2010 Autonomous Robot Design Competition Space Cats

Course Notes

Last Updated: January 5, 2010

Massachusetts Institute of Technology

`http://spacecats.mit.edu`

2010 Sponsors

We thank this year's sponsors:

- **MIT Department of Electrical Engineering and Computer Science**
  Financial funding, laboratory facilities, staff help and support.
- **MIT Copy Technology Centers**
  Copying.
- **3M**
  Technical support.
- **Akamai**
  Financial funding.
- **Analog Devices**
  Donation of MEMS gyros and the time, help, and knowledge of Jack Memishian, Mark Nelson, and Howard Samuels.
- **Broadway Technology**
  Financial funding.
- **DE Shaw**
  Financial funding.
- **GM**
  Financial funding.
- **Google**
  Financial funding.
- **Guidant Foundation**
  Financial funding.
- **Intempco**
  Gyro Interface Board.

- **Jump Trading**
  Prizes for our top competitors.
- **LEGO Dacta A/S, PITSCO**
  PITSCO LEGO packs.
- **LEGO Denmark**
  Bulk LEGO.
- **LEGO Shop-at-Home**
  LEGO boxed sets.
- **Lockheed Martin**
  Financial funding.
- **Microsoft**
  Financial funding
- **Oracle**
  Financial funding
- **QRST's**
  T-shirts
- **Schlumberger**
  Financial funding
- **Sharp Electronics**
  Distance sensors

And we extend our special thanks to:

- **Ron Roscoe and the 38-600 lab staff**
  For infinite patience and invaluable resources.
- **Anne Hunter and the Course 6 Undergraduate Office**
  A logistical powerhouse without whom we would truly be lost.

Contents

List of Figures

Chapter 1

Introduction to 6.270

6.270 is a hands-on, learn-by-doing course in which participants design and build a robot that will play in a competition at the end of IAP. Each team begins with a box of components from which members must produce a robot that can manipulate game objects on a playing field inhabited by an opponent. Unlike the machines in 2.007 Introduction to Design (formerly 2.70), robots in 6.270 are completely autonomous. Human intervention during the round is forbidden.

The goal of 6.270 is to teach students about robotic design by giving them the hardware, software, and information they need to design, build, and program their own robot. The concepts and applications taught in this class are related to various other MIT classes (e.g. 6.01, 6.002, 6.004 and 2.007). However there are no formal prerequisites for 6.270. Students with little or no experience will find that they will learn everything they need to know from working with each other, attending lectures and workshops, and hacking on their robots.

6.270 is an extremely challenging course and requires participants to be willing to put in a real effort. Most students will spend in excess of one hundred hours building their robots. If you are willing to commit the time and energy needed for this class, you will have a great time and even learn something along the way.

So, prepare yourself for three and a half weeks of immersion into the world of robotics. Welcome to 6.270!

1.1 Staff

The 6.270 staff is composed of volunteers chosen from course alumni. You should feel free to approach them for help or with any questions you might have. The staff consists of two groups of people: Organizers and Teaching Assistants.

The Organizers are the people responsible for running the course. In addition to teaching and staffing the lab, they handle all the administrative duties, such as speaking with sponsors, ordering parts, defining the contest, and ensuring everything runs smoothly. A course the size of 6.270 requires a large amount of work and planning, and the Organizers have spent over a year preparing for this competition.

The Teaching Assistants (TAs) are selected by the Organizers to assist in teaching the course. They work primarily during IAP and their job description requires that they help teach recitations, staff the lab, and build demonstration robots and placebos. They are often also called upon by the Organizers to assist in certain tasks.

Members of the staff and their emails are listed in Figure 1.1.

Organizers and TAs receive very little compensation for the work they do. They are here only because they love 6.270 and want others to have the same opportunity to enjoy it as they did. In return, the staff asks only that you put in the time and effort necessary to get as much as possible out of the course. If you enjoy your experience in this course and would like to see it continue to be offered, please consider joining the staff in future years. It is only through the continued enthusiasm and selflessness of course alumni that 6.270 is able to remain the most popular student-run activity at MIT.

Organizer	**Email**
Adeoye Esho	`aaesho`
Andrew Marecki	`amarecki`
Jessi Ambrose	`jambrose`
Steven Herbst	`herbst`
Peter Iannucci	`iannucci`
James Fox	`jcfox`
Mark Sullivan	`marks3`
TungShen Chew	`tshen2`
Kevin Yapsir	`kpyapsir`
Rachel Meyer	`remeyer`
Mike McCanna	`acrefoot`
Jesse Moeller	`jmoeller`

Figure 1.1: 2010 6.270 Staff and email list

As we compile it, we will be placing more information, including pictures, lab hours, and skill lists (who to contact for help with a certain aspect of the contest) on the web:

`http://spacecats.mit.edu/contestants/`

1.2 Kits and Tools

The 6.270 kit, valued at about $1500, is yours to keep at the end of the contest. This is made possible by financial support from the EECS department and the course's commercial sponsors. If your team does not present a robot to the Organizers at the qualifying round of the competition, or if you are asked to leave the course, you will be required to forfeit the kit back to the EECS department. Teams who do not return the entire kit when asked will be charged the full $1500 through the Bursar's office.

There are tools available for in-lab electronics work, but these resources will probably be over-burdened, especially towards the end of IAP. Therefore, in addition to the kit, a set of tools will be reserved for purchase by your team.

We offer the tools at a cost that is a lower bulk price for us. However, if you're looking to find tools on your own, you'll want to get the following:

- Soldering iron
- Soldering stand
- Wire cutters
- Needle-nose pliers
- Helping hands
- Diagonal cutters

1.3 Electronic Communication

1.3.1 Mailing Lists

Participants are encouraged to check their email daily, since notices are sent out often and without warning.

- **6.270-staff@mit.edu** is the main adminstrative list for 6.270. All questions or comments concerning the course should be directed to this list. Current staff as well as all past Organizers are members of this list and will be able to help answer your questions.

- **6.270-rules@mit.edu** is for question about the rules, contest proceedings, or the validity of your robot. The Rules Committee is comprised of a subset of the Organizers.
- **6.270-contestants@mit.edu** is the primary announcement list for the course. You will be added automatically and all important information for participants will be sent to this list.
- **6.270-fanclub@mit.edu** contains all previous participants in 6.270, as well as some other members of the MIT community. This is an extremely low-traffic list that you will be added to as a 6.270 alumnus. It is used only for general interest 6.270 announcements, and under no circumstances should anyone but the Organizers send email to it.

1.4 Laboratory Facilities

During the course of constructing your robot, you will have access to workspaces, tools, and computers in the following areas:

1.4.1 6th Floor Laboratory

The 6th floor lab (Room 38-600) is the center of activity for the course. This lab is supervised by the 6.270 staff, and other teams will be present to share ideas with. Among the useful facilities in this lab are workbenches for building your robot, computers for programming, and two contest tables for testing.

The lab will be open and staffed from 11 am to 11:30 pm on weekdays and noon to 10 pm on weekends. During the final few days of the course, the lab may be open 24 hours a day. If you need to call the lab, the phone number is x3-7350, but please do not place or receive personal calls too often. The phone line needs to be kept available for official use, and the staff is too busy to run a personal messaging service.

Since this lab is on loan to 6.270 by the EECS department, please be courteous. Do not touch equipment not explicitly designated for 6.270 use and treat all lab staff with respect. Be aware that the lab will be closed at night for the first three weeks of the competition. So be ready to leave the lab when it's time to close. We reserve the right to penalize you for offenses related to the abuses of the lab or its staff

1.4.2 Other Facilities

Since the course software is available for a number of computer platforms, some students choose also to program their robots from their own computers. Teams with access to laptops may find this option especially useful even when working in the lab, since it frees them from waiting for the lab computers.

1.4.3 Etiquette

When working in the lab or at Athena, you will be expected to be respectful to those around you. We expect you to obey the following rules at all times:

1. *Noise.* Your robot will be quite noisy. When working at Athena, please minimize the operation of your robot. If others are disturbed by the noise, stop running the robot or move to another cluster.

2. *Hardware.* Do not solder, cut, or glue any hardware in the clusters or around the computers in lab. Debris can get lodged in the keyboards and damage the computer. Furthermore, when working on the lab benches with solder or glue make sure you are working on top of a piece of cardboard area to prevent damage to the tables. Failure to do so may result in the loss of lab privileges.

3. *Tidiness.* Do not take up more space than you need. Be tidy. The lab will be crowded and people will need places to work. Your team should try to limit the area that it uses to two workbenches, or one if the lab is very crowded.

4. *Multiple Machines.* Do not log on at multiple machines. When the lab is crowded, please try also to minimize the number of people on your team who are logged on. The lab does not have enough computers to support everyone being logged on at once.

Violations of the rules of etiquette will not be met with severe penalty. Causing a disturbance in the Athena clusters will result in your team's forced withdrawal from the competition, and your kit will be confiscated. Repeated violations of lab etiquette will be investigated by the 6.270 staff, and will be dealt with on a case by case basis. Penalizing teams is unpleasant for both the staff and competitors, so please respect one another and follow the etiquette.

1.5 Credit Guidelines

6.270 is offered as MIT subject 6.185 for 6 units of Pass/No Record credit with the further option to receive 6 Engineering Design Points (EDPs). Taking the course for credit is optional, but we encourage your full input. Receiving credit will give you formal recognition on your transcript in addition to the academic credit.

It is the job of the instructors to ensure that credit is properly awarded to students deserving of it. In order to properly evaluate your performance, it is necessary that you report your work. The credit requirements are structured to allow your instructor to authorize credit and also assist you in the learning process.

The following guidelines must be completed in order to receive 6 units of academic credit, and if desired, 6 EDPs:

- **Robot Web Page.** We will describe what is required on the web page at the end of the course. You don't need to worry about this right now.
- **Assignment Completion.** Several assignments will be handed out during the first weeks of the course. These assignments were made to help guide participants in making effective and competitive robots. Participants wanting credit are expected to complete the assignments on-time. Failure to complete the assignments on time will result in gradual penalties ultimatly resulting in forfeiture of the competition.
- **Completed Robot.** Every team is expected to "show" a robot at the qualifying round. Its functionality, or lack thereof, has no effect on the team's members receiving credit for the work they have done.

If you need extra time on the assignments, please talk to an organizer—we are all students and are very understanding of the contest's demands. However under no circumstances will extensions be granted for impounding. For your own benefit (and ours), please avoid the need for extensions and start assignments early.

1.6 Schedule

- **General Lectures.** Lectures will be held during the first week of the course to introduce you to the basics of robotics. These lectures are meant to provide you with an overview of the information necessary to create a working 6.270 robot.
- **Laboratory Sessions.** Staff members will be present in the 6th floor lab to assist you in the construction of your robot. One of the goals of 6.270 is to encourage interaction, and the lab is a great place to share ideas with others and experiment with new ideas.
- **Workshops.** During the first two weeks of the course, workshops will be taught by experienced staff members. These workshops will cover many aspects of the course, from soldering to programming to construction. They are planned to help reinforce the material learned in lecture.

1.7 Web Resources:

- **http://spacecats.mit.edu/contestants** is the contest homepage. Check here frequently for updated schedules, materials, and announcements.
- **http://spacecats.mit.edu/wiki** is our underground homepage. We may post sample code, detailed tutorials, and other resources here. We encourage contestants to use and contribute to the wiki.
- **http://code.google.com/p/joyos** hosts the source code for the HappyBoard's operating system, in case you're curious. Talk to one of the organizers if you're interested in contributing or have any suggestions about JoyOS.

Chapter 2

Space Cats

THE YEAR IS 3010. *Long after the disappearance of the human race, relics of an advanced civilization haunt a now quiet solar system. Space colonies as far out as the dwarf planet Pluto show a species that harnessed fusion power, connected billions of people through near-light speed travel, and sent probes as far from their own home as Alpha Centauri. But our star is now a quiet star.*

No one knows what happened to some fifty billion people, or where they are now, but one thing remains. One legacy to remind us that these were an inventive people, creative, engineering, and indeed humorous. Robots.

These absurd beings, at one time built to perform mundane tasks, now enjoy the home of their creators - and, lacking any meaningful labor, frolic in the cosmic void now and until stellar death.

The vermin eradicator unit, commonly referred to as a Space Cat, is the dominant robot of this new age. Originally prototyped by undergraduate engineers on Earth to keep households free of pests, the Space Cat has evolved to be the fearsome mecha that we recognize today. A Space Cat comes equipped with a powerful mind-control beam, capable of herding mice into certain doom.

The Space Mouse is yet another creation of sleep-deprived terrestrial undergraduates. Due to the overwhelming success of the Space Cat project in exterminating real vermin, futurists feared a robot rebellion was imminent unless the Space Cats were sufficiently challenged to the end of their life cycle. Thus the Space Mice were engineered as placebo vermin to keep the Space Cats busy and away from thoughts of rebellion.

Space Cat society is forged as a tribal hierarchy with an Alpha Cat leading the pack. Alpha Cat status is not awarded to the bot with the oldest firmware, the newest software updates, or the largest solenoids. It is earned by the cat with the strongest ability to fulfill its primary objective, as programmed by the manufacturer - to eradicate vermin!

FIND THEM AND DESTROY THEM!

2.1 Rules of Space Cats

The goal for 6.270: 2010 Space Cats is very simple. Catch more mice than your opponents!

