10. Trajectory Planning 174

30. Exotic Robots 181
30.1. Special Robot Brains 181
30.2. Mechanical Robot Brains 181
30.3. Pneumatic Robot Brains 181
30.4. BEAM robotics 181
30.5. Cooperating Robots 181
30.6. Hazardous Environment Robots 182
30.7. Mars Pathfinder 183
30.8. Hubble Telescope 188
30.9. Pioneer 189
30.10. pipeline inspection 189
30.11. Talon Sword 189
30.12. Predator 190
30.13. Conclusion 190

31. Types of Robots 191
31.1. Contest Robot 192
31.2. Entertainment Robot 194
31.3. Arms 194
31.4. Wheeled 198
31.5. Tracked 209
31.6. Walkers 209
31.7. References 213
31.8. Modular and fractal robots 214
31.9. References 215
31.10. Further reading 215

32. The LEGO World 217
32.1. Why Lego? 217
32.2. History 218
32.3. Standard Components 218
32.4. Programming Options 219

32.5. Third Party Accessories 221
32.6. Lego Robotics Events, Challenges, and Acheivements 222
32.7. References 223
32.8. LEGO Robots 223
32.9. Interface 223
32.10. Programming and Software 223

33. Resources **225**
33.1. Other Wikibooks 225
33.2. External links 225
33.3. Free Wikis on Robotics 226
33.4. University and College Sites 226
33.5. Papers 226

34. Contributors **229**

List of Figures **235**

35. Licenses **241**
35.1. GNU GENERAL PUBLIC LICENSE 241
35.2. GNU Free Documentation License 242
35.3. GNU Lesser General Public License 243

The current version of this book can be found at `http://en.wikibooks.org/wiki/robotics` *.*

1. Introduction

Robotics can be described as the current pinnacle of technical development. Robotics is a confluence science using the continuing advancements of mechanical engineering, material science, sensor fabrication, manufacturing techniques, and advanced algorithms. The study and practice of robotics will expose a dabbler or professional to hundreds of different avenues of study. For some, the romanticism of robotics brings forth an almost magical curiosity of the world leading to creation of amazing machines. A journey of a lifetime awaits in robotics.

Robotics can be defined **as the science or study of the technology primarily associated with the design, fabrication, theory, and application of robots** . While other fields contribute the mathematics, the techniques, and the components, robotics creates the magical end product. The practical applications of robots drive development of robotics and drive advancements in other sciences in turn. Crafters and researchers in robotics study more than just robotics.

The promise of robotics is easy to describe but hard for the mind to grasp. Robots hold the promise of moving and transforming materials with the same elan and ease as a computer program transforms data. Today, robots mine minerals, assemble semi-processed materials into automobile components, and assemble those components into automobiles. On the immediate horizon are self-driving cars, robotics to handle household chores, and assemble specialized machines on demand. It is not unreasonable to imagine robots that are given some task, such as reclaim desert into photovoltaic cells and arable land, and left to make their own way. Then the promise of robotics exceeds the minds grasp.

In summary, robotics is the field related to science and technology primarily related to robotics. It stands tall by standing the accomplishments of many other fields of study.

1.1. Defining Robots

Robot used in English describes any construct that automates some behavior. For example, a garage door opener automates the behavior of opening a door. A garage door opener has a sensor to detect the signal from the remote control, actuators to open the door, and a control system to stop turn off the motors and lights when the garage is fully closed. In practice, this type of a machine is better described as a Mechatronic device, and is a subset of the more interesting robots that include **autonomy** or **resourcefulness** . This book will consider mechatronic devices to be degenerate robots.

A **Mechatronic Device** is a degenerate robot with these components:

1. **Sensors** , which detect the state of the environment
2. **Actuators** , which modify the state of the environment

3. A **Control System** , which controls the actuators based on the environment as depicted by the sensors

A **Robot** is a mechatronic device which also includes **resourcefulness** or **autonomy** . A device with **autonomy** does its thing "on its own" without a human directly guiding it moment-by-moment. Some authors would contend that all mechatronic devices are robots, and that this book's restriction on **robot** entails only specialized software.

Various types of robots[1] are usually classified by their capabilities. Two examples will be used to capture most of what we see as a "robot".

1. Machine Pet: A machine, capable of moving in some way, that can sense its surroundings and can act on what it senses autonomously. Most of these robots have no real useful purpose, other than to entertain and challenge. These are also commonly used for experimenting with sensors, artificial intelligence, actuators and more. Most of this book covers this type of robot.
2. Autonomous Machine: A machine with sensors and actuators that can do some sort of work "on its own". This includes things like robotic lawnmowers and vacuum cleaners, and also self-operating construction machines such as CNC cutters. Most industrial and commercial robots fall in this category.

What isn't considered a "robot" in this book? Pretty much everything you see on Robot-Wars; those are remote-controlled vehicles without any form of autonomy, no sensors, and just enough of a control system to drive the actuators. These devices use many of the same mechanical technologies described in this book, but not the advanced controls.

In short: If it has autonomy it's a robot (in this book). If it's remote controlled, it isn't.

1.2. Student Questions

1. Which of these studies would be considered robotics by this definition?
   a) Studying the strength and flexibility of a titanium alloy used to make a robotic arm?
   b) Integrating sensor data from sonar, laser, and CCD cameras and to build an accurate map of surroundings?
   c) The real-time software needed to drive two motors to make a robot go in a straight line?
2. Classify each of these as a robot, a mechatronic device, a machine, or something else?
   a) A spam email filter.
   b) A garage door opener.
   c) A remote controlled boat.
   d) A 1970s automobile.
   e) A current model automobile which includes lane-following.
   f) An Apple iPod.
   g) An actor in a silver suit.
3. What is a sensor?

1 Chapter 31 on page 191

2. Contributors

- CpE/EE 300: Introduction to Robotics[1] class at the Missouri University of Science and Technology[2] - Updates/Reworks to various pages
- T.R. Darr[3] - responsible for the (almost) complete reformat. If I knew anything about robotics, then I'd have contributed to the content as well.
- J.D. Cox[4] - Attempting to fill in certain areas with basic information.
- Omegatron[5] - I've built a handful of short-lived little robots, and since then I went and got myself an electronics degree. I'll probably just add to and clarify things that other people have contributed. I tend to only contribute to things that are already active, so be active!
- Patrik[6] - As time permits I'm adding more info I've found to be missing in many other sources. I've got a degree in electronics and I've designed and build several robots.
- E. Sumner[7] - Active member of the Dallas Personal Robotics Group; Trying to flesh things out a bit here.
- Mr Dom[8] - just added my two cents worth
- DavidCary[9] - degree in electrical engineering. So *in theory* I ought to know :-).
- Magnus Persson[10] - studying for Master of Science in Automation Engineering, added sections on PLCs and wireless communications.
- Piyoosh Mukhija[11] - Degree in Electronics & Communication Engineering. Working on Autonomous Robotics Research at L&T Infotech. Attempting to fill in some missing things I believe I know about.
- S.J Manderson[12] - Physics Student in New Zealand. Added sections on Electromagnetic Actuators and Pneumatics thus far.

1 `http://introrobotics.googlepages.com/`
2 `http://www.mst.edu`
3 `https://en.wikibooks.org/wiki/user%3A%20nsoyeblcyha`
4 `https://en.wikibooks.org/wiki/user%3A%20Joel7687`
5 `https://en.wikibooks.org/wiki/User%3AOmegatron`
6 `https://en.wikibooks.org/wiki/user%3A%20Patrik`
7 `https://en.wikibooks.org/wiki/user%3A%20kd5bjo`
8 `https://en.wikibooks.org/wiki/user%3A%20dom123`
9 `https://en.wikibooks.org/wiki/User%3ADavidCary`
10 `https://en.wikibooks.org/wiki/User%3ATERdON`
11 `https://en.wikibooks.org/wiki/User%3Apiyoosh`
12 `https://en.wikibooks.org/wiki/User%3ASamEEE`

Part I.

Design Basics

3. What you should know

Robotics spans multiple scientific and engineering disciplines, so when you want to design a better robot you should get some basic knowledge in these fields. How much you should learn depends on how complex you want to make your robot. To give an example: A small tethered tabletop robot would only require some basic knowledge in electronics and programming, a shoe box sized robot would require some additional knowledge on mechanics (mostly about balance) and a large robot might even require some knowledge on solid mechanics.

This page covers the fields that are very much used in robotics. You don't need to know everything about all of these subjects, however knowing the basics of each of these fields can help in building better robots and prevent you from making (some of the) beginner's mistakes.

3.1. Mechanics

Mechanics is about:

- how forces are transferred between the different parts of a construction.
- where the center of gravity lies.
- friction
- position, speed, acceleration
- Newton's laws
- inertia
- material properties

Mechanics helps keeping a robot in balance. Although you could build a robot without knowing anything about mechanics, it'll help in preventing your robot from tipping over when turning, or when picking up something.

Another point where mechanics pays off are axles. On small robots you can attach the wheels directly to the output shaft of the motor. However this doesn't work well for larger robots as this puts a lot of stress on the internals of the motor. A better way is to attach the wheel to an axle, and use gears to connect the motor to the axle. Knowledge of mechanics allows you to build such constructs.

If your robot is a small line follower almost any building material will work. However if your robot weighs a few kilos, something sturdier than cardboard and soda sippers is appropriate. And if your robot is human size you should consider metal and/or composite construction.

See Theoretical Mechanics[1] for an introduction in this field.

the "Theoretical Mechanics" is just started, so there isn't much to read just yet.

3.2. Electronics

Electronics is about:

- Electronic Components
- Analog Circuits
- Digital Logic
- MicroControllers

Electronics is something you can't go without (unless you want to build a complete mechanical robot or use pneumatics for control). Today there are plenty of books covering basic electronics (See Electronics[2]).

3.3. Programming

Computer programming is about:

- Control structures (sequence, selection, iteration)
- Data types (constants, variables, integer, real, string,...)
- Algorithms
- Hardware control (setting and reading registers, interrupts,...)
- logic

Anyone who has had an introductory course on programming (as they are given in American high schools) would be familiar with the first three points. The fourth point is rarely addressed in introductory courses, but is essential when programming microcontrollers. Although it might sound difficult, it can be very easy in practice (for most purposes). Much of this comes down to setting bits in a byte using simple Boolean logic, and writing this value into some register or memory location. Higher level languages like Bascom provide hardware addressing as special variables, which can be treated just like any other variable.

Microcontrollers (and processor boards) are one of the areas where using Assembly is still very valid. Memory (both RAM and program space) is very limited in these, although each new generation of microcontroller has more memory for about the same price. Many microcontrollers provide between 2K and 30K, and processor boards tend to have up to 256K. These numbers vary wildly, but are still significantly less than PCs have. However if you don't know an assembly language, most microcontroller and processor boards have high level language compilers available in many flavors (C, C++, Basic, Pascal, Fortran, etc.)

Robot programming is also about:

1 `https://en.wikibooks.org/wiki/Theoretical%20Mechanics`
2 `https://en.wikibooks.org/wiki/Electronics`

- The Event Loop. Most microcontrollers do not have the resources for threading. You will need to look at your robot's task a fraction of a second at a time, and choose which small action to take. What can the program do in that moment to get a little bit closer to its goal?
- Interpreting sensor data. Sensors have many ways of providing noisy or misleading information; how many types of error can you accept? Dirt in a rheostat, a broken switch, and a venetian blind flickering on a photodiode are obvious faults. How about temperature drift, non-linear response curves, or your robot seeing its own shadow?
- Decision making, or **Artificial Intelligence** is the art of making the right decision given the constraints of the current system.
- Motors and motion. Making motion on a robot often involves moving several motors at once, often with feedback from sensors.

3.4. Solid Mechanics

Solid mechanics is about how forces distribute inside solid materials. Knowledge about this subject is useful because it explains how materials respond to loads. This helps to prevent using too thick or too thin materials. This isn't required for small or medium robots, but it allows to be more efficient with building materials and gives insight to why and how materials fail (break and/or deform). See `http://en.wikibooks.org/wiki/Solid_Mechanics` this wikibook[3] for a start on solid mechanics. Be warned: heavy math ahead.

Even if you have a mortal fear of math, bite through this as it gives valuable insights on how materials break and deform. No need to memorize the math, as long as you get the idea behind it.

3.5. A.I.

Artificial Intelligence (in Robotics) is about:

- Finding the shortest way between 2,3 (or more) points
- Dealing with obstacles
- Handling new situations (machine learning)

There are many books[4] available on AI on many different levels. This area has had a short but already fruitful history, but still has a very long way to go. AI isn't just about getting a computer to think and reason. It's more about ordering, sorting and organizing knowledge in a machine and constructing algorithms for extracting real world conclusions from these databases. A search engine like Google[5] or Yahoo[6] are examples of real uses of AI.

3 `https://en.wikibooks.org/wiki/%20this%20wikibook`
4 `https://en.wikibooks.org/wiki/Artificial%20Intelligence`
5 `http://www.google.com`
6 `http://www.yahoo.com`

Other than pure AI books, books on how the brain works and such can provide interesting angles to AI on robots. Concepts like attention and concentration can have interesting uses in some form for integrating sensor data.

3.6. Math

Although math is generally seen as the ultimate theoretical science, it can be one of the most important skills in many of the more advanced domains of robotics. e.g. Mechanics uses a lot of math. For simple constructs you won't need much more than high school level math, for more complicated shapes it becomes necessary to use more complex math tools like integrals. But since robotics is a very practical craft many things can be done with approximations. Math however can be very helpful in making the right approximation.

3.7. See also

- Open hardware[7]

7 https://en.wikipedia.org/wiki/Open%20hardware

4. Physical Design

4.1. General Design Considerations

Designing a robot requires balance between size (mostly weight), motor power and battery power. These three elements are connected with each other (more battery power increases the weight of the robot and requires stronger motors) and finding the ”perfect” balance requires a lot of tweaking and experimenting. Try to describe heavy components in output/mass (e.g. motors: torque/Kg; batteries: mAh/Kg) and pick the one that gives the highest value.

Using light materials brings down the weight significantly (aluminum instead of steel). Building a frame out of light metal and using plastic plates as surfaces would be a lot lighter than using metal plates. For small robots acrylic plastic is a good material to use and it is easy to work with.

There are other ways to build a robot than to cut and drill your own aluminum plates. Toys like Lego Technic and Meccano, although expensive, are an alternative when you don't have the ability to cut and drill your own parts. Especially Meccano (or better: the cheap imitations) can be useful even when you do make your own parts. There is something convenient about having a collection of parts with standard holes and sizes. Of course you would need to drill your own holes at the right distances and size, if you intend to add Meccano parts. The values can (most likely will) differ between different ”brands” of imitations. So it's a good idea to buy a few boxes when they are on sale. Screws tend to be M5 (on the set I've seen so far this is the only value that is common. The hole spacing very rarely matches up).

Another kit that would be good to use is the Vex™ Robotic Design Kit from Radio Shack. The parts in this kit are all metal with holes predrilled every half an inch. That makes it easier to add to parts that you may already have. This kit gives you everything you need to get started with robots. If you do not want to buy the kit you could just buy the Vex™ Metal and Hardware Kit for Robotics Starter Kit.

When you start your design, first decide how big you want your robot to be. Don't think about exact sizes, compare it to the size of an object (”the size of a shoe box”) will be sufficient. Exact values can be ”calculated” after you've got your motors and batteries, as these have a large influence on the size and shape of the robot.

Make an estimate about the weight of the complete robot and pick your motors and wheels. Keep in mind you need high torque and low speed. A bare DC-motor has high speed and low torque, adding a gear reduction will solve this one. Motors with reduction gears are also available. The speed of your motor and the size of your wheels determine how fast your robot will be able to move.

For example: RB-35 is a motor with a 1:50 reduction. It makes 120 RPM or 2 rounds per second. Lets pick a wheel with a diameter of 20cm (radius R = 10cm). This wheel has a circumference of 2 x π x R = 2 x 3.14 x 10 = 62.8cm. This means in one turn the wheel moves 62.8cm. When we mount this wheel on the motor it'll turn twice every second and therefore move 2 x 62.8cm = 125.6cm. So its speed would be 125.6cm/second or 1.256m/s.

In reality this speed is going to be a little lower as the motor turns 120RPM without a load. But even a 1 m/s is pretty fast for indoor robots. You'll probably use PWM or other methods to slow it down.

Pick your batteries. Make sure you have enough power to keep the motors and all the electronics running for a sufficient amount of time and keep some reserve for future additions. Compare the weight of the batteries and motors you've chosen to what you had planned. You might need to go over this part again (picking different motors and/or batteries).Keep in mind that your robot's body has a significant weight.

4.2. Platforms

4.2.1. Wheeled Platforms

Wheeled platforms can have any number of wheels. Most common are 3, 4 and 6 wheeled vehicles (excluding wheels used for feedback). Other numbers are also possible, but can be hard to build, such as 1-wheeled or 2-wheeled robots, or have superfluous wheels which can make turning difficult or complex. Basically there are 2 types of wheels: powered wheels and unpowered wheels. The first are powered by the motors and are used to move the robot forwards (or backwards). Unpowered wheels are used to keep the robot in balance by providing a point of contact with the ground.

Turning

Turning can be accomplished in several different ways:

- Differential Steering (Tank-like Turning):
  - Moves one wheel forward and the other backwards. The robot turns around within a small circle which center lies in between the 2 powered wheels.
  - Move one wheel slower than the other, the robot turns in the direction of the slower wheel. How fast it turns depends on how large the difference between the 2 speeds is.

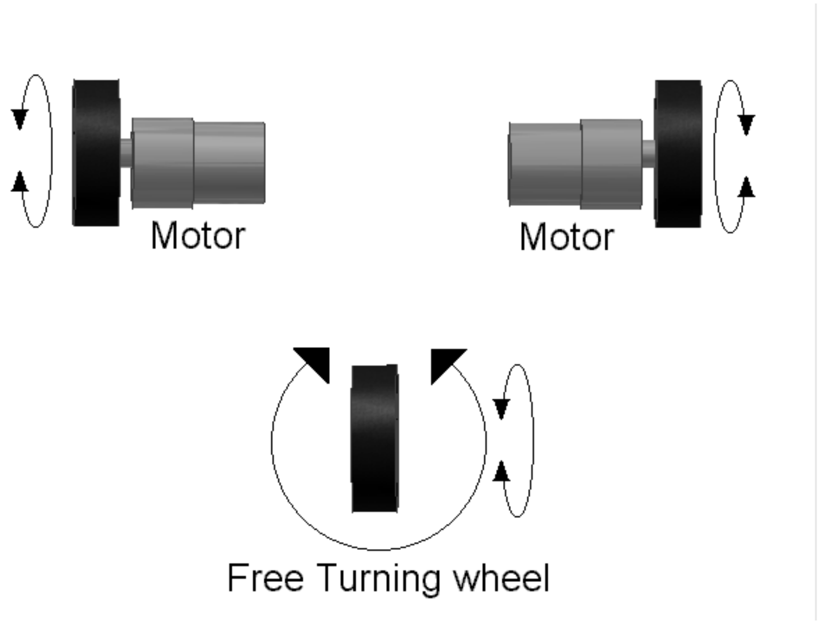


Figure 1 Tank-Like Steering

- Ackerman Steering: This is the same steering system as the one used in cars. It is relatively complicated to implement since the inner and outer wheels need to turn to different angles.
- Crab Drive: Each wheel can turn independently in crab drive steering. This can be very flexible, but requires complex mechanics which either turn the entire motor/gearbox/wheel assembly or transfer power from a statically mounted motor. The second option is much more difficult to build but may have advantages over the first.
- 3-wheeled platforms: These can come in a variety of forms, with the articulated wheel powered, or with the two fixed wheels powered, or a combination of the two. These are generally built for very specific purposes.
- Omnidirectional wheels: The omnidirectional wheels design is based upon the use of a series of free turning barrel-shaped rollers, which are mounted in a staggered pattern around the periphery of a larger diameter main wheel. For this you need 4 powered wheels. However these wheels allow movement in any direction without turning (including sideways and diagonal movement) and can turn the same way as in tanklike steering. Building these wheels is time-consuming, but it's a very powerful steering method. Also, inexpensive omnidirectional wheels are available commercially, often used in conveyors. One drawback, however, is the lack of sideways traction; if something is pushing the robot to the side, it relies on the strength of the motor or brakes to restrain it. Omnidirectional wheels used in place of caster wheels can provide quicker responses and can often roll over larger obstacles.

4.2.2. Tracked Platforms

Tracked platforms use tracks similar to tanks. This kind of propulsion is only useful on loose sand and mud, as concrete and carpet provide too much horizontal traction when turning and will strip the tracks off of their guides.

4.2.3. Walkers

Walkers are robots that use legs instead of wheels or tracks. These robots are harder to build than wheeled robots and can be a nice challenge for an experienced builder. Walkers are designed to imitate how animals (or humans) move.

2-Legged Walkers or Bipeds

This is the hardest type of walker. This type tries to imitate how humans walk. The biggest issue is balance.

Two-legged walkers are used for two main purposes: to imitate humans and to provide a great amount of force and traction. Taller walkers used to imitate humans are difficult to build, requiring many balancing circuits and devices, quick motions, and precise construction. Just like any human knows, these can also be knocked over, tripped, etc. Shorter, wider walkers can be used to move a large load. When using walkers, it is possible to use pneumatic systems, which can provide a much larger force than motors. However, turning with such a system is nearly impossible.

4-Legged Walkers

4 Legged walkers imitate 4 legged animals. Many of these designs end up moving one leg at a time, instead of the 2-legged movements typical of animals. It requires 3 legs to be on the ground to provide static balance. Dynamic balance moving 2 legs at a time provides faster and more fluid motion.

6-Legged Walkers or Hexapods

These walkers are imitations of insects. Many of these move 3-legs-at-a-time to provide static balance. Because half of the legs can be moved at one time without losing static balance, 6-legged walkers can actually be simpler to build than 4-legged.

Note: Static balance means the construction is at all time in balance. This means that if the robot would stop moving at any time it wouldn't fall over. In contrast there is *Dynamic Balance* . This means that the robot is only in balance when it completes its step. If it's stopped in the middle of its step it would fall over. Although this might sound like a bad thing, dynamic balance allows much faster and smoother movement, but requires sensors to sense balance. Animals and humans move with dynamic balance.

4.2.4. Whegs

There are various combinations of wheels and legs that are useful for varying terrain. For quick details see `http://biorobots.cwru.edu/projects/whegs/whegs.html`.

4.2.5. Ball Wheels

This means of propulsion is very similar to how a classic computer mouse works: A ball is mounted in a casing in such a way that it can freely rotate in any direction. Two wheels around the ball are mounted against this ball at an angle of 90° to each other, parallel to the ground. One wheel registers the up-down movements and the other the left-right movements.

A ball wheel uses the same setup but connects the internal wheels to motors. This way the ball can be made to rotate in any direction. A robot equipped with a ball wheel can move up-down and left-right, but can't rotate around its vertical axis. Using 3 ball wheels allows rotation as well.

4.3. Electronics

The electronics of a robot generally fall in 6 categories:

- Motor control: controls the movement of the motors, servos and such. Relays and PWM H-bridges fall under this category.
- Sensor reading: reads the sensors and provides this information to the controller.
- Communication: Provides a link between controller and an external PC, another robot, or a remote control.
- Controller: microcontroller board, processor board or logic board. This part makes decisions based on sensor input and the robot's program.
- Power management: parts that provide a fixed 5VDC, 12VDC or any other level coming from the batteries. Circuits that monitor the status of the batteries.
- Glue logic: Additional electronics that allow all the parts to be connected with each other. An example is a CMOS to TTL level converter.

Not all of these categories are present in all robots, nor does every circuit fall completely into one category. Many robots don't require a separate sensor board as a whole lot of sensors have built-in electronics which allow them to connect directly to a µcontroller/processor.

4.3.1. Some Tips

- Use low-power (or simply dimmer) LEDs. Always. This drastically reduces how much current your circuits consume. A normal LED consumes around 15 mA. A modern microcontroller consumes about the same. Disabling unneeded LEDs is advisable, but not always possible.

- Use CMOS ICs instead of classic TTL. Again this reduces current usage and allows a more relaxed supply voltage. But pay attention to their sensitivity for static electricity when soldering them.
- Use good quality IC-sockets (or better don't use IC-sockets at all). They're worth their price.
- Avoid using IC-sockets on sensitive circuits (high speed digital, clock signal and analog signals). The moving robot can shake those ICs loose over time. Using In-Circuit programmable microcontrollers removes the need to be able to unplug an IC.
- LEDs are very practical to make slow digital signals visible, adding them on some signal lines can be interesting for testing purposes, however, they do increase power consumption. Removing them when your circuit works correctly can make it more power efficient (replace LED with a wire, replace resistor with a higher value one).
- See if your microcontrollers can run with a lower clock speed. The higher the clock speed, the more they consume.
- Use your microcontrollers sleep functions whenever possible, disable any part that isn't required (e.g. on-chip ADC).
- Learn to make PCBs (Printed Circuit Boards). It's not that hard and it makes your electronics look more professional. Don't throw away your breadboard, PCBs aren't very practical for prototyping.
- If you're up to it: build your circuits in SMD components on PCBs. This reduces size, weight, and cost. However soldering SMD ICs isn't easy. SO-packages (Small Outline) aren't too hard for an experienced builder. Smaller packages are almost impossible to do by hand. Get good soldering tools before you attempt this. For many modern SMD microcontrollers are complete build and tested board available. These can be a solution for those who don't have the equipment, expertise and patience to solder these ICs themselves. These boards don't have to be large as the demo-boards most IC developers sell, e.g. the BasicStamp is such a board with the size of an IC.
- Buy a breadboard. They're invaluable for designing and testing circuits.
- If you intent to build your own electronic circuits, invest in a dual channel oscilloscope. Single channel is very restricting. Pick one with as high a bandwidth as you can afford. At least 4x the highest signal frequency you intend to use.
- A good variable power supply is very handy to test how your circuit operates at lower voltage levels (as happens when the batteries discharge).
- If you can choose between pull-up and pull-down resistors, pick the one that uses the lowest amount of power. If the circuits output is at +5V most of the time, use pull-up, if it's 0V use pull-down. Remember that such outputs consume power when the transistor is active (when it's inactive it consumes a small amount of power: a leak current through the resistor and through the input impedance of the next circuit).
- Use high value resistors as pull-up/pull-down. But keep in mind that high speed signal lines need lower value resistor in order to minimize signal distortion.

- Most electronic components on a robot will run on a 5V supply and need a 5V regulator. Use a regulator with low dropout to prevent the 5V supply from browning out.

4.4. Display

Very simple robots need only a few LEDs to show everything that it is ”thinking”.

When trying to debug more complex robot software, it is useful for the robot to display text. A few calculators and PDAs have a RS232 connector or some other simple way to connect to a robot.

Many robots have such a calculator or PDA or other display strapped to the top in order to show the humans what the microcontroller is ”thinking”, which is vastly more productive than trying to guess what's going on inside that little chip of silicon.

With large robots, sometimes a full-size laptop computer is strapped on top for such display purposes.

4.5. Mechanical Design

4.5.1. Balance

Everybody encounters balance every day. While walking, when putting down a glass of water or in so many other ways we have to keep a balance. Now in most of these cases you wouldn't need to think about it, but when designing your robot you'll have to keep an eye on this concept. For most designs, balance isn't hard to achieve, even without doing calculations on it. A few rules of thumb suffice. For more complex designs, e.g. a robot with an arm, you can get away with just some rules of thumb and some common sense, but there's no guarantee. Doing a simple calculation would make it clear if the robot is going to stay on its wheels or if it is going to tumble over and crush whatever it was trying to pick up.

If you're into walkers, you'll need to spend more attention to balance. The fewer amount of legs you have, the more important balance becomes.

4.5.2. Simple 4-Wheeled Robots

Achieving balance on this type of robot is pretty trivial. Keep the center of mass between the wheels (picture a rectangle between the centers of the wheels) and as low as possible. In practice it mean you should place the heavy components, e.g. the batteries, somewhat in the center of the robot and as low as possible.

4.5.3. Simple 3-Wheeled Robots

These designs are nearly as simple as 4-wheeled robots; the difference lies in that you need to keep the center of mass close to the center of the triangle formed by the wheels. If your robot is rectangular avoid placing weight at the two unsupported corners. These points are prone to making the robot tumble over.

4.5.4. Wheeled robots with an arm or gripper

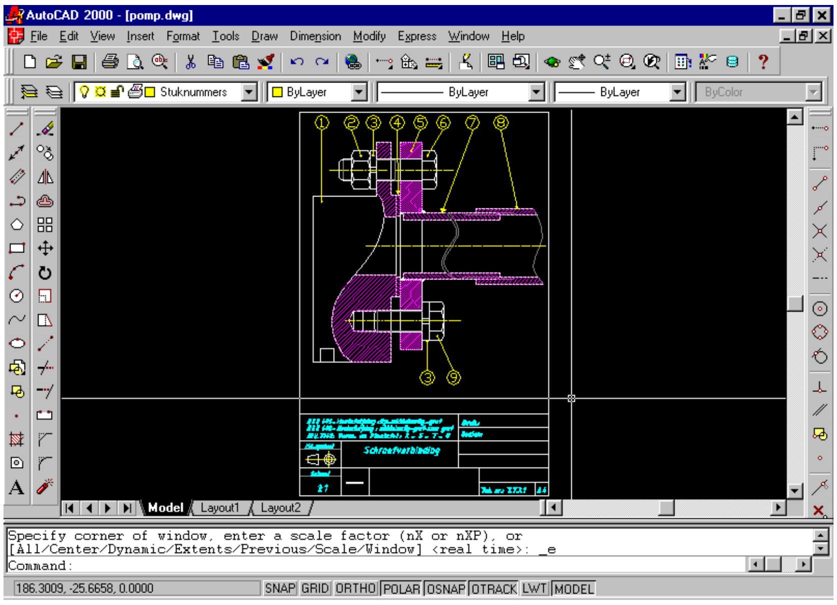
For the working of an arm or a gripper, we need to take the help of a stepper motor in simple cases or use sensors in sophisticated cases.

5. Design software

Robotics[1]: Design Basics: **Design software**

When designing your robot there are plenty of programs to help. Ranging from a simple tool to print wheel encoders, through CAD drawing programs up to mechanical simulation programs.

5.1. 2D CAD


AutoCAD 2000 - [pomp.dwg]
File Edit View Insert Format Tools Draw Dimension Modify Express Window Help
Stuknummers
ByLayer
ByLayer
ByLayer
ByColor
Model
Layout1
Layout2
Schroefverbinding

```
Specify corner of window, enter a scale factor (nX or nXP), or
[All/Center/Dynamic/Extents/Previous/Scale/Window] <real time>: _e
Command:
186.3009, -25.6658, 0.0000   SNAP GRID ORTHO POLAR OSNAP OTRACK LWT MODEL
```

Figure 2

Autocad 2000

e.g. AutoCAD. This type of software is used to turn a rough sketch into a nice professional drawing. This type of drawing is standardized for readability. (Meaning every different type of line has a particular meaning. Solid lines are visible edges, dashed lines are hidden edges, line-dash-line lines are center lines. Standards also include methods of dimensioning and types of views presented in a drawing.) Of course you're free to use your own standards, but using an industrial standard, such as ANSI or ISO, makes it easier to share your plans with other people around the world. While it may somewhat more tedious to make a drawing using 2D software, the results are generally better than using 3D solid modeling software.

1 `https://en.wikibooks.org/wiki/Robotics%23Design%20Basics`

Solid modelers still have problems translating 3D models into 2D drawings and adding proper notation to standards.

- Freeware CAD software: CADstd[2]
- Professional CAD software: AutoCAD[3]

- GPL CAD software: QCad[4] Wikipedia:QCad[5]
- GPL vector drawing software: Inkscape[6] Wikipedia:Inkscape[7]

5.2. Solid Modeling

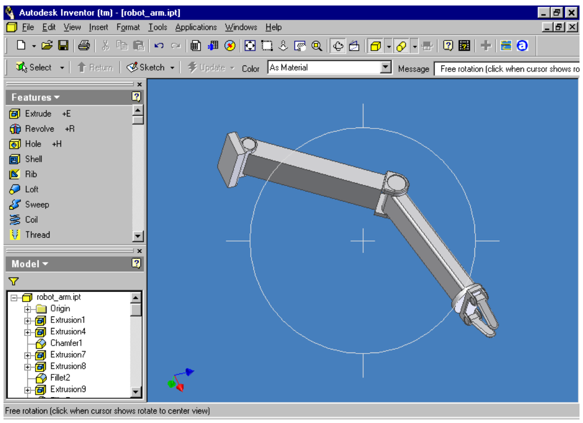

Autodesk Inventor (tm) - [robot_arm.ipt]
File Edit View Insert Format Tools Applications Windows Help
Select
Return
Sketch
Update
Color
As Material
Message
Free rotation (click when cursor shows ro
Features
Extrude +E
Revolve +R
Hole +H
Shell
Rib
Loft
Sweep
Coil
Thread
Model
robot_arm.ipt
Origin
Extrusion1
Extrusion4
Chamfer1
Extrusion7
Extrusion8
Fillet2
Extrusion9
Free rotation (click when cursor shows rotate to center view)

Figure 3

Inventor 5.3 Rotating a part

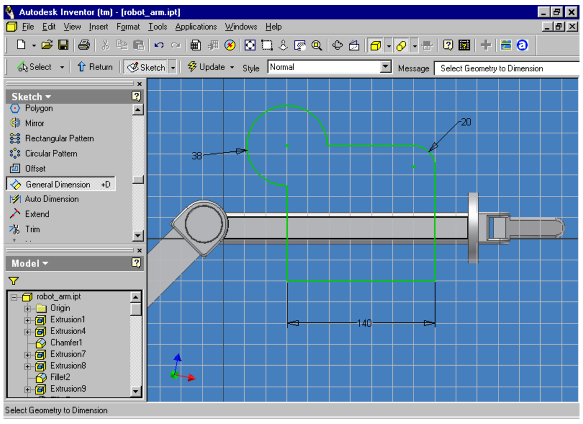


Figure 4

Inventor 5.3 Drawing a Schematic on a part

2 http://www.cadstd.com/
3 http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=2704278
4 http://www.ribbonsoft.com/qcad.html
5 https://en.wikipedia.org/wiki/QCad
6 https://en.wikibooks.org/wiki/Inkscape
7 https://en.wikipedia.org/wiki/Inkscape

e.g. SolidWorks or Pro/Engineer[8] Pro/Engineer[9] (Wikipedia:Pro/ENGINEER[10]). A newer way to draw parts and machines. With solid modeling you "build" the parts in 3D, put them together in an assembly and then let the software generate the 2D drawings (sounds harder than it is). The major advantage over 2D CAD programs is you can see the complete part/machine without actually building it in real life. Mistakes are easily found and corrected in the model. These 3D models are not yet completely standardized though there is a standard for digital data. At this time the 2D drawings this software generates do not conform completely to industrial standards. The 2D paper drawing is still the communication tool of preference in industry and clarity of intent is very important. Solid modeling software tend to generate overly complex drawing views with overly simplified dimensioning methods that likely do not correctly convey the fit, form or function of the part or assembly.

- BRL-CAD[11] c2:BrlCad[12] Wikipedia:BRL-CAD[13]
- lignumCAD (GPL)[14]

5.3. Pneumatic & Hydraulic Simulation

Festo[15] has a demo version of both a pneumatic and a hydraulic simulation program. Look for FluidSIM Pneumatiek and FluidSIM Hydraulica. (Pick country; click on industrial automatisation; and use the search field to the right.)

Limitations: Can't save nor print. Most of the didactic material isn't included.

IRAI[16] has a free demonstration version of electric / pneumatic and hydraulic simulation software : AUTOMGEN / AUTOMSIM. Go to Download / AUTOMGEN7.

5.4. Schematic Capture & PCB

8 `https://en.wikibooks.org/wiki/Pro_Engineer%20`
9 `http://en.wikibooks.org/wiki/Pro_Engineer`
10 `https://en.wikipedia.org/wiki/Pro%2FENGINEER`
11 `http://c2.com/cgi/wiki?BrlCad`
12 `https://en.wikibooks.org/wiki/c2%3ABrlCad`
13 `https://en.wikipedia.org/wiki/BRL-CAD`
14 `http://lignumcad.sourceforge.net/doc/en/HTML/index.html`
15 `http://www.festo.com`
16 `http://www.irai.com`

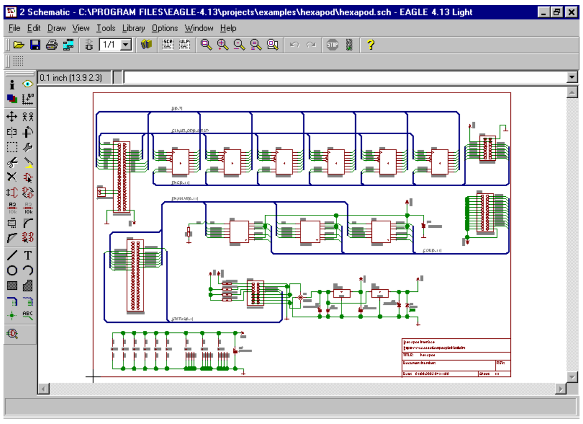

2 Schematic - C:\PROGRAM FILES\EAGLE-4.13\projects\examples\hexapod\hexapod.sch - EAGLE 4.13 Light
File Edit Draw View Tools Library Options Window Help
0.1 inch (13.9 2.3)

Figure 5
Eagle Schematic capture

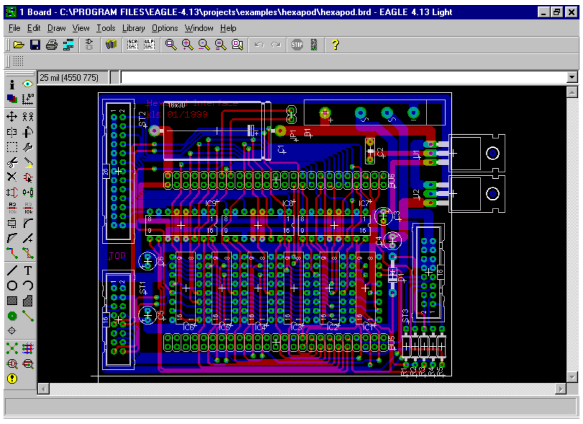

1 Board - C:\PROGRAM FILES\EAGLE-4.13\projects\examples\hexapod\hexapod.brd - EAGLE 4.13 Light
File Edit Draw View Tools Library Options Window Help
25 mil (4550 775)

Figure 6
Eagle printed circuit board design

Software to draw electronics schematics and designing Printed Circuit Boards (PCBs). These packages contain software to draw the schematic, libraries with symbols, and software to draw the PCBs (with autorouter).

In no particular order:

- Freeware: Eagle[17] is commonly used by beginners for their projects because a limited version is available for free. The toolset is well integrated and has a large hobbiest user base. However, once you progress beyond basic designs, you need to pay for the full version.
- Open Source: The open-source gEDA Project[18] has produced a mature suite of applications for electronics design, including: a schematic capture program, attribute manager, netlister supporting over 20 netlist formats, analog and digital simulation, PCB layout with autorouter, and Gerber viewer. The project was started in 1997 to write EDA tools useful for personal robotics projects, but as of this writing the tools are also used by hob-

17 `http://www.cadsoft.de/`
18 `http://geda.seul.org/`

biests, students, educators, and professionals for many different design tasks. The suite runs best on Linux and OSX, although Windows ports of some apps have been made.

- Open Source: Kicad[19], wikipedia: Kicad[20] includes schematic capture and PCB layout
- Open Source: Free PCB[21] is a mature Windows only open source PCB drafting tool.
- IntelligentCad.org[22] has a few links to FPGA and PCB design tools (GPL)

5.5. µControllers

5.5.1. Programming Languages

There are many different programming languages available for µControllers:

- Assembly: Every µcontroller can be programmed in Assembly. However the differences between µcontrollers can be huge. Assembly gives you the most power of the µcontroller but this power comes with a price: Hard to learn and (almost)no code reuse.
  Assembly code is in essence translated machine code. It provides only the instruction set of the processor: add, subtract, *maybe* multiply, move data between registers and/or memory, conditional jumps. No loops, complex selection or build in I/O as in C/C++, Basic, Pascal, ...
  The disadvantage is that you have to implement everything yourself (lots of work even for the most simple programs).
  The advantage is that you have to implement everything yourself (programs can be written extremely efficient both in speed and size).
  This language is intended for advanced users and is usually only used as an optimisation for code in tight loops or for pushing the performance of a limited device to the edge of its abilities.
  Reasons to learn it:
  - Teaches you how the computer works on its lowest level.
  - Provides high speed code which consumes little memory.
    Reasons to avoid it:
  - Limited use.
  - Non-portable.
  - very hard to master.
    Freeware: AVR[23]
- C: C offers power but is much more portable than Assembly. For most µcontrollers there is a C compiler available. The differences between µcontrollers is smaller here, except for using hardware.
  Learning C is much easier than learning Assembly, still C isn't an easy language to learn from scratch. However these days there are very good books available on this subject.

19 https://en.wikibooks.org/wiki/Kicad
20 https://en.wikipedia.org/wiki/%20Kicad
21 http://www.freepcb.com/
22 http://IntelligentCad.org/
23 http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725

- Freeware:GCC Tools for AVR Studio Software[24]

- Basic: For many µcontrollers there are special flavours of Basic available. This is the easiest and fastest way to code µcontrollers, however you'll have to sacrifice some power. Still modern basic compilers can produce very impressive code.
  - Limited Freeware/payware:Bascom AVR[25] Very good Basic compiler for AVR. Limited to 4Kb programs. There is also a version available for the 8051 µcontrollers.
  - Limited Freeware/payware:XCSB[26] PIC Basic compiler. Lite version. No 32-bit integer and floating point support. (OS/2 WARP, Win95, Win98, Win2K, XP and Linux)
- Forth:
  - PFAVR[27] (GPL) Needs external RAM.
  - ByteForth[28] Dutch and works without external RAM, there is also a building book (Dutch only for now) available for Ushi our robotic project.

- Python
  - Pyastra[29]
  - PyMite[30]

further reading:

- Embedded Systems/Embedded Systems Introduction#Which_Programming_Languages_Will_This_B
- Embedded Systems/PIC Programming#Compilers.2C_Assemblers[32]

5.5.2. Programmers

After you've written your program, you need to get it into your µcontroller. If you use C or Basic you'll have to compile it. Then use a programmer to upload the code into the µcontroller. There are several different methods for this last step.

- External programmers: This is a device that's connected to a PC. You'll plug the µcontroller IC, EEPROM or other memory IC in its socket and let the PC upload the code. Afterwards you plug the IC in its circuit and test it. Can be time consuming when updating your program after debugging.
- ISP In System Programming: The board with the µcontroller has a special connector to connect to a PC. Hook up the cable, download code, test and repeat. More modern method. Only disadvantage: it consumes some boardspace. Not all µcontrollers support this.

24 `http://www.avrfreaks.net/index.php?module=FreaksTools&func=viewItem&item_type=tool&item_id=560`
25 `http://www.mcselec.com/bascom-avr.htm`
26 `http://www.xcprod.com/titan/XCSB/download.html`
27 `http://wiki.forthfreak.net/index.cgi?PFAVR`
28 `http://wiki.forthfreak.net/index.cgi?ByteForth`
29 `http://pyastra.sourceforge.net/`
30 `http://code.google.com/p/python-on-a-chip/`
31 `https://en.wikibooks.org/wiki/Embedded%20Systems%2FEmbedded%20Systems%20Introduction%23Which_Programming_Languages_Will_This_Book_Use%3F`
32 `https://en.wikibooks.org/wiki/Embedded%20Systems%2FPIC%20Programming%23Compilers.2C_Assemblers`

- Bootloader[33], also called ”self-programming”: The CPU accepts a new program through any available connection to a PC (no special connector needed), then programs itself. Not all µcontrollers support this. And you also need some other programming method, to get the initial bootloader programmed in (telling it exactly which connector to watch for a new program, the baud rate, etc.).

5.5.3. Debuggers

Modern µcontrollers have on-chip debug hardware called w:JTAG[34].

5.6. Various tools

See This site[35] for:

- The Motion Applet – Path modeling for the differential steering system of robot locomotion.
- The Encoder Designer – A design tool for encoder wheel patterns. (Wikipedia:Rotary encoder[36])
- RP1 – A mobile-robot simulator.
- Map Viewer – A Mapping Tool For Mobile Robotics.

And ”Experimental Robotics Framework” for rapid prototyping of robotics algorithms.[37]

5.7. Further reading

- CAD & Linux[38] has a long list of CAD tools that run under Linux, some of them GPL.
- Linux online: CAD/CAM[39] has a long list of CAD tools that run under Linux, some of them GPL.
- Practical Electronics/PCB Layout[40] has more information on using PCB design software.
- Urbiforge[41] has free downloadable links to Urbi and tutorials on how to use Urbiscript. Urbi is AGPL.

33 https://en.wikiversity.org/wiki/Embedded_System_Engineering%23bootloaders
34 https://en.wikibooks.org/wiki/%3Aw%3AJTAG
35 http://rossum.sourceforge.net/tools/
36 https://en.wikipedia.org/wiki/Rotary%20encoder
37 SourceForge: ”Experimental Robotics Framework” ^{http://sourceforge.net/projects/miarn/} .
38 http://www.tech-edv.co.at/lunix/CADlinks.html
39 http://www.linux.org/apps/all/Graphics/CAD/CAM.html
40 https://en.wikibooks.org/wiki/Practical%20Electronics%2FPCB%20Layout
41 http://www.urbiforge.org/index.php/Main/Tutorial

6. Tools and Equipment

You can't build a robot without at least a few tools. This page will cover some of the tools and equipment that'll be useful.

6.1. Mechanical Tools

For building your robot you'll need some tools to form the body.

1. Small vise: you'll need this.
2. Hammer: A hammer is one of the standard tools you'll need.
3. Screwdrivers & Wrenches: their uses are obvious. Two spanners of equal size are required for locknutting.
4. Saw: Metal and wood saws. Miter saws can be very handy, but are pretty expensive. A miter box might suffice for many purposes.
5. Square, measuring tape, scriber and other marking out tools.
6. Vernier calipers: Allow very accurate marking out and measurement. Also can be used to check thread pitch on machine screws without a dedicated pitch gauge.
7. Files: especially when working with metal, as rough metal edges are sharp.
8. Centre Punch: Essential for accurate drilling of holes in metal to prevent the drill skating over the surface.
9. Drill Press: (small table top versions suffice) is very handy for drilling accurate holes. Can also provide the low speeds for drilling large holes in metal, which hand drills cannot do easily.
10. Hobby Tool: Useful for many purposes.
11. Sharp utility knifes: Mostly used when working with plastics.
12. Hot glue guns: handy for quickly mounting parts. Not too strong bound, but useful for many applications.
13. Arc Welder: Only useful when working with thick steel on large projects (use a gas welding torch for thin metal;arc welders tend to burn holes right through the workpiece). Aluminium cannot be welded with ordinary welders. (Unless you have a MIG/MAG or TIG welder available)
14. Paint stripper/Electric Heat Gun: like a hairdryer on steroids. Useful for bending plastics, also applying heat-shrink tubing to electric cables at low power.
15. Safety Goggles: You only get one pair of eyes, and machine tools are potentially dangerous. Safety goggles are essential for using anything other than hand tools.

6.2. Electronic Tools

6.2.1. Soldering iron

The soldering iron is a very useful tool for assembling electronic circuits and connecting copper wires together.

For electronic circuits you'll need a light soldering iron (˜25W) with a small point (shaped like a pencil point). Especially SMD components require small points (or even better: special SMD soldering points).

Soldering electronic components is done with "soft soldering": with a low temperature (less than 300°C). Usually for electronics the melting point of the solder lays around 238°C. When buying solder choose for a solder wire (60% lead, 40% tin) with non-corrosive flux. (There is also "eutectic solder" - 63% lead, 37% tin, which transitions from liquid to solid immediately, with no plastic state in between.) Take the thinnest wire you can find (<=1mm).

See this page[1] for an in depth explanation of soldering electronic components.

For connecting metal wires you'll need something more powerful (30W-100W) like a soldering pistol, but an ordinary soldering iron would do just as well. Note: not all materials are as good to solder. Copper is easy to solder and has a reasonable strong bond. Aluminum has a weak bond.

For stronger connections it's better to braze instead of soft soldering. Brazing involves higher temperatures (typical between 450°C and 1000°C) and different flux ("Borax") and solder (copper and zinc or silver alloys) it also requires a welding torch instead of a soldering iron.

see this online book[2] for more in depth information on brazing.

If you need even more strength you could use welding. However welding is only used for heavy materials like steel alloys and these are in most cases too heavy to be used in robots (unless you're building a very big or industrial robot). Aluminum can be welded but it isn't as simple as welding steel alloys.

See this site for basic welding information[3].

1 `http://www.epemag.wimborne.co.uk/solderfaq.htm`
2 `http://www.handyharmancanada.com/TheBrazingBook/bbook.htm`
3 `http://www.aussieweld.com.au/arcwelding/index.htm`

6.3. Breadboard

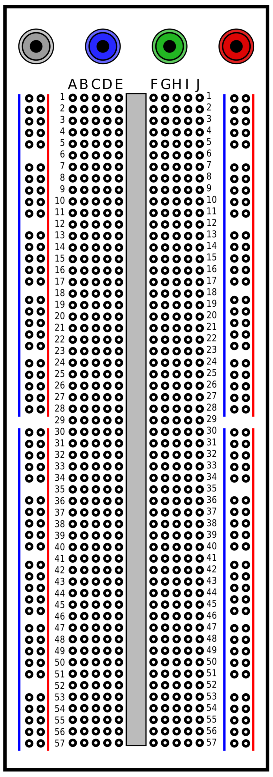


Figure 7

The boards allow you to build a temporary circuit in no time. Especially handy for testing new circuits. Connections are made with either ordinary thin stiff wire with the insulation removed at the ends or with special breadboard wires with stronger tips. Wires with crocodile clamps are needed for hooking up signal generator, oscilloscope, DMM, etc. Larger boards have connectors (typically banana plugs) for the power supply.

There are small breadboards with an adhesive strip at the bottom. These can be mounted on an empty part of a microcontroller board and can be used to build small circuits.

- **Note:** when you build a sensitive analog circuit on a breadboard, it can behave differently than when it's build as a PCB. This is because of parasitic components: the wires connecting the components on the board act as a combination of a resistor, capacitor and coil (all with very low values). Keep in mind that in some circumstances this can affect the working of a circuit. Usually this is only a problem when working with low amplitudes and/or high frequencies.