2.1.1 Formalities

- Robots must begin the game occupying no more space than a 1' cube. Extensions and tethers may develop beyond this 1' cube after the match begins.
- All robots must meet impounding specifications prior to every match or face DQ. No modifications in between rounds are permitted aside from repair & restoration to quality at impounding. Code may never be modified after impounding.
- Penalties are issued by the judge at the table at his/her own discretion, and cannot be contested. Unintentional and non-damaging collisions may be overlooked. Robots that do their best to avoid collisions will receive favorable ruling. Robots are expected to be defensively built and respectful of other robots on the board such that this isn't a major issue.
- Each team has sixty seconds prior to round start to calibrate and set up their robot. This setup may in no way give hint to starting orientation or choose an arbitrary strategy (especially to counter a particular opponent). Robots must operate identically and autonomously every round. Teams that violate this rule are DQ.
- Each team has five seconds from the round start to call a false start. Each team may claim one false start per round. A team that is not ready (double false start) forfeits the match and enters the losers-bracket / leaves the tournament.

2.1.2 Game Board Features

- 8’x8’
- Doorsteps: Two cats start in 16” squares in separate corners, both facing a random (but symmetric) direction, and the long edge (8’) distant from the Mouse Holes.
- Mouse Holes: Three entry zones for two concurrent mice separated by 1.5’ at the opposite end of the board as the cats.
- The Black Hole: An 18” diameter disc region at the board center, elevated 6”, allows the mice to hide or evade capture by the cats. The disc is supported by a 2” diameter cylinder at the center.

2.1.3 RF

- The score of both cats will be broadcast, in conjunction with the current game time.
- The xy position of cats and mice will be broadcast, in conjunction with the current game time.

2.1.4 Terminology

- Capture: Having a mouse under your control by magnetic proximity.
- Deliver: Bringing a mouse to your doorstep.
- Score: Both capture and delivery of a mouse, earning 1 point.
- Mouse Hole / Spawn: Where new mice enter the board.
- Black Hole / Table / Cover / Hill: The central ”safe zone” for mice to move under.
- Penalty: An action that earns the offending cat -1 points.
- Double Loss (DL): Both cats have a zero score and neither received a penalty. A winners-bracket cat enters the loser’s bracket; a losers-bracket cat is eliminated from the tournament.
- Sudden Death (SD): Only one mouse is running. The first cat to capture and/or score the mouse wins. If time runs out, the cat that is currently in possession wins; failing that, the cat that first captured the mouse wins. If neither applies both cats lose.

2.1.5 The Cats

- **Cats may not harm mice!** Cats that collide with, push, ensnare, or in any other way influence mice movement without the use of magnetism will be penalized.

- Cats may not intentionally collide with other cats. Tethers are allowed, but should not ensnare or damage the autonomy of other robots. Blocking access to the doorstep is allowed.

- Cats may not reach into the Black Hole, since they may inadvertently damage the mice.

- Cats may not reach into the Mouse Hole, or block the exit of a mouse onto the board, but are allowed to camp the spawn at a radial distance of 8".

2.1.6 The Mice

- The mouse is a staff-designed small robot that will move in the negative direction (anti-normal) of a magnetic field at close range. This means a mouse robot will follow a cat that dangles a magnet above the mouse.

- If you reverse the magnet polarity the mouse will run away until it is out of range.

- A mouse out of range of a magnet will wander randomly, trace a clever pattern, or tease the closest cat. The AI will be chosen arbitrarily by the table judge.

- Scored mice are respawned from arbitrary holes and will not broadcast position until they are on the board and live. The initial spawn is at symmetric holes for fairness.

- Mice spawn with 3 seconds grace period from being captured by magnetic attraction.

- No more than two mice will be active on the board at any time.

- A mouse will not camp in the Black Hole longer than 3 seconds, nor reenter the Black Hole for 5 seconds upon leaving.

- A mouse will not reenter a Mouse Hole.

- A mouse on a doorstep is not scored unless a cat has it captured. Mice may freely wander on the doorstep without being scored.

2.1.7 Draws

- If a draw occurs and one cat was penalized, the non-penalized cat wins. A zero-score draw with a penalty favors the non-penalizing cat as the victor, and is not a double loss.
- In the event that both cats have an equivalent number of points at the end of the match, the cat that has a mouse presently captured wins. If both mice have a mouse captured, or neither (and the score is greater than zero), a sudden death round determines the winner. If the score is zero, both cats lose.
- If a draw occurs outside of SD and both cats were penalized, a sudden death round is run.
- A cat that receives a penalty in SD immediately loses.

2.2 The Competition

The competition is a double elimination tournament. Each robot competes head to head in successive rounds until it loses twice. When all but one robot has been eliminated, that robot will be crowned champion.

All rules are subject to change at any point during competition, at the discretion of the Organizers. This power will be used sparingly.

- **Qualification** Every robot is require to qualify for the competiton, and to receive credit for the class. Qualification is simply demonstrating to an organizer that your robot is capable of capturing and scoring at least one mouse in game.
- **Impounding - Jan 27th 5pm** All work on robots must be complete by the impounding date. No further modifications in design or code may be made after impounding. All teams are asked to submit a copy of their robot code to be reviewed by the organizers.
- **Seeding - Jan 28th 12pm** Seed positions in the bracket will be determined by preliminary rounds. Robots that score more mice will earn a more favorable seed.
- **Final Tournament - Jan 28th 7pm** This is the main event that everyone comes to see. Robots will compete until all but the winner have been eliminated. Once two teams remain, a best-of-3 match will determine the winner.

We encourage contestants to explore novel strategies and have a lot of fun in 6.270. The organizers will award prizes to robots that excel in certain categories, regardless of their performance in the tournament.

2.3 General Rules

The following rules of play are meant to ensure a fair and interesting contest. Contestants are responsible for knowing and following these rules. If you have any questions or doubts about the legality of your robot, please ask the Rules Committee by emailing `6.270-rules@mit.edu` for an official ruling.

2.3.1 Period of Play

1. The contestants will have a 60 second calibration period to setup their robot. A team can use this time to tune sensors to the table and enter the goal points for the match. A team **cannot** use this time to enter any of the following information:

   (a) Starting orientation
   (b) Starting color/side
   (c) Specific strategy
   (d) Opposing team

   Your robot must dynamically sense and make decisions during the round.

2. By the end of the calibration period, teams must place their robots in the assigned starting zone and orient in the assigned direction. The direction will be one of four options: North, South, East, and West. The side marked "front" must point in this direction.

3. Robots may not supply power to their actuators at this point. If a robot does, it has false-started. Also, if a team takes more than the alotted 60 seconds for setup, this counts as a false start. If a robot false-starts twice, it forfeits that match.

4. The robots will recieve a radio signal informing them to start the match. After the start signal,the robots may turn on any motors or actuators and the robots have 120 seconds to compete and score points. Software to detect the signal will be provided.

5. During the match, the contestants must stand back from the table. Any contestant who touches the machines or otherwise interferes with the match will cause their robot to forfeit the match. All robots must be controlled solely by their onboard computer.

6. At the end of 120 seconds, the robot must turn off electrical power to its actuators. Any robot that fails to shutdown forfeits that match. Software is provided to do this.

7. The match ends when all robots and game objects on the table come to a rest.
8. If a robot forfeits a match for any reason, it's intended opponent will played a placebo robot.

2.3.2 Kits

1. Some parts in the kit are considered tools and may not be used on the robot.
2. Robots must be built only from the parts in the kit, except when explicitly allowed by other rules.
3. Teams may trade only functionally identical components. This includes trading identical LEGO parts of different colors and replacing broken components. Any other trading is forbidden.

2.3.3 Robots

1. The robot structure must fit within a one foot cube at the start of a match; however, they may expand once the match has begun. Wires may be compressed, if necessary, to fit this measurement.
2. All parts of a robot must be connected via LEGO. Robots may not separate or have a tendency to break into multiple parts.
3. Decorations may be added to a robot provided they perform only an aesthetic function, and not a structural one.
4. Robots may not intentionally damage, or attempt to damage, the opponent robot or its microprocessor board.
5. No parts or substances may be deliberately dropped onto the playing field.
6. Any robot that appears to be a safety hazard will be disqualified from the competition.

2.3.4 LEGO

1. Only LEGO parts may be used as robot structure.
2. A robot's structure may not be altered after impounding. Repairs may be made between rounds if time permits.

3. LEGO pieces may not be modified in any way, with the following exceptions:
   - The LEGO baseplate may be modified freely.
   - LEGO pieces may be modified to facilitate the mounting of sensors and actuators. However, such modification cannot be structural.
   - LEGO pieces may be modified to perform functions directly related to the operation of a sensor. For example, holes may be drilled in a LEGO wheel to help make an optical shaft encoder. Such modifications cannot be structural.
4. LEGO pieces may not be joined by adhesive.
5. Lubricants of any kind are not permitted.
6. Rubber band or tape may be applied to LEGO wheels and treads to alter the coefficient of friction. See Section 2.3.6 for more details on the use of rubber bands.
7. Wheels may be stuffed with any material within reason. Students usually stuff them with rubber bands, LEGO, or hot glue. Double-check with an Organizer before using another material.

2.3.5 Software

1. A robot's program cannot be altered after impounding.
2. In the event of a memory failure, a copy of the robot's program may be downloaded from an official computer between rounds. The program available for download will be the version submitted on impounding. Contestants are not permitted to download any code to the robot from their own computer or any other computer.
3. A robot may not be told its position or be given information about its opponent. It may only deduce this information after the match has begun.

2.3.6 Non-LEGO parts

1. Sensors, actuators, and other Non-LEGO parts may not be used as structural components.
2. Non-LEGO parts may be attached to no more than five LEGO parts.
3. Non-LEGO parts may be freely modified to assist in their operation.
4. A reasonable amount of cardboard, other paper products, and tape may be used for the purpose of creating optical shields for sensors.

5. Wire may only be used for electrical purposes and may not be dragged on the playing surface. Wires that extend outside of the robot should be tied back.

6. String may be used to convey force between moving parts (i.e. pulley systems) but may not be used for structural support. String may also be used to stuff tires for stiffening purposes.

7. Extraneous components may not be added to a robot for the purpose of adding weight.

8. Rubber bands cannot be used for structural support. Only thin rubber bands that are supplied by the organizers (#16 and #32) may be used. Rubber bands may be used only for the following purposes:

   - Rubber bands may be used to store energy to affect the motion of moving parts. Rubber bands used for this purpose must be touching at least one piece of LEGO. LEGO pieces connected by a single rubber band or a chain of two rubber bands must move relative to each other.
   - Rubber bands may be used to stuff tires for stiffening purposes.
   - Rubber bands may be used to add strength to tread between chain.
   - If you would like to use rubber bands for another purpose, please make sure you check with Rules Committee first.

2.3.7 Placebos

Placebos are staff-built "demo" robots, which should not present significant competition to a well-built entry. In matches involving only one robot player, a placebo will stand in for the other. Teams should consider a match against a placebo to be just like a match against any other robot. The placebo will conform to all rules.

2.4 Extra Electronics

Each team is given two pools of resources from which they can obtain more sensors and motors to their robots—20 sensor points and a 30 dollar allotment. A team's robot is disqualified if the total points of the sensors on the robot after impounding exceeds the sum of 20 points' worth of sensors and the basic set of sensors originally given in the kit, or if the electronics monetary value exceeds 30 dollars.

2.4.1 Electronic Modifications

Each team is free to modify any of the electronics or actuators. However, using electronics in a non-standard fashion is a risk that your team must consider before making any modifications. If electronics (including the Handy Board and expansion board) or actuators are broken because of these non-standard modifications, the team will not be given replacement parts. Any additional parts used for modifications count towards the 20 sensor points and 30 dollar limit.

Before making any modifications, each team *must* consult an Organizer. Before making the modification, the team must turn in a list of all the parts (kit or non-kit) used for the modification. The team must also turn in a design report that includes a description of the modification, a schematic of all added circuitry. This design report must be turned in *before* the modification is made on the robot.

All modifications made by all teams will be posted online once the design report is given the Organizers. Any questions or concerns about a potential modification must be emailed to

```
6.270-rules@mit.edu
```

.

2.4.2 The Sensor Store

In order to encourage variety in robot designs, each team will be given a sample set of sensors and an allowance of 20 sensor points with which they may obtain additional sensors from the Organizers. No refunds will be permitted, so contestants are encouraged to experiment with the sample sensors before making decisions on which sensors to get. Note that sensors purchased from the sensor store are considered kit parts and must be used in accordance with all applicable rules.

Sensors can be traded as long as the following rules are observed:

1. Teams are allowed to trade sensors of equal point value with other teams.

2. A team can trade a sensor with the Sensor Store as long as the sensor is returned in original condition.

3. A broken sensor may be traded in for a new sensor of the same type at a cost of 5 dollars or the at-cost price of the sensor rounded up to the nearest dollar, whichever is highest. Paying for a broken sensor this way does not count to the 30 dollar nor the 20 sensor point allotment.

2.4.3 30 Dollar Electronics Rule

A team may spend up to 30 dollars of its own funds to purchase electronic components from non-6.270 sources. This is not to be confused with the sensor points, which can only be used for the Sensor Store. Contestants are encouraged to use this rule to explore new ways of sensing or otherwise make their robot more interesting. Teams taking advantage of this provision, however, must abide by the following guidelines:

1. Each team is required to submit receipts for every additional component purchased. All receipts must be submitted at impounding.
2. If a team wishes to use parts obtained through means other than retail purchase, an equivalent cost will be assigned by the Organizers. This estimate must be obtained in writing from the Organizers.
3. Resistors rated less than 1 watt and capacitors valued less than 100 μ F may be used freely, without counting towards the 30 dollar total.
4. Extra servos can be purchased from 6.270 staff at varying prices based on quality.
5. If a team needs a replacement servo, the team must pay for the servo at retail price rounded up to the nearest dollar. The replacement fee does not count towards the 30 dollar allotment.
6. Only components on the actual impounded robot will count towards the 30 dollar allotment.

Chapter 3

Electronic Assembly

This chapter presents an introduction to electronic assembly followed by step-by-step instructions for assembling the hardware used in 6.270. The instructions assume no prior background in electronics and should provide enough information to get you started. It is recommended that you assemble the components in the order presented by this chapter. The sections are arranged to give you a gentle introduction before you go on to tackle the tougher tasks.

> If ever there was a place in life where neatness counts, it is in electronic assembly. A neatly built and carefully soldered circuit will perform well for years. A sloppily and hastily assembled circuit, however, will cause ongoing problems and failures at inopportune times. It is well worth the extra effort to make sure you get it right the first time.

3.0.4 Mounting Components

When mounting components on a circuit board, the general rule is to try to mount them as close to the board as possible. The primary exceptions to this rule are components that must be bent or folded over before being soldered. Resistors and diodes often fall into this category.

Components come in two standard packing types: radial and axial. The leads on radial components all point in the same direction and generally fit into the holes in the circuit board. The leads of axial components must be bent or modified for mounting. If space has been provided to mount the component flat, then do so. Otherwise, just bend one lead over parallel to the component and mount it vertically. Figure 3.1 shows how to mount an axial component.

After soldering the component into place, use a pair of cutters to clip off the extra length of each lead. When clipping the leads, face the board and the lead down into a

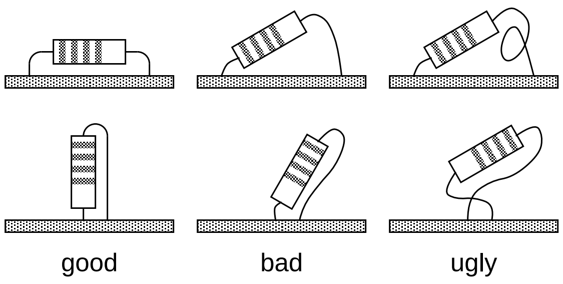


Figure 3.1: Axial component mounting

garbage can or into your hand. Leads tend to shoot off at high speeds and can fly into someone's eye.

3.0.5 Desoldering

Desoldering a component takes about ten times as long as it does to mount it in the first place, so you want to be very careful during the assembly process. Regardless of how meticulous you are, though, mistakes are inevitable and components can burn-out, so it is important to know how to fix them. Fortunately, two tools, desoldering pumps and desoldering wick, are available to help.

Desoldering pumps work by sucking up molten solder. You begin by depressing the plunger until it latches. Hold the desoldering pump in one hand and the soldering iron in the other. Use the soldering iron to melt the solder and then quickly remove the iron as you bring in the pump. Immediately trigger the pump to suck up the solder before it resolidifies. The next time the plunger is depressed, the collected solder will be ejected from the pump. The tip of the desoldering pump is made of Teflon so that solder cannot stick to it. While Teflon is heat-resistant, it is not invincible, so be careful not to touch the Teflon tip directly to the soldering iron.

Another option for removing solder is to use desoldering wick. The wick is composed of a number of small braided wires and is used in conjunction with the soldering iron to pick up solder. You simply melt the solder with the iron and touch the wick to it. Solder has a strong attraction to the wick, so the molten solder will flow into the braid. This allows you to collect the solder, but once the solder wick is used, that part of it cannot be used again. Note that the solder wick should also be hot for the solder to flow well.

3.1 Components

Electronic circuits are constructed from a number of different types of building blocks. These components come in all different shapes and sizes and have a variety of functions. Building them into a working circuit requires being able to identify their packages and read their values.

Some components are also *polarized.* They must be mounted in the correct orientation otherwise they will not function correctly and might even explode. Being able to reliably read the markings which identify the polarity of a device can save hours of frustration.

3.1.1 Resistors

Resistors are usually small cylindrical devices with color-coded bands indicating their value. Most of the resistors that you will use are 1/8 watt, which is a very low power rating, thus they are rather tiny devices. Resistors with higher power ratings tend to be much larger. A 2 watt resistor is a large cylinder, while a 5 watt resistor has a large rectangular package. Regardless of size, all resistors are nonpolarized, so they may be installed in either direction without causing problems.

The largest resistors often have their value printed on them, but all other resistors are labelled using a standard color code. The code consists of four colored bands around the resistor package. The first two bands form the mantissa, and the third is the exponent. The resistance is read by taking the number formed by the mantissa values and multiplying it by ten raised to the power of the exponent. The fourth band represents the tolerance of the resistor. It can be either silver for 10% tolerance or gold for 5% tolerance. If the fourth band is missing, then the tolerance is 20%.

Figure 3.2 shows the meaning of the colors. A few examples should demonstrate how to read a resistor:

- *brown, black, red*: 1,000Ω or 1kΩ
- *yellow, violet, orange*: 47,000Ω or 47kΩ
- *red, red, yellow*: 220,000Ω or 220kΩ

3.1.2 Resistor Packs

Resistor packs are a collection of resistors in a flat, rectangular package. The two basic types of resistor packs are shown in Figure 3.3:

- **Isolated Element** resistor packs contain three to five discrete resistors. The pack is labelled with a "V" in front of the resistance value, such as "V47kΩ ." These devices are not polarized and can be installed in either direction.

color	mantissa value	multiplier value
black	0	1
brown	1	10
red	2	100
orange	3	1,000
yellow	4	10,000
green	5	100,000
blue	6	1,000,000
violet	7	
gray	8	
white	9	

Figure 3.2: Resistor color code

- **Common Terminal** resistor packs contain anywhere from three to nine resistors per package with each resistor connected to the common terminal. The pack is labelled with an "E" in front of the resistance value, such as "E47kΩ ." These devices are polarized and are marked with either a dot or bar at the end of the package with the common pin.

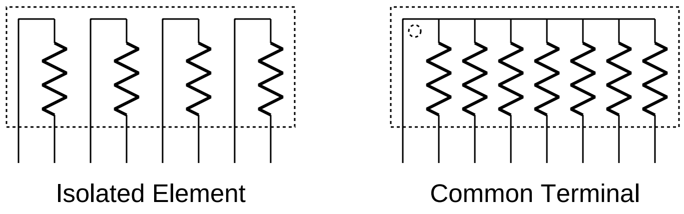


Figure 3.3: Resistor pack internal wiring

3.1.3 Capacitors

Capacitors are available in a variety of types and values:

- **Monolithic** capacitors are small components about the size and shape of the head of a match. They are excellent choices when small values ($1.0\mu F$ or less) are needed because they are compact and inexpensive. They are never polarized.

- **Electrolytic** capacitors look like minature tin cans with a plastic wrapper. They are available in large values ($1.0\mu F$ or greater), but become quite bulky as the value increases. They are fairly inexpensive, so they are a common choice for many applications. Except for a few special cases, electrolytics are usually polarized.

- **Tantalum** capacitors are compact, bulb-shaped components. They are excellent for larger values ($1.0\mu F$ or greater), since they are smaller and more reliable than electrolytic. Unfortunately, though, they are also much more expensive. They are always polarized.

Polarized capacitors have a tendency to explode when they are mounted backwards, so it is important to know how to read them correctly. Some of them are easy and have one or both of the leads marked with a plus (+) or minus (-). Others have the positive lead marked with either a dot or a vertical bar. This should not be confused with the stripe with several minus signs on it which marks the negative lead on some electrolytics.

Reading capacitor values can be even more confusing than determining their polarity. Capacitors often have numbers printed on the package which have nothing to do with the value, so the first task is to figure out which are the relevant numbers.

For large capacitors (1.0μF or greater), the value is often printed plainly on the packages, such as 4.7μF. In some cases, the μ symbol acts as a decimal point like 4μ 7 for a 4.7μF value. Small capacitors (1.0μF or less) have their values printed in picofarads ($1,000,000p$F $=$ 1μF). These values are coded in a manner similar to resistor values where there are two digits of mantissa followed by one digit of exponent. Hence, the value 473 represents $47,000p$F or 0.047μF.

3.1.4 Diodes and LEDs

Diodes and LEDs (Light Emitting Diodes) have two leads, called the *anode* and *cathode*. When the anode is connected to a positive voltage with respect to the cathode, current flows. If the polarity is reversed, no current will flow. Figure 3.4 shows how to identify the leads.

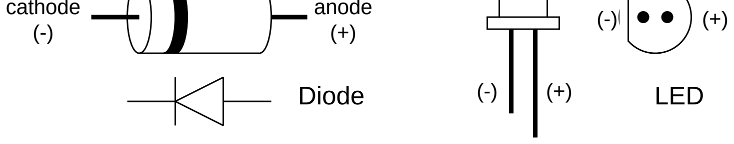


Figure 3.4: Identifying diode leads

Diodes usually come in small cylindrical packages similar to resistors. Most diodes have a marking, usually a band around the package, which identifies the cathode.

LEDs are special types of diodes that light up when current flows through them. The cathode is marked either by a small flat edge along the circumference of the casing or by the shorter of the two leads. The LED must be mounted in the correct direction or it will not work.

3.1.5 Integrated Circuits

Integrated Circuits (ICs) are packages containing complex circuits. They come in a variety of shapes and sizes, but the most commonly used variety for manually assembled circuit boards are DIPs (Dual-Inline Packages).

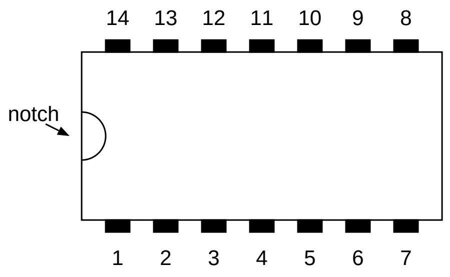


Figure 3.5: Top view of a 14-pin DIP

A marking on the component identifies pin 1 of the component's circuit, as shown in Figure 3.5. This may be a small dot, notch, or ridge in the package. After pin 1 is identified, pin numbering proceeds counterclockwise around the chip.

Instead of soldering the IC directly to the circuit board, a socket is often installed in its place. The IC is then mounted into the socket, so that it can be easily replaced if it fails. This also protects the delicate chip from the heat of soldering.

Sockets are not polarized, but they often carry a marking similar to the chips that they will be holding. Installing the socket with the notch in the proper orientation will make it easier to install the IC correctly.

3.2 Connectors

Sensors and actuators must be connected to the controller board using wires. Since it is desirable to be able to plug and unplug them, connectors are used which fit into the various ports. This provides a simple, modular design.

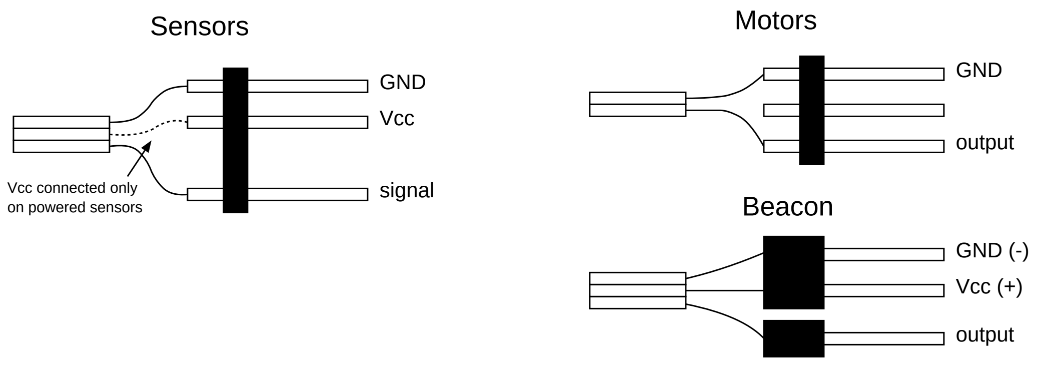


Figure 3.6: 6.270 connector standard

In order to keep connectors from being plugged into the wrong port, different types of components are built with different connectors. Figure 3.6 shows the configuration used for each device. When building connectors for polarized devices, it is important to attach the wires to the pins in the correct order.

1. Cut a length of ribbon cable with the appropriate number of wires for the device you are building. "Unzip" the ribbon cable by separating the individual wires.
2. Strip and tin both ends of the wires.
   (a) Remove just enough insulation from the ends of the wires to allow them to be soldered.
   (b) With your fingers, twist the threads of each individual wire end tightly.
   (c) Heat the wire with the tip of the soldering iron and touch the solder to the wire.
   (d) The solder should wick up the twisted wire, forming a single conductor.
3. Slip each wire through a $\frac{1}{4}''$ length of $\frac{1}{16}''$ heat shrink.
4. Cut a connector with the necessary number of pins from a piece of male header. Clip out any unneeded pins with a pair of cutters.
5. Solder the wires to the connector.
6. Slip the heat shrink over the soldered wire and connector. Use a heat gun to shrink the wrap tightly over the connection. A match or butane lighter may be used if a heat gun is unavailable, but beware of burning the insulation. Hold the joint so the heat shrink tubing is about $1''$ above the tip of the flame.