6.4. Electronic Equipment

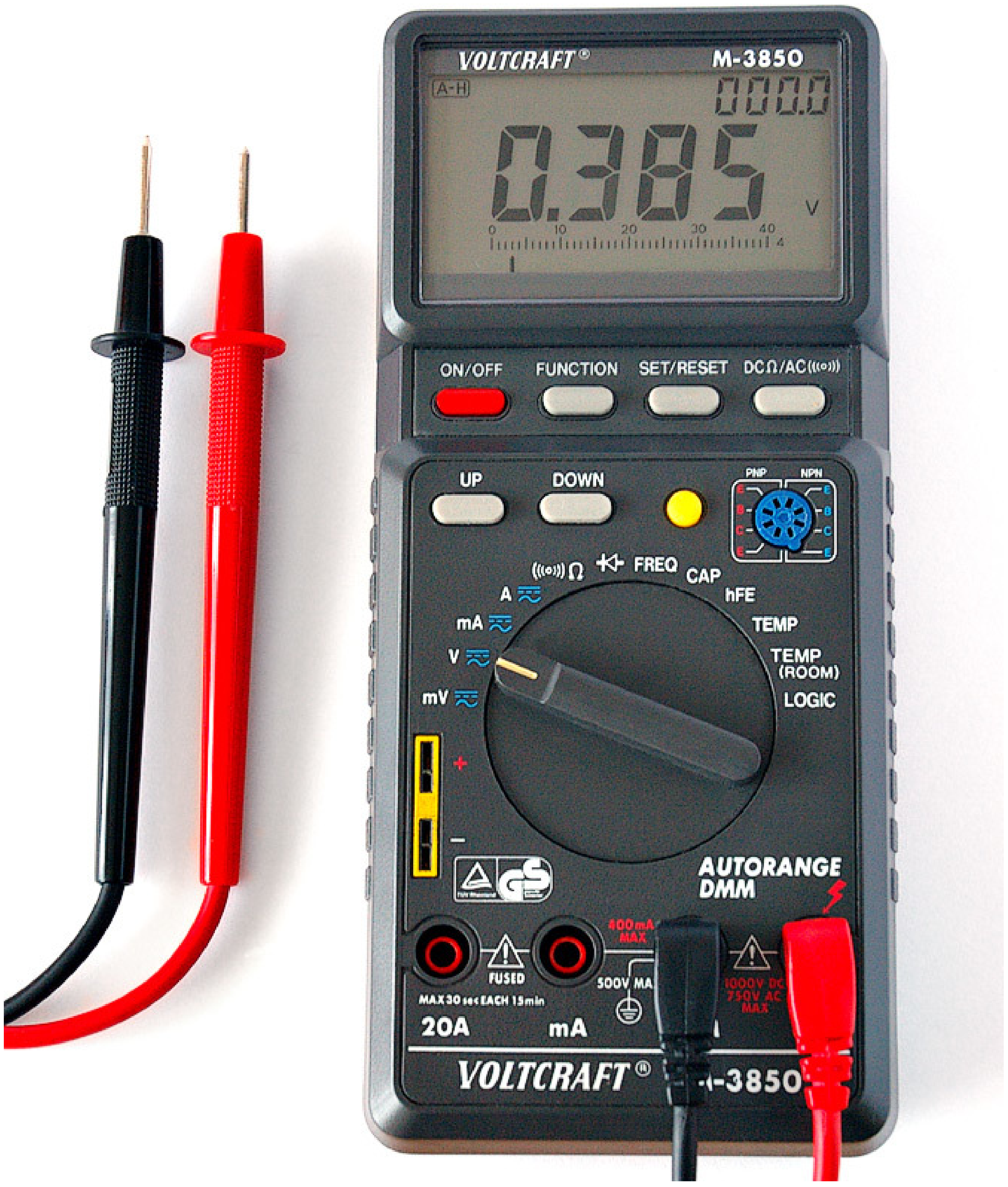


Figure 8 Digital Multimeter

- Multimeter: measures voltage, current & resistance. Many can measure transistor and diode characteristics, frequency and capacity. Some can measure temperature or light intensity.
  - Note: measuring voltage and current of a AC source isn't as simple as measuring DC levels. But since robots rarely use AC this would be out of the scope of this text. But if you would require to measure AC levels you should read up on this.

- Oscilloscope: makes a electric signal visible. Very useful when working with more complicated electronic circuits, especially analog signals and data communication. Oscilloscopes exist as stand-alone devices or as add-on modules for PCs. The latter provides extra abilities like spectrum analysing and recording of signals.
- Variable power supplies: power supplies with variable output. Either AC or DC. Either the output voltage or current can be regulated, although most power supplies let you set a max current.
- Signal generators: generates different shapes of signals (sine, square, saw and triangle), with variable frequency (1Hz up to 100MHz) and amplitude.
- Logic probe: pen-like devices that detect logic levels (either TTL or CMOS). Most can detect pulse signals. Very handy when working with digital electronic circuits.
- Frequency meters: measures the frequency of a signal. Can also be used as a pulse counter. Oscilloscopes can be used for measuring frequency, and storage scopes can freeze a waveform onscreen allowing pulses to be manually counted, but frequency meters are a good investment if this needs to be done very often.
- LEDs: An underrated test device for digital circuits. LEDs are far better than voltmeters for digital circuits in some situations, as you can see many input and output values concurrently, without connecting a multitude of voltmeters or constantly checking everything with a logic probe. In particular, they can instantly show the status of several logic signals simultaneously, impossible with a logic probe. Good breadboard building practice also includes an LED for each breadboard to show it is powered up correctly - this can help avoid the potentially frustrating situation of faultfinding a logic circuit that is actually sound, but has an intermittent or noisy power supply. It's also an excellent indicator if a component is short-circuiting at any time during operation, as the LED will likely dim or go out.

6.5. Connectors

6.5.1. Insulation Displacement Connectors (IDC)

Assembling parallel ribbon cables from ribbon and the IDC connectors:

Practical tips:

1. Note that IDC ribbon cable is usually not provided with multicoloured or 'rainbow' insulation, but with single-colour insulation — usually grey or white. However it also has a stripe of coloured ink or paint (red or black) down one side, to guide you with connector orientation. If you need to strip away some of the wires of a multi-way cable to suit the IDC

connectors you're using, remove them from the side furthest from the ink stripe so it's still present on the cable.

1. It's usual to fit IDC connectors to the cable so their pin 1 end is on the stripe side of the ribbon. This also

helps guide you when you're mating the cable connectors with those on the equipment, knowing that the stripe corresponds with pin 1.

1. Before clamping an IDC connector to a ribbon cable, make sure that the cable grooves are aligned with the contact jaw tips and that they are also aligned with the scallops moulded into the underside of the clamping strip.
2. Make sure too that the connector pin/jaw axis is as close as possible to 90° with respect to the ribbon cable wire axes. If the connector/ribbon angle is not close to 90°, some connections may not be made properly. If the connector is being fitted at the end of a ribbon cable, cutting the end of the ribbon cleanly square first will allow you to use it as a guide.
3. Try to squeeze the IDC connector and its clamping strip together as evenly as possible, so they remain as close as possible to parallel with each other during the operation. This too ensures that all joints are made correctly. The easiest way to squeeze them together evenly is by using a small machine vice or a special compound-action clamping tool.
4. If an IDC connector has a second cable clamping strip, don't attempt to fit this as part of the main assembly. Assemble the main parts of the connector first on the ribbon cable, and only then fit the second clamping strip.
5. When you are bending the ribbon cable around before fitting the second clamping strip, don't pull it hard. This may loosen some of the connections inside the IDC connector. Just bend the ribbon around gently — a small amount of slack won't do any harm, and may in fact protect the IDC connections from strain.

Properly-assembled IDC connector illustration:

`http://www1.electusdistribution.com.au/images_uploaded/IDCconnE.pdf`

Practical uses:

A common IDC cable in use is an IDC D9 socket to IDC 2 by 5 header socket. This cable is often used to connect a PC serial (RS232) port to a microcontroller development board. On the board there will be a 2 by 5 pin header.

6.5.2. RJ45 network connector

Figure 9 RJ45 Connector

These are the connectors used on UTP network cables. A smaller version (RJ11) is used for telephones. You need a crimping tool to attach the connector to the cable. These connectors are very useful for hooking up different PCB with each other. A good use for RJ45 connectors is for making serial (RS232) programming cables for small embedded systems (many credit card terminals use a DB9 to RJ45 cable to download software from a PC during development). If you are building small embedded controller boards an RJ45 can be a handy connector size to use.

6.6. further reading

- Practical Electronics/Breadboard[4]

4 https://en.wikibooks.org/wiki/Practical%20Electronics%2FBreadboard

7. Electronic Components

7.1. Basic Electronics

For basic electronics you may want to consult this wikibook[1] and this section[2] on Microcontrollers.

google[3] has some links on PCB fabrication.

7.2. Special Electronic Components

There are many electronic components that aren't described in most electronic textbooks, but are very useful in robotics. Most of these components are covered in other section in this book (e.g Sensors and Motors).

A typical robot needs a heat sink[4] on the power transistors connected to its motors and other actuators[5], but often does not need a heatsink connected to its CPU.

1 `https://en.wikibooks.org/wiki/Electronics`

2 `https://en.wikibooks.org/wiki/Robotics%3A%20Computer%20Control%3A%20The%20Interface%3A%20Microcontrollers`

3 `http://www.google.be/search?hl=nl&q=making+Pcb%27s+at+home&btnG=Google+zoeken&meta=`

4 `https://en.wikibooks.org/wiki/Power%20Electronics%23heatsink`

5 Chapter 13 on page 71

8. Mechanical Components

Robotics[1]: Design Basics: **Mechanical Components**

8.1. Gears

Gears are mechanical parts with cut teeth designed to mesh with teeth on another part so as to transmit or receive force and motion. The cut teeth are also sometimes called cogs. In Robotics Gears are used to transfer rotational forces between axles. They can change speed and direction. The axles can stand in any orientation, however not all orientation can be done with 2 gears. Commonly gears are used to reduce the speed of a motor. When they reduce the speed, the torque of the output axle increases.

Common types of gears as used in Robots are explored below. Each type of gear is used for different purposes and it has both advantages and disadvantages.

See this Wikipedia article[2] on gears. This page will cover how each gear is used specifically in robotics.

1 `https://en.wikibooks.org/wiki/Robotics%23Design%20Basics`
2 `https://en.wikipedia.org/wiki/Gear`

8.1.1. Spur gear

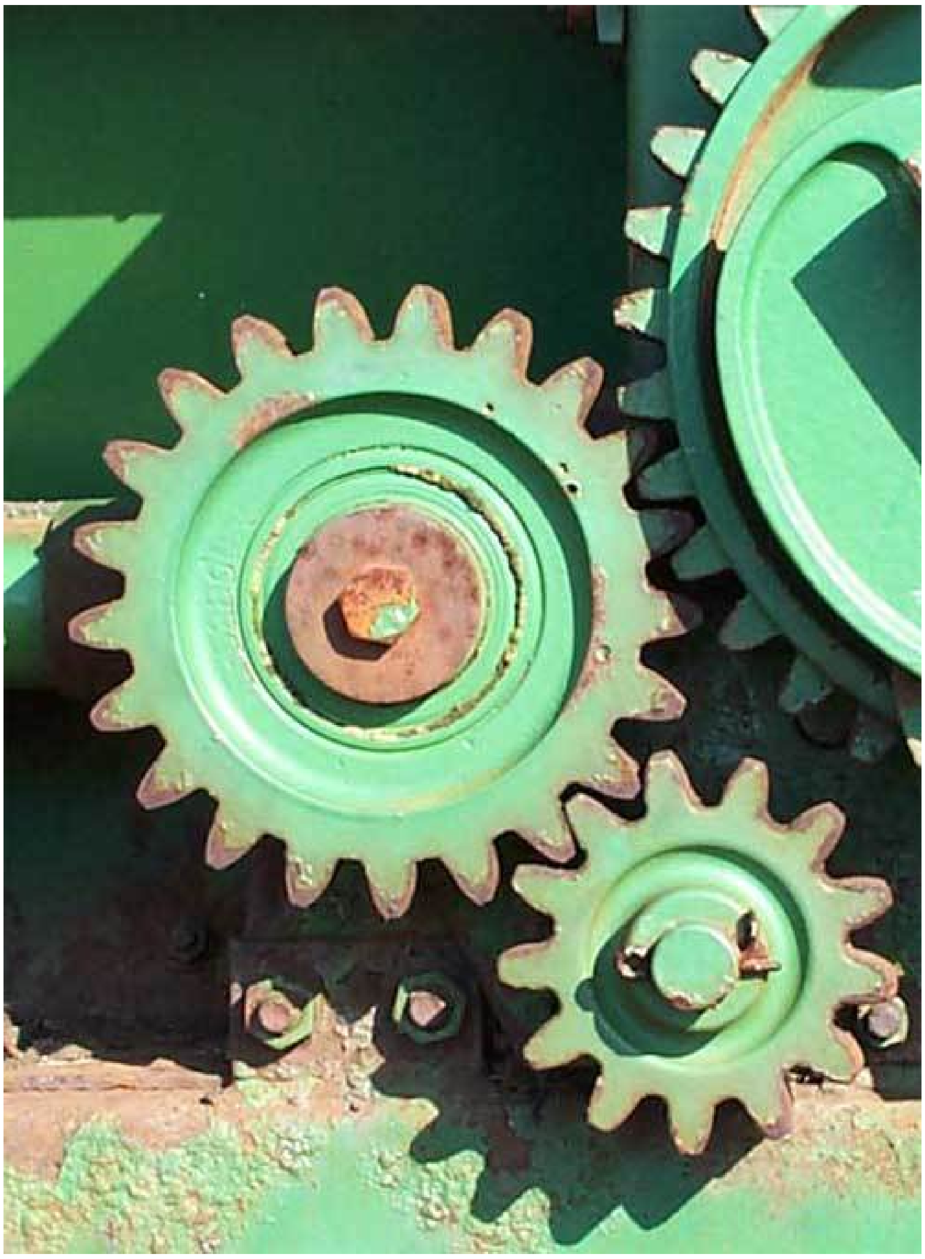


Figure 10 Spur gears found on a piece of farm equipment

Spur gears are the best known gears. These are the simplest form of gears and are commonly used in light machines as bikes, mixers, etc,...

They aren't used in cars as they are very noisy and their design puts a lot of stress on the teeth.

You might use them to transfer rotation from a motor output shaft (coming directly from the motor or from the gear box) to the axle on which the wheels are attached. This poses a limitation: the motor output axle and the wheel axle have to be parallel.

8.1.2. Bevelled gear

Bevelled gears are used when you wish to transfer work between two perpendicular shafts that are on the same plane (if the axles were to be extended they would hit). They can have straight, spiral or hypoid teeth.

8.1.3. Worm Gear

This is a gear that resembles a screw, with parallel helical teeth, and mates with a normal spur gear. The worm is in most cases the driving gear, there are however a few exceptions where the spur gear drives the worm. The worm gear can achieve a higher gear ratio than spur gears of a comparable size. Designed properly, a built in safety feature can be obtained: This gear style will self-lock if power is lost to the drive (worm) however this feature doesn't work if the pinion is powered.

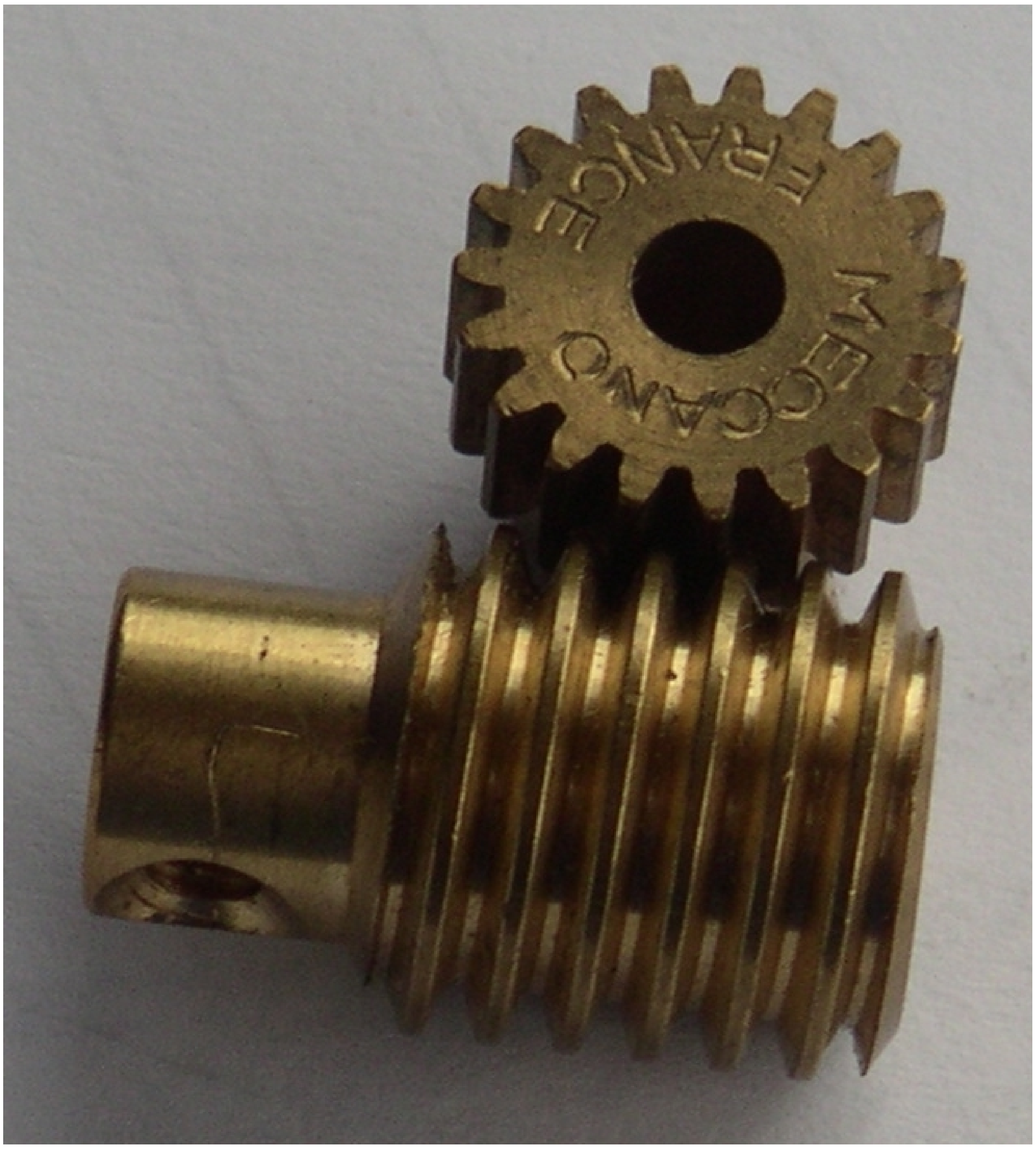


Figure 11 An Example of a Worm Gear and Pinion

8.1.4. Rack and Pinion

Torque can be converted to linear force by a rack and pinion. The pinion is a spur gear, and meshes with a toothed bar or rod that can be thought of as a sector gear with an infinitely large radius of curvature. Such a mechanism is used in automobiles to convert the rotation of the steering wheel into the left-to-right motion of the tie rod(s).

8.2. Internal gear

8.3. Chains & Belts

Chains and belts can be used to transfer rotational motion over a longer distance. The main difference between the two, lays in the fact that a belt can slip. This is not necessarily a bad thing as slipping occurs when the output shaft carries a load that is too heavy. In this case the belt slips instead of the motor being stalled. This protects the motor as it could otherwise be damaged by the stall current.

The disadvantage of belts is that since it can slip, the amount of rotation doesn't remain the same, e.g., the input axle may turn 4 times, while the output axle may turn only 3.8 times. This should be taken into account if the position of the output axle is important.

9. Building Materials

Robotics[1]: Design Basics: **Building Materials**

There is plenty of choice when it comes to picking the building materials for your robot. However not every material is a good choice.

There are three groups of materials. Each of these three groups have their own characteristics, possibilities and difficulties.

Note: There is a fourth group of materials called ceramics. However this group is only marginally useful for robotics.

Note In this book robot sizes are mentioned as small, medium and large. With small I mean any robot that can maneuver on a table. Medium means any robot that's too large to move on a table but still is light enough to lift on your own. Large mean anything bigger and heavier than that. Also robots that'll have to operate in real life (rough) condition should be counted as either medium or large.

9.1. Wood

Wood is probably the best material to start with. It's light, pretty strong and easy to work on. Not to mention cheap and easily available.

Even if you intend to use metal or plastic, wood can be handy for various purposes like prototyping and building jigs and other aids in working on metal or plastic parts.

The main reason that you will not see many wooden robots is because wood doesn't seem to fit in the picture of high tech machines. Funny thing is that (synthetic) composite materials have a very high coolness-factor (e.g Kevlar® Yes, Kevlar is a brand name. It's made of Aramide), yet wood, a natural composite material, hasn't.

- Useful for small or medium sized robots, prototyping and as construction aid.

9.2. Metal

There are 80 different pure metals each having different properties. However in the world of Robotics there are only a select few from the 80 that are useful. This list is increased

1 `https://en.wikibooks.org/wiki/Robotics%23Design%20Basics`

by Alloying. Alloying is the process of combining either in solution or compound, two or more elements, at least one of which is a metal, and where the resulting material has metallic properties. The resulting metallic substance has different properties (sometimes significantly different) from those of its components.The properties of some metals and their alloys are below.

Some alloys are limited in supply in the market due to the limited demand for them. In order to obtain these materials it is often required to look further than the general consumer market.

9.2.1. Aluminum

Aluminium (or aluminum, both are correct) is commonly available in extruded forms in different shapes. It's pretty cheap, light, strong, resistant to corrosion and easy to work with. However welding aluminum isn't practical as it needs special welding equipment (MIG/MAG or TIG welding) and the bond isn't very strong. While soldering is possible, it doesn't make a strong bond. Rather use nuts and bolts or rivets.

- Useful for small or medium sized robots.
- Useful for non-load bearing parts in large robots.
- Not very good for bearings.

There is an alloy of aluminium called *Duraluminium* it is almost as strong as soft steel but very light thus making it a natural choice for the construction of Aircraft. However as a trade off for the combination of being strong and light it is quite expensive (if you can find it at all). The best bet is to search aircraft wrecking yards.

9.2.2. Steel

Commonly available steel is an alloy of iron. It's stronger than aluminum, but it's also heavier and harder to work with - in particular, it blunts tools more easily and swarf is always razor sharp. Welding isn't much of a problem. However heating steel (at welding temperatures) changes its characteristics (strength, hardness, rust resistance). Note that drilling steel requires cooling and a slow drill speed (both rotational and feeding). If you drill too fast, you'll see your drill heating up red hot. Drills that have been red hot lose their hardness and will be dull in no time.

- Useful for large robots and robots intended to operate in rough conditions. Too heavy for small or medium sized robots.

9.2.3. Bronze

Heavy. Very good for bearings. Too expensive and heavy for most other purposes.

9.2.4. Brass

Heavier and more expensive than aluminum. Can be soldered.

- Useful for bearings.

9.2.5. Copper

Mostly available as wire or axles. Quite heavy, very good conductor.

- Useful for special parts and wires.

9.3. Synthetic Materials

Like steel, synthetic materials are a name for a very large group of materials. There are hundreds of different plastics each with different characteristics and uses. We'll be covering only a few of them, but many techniques work just as well with other synthetics.

Most synthetic materials can be bent into shape after they are heated. A hot air gun (used to remove paint) can be sufficient for this purpose. Drilling and sawing these materials requires low speeds or they have to be cooled with water so the material doesn't melt. Soft plastics can be cut with an utility knife.

9.3.1. PVC

Polyvinyl chloride: Used for plastic tubes.

9.3.2. Plexiglass

Polymethyl methacrylate: Transparent material. Can be bent when heated to 200°C.

9.4. Composite materials

Polymer composite materials are materials consisting of a polymer matrix and a reinforcing material. (think of reinforced concrete: the polymer matrix is the steel grid and the reinforcing material is the concrete)

These materials are considerably stronger and stiffer than steel and aluminium alloys.

Material	Specific strength	specific stiffness
Steel	150*10^3 Nm/kg	20*10^6 Nm/kg
E-fiberglass/epoxy	300*10^3 Nm/kg	10*10^6 Nm/kg
Aramide/epoxy	500*10^3 Nm/kg	25*10^6 Nm/kg

The reason why composite materials haven't replaced steel is because of cost. Composites are only used where weight is a more important factor than price, e.g., airplanes: less weight means less fuel consumption and/or more useful payload.

9.5. Other Materials

9.5.1. Foamcore

Foamcore is relatively weak, but it is very practical for making fast prototypes. (`http://zero-waste.blogspot.com/2011/12/this-is-foamcore.html` `http://web.mit.edu/imoyer/www/portfolio/foamcore/`).

9.5.2. Cardboard

Generally, it's weak and looks ugly, but it's very practical for making fast prototypes. Can be cut with a knife or scissors and put together with duct tape or glue gun. When dry, it's an insulator and can be used as a prototyping board for circuits.

With careful design, corrugated cardboard is strong enough to hold up full-sized humans (a[2] b[3] c[4]).[5][6][7][8][9][10]

Often a cardboard mockup is built to make sure the parts of the robot all fit together properly.

Sometimes cardboard is more than strong enough for many parts of a working robot (a[11] b[12]).

2 `https://web.archive.org/web/20111213114135/http://www.arcspace.com/gehry_new/cardb/cardb.html`

3 `http://www.erb.co.il/en/cooperations.asp`

4 `http://www.dezeen.com/2012/11/12/cardboard-bicycle-by-izhar-gafni/`

5 Nick Michelin. 'The cardboard version of le Corbusier's "le Gran Confort" club chair.' etc. "Cardboard Furniture" ^{`http://www.inspirationgreen.com/cardboard-inspiration.html`} .

6 les cartonnistes associés ^{`http://www.lescartonnistesassocies.com/`} . Courses, workshops, training, creation and sale of cardboard furniture.

7 Lazerian. "cardboard furniture range" ^{`http://www.lazerian.co.uk/projects/cardboard-furniture-range.php`} .

8 pacalowski. "cardboard furniture" ^{`http://www.pacalowski.com/cardboard_furniture.html`} .

9 Frank Gehry. "Easy Edges" and "Experimental Edges" series of cardboard furniture. "designer profile: Gehry Chair collection" ^{`http://www.dedece.com/docs/userManaged/newsletter_files/21/18/dedece06_gehry.pdf`} .

10 "Wiggle Side Chair. Frank Gehry" ^{`http://www.design-museum.de/en/collection/100-masterpieces/detailseiten/wiggle-side-chair-frank-o-gehry.html`} . quote: "Gehry named this material Edge Board: it consisted of glued layers of corrugated cardboard running in alternating directions, and in 1972 he introduced a series of cardboard furniture under the name "Easy Edges." The "Easy Edges" were extraordinarily sturdy..."

11 `http://www.kickstarter.com/projects/392540503/cardboard-robot-robotic-arm-and-smart-phone-camera`

12 `http://thecardboardrobot.com/`

10. Basic Programming

Sensors

10.1. Programmable devices

10.1.1. Microcontrollers

These days using programmable components is easy. In the early days you had to write code in assembly, burn it in an EPROM, plug the EPROM in its socket and hope you didn't make any mistake. Because if you did, you had to hunt down the error in hundreds of lines of cryptic assembly code and had to use a new EPROM. These days you've got flash memory which can be reprogrammed in circuit. You've got C and Basic compilers to write your code. Most µcontrollers have a whole lot of hardware on chip (UART, Watchdog, Real Time Clock, RAM, ROM, EEPROM,...) and libraries are available for different programming languages which make coding LC-displays or Servos very easy. Not to mention you've got emulators (A special connector plugged into the µcontroller socket which allows a PC to pretend to be the µcontroller) and simulators (simulates the µcontroller on a PC and allows to run its code) to hunt down the bugs.

AVR

A good µcontroller to start with is the AVR ATtiny2313[1] (previously: AT90S2313) or ATMEGA8535[2] (previously: AT90S8535) (Atmel). It's cheap, has all the essentials on chip and has a whole lot of websites dedicated to it. The 8535 is more expensive, but has more memory and has an on board ADC. An AVR combined with Bascom-AVR[3] is pretty much the easiest way to start with µcontrollers.

See Atmel AVR[4] for details.

See Robotics/Computer Control/The Interface/Microcontrollers[5] for other popular microcontrollers.

1 http://www.atmel.com/dyn/products/product_card.asp?part_id=3229
2 http://www.atmel.com/dyn/products/product_card.asp?part_id=2008
3 http://www.mcselec.com/bascom-avr.htm
4 https://en.wikibooks.org/wiki/Embedded%20Systems%2FAtmel%20AVR
5 Chapter 18.3 on page 119

10.1.2. PC

The PC has many advantages over µcontrollers when it comes to controlling a robot. However it has 2 flaws that make it nearly impossible to use in small robot-projects:

- First: it's large and heavy.
- Secondly: A modern pc consumes enormous amounts of power.

This means that PCs are limited to tethered or large robots. If power isn't a problem the PC offers enormous amounts of RAM and HD space and plenty of CPU power. Plus a very handy user interface. Also network capabilities can be very useful for some applications. Providing your robot with a wireless connection (IR, radio, WiFi, bluetooth,...) and a similar interface on the PC reduces the wire-problem to a limited-distance (or in the case of IR a line-of-sight) problem.}}}}

If you intend to use a PC in your project, you might want to use one of the flavors of linux as the OS, as it allows easy access to any of the parts of the computer. Another choice for an OS is the older MS-DOS, although MS-DOS lacks multithreading (but those can be added in your software) and has ugly memory management (remember the 640Kb limit?).

10.1.3. Laptop

The laptop has only its size against it. A laptop provides the benefits of a PC without the enormous power consumption. The downside to laptops is their limited battery life.

10.1.4. PDA

The **PDA** (**P** ersonal **D** igital **A** ssistant) is nothing more than a hand held **PC** (**P** ersonal **C** omputer). Most of the time these units used scaled down **RISC** (**R** educed **I** nstruction **S** et **C** hip) processors to allow for fast execution times and also allow the unit to remain smaller and lighter than your average laptop. Some PDA's even have built-in GPS which makes it very interesting for outdoor robot applications. The downside is their price. The PDA is useful for play, but also serves as a great business tool especially when they are equipped with a stylus.

10.1.5. Gameboy

The Gameboy (the classic, advance, color and DS) can be a very powerful device. These have been used in many projects (see google). The only drawback is constructing the special PCB to act as a connector.

10.1.6. Gameboy Advance (GBA)

The Gameboy Advance is also excellent as a robotics platform. There are a few complete open-source development suites available (GCC based C/C++ plus ARM assember). There is also a good kit (Charmed Labs) available to interface Lego robotics motors/sensors and give much more control compared to the Lego RCX. With 4MB of FLASH and a few hundred K of RAM the kit has huge potential. The kit is not really for the complete beginner though as setting up the GNU tools can be quite complicated although the development environment is consequently very rich.

10.1.7. Programmable Logic Controllers

Programmable logic controllers, PLCs, are special purpose industrial computers, made to be easy to interface with electrical circuits. Most PLC are expandable but there are some compact "all-in-one" solutions as well. They have excellent bit-manipulating possibilities, and add-on-cards for several special signals as high-speed counters, analog I/O, networks and fieldbuses (RS-232, RS422, RS-485, Ethernet, DeviceNet, Profibus...), pulse output, servo control, etc. Increasingly, special inputs are included in the PLC CPU unit. PLCs are traditionally used to control automated factories, possibly several machines per PLC. They normally use special-purpose programming languages, most commonly ladder or SFC.

PLCs are available in sizes from approximately 5x5x5 cm (there is a PLC IC available as well, but for the purposes of this book we choose to ignore that fact). The smallest ones are very limited though, with only a few I/Os and very limited memory, and also very limited instruction set with regards to data processing and calculations. PLCs are especially suitable if you have a lot of experience of electric systems, but feel intimidated by designing and soldering PCBs and programming computers (i.e., you are an electrician). They also shine in very large system with many special purpose sensors and outputs, as mostly there will be a standard solution from the manufacturer that is possible to use.

The downside with PLCs are their price - They can be quite costly, normally at least an order of magnitude higher than microcontrollers. They also draw more current than a microcontroller, however less than a computer. On the other hand, compared to a computer, they are computationally weak. They aren't suited for heavy signal-processing work.

PLC:s are best suited for large robots, but could be usable in medium-sized projects as well.

10.1.8. Combinations

It's common to use both a computer/laptop/PDA and (a) microcontroller(s) as those 2 complement each others limitations. e.g. The first has the advantage of large amounts of memory and processing power, however lacks specific outputs that are very handy in robotics, like PWM. Microcontrollers commonly have PWM channels that operate independently of the rest of the microcontroller, but are limited in their processing speed and memory.
Linking those 2 devices provides the best of both worlds.

10.2. Programming languages

This[6] page discusses some of the available programming languages and provides links to download compilers for those languages.

10.2.1. Which language to pick?

The choice of language depends on a few points:

- previous experience. If you're already comfortable with a particular programming language, you'll probably want to use that language for programming your robot.
- How much time and effort you intend to invest. Not every language is equally hard/easy to use. Most of the complexity of a language comes from allowing more low-level access. Basically the more control a language gives to the user, the harder it is to use, but also the more powerful it is. In theory Assembly would allow a programmer to write the fastest and smallest code. However this is only true for an experienced programmer. Modern compilers can generate code that comes near to hand written Assembly. In most cases, using Assembly needlessly complicates a program, with the exception of using inline assembly in a C or Basic program. Usually these are a few lines of code which work directly with memory, or have to execute in a known time. For many if not most projects, a language such as Basic or C suffices. Only if you intend to push a microcontroller or PC to the limits of its capabilities is it worth the trouble of writing Assembly code.
- your goals. If you want a simple robot, you wouldn't need Assembly code at all. With modern microcontroller you can easily write sloppy Basic code and still have sufficient speed and memory. If you intend to build a cutting edge robot with a few dozen sensors, image recognition and speech recognition you'll need to write perfect code and will need Assembly for some of the components in order to have enough speed and be able to fit the code in the limited memory of the microcontroller(s). Most projects will fall in between these two extremes and good written Basic or C code would be more than sufficient. Knowing (some) Assembly can be useful as it'll give some insight on how processors work. Such insights make you a better programmer.
- Availability. Not all languages are available for all microcontrollers. Urbi is open source, C and Basic and Forth are common, often as freeware, other languages might be available commercially, or not at all.

10.3. Writing your software

This section doesn't intend to teach the basics of programming, just those points that aren't addressed in most programming tutorials and beginners books.

- Style. "Style" is how you indent your lines, how you pick your variable and function names. Pick one and stick with it throughout your program. Especially for functions and

6 https://en.wikibooks.org/wiki/Robotics%3A%20Design%20Basics%3A%20Design%20software%23Programming%20Languages

variables you should decide when to use capital letters, underscores and when to stick words together. Doing this the same everywhere makes it easier not to miss-spell names.
- Use informative names. Functions, procedures and variables should have informative names. Their purpose should be clear.
- Plan before coding. In software development there are two important steps before coding: requirement analysis and software architecture. The first is about finding out what the program is expected to do, what inputs to expect, which output it should generate and what the limitations of its environment are going to be. The latter involves working out how the program is going to be structured, which data types to use, which algorithms, how the input is going to enter the program and how to make sure it's valid and how to format the output. Using a lite version of these steps will make it much easier to write a decent program. If you intend to build a more complicated robot, you'll going to need to invest more time in planning out your software. Have all your software requirements written out first. Then pick a few of the most critical requirements and refine them. Implement those first. After you're done with those (this includes throughfully testing them), pick a new set and refine and implement those. By doing it in small steps you avoid having to write out everything at the start (and making many assumptions) and get to use the insights you get by coding parts of it.
- Use PDL (Program Development Language) PDL is a method to write functions. It involves writing down the individual steps of a function in plain English without referring to language-specific things, then refining these steps into smaller steps, until it's easier to write the code than to split the steps further. Afterward leave the PDL lines in as comments.
- Simulation and Debugging Learn to use the simulator. It's easier to find errors there than in the hardware. Also learn to use the debugger. It's your best friend for finding errors.
- If your software runs on a PC a log-file can be useful. But be picky with what you let the software log. If you intend to run it for a long time those log-files can become massive. If you want to log sensor data over time, know that these can be imported in Excel by saving the data with spaces between them to separate columns and newlines to separate rows. *See w:Comma-separated values*[7].

10.4. Further reading

- Khepera III Toolbox[8]: a wikibook with detailed information on how to program one particular robot
- Urbiforge[9]: a website with tutorials on programming in Urbiscript

7 https://en.wikipedia.org/wiki/Comma-separated%20values
8 https://en.wikibooks.org/wiki/Khepera%20III%20Toolbox
9 http://www.urbiforge.org/index.php/Main/Tutorial

Part II.

Physical Construction

10.5. The Platform

Most robots need a framework of some sort to which the builder attaches the various components and subsystems that make the robot function. This framework is known as the **platform** .

A good platform has a few requirements:

- It has to be light. Don't use steel unless you really need the extra strength. Plexiglass (or any other hard plastic) with an aluminum framework for strength can be sufficiently strong for most purposes. Don't forget about wood. Wood is strong, flexible and light.
- It has to be easy to add new parts to it. By using one size of PCB you can drill holes ahead of time. And you'll be able to stack PCBs. If you're familiar with Meccano sets you know the advantage of using one standard hole size and spacing. Picking one for your robot can make adding parts very easy and fast, of course this is less efficient with space.
- It has to be easy to remove parts of it: Make sure you can access any of its parts without having to take most of the robot apart. Especially batteries should be accessible.
- It has to be in balance. The weight of the robot should be mostly within the figure formed by connecting the wheels of the robot (A triangle for 3 wheeled robots, a rectangle for 4 wheeled robots). Placing significant weight outside this figure makes it more easy for the robot to tip over. The center of gravity should also be as low as possible, otherwise the robot could tip over when turning too fast.
- It has to have a size that's practical: Don't make it too small or you end up with insufficient space for all the features you had in mind. Don't make it too large either or you end up with a robot that can't maneuver without bumping in furniture and people. Or worse: can't get through a door post.
- Avoid putting the wheels directly on the motors' axles. Use an axle and connect the motor to it with gears. For small robots this isn't important, but larger would displace the axle inside the motor and damage it.

11. Construction Techniques

11.1. Working with Metals

11.1.1. Measuring and Marking

- Do: Measure twice, cut once. Putting it back on is MUCH more difficult than taking it off.
- Do Not: Measure with a micrometer, mark with chalk, cut with a torch. Accuracy is important in moving parts.

11.1.2. Cutting and Sawing

- Sawing sheet metal requires a saw with a fine-tooth blade. As a general rule of thumb at least three teeth should touch the material being cut.
- The teeth should be aimed away from the handle.
- Only apply pressure when pushing the saw away from you, not when pulling back.
- A miter box can be very handy to make straight or 45° cuts. Use a metal box as wooden boxes wear out very fast.

11.1.3. Drilling

- Use a punch to make a small indentation where you want to drill the hole.
- Use a slower speed to drill the hole than you would use for wood.
- Use clamps when working with sheet metal. Don't hold it with your hands as metal is very sharp when cut.
- Large holes can be made by drilling a hole and using a file thread or small metal saw to cut the hole. Finish with a half round file.
- A groove can be cut by drilling two holes at the outer edges and using file thread or small saw to connect both holes.

11.1.4. Filing

- Files exist in different shapes and different roughness.
- Filing straight: Hold the file diagonal on the surface, apply pressure and move the file upwards while moving from left to right.

11.1.5. Bending

Place the sheet metal between 2 wooden boards in a vice. Let the line where you want the material to bend match up with the border of the boards. Use a hammer to bend the metal.

11.2. Working with Plastics

11.2.1. Cutting

Thin plastics can be cut with a sharp utility knife, thicker material can be sawed. When sawing plastic, care must be taken to not melt it.

11.2.2. Drilling

Drilling plastic requires a slow speed drill. Special drills for plastics are much better as metal drill can cause the material to crack.

Large holes and grooves can be made the same way as in metal.

11.2.3. Filing

11.2.4. Bending

Use a heat gun (paint stripper) to heat up the plastic, bend it and let it cool down. Bending can be done by hand.

11.3. Further reading

- ”the #1 rule of machining: Always properly secure the workpiece.” -- David Cook`http://www.robotroom.com/Monkey-Mints-Line-Following-Robot3.html`

11.4. Resourcefulness

Though many components for a serious robotics project will almost certainly have to be purchased, the total cost of the project can be reduced by harvesting parts from various electronic and electro-mechanical devices. The general rule for selection is ”the older, the better,” as newer devices are now dominated by specialized ICs and surface mount[1] components that builders might find difficult to use in their robots. SMD (Surface-mounted

1 `https://en.wikipedia.org/wiki/Surface-mount_technology`

devices) can be used to create very small robots, if one has the tools and patience to work with them.

An example of a good source of parts would be a printer from the 1980s. Without even disassembling such a printer, one can see all sorts of potentially useful components. Additionally, several metal rods and plates contained in the printer may be used as structural parts for the robot, and even the plastic case itself could be used.

Old floppy disk drives and copy machines are also good sources for parts such as stepper motors, optocouplers or microswitches, regular components, machined metal rods and hardware. Hard disk drives have fewer usable parts but can still be a source.

However stepper motors from printers and copy machines tend to consume a lot of power and may not be as good for battery operated robots.

Note to potential contributors: perhaps there could be a "case studies" section here, with examples of what can be obtained from a variety of devices.

Part III.

Components

12. Power Sources

Though perhaps other power sources can be used, the main sources of electrical power for robots are batteries and photovoltaic cells. These can be used separately or together (for practical applications, most solar-powered robots will need a battery backup).

12.1. Photo Voltaic Cells

Photo Voltaic Cells or solar cells are well known for their use as power sources for satelites, enviromentalist green energy campaigns and pocket calculators. In robotics solar cells are used mainly in BEAM[1] robots. Commonly these consist of a solar cell which charges a capacitor and a small circuit which allows the capacitor to be charged up to a set voltage level and then be discharged through the motor(s) making it move.

For a larger robot solar cells can be used to charge its batteries. Such robots have to be designed around energy efficiency as they have little energy to spare.

12.2. Batteries

Batteries are an essential component of the majority of robot designs. Many types of batteries can be used. Batteries can be grouped by whether or not they are rechargeable.

Batteries that are not rechargeable usually deliver more power for their size, and are thus desirable for certain applications. Various types of alkaline and lithium batteries can be used. Alkaline batteries are much cheaper and sufficient for most uses, but lithium batteries offer better performance and a longer shelf life.

Common rechargeable batteries include lead acid, nickel-cadmium (NiCd)and the newer nickel metal-hydride (Ni-MH). NiCd & Ni-MH batteries come in common sizes such as AA, but deliver a smaller voltage than alkaline batteries (1.2V instead of 1.5V). They also can be found in battery packs with specialized power connectors. These are commonly called race packs and are used in the more expensive RC race cars. They will last for some time if used properly. Ni-MH batteries are currently more expensive than NiCd, but are less affected by memory effect.

Lead acid batteries are relatively cheap and carry quite a lot of power, although they are quite heavy and can be damaged when they are discharged below a certain voltage. These batteries are commonly used as backup power supply in alarm systems and UPS.

1 Chapter 30.4 on page 181

An extremely common problem in robots is the ”the microcontroller resets when I turn the motor on” problemhttp://www.embedded.com/underthehood/200001798. When the motor turns on, it briefly pulls the battery voltage low enough to reset the microcontroller. The simplest solutionhttp://www.jonh.net/~jonh//robots/skguide.html http://books.google.com/books?id=WrVOqUFGP9sC&pg=PA284&lpg=PA284&dq=robot+reset+battery+motor&source=web&ots=uUNRRJssHI&sig=5uWmfb08g4L_WhOhJVsi1j-UD_w http://www.cs.cmu.edu/~mmcnaugh/kdc/as7/ http://www.bluebelldesign.com/Power1.htm is to run the microcontroller on a separate set of batteries.

HISTORY OF THE BATTERY:

The first evidence of batteries comes from discoveries in Sumerian ruins dating around 250 B.C.E. Archaeological digs in Baghdad, Iraq http://www.windsun.com/Batteries/Battery_FAQ.htm. But the man most credited for the creation of the battery was named Alessandro Volta, who created his battery in the year 1800 C.E. called the voltaic pile. The voltaic pile was constructed from discs of zinc and copper with pieces of cardboard soaked in saltwater between the metal discs. The unit of electric force, the volt, was named to honor Alessandro Volta http://inventors.about.com/library/inventors/bl_Alessandro_Volta.htm. A time line of breakthroughs and developments of the battery can be seen here http://inventors.about.com/library/inventors/blbattery.htm.

HOW A BATTERY WORKS:

Most batteries have two terminals on the exterior, one end is a positive end marked “+” and the other end is the negative marked “-”. Once a load, any electronic device, a flashlight, a clock, etc., is connected to the battery the circuit being completed, electrons begin flowing from the negative to positive end, producing a current. Electrons will keep flowing as fast as possible until the chemical reaction on the interior of the battery lasts. Inside the battery there is a chemical reaction going on producing the electrons to flow, the speed of production depends on the battery’s internal resistance. Electrons travel from the negative to positive end fueling the chemical reaction, if the battery isn’t connected then there is no chemical reaction taking place. That is why a battery (except Lithium batteries) can sit on the shelves for a year and there will still be most of the capacity to use. Once the battery is connected from positive to negative pole, the reaction starts, that explains the reason why people have gotten a burn when a 9-volt battery in their pocket touches a coin or something else metallic to connect the two ends, shorting the battery making electrons flow without any resistance, making it very, very hot. http://www.howstuffworks.com/battery.htm

MAIN CONCERNS CHOOSING A BATTERY:

- Geometry of the batteries. The shape of the batteries can be an important characteristic according to the form of the robots.

- Durability. Primary(disposable) or secondary (rechargeable)

- Capacity. The capacity of the battery pack in milliamperes-hour is important. It determines how long the robot will run until a new charge is needed.

- Initial cost. This is an important parameter, but a higher initial cost can be offset by a longer expected life.

- Environmental factors. Used batteries have to be disposed of and some of them contain toxic materials. `http://robocup.mi.fu-berlin.de/buch/chap6/ComparisonBattery.html`

PRIMARY (DISPOSABLE) BATTERY TYPES

• Zinc-carbon battery - mid cost - used in light drain applications

• Zinc-chloride battery - similar to zinc carbon but slightly longer life

• Alkaline battery - alkaline/manganese ”long life” batteries widely used in both light drain and heavy drain applications

• Silver-oxide battery - commonly used in hearing aids

• Lithium Iron Disulphide battery - commonly used in digital cameras. Sometimes used in watches and computer clocks. Very long life (up to ten years in wristwatches) and capable of delivering high currents but expensive. Will operate in sub-zero temperatures.

• Lithium-Thionyl Chloride battery - used in industrial applications, including computers and electric meters. Other applications include providing power for wireless gas and water meters. The cells are rated at 3.6 Volts and come in 1/2AA, AA, 2/3A, A, C, D & DD sizes. They are relatively expensive, but have a proven ten year shelf life.

• Mercury battery - formerly used in digital watches, radio communications, and portable electronic instruments, manufactured only for specialist applications due to toxicity `http://www.medicbatteries.com/battery-dry-cell-dry-cells.aspx`

Helpful link comparing the most popular types of batteries in many different types of categories `http://www.allaboutbatteries.com/Energy-tables.html` `http://www.allaboutbatteries.com/Battery-Energy.html`

SECONDARY (RECHARGEABLE):

(Will be discussing the two most popular secondary batteries)

Lithium-ion Batteries:

Advantages:

These batteries are much lighter than non-lithium batteries of the same size. Made of Lithium (obviously) and Carbon. The element Lithium is highly reactive meaning a lot of energy can be stored there. A typical lithium-ion battery can store 150 watt-hours of electricity in 1 kilogram of battery. A NiMH (nickel-metal hydride) battery pack can store perhaps 100 watt-hours per kilogram, although 60 to 70 watt-hours might be more typical. A lead-acid battery can store only 25 watt-hours per kilogram. Using lead-acid technology, it takes 6 kilograms to store the same amount of energy that a 1 kilogram lithium-ion battery can handle. Huge difference!

Disadvantages:

Begin degrading once they are created, lasting only two or three years tops, used or not. Extremely sensitive to high temperatures, heat degrades battery even faster. If a lithium battery is completely discharged, it is ruined and a new one will be needed. Because of size and ability to discharge and recharge hundreds of times it is one of the most expensive rechargeable batteries. And a SMALL chance they could burst into flames (internal short, separator sheet inside battery keeping the positive and negative ends apart gets punctured). http://electronics.howstuffworks.com/lithium-ion-battery.htm

Alkaline Batteries:

The anode, the positive end, is made of zinc powder because the granules have a high surface area, increasing the rate of reaction and higher electron flows. It also helps limit the rate of corrosion. Manganese dioxide is use on the cathode, or the negative side, in powder form as well. And potassium hydroxide is the electrolyte in an alkaline battery. There is a separator inside the battery to separate the electrolyte between the positive and negative electrodes. http://www.wisegeek.com/what-are-the-key-components-of-an-alkaline-battery.htm

12.3. Fuel Cells

Fuel cells are a possible future replacement for chemical cells (batteries). They generate electricity by recombining hydrogen gas and oxygen. (commercial fuel cells will probably use methanol or other simple alcohols instead of hydrogen). Currently these are very expensive, but this might change in the near future when these cells are more commonly used as a replacement for laptop batteries.

Note: since fuel cells use flammable products you should be extra careful when you build a power source with these. Hydrogen has no odor like natural gas and is flammable and in some conditions explosive.

Pressurized canisters have their own set of risks. Make sure you really know how to handle these. Or at least allow other people enough time to get behind something thick and heavy before experimenting with these.

12.4. Mechanical

Another way to store energy in a robot is mechanical means. Best known method is the wind-up spring, commonly used in toys, radios[2] or clocks[3].

Another example of mechanical energy storage is the flywheel. A heavy wheel used to store kinetic energy.

12.5. Air Pressure

Some robots use pneumatic cylinders to move their body. These robots can use either a bottle of pressurized air or have a compressor on board. Only the first one is a power source. The latter power source is the batteries powering the compressor. Pneumatic cylinders can deliver very large forces and can be a very good choice for larger walkers or grippers.

2 http://www.leeselect.com/baygen/baygenamfmsw.htm
3 http://home.howstuffworks.com/inside-clock.htm

Note: Pressurized canisters and pneumatic components can be dangerous when they are handled wrongly. Failing pressurized components can shoot metal pieces around. Although these aren't *necessarily* life threatening, they can cause serious injuries even at low pressures.

Canisters on their own pose additional risks: Air escaping from a pressurized canister can freeze whatever happens to be in its way. Don't hold any body parts in front of it.

Pneumatic and hydraulic cylinders can deliver large forces. Your body parts can't handle large forces. You get the picture.

12.6. Chemical Fuel

For model airplanes there exist small internal combustion engines. These engines can be used to power robots either directly for propulsion or indirectly by driving an alternator or dynamo. A well designed system can power a robot for a very long time, but it's not advisable to use this power system indoors.

Note: This is another dangerous way of doing things. Fuel burns and is toxic. Small amounts of fuel in a open container can explode when ignited. Exhaust is toxic and a suffocation risk. Make sure of that you know what you're doing or get good life insurance.

13. Actuation Devices

Actuation devices are the components of the robot that make it move (excluding your feet). Best known actuators are electric motors, servos and stepper motors and also pneumatic or hydraulic cylinders. Today there are new alternatives, some which wouldn't be out of place in a good SF-movie. One of the new types of actuators are Shape Memory Alloys (SMA). These are metal alloys that shrink or enlarge depending on how warm they are. Another new technique is the use of air-muscles.