7. If heat shrink is unavailable, hot glue may instead be used to strengthen and insulate the connection. For the really careful, you may also apply the heat shrink tubing after using hot glue.

   (a) Apply the glue to fill the void between the wires.

   (b) While the glue is still hot, use a pair of pliers to flatten it. The pliers will also work as a heatsink to cool the glue faster.

   (c) After ten seconds, carefully peel the connector from the pliers being careful not to break it. Trim off any excess glue.

3.3 Motors

The DC motors used in 6.270 are great for building robots because they are compact and powerful. Unfortunately, though they are not designed to be used with LEGO parts. To make them compatible with your robot, you will have to *legoize* them.

1. Mount a LEGO gear on the motor shaft

   (a) Push an 8-tooth gear over the existing metal gear. Make sure that it goes all the way on and that it is aligned correctly.

2. Legoize the motor

   (a) Construct the jig shown in Figure 3.7. It requires 4 1×10 beams, 2 1×6 beams, 1 1×4 beam, 2 1×2 beams, 1 2×2 brick, 4 2×4 plates, 10 black connectors, 2 axle connectors, 1 40-tooth gear, and 1 24-tooth gear. If you are left-handed, you might find it more useful to construct the mirror image of the jig.

   (b) Open the jig by disconnecting the top beams at the left side and flipping the top to the right.

   (c) Attach a piece of double-sided foam tape to the top of a 2×4 plate. This will become the bottom of the legoized motor. Mount this into the bottom of the jig, at the spot marked, “mount 1.”

   (d) Attach the motor to the 2×4 LEGO plate prepared in the previous step. Make sure that the gear on the motor’s shaft lines up correctly with the 40-tooth gear on the jig.

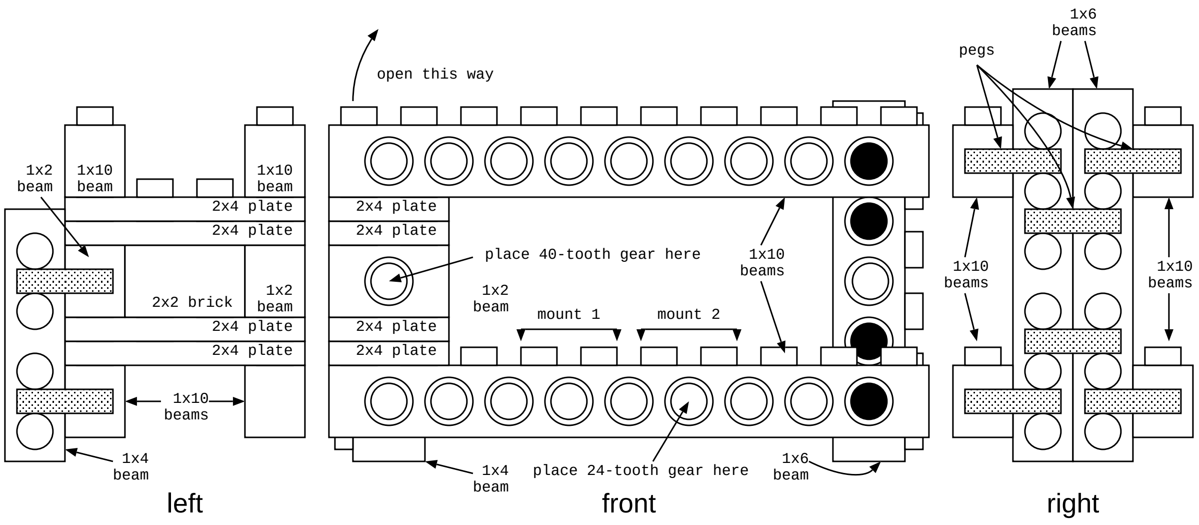


Figure 3.7: Jig for motor assembly

(e) Attach a piece of double-sided foam tape to the bottom of another 2×4 plate. This will become the top of the legoized motor. Mount this onto the top of the jig which you earlier flipped open, so that it will align with the top of the motor.

(f) Close the top of the jig to press the top onto the motor. Carefully remove the completed motor from the jig.

(g) You should then place the motor at “mount 2” to test that it meshes correctly with the 24-tooth gear. If it does not, you will need to figure out what went wrong.

Note: If you are building a non-standard 6.270 motor, you can use “mount 2” to determine the proper vertical spacing and revise the instructions above appropriately.

3. Wire a connector to the motor

(a) Cut a length of ribbon cable with two strands of wire.

(b) On the side of the motor are two metal leads or pads. Solder one wire to each.

(c) Solder the other end of the wire to a connector appropriate for a motor.

(d) Glue the wire to the case of the motor using hot glue. This will provide stress relief to protect the solder joints.

3.4 Servo

A servo is an electric motor with an internal gear train, position sensor, and driver circuitry which allows the motor to be positioned with reasonable precision based upon the input signal. It can be moved to any orientation in an approximately 180 degree range.

Fortunately, the servo already has the appropriate female header needed to connect it to the expansion board for the Handy Board. Because the servo is polarized, be *extremely* careful when attaching the servo to the expansion board. Make sure the black wire goes to ground (marked as “-” on the expansion board) and *not* to signal (marked “s”).

3.5 The Happyboard and Expansion Board

The “brain” of your robot is Ross Glashan’s Happyboard, a controller based on the Atmel ATMega128 processor. It provides:

1. Sixteen analog inputs and eight digital inputs.
2. Six bi-directional motor ports.
3. Six servo ports.
4. Four shaft encoder ports.
5. An LCD screen for debugging output.

Instructions for assembling the expansion board will be found in the assignment documents.

3.5.1

Chapter 4

Sensors

The concept of a sensor should already be familiar to you. You have an array of sensors which you use to feel, see, hear, taste, and smell. You rely on these senses for just about everything you do. Without them, you would be incapable of performing even the most simple tasks.

Robots are no different. Without sensors, they are merely machines, incapable of adapting to any change in the environment. Sensors give your robot the ability to collect information about the world around it and to choose an action appropriate to the situation.

After reading this chapter, you should take some time to play with your sensors. Wire at least one of each type and learn how it works, what values it returns, and under what conditions it will production those values. Every sensor has its own little quirks, and only through experimentation will you acquire the expertise necessary to integrate them into your robot.

4.1 Digital Sensors

Digital sensors work a lot like light switches. The switch can either be in the "on" position or the "off" position, but never in between. Even if you hold it in the center, the light will be either on or off. When a digital sensor is on, it returns a voltage which the controller interprets as a value of one. When it is off, the value is zero.

All digital sensors can be modelled as if they were switches. When plugged into a sensor port, digital sensors resemble one of the circuits shown in figure 4.1. When the switch is closed, electrical current flows freely through it, and the output is pulled down to GND. When the switch is open, the pullup resistor causes the signal line to float to Vcc. While Vcc usually represents a logic one, the value is inverted in software so that the value one represents the situation where the sensor is activated.

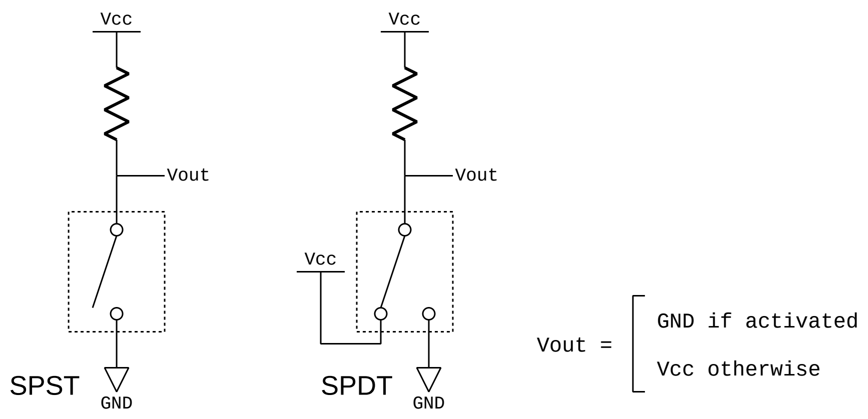


Figure 4.1: Digital Sensor Circuit

4.1.1 Switches and buttons

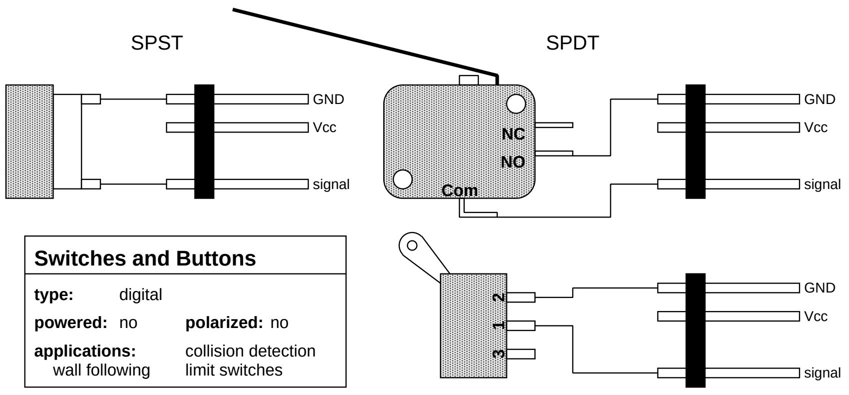


Figure 4.2: Switches and buttons

Switches and buttons are probably the easiest and most intuitive sensors to use. They are digital in nature and make great object detectors as long as you are only worried about whether the robot is touching something. Fortunately, this is usually enough for detecting when the robot has run into a wall or some other obstacle. They can also be used for limiting the motion of a mechanism by providing feedback about when to stop it.

Switches and buttons come in a wide variety of styles. Some have levers or rollers. Some look very much like computer keys. Some are computer keys. Whatever they look like, it should be obvious which of your sensors are switches.

Switches have two important properties which describe how they are wired inside:

number of poles and number of positions (throw). The number of poles tells how many connections get switched when the switch is activated. The throw represents how many different positions the switch can be placed into. The most common types are SPST (single pole, single throw) and SPDT (single pole, double throw). Most buttons fall into the SPST category.

An SPST switch has two terminals which are connected when the switch is activated and disconnected otherwise. An SPDT switch has three terminals: common (labelled "C"), normally open ("NO"), and normally closed ("NC"). When the switch is activated, common is connected to normally open, and when it is not, common is connected to normally closed. An SPDT switch can be used as an SPST switch by ignoring the normally closed terminal.

Switches and buttons should be wired as shown in Figure 4.2. SPST switches are not polarized, so it does not matter which terminal is connected to signal. SPDT switches, when not used as SPST switches, should have the common terminal connected to signal.

4.2 Resistive Analog Sensors

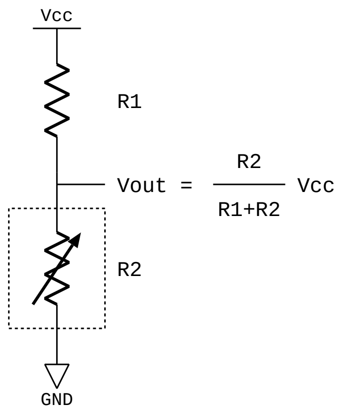
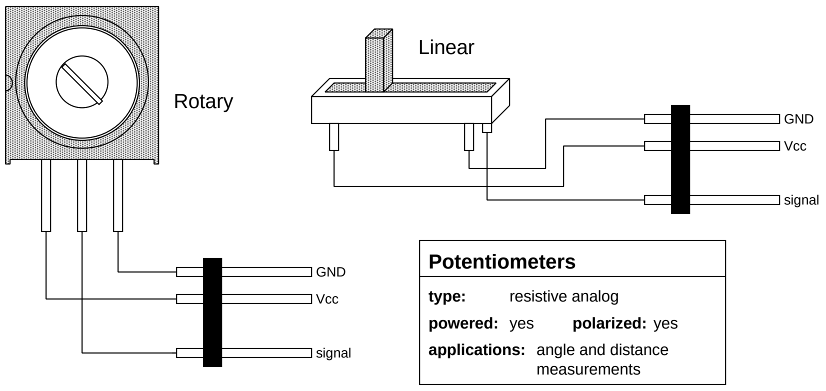


Figure 4.3: Resistive Analog Sensor Circuit

Resistive sensors change resistance with changes in the environment. When plugged into a sensor port, the sensor and pullup resistor form a voltage divider which determines the voltage at the signal input as shown in figure 4.3. When the resistance of the sensor is high, little current flows through the circuit, and the voltage across the pullup resistor is small, causing the signal voltage to approach Vcc. When the sensor's resitance is low, more current flows and the pullup resistor causes the signal voltage to drop.

4.2.1 Potentiometers



Potentiometers

type: resistive analog

powered: yes **polarized:** yes

applications: angle and distance measurements

Figure 4.4: Potentiometers

Potentiometers, often called “pots,” are variable resistors. They come in a variety of shapes and sizes, but can be grouped into two categories: rotary and linear.

Rotary pots have a knob which can be turned to vary the resistance. By wiring the two outside pins to power (Vcc) and ground (GND) and the center tap to signal as shown in Figure 4.4, the pot can be used to measure angles. They are very well-suited to measuring angles of joints in the robot.

Linear pots are very similar to rotary pots, except that they have a slider which changes the resistance. As the slider is moved back and forth, the output value changes, allowing motion in a straight line to be measured. Linear potentiometers should be wired as shown in Figure 4.4.

4.3 Transistive Analog Sensors

All transistors have three leads, the base, collector, and emitter. The voltage level present at the base determines how much current is allowed to flow from the collector to the emitter. This is easy to visualize in terms of a water faucet. As the knob (base) is turned, water is allowed to flow through the faucet.

Transistive sensors are analog and work just like regular transistors, except that the base is replaced with an element sensitive to some stimulus (usually light). When the sensor is exposed to this stimulus, the faucet opens, and current is allowed to flow from the collector to the emitter.

Figure 4.5 shows a circuit diagram of a phototransistor plugged into a sensor port. When the sensor is in the dark, no current flows through the circuit. This causes the reading on the sensor port to be pulled up to Vcc through the resistor. As the light level

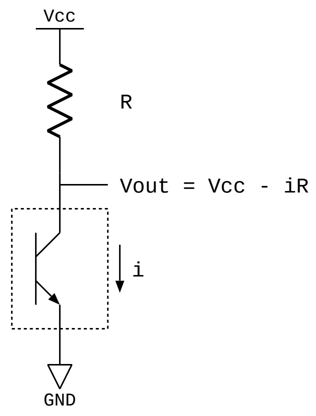


Figure 4.5: Transistive Analog Sensor Circuit

increases, however, current begins to flow from Vcc through the resistor to GND. The current causes a voltage drop across the resistor which decreases the voltage measured at the port. When the transistor is fully open, the measured voltage will hover around GND.

4.3.1 IR LED and Phototransistor

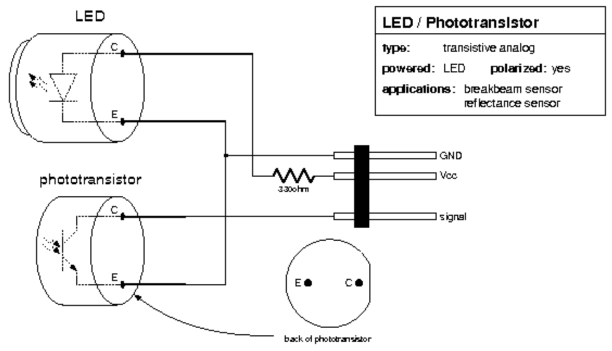


Figure 4.6: IR LED and Phototransistor

The IR phototransistor's output characteristics are perfect for use with the analog sensor ports, and in some cases, can be relied upon to produce digital signals. It responds very well to the accompanying LED, but barely responds to visible light. Since the components are tuned to work with each other, best results will be achieved when they are used together, as shown here.

The IR LED (part #LTE-4208) has a clear package, and its two leads have different lengths. The shorter lead on the flat side of the LED package is the cathode, and should be connected with the 330Ω resistor to ground.

The IR phototransistor (part #LTR-3208E) also has two leads, but its package is dark. The flat side of the phototransistor corresponds to the collector, and should be connected to the Signal header pin. (The longer lead on the rounded side is the emitter, and should be connected to ground, as shown above).

Because the phototransistor is so sensitive, it might be helpful to wrap some black heat shrink or electrical tape around it if you're having difficulty getting distinct readings for light and dark. This also useful when looking for light in a particular direction.

Once again, realize that the LED does not have to be perpetually on during the 90 seconds the robot competes. The LED can be plugged into a motor output to allow differential light measurements. And remember any number of LEDs can be plugged into a single motor port.

These 5mm LEDs and phototransistors are particularly convenient when working with LEGO because they are just the right size to fit into the LEGO axle holes. It is very easy to mount this sensor into your robot when constructing breakbeam sensors and shaft encoders.

4.3.2 Breakbeam Sensor Package

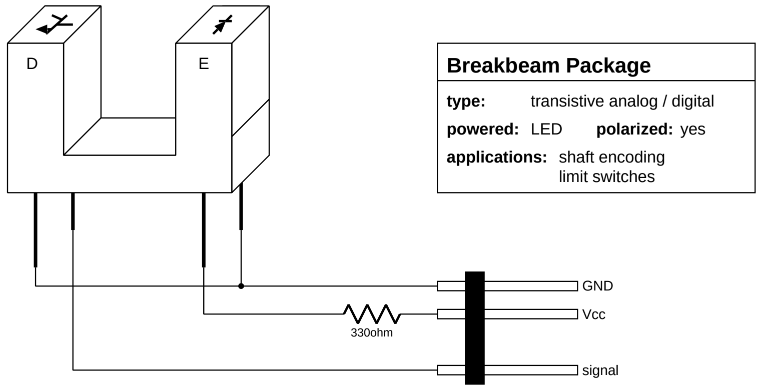

D
E
Breakbeam Package
type: transistive analog / digital
powered: LED polarized: yes
applications: shaft encoding
limit switches
GND
Vcc
330ohm
signal

Figure 4.7: Breakbeam sensor package

The breakbeam sensor package is composed of an infrared LED and a phototransistor which is sensitive to the wavelength of light emitted by the LED. The two components are mounted in the package so that they face each other with a gap in between them.

The sensor should be wired as shown in Figure 4.7. As usual with LEDs, a 330Ω resistor will be needed to limit the amount of current used to light it. Be sure to orient the sensor correctly, so that you do not confuse the phototransistor half with the LED half. The markings vary from one package to the next, but usually include one or more of the following:

1. an "E" (emitter) for the LED and a "D" (detector) for the phototransistor marked above each component.
2. arrows on the top of the package which point towards the phototransistor side.
3. a notch on the LED side of the package.

The sensor is valuable for detecting the presense of opaque objects. Normally, light from the LED shines on the phototransistor, but when a object blocks the path, the phototransistor only sees darkness. This can be useful in constructing mechanisms which must be stopped after moving a certain distance. Also, shaft encoders can be built by using the sensor to count the number of holes in a wheel as it rotates.

Although the breakbeam sensor is analog, it can often be used as a digital sensor. In most applications, the use of the sensor is digital in nature and involve measuring whether the light is blocked or not blocked. Conveniently, the sensor's output values for these two situations are valid digital signals, so the sensor can be used in a digital application.

4.3.3 Sharp Distance Sensor

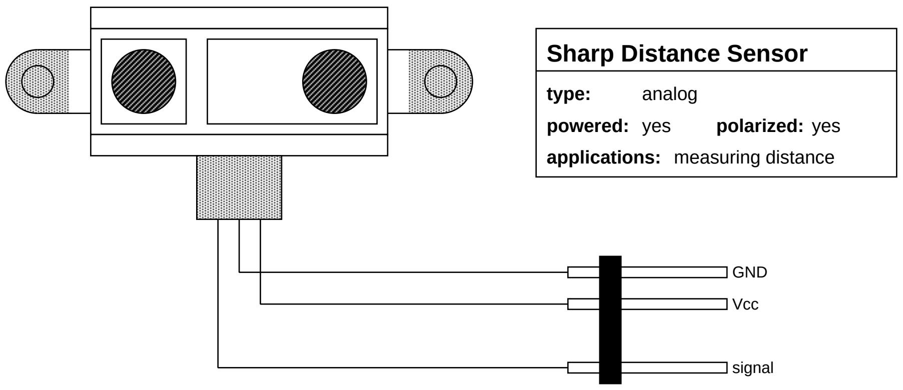


Figure 4.8: Sharp Distance Sensor

The Sharp GP2D12 Distance Sensor is capable of precisely measuring distances to walls or objects, using near-infrared light. Commercially, devices like these are used in a variety of places, from automatic toilets and sinks to photocopiers; your robot can use it to follow walls and avoid obstacles. This sensor is the only one which can detect things more than a few inches away—it has a useful range from about 6.5″ to several feet. The package has two parts:

1. an emitter that emits a narrow beam of barely visible light; and
2. an array of detectors that measure the angle of the spot the emitter projects on the wall.

Unlike other devices used in 6.270, this device supplies an analog value on its own, without the use of a pull-up resistor. Therefore, to use the sensor, you must make a slight, reversible modification to the Handy Board—cut the trace on the bottom of the main board that connects an analog input to its pull-up resistor. Please see a member of the staff for help with this procedure.

The sensor does not give you the distance in any type of standard unit. Instead, the value read on the analog input it is connected to varies smoothly from 0 to 255 as the distance decreases. Tests have found that the function

$$f[n] = -3.16'' + \frac{950.''}{8.58 + n}$$

gives an accurate estimate of the distance as a function of the analog value n, and that the reading is fairly independent of lighting, object color, or battery power level. However, the numeric parameters may have to be recalibrated for each individual sensor if an accurate measurement is desired.

4.4 Gyroscopes

From year to year we try out new sensors to improve the robots' performance.

For this year's competition, Analog Devices is making available surface-micromachined integrated circuit gyroscopes which can be used to measure your robot's rate of turn or (with some numerical integration) angle of turn, which can be a signicant aid to navigation. These devices integrate the mechanical parts of the gyro along with the necessary circuitry on a single chip, so they are very small (about 7mm square). To make them easier to handle and use, the gyros are supplied on 1.25" square printed-circuit boards containing all necessary external components and a pin header for connections. A complete data sheet for the gyros (ADXRS300) can be found on the Analog Devices website, www.analog.com (select iMEMs Gyroscopes under the MEMs Technology category). However, most or all of what you need to know to use the gyros successfully in your robot will be presented here and in a lecture during the First week of the course.

A gyro measures rate of turn, and these gyros have a full-scale range of +/-300 degrees per second. They operate on a single 5V power supply, as supplied by the Handyboard's I/O channels, and produce a single voltage output in the range of 0 to 5V, so they are wired and used like most other three-wire sensors (they are not, however, ratiometric to the power supply like accelerometers or potentiometers). The output of the gyro when it is not rotating is nominally 2.5V, midscale on the Handyboard's A/D converter. The nominal sensitivity of the gyro is 5mV per degree per second of rotation. (Calibration routines may be used to take out the gyro's offset and sensitivity errors at the beginning of a run.)

Since the A/D converter converts its whole 0 to 5V input range into 256 total codes, its resolution is only a bit better than 20mV. This means that the A/D can only resolve the output of the gyro to within about 4 degrees per second. To improve the resolution of gyro, Analog has designed and built a specialised board for 6.270. This board introduces some noise into the gyro output voltage, which allows us to "dither" the reading, increasing the effective resolution of the gyro.

To get from rate of turn to swept angle, it is necessary to take a series of readings of the gyro's output and do a simple numerical integration. Techniques and code for performing the integration will be presented in lecture and in a handout.

Figure 4.9 shows the mechanical configuration of the gyro board and the necessary connections. To operate properly, the gyro board should be mounted in an approximately horizontal position, assuming that you intend to measure the turn angle (yaw) of your robot. It can be fastened in place with double-sided-sticky tape.

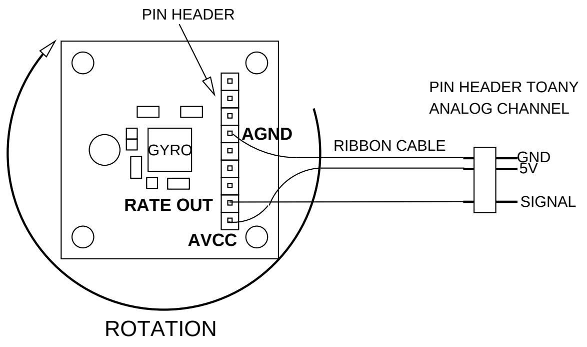


Figure 1. Gyro board showing axis of rotation and connections.

Figure 4.9: Mechanical configuration of the gyro board.

Chapter 5

Robot Construction

Most people consider LEGO to be a childhood toy, but the LEGO Technic system provides an excellent construction material for building robots. Since the pieces can be taken apart as easily as they are put together, no design must ever be final. It frees you from drawing out detailed plans and machining parts, so you can spend more time learning about and designing your robot. You can experiment with building, redesigning, and rebuilding components of your robot until you are satisfied with the results.

The best way to learn how to build with LEGO pieces is to play with them, but this chapter will present an introduction to a number of building techniques and design ideas. It is meant to provide you with the basic knowledge you will need to begin exploring and learning on your own. Reading this chapter, however, is no substitute for actual hands-on experience.

5.1 Design Concepts

When beginning work on a task as complex as building a robot, it helps to follow good design techniques. Although it requires continual practice to hone these skills, keeping the following ideas in mind while you build can help you produce a successful robot:

- **Simplicity.** The best way to build a reliable robot is to keep it as simple as is reasonably possible. In general, the more complicated a design is, the harder it is to build, and the more prone it is to failure. Try to minimize the number of moving parts and the overall complexity of the robot.

- **Strength.** During the course of operating and transporting your robot, it will be bumped and handled quite often. This can easily damage the robot and cause its performance to degrade over time. Building a strong robot will minimize the amount of time spent on repairs and will improve its overall performance.

- **Modularity.** Often, it will be necessary to upgrade or repair a component of the robot, but if the robot is built as one monolithic unit this may make it necessary to disassemble a substantial portion of the structure. If, however, you design your robot as a group of connected modules, the appropriate module can simply be removed and rebuilt.