13.1. Motors

There are several different types of motors. Each motor type has several advantages as well as disadvantages depending on a particular robots design. See this Wikipedia article[1] for basic information about the different electro motors.

13.1.1. AC-motors

There are several different types of AC-motors, but their use is limited to high power stationary industrial robots. They are harder to use than DC-motors.

13.1.2. DC-motors

DC-motors are very easy to use, but like most other motors their usefulness for robotics is very dependent on the gearing available. DC-motors are made much more effective if they have an efficient gear ratio for a particular task. If your priority is to have a fast spinning motor and torque is of little concern a low gearing or even no gearing may be what you need; however, most motors used in robots need torque over top speed so a motor with a high gear ratio could be more useful.

The control of a DC motor can be split into two parts: speed and direction.

1 `http://en.wikipedia.org/wiki/DC_motor`

Direction

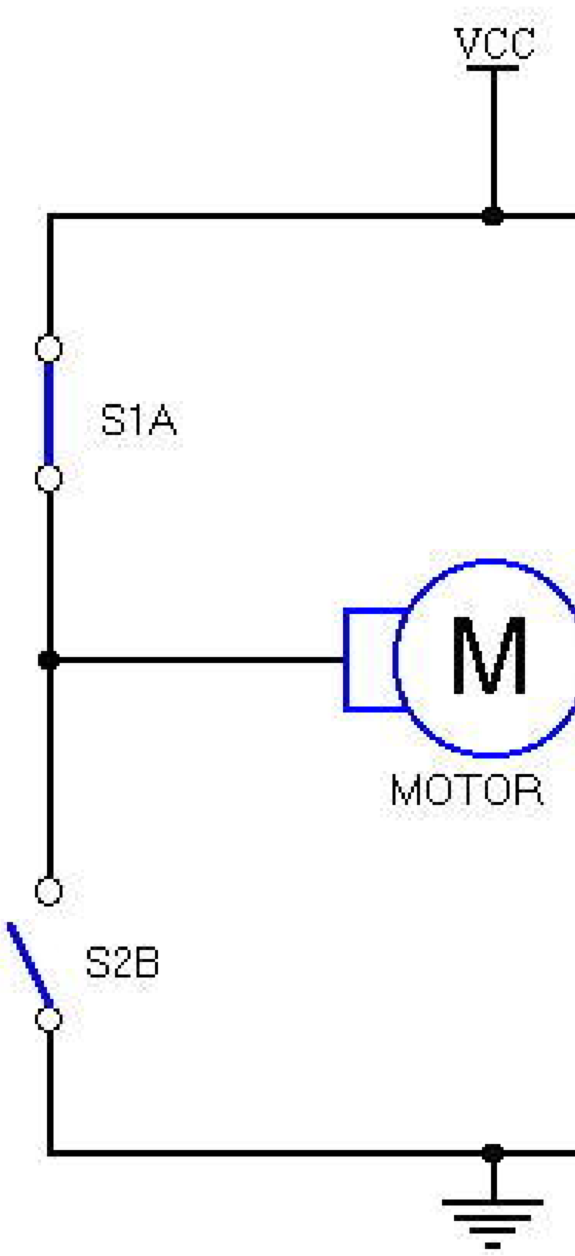


Figure 12 The principle behind a H-bridge

Changing which direction a DC-motor turns is very simple: simply reverse the polarity.

Both pairs of switches ((S1A,S1B) and (S2A, S2B))-see the picture on the right- will always switch together. This circuit is called an H-bridge. In a real design the switches can be several different components (Relays, transistors, FETs) or the whole circuit (without the motor) could be an IC (integrated circuit. use sugarcan relays

Speed

Speed is a little bit more complicated. Many beginners would try to slow down a motor by reducing its voltage with a variable resistor or other ways. This does not work well, because it will not only reduce the motor's speed, it will also reduce a motor's strength, while also consuming a lot of electricity as large amounts of heat are generated by the resistor.

A far better way is to use a PWM (Pulse-width modulation[2]) device.

13.1.3. Servo

What is a servo?[3]

Servos are used in robotics for different uses: e.g. to move a sensor around, or to move the legs of a robot. Some users modify the servo so they can use it as a DC-motor with a gearbox.

Controlling a servo is done with Pulse-width modulation[4]. The length of the pulse is relative to the position the servo will turn to. The length of this pulses is usually located between 1ms and 2ms, if so 1.5ms would be the center position. This pulse needs to be repeated with small intervals (otherwise the servo might turn to a "save" position or it might simply stay at its current position. This depends on the type of servo used).

13.1.4. Stepper motor

We cover stepper motors in more detail in the next chapter, Robotics/Stepper Motors[5].

(The category-box overlaps some of the text here. This line added to keep the real content on the screen)

13.2. Stepper Motors

13.2.1. Stepper Motor Basics

- **Degree of Rotation** - Every stepper motor has a **degree of rotation** . This value indicates how far the rotor will spin around it's axis for each signal sent.
- **Number of Coil** - These motors drive coils in a pattern to advance the rotor position
- **Driving a Coil** - This is done by passing current through the coil. In stepper motors, all coils must be driven in both directions over a complete revolution.
- **Sequencing** - These motors require the electronic control to change which coil in being driven at the correct time. Without this the motor will hold in one position.

2 `http://en.wikipedia.org/wiki/Pulse-width_modulation`
3 `http://en.wikipedia.org/wiki/Servomechanism`
4 `http://en.wikipedia.org/wiki/Pulse-width_modulation`
5 Chapter 13.2 on page 73

- **H-Bridge Style Driver** - ...

What's a stepper motor?[6]

In robotics stepper motors are primarily used in stationary robots as they tend to consume quite a lot of power. They are ideal for movements that have to be accurate and are larger than 180°.

13.2.2. Common Uses/Part Sources

- **Floppy, hard disk, or CD/DVD-ROM drives** - The heart of most these is the stepper motor. In most of these drives, there are two stepper motors. They position the read/write heads and they spin the medium.
- **Printers, scanners, plotters, and copy machines** - Most of these devices utilize stepper motors. They are used primarily to position the print head or optics on an (X,Y) coordinate system. One motor positions on the X axis and the other positions on the Y axis.
- **CNC machines, routers, scanners, lathes etc** - Used for X/Y/Z positioning of heads.

13.3. Shape Memory Alloys

Here[7] is a good introduction about what Shape Memory Alloys (SMA) are.

The main use of SMA in robotics is to imitate human muscles. One of the better known SMAs is Nitinol wire. It contracts about 5 to 7% of its length and consumes a lot of power.

One of the most popular nitinol driven robots is **Stiquito**[8] .

13.4. Air muscle

The concept of a fully autonomous, mission capable, legged robot has for years been a Holy Grail of roboticists. Development of such machines has been hampered by actuators and power technology and control schemes that cannot hope to compete with even some of the “simplest” systems found in the natural world.

13.4.1. Biomimetics

Faced with such a daunting task, it is not surprising that more and more researchers are beginning to look toward biological mechanisms for inspiration.

Biology provides a wealth of inspiration for robot design. There are millions of species of animals that have evolved efficient solutions to locomotion and locomotion control.

6 http://en.wikipedia.org/wiki/Stepper_motor
7 http://www.cs.ualberta.ca/~database/MEMS/sma_mems/sma.html
8 http://en.wikipedia.org/wiki/Stiquito

13.4.2. Legs, wheels, treads

Insects in particular are well known not only for their speed and agility but also for their ability to traverse some of the most difficult terrains imaginable; insects can be found navigating rocky ground, walking upside down, climbing vertical surfaces, or even walking on water. Furthermore, insects almost instantly respond to injury or removal of legs by altering stance and stepping pattern to maintain efficient locomotion with a reduced number of legs [1]. Given the ultimate goal of autonomy, this ability to reconfigure locomotion strategies will be crucial to the robustness of autonomous robots [2].

There are of course other mechanisms capable of producing locomotion, most notably wheels and caterpillar treads. While these devices are admittedly much easier to design and implement, they carry with them a set of disadvantages that inhibits their use in military or exploratory applications. Primary amongst these limitations is the simple fact that wheels, and to a lesser extent treads, are not capable of traversing terrain nearly as complex as that which a legged vehicle is capable of maneuvering over [2]. Even wheeled and tracked vehicles designed specifically for harsh terrains cannot maneuver over an obstacle significantly shorter than the vehicle itself; a legged vehicle on the other hand could be expected to climb an obstacle up to twice its own height, much like a cockroach can. This limitation on mobility alone means that in any environment without fairly flat, continuous terrain, a walking vehicle is far preferable to a wheeled or tracked one. Legged vehicles are also inherently more robust than those dependent on wheels or tracks. The loss of a single leg on a hexapod will result in only minimal loss in maneuverability; on a wheeled vehicle a damaged wheel could spell the end of mobility, and a damaged caterpillar tread almost always results in catastrophic failure. Finally, legged vehicles are far more capable of navigating an intermittent substrate—such as a slatted surface—than wheeled vehicles [3].

Given the preceding argument for the use of legged locomotion in certain environments, one is left with the daunting task of actually designing an efficient legged robot. While such a task is difficult to say the least, nature has provided us—literally—with a world full of templates. Animals can be found that are capable of navigation over almost any surface, and it is from these natural solutions to locomotion problems that engineers are more and more often seeking inspiration.

13.4.3. Actuators

2. Actuator Selection The selection of actuators plays a pivotal role in any mobile robot design, as the shape, size, weight and strength of an actuator must all be taken into account, and the power source of the actuators often provides the greatest constraint on a robot's potential abilities.

Muscle

Biological organisms have a great advantage over mechanical systems in that muscle, nature's actuator of choice, has a favorable force-to-weight ratio and requires low levels of activation energy. Their tunable passive stiffness properties are also well suited for energy efficient legged locomotion. The most frequently used actuators, electric motors and pneumatic/hydraulic cylinders, are far from equivalent to their biological counterparts.

Electric motor

Electric motors are probably the most commonly used actuation and control devices in modern day robotics, and with good reason. Motors in a wide range of sizes are readily available and they are very easy to control. These devices are also fairly easy to implement, normally requiring just a few electrical connections. However, electric motors have several disadvantages. Most importantly, their force-to-weight ratio is far lower than that of pneumatic and hydraulic devices, and in a field such as legged robotics, where weight is of the utmost importance, this makes them unsuitable for many applications. Typically, electric systems have a power to weight ratio of 50-100 W/kg (including only motor and gear reducer, operating at rated power), whereas fluid systems produce 100-200 W/kg (including actuator and valve weights) [4] and biological muscle, which varies widely in properties, produces anywhere from 40-250 W/kg [5]. In addition, when trying to take advantage of an animal's efficient biomechanical[9] design, the drastic difference between the rotary motion of most electric motors and the linear motion of muscle can cause complications.

Pneumatic and hydraulic cylinder

Pneumatic and hydraulic cylinder systems eliminate some of the problems associated with electric motors [6]. As a general rule, they provide a significantly higher force-to-weight ratio than motors; an advantage that in itself often leads to their use, even given the increased complexity and weight of control valves and pressurized fluid lines required for operation. These actuators also produce linear motion, which makes them more suitable to serving a role equivalent to muscle. Unfortunately, air cylinders are better suited to "bang-bang" operation; that is, motion from one extreme to another with mechanical stops to halt motion. Smooth walking motion requires a much larger range of states, and the stiction[10] present in most pressure cylinders makes even coarse position control difficult. Fluid pressure devices are still quite massive; for example, almost seventy-five percent of CWRU's Robot III's weight is composed of its actuators and valves [7].

Braided pneumatic actuator

Braided pneumatic actuators (BPAs) provide a number of advantages over conventional actuation devices, and share some important characteristics with biological muscle. These devices consist of two major components: an inflatable bladder around which is wrapped an expandable fiber mesh (Figure 1). The resulting actuator is significantly lighter than a standard air cylinder; however, the braided pneumatic actuator is actually capable of producing greater forces (and thus possesses a much higher force-to weight-ratio) than its heavier counterpart. When the bladder is filled with pressurized air, its volume tends to increase. Because of the constant length of the mesh fibers, this can only be accomplished by

9 `https://en.wiktionary.org/wiki/%20biomechanical`
10 `https://en.wikipedia.org/wiki/%20stiction`

the actuator expanding radially while simultaneously contracting along its axis. The result is a muscle-like contraction that produces a force-length curve akin to the rising phase of actual muscle [8].

An important property of BPA to note is that at maximum contraction (L/Lo≈0.69) the actuator is incapable of producing force; conversely, the maximum possible force is produced when the actuator is fully extended. Therefore, similar to muscle, the force output of these actuators is self-limited by nature. While an electric motor controller could conceivably become unstable and drive a system until failure of either the structure or the motor, a braided pneumatic actuator driven by an unstable controller is less likely to be driven to the point of damaging itself or the surrounding structure. Because of this property, braided pneumatic actuators are well suited for the implementation of positive load feedback, which is known to be used by animals including cockroaches, cats and humans [9].

BPAs are also known as McKibben artificial muscles [10], air muscles, and rubbertuators. They were patented in 1957 by Gaylord and used by McKibben in orthotic devices [11]. Like biological muscle, BPAs are pull-only devices. This means that they must be used in opposing pairs, or opposing some other antagonist. This property is of significant importance for useful application of these devices, for although it requires the use of two actuators or sets of actuators at each joint, it allows the muscle-like property of co-contraction, also known as stiffness control. If one considers a joint in the human body, such as the elbow or knee, it should be obvious that whatever

```
Figure 1: A placeholder
```

position the joint is in the muscles that control that joint can be activated (flexed) without changing the joint angle. From an engineering standpoint, this is accomplished by increasing the force produced by each muscle in such a way that the net moment produced at the joint is zero. As a result, the joint angle remains the same but perturbations, such as the application of an outside force, result in less disturbance. From a practical standpoint, this means that the joint can be varied through a continuum of positions and compliances independently. The resulting joint can be stiff when needed, such as when bearing weight while walking, or compliant, as in cases of heel strike where compensation for uneven terrain may be needed.

The greatest impediment to widespread use of BPAs has been their relatively short fatigue life. Under operating conditions such as we desire, these devices are capable of a service life on the order of 10,000 cycles as they were originally designed. A significant improvement to these devices has been made by the Festo Corporation, which has recently introduced a new product called fluidic muscle. This operates on the same principles as a standard BPA, with the critical difference being that the fiber mesh is impregnated inside the expandable bladder. The resulting actuators have a demonstrated fatigue life on the order of 10,000,000 cycles at high pressure.

3. Previous Robots

Two previous robots developed at CWRU have provided significant insight and impetus for the design of Robot V. Both of these cockroach-based robots are non-autonomous, and rely on off board controllers and power supplies for operation.

Robot III was the first pneumatically powered robot built at CWRU, and relied on conventional pneumatic cylinders for actuation. This 15 kilogram robot was powerful, and was demonstrated to be capable of easily lifting payloads equivalent to its own weight. The fundamental failing of this robot was the difficulty inherent in the control of the pneumatic cylinders; although capable of maintaining stance robustly and cycling its legs in a cockroach manner, to date this robot has not demonstrated smooth locomotion [12].

Kinematically similar to its predecessor, Robot IV implemented braided pneumatic actuators in place of Robot III's pneumatic cylinders. This robot was underpowered; it was barely able to lift itself, the valves were moved off-board for walking experiments. However, this robot was significantly easier to control, in large part because the valves allowed air to be trapped inside the actuators, so that joint stiffness could be varied as well as joint position. Using an open-loop controller, this robot was able to locomote [13]

4. Overview of Robot V

Design Case Western Reserve University's most recent robot, Robot V (Ajax) like its predecessors Robot IV and Robot III is based on the death head cockroach *Blaberus discoidalis* . Although it is not feasible to capture the full range of motion exhibited by the insect—up to seven degrees of freedom per leg—analysis of leg motion during locomotion suggests that this is not necessary. This is because in many cases joints demonstrate only a small range of motion, while the majority of a leg's movement is produced by a few joints. We have determined that three joints in the rear legs, four in the middle legs, and five in the front legs are sufficient to produce reasonable and robust walking [7] [14]. The different number of DOF in each set of limbs represents the task-oriented nature of each pair of legs. On the insect, the front legs are relatively small and weak, but highly dexterous (Figure 2), and are thus able to effectively manipulate objects or navigate difficult terrain. This dexterity is attained in the robot through three joints between the body and coxa. These joints are referred to (from most proximal to most distal) as γ, with an axis parallel to the intersection of the median and coronal planes (in the z direction); β, with an axis parallel to the median and transverse planes (in the y direction); and α, with

Figure 2: Schematic of front leg with axes of joint rotation

an axis parallel to the coronal and transverse planes (in the x direction). The two remaining joints are between the coxa and femur and the femur and tibia. The middle legs on the insect play an important role in weight support, and are critical for turning and climbing (rearing) functions; however they sacrifice some dexterity for power. On Robot V, the middle legs have only two degrees of freedom—α and β—between the body and coxa, and retain the single joint between the coxa and femur and the femur and tibia. Finally, the cockroach uses its rear legs primarily for locomotion, and although these limbs are not as agile as the others, they are larger and much more powerful; likewise, the rear legs of the robot have only one joint between each of the segments. The body-coxa joint uses of only the β joint. Although each leg has a unique design, one component they have in common is the tarsus, or foot, construction. This consists of a compliant member attached to the end of the tibia and a pair of claws. The compliant element is capable of bending to maintain contact with the ground, thus providing traction. The claws are angled differently on each leg to assist in its specific task; for example, the claws on the rear leg are angled backwards like spines, allowing the foot additional traction when propelling the robot forward.

4.1 Valves

Each joint is driven by two opposing sets of actuators, allowing for controlled motion in both directions (previous robots have used a single actuator set paired with a spring) [15]. Each actuator set is driven by two two-way valves; one for air inlet and one for air exhaust. This scheme doubles the number, and thus the weight, of valves as compared to Robot III; however, it allows for the implementation of stiffness control, or co-contraction. Because the pressures in opposing actuators can be independently varied, the same joint angle can be achieved using different combinations of actuator pressures; all that is required is that the moments on a given joint sum to zero at the desired position. As a result, a joint can be made very stiff by pressurizing both sets of actuators, or very compliant, by pressurizing one actuator only enough to overcome the mass properties of the limb to reach a desired position.

4.2 Stance bias

The actuators onboard[11] a legged robot can generally be subdivided into two classes: those used to move the limb through the swing phase and those required to maintain stance and generate locomotion. One of the fundamental differences between these two types of actuators is the load that is required of them. The swing actuators need only provide the force necessary to overcome the weight and inertia of the limb, whereas the stance actuators must support not only a significant portion of the entire mass of the robot, but also provide the force necessary for locomotion. This disparity between operational demands can potentially lead to large, powerful stance actuators and small swing actuators (as can be seen in the human body with powerful quadriceps muscles which maintain stance, and the respectively weaker hamstring muscles, which are used for swing); however, because of limited options for robot actuator sizes, it is more often the case that the swing actuators are overpowered, whereas the stance actuators are either underpowered or just capable of meeting the demands placed on them. On Robot V this problem was resolved through the placement of torsion springs at some critical load bearing joints (specifically the coxa-femur and β joints) to provide a bias in the direction of stance. As a result, the forces required of the stance actuators are significantly reduced while the swing actuators must produce greater forces, but still remain within their operational range.

5. Initial Trials

Robot V, like Robot IV, was designed as an exoskeleton, where the structural members are placed outside and around the actuators. Not only did this allow a significant reduction in weight, but it also provided a limited protection for the actuators, which are susceptible to puncture and abrasion (Figure 3). The vast majority of the structural elements were made of 6061-T6 Aluminum, although axles and actuator mounting shafts were made of 1018 steel, and fasteners were made from stainless steel. All joint axles were mounted in nylon journal bearings.

```
Figure 3: Robot V (Ajax)
```

Whenever possible, actuators were directly mounted to both their insertion and their origin. This precluded the need for tendons, allowing the maximum possible length of actuator to be used. This in turn maximized the force and stroke available for each individual joint.

11 https://en.wiktionary.org/wiki/%20onboard

The notable exception to this strategy was the β actuators, which were attached to a tendon and mounted parallel to the body. This was done to reduce the overall height of the robot. The first legs to be built were the middle legs. These were chosen for initial tests because they must be dexterous and forceful to maintain stance in a tripod gait. After completion of the first leg the range of motion (ROM) of each of its joints was measured and compared to the design values. These data are summarized below:

Joint ROM Desired ROM β 20° 30° α 25 40 c-f (coxa-femur) 40 50 f-t (femur-tibia) 75 75

These tests were performed at both 5.5 and 6.25 bar, with no significant difference between the results of the two, suggesting that at these pressures the actuators had reached their full contraction. Although the desired ROMs were not reached, the measured ROMs are in excess of those demonstrated by animal. The demonstrated ROM's of the leg were deemed sufficient for walking and climbing. A gantry was constructed to support the middle legs for preliminary stance and motion tests. With only horizontal support—to prevent tipping—the legs were able to maintain stance while supporting their weight (three kilos) plus the weight of the valves for the actuators (one half kilo) and a gantry element (one kilo) without any pressurized air in the actuators. This capability, a result of the aforementioned stance bias, clearly demonstrates the ability of these legs to support not only the weight of the robot, but a significant payload as well. An open loop controller was then used to cycle the legs through "push-ups"; raising themselves from a minimum to a maximum height. In this fashion, the legs were able to lift the body approximately 6 cm. This process was repeated with additional payloads (beyond valve and gantry weight) of two and a half and five kilograms using 6 bar air. In both cases, the legs were able to attain the same height.

6. Ajax

Fully assembled—including valves—Robot V weighs 15 kilograms. Range of motion tests have been performed for all joints, and are summarized below. In many cases, specifically the femur-tibia joints of all legs, these ranges of motion are in excess of the desired ROM. In all cases, they are sufficient for walking and climbing.

JOINT ROM Front Leg γ 35° β 45 α 25 c-f 40 f-t 75 Middle Leg β 20 α 25 c-f 40 f-t 75 Rear Leg β 25 c-f 50 f-t 80

Ajax demonstrates a propensity to stand due to the preloads placed on the torsion springs; even without pressure in the actuators, the middle and rear

Figure 4: Robot V without activated actuators (top) and standing (bottom). Note that even when the actuators are un-pressurized, they maintain a near-stance position, with only the feet contacting the surface.

Legs maintain a near-stance position. Initial tests of the robot have shown that it is capable of supporting its weight in a standing position and of achieving stance both unloaded and with a five kilogram payload (Figure 4).

Further tests have shown that the robot is able to achieve a tripod stance and alternate between tripods, which is important for walking. These tasks were achieved using a simple open loop controller. Furthermore, the passive properties of the BPA's are clearly highlighted in the robot's ability to return to its desired position after suffering perturbations without the use of any form of active posture control. Using a feed forward controller with absolutely no feedback, the robot can produce reasonable forward locomotion. Although this is by no means the robust, agile walking that is the ultimate goal of this project, it is a clear demonstration of not only the robot's capabilities, but also the advantages offered by the BPA's. The ability to move using only an open loop controller is in large part a result of the passive properties of the actuators, which provide compensation for any instabilities in the controller itself and immediate response to perturbations without the need for controller intervention.

This can be contrasted with Robot III, which, even with kinematic and force feedback, was not able to walk. This failure of Robot III is attributed to the inability of both the pneumatic cylinders and posture controller to deal with the sudden changes in load associated with locomotion.

In short, the BPA's act as filters, providing immediate response to perturbation; a task the controller is incapable of. This same process occurs in biological muscle, which responds nearly instantaneously to perturbation, but only slowly to neurological input [16]. With the addition of a biologically inspired closed loop controller in the future, Ajax is expected to display robust, insect-like locomotion.

7. Future Work

Although the mechanical aspects of this robot have been completed, the control system is still in its infancy. Because the mechanics of a system are inextricably linked to its control circuits, Ajax's controller is expected to benefit from the close relationship between its design and that of the actual insect. This relationship is perhaps most prominent in the muscle-like nature of the braided pneumatic actuators.

Sensors will be added to provide not only joint position feedback, but force feedback as well. Joint angle can easily be determined with a potentiometer, as has been done on our previous robots. Force feedback will be attained through pressure measurements from the actuators, which, given actuator length, can be used to determine actuator force. Although strain gauges properly placed on the mounting elements of the actuators can produce sufficient force feedback, previous work has shown many desirable characteristics inherent in pressure transducers: they have much cleaner signals, do not require amplifiers, and do not exhibit cross talk; all disadvantages of strain gauges. In addition, strain gauges must be mounted directly adjacent to the actuator they are recording from; this requires more weight at distal points of the limb, (thusly increasing the moment of inertia of the limb) and generally reduces the usable available stroke of the actuator. We have demonstrated that a pressure transducer located down-line from an actuator produces a sufficient signal to determine actuator force.

An insect-inspired controller was developed for Robot III, and this will be modified for use on Robot V. It is a distributed hierarchical control system. The local to central progression includes circuits that control joint position and stiffness, inter-leg coordination and reflexes, intra-leg gait coordination, and body motion. The inter-leg coordination circuit solves the inverse kinematics problem for the legs and the centralized posture control system solves the force distribution problem.

13.4.4. Further reading

w: Pneumatic artificial muscles[12]

- ”Biorobotics - Build Your Own Robotic Air Muscle Actuator”[13] is a page that explains what air muscles are and how to build one at home.
- Shadow robot company: Air Muscle overview[14]: ”Air Muscles are normally operated using compressed air in the 0-60psi (0-4 bar) range.”

13.4.5. References

1. Delcomyn, F Foundations of Neurobiology W.H. Freedman and Company, New York, 1998.
2. Raibert, M.H., Hodgins, J.K., Legged Robots, “Biological Neural Networks in Invertebrate Neuroethology and Robotics” ed. By Beer, R.D., Ritzmann, R.E., and McKenna, T. 1993.
3. Espenschied, K.S., Quinn, R.D., Chiel, H.J., Beer, R.D. (1996). Biologically-Based Distributed Control and Local Reflexes Improve Rough Terrain Locomotion in a Hexapod Robot. Robotics and Autonomous Systems, Vol. 18, 59-64.
4. Binnard, M.B. (1995) Design of a Small Pneumatic Walking Robot. M.S. Thesis, M.I.T.
5. Davis S.T., Caldwell D.G “The Bio-Mimetic Design of a Robot Primate Using Pneumatic Muscle Actuators” Proceedings of the 4th International Conference on Climbing and Walking Robots (CLAWAR 2002), Karlsruhe, Germany, 24-26 Sept. 2001.
6. Song, S.M., Waldron, K.J., Machines That Walk MIT Press, Cambridge, Mass., 1989.
7. Bachmann, R.J. (2000) A Cockroach-Like Hexapod Robot for Running and Climbing. M.S. Thesis, CWRU.
8. Klute, G.K., B. Hannaford, “Modeling Pneumatic McKibben Artificial Muscle Actuators: Approaches and Experimental Results,” Submitted to the ASME Journal of Dynamic Systems, Measurements, and Control, November 1998, revised March 1999.
9. Prochazka, A., Gillard, D., and Bennett, D.J., “Implications of Positive Feedback in the Control of Movement” The American Physiological Society, 1997.
10. Nickel, V.L., J. Perry, and A.L. Garrett, “Development of useful function in the severely paralyzed hand,” “Journal of Bone and Joint Surgery,” Vol. 45A, No. 5, pp. 933-952, 1963.
11. Caldwell, D.G, Medrano-Cerda, G.A., and Bowler C.J. “Investigation of Bipedal Robot Locomotion Using Pneumatic Muscle Actuators” . IEEE International Conference on Robotics and Automation (ICRA’97), Albuquerque, NM.
12. Nelson, G.M. (2002) Learning About Control of Legged Locomotion Using A Hexapod With Compliant Pneumatic Actuators. Ph.D. Thesis, CWRU.
13. Bachmann, R.J., D.A. Kingsley, R.D.Quinn, and R.E. Ritzmann, ”A Cockroach Robot with Artificial Muscles,” Proceedings of the 5th International Conference on Climbing and Walking Robots (CLAWAR 2002), Paris, 25-27 Sept. 2002.

12 `https://en.wikipedia.org/wiki/%20Pneumatic%20artificial%20muscles`
13 `http://www.imagesco.com/articles/airmuscle/AirMuscleDescription01.html`
14 `http://www.shadowrobot.com/airmuscles/overview.shtml`

14. Watson, J. T. and R. E. Ritzmann (1998). Leg kinematics and muscle activity during treadmill running in the cockroach, Blaberus discoidalis:slow running. J. Comp. Physiol. A 182: 11-22.
15. Powers, A.C. (1996) Research in the Design and Construction of Biologically-Inspired Robots. M.S. Thesis, University of California, Berkeley.
16. Loeb, G.E., Brown, I.E., Cheng, E.J. (1998). A hierarchical foundation for models of sensorimotor control. Exp. Brain Res. 1999, 126: 1-18.

13.5. Linear Electromagnetic

Linear Electromagnetic actuators consist of a hollow coil (solenoid) and a ferrometal rod. The rod is mounted loose in the coil and can move up and down. When current flows through the coil, the rod is pulled to the center of the coil. If the direction of the current is then reversed the solenoid will pull in the ferrometal rod. Due to Lenz's Law which states "For a current induced in a conductor, the current is in such a direction that its own magnetic field opposes the change that produced it." This means that an EMF will flow through the solenoid of the actuator to oppose the *change* in magnetic flux thus an electromagnetic actuator can never fully extend the full length of the rod.

Note to further modifiers: Feel free to tidy my Lenz's Law explanation up - i'm sure it could be written in a more elegant manner! --SamEEE[15] 03:18, 3 February 2007 (UTC)

13.5.1. Applications

This type of actuator is useful for momentary linear motion. e.g. closing a gripper.

13.5.2. Design Considerations

Solenoids use a large amount of power this requires more battery power which in turn requires more battery power to move the robot. A general rule of thumb is to use a electromagnetic actuators for small operations and for larger operations prehaps consider the use of Pneumatics the output power/weight ratio is usually more favorable. For more information on Pnuematics please consult the corresponding section of this book.

Calculating Force

Calculating how strong the actuator is, isn't very easy. However it is possible by using a Newtonmeter to the line of action and measuring how many Newtons are exerted by the solenoid on the measuring device. There are also a way of measuring force though use of a formula however the forementioned method is usually the most practical.

15 `https://en.wikibooks.org/wiki/User%3ASamEEE`

13.6. Piezoelectric Actuators

Piezoelectric actuators are actuators that take advantage of the piezoelectric effect found in certain materials.

13.6.1. Piezoelectric Effect

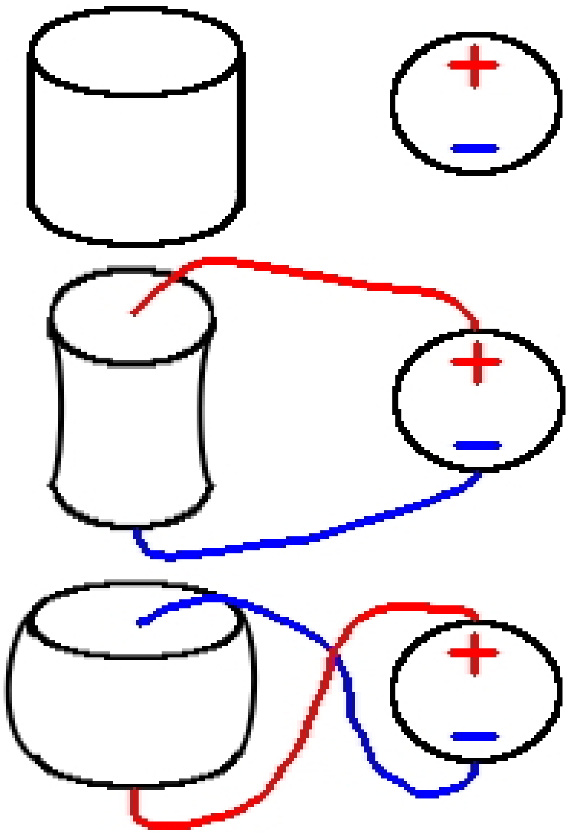


Figure 13 The piezoelectric effect as shown when an electric potential is applied

Certain materials have a characteristic of generating an electric potential when compressed or expanded. The amount of potential across the surface is determined by the force of displacement.

Because an electric potential is created from a change in volume, a change in temperature also has the ability of generating an electric potential.

The piezoelectric effect also has an inverse effect. When a voltage is applied to a material with piezoelectric properties, the material expands or contracts depending on the polarity of the voltage applied. The inverse piezoelectric effect is the basis of piezoelectric motors.[EEHandbook]

Materials with the Piezoelectric properties

Piezoelectricity naturally occurs in symmetrical crystals. Piezoelectric materials can also be manufactored as ceramics.

Crystals

- Quartz
- Tourmaline

Ceramics

- Barium Titanate

More Materials can be found by going here[16], and filtering for the piezoelectric category.

13.6.2. Piezoelectric Actuators

The ability to change shape when a voltage is applied can be used to displace objects. This basically allows electrical energy to be converted to mechanical energy.[elecman] This can be used to rotate motors or even

Piezoelectric Motors

Linear
The concept behind a linear piezo motor is layering a stack of piezoelectric discs that will push on each other and eventually push on a surface at an angle which will propel the object forward. The distance the propeller pushes is limited, but the frequency can be very rapid. The input voltage can change the output speed and allows a great amount of accurate movements. The force exerted is minimal and lessens the worry of breaking smaller objects.

Rotary
A rotary motor can work in the same fashion as the linear motor, except when you apply this concept at a much steeper angle to an axle, the linear motion is turned into a rotational motion. Two of these linear motors applied on both sides of a disc in the same direction can spin the disc. This is the same concept in a pitching machine used to sling baseballs.

Stepper
Using a piezoelectric material for a stepper motor can give you better precision due to the short distances involved. By placing piezoelectric motors in oppposing directions operating at frequencies that are slightly out-of-phase, the rotation is slight between the time when piezoelectric motors have the disc in a “hold” state. The slight movements and rapid response allow the rotation to be precise.

16 `http://www.matweb.com/search/MaterialGroupSearch.aspx`

Squiggle Motor[squiggle]

New Scale Technologies has taken this concept and applied it to a nut and bolt to create a type of screwing effect. The vibrations caused by the piezoelectric material cause the nut to exhibit a “hula hoop” effect to the threads, moving the screw up or downward depending on polarity. This can be used to push an object on a small scale very accurately.

13.6.3. Robotics Applications

Piezoelectric motors have the benefit of being very precise at very fast frequencies. They can also operate on a very small scale allowing robots to become smaller. Piezoelectric motors have the benefit of being able to exert a large displacement with a low driving voltage. Another benefit is the minimal amount of EMI/RFI noise created.

They can be wired in such a way to be senors as well. They can be used to determine the number of rotations, or if there is a force being exerted against a surface.

13.6.4. References

1. [EE Handbook]CRC and IEEE Press. The Electrical Engineering Handbook, 2nd Ed. Ed. Richard C. Dorf. CRC Press, 1997. Pg 1280.
2. [elecman]“Piezoelectric Actuator.”[17] electronics-manufacturers.com. 21 October 2008.
3. Stutts, Dr. Daniel. Piezoelectric Motor Research.[18] 21 October 2008.
4. Cook, Gordon. ”An Introduction to Piezoelectric Motors.”[19] Sensors. 01 December 2001. 21 October 2008.
5. [squiggle]”Squiggle Motors.”[20] New Scale Technologies. 21 October 2008.

13.6.5. External Links

- Piezo Motor/Actuator Kit[21]
- Piezo Linear Motor in Space Application[22]
- White Papers on Small Ultrasonic Piezo Motor Design[23]

17 http://www.electronics-manufacturers.com/products/sensors-transducers-detectors/piezoelectric-actuator/
18 http://web.mst.edu/~piezo/
19 http://www.sensorsmag.com/articles/1201/33/main.shtml
20 http://www.newscaletech.com/squiggle_overview.html
21 http://www.piezo.com/prodproto1MAkit.html
22 http://lisa6.gsfc.nasa.gov/conf/lisa6/posters/Poster_Biserni.pdf
23 http://www.piezo-motor.net/pdf/Piezo_Motor_White-Paper_Ultrasonic_Piezoelectric_Precision_Motor.pdf

13.7. Pneumatics Hydraulics

Pneumatic and hydraulic systems are actuators which provide linear movement. Hydraulic systems, especially, can produce extremely high forces.

13.7.1. Pneumatic systems

The Pneumatic Actuator is sometimes called a Pneumatic Ram. The basic Pnuematic Ram consists of 3 main parts. These 3 parts are the *Cylinder* , *Piston* and *Rod* .

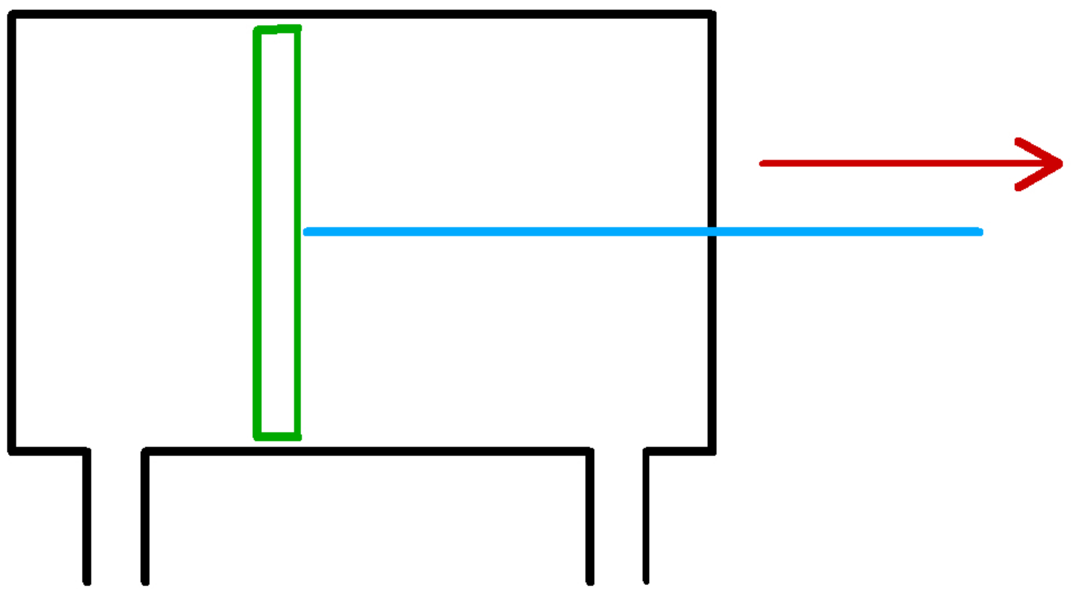


Figure 14 Dual acting cylinder extending.

The *Cylinder* , which is black in the diagram, is where the process takes place. Most of the cylinders in use for hobby robotics are constructed of stainless steel and depending on the type of ram it will either feature one or two ports for compressed air.

The *Piston* , marked in green, is similar to one that you would find in a car engine. It essentially provides a surface for the compressed gas to act upon.

The *Ram* , marked in blue, is connected to the piston and transfers the force from the piston to the mechanism that needs moving.

There are several different types of cylinders:

- Single acting cylinder: This cylinder has one input and has a spring. If the pressure is released from the cylinder the spring pushes the piston rod back to its starting position.
- Double acting cylinder. This type of cylinder is moved by pressurizing one side at a time of the piston. (see images below).

Applications

Pneumatics are useful for generating linear motion.

- Movement of a lever in a walker robot design.
- Closing a gripper

Design Considerations

The use of Pnuematics requires a supply of compressed gas. For a stationary robot on a production line a compressor could be set up nearby to supply it however this isn't possible for a robot that is on the move. The solution is to have a container that holds compressed air. If lots of compressed air is required the addition of a compressor into the design may be required however bear in mind that a compressor will consume a large amount of power.

Calculating Output Force

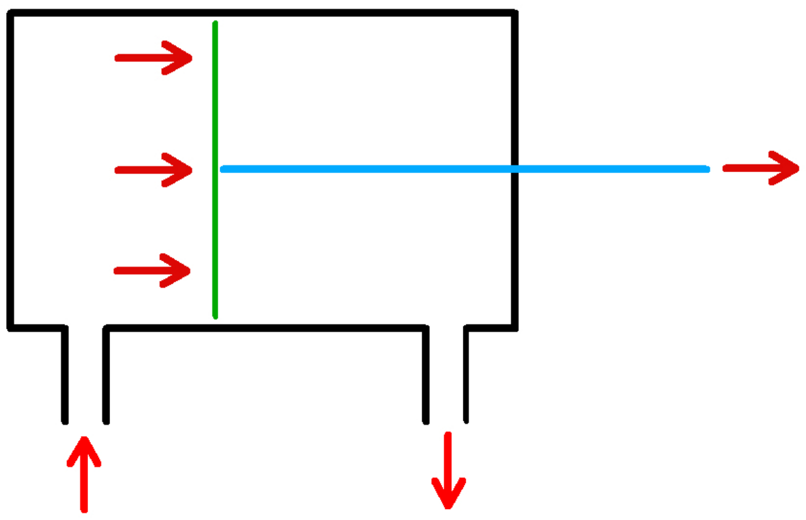


Figure 15 Dual acting cylinder extending.

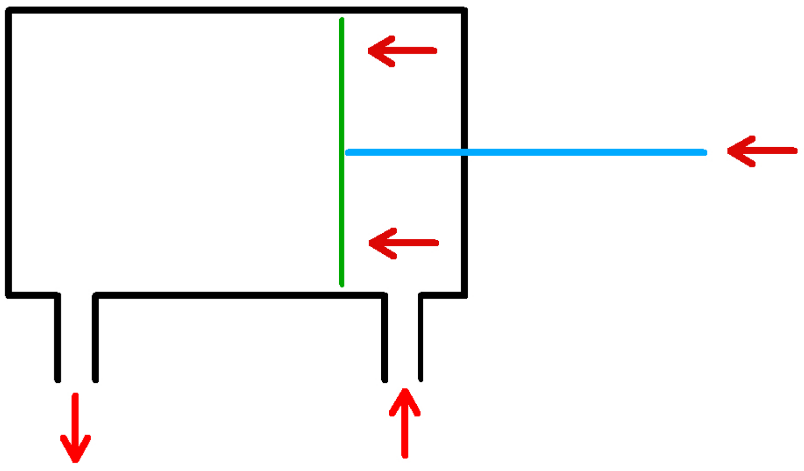


Figure 16 Dual acting cylinder contracting.

The force a cylinder can be expressed as

$$F = pA_{Piston}$$

- F : Force in Newtons
- p : Air pressure in cylinder in Pascals. Pressure can be obtained through use of a Pressure Gauge in the air circuit before the Ram.
- A : Surface area of the piston in square meters.[Sam1] This can be obtained from the manufacturers specifications.

Notes

1. [Sam1] Note: the surface area of the rod side of the piston is smaller than the other side.

13.8. Miniature internal combustion engines

These are the small internal combustion engines used in model cars and planes.

14. Grippers

A 'gripper' is a component of the robot used to manipulate an object loose from the robot itself. This can be a ball it needs to pick up, or the dirty socks it was programmed to find and dispose of. Grippers exist in many varieties ranging from simple; a thong-like design to complex arms.

How complex your robot's gripper has to be depends on what it's supposed to do with it. Picking up a ball requires a different approach than picking up socks.

14.1. Kind of grippers

Grippers are commonly modeled after tools, machines or the human hand. The choice behind a gripper doesn't differ much from the choice of the tool on a excavating machine, forklift, crane or other machine.

14.2. Scoops

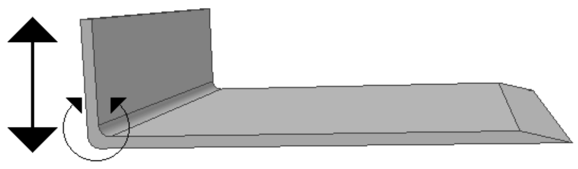


Figure 17
Flat Scoop

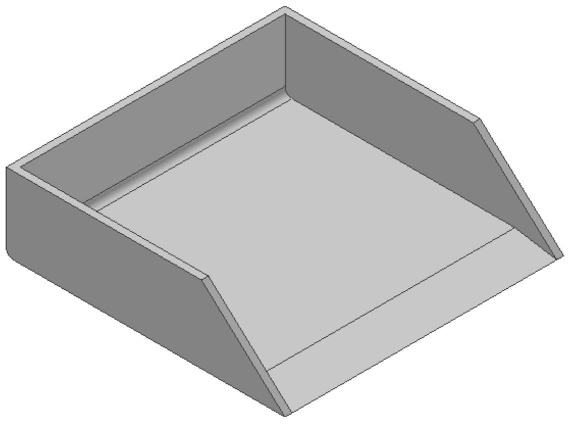


Figure 18
Raised Scoop

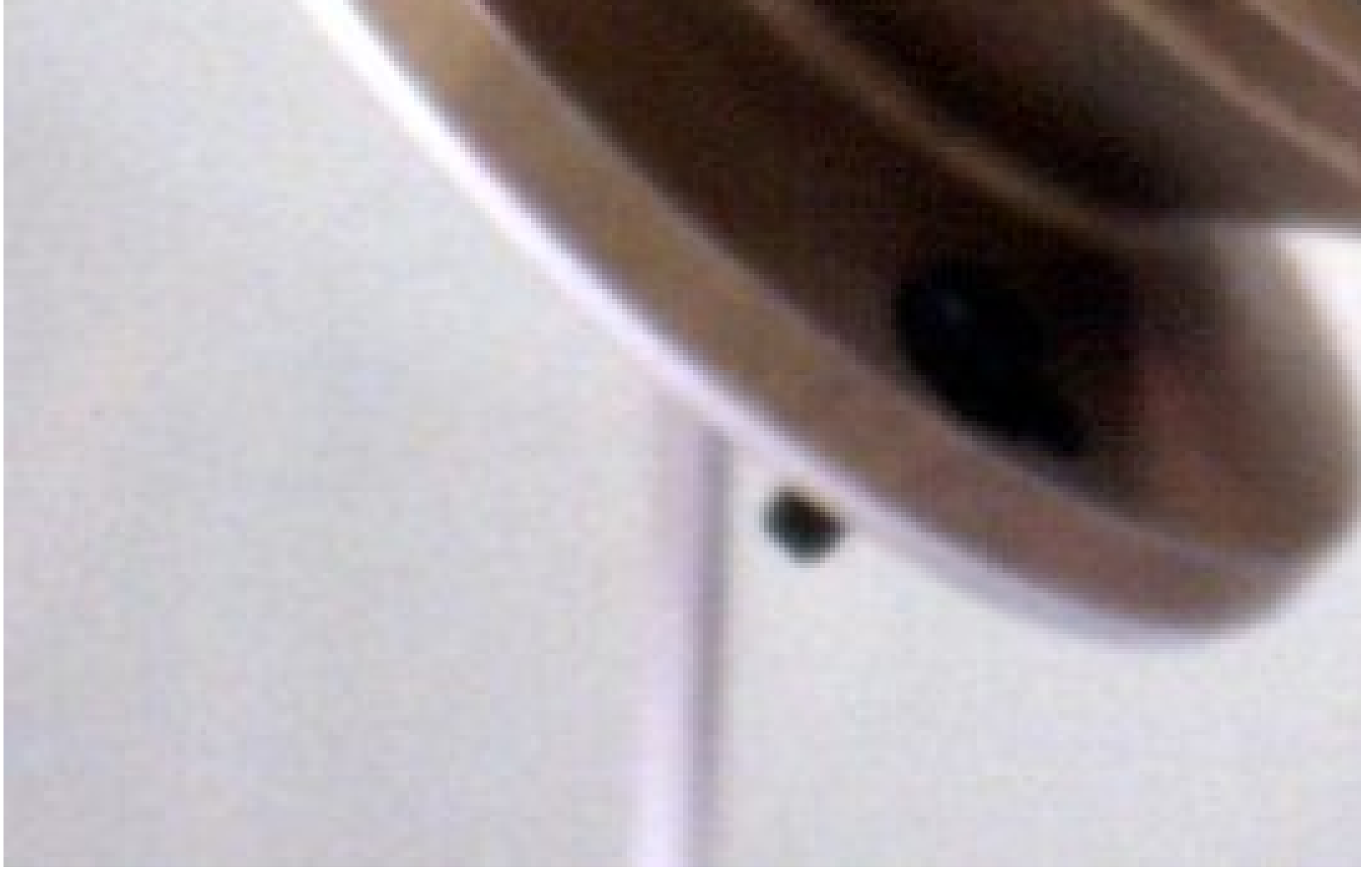


Figure 19 Robot Gripper

Many designs for robot hands have been tried including quite anthropomorphic hands.

Robotics has also led to a renewed interest in the 'design' of the human hand. The asymmetry of the human hand and in particular the fact that a rod like object gripped in the hand naturally falls at about 45° to the axis of the hand givews much greater flexibility in tool use than a symmetrical design would.[1]

14.5. Degrees of Freedom

Grippers also differ in their degree of freedom. Basicly there are 6 degrees of freedom: 3 translations and 3 rotations. See Theoretical Mechanics[2] about these degrees of freedom.

But generally, unless you're building an arm, you wouldn't need all 6 degrees of freedom (DOF). You get one DOF for free on a mobile robot: it can already move forwards and backwards on its own. Although adding gripper movement along this direction will increase your robot's accuracy, it's not really necessary.

How many DOF you need depends on how complicated the movement of your robot has to be. If it only needs to pick up a glass and move towards a goal and put it down again, it would be sufficient to make a gripper that can tilt back (towards the robot).

If you intend that your robot will have to be able to pick up Lego blocks and build a structure with them, you'll need more DOF.

1 The Physiology of the Joints, Volume One. Church Livingstone , ,
2 `https://en.wikibooks.org/wiki/Theoretical%20Mechanics`

14.6. References

15. Audio

The audio components of a robot are those parts that allow it to make sounds, either for entertainment purposes or to convey information.

This can be done through simple sound chips: cheap integrated circuits which allow certain sounds to be played with very few external components. These are commonly used in toys.

Other means are memory chips storing samples, a DAC and an amplifier, an MP3 player or voice synthesiser software on a portable PC.

15.1. The Theoretical Edge

Generating simple sounds is easy. All you need is an oscillator in the range 20 to 20.000Hz, an amplifier and a speaker. Such a circuit can generate any tone that a human can hear. By changing the frequency of the oscillator you change the emitted tone. Qbasic has a command that does just this: sound. Sound lets you specify a few parameters, with the most important being the frequency. By changing this frequency over time simple sound effects could be produced.

A better way to generate sound is sampling:

A sound wave has an intensity that varies over time. A microphone (with preamplifier) can convert this intensity to an electric signal with the same shape. An Analog to Digital Converter (ADC) can then convert this electric signal in a digital number. This is, however, not a continuous process but a discrete process, this means that every x ms the amplitude (voltage or current) of the signal is measured and converted to a digital number. This results in a digital approximation of the original signal. How good this approximation is depends on how frequently the signal is sampled. Of course more samples means more memory space is consumed and also more expensive hardware is needed. The price of an ADC depends on precision (how many bits represent the sample) and how many samples it can take per second (there are more parameters, but those are pretty much out of scope for this book).