5.2 The LEGO Technic System

The Technic system is similar to the LEGO parts that you may have played with as a child, except that in addition to the regular bricks and plates, this set includes pieces for building more complicated structures and moving parts. These components allow you to create robots and other wonderful things, but you must become familiar with their functions before you can use them effectively.

5.2.1 LEGO dimensions

The first thing you will notice about the LEGO parts in your kit is that the structural pieces come in a variety of sizes and shapes, but can be roughly grouped into two sets according to their height. The taller ones, bricks and beams, are $\frac{3}{8}''$ tall, while the shorter ones, flats and plates, are $\frac{1}{8}''$ tall. These are convenient measurements, since three flats can be stacked to equal the height of one brick. The dimensions of these basic LEGO pieces are shown in Figure 5.1.

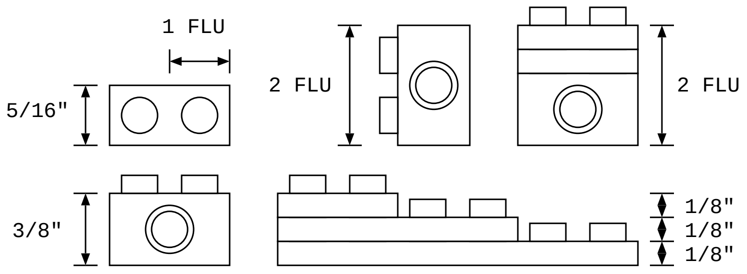


Figure 5.1: LEGO Dimensions

The curse of LEGO is that neither of these heights is the same as the standard LEGO width. Instead, this distance, the fundamental LEGO unit (FLU), is $\frac{5}{16}''$, making the ratio of height to width of a LEGO beam 6:5. All is not lost, though, because with some creative stacking, you can make vertical spacings which are integral multiples of horizontal spacings.

The simplest such stack is one beam and two flats. This yields a structure with a height of 2 FLU ($\frac{3}{8}'' + \frac{1}{8}'' + \frac{1}{8}'' = 2 \times \frac{5}{16}'' = 2$ FLU). This, as you will find, is a very important property, and you should remember it. With a little experimentation, you can construct any other even number of FLUs, though odd heights are not possible.

5.2.2 Beams, Connectors, and Axles

One of the most important types of parts in the LEGO Technic system is the beam. Beams are long structural pieces with holes through their sides. Besides their obvious use as structure components, they can be used in conjunction with other pieces to build elaborate structures.

The connectors fit into the holes in the side of the beams and allow them to be joined side to side. This frees you from only being able to stack pieces on top of one another, thus opening up the ability to build significantly more complicated structures. Since the connections created in this manner can be rotated to any angle, you can even introduce diagnol constructs to your robot or create moving joints.

Note that the two types of connector are functionally different. The black ones fit more snugly into the holes and resist rotation. The gray ones, on the other hand, rotate freely inside the holes for use in moving parts. It is alright to use the gray connectors in place of the black ones, but using a black connector in a moving joint will damage the connector and hole.

The holes through the beams also serve a further function when coupled with axles. The axles can be passed through a hole, and if supported properly between multiple beams, can rotate freely. This allows for the construction of the gearboxes necessary to drive the robot, as will be discussed later in this chapter.

5.3 Bracing

In order to build a strong robot, you will have to master the technique of bracing. Structures built simply by connecting pieces together with their nubs will not be able to handle the stresses imposed on them by the operation of the robot. Instead, you must find a way to augment the structure with braces to make it stronger.

The basic idea of bracing is to create a stack of pieces between two beams so that the holes in the top and bottom beams are separated by an integral FLU spacing (actually, only even numbers are possible). Then, using the connectors, you attach a beam vertically alongside the stack so that it holds the pieces together. Such a stack can be built in a number of ways, but a simple one is shown in Figure 5.2. The concept is simple but important for building an effective robot.

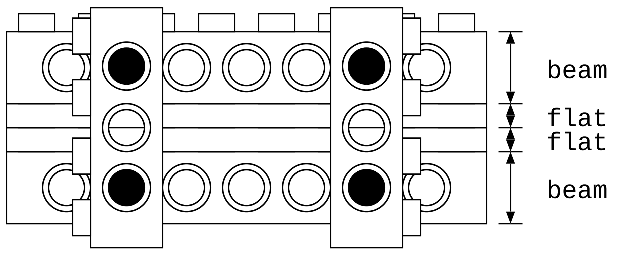


Figure 5.2: A Simple Braced Structure

Since bracing imposes constraints on how a structure can be built, it will be necessary to consider how your robot's structure will be braced from very early on in the design process. With experience, you will be able to build robots which can carry heavy loads and resist falling apart even when dropped on the floor. You will also be able to determine where braces are needed (and also of importance, where they are not).

5.3.1 Drop Testing

How do you know if your structure is strong or not? If you are daring, drop it on the floor from about waist height. If it shatters into little pieces, it failed the test. If it only suffers minor damage which can be easily repaired, you can be fairly certain that it can handle the rigors of everyday life. In the past, some particularly strong robots have been known to drive off of tables and still be in working condition.

Drop test at your own risk.

5.4 Gears

Most electric motors are really lacking in torque, or in other words, they cannot push very hard. If you hook a wheel directly up to the motor's shaft, you will find that it can hardly turn the wheel, let alone budge an entire robot. What they do have a lot of, though, is speed. In fact, when you let the shaft run freely, it can spin at a rate of thousands of revolutions per minute. This is much faster than you want your robot to drive anyway, so you will have to build gearboxes to trade some of this speed for more torque.

The LEGO Technic system contains a wide variety of gears with varying functions, but for building simple gearboxes, you will mostly rely on the 8, 24, and 40-tooth gears

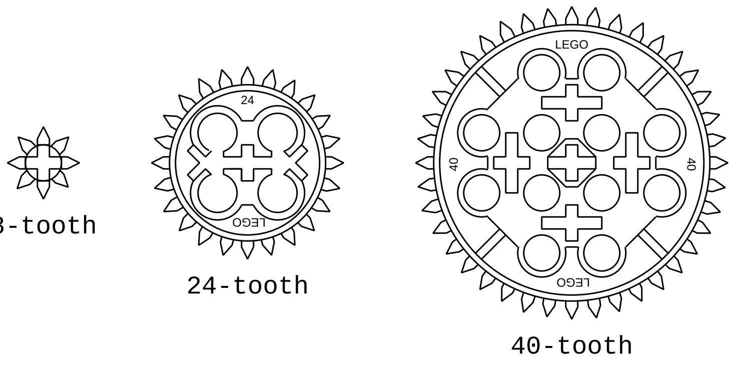


Figure 5.3: LEGO Gears

shown in 5.3. These are the most efficient and easiest to use of the bunch because their diameters are chosen such that they can be meshed with each other at regular LEGO distances. It is recommended that you begin by using only these gears at first, and then only use the other gears when you become an experienced builder.

5.4.1 Gearboxes

Gear reductions allow you to convert speed into torque (or vice versa by applying this technique in reverse). Suppose an 8-tooth gear is used to turn a 24-tooth gear. Since the smaller gear must rotate three times to turn the large one once, the axle with the 24-tooth gear spins slower than the other. In exchange for this decrease in speed, the axle is able to exert three times as much torque. This produces a gear reduction of 3:1, which means that you are giving up a factor of three of speed in exchange for producing three times the torque.

When a single gear reduction is not enough, it is possible to cascade a number of reductions in a gear box to achieve a higher gear ratio. For example, in Figure 5.4, two 3:1 and one 5:1 gear reductions are combined to create a 45:1 (3x3x5:1) gearbox. This means that the leftmost axle must turn 45 times in order to turn the rightmost axle (the output shaft) one time. If a motor with an 8-tooth gear was used to turn the 24-tooth gear on the leftmost axle, this would add another 3:1 reduction, bringing the total reduction 135:1.

There is no simple guide for choosing the gear ratio of a gearbox because it depends very heavily on the application. For heavy loads, high gear ratios will provide more force, but at the expense of speed. For fast, light robots, however, a lower gear ratio would be more appropriate. In order to find the correct match for your robot, you will have to experiment with a number of possible ratios.

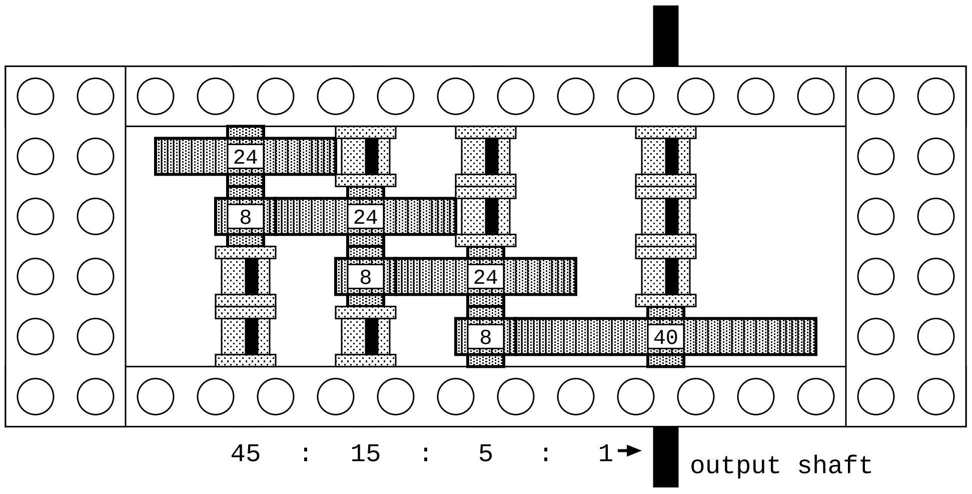


Figure 5.4: A LEGO Gearbox

5.4.2 Strange Gears

Occasionally, you will run into situations where the basic three gears are inadequate. In these cases, you may find that one of the strange gears will fit the purpose. You should use these gears sparingly, since they tend to be inefficient and prone to mechanical failure, but when they are needed, they can be life savers.

- *16-tooth gears* are just like the basic three described above, except that they only mesh straightforwardly with other 16-tooth gears. They are very efficient, but because of their antisocial behavior are only really useful for transferring force with no gear reduction.

- *Worm gears* look like cylinders with a screw thread wound around them. They act like a 1-tooth gear and are useful for building small, high-ratio gearboxes. They are extremely inefficient and tend to wear down quickly when subject to anything but the lightest loads. Avoid using these gears whenever possible and never use them in a robot's drive train.

- *Angle and crown gears* look similar to the standard gears, except that they have angled teeth. This allows them to be meshed at 90 degree angles with other gears, so they can transfer force around a corner. Since it is awkward to brace such a structure, working with these gears can be difficult as well as inefficient.

- *Differentials* allow two axles in the gearbox to divide the force between them while turning at different speeds. The differentials in your kit must be assembled by placing three angle gears inside the differential casing. Because of their complexity, they can be difficult to build into a gearbox, but they do fulfill a purpose that no other gear can peform.

5.4.3 Chain Drives and Pulleys

The chain drive is an invaluable tool for transferring motion from one place to another. It is assembled from the small connectable links and two or more regular gears (usually the 24 or 40-tooth gears). This allows it to transfer force from one gear to another. Unfortunately, though, the chain links are not sized to standard LEGO dimensions, so trial and error is often necessary to find a workable gear spacing. If the chain is too loose, it may skip under heavy load, and if it is too tight, you will lose power. Since the chain drive tends to be a bit inefficient, it is best when used in the lower stages of a geartrain.

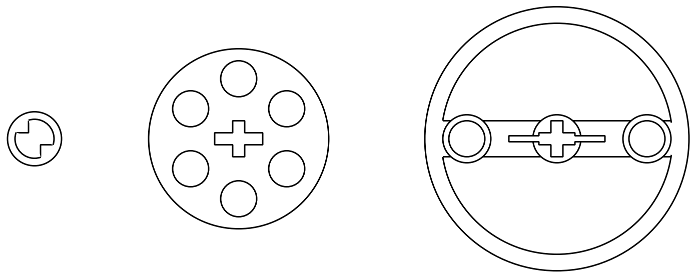


Figure 5.5: LEGO Pulleys

Pulley systems work in a similar manner and can be built using the pulley wheels shown in Figure 5.5 with a string or rubberband. They allow a great deal more flexibility in their arrangement than the chain drive, but they tend to slip easily under a load. They work best when used in the upper stages of a geartrain where there is the least amount of force. Be sure to make the string or rubberband the correct length. If it is the slightest bit too loose, it tends to slip, and if it is too tight, it will lose efficiency.

5.4.4 Efficiency

The biggest enemy of any gearbox is friction. Every place where something rubs, energy is lost which makes your robot slower and weaker. In the short-run, this causes your robot to perform poorly, but in the long-run, it will cause wear and tear on the moving parts. More damage means more friction, and after a while, the gearbox will stop working. In order to minimize the amount of friction in your gearbox and maximize its efficiency, follow the tips below:

1. The spacing between gears is very important. If they are too close to each other, they will bind up. If they are too far, the teeth will slip past each other. Make sure that gears are spaced at exact LEGO dimensions and avoid meshing gears at an angle.

2. The axles are made out of plastic and can bend if not properly supported. Try to always support the axle between two beams and do not place a gear more than one space outside of the supports.

3. The gearbox will often be subjected to stresses when used within a robot. Make sure that the beams supporting the axles are attached to each other with more than one cross-support and that the whole structure is braced. If the beams are not perfectly parallel, the axles will rub against the insides of the holes.

4. During operation the gears can slide along the axle or bump into nearby gears. Use spacers to fill in any empty spots along the axle.

5. Make sure that the axles can slide back and forth a tiny bit. If they cannot, the gears or spacers are probably pushing up against a beam. This is probably the most common (and easiest to fix) mistake which saps efficiency from a gearbox.

If you want to know how good your gearbox is, try backdriving it. Remove the motor and try to turn the output shaft (the slow axle) by hand. If your geartrain is efficient, you will be able to turn all the gears this way, and if it is really efficient, they should continue spinning for a second or two after you let go. If your gearbox cannot be backdriven, something may be wrong with it.

5.5 Drive Mechanisms

Perhaps the single most important aspect of a robot's physical design is its drive system. It is responsible for moving the robot from place to place by providing the appropriate motive force and steering mechanisms. Figure 5.6 shows the three most popular drive arrangements.

5.5.1 Differential Drive

A differential drive is much like the drive mechanism on a tank. It consists of two independently driven wheels arranged side by side. When both wheels are driven at the same rate in the same direction, the robot will move straight. When the wheels are driven at the same rate in opposite directions, the robot will spin in place. By varying the relative speeds of the two wheels, any turning radius is achievable. Since this system

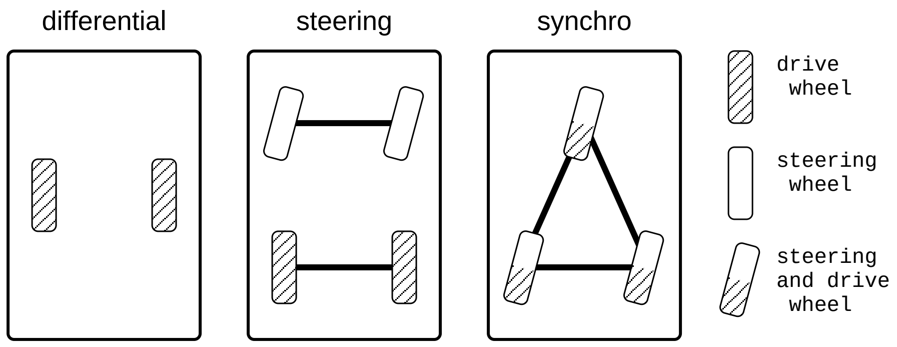


Figure 5.6: Popular Drive Arrangements

minimizes the number of moving parts, it tends to be the simplest and most robust. The complexity is increased slightly, though, by the need for a caster in the front or back of the robot to keep it from tipping over.

An especially useful property of this drive system is that the change in orientation of the robot depends only on the difference between the distances travelled by each of the wheels. It does not matter if the left wheel moves forward 10cm and the right back 10cm, or if only the left wheel moves forward 20cm; the final location will be different, but the orientation will be the same (ignoring slippage). This greatly simplifies the process of turning, by making the final orientation of the robot easily predictable.

The differential drive is the most popular because of its simplicity, though it does have some limitations. Since it is difficult to calculate the final location of the robot if it turns and moves at the same time, most navigation algorithms consist of driving straight for a distance, turning through a specified angle in place, and then driving straight again. More sophisticated algorithms allow the robot to turn while moving, but typically, such turns are still constrained to easily calculated curves.

5.5.2 Steering System

Steering systems should already be familiar to you because they are widely used in automobiles. They usually consist of one or two steerable wheels at the front of the robot and two powered wheels at the rear. The turning radius is determined by the angle of the steerable wheels, but the robot must be moving in order to make a turn. This means that it cannot turn in place and can only make turns of limited sharpness while driving.

The advantage to using a steering system comes from the separation of the steering and drive mechanisms. Such robots tend to be quick and fairly agile. They are well-suited to driving in open spaces or performing "follow" tasks since these tasks usually require making course corrections to the left and right as the robot drives. When a steering robot turns, though, the two rear wheels will take paths of different lengths. The outer one will travel a longer distance than the inner one, so it is helpful to use a differential in the gearbox to transfer force between the wheels in order to avoid slippage.

5.5.3 Synchro Drive

The synchro drive is an exotic mechanism where all the wheels are driven and steered together. Usually, there is one gearbox which turns the wheels to the desired orientation and then another which drives the robot in that direction. This may seem strange, but it allows the robot's instructions to be phrased in terms of world coordinates, instead of having to compute everything in terms of the robot's perspective. The robot can also be commanded to move in any direction, making this the most mobile of the drive systems.

This mechanism simplifies control at the expense of complexity in construction. All the wheels must be both steerable and drivable, so building drivetrains for such a system often requires an elaborate system of gears and chain drives. Also, since only the wheels turn, the robot's body remains in a fixed orientation, unless it is also turned. This makes it inconvenient for such a robot to have a front as many applications require.

5.5.4 Legs

Robots with articulate legs can do everything that a wheeled robot can do and more. This includes walking in arbitrary directions, turning in place, and even climbing over otherwise inaccessible terrain. Unfortunately, though, legged robots are prohibitively difficult to build out of LEGO and comparably difficult to control. In fact, a great deal of research is currently being performed to study ways of making robots walk. If you think you are up to the challenge, a legged robot makes a very exciting project, but be warned that building legs can be much more difficult than it appears.

Chapter 6

Robot Control

To the uninitiated, the term "robot" conjures up images of machines with human-like abilities. Unfortunately, technology has not yet reached the point where robots can mimic the intelligence of humans. Instead, the robot that you will be constructing will require simple, step-by-step instructions for completing even the most simple of tasks.

A robot's ability to interact with the environment centers around its sensors and control system. The sensors convert information about the environment into a form that can be used by a computer. They are limited in their abilities, however, and often give back information that is cryptic, ambiguous, or even inaccurrate. The control system must decode this information and determine the best course of action. The task of endowing the control system with these abilities falls to you, the programmer. This chapter will explore the design of systems for controlling a robot in a constantly changing and unpredictable environment.

6.1 Control Systems

Control systems is an entire field of study and reducing it to one section of one chapter of one book certainly does not do it justice. The material presented here covers only the very basic concepts, but for this course, it will be sufficient for your needs. If you are interested in understanding these concepts in more depth, there are many relevant courses and a large body of literature dedicated to the subject.

6.1.1 Open Loop

The most obvious approach to programming a robot is to give it a sequence of instructions to follow. The robot then executes its orders without worrying about the consequences of its actions. Information flows only from the controller to the actuators to the world as shown in Figure 6.1.

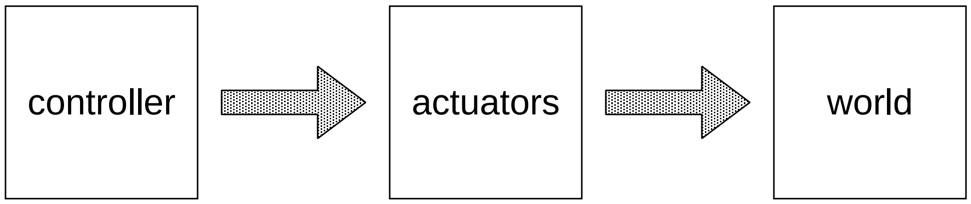


Figure 6.1: Open loop information flow

Although the controller can send commands to the actuators, it cannot tell whether or not the correct action occurred. Information flows from the controller to the world, but not back again. For this reason, such a system is known as *open loop* control. Open loop control is rarely used in the real world because it lacks robustness. Small changes in the robot and environment cannot be accounted for, and after a while, error begins to build up. Even the smallest errors will eventually build up to a point that the robot becomes lost.

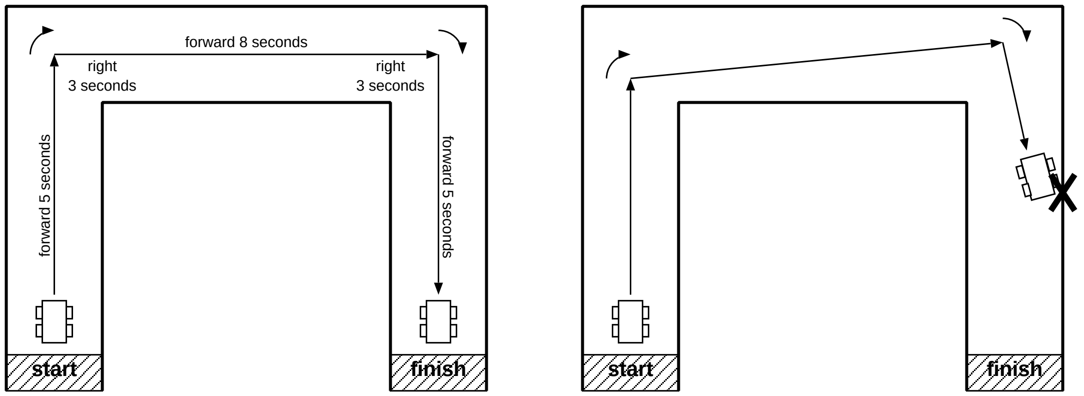


Figure 6.2: A robot trying to navigate with open loop control

Consider, for example, the path the robot must take to navigate the course in Figure 6.2. On the left is the path that the robot is supposed to follow labelled with the appropriate instructions. As the batteries lose power, though, the speed of the robot will decrease. Since the durations of each action are no longer valid, the robot might take the path shown at right, leading to a collision with the wall. Clearly, open loop control is not going to get the job done.

6.1.2 Feedback

To avoid the problem from above, the robot needs to take advantage of some of the information available in the environment. The robot can use its sensors to correct errors and compensate as needed before the situation gets out of control. This closes the loop of information flow, as shown in Figure 6.3.

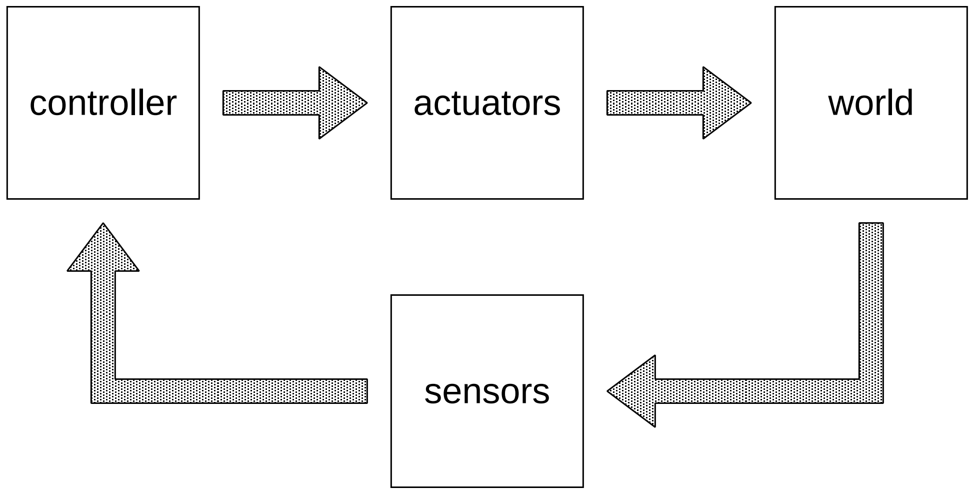


Figure 6.3: Closed loop (feedback) information flow

The feedback approach requires a different way of thinking about the problem. Rather than performing a single action designed to move the robot to its goal, the robot repeatedly makes small corrective actions in response to its current situation. Through repetitive application of these small maneuvers, the robot eventually achieves its overall desired goal.

Figure 6.4 shows how the robot can apply feedback control to the situation from above. In this example, the robot is repeatedly applying the following set of rules:

1. If the front of the robot is in contact with a wall, back up to the right.

2. If the left of the robot is in contact with a wall, turn right.

3. If the right of the robot is in contact with a wall, turn left.

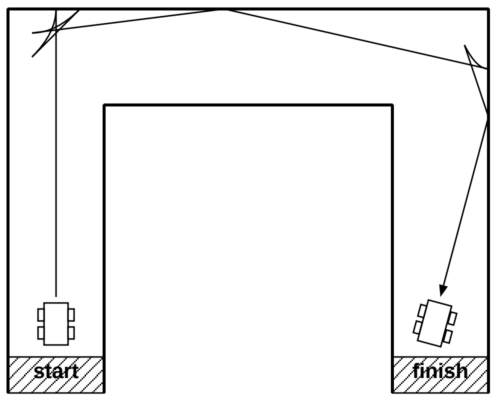


Figure 6.4: A robot using feedback to navigate

4. Otherwise, drive forward.

Rule 4 is the default behavior of the robot. Whenever it is out in the open, it simply drives forward. Rules 2 and 3 allow the robot to drive down a corridor. Whenever it bumps into one of the walls, it turns away from it and continues down the corridor. Rule 1 turns the robot when it reaches a corner. The robot repeatedly runs into the wall and backs away from it, each time turning a little bit more. After a few collisions, the robot completes the turn and continues down the next segment of the course.

With this algorithm, a number of assumptions about the robot's performance have been relaxed. It does not matter how fast the robot moves or even if it drifts a bit to one side as it drives. The algorithm is now much more robust because there are less unseen factors in the environment and the robot that can make it perform incorrectly. The use of feedback has greatly improved the design.