Now you have an array of numbers. Of course now you want to turn this array back into sound. This is done with a Digital to Analog Converter (DAC). This part converts a digital value back into an electric signal. After this DAC there is a filter to smooth the wave, an amplifier and a speaker. Feed the array sample by sample to the DAC at the same rate as they were sampled and you hear the original sound back.

The advantage of this approach is it's (relative) simplicity. The disadvantage is its enormous memory consumption.

More advanced methods would be storing encoded sound (e.g. MP3). This is more tricky since it needs a decoder, yet these days integrated circuits with complete mp3 decoders exist, which makes this an interesting method to look for.

15.2. The Different Components

this section should cover some basic info on these circuits and ICs

15.2.1. Preamplifier

A preamplifier is used when a signal with very small amplitude has to be amplified. Preamplifiers are very linear (they cause very little distortion in the signal) but are very inefficient with power. They amplify a signal, coming from e.g. a microphone, which commonly has an amplitude of 1mV or less to a more useful amplitude of 100mV and up. Since they are so inefficient preamplifier circuits aren't used for power amplifiers.

15.2.2. Amplifier

Amplifiers allow a low power signal to be outputed through a speaker with significant power. Amplifiers commonly allow some distortion of the signal in order to preserve power. (except for class A amplifiers which consume a lot of power, but have little distortion).

Amplifiers can be built with ICs, discrete components or bought as a module.

15.2.3. Analog to Digital Converter ADC

An analog to digital converter (short: ADC) converts an analog signal to a digital value. ADCs have a certain accuracy, expressed in *bit* , e.g. 8bit, 10bit and 16bit. This means that the ADC will have a binary number with 8, 10 and respectively 16 bits. The highest value represented by this binary number correspondents to the reference voltage put on a special input of the ADC. e.g. If we have an 8bit ADC and we put on this special input 5V, then the value 255 means 5V. It's precision will be 1/255 * 5 V or 0.02V. Each time we add 1 to the binary number, the represented voltage increases by 0.02V.

A second point about ADCs is their timing. ADCs take some time to do a conversion. The faster they are, the more accurate they digitalise the signal (or in other words the higher the frequency of a signal can be), and the more expensive they are.

15.2.4. Digital to Analog Converter

The DAC converts a digital value into a voltage level. The work either with a reference voltage or "rail-to-rail", which means that their highest digital input value represents a voltage level (almost) equal to the DACs supply voltage. Simulare to the ADC the DAC has a certain accuracy in *bit* and requires some time to make a conversion.

15.2.5. Oscillators

An oscillator generates a repetitive signal. This can be of pretty much any shape you want, but most common shapes are sine, square, saw-tooth and triangle waves and many different forms of pulses.

15.2.6. Dedicated sound Chips

There are many ICs specially designed for producing sound effects.

- Oscillators: These output a signal with a user definable frequency. By varying the frequency you can make different sound effects.
- Sound chips: These output particular sound effects (e.g. motor sounds). These can't be changed.
- Sound recording and playback chips: These can record sounds for a short period of time (some can record several different samples) and can play these back afterward.

16. Video

Video components split up in 2 categories. On one hand you have a video camera, some form of transmission (wire or radio) and a display. On the other hand you have computer vision.

16.1. Camera, transmission, display

There are very small cameras, some even with built in wireless connection, which are cheap and need hardly any external components. These cameras can be mounted on a robot and let the user see through the robots ”eyes”.

If the robot has an on-board computer (single board or laptop) a webcam can be used. This allows robot control over a network or even the internet.

16.2. Computer Vision

Computer vision is a relatively new area. It tries to provide a computer with eyes and the ability to ”understand” what it sees. The first step is easy. Webcams have been around for quite some time. However the second step, understanding the image, is the hard one.

Image processing plays a substantial part in robotics where computer vision is used. There are many aspects to image processing including: demosaicing, image registration, differencing, recognition and segmentation.

16.3. Demosaicing

Demosaicing is perhaps the first step in image processing as it pertains to computer vision due to the fact that this process occurs just after the image is initially captured. In general, demosaicing is an algorithm that takes the image after being processed by a color filter array (CFA), such as a Bayer filter, and reconstructs a full color image. The image shown on the left is the output from the CFA and the image on the right shows the reconstructed image using the demosaicing algorithm. The result is an overall loss in detail and sharpness leading to a loss in resolution.

16.4. Image Registration

Image registration is the process used to combine multiple images into a single coordinate system. Combining images into a single coordinate system makes it easier to process the data contained in images taken at different times or from different perspectives. There are two main types of image registration based on how the images are combined into a single coordinate system. The first of which is intensity based, which uses correlation metrics to compare intensity patterns within the images to combine them. The other method is feature based, which uses points, lines and contours within the images to combine the images. Image registration has also been used in the realm of digital photography with the process of photo-stitching.

16.5. Image Differencing

Image differencing is used to obtain the changes between two images and is done by comparing the images pixel by pixel. A typical application of image differencing is with a traffic camera that needs to determine where cars are located in an image. By taking an image and finding the difference between it and the frame before it, the result is an image similar to the one shown where the outlines of the vehicles are neatly shown. The problem with this method is that there is a gap between the images in where the car is located. To solve this problem, taking the difference between the current image and a stock image that was taken when no cars were present gives a much clearer image depicted where the cars are located.

16.6. Image Recognition

The process of image recognition is difficult to pull off in the realm of computer vision. Nothing comes close to what a human can accomplish, but robots can do simpler tasks. Some of the simpler tasks that can be accomplished within computer vision are recognizing simple geometric shapes, human faces and fingerprints. Different aspects of the problems involved with image recognition include: object recognition, identification and detection. An implementation of image recognition is known as geometric hashing. The geometric hashing algorithm uses a model image of a particular object such as a computer mouse to see if it is located in a given image.

16.7. Image Segmentation

The process of image segmentation is used to simplify other aspects of image processing. This includes breaking an image down into smaller pieces making object detection and edge detection easier. The K-means algorithm can be used to segment the image into K clusters. The basic algorithm is shown below.

// Basic K-means algorithm

1. Pick K cluster centers, either randomly or based on some heuristic

2. Assign each pixel in the image to the cluster that minimizes the variance between the pixel and the cluster center

3. Re-compute the cluster centers by averaging all of the pixels in the cluster

4. Repeat steps 2 and 3 until convergence is attained (e.g. no pixels change clusters)

Part IV.

Computer Control

17. Control Architectures

Robots need some form of artificial intelligence in order to be able to navigate around a room or to solve a problem. Such AI can range from simple, as in following particular rules when a specific input is received, through more complicated, as in fuzzy logic rules or simple neural nets, to complicated, as in state of the art machine learning methods.

How complicated you can make your AI depends on a few factors:

- Processing power: How fast is your microprocessor, is it 8-, 16-, or 32-Bit, How many microprocessors do you use?
- Available memory: Both temporary and "permanent" memory. Limited amounts of memory restrict how much data can be held at a time, e.g. a map of the surroundings, sensor history, movement history. Systems with much permanent storage and little ram are slower than systems with large amounts of ram.
- Your knowledge: AI is a very broad subject. Parts of it are very math intensive and can be tricky to understand.

17.1. Reactive Systems

These are the simplest and one of the earliest form of control systems for robotics. Some Toys are a common example of these systems. They have a very fast response to the sensed information. However, the drawback of reactive systems is that they have to be very simple. The reactions should be simple and quick enough for the system to work in real-time.

These systems have negligible amount of intelligence, but a simple and well-defined behavior. This forms the basis of a Behavior controlled architecture or the Subsumption Architecture[1] (w:Subsumption architecture[2]). However, it's mentioned separately from the Subsumption Architecture as it can be used independently in its simplest form to make robots with low intelligence, but high real-time requirements.

17.2. Sense-Plan-Act

As obvious from the term, this architecture works through a cycle of Sense, Plan and Act processes.

1 https://en.wikibooks.org/wiki/Robotics%2FComputer%20Control%3A%20Control%20Architectures%3A%20Brooks%27%20Subsumption%20Architecture
2 https://en.wikipedia.org/wiki/Subsumption%20architecture

- By *sensing* , it means getting the required information from the available sensors and converting it to some usable form.
- *Planning* refers to the use of available information from sensing phase to determine the control parameters and sequences required for various components in order to make the robot proceed towards final goal.
- Finally, *act* phase is simply the implementation of the processes and sequences underlined by the planning phase.

This kind of approach is useful if the sensing process is slow and the environment is quite static. For example, on a low speed processor, image processing[3] may require lots of time to process each frame. Hence, it might not be feasible to process every frame and react to the gathered information in real-time. If we use this technique, the information will be processed at a very low frequency, providing more time for other real-time processes to run while still doing the work of planning and acting towards the goal.

17.3. Brooks' Subsumption Architecture

Behavior Based Control (BBC): Acting and thinking in layers, with complex logic divided into multiple simpler logic layers.

17.4. Hybrid Systems

Hybrid Systems, as evident from name, are a combination of different architectures. For example, a hybrid system may make use of a Subsumption Architecture[4] (w:Subsumption architecture[5]) for certain dynamic requirements and a Sense-Plan-Act[6] cycle for some other requirements in a static environment.

17.5. Swarm Robotics

17.5.1. Overview:

Swarm robotics is a relatively new study in the world of autonomous robotics. Swarm optimizations however have been around for over a decade for some functions. These techniques have recently been adapted and applied to autonomous robotic applications. On the topic of swarm robotics the word swarm is defined with several variations. On first hearing the phrase “swarm robotics” many simply think of basic interactions of robots such as follow the leader or simply data relayed between robots all traveling towards a common goal. However

3 https://en.wikipedia.org/wiki/image_processing
4 https://en.wikibooks.org/wiki/Robotics%2FComputer%20Control%3A%20Control%20Architectures%3A%20Brooks%27%20Subsumption%20Architecture
5 https://en.wikipedia.org/wiki/Subsumption%20architecture
6 https://en.wikibooks.org/wiki/Robotics%2FComputer%20Control%3A%20Control%20Architectures%3A%20Sense-Plan-Act

swarm robotics has taken further advancement in complex problem solving. Robots may now be given a goal and a basic suggestion on how to meet that goal. This suggestion may not be the most efficient way to achieve the goal much less even make it to the goal. So there must be a way to improve upon or optimize that suggestion. Optimization functions are then introduced. Several techniques covered here include Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO). These optimization functions will be covered in depth in the following.

17.5.2. Basic Swarm Robotics:

As stated earlier basic swarm robotics includes simple functions such as follow the leader and robot-to-robot interactions such as tag. In these scenarios the robots do not actively improve upon their current goal. They simply interact with one another in basic principles to achieve a predefined goal.

Examples:

- **EU Funded: I-SWARM**[7]

Figure 20

Swarm robotics is seen many times on a microrobotic scale as pictured above due to a number of factors. One of the main contributing factors to size is size in itself. A robot swarm composed of large robots can be bulky, expensive, and inefficient in data gathering. Obviously size does limit onboard equipment but that is where the principle of swarm shines. All the robots combined contribute a little more detail to the situation offering a complete picture. The swarm above funded by the EU is designed to combine all electrical aspects onto a single flexible board. The swarm has no major pending goals at this time and their behaviors are modeled after biological insects.

- **Rice University: James McLurkin**[8]

Figure 21 5px

James McLurkin, assistant professor at Rice University has developed his own DARPA funded swarm. His experimentation has explored physical data structures such as swarm arrangement according to ID number. Along with this he has also written a uniform dispersion algorithm. Additional swarm research includes his work on Robot Speed Ratios which is the study of message propagation versus the physical speed of the robot. This study was to uncover issues with the robots moving faster than they can physically interpret relayed messages.

- **Carnegie Mellon: Magnetic Swarms**

7 `http://www.i-swarm.org/`
8 `http://people.csail.mit.edu/jamesm/`

Research at Carnegie Mellon has been underway with a magnetic swarm of robots. The goal of this project is to develop a swarm of robots that could magnetically shape shift without any plane limitations. This could be used on the microscopic level as a three dimensional physical modeling aid. The robots pictured above use an array of electromagnets to control interactions between adjacent robots.

- **Free University of Brussels ("L'Université libre de Bruxelles")**

Several IRIDIA projects at ULB involve Swarm Intelligence[9]. Some of their software and hardware designs are posted at GitHub.

17.5.3. Ant Colony Optimization (ACO):

ACO is one of the simplest swarm optimization techniques proven to work reliably. This optimization technique was drawn directly from nature itself, showing that it was already a viable solution. The concept requires a brief understanding of how ants function in nature. Keep in mind, this definition has been tailored to our "perfect" example. In nature, ants wander at random. If an ant is to discover food it begins emitting pheromones and starts to wander back to its nest. At some point another ant discovers food and wanders back to the nest. This cycle happens many times. If an ant wandering back to the colony is to discover one of these pheromone trails it is more likely to travel it. If a fork is encountered biasing occurs via pheromone intensity. This means, ants are more likely to travel a trail with a stronger intensity of pheromones. Pheromones being volatile will evaporate over time. This means that the more ants traveling a trail the more pheromones are emitted. However if the same number of ants travel a long trail and a short trail over the same period of time the intensity of pheromones will be greater on the shorter trail. This means bias will always be placed on the shorter, more efficient trails thus naturally selecting the best solution.

This same process can be simulated in our electrical world. A proposed solution is established and the "ants" navigate to it. The initial solution is then modified to create the most efficient solution. Below is a basic depiction of ACO. As seen there are initially two trails established which happen to overlap in the middle. These two trails offer up four possible solutions to the optimized problem. Ants then randomly travel all four possibilities. The possibilities are narrowed down via the pheromones. All four solutions are traveled by the same number of ants but the shorter solutions have a greater intensity of pheromones biasing more ants towards the shorter trails thus eliminating all other solutions and converging on the best fit solution.

9 `http://code.ulb.ac.be/iridia.research.php?id=1`

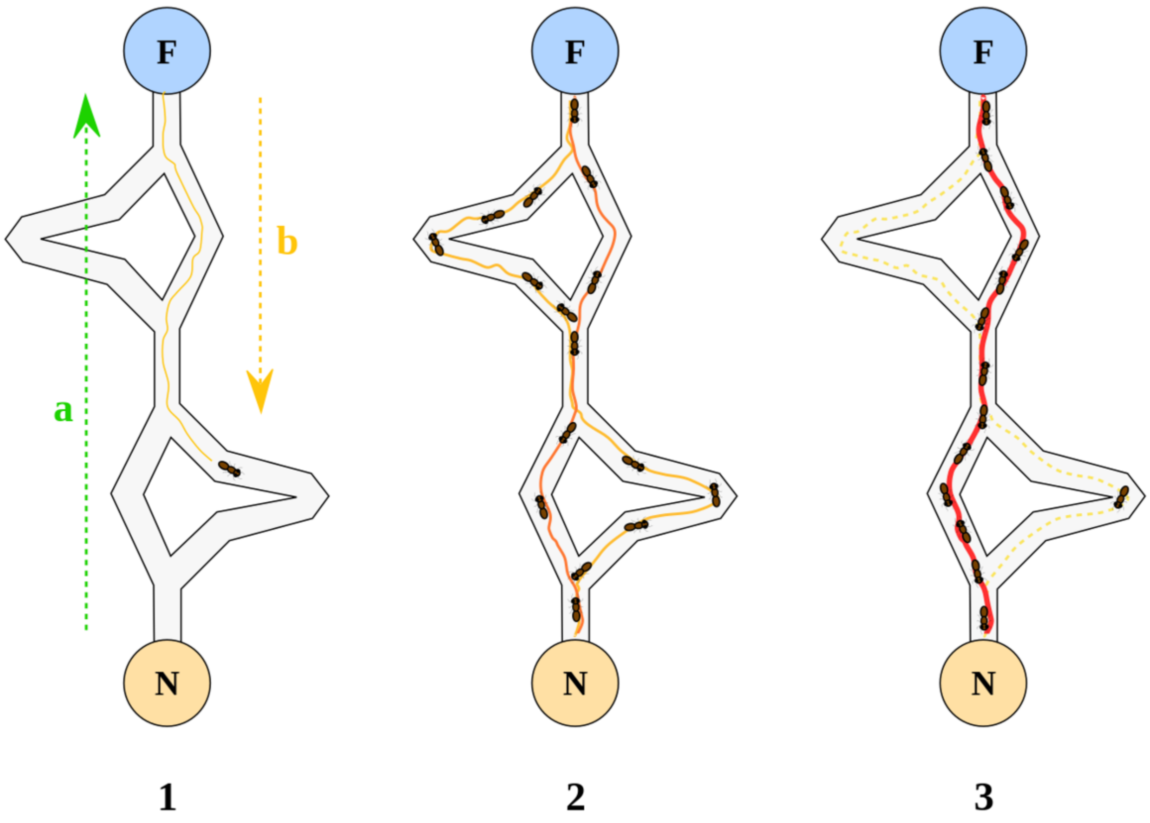


Figure 22

17.5.4. Particle Swarm Optimization (PSO)

PSO steps further away from stochastic functions towards an even more intelligent solution to a problem. Much of the PSO functionality is modeled after natural associations such as a flock of birds or a school of fish. In a PSO function, a generic solution is first given. Then a swarm of "particles" are initialized. They begin to attempt to make it to their goal. As they move towards their goal, they monitor certain other particles within a user-defined range of their own location. The particles around them contribute to a local best (lbest). This is the most fit solution discovered so far. Each particle also monitors its personal best (pbest) along with the global best (gbest) so far. Keep in mind that in some simulations lbest=gbest and the particles around each other do not have a local effect on each other. The particles are then biased towards a compilation of the lbest and pbest along with gbest. Their new velocity vector also includes a random scaling factor to prevent too early of a convergence on an inefficient solution. An equation is provided below to provide basic understanding of a swarm monitoring their pbest and gbest.

General PSO Equation

$v[] = v[] + c1 * rand() * (pbest[] - present[]) + c2 * rand() * (gbest[] - present[])(a) present[] = persent[] + v[](b)$

This equation also provides learning factors which in this case are scaled to “2”. From this equation we can see that the new velocity is equal to the scaled value of the sum of the difference of pbest-present and gbest-present. As it can be seen this optimization techniques offers a highly accurate convergence on the best fit solution to the problem.

Figure 23

- Provided above is a link to a java applet developed to help understand the operation of PSO. The animation applet can be found here[10].

10 `http://www.projectcomputing.com/resources/psovis/index.html`

18. The Interface

This chapter covers the hardware and its interface used to control the robot.

1. /Computers/[1]
2. Single Board Computers and multichip modules/[2]
3. /Microcontrollers/[3]
4. /Remote Control/[4]
5. /Networks/[5]

18.1. Personal Computers

Personal computers (PC) have a large number of ports to which you could add you own hardware to control your robot. Some of these are very easy to use, while others are nearly impossible without special (expensive) ICs. Not all of these interfaces are available on all computers. This section gives an overview of some of the best known ports on a PC. These ports and their uses are well document all over the internet.

18.1.1. External Ports

These are all the ports that are available on the outside of a PC. Most computer users are familiar with them (or at least know them by name and looks).

Serial Port

The serial port is one of the two easiest to use ports on a PC. This port consist of 2 wires to transfer your data (one for each direction) and a number of signal wires. This port is reasonably sturdy, and if you know some digital electronics or use a microcontroller, is pretty easy to use too. It is limited on speed and can only connect two devices directly. By using special ICs you can connect the serial port to a RS-485 network[6] and connect it to multiple devices.

- Serial Programming/RS-232[7]

1 https://en.wikibooks.org/wiki/%2FComputers%2F
2 https://en.wikibooks.org/wiki/%2FSBC%20and%20multichip%20modules
3 https://en.wikibooks.org/wiki/%2FMicrocontrollers%2F
4 https://en.wikibooks.org/wiki/%2FRemote%20Control%2F
5 https://en.wikibooks.org/wiki/%2FNetworks%2F
6 https://en.wikibooks.org/wiki/Serial%20Programming%2FRS-485
7 https://en.wikibooks.org/wiki/Serial%20Programming%2FRS-232

- This site[8] contains a lot information on serial ports (among others).

Parallel Port

The parallel port is the second of the easiest to use ports on a PC. This port uses 8 lines to transfer data and has several signal lines. Modern parallel ports can send and receive data. This port is easier to use, but less sturdy than the serial port. Since it operates on TTL voltage levels (0 and 5V) it can connect directly to microcontrollers and TTL logic.

This page[9] covers parallel ports and the differend modes of operation.

USB

USB is the successor of the serial port. It's faster and allows connecting devices without turning off the PC. Some modern microcontrollers have built in USB support.

- Everything[10] you'd need to know about USB.
- Serial Programming:USB Technical Manual[11]
- Embedded_Systems/Particular_Microprocessors#USB_interface[12]

IEEE 1394: Firewire, i.link, Lynx

w:FireWire[13]

IEEE 1394 also known as FireWire, i.link or lynx is a (much) faster port, similar to the USB port. It reaches speeds up to 400Mbit/s.

- The Connector[14]
- Article[15] about IEEE 1394.
- Linux FireWire wiki[16]

Keyboard Connector

Keyboards use TTL level signals to transfer button presses and releases to the PC. A keyboard sends a code when a button is pressed and sends another one when the button is released. This port could be used for some purposes.

8 `http://www.beyondlogic.org/`
9 `http://www.beyondlogic.org/`
10 `http://www.beyondlogic.org/`
11 `https://en.wikibooks.org/wiki/Serial%20Programming%3AUSB%20Technical%20Manual`
12 `https://en.wikibooks.org/wiki/Embedded_Systems%2FParticular_Microprocessors%23USB_interface`
13 `https://en.wikipedia.org/wiki/FireWire`
14 `http://www.hardwarebook.net/connector/bus/ieee1394.html`
15 `http://www.commsdesign.com/main/multsupp/0002/0002fire.htm`
16 `http://wiki.linux1394.org/`

See this site[17] for a in depth explanation and example of how to interface the keyboard. (menu: miscellaneous -> interfacing the AT keyboard) Covers both how to replace the keyboard as how to use the keyboard for other purposes.

Ethernet

Ethernet can be used to connect other devices to a PC. Complete webserver-on-a-chip are available these days, and an ethernet network can be a way to connect multiple devices in a robot (and even hook it up to the internet and let people control the robot from all over the world).

18.1.2. Internal Ports

These are the connectors inside the PC, generally these are used with special PCBs (called cards). Although harder to use, they offer great speed.

ISA

ISA was the (8-, later 16-bit) bus where you plugged your video, sound, IDE and network card in the old days. You'll find these on PC up to (some) early Pentium II (the latter usualy has only 1 E-ISA socket, if any). This bus is pretty easy to use for your own projects and well documented on the internet.

ISA bus connector[18] More indepth explaination of the ISA bus[19]

PCI

PCI is the successor of the ISA bus. It's a faster 32bit bus. Since it support plug and play, a PCI device needs a few registers which identify the component and manufacturer. This page[20] covers most of the basics of the PCI bus.

AGP

The Accelerated Graphics Port is aimed at 3D graphic cards. It's a fast bus, but optimized for graphics.

17 `http://www.beyondlogic.org/`
18 `http://www.hardwarebook.net/connector/bus/isa.html`
19 `http://www.hardwarebook.net/connector/bus/isa_tech.html`
20 `http://www.techfest.com/hardware/bus/pci.htm`

PCI Express

PCI Express replaces both PCI and AGP. It's quite different from all the other busses, as it uses serial communication, rather than parallel. Its speed depend on the number of "lanes" (serial connections) used PCI Express support 1, 2, 4, 8 and 16 lanes.

18.1.3. Wireless

These are "ports" too as they can be used to connect other devices to the PC.

IRDA

IRDA is an infrared communication port. Modern versions reach speeds up to 4Mbits/s. IRDA may be a good alternative to wires for table top robots. Since it's an Infrared port it needs a line of sight to work reliable and its range is limited to 1m. Note that this port works at a much higher speed than remote controls and therefor standard remote control repeaters may not work reliable for IRDA.

This[21] site covers the basics of IRDA.

Here[22] is a pinout of the mainboard IRDA connector.

WiFi / WLAN / 802.11 / Wireless Ethernet

All the names in the headline are synonyms for the same technology. WLANs are commonly used in PCs (especially laptops) as data networks. The bandwidth available is in the order of several megabits per second or more, far more than normally is necessary in any robotics project. A WLAN typically reaches about 100m, but with special directional antennas far more is possible (in a specific direction).

A WLAN is the obvious choice if your robot has an inbuilt PC or perhaps even PDA for control. Also, when you have ethernet connectivity in your controller (reasonably low cost but not a standard feature except in certain kits), there are low cost (˜€50) WLAN bridges available, such as the D-Link DWL-810+ and DWL-G810.

If you only have a serial port available, a *wireless device server* could be used. The cost of one of them is, however, over €100.

Bluetooth

Bluetooth is a low bandwidth protocol most commonly found in cellular phones. It is increasingly being deployed in laptops, and there are separate USB "sticks" available as well. Bluetooth can be used for making a serial connection wireless - there are Bluetooth

21 `http://www.hw.cz/english/docs/irda/irda.html`
22 `http://www.hardwarebook.net/connector/pc/mbirda.html`

serial ports available on the market, which can be used as converters. Total bandwidth in the system is about a megabit per second, with range up to about ten meters (standard Bluetooth, 2.5 mW), or about hundred meters (industrial Bluetooth, 100 mW). There are limitations on scaling with Bluetooth - it is mostly deployed in 1:1 links even though the standard includes networks with up to 8 active nodes (and even more passive ones). This means that if you plan on building large amounts of robots with a common communication network, Bluetooth might be less well suited.

- Bluetooth resources[23]

ZigBee

ZigBee is a protocol stack based on the 802.15.4 wireless network communication standard. It is low-cost, low-power and all-in-all perfectly suited for low-bandwidth communication needs. The bandwidth is on the order of tens to hundreds kilobits per second, and the range is up to about a kilometer, depending on equipment.

Interesting solutions are XBee from Maxstream[24], which basically provide a wireless serial link. There also is a list of other vendors at w:ZigBee[25].

UWB

Wireless USB

Cellular networks

A possibility is to use standard cellular networks (mobile phones). It is only a viable solution for large-scale geostationary outdoor applications with low communication needs though, because of cost, latency and bandwidth limitations.

Radio modems

Radio modems are normally older proprietary solutions for wireless linking of serial ports. Proprietary solutions probably shouldn't be considered for new designs.

18.1.4. Using a PC or laptop in a robot

PCs have the benefit of abundance of memory, program space and processing power. Additionally they provide the best debug I/O (screen, keyboard and mouse) you could wish for. But they do have a few flaws that limit their usefulness in a mobile robot.

- First of all their size. Even the smallest PC, a laptop, is quite bulky and forces you to use a rather large frame.

23 http://www.telecomspace.com/wirelessnw-bluetooth.html
24 http://www.maxstream.net/products/
25 https://en.wikipedia.org/wiki/ZigBee

- Secondly, except for a laptop, power consumption is large and provide AC-power on a mobile unit is bulky as you need heavy batteries and an inverter.
- Lastly Pcs are lousy when it comes to getting a reliable accurate timing from the outside world.

The first two points basically shape most of your robot's frame and other than using a different controller not much you can do about it. Picking the best hardware you can find is pretty much all that can make these points a little less troublesome.

The last point is quite easy to get around. Most PCs have a serial port. Most microcontrollers have a serial port as well. Use a level converter to connect the TTL level serial port of the microcontroller with the RS232 level computer serial port and use a little program that handles the accurate timing in the microcontroller and transfers this information to the computer through the serial connection. This is a very powerful setup that combines the strengths of both the PC and the microcontroller.

See this site[26] for example interfacing hardware for the serial port. Covers an I/O module and RS232 to TTL level converter for use with robotics or microcontroller based projects.

Some microcontrollers provide USB or Ethernet ports, which can be used in pretty much the same way, although the software would be a bit harder to implement.

18.2. SBC and multichip modules

18.2.1. SBC: Single Board Computers

Single Board Computers are a complete computer on a single printed circuit board. These usually require only a single power supply. SBCs commonly provide all standard PC I/O support like keyboard, mouse, SVGA, serial and parallel ports, ethernet, IDE, SCSI and USB. Some provide a PCMCIA connector. Late model SBCs are also replete with multiple CPU/Pentium 4/Xeon/AMD configurations.

SBC are commonly used in the industry in process control. Although they are quite expensive and hard to find (in retail), they can be a great way to control a larger robot. For small robots their power consumption would be a problem.

Stacking modular connector boards

Many robots include a "stack" of boards, typically with a processor on one board[27], H bridge motor driver on another board, and a wireless communication on another board.

Many people prefer to build rapid prototypes out of many single-purpose boards that can be disassembled and re-used for the next prototype, rather than making a single big specialized

26 `http://www.tronisoft.com/cat_rs232ioboard.php`
27 Chapter 18.2 on page 116

PCB prototype that is used once for testing and then thrown away. There are many standards for such electronic pieces.[28]

- w:PC/104[29]
- ”PlugaPodS”[30]
- gumstix[31] and gumstix packs[32] and more gumstix packs[33]
- Stackable USB (USB is smaller and faster than the ISA bus used in the original PC/104)
- Virtual Cogs[34] uses a stacking connector. Virtual Cogs wiki[35].
- *(Modular interface extension (MIX) stacking and communications interface)*
- ”pass-through 40-Pin OOPic expansion connectors”[36]
- R-Dev-Ino is a Robotic Development Arduino compatible board, easily stackable.`http://www.seeedstudio.com/wiki/index.php?title=R-Dev-Ino`
- the JeeNode Arduino-compatible board and JeeLabs modules that plug into it: a[37] b[38] c[39] d[40]
- The Tower System[41]
- Bug Labs[42]
- Xadow[43]
- The Grove system:[44]

[45][46][47] ”Cluster mode”[48] and ”Jigsaw mode”[49]

- Microsoft .NET Gadgeteer[50]

28 Timm Linder. ”A comparison: Arduino vs. .NET MF vs. .NET Gadgeteer + others” ˆ{`http://blog.timmlinder.com/2012/06/comparison-gadgeteer-netmf-arduino/`} . Compares and contrasts the ”DaisyLink” system, the ”.NET Gadgeteer” system, and the ”Seeedstudio Grove” system.
29 `https://en.wikipedia.org/wiki/PC%2F104`
30 `http://www.newmicros.com/`
31 `http://docwiki.gumstix.org/Roboaudiostix`
32 `http://docwiki.gumstix.org/index.php/Packs_and_boards_setup`
33 `https://www.gumstix.com/store/packs.php`
34 `http://virtualcogs.com/`
35 `http://virtualcogs.com/wiki/`
36 `http://www.oopic.com/protop1.htm`
37 `http://jeelabs.org/2011/01/29/new-ook-and-dcf-relay/`
38 `http://jeelabs.org/2010/08/25/sideways/`
39 `http://jeelabs.org/2010/11/20/indoor-temperature-etc/`
40 `http://jeelabs.org/2011/05/04/meet-the-jeenode-smd/`
41 `http://freescale.com/tower`
42 `http://news.cnet.com/8301-17939_109-9809763-2.html`
43 `http://www.seeedstudio.com/wiki/Xadow`
44
45 ”GROVE System” at Seeed Studio ˆ{`https://www.seeedstudio.com/wiki/GROVE_System`}
46 Grove-related posts on the Seeed Studio blog ˆ{`http://www.seeedstudio.com/blog/tag/grove/`} .
47 ”Better connector for Electronic Bricks?” ˆ{`http://www.seeedstudio.com/blog/2010/09/01/betterhopefully-connector-for-electronic-bricks-needs-inputs/`} discusses about the advantages and disadvantages of several connectors.
48 ”GROVE Starter Bundle” ˆ{`http://www.seeedstudio.com/blog/2010/10/16/grove-starter-bundle-alpha-cant-live-without-a-logo/`} has photographs illustrating ”Cluster mode” and ”Jigsaw mode”.
49 ”New electronic brick idea” ˆ{`http://www.seeedstudio.com/blog/2010/10/07/new-electronic-brick-idea-survived-some-intense-brainstorming/`} describes the electrical connections in ”Jigsaw mode” in more detail.
50

515253

- DaisyLink interface
- gobus ports

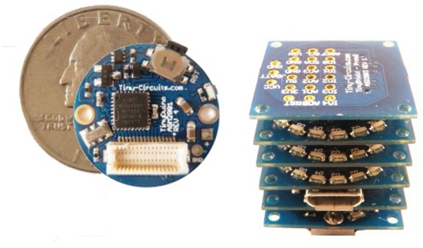


Figure 24 TinyDuino uses stacking connectors

- TinyDuino stacking connector
- Wouter van Ooijen's Dwarf Boards have 10 pin shrouded header connected by IDC ribbon cable connectors (GND, +5V, and 8 GPIO pins).
- "Stacking Arduino Shields"[54] (compatibility between shields can be a little tricky)
- ...
- ... *(Add to me)*

(If I think the stacking idea is good, but my robot is too small for PC/104, do you have any tips for picking an appropriate stacking connector, and arranging which electrical signal/power goes where?)

The "stackable headers"[55] are one way to stack boards together. What other options do we have for stacking connectors?

18.2.2. Multichip Module Boards

Is there a better name for this sort of thing? See Talk:Robotics#terminology[56].

Multichip Module boards are lighter versions of the SBC. These boards provide less I/O abilities than a full fledged SBC, but are considerably less expensive. For example the Acme Foxboard[57] provides a 100Mips processor with 16MB RAM and 4MB Flash running Linux. This board has IDE, SCSI, USB, Ethernet, I2C and more on a surface of aprox. 6x7cm. It consumes about 280mA and costs around €170. While this particular board is designed for embedded internet-enabled applications, it's a great board for controlling your robot. Another alternative is to use a Linksys router and install OpenWRT on it. You can usually

51 "Introducing .NET Gadgeteer" ^{http://gadgeteer.codeplex.com/}
52 "Home - Gadgeteer" ^{http://www.netmf.com/gadgeteer/}
53 http://gadgeteer.codeplex.com/wikipage?title=.NET%20Gadgeteer%20Socket%20Types ".NET Gadgeteer Socket Types"]
54 http://www.freetronics.com/pages/stacking-arduino-shields
55 "Stackable Header Kit" ^{https://www.sparkfun.com/products/11417}
56 https://en.wikibooks.org/wiki/Talk%3ARobotics%23terminology
57 http://www.acmesystems.it/index.php/Welcome_to_the_Acme_Systems_home_page

pick these up for around $50. If you spend some time shopping around you may find similar boards that are better or cheaper.

18.2.3. Further reading

w:single-board computer[58]

- linuxstamp board[59] runs Linux -- open hardware
- ARMUS Embedded Linux Board[60] -- open hardware
- The Balloon Project[61] designs boards that can run Linux -- open hardware
- "Teeny weeny Linux SBCs"[62]
- "Linux computer fits in USB key"[63]. . Retrieved
- TINI, the Tiny INternet Interface: TINI reference design[64] TINI board webring[65] "Unofficial TINI Information Site"[66]
- small motherboards that run Linux[67]

18.3. Microcontrollers

Microcontrollers are the core of many robots. They have considerable processing power packed on to one chip, allowing lots of freedom for programmers. Microcontrollers are low level[68] devices and it is common to program them using an assembly language[69], this provides a great deal of control over the hardware connected to the controller. Many manufacturers also provide high-level language[70] compilers[71] for their chips, including BASIC[72] and C[73].

What's the difference between a *microcontroller*[74] , *microprocessor*[75] , and a *CPU*[76] ? The CPU is the part which actually executes the instructions (add, subtract, shift, fetch, etc.).

A *microprocessor* is any CPU[77] on a single chip.

58 https://en.wikipedia.org/wiki/single-board%20computer
59 http://opencircuits.com/Linuxstamp
60 http://opencircuits.com/ARMUS_Embedded_Linux_Board
61 http://balloonboard.org/
62 http://web.archive.org/web/20060318191955/www.linuxdevices.com/articles/AT8498487406.html
63
64 http://www.maxim-ic.com/products/tini/chipsetrefdesign.cfm
65 http://e.webring.com/hub?ring=tiniring
66 http://www.rawbw.com/~davidm/tini/
67 http://opencircuits.com/Projects#Motherboards_that_Run_Linux
68 https://en.wikipedia.org/wiki/Low-level_programming_language
69 https://en.wikipedia.org/wiki/Assembly_language
70 https://en.wikipedia.org/wiki/High-level_programming_language
71 https://en.wikipedia.org/wiki/Compiler
72 https://en.wikipedia.org/wiki/BASIC_programming_language
73 https://en.wikipedia.org/wiki/C_programming_language
74 https://en.wikipedia.org/wiki/microcontroller
75 https://en.wikipedia.org/wiki/microprocessor
76 https://en.wikipedia.org/wiki/CPU
77 https://en.wikipedia.org/wiki/CPU

A *microcontroller* is a kind of microprocessor, because it includes a CPU, but it typically also contains all of the following components on the same single chip:

- (discrete) inputs
- (discrete) outputs
- ROM[78] for the program
- RAM[79] for temporary data
- EEPROM[80] for non-volatile data
- counter/timer
- clock

Some microcontrollers even include on board Analog-to-Digital converters (ADCs). This allows analog sensors to be directly connected to the microcontroller.

With this capability, microcontrollers are quite convenient pieces of silicon.

The outputs of a microcontroller can be used to drive many things, common examples include LEDs[81] and transistors[82]. The outputs on a microcontroller are generally low power. Transistors are used to switch higher power devices (such as motors) on and off.

All CPUs are useless without software.

Most software for a PC is stored on the hard drive. But when you first turn one on, it starts executing the software in the boot ROM. If you wanted to change that software, you would have to pull out the ROM chip, program a new ROM chip (in a ”chip programmer”), and then plug the new chip into the PC.

Most robots don't have a hard drive -- **all** their software is stored in ROM. So changing that software is exactly like changing the boot code of a PC. (If your robot has an external ROM chip, then that is the one that is pulled and replaced. If your robot uses a microcontroller with internal ROM, then the microcontroller is pulled and replaced).

Many recent PC motherboards and microcontrollers use Flash[83] instead of ROM. That allows people to change the program without pulling out or putting in any chips. They can be rewritten with new data, like a memory chip, but permanently, and only a certain number of times (10,000 to 100,000 erase/write cycles).

Here are a few pages about specific µcontrollers:

- 8051[84]
- Atmel AVR[85]
- Microchip PIC[86]
- Embedded Systems/ARM Microprocessors[87]

78 https://en.wikipedia.org/wiki/Read_only_memory
79 https://en.wikipedia.org/wiki/Random_access_memory
80 https://en.wikipedia.org/wiki/EEPROM
81 https://en.wikipedia.org/wiki/LED
82 https://en.wikipedia.org/wiki/Transistor
83 https://en.wikipedia.org/wiki/flash%20memory
84 https://en.wikibooks.org/wiki/Embedded%20Systems%2F8051%20Microcontroller
85 https://en.wikibooks.org/wiki/Embedded%20Systems%2FAtmel%20AVR
86 https://en.wikibooks.org/wiki/Embedded%20Systems%2FPIC%20Microcontroller
87 https://en.wikibooks.org/wiki/Embedded%20Systems%2FARM%20Microprocessors

18.3.1. Further reading

- Embedded Systems/Particular Microprocessors[88] describes some of the more popular microcontrollers.

18.4. Remote Control

Remote control is about controlling a robot (or any other electronic device) from a distance, either with or without a wire. Remote control methods can be split into two categories: wireless and wired.

18.4.1. Wired Remote Control

Wired remote control or tethered control can be the right way to interface a computer with a stationary robot. For mobile robots the cable can become a burden for the robot.

Issues With Wired Remote Control

Limited Range

- Electric signals transferred over a wire lose energy because of the wires resistance. The result is that the amplitude of the signal decreases as distance increases.
- Reflections can be a problem when the data rate is high. This means a previous signal doesn't disappear before the next is transmitted. This is why transmission lines are "terminated" with a resistor to ground.
- Interference is caused by the magnetic fields of other conductors or capacitive coupling of high speed signals in other conductors. Shielding cables reduces interference, as does using differential signals (instead of using amplitude relative to ground to transmit '1's and '0's, using amplitude between 2 signal wires) through a twisted pair of conductors.

Mechanical Issues With Cables

- Cables have fixed number of wires in them, if you need more, you'll have to replace the whole cable, which can be very time consuming.
- Cables have a certain stiffness. The thicker the cable the more force you need to apply to bend the cable.
- Cables have a weight. This can make it hard for smaller robots to drag it around.
- They can get in the way of the robot.

88 `https://en.wikibooks.org/wiki/Embedded%20Systems%2FParticular%20Microprocessors`

Methods

Many of the mechanical issues of cables can be reduced by using thin cables with as few conductors as possible. Ideally such a cable would have only 3 or 4 conductors: Ground, power and one or 2 signal wires. See Networks[89] for sending multiple signals through as few wires as possible.

Advantages

By using a cable you get around the problem of heavy batteries. The robot can be powered by an AC-outlet. Another benefit of a tether is the ability to easily use a PC to control the robot.

18.4.2. Wireless Remote Control

IR

IR remote control is the best known form of wireless remote control. It's cheap and reliable, but limited to line-of-sight communication. Complete IR-receiver modules, like the TSOP1736, are available cheaply and can be interfaced with most controllers without much extra components. TV remote controls using RC5 (Phillips) can be used with such modules. If you want a faster data link, IRDA components could boost it significantly. Bluetooth and Wifi have replaced it on modern laptops, but IRDA components are still available.

RF

RF is widely known in model race cars, Wifi, and various other applications. These days complete RF transmitter/receiver modules are available at reasonable low prices. These modules are very easy to use and have ranges of around 100m depending on their environment. RF remote controls for high end model race cars have larger range but are much more expensive and limited in their use.

While it is definitely possible to build RF transmitters from scratch, this is not advisable. Radio frequencies are strictly governed and building a transmitter that uses the wrong frequency quickly leads to a fine or worse. Know what you're allowed to do, before building one of these. It is possible to use a wireless telephone to provide an RF connection to your robot. The major restriction being data rates limited to 9.6kbaud or so.

Speech Recognition

In essence speech recognition is a form of remote control. Probably one of the hardest forms of remote control, but also one of the most impressive ones. Although today there are

89 `https://en.wikibooks.org/wiki/Robotics%3A%20Computer%20Control%3A%20The%20Interface%3A%20Networks`

modules that contain a full speech recognition system capable of learning a dozen commands, those systems are still very limited as they can't handle sentences (just commands), need to be trained before they are useful and usually can only be used by one person.

Sound

Sound can also be used as remote control, generating a tone of a particular frequency isn't hard, building a receiver to detect this tone isn't too difficult either. Sounds like whistling and clapping hands have been used for remote control before (e.g. the keyring which makes a sound when you whistle).

18.4.3. Network control

A further step would be to do the control over a network, from another device. This could be a wired network, like RS-232, RS-485 or Ethernet, or a wireless one, as WLAN, Bluetooth or ZigBee.

18.5. Networks

Sometimes a single µcontroller isn't sufficient to control your robot. Then you'll be needing a way to connect all those µcontrollers, preferably without sacrificing too many pins or expensive ICs. Of course this problem has been solved a long time ago and there are quite a number of different standards each having their own advantages and disadvantages.

18.5.1. Standards

There are many different standards on connecting 2 or more µcontrollers (or computers), but here are the most used standards:

(See Serial communications bookshelf[90] *for more detailed information about serial communication standards)*

I²C

Inter-Integrated-Circuit-bus or Two-wire serial bus: Used to connect ICs on a single board. The bus has one clock and one data line. Both the clock and data line are pulled high and device only drives the lines low. There are plenty of ICs available with build-in I²C interface including many of the modern µcontrollers.

µcontrollers with build in I²C support:

- ATMEGA8
- ATMEGA16

90 `https://en.wikibooks.org/wiki/Serial%20communications%20bookshelf`

- ATMEGA32
- ATMEGA64
- ATMEGA128
- ATMEGA8535

Some I²C ICs:

- MAX5380/5381/5382: 8Bit DAC
- PCF8574: 8bit I/O-expander for I²C-bus
- LM75: digital temperature sensor

The I²C protocol can also be performed in software and is usually referred to as bit-banged I²C.

See:Philips Semiconductors[91]

RS232

Recommended Standard 232: Better known as the serial port on a PC. Used to connect two devices.

See: Further down the page[92]

RS422

Recommended Standard 422: industrial version of RS232. Much better than RS-232 at resisting interference.

See: RS422 tutorial[93]

RS485

Recommended Standard 485: Better version of RS422: allows more than 2 devices to be connected together. (usually up to 32 devices)

RS232 <-> RS485 converters:

- LTC485CN8: DIL8
- SN75LBC176D: SOIC8

See: RS485 tutorial[94]

91 `http://www.semiconductors.philips.com/markets/mms/protocols/i2c/`
92 `http://www.beyondlogic.org/`
93 `http://www.arcelect.com/rs422.htm`
94 `http://www.arcelect.com/RS485_info_Tutorial.htm`

CAN

Acronym stands for "Controller Area Network." More complicated network. Used in automotive and domotica. Originally developed by Bosch in Germany. Limited to 1 Mbps in theory; with required overhead, protocol is slower than 1 Mbps. Data is delivered in packets of 8 bytes. CAN is frequently referred to as CANbus.

See: Philips Semiconductors[95]

1wire

This bus uses 1 wire to supply power and communication to a IC. Used for temperature sensors and other low-power ICs. Despite the name, a second wire, ground, is used to complete the circuit. Data is sent to the IC by switching the wire between high and low. A built-in capacitor provides the IC with power during the low parts of the signal. This bus is intended for low-power devices, like temperature sensors.

See: Maxim-IC[96]

SPI

SPI(Serial peripheral interface) is a 4-wire full duplex bus. The bus has a clock, transmit, receive, and select line. One device controls communication on the bus. When connecting multiple devices, each device is connected to the master with a separate select line and the master selects only one device at a time.

18.5.2. Further reading

- Embedded Systems/Common Protocols[97]

95 `http://www.semiconductors.philips.com/markets/mms/protocols/can/`
96 `http://www.maxim-ic.com/appnotes.cfm/appnote_number/1796`
97 `https://en.wikibooks.org/wiki/Embedded%20Systems%2FCommon%20Protocols`

Part V.

Sensors

19. Environment sensors

Environment sensors are sensors that detect and interpret things in the outside world (such as obstacles, sounds, and signs) or about the outside world (such as temperature, pressure, and viscosity).

Some of the more common kinds of environment sensors are discussed here.

19.1. Object Sensors

There are several sensors that are designed to determine the location and size of various objects in the world.

- Digital cameras[1] can, with enough processing power, locate specific objects in their field of view.
- Tactile sensors[2] can detect how much force is being applied to a specific point.
- Distance sensors[3] use various methods to determine the distance to the nearest object in a given direction.

19.2. Media Sensors

Some sensors detect properties of the immediate environment, such as:

- Temperature[4]
- Pressure & Sound[5]

1 https://en.wikibooks.org/wiki/..%2FSensors%2FComputer%20Vision
2 https://en.wikibooks.org/wiki/..%2FSensors%2FTactile%20Sensors
3 https://en.wikibooks.org/wiki/..%2FSensors%2FDistance%20Sensors
4 Chapter 20 on page 131
5 Chapter 21 on page 137

20. Thermal Sensors

Robotics[1]: Sensors[2]: **Thermal Sensors**

Temperature can be an interesting parameter to measure. The temperature of the surroundings might not be very useful for a mobile robot, the temperature of a motor or battery back is.

20.1. What is temperature?

"Temperature: A measure proportional to the average translational kinetic energy associated with the disordered microscopic motion of atoms and molecules. The flow of heat is from a high temperature region toward a lower temperature region.[3]

20.2. Units and their conversions

The most used units for temperature are Degrees Celsius, Kelvin, and Degrees Fahrenheit .

20.2.1. Degrees Celsius

This temperature scale is defined around water. It defines 0°C as the temperature on which water freezes at standard atmospheric pressure. 100°C is defined as the temperature at which water boils.

Conversion:

To Fahrenheit: Tc * 9/5 +32

To Kelvin: Tc + 273,15

1 https://en.wikibooks.org/wiki/Robotics

2 https://en.wikibooks.org/wiki/Robotics%23Sensors

3 -Courtesy of the references cited below but mostly based on the operational definition on the HyperPhysics pages at Georgia State University by R. Nave. and the discussion in "Traceable Temperatures" Second Ed. by Nicholas and White-See the discussions and definitions below for our rationale." Note, the Hyperphysics page is located at: http://hyperphysics.phy-astr.gsu.edu/hbase/thermo/temper.html#c1 From the page at [About Temperature Sensors entitled "What is Temperature" (http://www.temperatures.com/wit.html)]. Permission to reproduce this definition has been granted by its author and the webpage copyright holder.

20.2.2. Kelvin

Kelvin is a temperature scale with one defined temperature: 0 Kelvin is the lowest possible temperature, which is about -273,15°C. An increase of 1 Kelvin has the same size as an increase of 1°C.

The Kelvin (note it's not **degrees** Kelvin) temperature scale is used in science and engineering, most formulas containing temperature require Kelvin, unless it involves temperature differences, then one can use both Celsius and Kelvin as the difference between 10 and 20 Kelvin and between 10 and 20°Celsius are equal.

Conversion:

To Fahrenheit: Tk *9/5 -459,67

To Celsius: Tk - 273,15

20.2.3. Fahrenheit

This scale uses the lowest temperate that could be measured at the time when this scale was developed and the temperature of the human body. The first temperature is defined as 0°F the latter as 100°F. This places 0°C at 32°F and 100°C at 212°F. A difference of 1°F isn't the same size as a difference of 1°C.

Conversion:

To Kelvin: (Tf + 459,67) * 5/9

To Celsius: (Tf - 32) * 5/9

20.3. Methods of measuring temperatures

20.3.1. Non-electric methods

Are any of these non-electric methods useful for robotics?

The old fashioned way to measure temperature uses the expansion of mercury in a glass capillary tube. Newer thermometers use other liquids like alcohol. Other methods use two thin metal strips with different thermal expansion rates fused together and are called bimetal or bimetallic.

Yet another way uses the expansion of gas or liquid or a gas-liquid combination in a closed metal capillary tube system, called gas thermometers and filled systems. Other ways use the change of phase of some materials, such as paints and pastes whereby their reflectivity or color changes permanently when a specific temperature is reached. These are often called temperature or thermal paints and labels.

Another method uses the variation in reflectivity of certain liquid crystals with temperature. One of the original thermometers, named after Galileo, uses several precision weights in labeled small glass globes, of known volume, all placed in a vertical cylinder of liquid like water, sufficient in size to allow all the globes to fit and with sufficient width to enable the globes to move freely past one another. As the temperature of the assembly changes, the density of the water changes and the globes assume differing heights in the cylinder according to their density with the globe in the highest position indicating the temperature to be at or below its labeled value.

20.3.2. Electric or thermoelectric methods

Thermocouples

Thermocouples consist of a pair of wires of different metals connected together at both ends and insulated but kept thermally close to each other over their length. When they encounter a temperature gradient form one end to the other, an electrical current flows through the circuit due to the Seebeck Effect. If one of the wires is opened and the voltage between the open ends is measured with a suitably high impedance voltmeter, a voltage nearly equal to the average relative Seebeck coefficient times the temperature difference between the hot and cold ends will be measured.