6.1.3 Open Loop Revisited

The primary drawback to using feedback is that some tasks require a large amount of time to complete. In the feedback example above, the robot had to make a number of forward and backward motions in order to go around a corner. This takes a lot of time and can be a major handicap for speed-critical applications. It would be nice if there was a way to turn more quickly.

Fortunately, open loop control may be good enough for some situations. Instead of the feedback-only algorithm, a hybrid algorithm can be used as in Figure 6.5. The robot navigates the corridor using the feedback algorithm, except that when it runs into a wall, it backs up a little and performs the quicker open loop turn instead of the

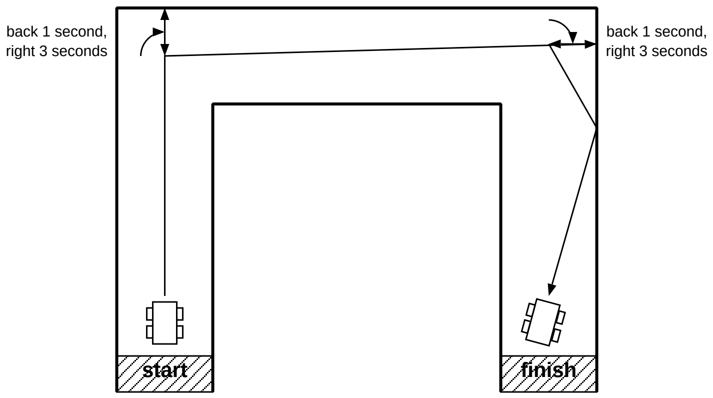


Figure 6.5: A robot combining open loop and feedback control

bumping method. After completing the turn, it then returns to the feedback algorithm for navigating the next corridor.

As long as the open loop turn is accurate enough that the robot does not get stuck in a corner, the feedback mechanism will be able to compensate for any error that it introduces. Since the error is erased in this manner before the next open loop command, it cannot snowball out of control like in the purely open loop algorithm. By using this clever sandwiching of open loop and feedback control, it is possible to develop an algorithm that has most of the efficiency of open loop control while only sacrificing a little of the robustness of feedback control.

6.2 Sensors

Sensors allow a robot to collect information about the environment. They convert physical measurements from the world into electrical signals that can be understood by the controller. This gives the robot a link to the real world allowing it to respond appropriately to changes in its environment.

6.2.1 Sensor Problems

The real world is a noisy place compared to the digital world of a computer, and in robotics, this noise can make sensor readings unreliable. Sensor data mirrors the nonideal nature of the robot's environment and must be interpreted by the computer. Below are a few of the common types of errors that appear when reading sensors:

- **Glitchy data** is often a problem due to faults in the sensor hardware. When a glitch occurs, the sensor may return an unexpected value or a value outside of the range of possible values. Often, loose connector plugs or shorted wires can cause spurious input by intermittantly losing or making contact. This can be identified and fixed in software by ignoring values that fall outside of the expected input.

- **Noisy data** is a problem for most sensors. The world is full of flickering lights, uneven surfaces, and magnetic fields which can all cause errors in measurements. Since noise tends to be a random process, there is no way to predict it, but its effects can be minimized by averaging multiple values together when reading a sensor.

- **Drifting data** is often the result of sensors retaining some sort of memory. This problem is particularly prominant in light sensors which are affected by changes in the ambient lighting of the room. As the robot moves from place to place, the amount of light reaching the sensor can vary and change the range of the measurements. Some light sensors are also sensitive to heat and return different values as they warm up. These effects occur slowly over time, so they can be very difficult to detect. One option is to give the robot the ability to recalibrate itself whenever it finds itself in a known situation. This way, calibration values can be updated on the fly as the robot moves from one part of the environment to another.

6.2.2 Bouncing Switches

When a mechanical switch closes, two conductive contacts are physically brought together. When they first touch lightly, conduction may be intermittent due to the presence of insulating contaminants on the contact surfaces. Even after good contact is established, the contacts may momentarily separate and reconnect one or more times as a result of the movable contact bouncing after it lands (the moving part of a switch is usually somewhat springy and has nonzero mass, so it resonates). Intermittent contact can also occur as a switch opens. All of these phenomena are generally lumped under the heading of ”switch bounce”. They all give the appearance of multiple switch events when only one actually occurred. All of these effects also respond to the same treatment: after the first detected closing or opening of a switch, wait an appropriate amount of time (10ms - 20ms is usually sufficient) and then check again to see if the contact is still closed (or open). Adjust the delay time until you reliably detect a single event for a single motion of the switch. It is also possible to check several times over shorter time intervals if time is critical in your application.

6.2.3 Calibration

Light sensors are particularly sensitive to changes in the world. Differences in ambient lighting from one room to another can wreak havoc on the readings that the sensor returns. When the robot needs to be moved from one environment to another, calibration is often necessary to adjust the settings used to interpret the data.

Even though most light sensors are analog in nature, you will often use them as if they were digital sensors. Rather than asking the question, “How bright is it?,” you will instead ask “Is it light or dark?” The goal of calibration, then, is to choose a threshold value which will separate light values from dark values.

The most common way to choose a threshold is to take readings in a controlled situation, such as during a calibration routine. By averaging two readings, one for light and one for dark, a value can be chosen to separate the two conditions. Then, each time a the sensor takes a reading, it can be compared to the threshold to determine if the sensor is seeing light or dark.

6.3 Simple Navigation

Robot navigation is a difficult problem, but it can often be decomposed into a number of subtasks. These simple tasks can be implemented with feedback mechanisms and represent the basic skills that most 6.270 robots utilize as part of their grand strategy. Calling them “simple,” however, is a bit misleading because, as you will see, making them work together reliably requires a lot of fine tuning and patience.

6.3.1 Wall Following

Imagine yourself in a dark hallway. You have to get to the other end, but you cannot see anything. What do you do? If you are like most people, you probably begin by finding one of the walls with your hand. Then, as you walk, whenever your hand detects that you are too close to the wall, you move farther from it, and when you are too far, you move closer. This, it turns out, is pretty much the way your robot will do it.

Wall following is very simple because it only requires a single sensor mounted on one of the front corners of your robot. This sensor will deliver just one bit of information which determines whether the robot is touching the wall or not. To follow the wall, the robot then executes the algorithm mentioned above:

- If the robot is touching the wall, turn away from the wall.
- If the robot is not touching the wall, turn towards the wall.

This will cause the robot to repeatedly bump into the wall, as shown on the left side of Figure 6.6, alternately activating and releasing the sensor. This oscillation can be minimized by tuning how sharply the robot turns or by using a sensor which gives more precise distance information.

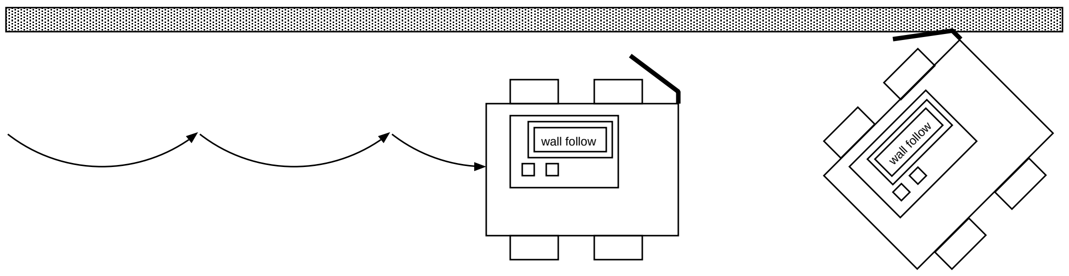


Figure 6.6: Wall following and a jammed robot

While following the wall is easy, finding it in the first place can often be difficult. If the robot begins close enough to the wall, it can simply use the wall following algorithm to find it. If the robot approaches the wall at a steep angle, however, it is likely to jam. When the robot's initial position is unpredictable, it is advisable to use some strategy for aligning with the wall before trying to follow it.

6.3.2 Line Following

Line following is usually accomplished by mounting one or more reflectance sensors on the underside of the robot. By measuring the intensity of the reflection, these sensors can determine whether they are over a light or dark area. By adding a little "intelligence," the robot can be made to follow lines.

The number of sensors and the configuration used depends on the robot and application, but the method is almost always the same. The sensors detect when the robot begins straying from the line, and the controller issues orders to correct for the error. Developing this algorithm begins with constructing a chart of all possible combinations of sensor inputs and the appropriate action for each. This information can then easily be coded into the robot's program.

Figure 6.7 shows the table for constructing a program for a robot with three reflectance sensors arranged in a line across the robot with the left and right ones spaced wider than the line. Whenever the robot begins to drift, one of the outer sensors detects the line and tells the robot to correct itself. If the robot continues to stray, the middle sensor loses the line and tells it to turn sharper. It is interesting to note that in theory, some of the states cannot occur, but in practice, erroneous sensor readings and unexpected situations can cause them to arise. It is usually wise to assign best-guess actions to these states to make sure that the robot always has something to do.

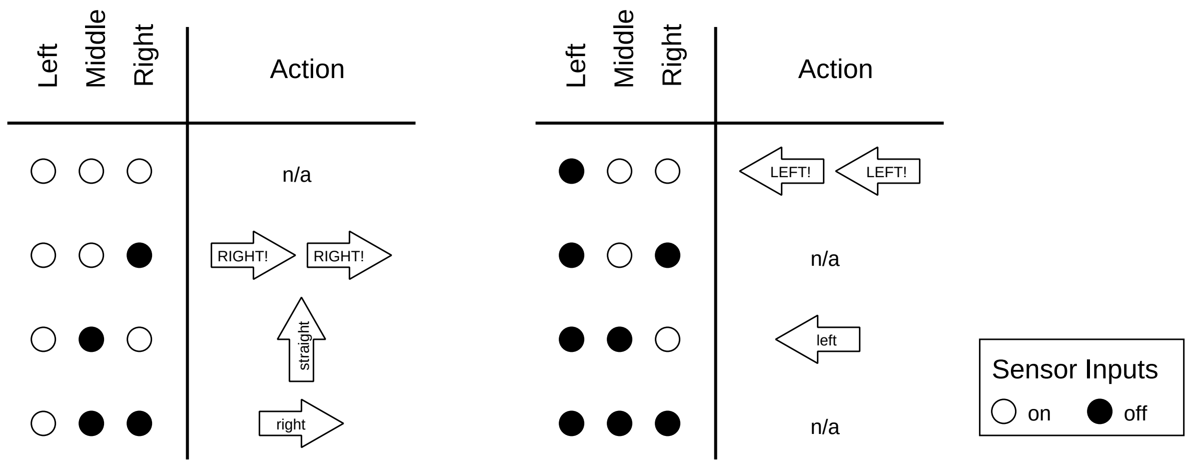


Figure 6.7: Line following with 3 reflectance sensors

6.3.3 Shaft Encoders

Shaft encoders are a wonderful tool for robot navigation. When placed on a wheel or in the wheel's accompanying gear box, encoders can very accurately measure the distance the wheel has travelled. This provides the robot with a number of abilities, including measuring distances, driving straight, and turning accurately.

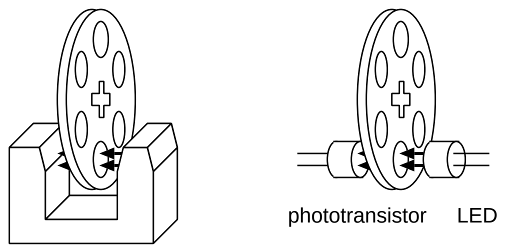


Figure 6.8: Shaft encoding using a LEGO pulley wheel

A shaft encoder is constructed from a breakbeam sensor and a Lego pulley wheel as shown in Figure 6.8. As the wheel turns, the holes in the wheel alternately allow and then block the light from hitting the detector in the sensor. The robot can then count the number of passing holes and compute how far the wheel has turned.

If both sides of the robot are driven at the same rate, the robot will move in a straight line. For a robot with a differential drive system, the easiest way to do this is to place a shaft encoder into each of the drive trains and monitor the distance each wheel has travelled. Whenever one wheel gets ahead of the other, it is slowed down.

The problem of driving straight can be solved with a very simple algorithm. Whenever the left side of the robot is ahead, the right wheel is driven faster, and the left is slowed down. When the right side of the robot is ahead, the opposite action is taken. The robot constantly corrects for small deviations allowing it to drive in a straight line.

Shaft encoders are so useful that many robots make extensive use of them, but they are not the entire solution to robot navigation. The feedback provided by the encoders comes not from the environment, but rather from within the robot. Slippage of the wheels on the floor is not accounted for, and can lead to measurement errors, especially when turning. In order to correct for these errors, additional feedback mechanisms must be used in conjunction with the shaft encoders.

6.4 Timeouts

Control systems often fail when something unexpected happens, and the available sensor information is not able to account for the situation. Most of the time, when a robot gets lost, it will wind up stuck on some obstacle. If none of the sensors register the collision, the robot will not even know that anything is wrong. It will continue trying to drive, oblivious to the fact that it is not getting anywhere. If the robot does not reach its goal after a certain amount of time, though, it can usually assume that a problem has occurred along the way.

When an action times out, the robot only knows that something has gone wrong, but not what it is. Regardless, the robot can often act to increase its chances of recovering. In many cases, backing up a little or thrashing around may be sufficient to free the robot from an obstacle so that it can continue on its course. In any case, detecting that something has gone wrong can considerably increase the robots ability to make the best of a bad situation.