Thermoelectric properties of many material combinations have been developed over the past hundred years or so and also have been standardized for several of them, more commonly known by their ISA letter designations, as standard. Their electrical outputs for cold end reference temperature at 32 Deg F and 0 deg C are widely published in ASTM Standard E230. Many sources on the web provide these tabular values including the National Institute for Science & Technology (NIST) in the USA.

Temperature sensitive resistors

Many electrically conductive materials have a resistance that varies with temperature -- they are temperature sensitive resistors.

There are two generic types of temperature sensitive resistors: Positive Temperature Coefficient (PTC) devices and Negative Temperature Coefficient (NTC) devices.

Usually the PTC devices, if made of metal, are termed Resistance Temperature Detectors or RTDs. Ones made from other materials are usually called Thermistors, short for Thermal Resistors.

RTDs are a highly developed technology worldwide and several international standards exist for the most widely used type, those made of pure Platinum. They are usually called Platinum Resistance Thermometers, or simple PRTs. Very special PRTs are used as interpolating devices in the International Temperature Scale of 1990 (ITS-90), these and similar devices sold commercially are often known as Standard Platinum Resistance Thermometers, or SPRTs.

Tables for the most commonly used RTD materials, Platinum, Nickel and Copper, can be found in several International and national standards as well as many locations on the Web.[4]

Thermistors are also very highly developed, but there are no uniform standards for their properties. Each supplier makes their own products and provide their own calibration tables. Some of these tables have been cataloged on the measurement database website.[5]

Many popular thermistors are manufactured using material with a B of 3300 K or a material with a B of 5133 K or some other material with an intermediate value.

Independent of the material, most thermistors are manufactured such that the resistance at room temperature (298.15 K = 25 °C) is one of these preferred value[6]s: 1 kΩ, 2 kΩ, 5 kΩ, 10 kΩ, 20 kΩ, 50 kΩ, 100 kΩ, 200 kΩ, 500 kΩ, 1 MΩ.

- PBASIC Programming/RCTIME[7] and ”RepRap: ExtruderIO” [8] have more details on measuring the RC time constant of a capacitor + thermistor circuit; then from that time calculating the temperature.
- Embedded Systems/Low-Voltage Circuits[9] and ”RepRap: Thermistor”[10] have more details on measuring the output voltage of a ”resistor divider” resistor + thermistor circuit; then from that voltage calculating the temperature.

Semiconductor

Semiconductors, such as diodes and transistors have properties that vary greatly with the temperature. This arises from the temperature sensitivity of the bandgap in such devices which is highly repeatable and very stable, especially in Silicon semiconductors.

Several Integrated Circuit (IC) manufacturers, such as Analog Devices, Dallas Semiconductors and National Semiconductors manufacture many variants of the basic diode temperature sensors with additional circuitry to scale the output for use with simple digital display meters.

Radiation Thermometer & Optical Pyrometer

The popular ear thermometer and low cost ”laser” thermometer of today are variants of the radiation thermometer (pyrometer) that began its development in the 19th Century. The devices are all based on the thermal radiation properties of literally everything in nature, whether it be gas, liquid or solid. The physical processes at work inside all materials includes emission of electromagnetic radiation that is a function of the temperature and optical properties of an object.

4 ”Temperatures.com: Resistance Temperature Detectors (RTDs)” ^{http://www.temperatures.com/rtds.html}
5 http://measurementdb.com/
6 https://en.wikipedia.org/wiki/%20preferred%20value
7 https://en.wikibooks.org/wiki/PBASIC%20Programming%2FRCTIME
8 http://reprap.org/wiki/ExtruderIO
9 https://en.wikibooks.org/wiki/Embedded%20Systems%2FLow-Voltage%20Circuits
10 http://reprap.org/wiki/thermistor

The physical emission of thermal radiation was first described correctly by Max Planck in the last year of the 19th Century. His theory led to the quantum revolution in Physics. It has stood the test of time as an accurate and reliable theory of thermal emission of radiation and enables instrument designers to develop instruments that can measure temperature without contacting the object that is being measured with remarkable levels of accuracy and repeatability. There are very few standards for such devices.[11]

20.4. Further reading

11 The only radiation thermometer standards of which we are aware are described on the webpage: INFRARED RADIATION THERMOMETER and THERMAL IMAGER STANDARDS `http://www.temperatures.com/stds-rts.html`

21. Pressure Sensors

The ability to measure pressure is very beneficial to the subject of robotics. The density of the atmosphere changes with altitude. If the air pressure is known then the altitude can then be established. If the robot contains fluids under pressure (for example, hydraulic systems), pressure sensors can be used to detect faults in the system. Pressure sensors can also be installed in a compressed tank of gas to measure the amount of gas left.

21.1. Measuring Pressure

21.1.1. Analogue Methods

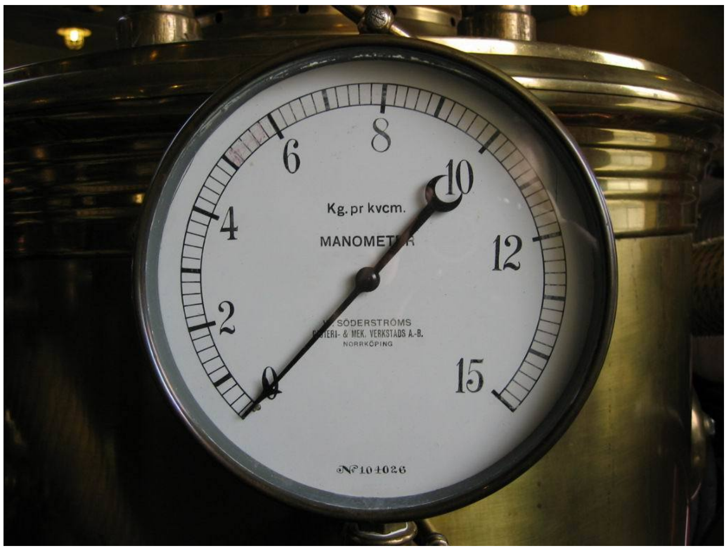


Figure 25 A Bourdon gauge

A Bourdon gauge uses a coiled tube which causes a rotation of an arm connected to the tube as it expands due to pressure increase. However, this is not particularly useful in Robotics unless you design a way to get voltage to vary with respect to needle displacement.

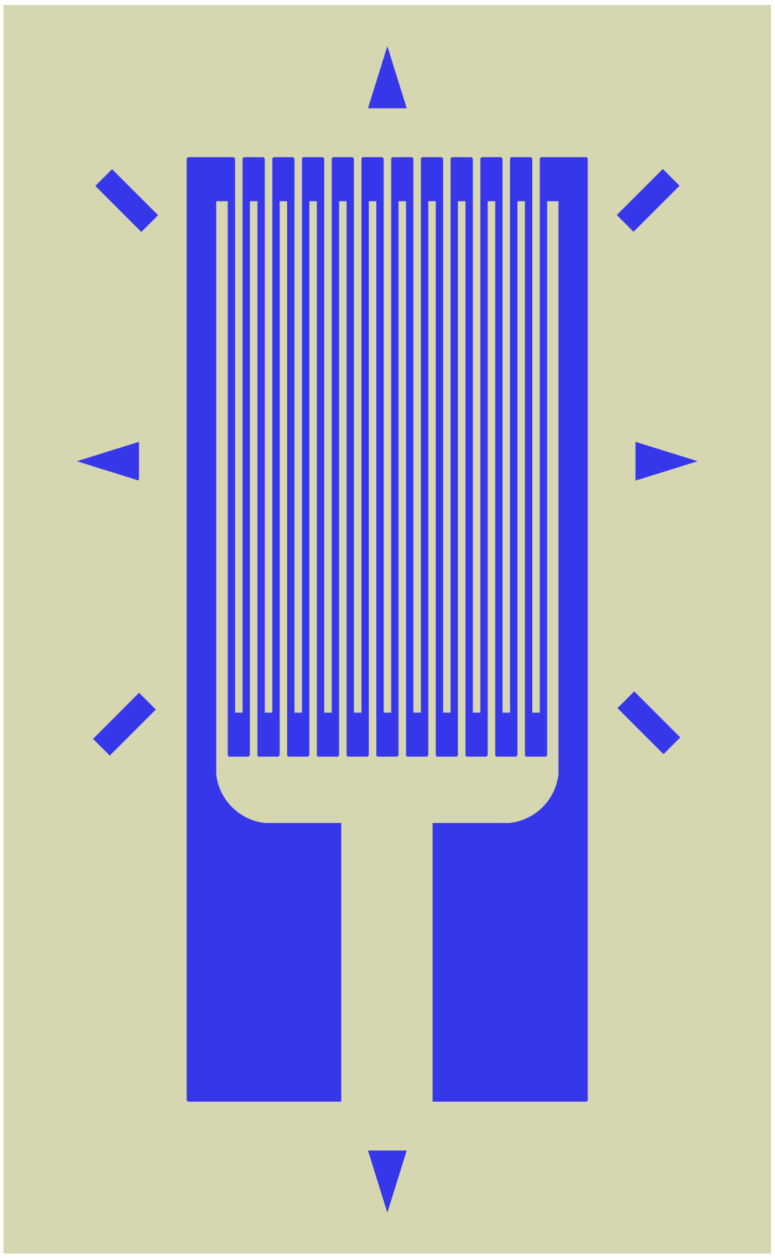


Figure 26 Typical foil strain gauge. The gauge is far more sensitive to strain in the vertical direction than in the horizontal direction. The markings outside the active area help to align the gauge during installation.

21.1.2. Electric Methods

- Resistive (Strain Gauge) - As the object is deformed, the foil is deformed, causing its electrical resistance to change. This resistance change, usually measured using a Wheatstone bridge, is related to the strain by the quantity known as the gauge factor. Foil strain gauges are only useful to detect large pressure changes and are not very precise over small changes.

- Capacitive - The deflection of the piston is often one half of a capacitor, so that when the piston moves, the capacitance of the device changes. This is a common way (with proper calibrations) to get a very precise, electronic reading from a manometer, and this configuration is called a capacitive manometer vacuum gauge.
- Piezoelectric/Piezoresistive - For measurements of small strain, semiconductor strain gauges, called piezoresistors, are often preferred over foil gauges. A semiconductor gauge usually has a larger gauge factor than a foil gauge. Semiconductor gauges tend to be more expensive, more sensitive to temperature changes, and are more fragile than foil gauges.

22. Ranging Sensors

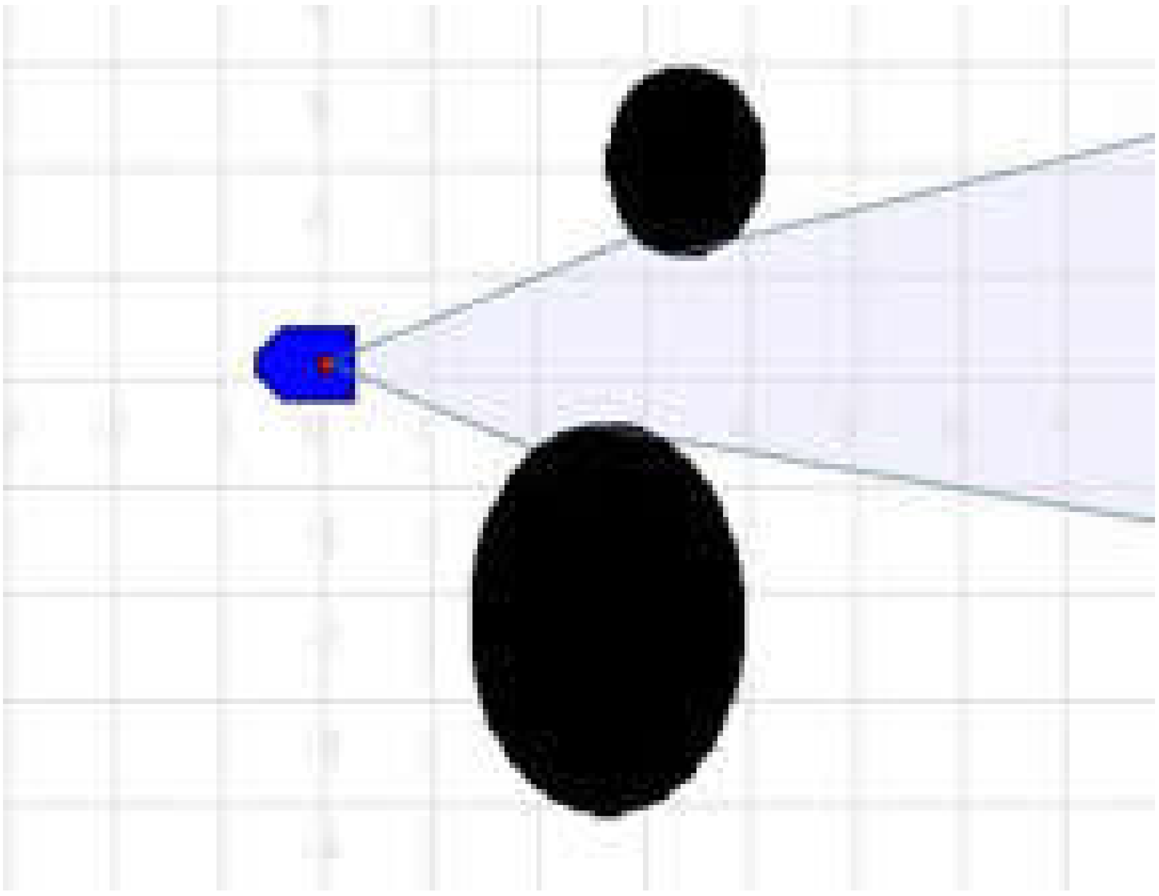


Figure 27 Player simulation of robot using LIDAR

22.1. Description

Ranging sensors include sensors that require no physical contact with the object being detected. They allow a robot to see an obstacle without actually having to come into contact with it. This can prevent possible entanglement, allow for better obstacle avoidance (over touch-feedback methods), and possibly allow software to distinguish between obstacles of different shapes and sizes. There are several methods used to allow a sensor to detect obstacles from a distance. Below are a few common methods ranging in complexity and capability from very basic to very intricate. The following examples are only made to give a general understanding of many common types of ranging and proximity sensors as they commonly apply to robotics. Many variances can exist within each type.

22.2. Sonic Sensors

Sonic sensors use sound waves[1], usually ultrasonic, through a medium as their means of detection. The medium is typically the atmosphere or a body of water. A pulse of sound is emitted from some source. One or more receivers then pick up the sound wave after it has bounced off any obstacles. This echo is then interpreted in various ways to obtain information about an obstacle.

22.2.1. Sonic Ranging

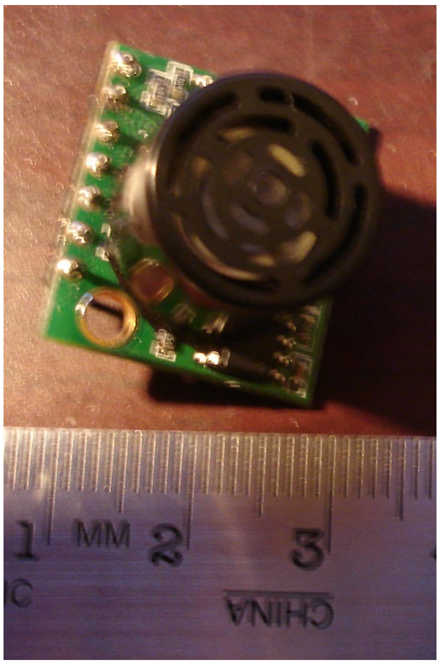


Figure 28 Single Ultrasonic Range Finder

1 http://en.wikipedia.org/wiki/Sound#Physics_of_sound

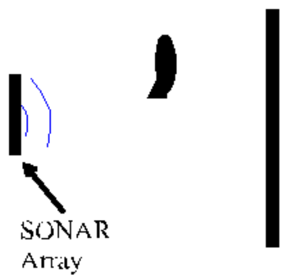


Figure 29 Animation of basic SONAR operation

Sonic ranging sensors, sometimes referred to as SONAR[2] send out a pulse of sound and wait for the echo to return. The time it takes for the echo to return is used to determine the distance to the obstacle. They are popular in hobby and research robotics due to their simplicity and relatively low cost. These sensors are generally limited to about a 6m range.[3][4] Divergence can be a problem because the sound wave spreads out rapidly as it moves away from the source. The sensor cannot determine where along this projected arc an obstacle was found. 'Ghost' echoes can cause problems as well when the sound wave bounces off multiple obstacles before returning.[5]

- Sonic Ranging
  - Pros
    - Cheap
    - Easy to Use
  - Cons
    - Resolution rapidly decreases with distance
    - Ghost echoes can give false readings
    - Physical properties of objects can give very different responses

22.2.2. Scanning Ranging

Sonic scanning ranging sensors are often (and hereafter) referred to as SONAR[6] or SONAR arrays. SONAR uses the same principle as sonic ranging but with more sophisticated detecting hardware. Typically a SONAR system will produce multiple pulses of sound and utilize multiple detectors (called an array) to calculate the distance and shape of objects with greater accuracy than one stationary sensor/emitter pair can achieve. However, SONAR performance can be poor in situations similar to those of other sonic ranging

2 `http://en.wikipedia.org/wiki/Sonar`
3 `http://acroname.com/robotics/parts/R287-SRF02.htmlhttp://acroname.com/robotics/parts/R287-SRF02.html`
4 `http://parallax.com/Store/Sensors/ObjectDetection/tabid/176/CategoryID/51/List/0/Level/a/ProductID/92/Default.aspx?SortField=ProductName,ProductName`
5 `http://www.societyofrobots.com/sensors_sonar.shtmlhttp://www.societyofrobots.com/sensors_sonar.shtml`
6 `http://en.wikipedia.org/wiki/Sonar`

systems. Dense obstacle distribution or complex surfaces can produce overwhelming ghost echoes or poor echo returns.

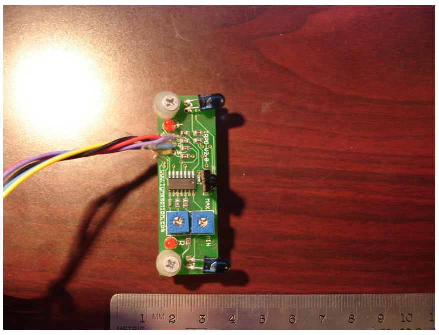


Figure 30 An IR proximity sensor with two emitters

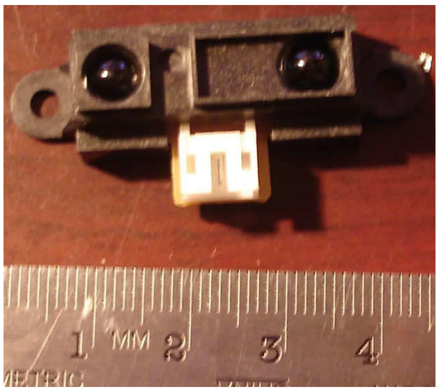


Figure 31 A Sharp GP2D120 infrared range sensor

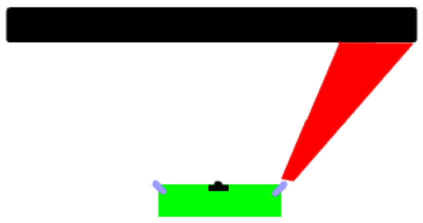


Figure 32 Animation of a two-emitter proximity sensor

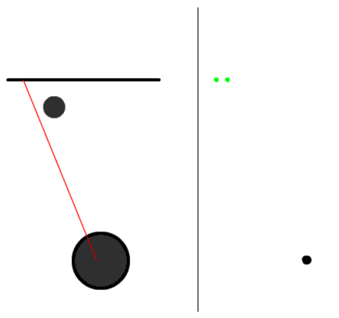


Figure 33 Animation of simple LIDAR

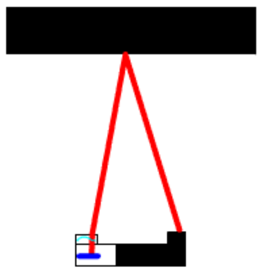


Figure 34 Animation of an infrared range sensor

22.3. Visible or Infrared Light-Based

Another very popular method uses projected light waves[7], usually infrared, to detect obstacles. This system projects a pulse of light and looks for the reflection. Properties of the reflected light are analyzed to determine characteristics about the object detected. Light has the advantages of traveling extremely fast, allowing for fast sensor response time, high resolution, and less error to account for. Light from this type of sensor is often formed into a narrow beam or many times a laser is used. This provides good resolution over large distances.

22.3.1. Basic Proximity

The simplest light-based obstacle sensor projects a light and looks for a reflection of a certain strength. If the reflection is strong enough, it can be inferred that an obstacle lies within a certain range of the sensor. Multiple light sources can be pulsed on in sequence to give some resolution to the sensor as in the figures.

7 http://en.wikipedia.org/wiki/Electromagnetic_radiation

22.3.2. Ranging Light-Based Sensors

Light-based ranging sensors use multiple methods for detecting obstacles and determining range. The simplest method uses the intensity of the reflected light from an obstacle to estimate distance. However, this can be significantly affected by the color/reflectivity of the obstacle and external light sources. A more common method is to use a beam of light projected at an angle and a strip of detectors spaced away from the emitter as in the animation to the right. The pictured Sharp sensor uses this method. This method is less affected by the color/reflectivity of the object and ambient light.

LIDAR[8], a more advanced method of range detection, uses a laser that is swept across the sensor's field of view. The reflected laser light is usually analyzed one of two ways. Units with longer ranges sometimes actually determine distance by measuring the time it takes for the laser pulse to return to the sensor. This requires extremely fast timing circuitry. Another method uses phase shift detection[9] to determine range by analyzing the incoming light and comparing it to a reference signal.

8 http://en.wikipedia.org/wiki/LIDAR

9 http://www.google.com/search?hl=en&rlz=1C1GGLS_en-USUS304US304&q=phase+shift+ranging&btnG=Search

23. Touch Sensors

23.1. Construction

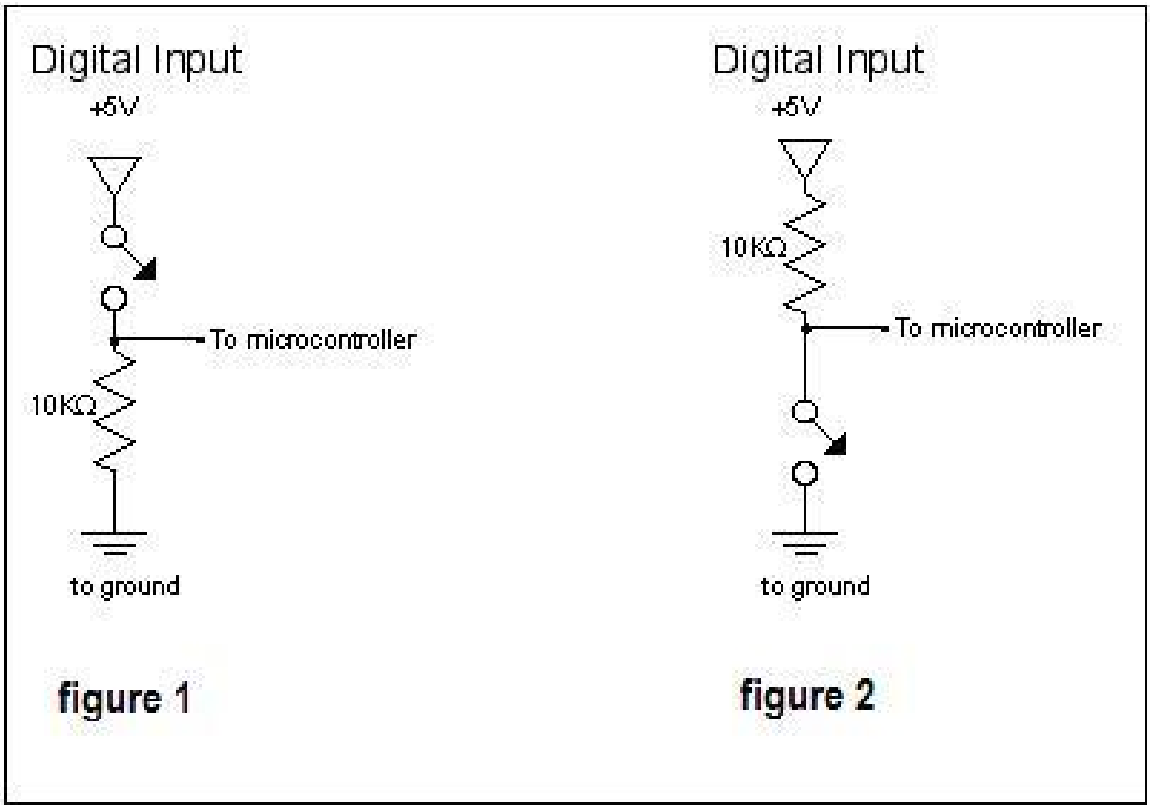


Figure 35 Schematic showing active *high* and active *low* configuration

A simple touch switch can be constructed from a momentary SPST (single-pole single-throw) switch. A wire extension can be added to the switch lever to act as an "antenna", "feeler", "whisker", etc.

Figure 36 A switch, with an *"antenna"* extension attached, and wired with a series resistor

The switch can be wired in such a way that when the state of the switch changes, either a "high" (a digital logic 1) is sent to the micro controller or a "low" (digital logic 0) is sent to the micro controller – depending on the application. The wiring diagram for active high, and active low are shown in figures 1 and 2, respectively.

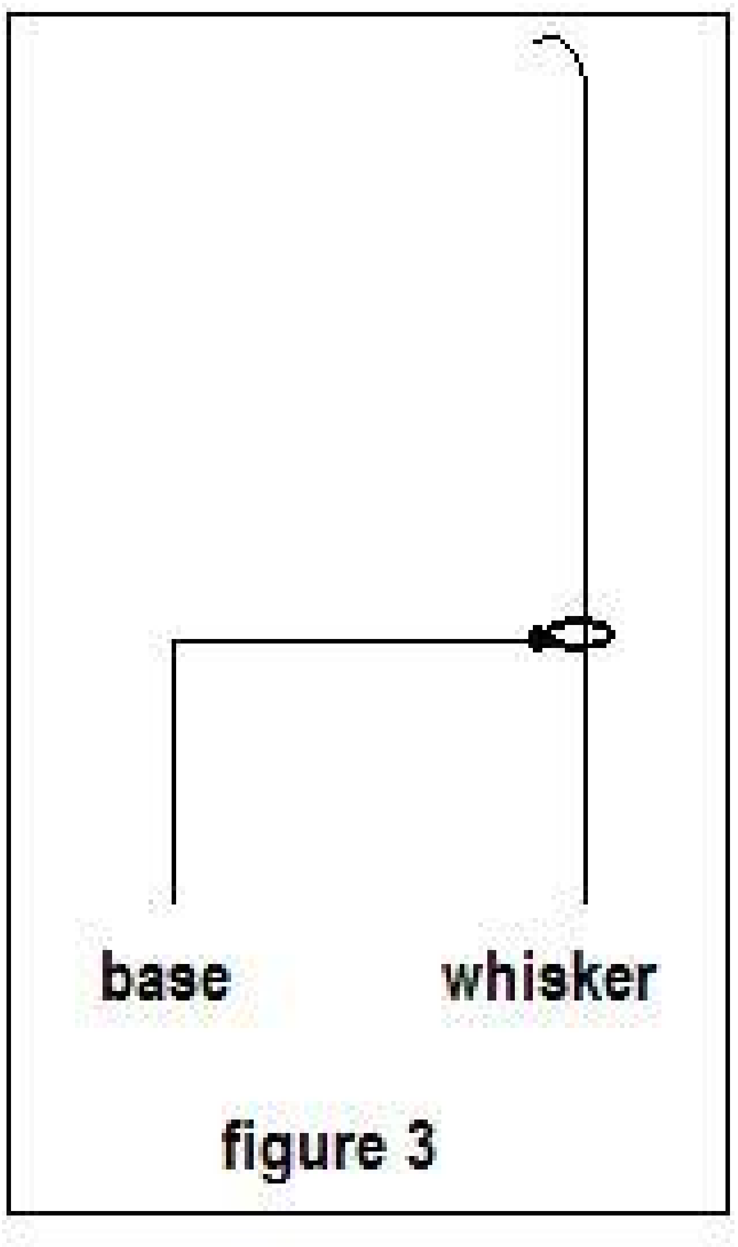


Figure 37 Simple *whisker* type switch

Another even simpler and cheaper method of touch sensor construction is with 2 pieces of wire. The first, the *base* , is L-shaped with a small loop at one end. The second wire, the *whisker* is passed through the loop on the end of the *base* wire. Under normal or *inactive* conditions, the whisker wire does *not* make contact with the loop – thus creating an *open circuit* . Only when the whisker wire *bumps* into an object, is the circuit complete. This circuit could be made to be either active-high or active-low using the schematics shown in figures 1 and 2, by simply replacing the switch contacts with the base and whisker wires.

Other types of touch sensors include simple momentary push-button switches.

23.2. Advantages

- Binary – interfaces easily with all microcontrollers
- Low cost
- Simple to construct and to connect
- Simple to operate – either the switch *is active* , or it is *not*
- Reliable

23.3. Disadvantages

- Tactile – object must be "touched" to know it exists
- Low range
- Less accurate for acquiring range data – you only know that there is an object there within the motion range of the switch – not exactly how far it is, or exactly where the sensed object's XY position is relative to the robots position

24. Feedback Sensors

The way in which actuators perform is determined by the combination of several factors. The robot's control system sends the actuator some sort of control signal, and various factors of the environment affect what happens for any given control signal. Feedback sensors are used to detect the actuator's output so that the control system can correct for external factors.

For example, many robots are propelled forward by wheels connected to DC motors. The control system specifies the amount of power that should be applied to the motors to make them turn, but their speed is also affected by the amount of load on the motor. In order for the robot to travel at a constant speed, it is necessary to have a sensor that determines how fast the robot is moving, and adjusts the power level to the motors as needed.

There are several standard kinds of feedback sensor that are used:

- /Encoders/[1]
- /Tachometers/[2]
- /Accelerometers/[3]
- /Rate Gyroscopes/[4]

24.1. further reading

- the OpenServo wiki[5] discusses one way to measure and control servo position, speed, voltage, and power consumption.

1 https://en.wikibooks.org/wiki/%2FEncoders%2F
2 https://en.wikibooks.org/wiki/%2FTachometers%2F
3 https://en.wikibooks.org/wiki/%2FAccelerometers%2F
4 https://en.wikibooks.org/wiki/%2FRate%20Gyroscopes%2F
5 http://openservo.com/

25. Communication Sensors

25.1. Data transmission channels

Being able to send data to and from your robot can be a very handy addition. There are 2 commonly used methods for wireless data transmission: IR (InfraRed) and RF (Radio Frequency). Both methods have their own advantages and disadvantages. Which to use depends on several factors.

25.1.1. IR

IR data transmission best known example is the TVs remote control. Using IR on a robot is quite easy and very cheap. The disadvantage of IR is that it's limited to line-of-sight transmission.Line-of-sight or the distance of operating can be increased by use of microwaves (transmission-receiver) systems

25.1.2. RF

RF are well known in radio controlled cars. RF is more expensive than IR, but doesn't have the line-of-sight limitation. These days there are complete RF "modems" available which can be connected to a robot without much (or even any) extra components. Although possible building your own RF communication modules isn't advisable. There are strict laws about which frequencies you can use and with how much power you can transmit.

25.2. Using IR

IR is not much more than modulated lightflashes. Since IR falls outside of the visual spectrum humans can't see these flashes without e.g. a digital camera (the CMOS image chip can see IR, on the screen those IR flashes appear bright white).

25.2.1. Remote Control

The great thing about IR remote controls is that you can use many of these directly to control your robot. Although there are several (very) differend IR remote control standards, there is one standard that is used by multiple manufacturers. This standard, called RC5, is very easy to use as many programming languages for microcontrollers have build in RC5 support. The hardware is limited to an integrated receiver module (e.g. TSOP1736), a capacitor and a resistor. [1][2]

25.3. Further reading

- Very often RF communications are ”noisy”, and require a higher-level protocol to tolerate the noise. Serial Programming/Forming Data Packets[3] describes some of the details.

1 De Vleeschauwer David. ”Phillips RC5 infrared remote protocol page” ^{http://users.pandora.be/davshomepage/rc5.htm} . Detailed information on how to encode and decode RC5 bits.

2 Michiel Niemeijer & Eric Toonen. ”The RC5 code” ^{http://users.skynet.be/luc.pauwels/luc/HP28/hardware/remote/} and ”Survey remote control transmitters CTV sets” ^{http://users.skynet.be/luc.pauwels/luc/HP28/hardware/remote/codes.html} . A list of standard RC5 command codes. (was: http: //web .archive.org/web/20060226134827/http: //193.23.168 .87/Mikrocontroller/Kohlert/rc5_codes.html ”The RC5 code”).

3 https://en.wikibooks.org/wiki/Serial%20Programming%2FForming%20Data%20Packets

26. Real World Sensors

This section covers the topics on sensor imperfection and how these components behave in the real world.

There is no such thing as a "distance sensor". period. Those components commonly called "distance sensor" or similar names measure something and extract distance information out of that. This extraction works pretty good in particular circumstances, but are worthless in many others. The key to successfully measuring e.g. a distance, is knowing, exactly, what your sensors measures and how external factors influence the measurement. This doesn't mean you just need to know how accurate the sensor is, it means you need to know what part of physics is used, and what the laws of physics say about that.

I'll cover some of the most commonly used sensors and what laws of physics apply to those. I'm not going very deep into explaining physics, there are better sources for that (a wiki physics book for example), just enough to give you the idea of what problems you may expect and where to look for a solution.

26.1. Light Based Sensors

26.1.1. Reflection: short range sensors

This type of sensor detects objects at a range up to a few centimeter. These sensors are quite simple. They consist of a light source, usually an IR diode which signal can be modulated and a light detector, which can be as simple as a light sensitive diode or transistor with an amplification circuit or a more complex IC complete with filters and TTL level outputs.

These sensors work by detecting the reflection of the emitted light. The range at which an object is detected depends on a number of properties of the object:

- reflectivity/color: how well does the object reflect IR-light? Every object has a color. A green object means that it reflects light with wavelengths that we interpret as the color green. This can be a pretty large range. IR is also a color. Like any other color, some objects reflect IR, and other objects absorb IR.
- surface smoothness: A very smooth surface (like a mirror) reflects more light than a rough surface. (For example, a photograph of a black billiards ball usually shows a white spot caused by w:specular reflection[1]).
- Angle: The more the surface is turned away from the sensor the more light is going to be reflected away from the sensor.

1 `https://en.wikipedia.org/wiki/specular%20reflection`

- Lightsources: Other lightsources like light bulbs or the sun emit IR light as well. Especially the sun can prevent an IR based sensor to operate.

26.1.2. Reflection: medium range sensors

Medium range sensors are a bit more complicated than short range sensors. These sensors consist of an IR emitting diode which signal is modulated, the receiver has a lens to focus the reflected light onto a light sensitive strip. Moving the sensor back and forward towards an object moves the reflection beam along the light sensitive strip. The resistance of the strip depends on where the light hits the strip.

Its range has the same limiting factors as short range sensors.

26.1.3. Reflection: Long range sensors

Long range sensors use the time a laser pulse takes to travel from the emitter to the object and back. There are several methods of measuring this time of flight, but most involve correlating the transmitted and received light pulse. By comparing the phase of these two pieces of data, a very accurate time value can be extracted. These sensors can operate over a wide range, typically between a few centimeters up to several kilometers.

Its range is also limited in the same way as the previous IR sensors. Another thing than can limit is haze, smoke and other particles on the air.

26.1.4. Camera

Cameras used in robotics are commonly built around a image Sensor[2]. These cameras are sensitive to IR-light and usually have a IR-filter in front of the lens. Cheap webcams may not contain such a filter, which makes them very sensitive to sunlight.

26.1.5. Stereo vision

These sensors consist of (at least) 2 cameras mounted some fixed distance from each other.

This is rarely used because solving the correspondence problem[3] is difficult.

See also Robotics: Sensors: Computer Vision[4].

2 https://en.wikipedia.org/wiki/Image%20sensor
3 https://en.wikipedia.org/wiki/correspondence%20problem
4 https://en.wikibooks.org/wiki/Robotics%3A%20Sensors%3A%20Computer%20Vision

26.2. Sound Based Sensors

26.2.1. What is sound?

Sound is in essence vibrations and pressure differences in the air. These vibrations are split into 3 groups by their frequency. The first group, called *infrasone* has a frequency below 20Hz. The second group, called *ultrasonic* , has a frequency above 20Khz and an upperbound of 2Mhz in air or 30Mhz in water. The last group is what is commonly called *sound* . This groups range lays between 20Hz and 20Khz and can be heard. Although only newborn babies can really hear all the way up to 20Khz. The older you get the less frequencies you can hear.

Most sensors use ultrasonic sound, usually around 40Khz. Such a signal can't be heard, while it's still easy to use (generate, detect,...).

26.2.2. Doppler effect

If both the source and the receiver are motionless relative to each other, the receiver will hear the same frequency as the source emitted. However if one or both of them move relative to the other, the receiver will detect a different frequency. This change in frequency is called the Doppler Effect[5]. Most people know this effect from the sirens of passing police cars or ambulances. When they pass you'll hear one sound as they approach and a somewhat different sound as they move away.

Calculating what frequency the receiver will hear is quite easy:

$f_r = f\frac{c+v_r}{c-v_s}$

With:

f_r = The frequency the receiver hears

f = the frequency the source emits

c = the speed of sound

v_r = the speed of the receiver

v_s = the speed of the source

26.2.3. Speed of sound

The speed of sound depends on the medium it traverse through and its temperature. For air this is approximately 330m/s at 0°C. Of course most of the time the temperature is a bit higher than this. Calculating the actual speed is fairly easy:

$c = c_0\sqrt{\frac{T}{T_0}}$

5 `http://en.wikipedia.org/wiki/Doppler_effect`

with:

c = the actual speed at the current temperature.

c_0 = the speed of sound at 0°C: 330m/s.

T = the current temperature in Kelvin.

T_0 = 273,15 K (this is 0°C in Kelvin)

26.2.4. Reflection

26.2.5. Resonance

26.2.6. Diffraction

26.3. Ultrasonic Distance sensors

These sensors are pretty simple. In theory that is. In practice these sensors can be a real pain in the pinky. In this section I'll cover some of the troubles you may run into when trying to get them to work.

Ultrasonic distance sensors consist of 3 major parts: A transmitter, a receiver and a timer. To measure a distance the timer triggers the transmitter which emits a series of pulses, then the timer waits until the receiver detects the reflection of the pulses and stops the timer. The time measured is then divided by 2 and multiplied with the speed of sound. The result is the distance between the sensor and the object in front of it. The transmitter send out a stream of pulses on a carrier frequency. The maximum frequency humans can hear is about 20 KHz. A frequency higher than that is picked to avoid annoying humans with the constant beep -- 40KHz is a common value.

The receiver triggers when it receives a signal with that particular frequency. This is not necessary the signal the transmitter sent. If more than one ultrasonic sensor with the same carrier frequency are used, they can detect each others signals.

Sound doesn't move in a straight line, but rather as a 3D expanding wave. When the wave reaches an object part of it bounces back and moves again as a 3D expanding wave in the opposite direction. Such a wave can easily bounce multiple times before disappearing. So it is very possible that you receive pulses that have travel a much larger trajectory than just to and back from the object in front of the sensor. While some part of this problem can be solved by letting the sensor wait some time before starting another measurement, other situation can produce incorrect measurements which are fairly tough to correct. For example moving through a doorway can fail because the sensors emitted pulses bounce from the walls back to the sensor and so giving a measurement that indicates an object in front of the sensor. One way of correcting this is using another sensor, for example a IR distance sensor to see if there really is an object. However such solution pose another problem: which sensor to believe? 3 sensors allow you to go with the majority, but then things become quite complicated in constructing and interfacing such systems, not to mention what it does to the power consumption.

26.3.1. Distance Formula

The formula for calculating distance from a sonar pulse looks like:

$Distance = 343m/s * \frac{ElapsedTime}{2}$

343 m/s is the speed of sound, and we need to divide the time by 2 because the sound travels out and back.

26.3.2. Availability & Range

Sonar sensors are widely available and relatively inexpensive, ranging from $15 to $40 depending on the desired range. On average the maximum range of a midlevel sonar sensor will be between 4 and 6 meters. Unlike infrared or laser sensors, sonar sensors also have a minimum sensing distance as well. This is due to the fact that the distance measurements are based on the speed of sound, and over very short ranges the sound travels out and back more quickly than the circuitry can respond. This minimum distance will vary by sensors, but is typically around 2 to 5 centimeters. Also unlike infrared sensors, sonar sensors don't have a perfect "cone" of vision. Because sound propagates as a 3D pressure wave, the sensor actually has a range that resembles a sinc function wrapped around a curve.

26.3.3. Potential Problems

Sonar sensors work very well for collision avoidance in large areas, but they do have some limitations. Because sound propagates out as a 3D pressure wave and echoes, your robot may see things that are not really in its way. For instance, near angled walls the sound waves may bounce several times before returning to the sonar receiver. This makes it difficult for the robot to know which echo is actually the correct distance to the nearest obstacle.

Similar to this is the problem of having multiple sonar sensors operating the in the same area. If the frequencies of nearby sonar sensors are too similar they may cause false readings since the sensors have no method besides frequency to distinguish pulses it sent out from those other sensors send out.

Another common problem is the difference in absorbency and reflection of different materials. If you shoot a sonar pulse at a cloth covered wall (i.e. cubicle), it is likely that the cloth will absorb a significant amount of the acoustic energy and that the robot will not see the wall at all. On the opposite end of the spectrum, a floor with very high acoustic reflection may register as an obstacle when it is really a clear plane.

26.4. Magnetism Based Sensors

26.4.1. Compass sensors

These sensors are used to measure the orientation of the robot relative to the magnetic north. It is important to remember that the magnetic north is not exactly the same as the geographical north. They differ several degrees.

The magnetic field of Earth is quite weak. This makes that these sensors will not operate well along other magnetic fields. E.g. speakers would mess up the reading. If you use these sensors it is best to mount them as far away from your motors as possible. While you can't shield them without making them useless, paying attentions to where you mount them can make a considerable difference in reliability.

26.5. Other distance sensors

- *(How do those "stud sensors" detect the lumber in the walls behind the sheetrock?)*

26.6. Further reading

- the sensor wiki `http://sensorwiki.org/`
- ultrasonics[6]
- Robot Sensors[7]
- "Migrating from x86 to PowerPC, Part 9: Sensors, sensors, sensors!"[8] recommends using a optocoupler for isolation for all sensors.

6 `http://david.carybros.com/html/ultrasonic.html`
7 `http://www.andrew.cmu.edu/user/rjg/websensors/robot_sensors2.html`
8 `http://www.ibm.com/developerworks/power/library/pa-migrate9/index.html?S_TACT=105AGX16&S_CMP=EDU`

27. Digital image Acquisition

27.1. Image Sensors

There are 2 types of image sensors commonly used to digitally acquire an image, CCD and CMOS. While both have similar image quality, their core functionality and other features greatly differ.[3]

27.1.1. CCD

A CCD, or Charge-Coupled Device, is an older technology based on an analog system. A photoelectric surface that coats the CCD creates an electric charger when light hits it, and the charge is then transferred and stored in a capacitive bin that sits below the surface of each pixel.[2] The CCD then functions like a shift register, and transfers the charge in the bin below each pixel one spot over by applying a voltage in a cascading motion across the surface. The charge that reaches the edge of the CCD is transferred to an analog to digital converter, which turns the charge into a digital value for each pixel. This process relatively slow because of the way it has to shift each pixel to the edge of the CCD to then turn it into digital information.[4]

27.1.2. CMOS

A CMOS image sensor is a type of Active-Pixel Sensor (APS) constructed of Complementary metal–oxide–semiconductors. CMOS sensors can be much faster at generating a digital image, and consume less power than a CCD. They can also be larger than a CCD, allowing for higher resolution images, and can be manufactured through cheaper methods than a CCD. Each pixel in a CMOS sensor contains a photodetector and an amplifier.[4] The simplest type of CMOS image sensor is the 3T model, where each pixel is made up of 3 transistors and a photodiode. The transistor Mrst is used to clear the value of the pixel and reset it to acquire a new image. The Msf transistor buffers and amplifies the value from the photodiode until the pixel can be read and is reset. The Msel transistor is the pixel select transistor that only outputs the pixel's value to the bus when the device is reading the row it is in. In this model the data is gathered in parallel via a shared bus, where all pixels in a column share the same bus, and then the data is sent down the bus one row at a time. This method is faster at shifting the charge value to the digital converter. There are other variations of the CMOS sensor which help reduce image lag and noise. Image lag is created where some of the previous image remains in the current one. This is often caused by a pixel not getting fully reset, so some of the charge from the previous image still exists. Image noise is a measure of how accurately the amount of light that hits the pixel is measured.It is very important tool for robotics.

27.2. Color Images

The 2 types of image sensors do not natively measure color; they simply convert the amount of light (regardless of color) to a digital value.[1] There are several different ways to gather color data. The 2 most common are to use a color filtering array, the Foveon X3 specialized sensor and using a trichroic prism and 3 image sensors.

27.2.1. Color Filtering Array

The most common method is to use a color filtering array. The most common type of filter used is the Bayer filter, developed by Kodak researcher Bryce Bayer in 1976. A color filtering array filters the light coming into each pixel so that the pixel only detects one of the primary colors. The full color image can later be recreated by adding the colors from each pixel together to create a full color image.[1] The Bayer filter uses a pattern of 50% green, 25% red and 25% blue to match the sensitivity of the human eye to the 3 primary colors. The Bayer filter repeats over 4 pixels.[1] Because the images have to be reconstructed and you only know one of the colors in each pixel, some of the image fidelity is lost in the process of image reconstruction called demosaicing. Edges of objects in the image can often be jagged and have non uniform color along the edges. In addition to the Bayer filter, there are many other filter patterns that can achieve the same result.[1] The main problem with using a filter is that it reduces the amount of light that reaches each photodetector, thus reducing each pixel's sensitivity to light. This can cause problems in low-light situations because the photodetectors will not receive enough light to produce a charge, resulting in large amounts of noise. To help reduce this effect, there is another type of filter in use that has some pixels that are not filtered. These Panchromatic filters mimic the human eye, which has detectors of color and detectors of light and dark. These do much better in low light, but require a much larger area to mimic the pattern than a traditional Bayer filter. This causes some loss in fidelity

27.3. Image Transfer

The two most common methods for connecting cameras to a robot are USB and FireWire (IEEE 1394). When Apple developed FireWire they had in mind that it would be used to transfer audio and video. This resulted in a greater effective speed and higher sustained data transfer rates than USB, which are needed for audio and video streaming. FireWire also has the benefits of being capable of supplying more power to devices than USB can. FireWire can also function without a computer host. Devices can communicate with each other over FireWire without a computer to mediate.[5]

27.4. References

1. Color Filter Array[1]

1 http://www.dpreview.com/learn/?/key=color+filter+array

2. Charge-Coupled devices (CCDs)[2]
3. How Digital Cameras Work[3]
4. What is the difference between CCD and CMOS image sensors in a digital camera?[4]
5. How FireWire Works[5]

----

2 http://www.physics.pdx.edu/~d4eb/ccd/index.htm
3 http://electronics.howstuffworks.com/digital-camera.htm
4 http://electronics.howstuffworks.com/question362.htm
5 http://www.howstuffworks.com/firewire.htm

28. Navigation

28.1. What is Navigation?

Navigation comprises everything a robot needs to get from point A to point B as efficient as possible without bumping into furniture, walls or people.

This problem breaks down in several sub-problems:

- It needs to know where (Localization)[1] it is.
- It needs to detect and avoid obstacles (Collision Avoidance)[2].
- It needs memory of its surroundings (Mapping)[3].
- It has to be able to plan a route (Trajectory Planning)[4] to point B.
- It has to be able to explore (Exploration)[5] new terrain.

28.2. Localization

Localization involves one question: Where is the robot now? Or, robo-centrically, where am I, keeping in mind that "here" is relative to some landmark (usually the point of origin or the destination) and that you are never lost if you don't care where you are.

Although a simple question, answering it isn't easy, as the answer is different depending on the characteristics of your robot. Localization techniques that work fine for one robot in one environment may not work well or at all in another environment. For example, localizations which work well in an outdoors environment may be useless indoors.

All localization techniques generally provide two basic pieces of information:

- what is the current location of the robot in some environment?
- what is the robot's current orientation in that same environment?

The first could be in the form of Cartesian or Polar coordinates or geographic latitude and longitude. The latter could be a combination of roll, pitch and yaw or a compass heading.

1 https://en.wikibooks.org/wiki/%2FLocalization
2 https://en.wikibooks.org/wiki/%2FCollision%20Avoidance
3 https://en.wikibooks.org/wiki/%2FMapping
4 https://en.wikibooks.org/wiki/%2FTrajectory%20Planning
5 https://en.wikibooks.org/wiki/%2FExploration

28.3. Current Location

The current location of a robot can be determined in several very different ways:

28.4. Dead Reckoning

Dead reckoning uses odometry to measure how far the robot moves. Trigonometry and the equations of kinematics are all that is needed to calculate its new position.

This method has 2 requirements:

- It needs a way to determine its initial location.
- Its accuracy usually decreases over time as every measurement has an error and errors accumulate.

Dead reckoning is often used with other methods to improve the overall accuracy.

28.5. Least Mean Squares

There are numerous solutions to the localization robotics problem. These range from simple Dead Reckoning methods to advanced algorithms with expensive radar or vision system. The most important factor is picking an algorithm to find the robotic location is the availability of accurate relative and global position data. For simple systems with basic relative position sensors and some form of a global position sensor, the most practical and easiest to implement localization method is that of Least Mean Squares.

See Least Squares: `http://en.wikipedia.org/wiki/Least_squares` for more information on using the method of Least Squares to find solutions for over determined systems.

To look at the Least Mean Square (LMS) algorithm in a general sense first, it is important to look at the general principles that govern it. The goal of all localization methods is to minimize the error between where the robot is and where the robot should be. The algorithm must be able to follow a preset track or find the least error in the robot location (Adaptive Algorithms and Structures, p. 97). These goals for robot localization are applied to the LMS algorithm by adaptively adjusting weights to minimize the error between the actual function and the function generated by the LMS algorithm. As a subset of the Steepest Descent Algorithm (`http://en.wikipedia.org/wiki/Gradient_descent`), this method was created to minimize the error by estimating the gradient (`http://en.wikipedia.org/wiki/Gradient`) and is known as the simplest localization method with global position output.

28.5.1. Derivation of LMS Algorithm

The LMS algorithm can be applied to two cases:

- Parallel multiple inputs

• Series single input

Figure 38

Figure 1: Linear Combiner Configuration [1]

Figure 39

Figure 2: Curve Fit to Data to Minimize Error [3]

Figure 40

Figure 3: Transversal Combiner Configuration [1]

In both cases, the gradient is calculated as

$\varepsilon_k = d_k - X^T{}_k W_k$

where ‘Y_k’ is the desired output, ‘$X^T{}_k$’ is the actual input, and ‘W_k’ for the kth input element. Next, the gradient estimate is calculated as

Now that we have the gradient estimate, the weights can be updated by

$W_{k+1} = W_k - \mu \mathrm{del}_k = W_k + 2\mu\varepsilon_k X_k$

to minimize the error through iterative change. ‘μ ’ is the gain constant for training the algorithm. The use of training is to adjust how fast the error is corrected for by the weight update function [1]. If the training constant is too high, the system will oscillate and never converge to the minimal error value.

To illustrate the process other than in mathematical format, the LMS algorithm can be demonstrated through the use of pseudo code. The purpose of the code below is for illustration purposes only. This example is intended to provide a step-by-step visualization tool for beginning the coding process.

28.5.2. Procedure LMS

Initialize the Weights repeat choose training pair (x,d) for all k do

```
    $y_k = W_k X_k$
```

end for all k do

```
    $\varepsilon_k = Y_k - d_k$
```

end for all k do

```
    $W_k(j+1) = W_k(j) + \mu\varepsilon_k X_k$
```

end until termination condition reached

28.5.3. Various Variations

Modified LMS Method

For problems with sequentially arriving data, the Modified LS method can be used to add or delete rows of data. In various time-series problems, a moving 'window' can be employed to evaluate data over a predefined timeframe. This method is useful in LMS problems that arise in statistics, optimization, and signal processing [2].

Constrained LMS Method

For curve and surface fitting application problems, the Constrained LMS method is very useful. Such problems require satisfying linear equations systems with various unknowns. This method only provides a useful solution if and only if each input can be scaled by some magnitude and still remains a constant [2].

Direct Methods for Sparse problems

A sparse LMS problem involves solving the LMS problem when the matrix has 'relatively few' nonzero elements. "J. H. Wilkinson defined a sparse matrix to be 'any matrix with enough zeros that it pays to take advantage of them'" [2]. A more precise definition is more difficult to provide. Such problem usually involve large to enormous size and involve numerous unknowns. The following are examples of problem that can be classified as sparse problems: pattern matching, cluster analysis, and surface fitting [2].

Iterative Methods for LMS problems

A second method for solving sparse problems (in addition to the Direct Method mentioned above) is the Iterative Method. In general, the Iterative Method is used for analysis of under-determined, sparse problems to compute a minimum solution based on the norm of the input data [2].

Nonlinear LMS Method

For problems with nonlinear systems, the Nonlinear LMS method is useful for iteratively calculating the relative LMS solution to the problem. This method can be used for unconstrained optimization and various survey methods [2].

28.5.4. Additional Sources for Application and Theory

For further reference and additional reading, the following resources can be used for a more extensive overview of the subject.

- Adaptive Antenna Control by LMS[6]

6 `http://www.latticesemi.com/products/intellectualproperty/referencedesigns/lmsadaptivefilter.cfm`

- LMS Estimation Techniques[7]
- Least Squares[8]
- Least Squares Filter[9]

28.5.5. References

- Adaptive Signal Processing [1]
- Numerical Methods for LS Problem book [2]
- Least Squares Estimation[10] [3]

28.6. GPS Global Positioning System

On large terrains GPS can provide more or less accurate coordinates, dead reckoning and the data from its other sensors can fill in the gaps. However on small terrains or indoors the GPS's inaccuracy becomes a problem and the dead reckoning and sensor data becomes the dominant factor in determining the location.

28.7. GPS-like Systems

Although GPS isn't very useful indoors, similar techniques as those used in GPS can be used indoors. All a robot needs is to measure it's distance to 3 fixed beacons. Each of these distances describe a circle with the beacon in its center. 3 circles will only intersect in one point.

28.8. Line following

Probably the easiest way: Draw a line on the ground and use IR reflection sensors to follow it. Useful as a track or to mark the edges of the robot's work area.

This can also be done by putting an electric cable in the ground and sending a modulated signal through it. The robot can detect this cable with magnetic sensors (Hall-sensors or coils).

The complex versions of line-following involves using sensors like vision (camera) which helps reducing overall cost of sensors and implementation and also provides versatility to detect lines of various colours. It allows a great scope for autonomy in the system too.

7 http://www.aiaccess.net/English/Glossaries/GlosMod/e_gm_least_squares_estimation.htm
8 http://en.wikipedia.org/wiki/Least_squares
9 http://en.wikipedia.org/wiki/Least_mean_squares_filter
10 http://www.aiaccess.net/English/Glossaries/GlosMod/e_gm_least_squares_estimation.htm

28.9. Landmarks

Placing landmarks is another way to help your robot to navigate through its environment. Landmarks can be active beacons (IR or sound) or passive (reflectors). Using bar code scanners is another possibility.

A special trick, if the environment is known beforehand, is to use collision detection (i.e. sensor bumpers or similar) with known objects. Together with dead reckoning the precision can be extraordinary even when using cheap equipment.

In new environments landmarks often need to be determined by the robot itself. Through the use of sensor data collected by laser, sonar or camera, specific landmark types (e.g. walls, doors, corridors) can be recognised and used for localization.

28.10. Current Heading

Determining a robot's heading can be done with a compass sensor. These sensors usually consist of 2 magnetic field sensors placed at a 90° angle.

These sensors measure the earth's magnetic field and can be influenced by other magnetic fields. Speakers, transformers, electric cables and refrigerator magnets can reduce the accuracy of the compass. Compass sensor modules can have built-in filtering for the magnetic fields of AC.

29. Collision Avoidance

Collision avoidance, in comparison to collision detection, is what is done after a possible collision is detected. Possible collisions will depend on the shape and size of both the object and robot as well as the position and motion projection of the object and the robot. When making an attempt to avoid an obstacle, there are many decisions to face. The decisions available depend on the available paths the robot can take to avoid the obstruction. These paths can either be determined upon the robot's realization of the obstruction or, in case all of the obstructions' positions are known, once the map is loaded. Finding the "correct" path to use can be troublesome depending on which algorithm is used and the granularity associated with it.

29.1. Basic Idea

If there is an object in the robot's way, avoid it.

Essentially, this is the basic idea in collision avoidance. However, this simple matter to humans is incredibly complex in the computer world. Which algorithm is best in avoiding obstacles? Which is least complex, most efficient? Which algorithm can utilize the robot's full capabilities effectively? All of these are questions one should ask when searching for a valuable collision avoidance algorithm.

29.2. Example Algorithm

Figure 41 Example algorithm being put to use using Player/Stage[a]

[a] `http://en.wikipedia.org/wiki/Player/Stage_Project`

In the figure to the right, a simple "tank-drive" robot is presented making it's way to three pre-determined waypoints. The robot's algorithm is a dignified version of the simple one-liner given at the beginning of this section:

1. Turn to the nearest waypoint if there is one left.
   a) If there is no waypoint left, stop.
   b) Move toward the nearest waypoint.
2. If there is something in the way:
   a) Go around it.
   b) After every 1.5 body lengths, go to step 1.

3. If the robot has reached the waypoint:
    a) Remove the reached waypoint from the list.
    b) Go to step 1.

Albeit this algorithm works well for an over-simplified map where all obstacles are simple geometric shapes, in more complex scenarios it may not fare as well. For instance, nestling a waypoint in the middle of a U-shaped obstacle would cause this algorithm to never end.

29.3. Major Problem

With robots having greater and greater importance in our everyday lives, it is no wonder that collision avoidance (particularly dealing with human safety) is becoming an increasingly greater concern. Such concerns are not to be taken lightly. There is already one death associated with roboticide[1].

In addition to safety, other concerns can be generated:

- Collision Avoidance Algorithmic Complexity
- Computer Power
- Sensor Accuracy / Throughput
- Sensor Data Interpretation

Of course, as the algorithmic complexity of the collision avoidance algorithm increases, so does the required computer power. Some researchers are even finding it beneficial to use multiple processors in parallel (parallel processing) to "divide and conquer" such problems[3].

Sensor accuracy and throughput is becoming less a threat to successful collision avoidance. However, this doesn't mean it is non-existent. During the process of engineering a collision-avoiding robot, trade-offs can be made that could potentially compromise the effectiveness of the sensors in how the robot behaves.

29.4. Detection vs. Avoidance

Collision detection is simply the act of surveying the known vicinity of the robot and detecting the presence (or absence) of a possible collision. In order for a robot to avoid a collision, a collision must first be detected. Without collision detection, it doesn't seem reasonable to have collision avoidance because there wouldn't be anything to avoid (in the robot's scope).

Collision avoidance is the plan for action the robot takes to evade the oncoming collision. As previously stated, there is no need for collision avoidance if there are no collisions to avoid.

1 Smart software helps robots dodge collisions[2]. . Retrieved

3 A "Divide and Conquer" Strategy based on Situations to achieve Reactive Collision Avoidance in Troublesome Scenarios[4]. . Retrieved

29.5. Collision Avoidance Algorithms

29.5.1. Quad Trees

The use of quad trees[5] allows for a simple collision avoidance algorithm. An approximation technique used here uses a type of data structure called a quad tree. A quad tree is similar to that of a binary tree except that a quad tree has four child nodes for each parent instead of just two.

Each child node is represented by a square of pixels in a "top-down" image of the robot's surrounding landscape. Each square simply acts as a binary answer to the question, "Is there any possibility for a collision inside of this square?" If the answer is yes, then the square is avoided, otherwise, the square is noted for safe travel. An example of this is given in the figure below.

Figure 42 Approximation of a triangle using squares.

Advantages

An advantage to using this algorithm is that retreating can be done fairly easily – by climbing back up the tree and trying a different approach. Of course, in a case where all of the obstacles have been found, a minimal path can easily be created using simple distance algorithms.

Disadvantages

A disadvantage to using this algorithm is that, due to the granularity of each square of pixels, it is possible for the obstacles to be close enough such that a robot wouldn't notice that it could actually fit through a gap.

29.6. References

- Player/Stage Wiki Page[7]

29.7. Exploration

Exploration means having the robot filling in parts of the map. This requires sensors that detect the obstacles and localization to know where the robot is. Fusing this data allows the robot to fill in where the obstacle is on the map.

5 A Comprehensive Robot Collision Avoidance Scheme by Two-Dimensional Geometric Modeling[6]. . Retrieved

7 `http://en.wikipedia.org/wiki/Player/Stage_Project`

29.8. Mapping

To find routes efficiently a robot has to know it's surroundings. Maps can be very handy for this. Either the robot has the memory and tools for it and draws its own map, gets its map in advance or get a partial map (e.g. only walls) and adds the other obstacles to it itself.

- Getting the map in advance works fine in a static environment.
- The partial map is a more efficient approach as it already contains all the static parts of the *world* .
- Working from scratch is the best approach when the robot is often placed in new environments.

Drawing a map has 2 major issues:

- Not all elements in the surroundings are stationary. Most sensors are too limited to measure directly if the obstacle is a wall, a chair or a person. A method for solving this issue is to use a "score" for each obstacle measured. Add to the score if the object is detected again when the robot passes the same location.
- Localization isn't perfect. Small accumulating errors can make it hard to determine the exact location of the robot and therefor can make it hard to find where a detected obstacle has to be noted on the map. One solution to this problem is Simultaneous Localization and Mapping (SLAM), which corrects the location of detected objects based on the estimated error of the robots location.

A third minor issue is memory. On a robot build without a computer memory is limited and storing an entire map can be next to impossible. If you have the memory the map can be stored in different ways:

- 2D array: Easy and fast to access, very wasteful with memory.
- linked list: slower to use, more efficient with memory.
- Quadtree: faster than linked list, much harder to implement, very efficient with memory.

Implementing these structures on a PC is reasonably easy, however constructing them in a limited environment as a microcontroller is a serious challenge.

29.9. Trajectory Planning

29.10. Trajectory Planning

Trajectory planning is moving from point A to point B while avoiding collisions over time. This can be computed in both discrete and continuous methods. Trajectory planning is a major area in robotics as it gives way to autonomous vehicles.

Trajectory planning is sometimes referred to as motion planning and erroneously as path planning. Trajectory planning is distinct from path planning in that it is parametrized by time. Essentially trajectory planning encompasses path planning in addition to planning how to move based on velocity, time, and kinematics.

29.10.1. Problem Constraints

Holonomicity

Holonomicity is the relationship between the controllable degrees of freedom of the robot and the total degrees of freedom of the robot. If the number of controllable degrees of freedom are greater than or equal to the total degrees of freedom a robot is said to be holonomic. By using a holonomic robot many movements are much easier to make and returning to a past pose is much easier.

A car would be non-holonomic, as it has no way to move laterally. This makes certain movements, such as parallel parking, difficult. An example of a holonomic vehicle would be one using mecanum wheels, such as the new Segway RMP.[8]

Dynamic Environments

In dynamic environments, such as the real world, many possible collision objects are not stationary. This makes trajectory planning more difficult as time is constantly changing and objects are moving. A robot cannot simply move backwards in time as it might simply back away from a stationary collision. In addition to this many choices are completely irreversible due to terrain, such as moving off of a cliff.

8 Phillip Torrone. ”Segway's new RMP” ˆ{`http://blog.makezine.com/archive/2008/04/segways_new_rmp.html`} . Make magazine. 2008.

29.10.2. Concepts

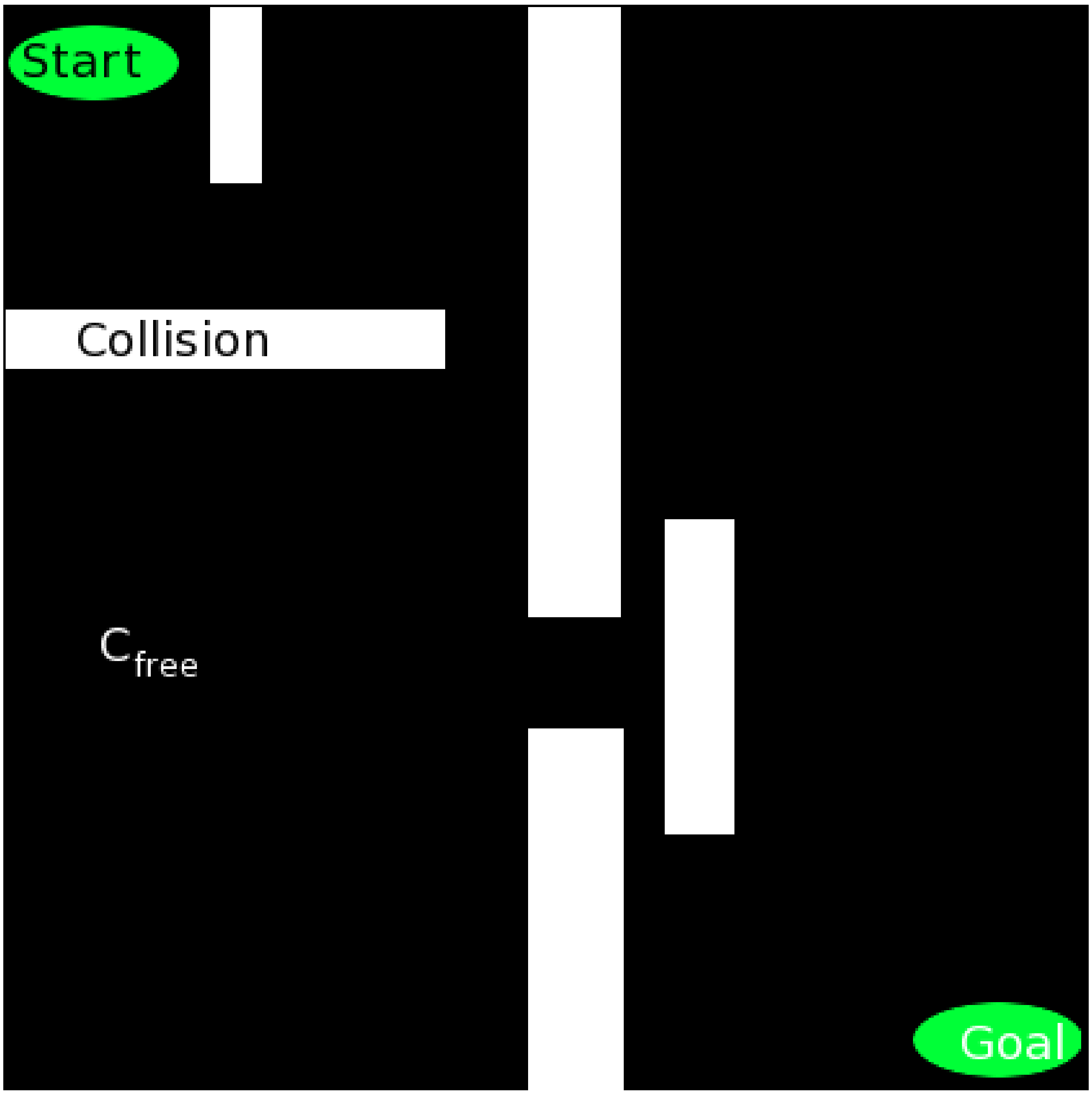


Figure 43 Concepts of Trajectory Planning

Trajectory planning gives a path from a starting configuration S to a goal configuration G avoiding collisions in a 2D or 3D space.

A configuration is the pose of a robot describing its position. Configuration Space C, is the set of all configurations. For instance in two dimensions a robot's configuration would be described by coordinates (x, y) and angle θ. Whereas in three dimensions a robot's configuration would be described by coordinates (x, y, z) and angles (α, β, γ).

Free space C_{free} is the set of all configurations that are collision free. Computing the shape of C_{free} is not effecient, however computing if a given configuration is collision free is by simply using kinematics and collision detection from sensors.

29.10.3. Planning Algorithms

Potential Field Planning

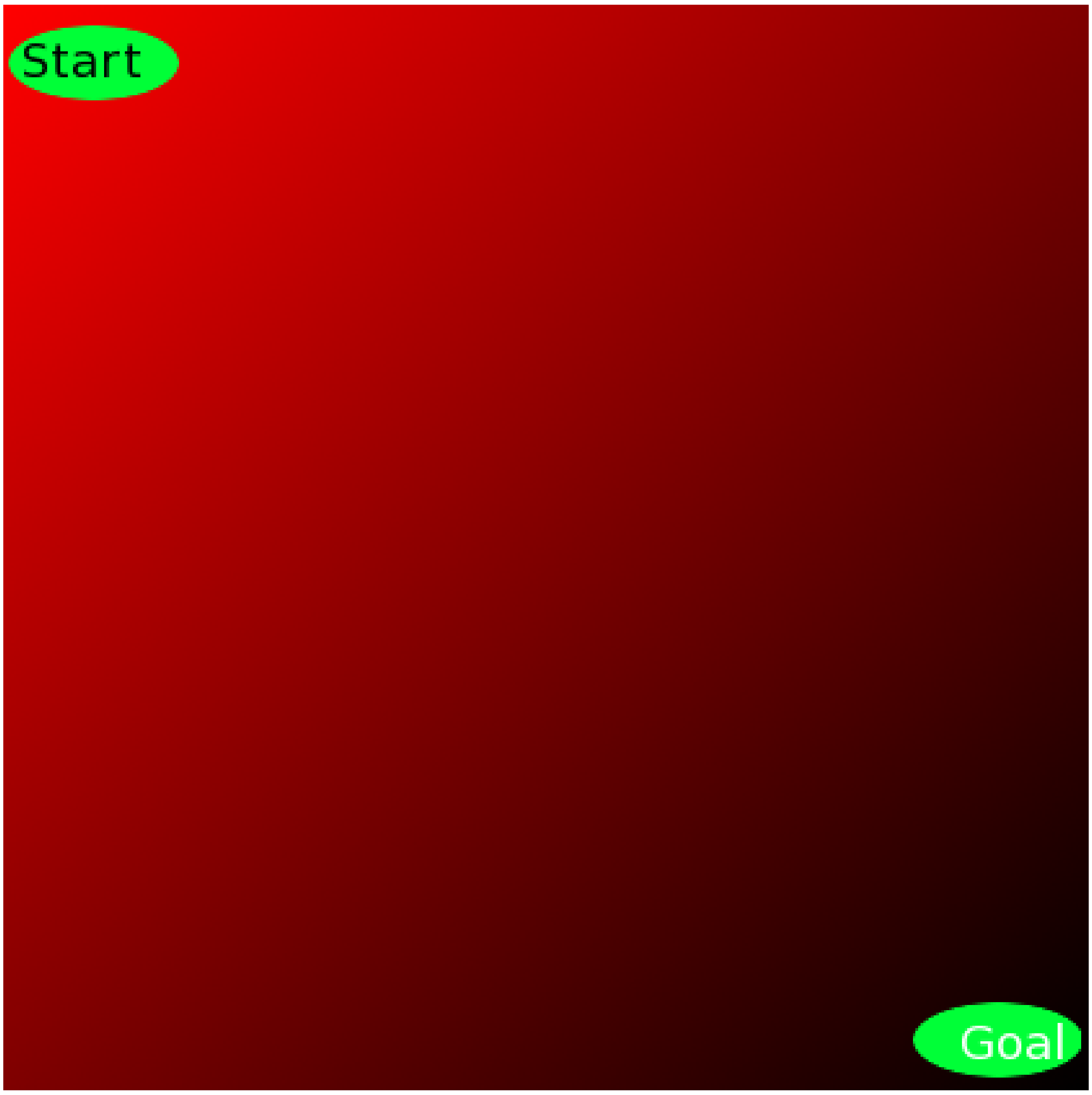


Figure 44 Example of a Potential Field

Potential Field Planning places values over the map with the goal having the lowest value raising the value depending on the distance from the goal. Obstacles are defined to have an incredibly high value. The robot then simply moves to the lowest potential value adjacent to it, which should lead it to the goal. However this technique often gets trapped in local minima. Certain techniques can be used to avoid this, such as wavefront potential field planning.

Sampling Based Planning

Roadmap method is one sampling based planning method. First a sample of N configurations in C as milestones. Then a line PQ is formed between all milestones as long as the line PQ is completely in C_{free}. Then graph search algorithms can be used to find a path from start to the goal. As N grows better solutions are found, however this increases computation time.

Grid Based Planning

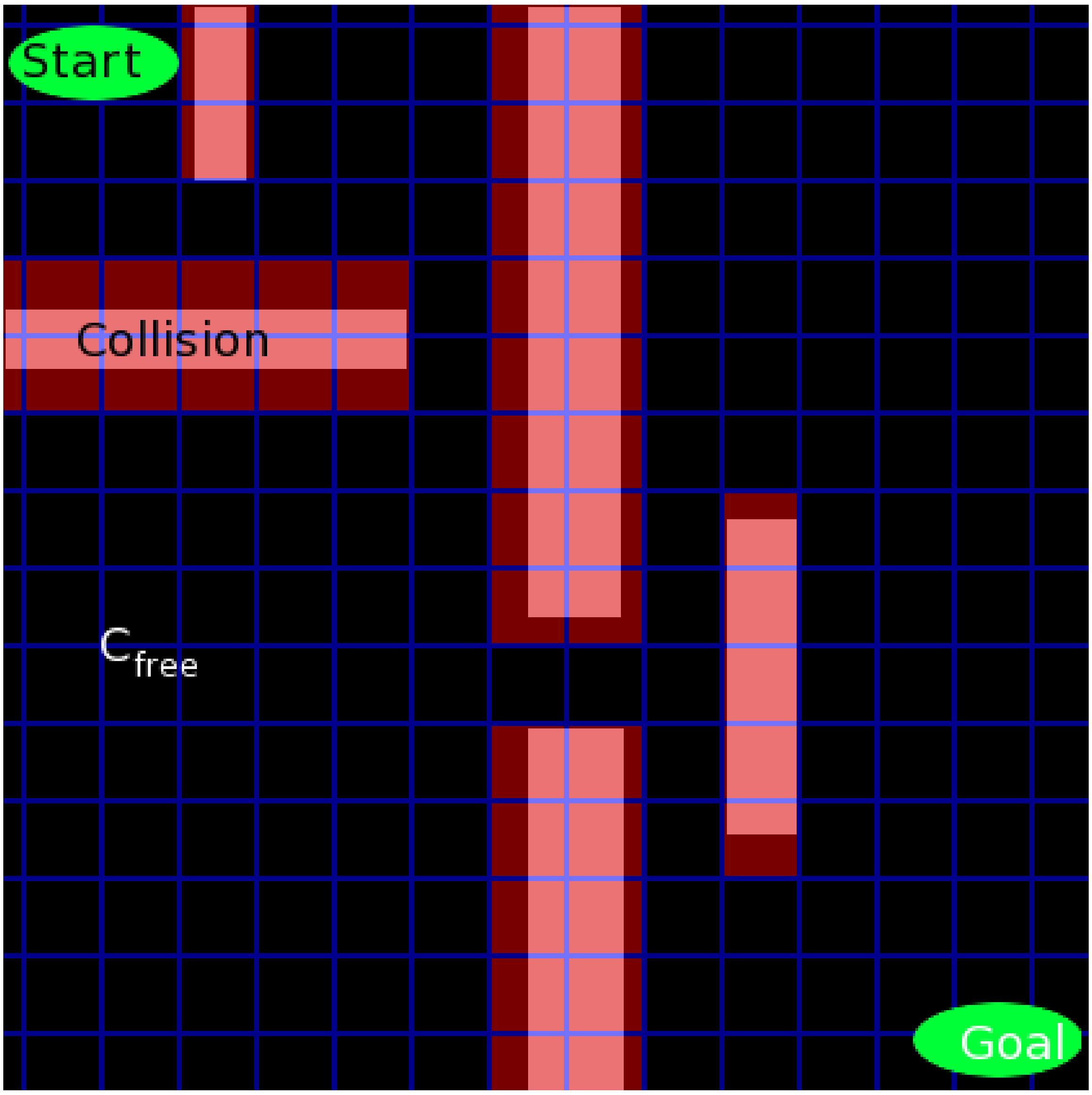


Figure 45 Example of Grid Based Planning

Grid Based planning overlays a grid on the map. Every configuration then corresponds with a grid pixel. The robot can move from one grid pixel to any adjacent grid pixels as long as that grid pixel is in C_{free}. Then a search algorithm such as A* can be used to find a path to get from start to the goal. One potential tradeoff with this method is with a lower resolution grid(bigger pixels) the search will be faster, however it may miss paths through narrow spaces of C_{free}. In addition as the resolution of the grid increases memory usage increases exponentially, therefore in large areas using another path planning algorithm may be necessary.

910111213

----

9 Artificial Intelligence: a Modern Approach. Stuart Russell. Peter Norvig. 2003.

10 Roadmap Methods. `http://parasol.tamu.edu/~amato/Courses/padova04/lectures/L5.roadmaps.ps`

11 ICAPS 2004 Tutorial. `http://www-rcf.usc.edu/~skoenig/icaps/icaps04/tutorial4.html`

12 Steven M. LaValle. "Planning Algorithms". 2006. Cambridge University Press. (The book can be read online at

- `http://planning.cs.uiuc.edu/`

)

13 `http://www.contrib.andrew.cmu.edu/~hyunsoop/Project/Random_Motion_Techniques_HSedition.ppt`

30. Exotic Robots

30.1. Special Robot Brains

Most robots have an electronic ”brain” (part that uses sensor input to determine actuator output). This can be either a simple electronic circuit, a µcontroller or a PC. This is not the only way to build a robot ”brain”.

30.2. Mechanical Robot Brains

One of the first robots was designed by Da Vinci, this was a small vehicle powered by wind-up springs and it was ”programed” by changing gears. Although possible this kind of robot brains are very hard to design and build.

30.3. Pneumatic Robot Brains

If you need to automate some part of a plant where explosive gases are used it might be too dangerous to use electronic circuits. Instead pneumatic circuits are used. It’s possible to construct logical circuits, timers, sequences,... using pneumatic valves and cylinders. You can find a lot of pneumatic valves for automation here: www.ngtvalves.com

30.4. BEAM robotics

This page should discus BEAM robots, The Wikipedia link explains what BEAM robots are, other info over this type of robot can be added.

A special kind of robots are BEAM robotics[1].

30.5. Cooperating Robots

Ever take a good look at a swarm of flying insects? If you did you’d have seen very complicated patterns as if they coordinate their flight with precision. Yet each of these individual insect follow only 2 simple rules:

- Get to the center of the swarm (the safest place)

1 `https://en.wikipedia.org/wiki/BEAM%20robotics`

- Keep your distance from the other insects.

Because of the large number of creatures in such a swarm you get complicated behaviour even from such very simple rules. The same principle applies to ants. Although none of the ants are civil engineers they do succeed in building very complicated habitats.

This same principle can be applied to robots. A large number of small (and cheap) robots can cooperate to tackle a large problem. These swarms of robots are another challenge for robot designers as their behaviour isn't as simple to program as a single robot.

We talk more about "simple" swarming in Robotics/Computer Control/Control Architectures/Swarm Robotics[2] and more complicated coordination in Robotics/Exotic Robots/Modular and fractal robots[3].

30.6. Hazardous Environment Robots

The definition of hazardous is involving or exposing one to risk.[4] Some examples of hazardous environments would be nuclear reactors, outside earth's atmosphere, and even behind enemy lines in war. Robots help humans in many different ways; mostly being able to do task that could be harmful toward the average human being. So in society, robots perform important tasks that your average human being can not perform because of being to dangerous or harmful to their health.

2 Chapter 17.5 on page 106

3 Chapter 31.8 on page 214

4 "Merriam-Webster: hazardous" ^{http://www.merriam-webster.com/dictionary/hazardous} . Retrieved on 2008-12-01.

30.7. Mars Pathfinder

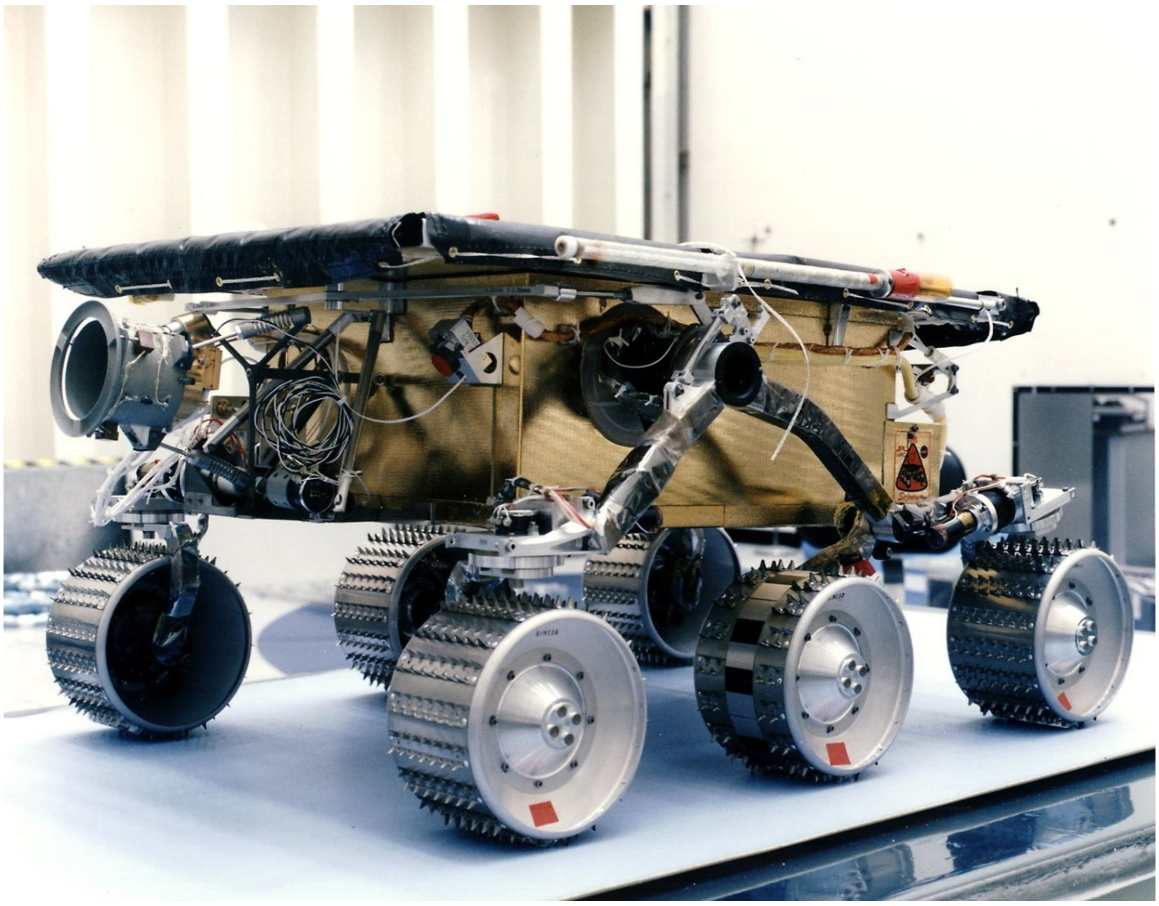


Figure 46 Sojourner, the Mars Pathfinder rover

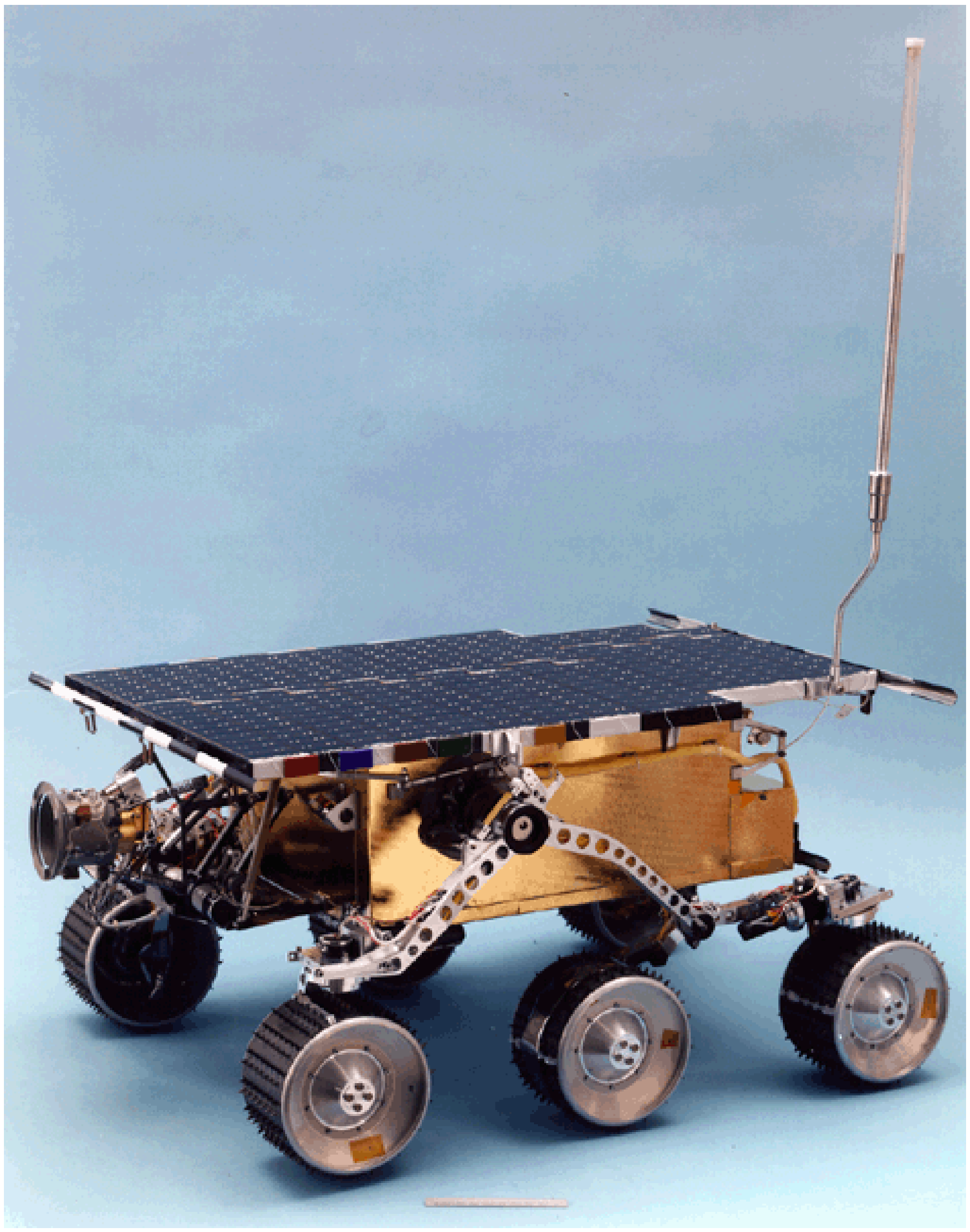


Figure 47 Sojourner, the Mars Pathfinder rover

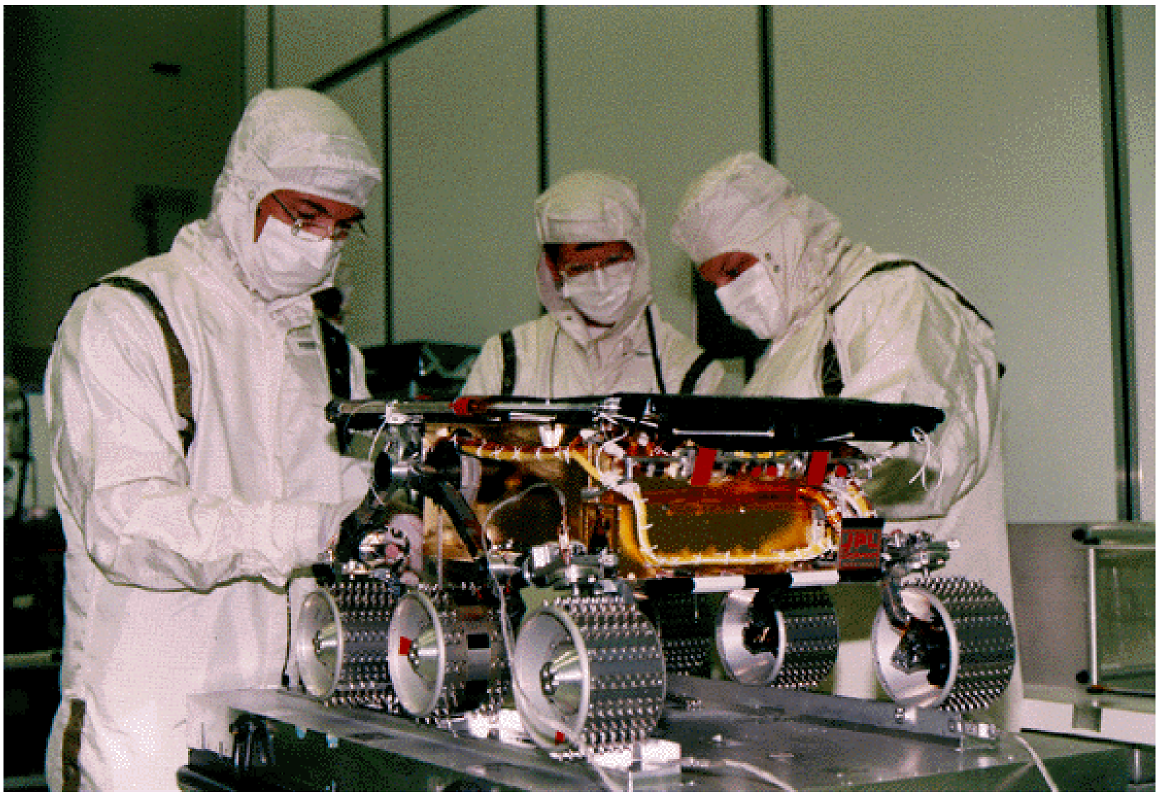


Figure 48 Sojourner, the Mars Pathfinder rover

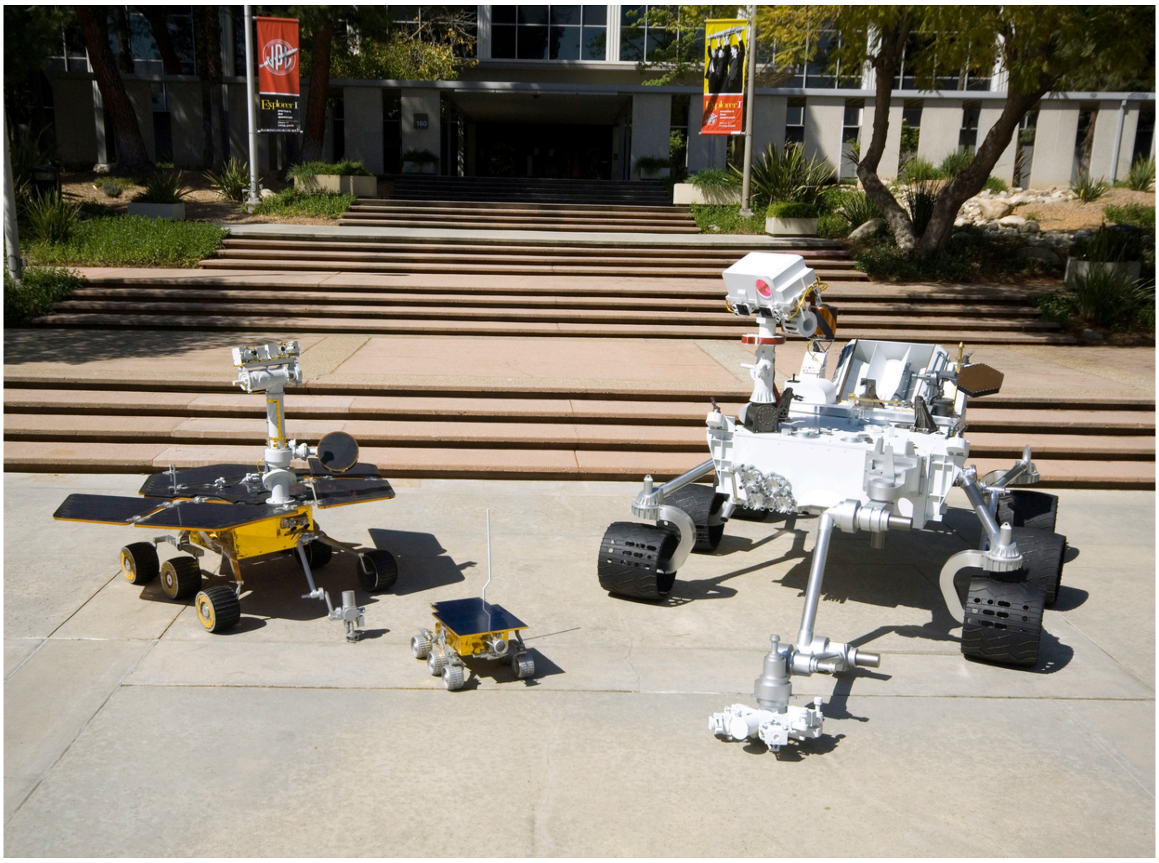


Figure 49 Mars Science Laboratory mockup (right) compared with the Mars Exploration Rover (left) and Sojourner rover (center)

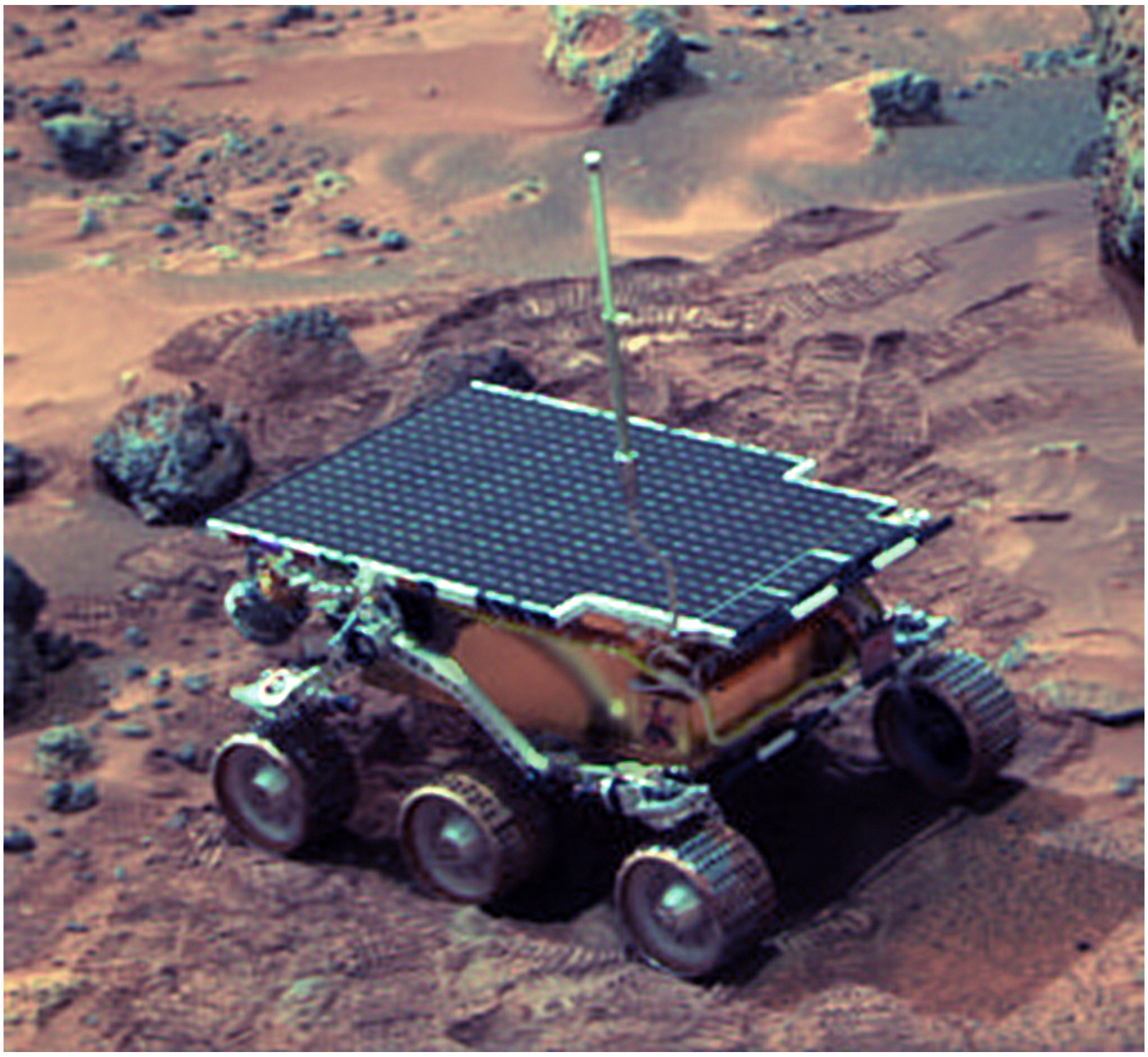


Figure 50 Sojourner on Mars

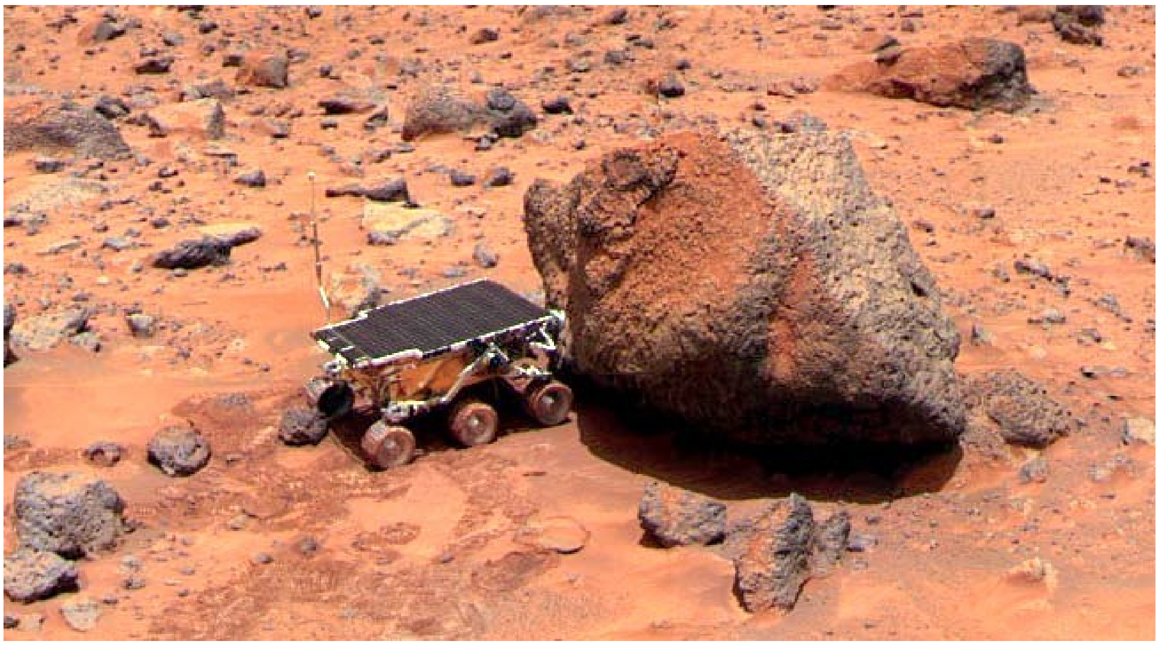


Figure 51 Sojourner taking an X-ray measurement of Yogi.

In space robot have been use as rovers that travel to a distant planets and take data from its surroundings, such as Mars Pathfinder. Using a rover is cheaper than sending humans into space. One of the missions was to "demonstrate a simple, low-cost system, at fixed price for placing a science payload on the surface of Mars at 1/15 the Viking price tag" (jpl.nasa). This is cheaper because going to Mars for humans would require the space shuttle to carry food, oxygen, be able to stand the extreme temperature differences, and especially, be safe enough to carry humans.[5]

30.8. Hubble Telescope

Another robot that is in space right now is the Hubble Telescope. The Hubble telescopes purpose is to provide "unprecedented deep and clear views of the Universe, ranging from our own solar system to extremely remote fledgling galaxies forming not long after the Big Bang 13.7 billion years ago" (hubble). This robot takes clear photos of the solar system. This is a great example of hazardous environment because the Hubble has been taking pictures since 1990 and aside of a few repairs, it has lasted a lot longer than was originally expected to last. `http://www.sc.gov/NR/rdonlyres/93C9CEAE-361E-4F3B-B4C4-CEDD151A6321/2769/FloatingHubble.jpg`

5 California Institute of Technology: Jet Propulsion Laboratory: Robotics Section ^{`http://www-robotics.jpl.nasa.gov/`} .

30.9. Pioneer

Another application is in nuclear environments such as the aftermath of Chernobyl which a disaster that was cause by a nuclear reactor that released a large amount of radioactivity into the environment followed power explosion which destroyed the reactor. The Pioneer is a robot that was built to withstand the radioactive environment in order to deploy sensors, sampling payloads, map the environment in 3D, and cut and retrieve samples of structural materials. A robot has to be used because of rugged terrain, high radiation, and complete darkness (frc.ri.cmu). `http://www.frc.ri.cmu.edu/projects/pioneer/pics/PioneerAtUnit4.jpg`

30.10. pipeline inspection

A variety of robots and robot-like devices are used to inspect pipelines. Some of them visually inspect more-or-less empty pipelines or pipelines full of clear water. Others use "fingers" to mechanically feel the interior shape of the pipeline, sensing interior pitting; or magnetically measure the thickness of the pipeline, indirectly sensing exterior pitting; or use other kinds of sensors while inspecting pipelines full of oil/sewage/etc. Most pipelines have at least one characteristic making robots more practical than direct human inspection: too narrow for an adult human to squeeze through, and too long to inspect the entire length from the ends; or filled with some unpleasant substance, or filled with some opaque substance, or some combination.

A common form of locomotion in a pipeline is to passively let the pressure of the product in the pipeline sweep the robot downstream.

Active locomotion is necessary when the pipe is empty or the robot needs to travel upstream. A robot can move through pipes using peristaltic locomotion,[6] wheels,[7] or a variety of other locomotion techniques.

30.11. Talon Sword

Robots are also used in war. From unmanned aerial vehicles or UAV to ground gunned vehicles, they are used because they put the soldiers out of the way of the bullets. The Talon Sword is used as a military ground robot that ranges from reconnaissance to weapons delivery. "The suitcase-portable robot [Talon Sword] is controlled through a two-way RF or F/O line from a portable or wearable Operator Control Unit (OCU) that provides continuous data and video feedback for precise vehicle positioning" (globalsecurity). This makes it easier for the soldier to operate out of range. `http://xirdal.lmu.de/xirdalium/xpix/talon_swords.png`

6 Alexander Boxerbaum, et. al. "Continuous Wave Peristaltic Locomotion" ^{`http://biorobots.cwru.edu/projects/softworm/`}

7 Kyle Sherer. "Scientists developing intelligent pipe-inspection robot" ^{`http://www.gizmag.com/sintefs-pipe-inspection-robot/9549/`}

30.12. Predator

Another robot used in the military is the Predator. The Predator is a UAV that was developed originally in 1996 and main purpose is to be used in armed reconnaissance, airborne surveillance, and target acquisition (af.mil). http://www.defenseindustrydaily.com/images/AIR_UAV_MQ-1_Predator_lg.jpg

30.13. Conclusion

A major reason for robots is the fact that they can go more places than a human can. They are efficient and in most case get the job done that a human can not. And even though a robot that is built to perform in these hazardous environments cost anywhere from a couple hundred thousand to a couple hundred million, no robot cost is worth more than a human's life.

8910111213

8 w:pigging ^{https://en.wikipedia.org/wiki/pigging} w:pipeline video inspection ^{https://en.wikipedia.org/wiki/pipeline%20video%20inspection}

9

- http://mpfwww.jpl.nasa.gov/MPF/mpf/mission_obj.html. Retrieved on 2008-12-01

10 "Military TALON Small Mobile Robot" ^{http://www.globalsecurity.org/military/systems/ground/talon.htm} . Retrieved on 2008-12-01

11 Air Force Link. "MQ-1 PREDATOR UNMANNED AIRCRAFT SYSTEM" ^{https://web.archive.org/web/20081015180400/http://www.af.mil/factsheets/factsheet.asp?fsID=122} . Retrieved on 2008-12-01.

12 NASA. "Hubble Space Telescope" ^{http://hubble.nasa.gov/} . Retrieved on 2008-12-01.

13 "Project Pioneer" ^{http://www.frc.ri.cmu.edu/projects/pioneer/} . Retrieved on 2008-12-01.

31. Types of Robots

In general, robots are classified based on their capabilities. Some standard classifications of robots include their domain of operation[1], degree of autonomy[2], and the goal[3] they are designed to fulfill.

31.0.1. Domain of Operation

Robots can be designed and built for any environment imaginable. One popular way of classifying robots is by what environments they're designed to operate in. Some typical examples include:

Stationary

These robots are fixed in one place and cannot move. This category includes robotic arms[4], computerized machine tools, and most other Industrial Robots.

Industrial Robots are robots used in mass production e.g. welding robots, CNC plate cutters or CNC drills. The large majority of these robots are stationary and tethered to a computer.

Ground

These robots are designed to operate on the surface of the earth or other planet, and are usually sub categorized by their drive train:

- **Wheels**[5]
- **Tracks**[6]
- **Legs**[7]

Underwater

Also known as Autonomous Underwater Vehicles, these are designed to operate underwater, possibly at great depth.

Aerial

Unmanned Aerial Vehicles are various kinds of robotic flying machines, including planes and helicopters.

Microgravity

1 Chapter 31.0.1 on page 191
2 Chapter 31.0.2 on page 192
3 Chapter 31.0.3 on page 192
4 https://en.wikibooks.org/wiki/%2FArms
5 https://en.wikibooks.org/wiki/%2FWheeled
6 https://en.wikibooks.org/wiki/%2FTracked
7 https://en.wikibooks.org/wiki/%2FWalkers

Robots that have been designed to operate in low-gravity environments, such as earth orbit.

other specific Hazardous Environments[8]

31.0.2. Degree of Autonomy

Autonomy is the quality of being self-controlled. One measure of autonomy is the amount of human control that is required for the robot's operation. An autonomous robot can operate properly without intervention indefinitely and can deal with unexpected problems gracefully. Teleoperated robots constantly require humans to send the robot control signals. These are only the endpoints; there is a continuum of possibilities between them.

A robot can also be classified by how self-contained it is. Power, logic circuitry, and other things may be located either on the main chassis or connected via a cable tether or wireless link from another location.

31.0.3. Goal

Robots are also routinely categorized by the goals they are designed to achieve. These include contests[9], personal enrichment, manufacturing, and entertainment[10].

31.1. Contest Robot

Here are a few examples of contests with robots:

(Here's[11] a list of different contests)

31.1.1. Fire fighter

In a fire fighter contest the robot has to find the flame of a candle and extinguish it.

See this site[12] for an example.

31.1.2. Maze or obstacle course navigator

Maze

Find the way through the maze as fast as possible.

8 Chapter 30.6 on page 182
9 `https://en.wikibooks.org/wiki/%2FContest%20Robot`
10 `https://en.wikibooks.org/wiki/%2FEntertainment%20Robot`
11 `http://robots.net/rcfaq.html`
12 `http://www.trincoll.edu/events/robot/`

For example: see this site.[13]

For a video examples:

- video 1[14]
- video 2[15]
- video 3[16] (multiple contestants--both successful and not)

Obstacle Course

There are many different obstacle course to build for a robot contest. The Melexis Safety Trophy has tin cans with small candles inside and concrete bricks as obstacles and tennis balls (which the robot had to touch) for extra points. Another contest uses traffic signs with arrows to direct the way.

See this site[17] for an example.

31.1.3. Soccer robot

Robots playing soccer. The goal of this type of contest is to build a team of robots that can work together.

See this site[18] for an example.

31.1.4. Sumo Robot

In this type of contest 2 robots are placed on a playground within a large circle. The goal is to push the other robot out of this circle.

See here[19] for an example.

31.1.5. further reading

- "Knox Computer Club Robot Sumo Competition"[20]
- "Building a Sumo robot"[21] by ian 2006
- How to make a "Mini Robot Sumo Ring"[22] by David Cook

13 http://imd.eng.kagawa-u.ac.jp/maze/index_j.html
14 http://www.youtube.com/watch?v=_L9rkLAskWU
15 http://www.youtube.com/watch?feature=endscreen&NR=1&v=ZtOtrVB_rOw
16 http://www.youtube.com/watch?feature=endscreen&NR=1&v=ZtOtrVB_rOw
17 http://www.trophy.melexis.com/en/home.htm
18 http://www.newtonlabs.com/soccer/
19 http://www.botlanta.org/MonthlyContest/sumo/mini-sumo.html
20 http://cs.knox.edu/Sumo/Sumo2008.html
21 http://www.mouldy.org/project/Sumo-Robot-Overview
22 http://www.robotroom.com/SumoCircleMini.html

31.2. Entertainment Robot

These are robots that have no real *useful* application. They're build for fun and practice but little else more. Most homebuilt robots would fall in this category.

They can be simple robots that use simple sensors (light sensor, sound, bumper switches) and have simple behavior, but they could as well have complex sensors and extensive programming.

One other category of robots that's very close related to this category are robot build as "tech-demos". For example the Asimo robot.

31.3. Arms

Arms are types of jointed robot manipulator that allow robots to interact with their environment. Many have onboard controllers or translators to simplify communication, though they may be controlled directly or in any number of ways. Due to this fact, standalone arms are often classified as full robots.

Types of Robotic Arms

There are many different types of robotic arms, but most can be characterized into one of six major categories by their mechanical structure. Cartesian (also known as Gantry) robots have three joints that are coincident with the standard X-Y-Z Cartesian axes. Cylindrical arms have any number of joints that operate on a cylindrical axis, normally rotating about one fixed rod. Spherical (polar) arms are those with joints that allow it full rotation throughout a spherical range. SCARA robots have two parallel rotary joints to allow full movement throughout a plane, typically for pick-and-place work. Articulated robots are used for complex assembly operations, and consist of three or more rotary joints. Parallel robots have three concurrent prismatic or rotary joints, and allow for tilting of heavy or sensitive platforms.

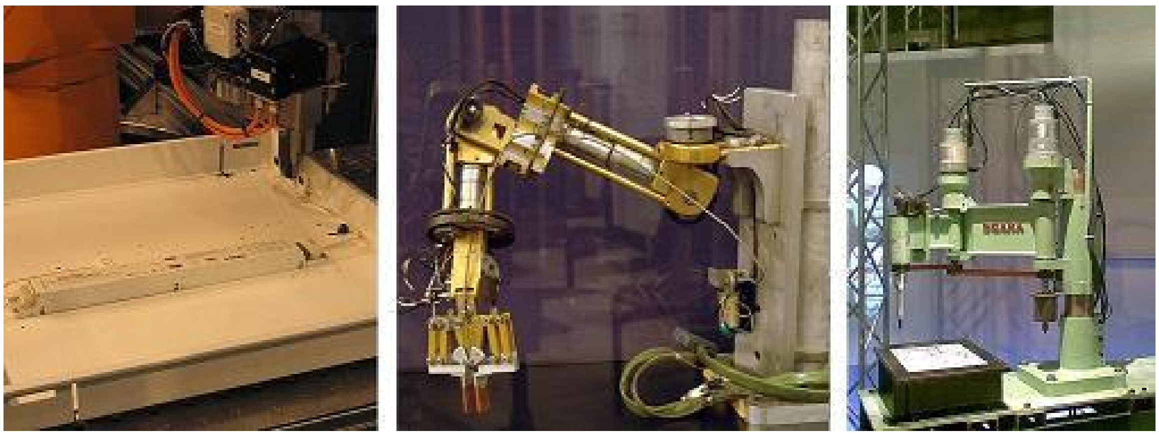


Figure 52
(Left to Right) Gantry Robot, Articulated Robot, SCARA Robot

Applications

Robotic arms are typically used in industry. Repetitive autonomous robots perform one task repeatedly based upon predetermined movements and specifically located objects. Start and stop commands are determined by position, acceleration, deceleration, distance, and direction. More complex actions are executed based upon sensor processing. If object orientation or position is unknown, arms are often paired with machine vision and artificial intelligence to identify the object and subsequently control the arm.

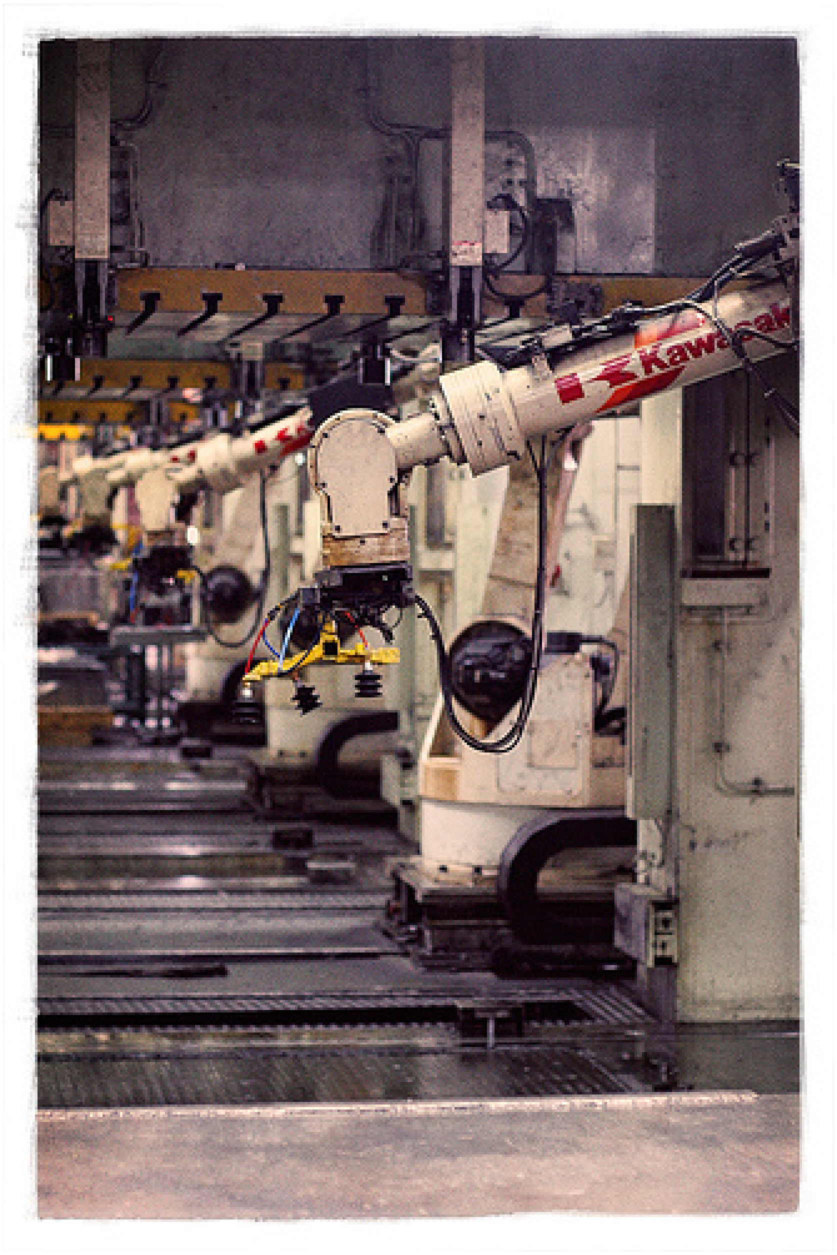


Figure 53

Standard Robotic Factory Setup

Parameters

Arms are typically defined by fourteen different parameters.

Number of Axes – Two axes are needed to reach any point in a plane. Three are required to reach a point in space. Roll, pitch, and yaw control are required for full control of the end manipulator.

Degrees of Freedom – Number of points a robot can be directionally controlled around. A human arm has seven degrees; articulated arms typically have up to 6.

Working Envelope – Region of space a robot can encompass.

Working Space – The region in space a robot can fully interact with.

Kinematics – Arrangement and types of joints (Cartesian, Cylindrical, Spherical, SCARA, Articulated, Parallel)

Payload – Amount that can be lifted and carried

Speed – May be defined by individual or total angular or linear movement speed

Acceleration – Limits maximum speed over short distances. Acceleration is given in terms of each degree of freedom or by axis.

Accuracy – Given as a best case with modifiers based upon movement speed and position from optimal within the envelope.

Repeatability – More closely related to precision than accuracy. Robots with a low repeatability factor and high accuracy often need only to be recalibrated.

Motion Control – For certain applications, arms may only need to move to certain points in the working space. They may also need to interact with all possible points.

Power Source – Electric motors or hydraulics are typically used, though new methods are emerging and being tested.

Drive – Motors may be hooked directly to segments for direct drive. They may also be attached via gears or in a harmonic drive system

Compliance – Measure of the distance or angle a robot joint will move under a force.

End Effectors

The arm itself is only responsible for positioning. An end effector is necessary for actual environmental interaction. Some common choices are grippers, sprayers, grinders, welders, and vacuums, though many other options are available. There is a large variance in complexity, ranging from flush mounted, non-moving parts (magnets or sticky pads) to multi-jointed, multi-sensor parts with various inputs and outputs. End effectors are typically chosen based upon the application, and many arms will fit multiple end effectors.

Distribution

A 2006 Japanese report shows that 60% of robotic arm installations were for articulated robots, 22% were for gantry, 13% were SCARA, and 4% were Cylindrical. The automobile industry has the highest concentration of industrial robots with nearly 1 robot per 10 people.

Biological Modeling

Biological design of robotic components is becoming increasingly popular. Currently, the Shadow Company (famous for the Shadow Hand) is working on a biomorphic arm that accurately models a human arm. The arm has seven degrees of freedom like the human arm which greatly increases its efficiency.

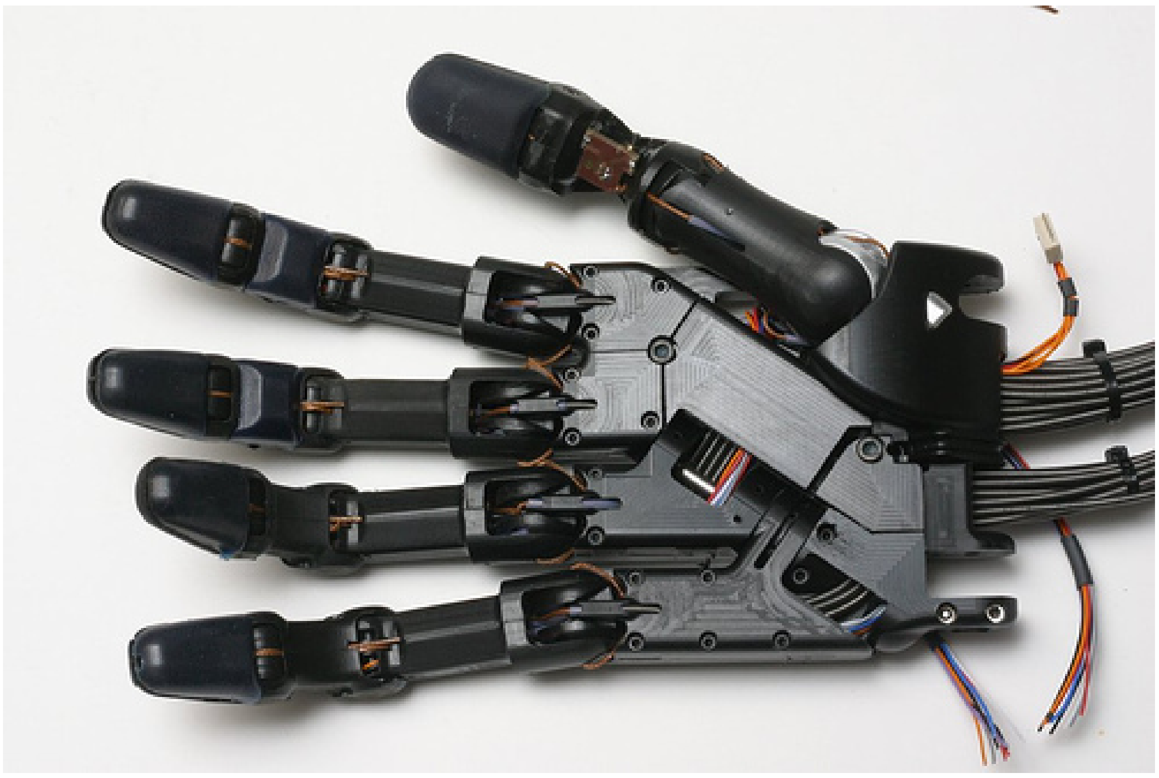


Figure 54

The Shadowhand

Researchers at Fraunhofer Institute for Manufacturing Engineering and Automation IPA in Stuttgart have begun working on an arm that is modeled after an elephant's trunk. The arm uses a cord that is connected to a drive shaft at its midpoint. When the shaft turns, the cord wraps around it both ways forming a double helix. Their prototype arm uses ten of these cord systems with two motors attached to each (the second acts as a failsafe for increased safety).

Similarly, biologists at Hebrew University in Jerusalem are studying Octopi tentacles as a possible model for future robotic arm development. An prototype has not yet been built, but arm design has recently begun.

Additionally, two monkeys learned to control robotic arms through sensors implanted on their brains in a study performed the University of Pittsburgh and Carnegie Mellon University. While not currently practical, the study confirms that brain-controlled prosthetics are feasible and has sparked interest in the area.

31.4. Wheeled

Wheeled robots are robots that navigate around the ground using motorized wheels to propel themselves. This design is simpler than using treads or legs and by using wheels they are easier to design, build, and program for movement in flat, not-so-rugged terrain. They are also more well controlled than other types of robots. Disadvantages of wheeled robots are that they can not navigate well over obstacles, such as rocky terrain, sharp declines, or areas with low friction. Wheeled robots are most popular among the consumer market, their differential steering provides low cost and simplicity. Robots can have any number of wheels, but three wheels are sufficient for static and dynamic balance. Additional wheels can add to balance; however, additional mechanisms will be required to keep all the wheels in the ground, when the terrain is not flat.

31.4.1. Navigation

Most wheeled robots use differential steering, which uses separately driven wheels for movement. They can change direction by rotating each wheel at a different speed. There may be additional wheels that are not driven by a motor these extra wheels help keep it balanced.

31.4.2. 2-wheeled robots

Two wheeled robots are harder to balance than other types because they must keeping moving to maintain upright. The center of gravity of the robot body is kept below the axle, usually this is accomplished by mounting the batteries below the body. They can have their wheels parallel to each other, these vehicles are called dicycles, or one wheel in front of the other, tandemly placed wheels. Two wheeled robots must keep moving to remain upright and they can do this by driving in the direction the robot is falling. To balance, the base of the robot must stay with under its center of gravity. For a robot that has the left and right wheels, it needs at least two sensors. A tilt sensor that is used to determine tilt angle and wheel encoders which keep track of the position of the platform of the robot.

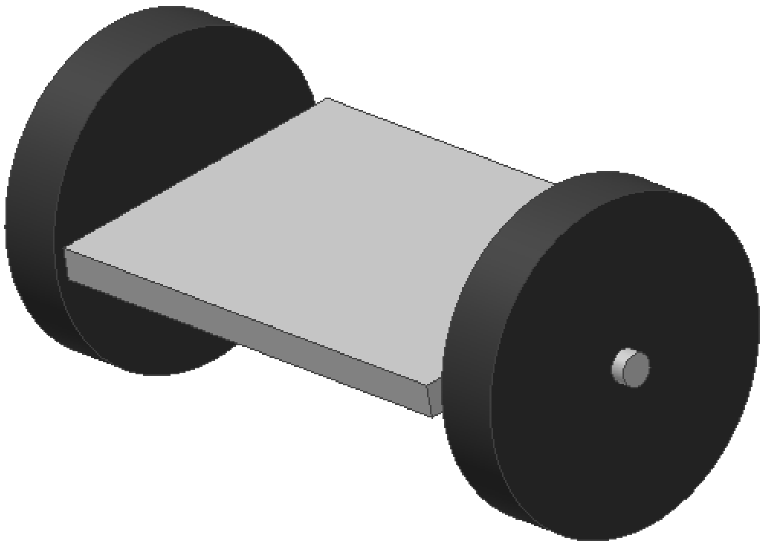


Figure 55 Swing-type robot

Examples

Roomba

Roombas are two-wheeled vacuum cleaners that automatically moves around cleaning up a room. They utilizes a contact sensor in the front and a infrared sensor on its top.

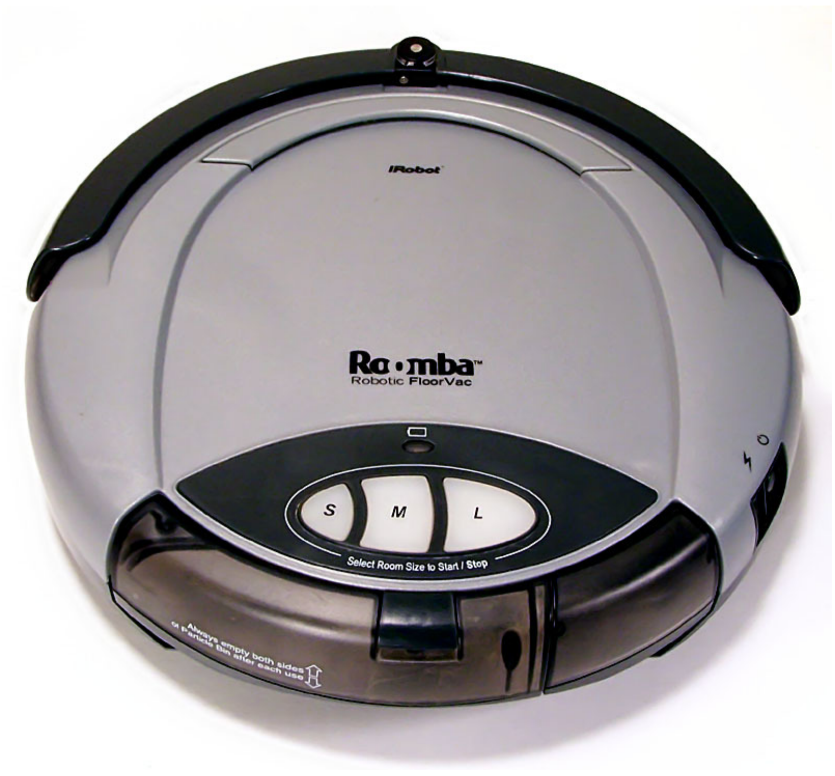

iRobot
Roomba™
Robotic FloorVac
S
M
L
Select Room Size to Start / Stop
Always empty both sides of Particle Bin after each use

Figure 56 Roomba

Segway
Segways[23] are self-balancing dicycle electric vehicles.

Ghost Rider
Ghost Rider[24] was the only two wheeled robot entered for the Darpa Grand 2005 Challenge[25]. It was unique because of its motorcycle design, unlike the other two-wheeled robots, the wheel alignment is front and back, which makes it harder to balance as it turns. This tandem design of the wheels is much less common than that of a dicycle.

31.4.3. 3-wheeled vehicles

3-wheeled robots may be of two types: differentially steered (2 powered wheels with an additional free rotating wheel to keep the body in balance) or 2 wheels powered by a single source and a powered steering for the third wheel. In the case of differentially steered wheels, the robot direction may be changed by varying the relative rate of rotation of the two separately driven wheels. If both the wheels are driven in the same direction and speed,

23 `http://www.segway.com/`
24 `http://www.ghostriderrobot.com/`
25 `http://www.darpa.mil/grandchallenge05/`

the robot will go straight. Otherwise, depending on the speed of rotation and its direction, the center of rotation may fall anywhere in the line joining the two wheels.

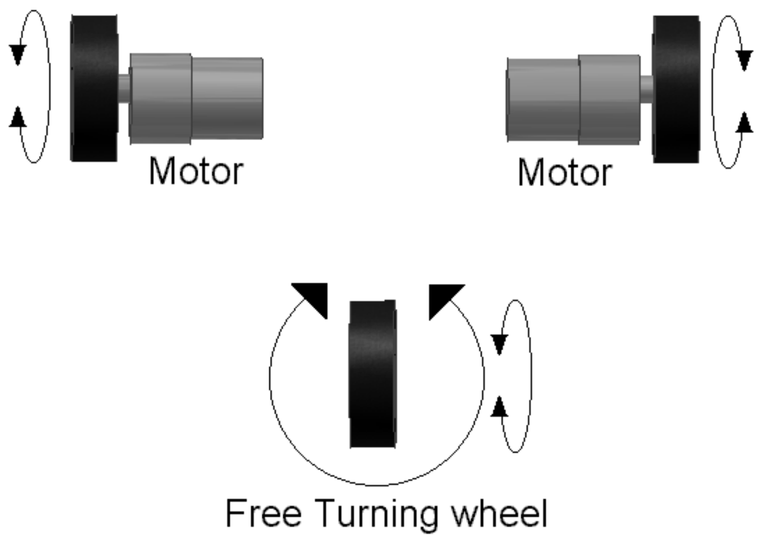


Figure 57
Differentially steered 3 wheeled vehicle

The center of gravity in this type of robot has to lay inside the triangle formed by the wheels. If too heavy of a mass is mounted to the side of the free rotating wheel, the robot will tip over.

31.4.4. Omni Wheels

Another option for wheeled robots that makes it easier for robots with wheels not all mounted on the same axis to have Omni Wheels. An omni wheel is like many smaller wheels making up a large one, the smaller ones have axis perpendicular to the axis of the core wheel. This allows the wheels to move in two directions, and the ability to move holonomically, which means it can instantaneously move in any direction. Unlike a car, which moves non-holnomicallly and has to be in motion to change heading. Omni-wheeled robots can move in at any angle in any direction, without rotating beforehand. Some omni wheel robots use a triangular platform, with the three wheels spaced at 60 degree angles. Advantages of using 3 wheels and not 4 are that its cheaper, and 3 points are guaranteed to be on the same plane, so each wheel in contact with the ground, but only one wheel will be rotating in the direction of travel. The disadvantages of using Omni wheels is that they have poor efficiency due to not all the wheels rotating in the direction of movement, which also causes loss from friction, and are more computationally complex because of the angle calculations of movement.

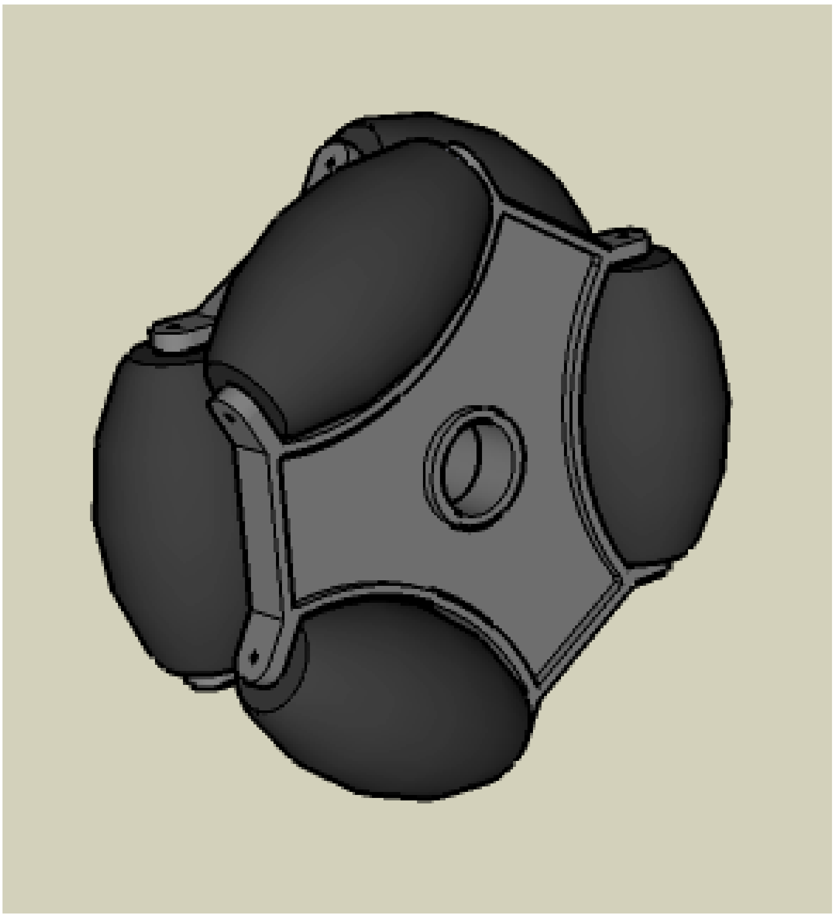


Figure 58
A simple omni wheel. The free rotating rollers (dark gray) allow the wheel to slide laterally.

31.4.5. 4-wheeled vehicles

2 powered, 2 free rotating wheels

Same as the Differentially steered ones above but with 2 free rotating wheels for extra balance.

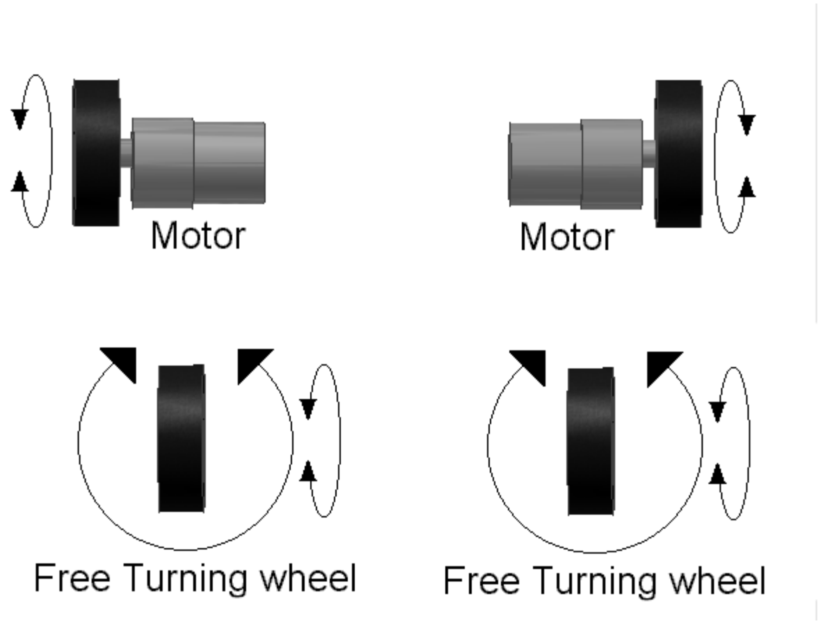


Figure 59

2 powered, 2 free rotating wheels

More stable than the three wheel version since the center of gravity has to remain inside the rectangle formed by the four wheels instead of a triangle. This leaves a larger useful space. Still it's advisable to keep the center of gravity to the middle of the rectangle as this is the most stable configuration, especially when taking sharp turns or moving over a non-level surface.

2-by-2 powered wheels for tank-like movement

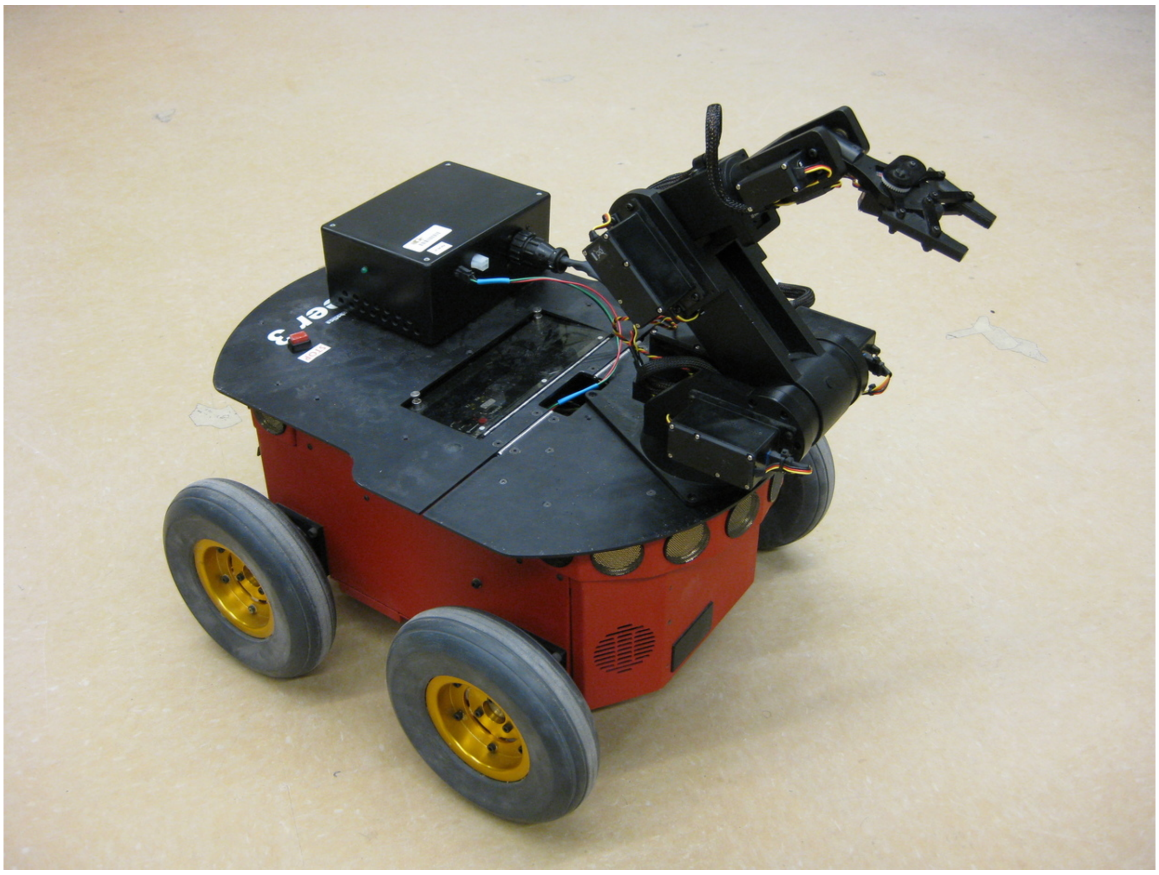


Figure 60 The Pioneer 3-AT robot has four motors and four unsteered wheels; on each side a pair of motors drives a pair of wheels through a single belt.

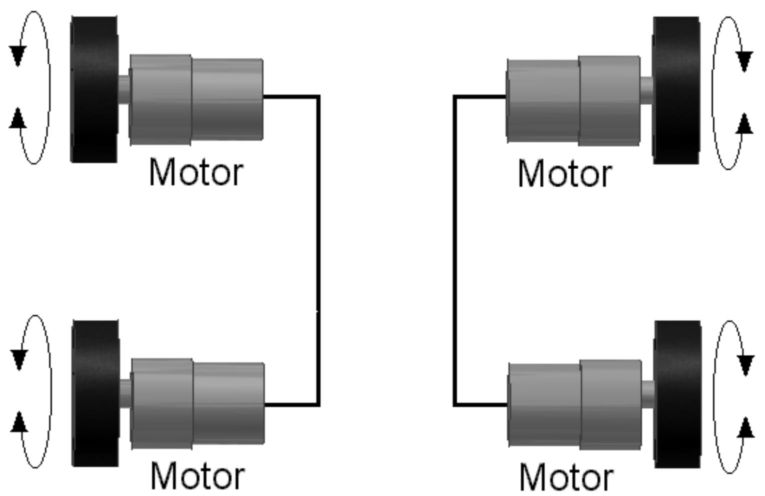


Figure 61
4 wheel drive

This kind of robot uses 2 pairs of powered wheels. Each pair (connected by a line) turn in the same direction. The tricky part of this kind of propulsion is getting all the wheels to turn with the same speed. If the wheels in a pair aren't running with the same speed, the slower one will slip (inefficient). If the pairs don't run at the same speed the robot won't be able to drive straight. A good design will have to incorporate some form of car-like steering.

Car-like steering

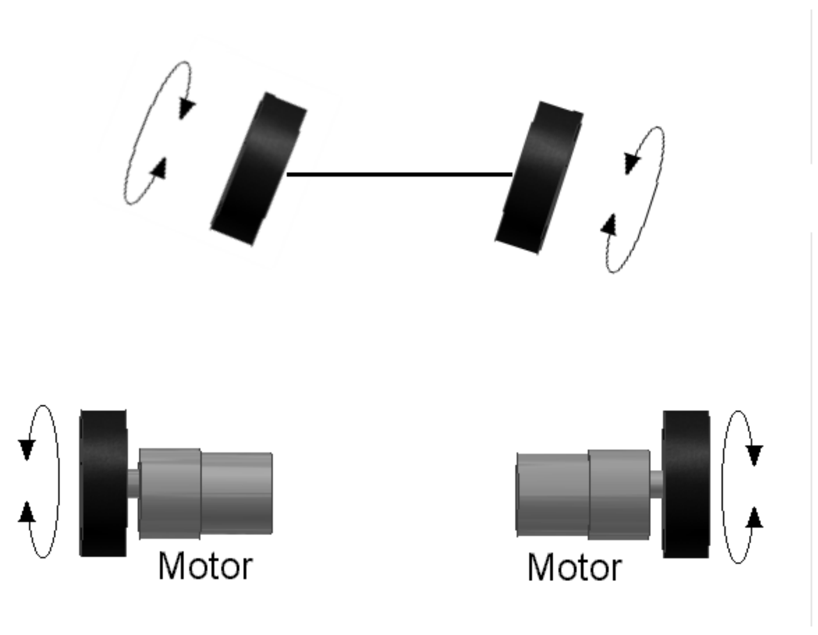


Figure 62
Differential Steering

This method allows the robot to turn in the same way a car does. This is a far harder method to build and makes dead reckoning much harder as well. This system does have an advantage over previous methods when your robot is powered by a combustion engine: It only needs one motor (and a servo for steering of course). The previous methods would require either 2 motors or a very complicated gearbox, since they require 2 output axles with independent speed and direction of rotation.

Examples

The DARPA Grand and Urban Challenges[26] pit robotic cars against one another in a series of navigational tests. These robots are fully automated and drive themselves along the test course. The DOD sponsors the competition and it is used to facilitate robotic development.

31.4.6. 5 or more wheeled vehicles

For larger robots. Not always very practical.

Especially when more powered wheels are used the design becomes much more complex as each of the wheels have to turn with the same speed when the robot has to move forwards.

26 `http://www.darpa.mil/grandchallenge/index.asp`

Differences in speed between the left and right wheels in differentially steered robots cause the robot to move to the side instead of in a straight line. Difference in speed between wheel on the same side cause slipping of the slowest wheel.

Sometimes an extra free rotating wheel with odometry is added to the robot. This measures more accurately how the robot moves. Odometry on the powered wheels excludes slip and other movements and thus could be erroneous.

Examples

The Mars Rovers[27] (Sojourner, Spirit, Opportunity) are six wheeled robots that navigate across Martian terrain after landing. They are used to examine territory, interesting landmarks and make observations about the surface of Mars. They have a suspension system which keeps all six wheels in contact with the surface, and helps them traverse slopes ans sandy terrain.

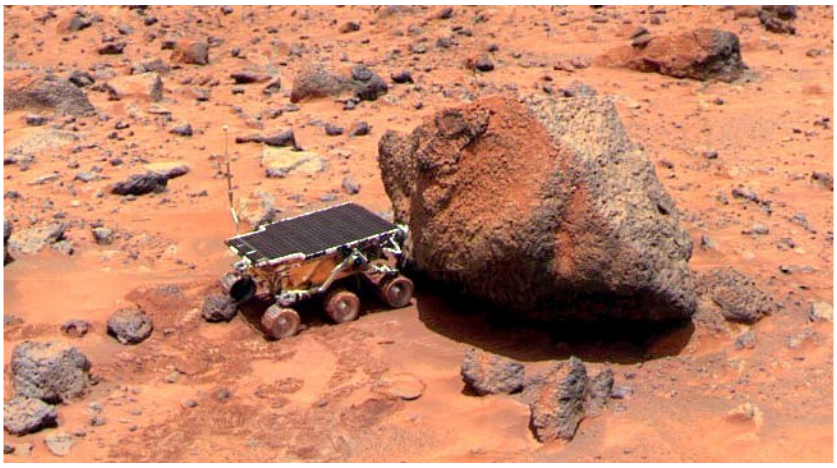


Figure 63

The Sojourner Rover

27 http://marsrover.nasa.gov/home/index.html

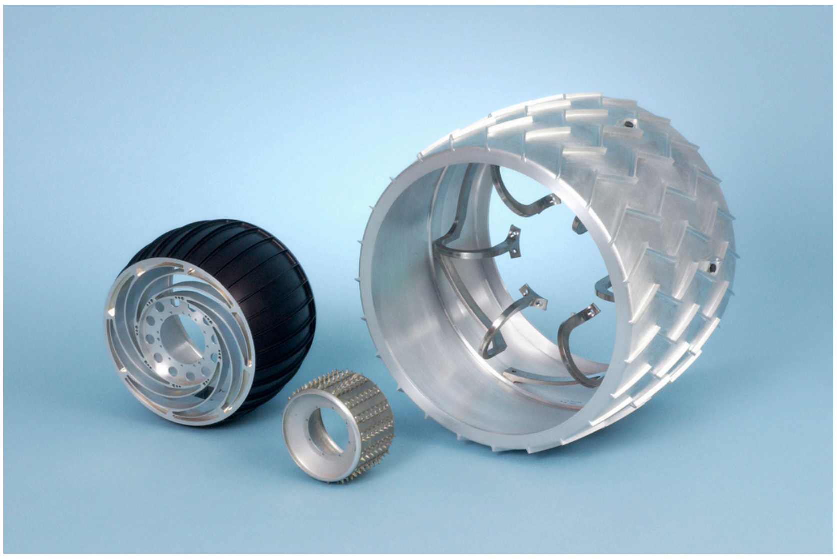


Figure 64

Mars rover wheels sizes: Mars Exploration Rover (MER), Sojourner (Mars Pathfinder mission) and Mars Science Laboratory (from left to right)

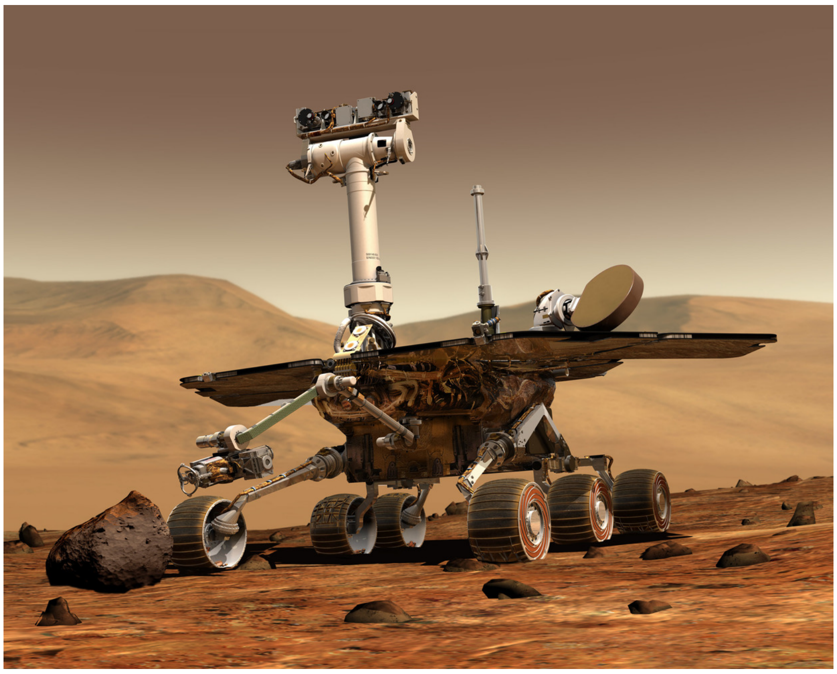


Figure 65

Artist's Concept of Rover on Mars

31.4.7. One Wheel

One wheeled robots are extremely difficult to keep balanced due to the single point of contact with the ground. There have been experimental designs and robots that do only have one wheel. It is easier to use a spherical wheel rather than a typical disc wheel, as the robot can move in any direction along the sphere. An example sketch[28] shows the basic idea of using gyroscopes and counter-torque mechanisms to keep the robot upright. The spinning flywheels stabilize and tilt it, allowing for non-holonomic movement.

31.4.8. External links

- Dicycles and Diwheels throughout history[29]
- Differential robots[30]
- Two wheeled nbot[31]
- Three omni wheeled robot[32]
- Omni wheels[33]
- Darpa Grand Challenge[34]
- Ghost rider Robot[35]
- Rosie the Robot - a Question of Balance[36]
- Different types of Wheeled robots[37]

31.5. Tracked

This type of robot uses tracks like tanks for movement. it's very effective in outdoor environments, especially on loose sand. However on concrete it's less effective than wheeled vehicles.

31.6. Walkers

"In the early days of air travel, detractors used to argue that, if God had meant us to fly, he would have given us wings. Had he meant us to roll, he might also have given us wheels – but instead we, along with the great preponderance of land animals great and small, have wound up traveling on legs."[McGeer92]

28 http://mrl.nyu.edu/~perlin/experiments/rosie/rosie-sketch.html
29 http://www.aqpl43.dsl.pipex.com/MUSEUM/TRANSPORT/diwheel/diwheel.htm
30 http://en.wikipedia.org/wiki/Differential_wheeled_robot
31 http://www.geology.smu.edu/~dpa-www/robo/nbot/
32 http://www.societyofrobots.com/robot_omni_wheel.shtml
33 http://www.omniwheel.com/omniwheel/omniwheel.htm
34 http://www.darpa.mil/GRANDCHALLENGE/
35 http://www.ghostriderrobot.com/
36 http://mrl.nyu.edu/~perlin/experiments/rosie/
37 http://www.robotplatform.com/knowledge/Classification_of_Robots/Types_of_wheeled_robots.html

31.6.1. Legs v. Wheels

Most man-made vehicles today travel on wheels and for good reason: wheels are much easier to construct and control. In today's economy, they also tend to be much cheaper than their legged counterparts. However legs have distinct advantages over wheels. The biggest advantage is in transversability and efficiency. Legged robots have a unique ability to:

- Isolate their body from terrain irregularities
- Avoid undesirable footholds
- Regulate their stability
- Achieve energy efficiency[Gait]

These advantages are very desirable in modern robotics, and therefore a lot of research is being put into creating robots that can walk. The most challenging task in designing a legged robot is to create a system that can generate the proper gait. [Gait]

31.6.2. Dynamic v. Static

Locomotion techniques can be divided into two main categories: static and dynamic.

Robots that use static movement are always balanced; that is, their center of gravity is always within their ground contact base. While this technique has been successfully used to create many robots (included wheeled ones), it is more akin to wheeled movement than true dynamic walking and as such retains fewer of the advantages. While more adept at transversing uneven terrain than most wheeled robots, robots that use static walking are very inefficient as power is put into every movement. However, robots that use static walking are much easier to control than their dynamic counterparts and thus often more viable.

Dynamic walking is characterized in that the robot is not always in balance. Many robots that use dynamic walking are continually "falling" and thus much more energy efficient. Dynamic walking requires much more complex control systems in order to not fall. Robots utilizing dynamic walking cannot use the same motions at different speeds to attain different speeds of movement, but must use entirely different motions at different speeds. However, dynamic walking can achieve many more advantages over wheeled locomotion. Dynamic walking is found very abundantly in nature.

A subset of dynamic walking is called passive dynamic movement. Most dynamic walking systems use active control to move the legs to the correct orientations for walking (hence active dynamic walking). Passive dynamic walking is characterized by a system where "gravity and inertia alone generate the locomotion pattern." [McGeer90] Passive dynamic movement can be achieved with maximum efficiency, as the vehicle uses its own forward momentum to propagate its next movement. Very little energy is lost from the system. Most of the concepts of passive dynamic walking and research conducted in the field was done by aeronautical engineer Tad McGeer between 1988 and 1992.[]

31.6.3. More Than Four

Many different walking robots have been developed that use six or more legs. This is due to the fact that a robot using this many legs can be controlled with static walking techniques rather than dynamic walking. Most of the walking techniques can be demonstrated sufficiently using the six legged model:

Wave Gait

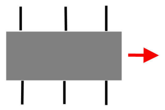


Figure 66
(1)

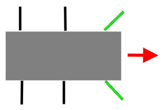


Figure 67
(2)

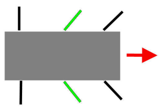


Figure 68
(3)

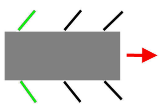


Figure 69
(4)

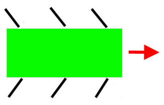


Figure 70
(5)

1. Six legged robot in neutral position
2. Front pair of legs move forward
3. Second pair of legs move forward
4. Third pair of legs move forward
5. Body follows legs forward

Tripod Gait

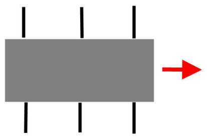


Figure 71
(1)

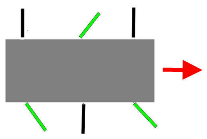


Figure 72
(2)

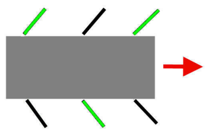


Figure 73
(3)

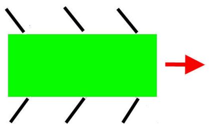


Figure 74
(4)

1. Six legged robot in neutral position
2. Alternating legs move forward on either side
3. All other legs move forward
4. Body follows legs forward

Most robots using six or more legs use a variation of one of these two gait models.

31.6.4. Four Legs

A system on four legs is another walking scheme found readily in nature. Four legged robots have the advantage of being statically stable when not moving, but require dynamic walking control. There are many different ways for a four legged robot to walk including alternating pairs and opposite pairs as in six legged robots. However these techniques now cease to be statically stable and thus require dynamic control.

Boston Dynamics has developed a four legged robot for DARPA (Defense Advanced Research Projects Agency) called “Big Dog,” that they claim is “the most advanced quadruped robot on earth.”[BigDog] Big Dog can run at four miles per hour, climb thirty five degree slopes, and carry 340 pounds. But the most impressive feature is its dynamic walking: Big Dog can recover from slipping and even being pushed. Its behavior is such that it approaches the infamous “uncanny valley” (`http://en.wikipedia.org/wiki/Uncanny_`

`Valley)`. The Boston Dynamics website (`http://bostondynamics.com/content/sec.php?section=BigDog`) features a video demonstration of Big Dog's abilities.

31.6.5. Three Legs

Three legged robots are not very common, especially since they have no biological counterparts. However, researchers at Virginia Tech's RoMeLa lab have developed a three legged robot STriDER that uses a "revolutionary" passive dynamic walking technique. STriDER is short for Self-excited Tripedal Dynamic Experimental Robot. STriDER sways until it can lift one leg, and using the other two as an A-frame, swing it in between the other two "stance" legs moving forward at a sixty degree angle. This patent pending "tripedal gait" is extremely energy efficient and requires minimal control. It also allows STriDER to easily change directions by changing the sequence of its steps. A video posted by the development team can be found at `http://www.youtube.com/watch?v=7XsaJwKKBYo&feature=related`.

31.6.6. Two Legs

Two legged robots have probably seen the most development dollars since humanoid robots have been envisioned since the very beginning of the field. Much of the development in passive dynamic walking has been done in this area. The design of a bipedal passive dynamic walker begins with the concept of a wheel with spokes. If the wheel is divided into sections, and all but two removed, we have what appears to be a set of legs. When the mass is properly distributed, the legs each act as inverted pendulums and the robots "rolls" through its steps.

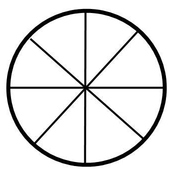


Figure 75

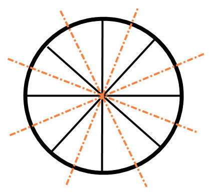


Figure 76

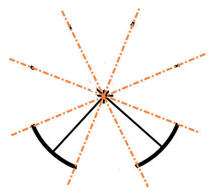


Figure 77

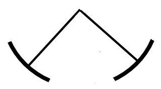


Figure 78

Further complexity can be added to the model by using knee joints to shorten the legs (allowing one to swing past the other without touching the ground) and ankle joints that can provide a "spring" to the step to add lost energy back into the system. For a more in-depth explanation of passive dynamics walking visit `http://www-personal.umich.edu/~artkuo/Passive_Walk/passive_walking.html`.

There are several robots that have used these concepts to achieve firsts in the field of robotics. "RunBot," developed in Germany and Scotland, broke the speed record per size for a robot in April 2006 by walking at 3.5 leg-lengths per second.[robotRecord] The Cornell Ranger, while not truly passive, is passive inspired and one of many robots that has more than two legs but is still classified as bipedal. When viewed from the side, Ranger appears to have only two legs, but it actually has four legs. These four legs act in pairs of two, qualifying it as bipedal but providing better lateral stability. On April 3, 2008 Ranger walked 9.07 kilometers without stopping, an unofficial record at that time (it has since been surpassed, according to the Cornell team, by Boston Dynamic's Big Dog).[Ranger]

One of the most successful companies at building bipedal robots over the years has been Honda. Their most recent model, ASIMO, is one of the few bipedal robots that appears humanoid, can climb stairs, and carries its own power supply. ASIMO can also change its gait in real time using Honda's i-WALK technology. This allows ASIMO to continuously change speeds and direction. The robot can walk up stairs and run up to four miles per hour.[Asimo]

31.6.7. One Leg

1980 and 1993 there was a lot of research in making one legged robots at the Massachusetts Institute of Technology (MIT). The MIT lab turned out a series of "MIT hoppers" that could balance themselves and traverse a path. The biggest challenge with the hoppers was that they could not stand still; they needed to continue hopping in order to maintain their balance. Researchers were able to build a 3-D One-Leg Hopper that "hopped in place, traveled at a specific rate, followed simple paths, and maintained balance when disturbed."[MITLegs] They also constructed a hopper named Uniroo that used an actuated tail to maintain its balance.[MITLegs]

31.7. References

1. [McGeer92]"McGeer_1992_Chap.pdf (application/pdf Object)." 10 Oct 2008 <http://www-personal.umich.edu/~shc/McGeer_1992_Chap.pdf>.
2. [Gait]"Gait Generation for Orthogonal Legged Robots." 13 Oct 2008 <http://ranier.hq.nasa.gov/telerobotics_Page/Technologies/0302.html>.
3. [McGeer90]"mcgeer_1990_passive_dynamic_walking.pdf (application/pdf Object)." 13 Oct 2008 <http://ruina.tam.cornell.edu/research/topics/locomotion_and_robotics/history/papers/mcgeer_1990_passive_dynamic_walking.pdf>.
4. [HistToys]"History of Passive Dynamics & Toys." 13 Oct 2008 <http://ruina.tam.cornell.edu/research/topics/locomotion_and_robotics/history.htm>.
5. [BigDog]"Boston Dynamics: The Leader in Lifelike Human Simulation." 13 Oct 2008 <http://bostondynamics.com/content/sec.php?section=BigDog>.
6. [robotRecord]"Robot Shatters Speed-Walking Record | LiveScience." 13 Oct 2008 <http://www.livescience.com/technology/060428_speedy_robot.html>.
7. [Ranger]"Cornell Ranger, walking robot." 13 Oct 2008 <http://ruina.tam.cornell.edu/research/topics/locomotion_and_robotics/papers/CornellRanger/index.html>.
8. [Asimo]"ASIMO - The World's Most Advanced Humanoid Robot." 13 Oct 2008 <http://asimo.honda.com/InsideASIMO.aspx>.
9. [MITLegs]"MIT Leg Lab Robots." 13 Oct 2008 <http://www.ai.mit.edu/projects/leglab/robots/robots-main.html>.

31.8. Modular and fractal robots

A robot is generally designed for one particular activity, or at best a few closely related activities. This of course isn't the ideal situation as many jobs require a large number of very different activities. For example, building the walls of a house and installing plumbing and electricity require very different tools and techniques. In practice this means you'll end up designing either several different robots or a very complicated one.

Modular robotics is one answer to such problems. Typical modular robots consist of a number of independent cubes, which can move and connect themselves in many different ways. Each of the cubes have their own power supply and intelligence and can have specific tools. When a problem is fed to its computer, it analyzes it and connects the cubes in such a way that it can solve the problem.

Ideally such a robot would consist of a very large number of very small identical cubes. This is similar to the way plants, animals or humans are consisting of many cells, which are practically identical, with the difference that specification would happen in software rather than in chemical composition.

The difference between fractal and modular robotics is that the former allows multiple sizes of cubes and the latter does not.

Most people are familiar with the fact that sufficient quantities of clay bricks (architectural bricks) or Lego bricks can be used to approximate nearly any shape, and so our introduction to modular robotics was in terms of cubes. Instead of making each cube independent, some researchers make pairs or triplets of cubes independent[38]. However, some roboticists point out that rhombic dodecahedron are, in some ways, superior to cubes[39].

The open-source Unified Simulator for Self-Reconfigurable Robots (USSR) can be used to simulate and test various modular robot designs and algorithms before building physical hardware.[40][41]

The open-source ARGoS robot simulator can be used to simulate large heterogeneous swarms of robots.[42]

Many people have built physical prototypes of modular robots.[43]

Some people use the phrase "modular robot" to indicate that its interfaces have been standardized so elements can be swapped out without custom-made adapters. (See Space

38 MIT Distributed Robotics Lab wiki: "The Self-Reconfiguring Robotic Molecule" ^{http://groups.csail.mit.edu/drl/wiki/index.php/The_Self-Reconfiguring_Robotic_Molecule}

39 "Rhombic Dodecahedron Shape for a Self-Assembling Robot" ^{http://www2.parc.com/spl/projects/modrobots/publications/pdf/rdtechreport.pdf} by Mark Yim, John Lamping, Eric Mao, J. Geoffrey Chase. (1997 ?)

40 David Christensen, David Brandt, Kasper Stoy, and Ulrik Pagh Schultz. "A Unified Simulator for Self-Reconfigurable Robots" ^{http://modular.mmmi.sdu.dk/w/upload/5/53/Schultz-iros08.pdf} .

41 GitHub: unified simulator for self-reconfigurable robots ^{https://github.com/mimog/ussr} ; Unified Simulator for Self-Reconfigurable Robots (USSR) ^{http://modular.mmmi.sdu.dk/wiki/USSR} at the USD Modular Robotics Research Lab wiki.

42 ARGoS ^{http://iridia.ulb.ac.be/argos/}

43 "Self Reconfigurable Modular Technology" ^{http://www.selfreconfigurable.com/}

Transport and Engineering Methods/Advanced Manufacturing#Modular Robot[44], Seed Factories/Basics#Modular Design[45], Seed Factories/Notes3#Modular Design:[46], Seed Factories/Notes8#1A.3.4 Supply Internal Transport[47], etc.). Such a modular robot has:

- the wrist of a robot arm has a standard mounting pattern so several "hands" with different kinds of tools can be swapped out;
- the shoulder of a robot arm has a standard mounting pattern that can be attached to a fixed pillar or various motion bases;
- a standardized power connector makes it easy to swap between a small lightweight battery, a heavier longer-lasting battery, and mains power;
- etc.

31.9. References

31.10. Further reading

w:Self-Reconfiguring Modular Robotics[48]

See AboutAI.net[49] and Fractal Robots[50] and modular robots[51] for more information in this kind of robots.

- USD Modular Robotics Research Lab wiki[52]
- ANAT modular robotic technology[53]
- Molecubes For Everyone wiki[54]
  - Festo: "Molecubes"[55]
  - Sourceforge: "Molecubes"[56]
- hplusroadmap wiki: "matter compiler"[57]
- "SuperBot"[58] at ISI USC (Information Sciences Institute of University of Southern California)
- "M-TRAN (Modular Transformer)"[59] at AIST and Tokyo-Tech
- REPY-1 modules: open-source design[60]

44 https://en.wikibooks.org/wiki/Space%20Transport%20and%20Engineering%20Methods%2FAdvanced%20Manufacturing%23Modular%20Robot
45 https://en.wikibooks.org/wiki/Seed%20Factories%2FBasics%23Modular%20Design
46 https://en.wikibooks.org/wiki/Seed%20Factories%2FNotes3%23Modular%20Design%3A
47 https://en.wikibooks.org/wiki/Seed%20Factories%2FNotes8%231A.3.4%20Supply%20Internal%20Transport
48 https://en.wikipedia.org/wiki/Self-Reconfiguring%20Modular%20Robotics
49 http://www.aboutai.net/DesktopDefault.aspx?tabindex=1&tabid=2&article=aa040400a.htm
50 http://www.autopenhosting.org/robots/
51 http://david.carybros.com/html/robot_links.html#modular
52 http://modular.mmmi.sdu.dk/wiki/
53 http://roboticsdesign.qc.ca/our-technology
54 http://www.molecubes.org/
55 http://www.festo.com/cms/en_corp/9774.htm
56 http://sourceforge.net/projects/molecubes/
57 http://heybryan.org/mediawiki/index.php/matter_compiler
58 http://www.isi.edu/robots/superbot/
59 http://unit.aist.go.jp/is/dsysd/mtran3/
60 http://www.thingiverse.com/thing:688

- ”Modular robotics & Robot locomotion Group”[61] at the School of Mechanical and Production Engineering, Nanyang Technological University.
- ”Formica: Affordable, open source swarm robotics”[62]
- Modular Robotics: ”Cubelets”[63]
- Modular Robotics Labs at The University of Southern Denmark`http://www.youtube.com/user/mrlusd`
  - Østergaard, Kassow, Beck, and Lund. [ftp://sccn.ucsd.edu/pub/robot.pdf ”Design of the ATRON lattice-based self-reconfigurable robot”].

61 `http://www.mae.ntu.edu.sg/AboutMAE/Divisions/RRC/Pages/Activities.aspx`
62 `http://warrantyvoidifremoved.com/formica`
63 `http://www.modrobotics.com/cubelets`

32. The LEGO World

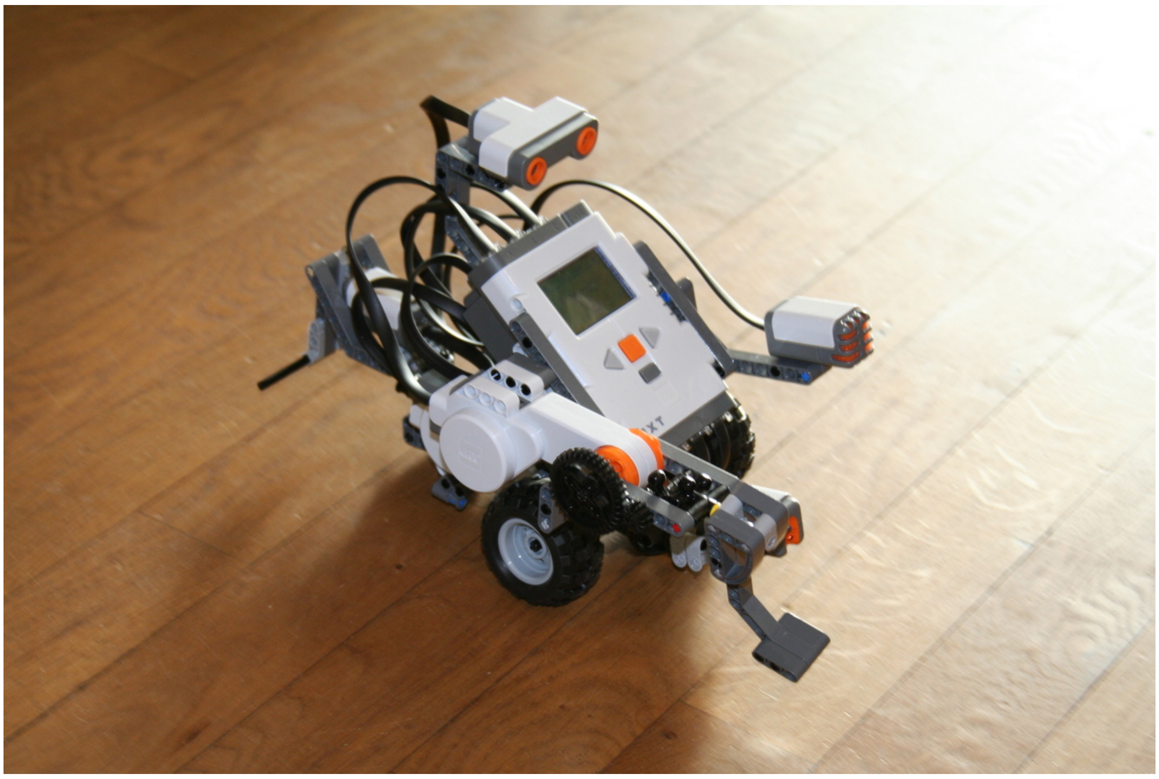


Figure 79

32.1. Why Lego?

Legos have become a popular robotics resource, primarily as an educational tool, but also as building materials for fast and easy prototyping. Legos are specifically designed for ease of use, with snap together parts and pre-placed holes. This makes them ideally suited for use in projects where reconfiguration is necessary. With a basic kit of parts, many different robotic machines and mechanisms can be created, tested, modified, disassembled and recreated easily without any damage to the building materials. The pieces are also standardized, so designs can be easily documented and rebuilt. Because of the popularity of Legos, there is also a huge wealth of third party resources available – hardware, software, instructional material, and challenges.

32.2. History

The origins of the Lego Mindstorms[1] Robotics kits trace back to Seymour Papert's[2] book *Mindstorms: Children, Computers, and Powerful Ideas* , in which Papert proposed that rather than using computers to provide exercises for children – "the computer programming the child", the situation should be reversed and the child given control – "the child programs the computer". Papert thought that in this way, the child would gain a more active role in building his own knowledge in response to a recognizable personal purpose rather than simply listening to explanations. After co-founding the MIT Artificial Intelligence Lab, Papert developed the programming language "Logo[3]" as a tool to enable children to use simple instructions to control robotic 'turtles'.

In the mid-1980's, while working at the MIT Media Lab, Mitchel Resnick and Steve Ocko created a control box to interface Logo with Lego Technic elements along with motors lights and sensors. This setup was marketed as "LEGO TC Logo" and became a popular educational tool and marked the beginning of a partnership between Lego and the MIT Logo researchers. A second version was released in 1993 as the "Control Lab"[4]

In the mid-1990's, MIT Media Lab researcher Fred Martin developed the MIT programmable brick. This brick was programmed using a Logo based software but was not required to be connected to the computer with wires and so was able to be more mobile. This design was the basis for the Lego RCX brick, released in 1998. The Lego RCX was the first of the Lego Mindstorms products, which are named after the Papert book. Mindstorms robotics kits quickly became an extremely popular educational resource, being used for robotics classes and competitions around the world.[5]

In 2006, the NXT became the successor to the RCX. The NXT has an extra sensor port and Bluetooth communication and the basic kit adds an ultrasonic sensor as well as built-in rotation sensors on all the motors.

32.3. Standard Components

w:Lego_Mindstorms[6] The RCX kits come with the following standard components as well as a variety of building pieces:

- 2 Motors
- 1 Light Sensor
- 2 Lamps
- Built in IR Communication

The following Lego accessories are available for the RCX:

- Rotation Sensor

1 `https://en.wikipedia.org/wiki/Lego_Mindstorms`
2 `https://en.wikipedia.org/wiki/Seymour_Papert`
3 `https://en.wikipedia.org/wiki/Logo_%28programming_language%29`
4 Education and Technology: An Encyclopedia, By Ann Kovalchick, Kara Dawson, 2004, pgs 421-426
5 `http://el.media.mit.edu/logo-foundation/pubs/logoupdate/v7n1/v7n1-pbrick.html`
6 `https://en.wikipedia.org/wiki/Lego_Mindstorms`

- Temperature Sensor
- Sound Sensor

The NXT kits come with the following standard components as well as a variety of building pieces:

- 3 Servo Motors (with built-in rotation sensors)
- 1 Light Sensor
- 1 Touch Sensor
- 1 Ultrasonic Sensor
- 1 Sound Sensor
- Built in Bluetooth Communication

The NXT kit also comes with adaptors that allow any of the RCX components to be used with the NXT.

32.4. Programming Options

There are many options available for programming the Mindstorms bricks. The standard NXT kits may be purchased with either the NXT-G or Robolab programming software. Both are LabVIEW based visual programming languages – Robolab allows more advanced programming, and NXT-G is a bit more beginner-friendly. There are also many other programming languages available.

32.4.1. NXT-G

NXT-G (Windows, Mac)

- Pros
  - Easy to quickly create simple programs
  - Programming flow is easy to see
  - Included in standard kit
- Cons
  - Somewhat limited capabilities
  - Integers only – floating point numbers not supported
  - Each basic math operation (addition, subtraction, multiplication, division) requires a separate block
  - Comparatively slow execution speeds
  - High memory usage.

32.4.2. Robolab

Robolab (Windows, Mac)

- Pros
  - Fairly easy to use
  - Fairly advanced programming possible

  - Very similar to LabVIEW environment
  - Included in standard educational kit
- Cons
  - Block connections can become confusing
  - No good method for creating block set functions for reuse

32.4.3. RobotC

RobotC[7](Windows)

- Pros
  - Fast execution
  - Advanced programming
- Cons
  - Text based language is harder for beginners
  - Must be bought separately from kit.

32.4.4. LabVIEW Toolkit

LabVIEW Toolkit[8] (Windows, Mac)

- Pros
  - Free (restrictions apply)
  - Can create blocks for use with NXT-G programming
  - Advanced data analysis
  - Common industry programming environment
- Cons
  - Somewhat harder for beginners
  - Advanced programming more limited than text based languages

32.4.5. BricxCC

BricxCC[9] (Windows)

- Free Windows IDE that supports many programming languages
  - NQC (C-based language for the RCX)
  - NXC/NBC (C-based and assembly code for the NXT)
  - C/C++
  - Pascal
  - pbForth
  - leJOS (Java)

7 `http://www.robotc.net`
8 `http://www.ni.com/academic/mindstorms/`
9 `http://bricxcc.sourceforge.net/`

32.5. Third Party Accessories

32.5.1. HiTechnic

HiTechnic[10] accessories are packaged in standard Lego NXT sensor cases and can be purchased through Lego:

- 3-axis Accelerometer
- Gyro Sensor
- Color Sensor
- Compass Sensor
- RFID Sensor
- IRLink Sensor
- IRSeeker Sensor
- Electro Optical Proximity Detector
- Touch Sensor Multiplexor
- Prototyping Boards

32.5.2. Vernier

Any Vernier[11] sensors can be connected to Lego Mindstorms through an adapter cable.

- 25-g Accelerometer
- Barometer
- Charge Sensor
- Colorimetric
- Conductivity Probe
- Current Probe
- Differential Voltage Probe
- Dissolved Oxygen Probe
- Dual-Range Force Sensor Probe
- Electrode Amplifier
- Extra Long Temperature Probe
- Flow Rate Sensor
- Force Plate
- Gas Pressure Sensor
- Hand Dynamo meter
- Instrumentation Amplifier
- Light Sensor
- Low-g Accelerometer
- Magnetic Field Sensor
- O2 Gas Sensor
- ORP Sensor
- pH Sensor
- Relative Humidity Sensor

10 `http://www.hitechnic.com`
11 `http://www.vernier.com/nxt/`

- Salinity Sensor
- Soil Moisture Sensor
- Sound Level Meter
- Stainless Steel Temperature Probe
- Surface Temperature Sensor
- Thermocouple
- Turbidity Sensor
- UVA Sensor
- UVB Sensor

32.5.3. Mind Sensors

Mind Sensors[12] has the following available accessories:

- Sony PS2 Controller Interface
- Vision Subsystem
- 8-channel Servo Controller
- Multi-sensitivity Acceleration Sensor
- Dual Infrared Obstacle Detector
- High/Low Range IR Distance Sensors
- Real time Clock
- Pneumatic Pressure Sensor
- Magnetic Compass
- RCX Motor Multiplexor
- RCX Sensor Multiplexor

Hi

32.6. Lego Robotics Events, Challenges, and Acheivements

First Lego League[13]`http://www.firstlegoleague.org` hosts regional competitions for middle school students. A new competition board is created each year with theme-based tasks for the robots to accomplish within a set time limit. The students also are required to research real-world issues related to the selected theme.

RoboCup Junior[14]`http://www.robocupjunior.org` is a robotics soccer competition. Lego Mindstorms was originally the primary construction kit used by most teams. Teams are not limited to the use of Legos, so more advanced teams also use more advanced technology.

The Botball[15]`http://www.botball.org` competition utilizes Lego pieces as the building materials although more advanced controllers and sensors are used. The Lego RCX served as a primary controller in early years of the competition.

12 `http://www.mindsensors.com`
13 `https://en.wikipedia.org/wiki/FIRST_Lego_League`
14 `https://en.wikipedia.org/wiki/RoboCup_Junior`
15 `https://en.wikipedia.org/wiki/Botball`

To commemorate the 10th anniversary of Lego Mindstorms, the High Altitude Lego Extravaganza[16] sent 9 NXT controlled experiments over 99,500 ft. One NXT was released in order to make the longest recorded NXT freefall of 80 seconds before releasing a parachute.

32.7. References

32.8. LEGO Robots

LEGO[17] has created the Robotics Invention System[18], which is essentially a LEGO set which may be used to build and prototype robot designs.

The main components provided are,

- 2 Motors
- 2 Touch sensors
- Light detector
- RCX (Robotic Command Explorer) controller

These can be used with Technic Lego such as cogs, axles, gears and other components to build reasonably complex systems.

32.9. Interface

The robot interfaces to a PC through the RCX controller using either an RS-232[19] serial port and cable, or an infrared (IR) transceiver connected to a USB[20] port. The program can be executed through the connection in real time or downloaded to and stored on the RCX controller. Downloading and storing the program allows the robot to run autonomously without the need to be tethered to the computer.

32.10. Programming and Software

Lego provide their own GUI[21] Software `http://mindstorms.lego.com/eng/products/ris/rissoft.asp` to program the RCX controller. The versatility of the LEGO system, and the added sophistication you get with scripting languages has meant other people have developed their own programming languages to control the RCX, namely,

- NQC[22] - Not quite C: A C derived scripting language.

16 `http://www.unr.edu/nevadasat/hale/`
17 `http://en.wikipedia.org/wiki/LEGOs`
18 `http://mindstorms.lego.com`
19 `http://en.wikipedia.org/wiki/RS-232`
20 `http://en.wikipedia.org/wiki/Universal_Serial_Bus`
21 `http://en.wikipedia.org/wiki/GUI`
22 `http://bricxcc.sourceforge.net/nqc/`

- lejos[23] - Robots control in Java.

----

23 `http://lejos.sourceforge.net`

33. Resources

33.1. Other Wikibooks

- Advanced Robotics Book[1]
- Electronics[2]
- Embedded Systems[3]
- Embedded Control Systems Design[4]
- Theoretical Mechanics[5]
- System Troubleshooting[6]
- Robotics Kinematics and Dynamics[7]: a more theoretical approach.
- Electric Motors And Generators[8]: theory and details of what goes on inside a motor
- Practical Electronics/Stepper Motors[9]
- Power Electronics[10]

33.2. External links

w:Robot[11] w:Microbotics[12] w:Mechatronics[13]

- Robotics video tutorial[14]
- Robotics Education Website[15]
- Robotics for hobbyists[16]
- McComb, Gordon 2000 ”The Robot Builder’s bonanza 2nd ed.” ISBN 0-07-136296-7 -- One of, if not the, best book on home-built mobile robotics.
- ”Controlling The Real World With Computers” by Joe Reeder[17]

1 https://en.wikibooks.org/wiki/Advanced%20Robotics%20Book
2 https://en.wikibooks.org/wiki/Electronics
3 https://en.wikibooks.org/wiki/Embedded%20Systems
4 https://en.wikibooks.org/wiki/Embedded%20Control%20Systems%20Design
5 https://en.wikibooks.org/wiki/Theoretical%20Mechanics
6 https://en.wikibooks.org/wiki/System%20Troubleshooting
7 https://en.wikibooks.org/wiki/Robotics%20Kinematics%20and%20Dynamics
8 https://en.wikibooks.org/wiki/Electric%20Motors%20And%20Generators
9 https://en.wikibooks.org/wiki/Practical%20Electronics%2FStepper%20Motors
10 https://en.wikibooks.org/wiki/Power%20Electronics
11 https://en.wikipedia.org/wiki/Robot
12 https://en.wikipedia.org/wiki/Microbotics
13 https://en.wikipedia.org/wiki/Mechatronics
14 http://www.expertcore.org/viewtopic.php?f=73&t=335
15 http://www.razorrobotics.com/
16 http://www.quacktu.com/index.html
17 http://learn-c.com/

- "Get Started in Robotics"[18]
- `http://www.robotix.in` tutorials on robotics
- CLARAty http://claraty.jpl.nasa.gov[19] reusable robotic software framework. Includes many miscellaneous algorithms used in robotics: pose estimation, navigation, path planning, machine vision, coordinate transforms, locomotion inverse kinematics, Bayes networks, etc.
- Robotics[20] at stack exchange
- "The TAM: a device for task abstraction in swarm robotics research"`http://iridia.ulb.ac.be/supp/IridiaSupp2012-002/`

33.3. Free Wikis on Robotics

- OpenServo wiki[21]
- "The Robot Group Inc."[22] of Austin TX: many artistic and entertainment robots
- Open Circuits wiki: motor drivers[23]

33.4. University and College Sites

- Connecticut College[24]
- MIT Open Course Ware[25] - MIT has published many of its courses online as the OCW project. Some of these provide practically everything, others list course books. Probably one of the most reliable and extensive sources of information on the internet.

33.5. Papers

This section list sites that carry papers on various subjects relevant to robotics.

cs.conncoll.edu/Parker/_papers.html[26]

- Co-Evolving Team Capture Strategies for Dissimilar Robots
- Fitness Biasing to Produce Adaptive Gaits for Hexapod Robots
- Competing Sample Sizes for the Co-Evolution of Heterogeneous Agents
- Partial Recombination for the Co-Evolution of Model Parameters
- Varying Sample Sizes for the Co-Evolution of Heterogeneous Agents
- Punctuated Anytime Learning for Evolving Multi-Agent Capture Strategies

18 `http://mdubuc.freeshell.org/Robotics/GetStarted.html`
19 `http://claraty.jpl.nasa.gov`
20 `http://area51.stackexchange.com/proposals/40020/robotics`
21 `http://openservo.com/`
22 `http://wiki.therobotgroup.org/`
23 `http://opencircuits.com/Motor_driver`
24 `http://cs.conncoll.edu/Parker/`
25 `http://ocw.mit.edu/index.html`
26 `http://cs.conncoll.edu/Parker/_papers.html`

- Cyclic Genetic Algorithms for Evolving Multi-Loop Control Programs
- Continuous Power Supply for a Robot Colony
- Comparison of Sampling Sizes for the Co-Evolution of Cooperative Agents
- Evolving Towers in a 3-Dimensional Simulated Environment
- Evolving Neural Networks for Hexapod Leg Controllers
- Learning Adaptive Leg Cycles Using Fitness Biasing
- Cyclic Genetic Algorithms for Stiquito Locomotion
- Evolving Gaits for the Lynxmotion Hexapod II Robot
- Sampling the Nature of a Population: Punctuated Anytime Learning for Co-Evolving a Team
- Punctuated Anytime Learning for Hexapod Gait Generation
- Learning Area Coverage Using the Co-Evolution of Model Parameters
- Evolving Neural Network Controllers to Produce Leg Cycles for Gait Generation
- Punctuated Anytime Learning for Evolving a Team
- Evolving Cyclic Control for a Hexapod Robot Performing Area Coverage
- The Incremental Evolution of Gaits for Hexapod Robots
- Gait Evolution for a Hexapod Robot
- Learning Control Cycles for Area Coverage with Cyclic Genetic Algorithms
- Co-Evolving Model Parameters for Anytime Learning in Evolutionary Robotics
- Evolving Leg Cycles to Produce Hexapod Gaits
- Punctuated Anytime Learning for Evolutionary Robotics
- The Co-Evolution of Model Parameters and Control Programs in Evolutionary Robotics
- Adaptive Hexapod Gait Control Using Anytime Learning with Fitness Biasing
- Generating Arachnid Robot Gaits with Cyclic Genetic Algorithms
- Locomotion Control Cycles Adapted for Disabilities in Hexapod Robots
- Metachronal Wave Gait Generation for Hexapod Robots
- Evolving Hexapod Gaits Using a Cyclic Genetic Algorithm
- Learning Gaits for the Stiquito
- Using Cyclic Genetic Algorithms to Reconfigure Hardware Controllers for Robots
- Cyclic Genetic Algorithms for the Locomotion of Hexapod Robots
- Genetic Algorithms for the Development of Real-Time Multi-Heuristic Search Strategies

----

34. Contributors

Edits	User
5	Abozzay[1]
7	AdRiley[2]
1	AdamN~enwikibooks[3]
85	Adrignola[4]
1	Afaqueazam[5]
1	Alex S~enwikibooks[6]
2	Animalinside110[7]
2	Anudhyan[8]
2	Atcovi[9]
1	Az1568[10]
1	Bigr34[11]
1	Billinghurst[12]
1	Billmania[13]
1	Bmg262[14]
2	Bobstar1020[15]
4	Bserniak[16]
6	CVince13[17]
2	Canadiansteve[18]
3	Carmicp[19]
1	Charles Merriam[20]

1 https://en.wikibooks.org/w/index.php%3ftitle=User:Abozzay&action=edit&redlink=1
2 https://en.wikibooks.org/wiki/User:AdRiley
3 https://en.wikibooks.org/w/index.php%3ftitle=User:AdamN~enwikibooks&action=edit&redlink=1
4 https://en.wikibooks.org/wiki/User:Adrignola
5 https://en.wikibooks.org/w/index.php%3ftitle=User:Afaqueazam&action=edit&redlink=1
6 https://en.wikibooks.org/wiki/User:Alex_S~enwikibooks
7 https://en.wikibooks.org/w/index.php%3ftitle=User:Animalinside110&action=edit&redlink=1
8 https://en.wikibooks.org/w/index.php%3ftitle=User:Anudhyan&action=edit&redlink=1
9 https://en.wikibooks.org/wiki/User:Atcovi
10 https://en.wikibooks.org/wiki/User:Az1568
11 https://en.wikibooks.org/w/index.php%3ftitle=User:Bigr34&action=edit&redlink=1
12 https://en.wikibooks.org/wiki/User:Billinghurst
13 https://en.wikibooks.org/w/index.php%3ftitle=User:Billmania&action=edit&redlink=1
14 https://en.wikibooks.org/w/index.php%3ftitle=User:Bmg262&action=edit&redlink=1
15 https://en.wikibooks.org/w/index.php%3ftitle=User:Bobstar1020&action=edit&redlink=1
16 https://en.wikibooks.org/w/index.php%3ftitle=User:Bserniak&action=edit&redlink=1
17 https://en.wikibooks.org/w/index.php%3ftitle=User:CVince13&action=edit&redlink=1
18 https://en.wikibooks.org/w/index.php%3ftitle=User:Canadiansteve&action=edit&redlink=1
19 https://en.wikibooks.org/w/index.php%3ftitle=User:Carmicp&action=edit&redlink=1
20 https://en.wikibooks.org/w/index.php%3ftitle=User:Charles_Merriam&action=edit&redlink=1

3	Chuckhoffmann[21]
1	Coffeenut[22]
3	CommonsDelinker[23]
1	Computer tom[24]
1	Daleh~enwikibooks[25]
51	Dallas1278[26]
1	Darklama[27]
96	DavidCary[28]
1	Davlil~enwikibooks[29]
2	Defender[30]
5	Derbeth[31]
43	Dirk Hünniger[32]
22	Doug.beach[33]
3	ElieDeBrauwer[34]
1	Eugenio Hansen, OFS[35]
4	Glaisher[36]
2	GraphiteFingers[37]
1	HethrirBot[38]
2	Jakec[39]
3	JamesCrook[40]
1	Jcwf[41]
1	Jdv26c[42]
1	Jggmb7[43]
62	Jguk[44]
5	Jjbnq2[45]

21 https://en.wikibooks.org/wiki/User:Chuckhoffmann
22 https://en.wikibooks.org/w/index.php%3ftitle=User:Coffeenut&action=edit&redlink=1
23 https://en.wikibooks.org/wiki/User:CommonsDelinker
24 https://en.wikibooks.org/wiki/User:Computer_tom
25 https://en.wikibooks.org/wiki/User:Daleh~enwikibooks
26 https://en.wikibooks.org/w/index.php%3ftitle=User:Dallas1278&action=edit&redlink=1
27 https://en.wikibooks.org/wiki/User:Darklama
28 https://en.wikibooks.org/wiki/User:DavidCary
29 https://en.wikibooks.org/w/index.php%3ftitle=User:Davlil~enwikibooks&action=edit&redlink=1
30 https://en.wikibooks.org/wiki/User:Defender
31 https://en.wikibooks.org/wiki/User:Derbeth
32 https://en.wikibooks.org/wiki/User:Dirk_H%25C3%25BCnniger
33 https://en.wikibooks.org/w/index.php%3ftitle=User:Doug.beach&action=edit&redlink=1
34 https://en.wikibooks.org/w/index.php%3ftitle=User:ElieDeBrauwer&action=edit&redlink=1
35 https://en.wikibooks.org/w/index.php%3ftitle=User:Eugenio_Hansen,_OFS&action=edit&redlink=1
36 https://en.wikibooks.org/wiki/User:Glaisher
37 https://en.wikibooks.org/w/index.php%3ftitle=User:GraphiteFingers&action=edit&redlink=1
38 https://en.wikibooks.org/wiki/User:HethrirBot
39 https://en.wikibooks.org/wiki/User:Jakec
40 https://en.wikibooks.org/wiki/User:JamesCrook
41 https://en.wikibooks.org/wiki/User:Jcwf
42 https://en.wikibooks.org/w/index.php%3ftitle=User:Jdv26c&action=edit&redlink=1
43 https://en.wikibooks.org/w/index.php%3ftitle=User:Jggmb7&action=edit&redlink=1
44 https://en.wikibooks.org/wiki/User:Jguk
45 https://en.wikibooks.org/w/index.php%3ftitle=User:Jjbnq2&action=edit&redlink=1

11	Jomegat[46]
3	Kd5bjo[47]
1	LIL DEBUL[48]
1	Lagoset[49]
1	Mabdul[50]
6	Marcushan˜enwikibooks[51]
22	Mastraut[52]
2	Material˜enwikibooks[53]
1	Mathonius[54]
1	Meatmanek[55]
1	MerlLinkBot[56]
1	Metric1000[57]
52	Mike.lifeguard[58]
2	MrPyro[59]
1	Mseethar[60]
4	Nsoyeblcyha[61]
14	Omegatron[62]
3	Otto˜enwikibooks[63]
3	Panic2k4[64]
367	Patrik[65]
1	Peratflexibilityenvelope[66]
70	Pi zero[67]
8	Piyoosh[68]
2	Prandall[69]
15	QuiteUnusual[70]

46 https://en.wikibooks.org/wiki/User:Jomegat
47 https://en.wikibooks.org/w/index.php%3ftitle=User:Kd5bjo&action=edit&redlink=1
48 https://en.wikibooks.org/w/index.php%3ftitle=User:LIL_DEBUL&action=edit&redlink=1
49 https://en.wikibooks.org/w/index.php%3ftitle=User:Lagoset&action=edit&redlink=1
50 https://en.wikibooks.org/wiki/User:Mabdul
51 https://en.wikibooks.org/wiki/User:Marcushan~enwikibooks
52 https://en.wikibooks.org/w/index.php%3ftitle=User:Mastraut&action=edit&redlink=1
53 https://en.wikibooks.org/w/index.php%3ftitle=User:Material~enwikibooks&action=edit&redlink=1
54 https://en.wikibooks.org/wiki/User:Mathonius
55 https://en.wikibooks.org/w/index.php%3ftitle=User:Meatmanek&action=edit&redlink=1
56 https://en.wikibooks.org/wiki/User:MerlLinkBot
57 https://en.wikibooks.org/wiki/User:Metric1000
58 https://en.wikibooks.org/wiki/User:Mike.lifeguard
59 https://en.wikibooks.org/w/index.php%3ftitle=User:MrPyro&action=edit&redlink=1
60 https://en.wikibooks.org/w/index.php%3ftitle=User:Mseethar&action=edit&redlink=1
61 https://en.wikibooks.org/wiki/User:Nsoyeblcyha
62 https://en.wikibooks.org/wiki/User:Omegatron
63 https://en.wikibooks.org/wiki/User:Otto~enwikibooks
64 https://en.wikibooks.org/wiki/User:Panic2k4
65 https://en.wikibooks.org/wiki/User:Patrik
66 https://en.wikibooks.org/w/index.php%3ftitle=User:Peratflexibilityenvelope&action=edit&redlink=1
67 https://en.wikibooks.org/wiki/User:Pi_zero
68 https://en.wikibooks.org/w/index.php%3ftitle=User:Piyoosh&action=edit&redlink=1
69 https://en.wikibooks.org/w/index.php%3ftitle=User:Prandall&action=edit&redlink=1
70 https://en.wikibooks.org/wiki/User:QuiteUnusual

1	Ravikumaradelolla[71]
12	Recent Runes[72]
3	Reece[73]
1	Roboticist~enwikibooks[74]
2	Rotlink[75]
1	Ruy Pugliesi[76]
5	Sahodaran~enwikibooks[77]
49	SamEEE[78]
1	Sebastian Goll[79]
3	Shaunpress~enwikibooks[80]
69	Sleepisfortheweak[81]
1	Snarius~enwikibooks[82]
1	Sonia[83]
1	Sparkega[84]
4	Starryknight64[85]
1	Syum90[86]
1	TERdON~enwikibooks[87]
10	Tel36f[88]
8	Tigmaster[89]
52	ToyMaker[90]
6	Tring[91]
1	Umar420e[92]
1	Uncle G[93]
1	Van der Hoorn[94]

71 https://en.wikibooks.org/w/index.php%3ftitle=User:Ravikumaradelolla&action=edit&redlink=1
72 https://en.wikibooks.org/wiki/User:Recent_Runes
73 https://en.wikibooks.org/wiki/User:Reece
74 https://en.wikibooks.org/w/index.php%3ftitle=User:Roboticist~enwikibooks&action=edit&redlink=1
75 https://en.wikibooks.org/w/index.php%3ftitle=User:Rotlink&action=edit&redlink=1
76 https://en.wikibooks.org/wiki/User:Ruy_Pugliesi
77 https://en.wikibooks.org/w/index.php%3ftitle=User:Sahodaran~enwikibooks&action=edit&redlink=1
78 https://en.wikibooks.org/wiki/User:SamEEE
79 https://en.wikibooks.org/wiki/User:Sebastian_Goll
80 https://en.wikibooks.org/w/index.php%3ftitle=User:Shaunpress~enwikibooks&action=edit&redlink=1
81 https://en.wikibooks.org/wiki/User:Sleepisfortheweak
82 https://en.wikibooks.org/wiki/User:Snarius~enwikibooks
83 https://en.wikibooks.org/wiki/User:Sonia
84 https://en.wikibooks.org/w/index.php%3ftitle=User:Sparkega&action=edit&redlink=1
85 https://en.wikibooks.org/w/index.php%3ftitle=User:Starryknight64&action=edit&redlink=1
86 https://en.wikibooks.org/wiki/User:Syum90
87 https://en.wikibooks.org/wiki/User:TERdON~enwikibooks
88 https://en.wikibooks.org/w/index.php%3ftitle=User:Tel36f&action=edit&redlink=1
89 https://en.wikibooks.org/w/index.php%3ftitle=User:Tigmaster&action=edit&redlink=1
90 https://en.wikibooks.org/wiki/User:ToyMaker
91 https://en.wikibooks.org/w/index.php%3ftitle=User:Tring&action=edit&redlink=1
92 https://en.wikibooks.org/w/index.php%3ftitle=User:Umar420e&action=edit&redlink=1
93 https://en.wikibooks.org/wiki/User:Uncle_G
94 https://en.wikibooks.org/wiki/User:Van_der_Hoorn

1	Webaware[95]
2	Whiteknight[96]
1	WikiY˜enwikibooks[97]
1	Wikibooks is Communism[98]
1	Wutsje[99]
3	Xania[100]
6	Xenodevil[101]
2	Y0ssarian˜enwikibooks[102]
1	Yamamoto Ichiro[103]

95 https://en.wikibooks.org/wiki/User:Webaware
96 https://en.wikibooks.org/wiki/User:Whiteknight
97 https://en.wikibooks.org/w/index.php%3ftitle=User:WikiY~enwikibooks&action=edit&redlink=1
98 https://en.wikibooks.org/w/index.php%3ftitle=User:Wikibooks_is_Communism&action=edit&redlink=1
99 https://en.wikibooks.org/wiki/User:Wutsje
100 https://en.wikibooks.org/wiki/User:Xania
101 https://en.wikibooks.org/wiki/User:Xenodevil
102 https://en.wikibooks.org/w/index.php%3ftitle=User:Y0ssarian~enwikibooks&action=edit&redlink=1
103 https://en.wikibooks.org/wiki/User:Yamamoto_Ichiro

List of Figures

- EPL: Eclipse Public License. `http://www.eclipse.org/org/documents/epl-v10.php`

Copies of the GPL, the LGPL as well as a GFDL are included in chapter Licenses[104]. Please note that images in the public domain do not require attribution. You may click on the image numbers in the following table to open the webpage of the images in your webbrower.

104 Chapter 35 on page 241

1	User:Patrik[105]	
2	Darklama, Patrik	
3	Patrik	
4	Patrik	
5	Az1568, Patrik	
6	Az1568, Patrik	
7	jjbeard	
8	Aka, Amada44, Angeloleithold, Glenn, JarektBot, Javier Carro, Leotard, Mardus, Mindmatrix, Paddy, Skim, Wdwd, Wwongbc, ⊠⊠	
9	32bitmaschine, BotMultichill, David.Monniaux, JarektBot	
10	Jared C. Benedict (user Redjar[106])	GFDL
11	Arthur Clarke[107] at English Wikipedia[108]	
12	Az1568, Patrik, Webaware	
13	Sparkega	
14	SamEEE	
15	SamEEE	
16	SamEEE	
17	Patrik	
18	Az1568, Patrik	
19	Rama[109], Rama[110]	CC-BY-SA-2.0
20		
21		
22	Alno, BotMultichill, Emijrpbot, Hazard-Bot, JarektBot, Nojhan, Putnik, Snek01, Waldir	
23		
24	Tinycircuits[111], Tinycircuits[112]	CC-BY-SA-3.0
25	Christophe.Finot, JarektBot, MGA73bot2, Saperaud commonswiki, Zaphod	
26	AnonMoos, Emijrpbot, Glenn, Hazard-Bot, JarektBot, Juiced lemon, MADe, SchlurcherBot	
27	Sleepisfortheweak	
28	Adrignola, Sleepisfortheweak	
29	Sleepisfortheweak	
30	Sleepisfortheweak	
31	Sleepisfortheweak	
32	Recent Runes, Sleepisfortheweak	
33	QuiteUnusual, Sleepisfortheweak	
34	Sleepisfortheweak	
35	Josh Vance	
36		
37	Jdv26c	

105 http://en.wikibooks.org/wiki/User:Patrik
106 http://en.wikipedia.org/wiki/User:Redjar
107 http://en.wikipedia.org/wiki/User:Arthur_Clarke
108 http://en.wikipedia.org/wiki/
109 http://commons.wikimedia.org/wiki/User:Rama
110 https://commons.wikimedia.org/wiki/User:Rama
111 http://commons.wikimedia.org/w/index.php?title=User:Tinycircuits&action=edit&redlink=1
112 https://commons.wikimedia.org/w/index.php?title=User:Tinycircuits&action=edit&redlink=1

38		
39		
40		
41		
42		
43	Otto enwikibooks	
44	Otto enwikibooks	
45	Otto enwikibooks	
46	NASA	PD
47	Bricktop, Carbenium, JarektBot	
48	Bricktop, Harlock81, JarektBot	
49	NASA/JPL/Thomas “Dutch” Slager	PD
50	Bricktop, Emijrpbot, Harlock81, Hazard-Bot, JarektBot, Vearthy	
51	BJ Axel, Bricktop, Campani commonswiki, ComputerHotline, Dodo, Harlock81, JarektBot, Li-sung, Soerfm	
52	Bmg262	
53	Bmg262	
54	Bmg262	
55	Adrignola, Lcarsbot, Mike.lifeguard, Patrik	
56	Larry D. Moore[113]	GFDL
57	User:Patrik[114]	
58	Pacaro[115] at English Wikipedia[116]	
59	Luckas Blade, Mike’s bot account, Mike.lifeguard, Patrik	
60	Jiuguang Wang[117], Jiuguang Wang[118]	GFDL
61	Mike’s bot account, Mike.lifeguard, Patrik	
62	Az1568, Patrik	
63	BJ Axel, Bricktop, Campani commonswiki, ComputerHotline, Dodo, Harlock81, JarektBot, Li-sung, Soerfm	
64	Bricktop, Emijrpbot, FrescoBot, Harlock81, Hazard-Bot, Huntster, JarektBot, Julian Herzog	
65	NASA/JPL/Cornell University, Maas Digital LLC	PD
66	Chris Vincent	
67	Chris Vincent	
68	Chris Vincent	
69	Chris Vincent	
70	Chris Vincent	
71	Chris Vincent	
72	Chris Vincent	
73	Chris Vincent	
74	Chris Vincent	
75	Chris Vincent	
76	Chris Vincent	
77	Chris Vincent	

113 http://en.wikipedia.org/wiki/User:Nv8200p
114 http://en.wikibooks.org/wiki/User:Patrik
115 http://en.wikipedia.org/wiki/User:Pacaro
116 http://en.wikipedia.org/wiki/
117 http://commons.wikimedia.org/wiki/User:Jiuguang_Wang
118 https://commons.wikimedia.org/wiki/User:Jiuguang_Wang

78	Chris Vincent	
79	Eirik Refsdal[119] from Trondheim, NORWAY	

119 `http://www.flickr.com/people/34034335@N00`

35. Licenses

35.1. GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed. Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow. TERMS AND CONDITIONS 0. Definitions.

“This License” refers to version 3 of the GNU General Public License.

“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as “you”. “Licensees” and “recipients” may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion. 1. Source Code.

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work. 2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary. 3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures. 4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee. 5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

* a) The work must carry prominent notices stating that you modified it, and giving a relevant date. * b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to “keep intact all notices”. * c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it. * d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate. 6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

* a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange. * b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge. * c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b. * d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements. * e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying. 7. Additional Terms.

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

* a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or * b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or * c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or * d) Limiting the use for publicity purposes of names of licensors or authors of the material; or * e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or * f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way. 8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10. 9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so. 10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it. 11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's “contributor version”.

A contributor's “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law. 12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program. 13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such. 14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version. 15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. 16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. 17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

<program> Copyright (C) <year> <name of author> This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'. This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

35.2. GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed. 0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference. 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License. 2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies. 3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document. 4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

* A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission. * B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement. * C. State on the Title page the name of the publisher of the Modified Version, as the publisher. * D. Preserve all the copyright notices of the Document. * E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices. * F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below. * G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice. * H. Include an unaltered copy of this License. * I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence. * J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission. * K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein. * L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles. * M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version. * N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section. * O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version. 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements". 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document. 7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate. 8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title. 9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it. 10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document. 11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing. ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (C) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with … Texts." line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

35.3. GNU Lesser General Public License