2002 6.270 Autonomous Robot Design Competition Chicken

Course Notes

Last Updated: January 7, 2002

Massachusetts Institute of Technology

http://web.mit.edu/6.270/www/

2002 Sponsors

We thank this year's sponsors:

- **MIT Department of Electrical Engineering and Computer Science**
  Financial funding, laboratory facilities, staff help and support
- **Analog Devices**
  MEMS accelerometers
- **Ford Foundation**
  Financial funding
- **General Motors**
  Financial funding
- **Gleason Research**
  HandyBoard Controllers
- **Guidant**
  Financial funding
- **Hawker Energy Products, Inc.**
  Lead-acid batteries, battery shrink wrap
- **LEGO Dacta A/S, PITSCO**
  PITSCO LEGO packs
- **LEGO Denmark**
  Bulk LEGO plates, tires, hubs
- **LEGO Shop-at-Home**
  LEGO boxed sets
- **Microsoft**
  Financial funding

- **Newton Labs, Inc.**
  Iteractive C
- **Seagate Corporation**
  Financial funding
- **Sharp Electronics**
  Distance sensors

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, and require no human intervention.

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 MIT classes (e.g. 6.001, 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, being introduced to some material in class, and hacking on their robots.

6.270 is a very 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 these people for help or with any questions you might have. The staff consists of two groups of people, Organizers and teaching assistants, each with different responsibilities, but all of them will be available to assist you.

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 recruited by the Organizers to assist in teaching the course. Their job primarily occurs during IAP and 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 and are here only because they love 6.270 and want others to have the same opportunity to take 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.

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://www.mit.edu/~6.270/staff/
```

1.2 Kits and Tools

The 6.270 kit, which is 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 kit when asked will be charged the full $1500 through the Bursar's office.

In addition to the kit, a set of tools will be reserved for purchase by your team. This set will include all of the tools necessary for building your robot (i.e. soldering iron and stand, wire cutters, long nose pliers, etc.). You will be expected to either provide your own electronic assembly tools or purchase them from the Organizers. Since 6.270 buys

Organizer	Email	TA	Email
David Bailey	dbailey	Neil Basu	ssb
Jessica Forbess	jforbess	Lauren Bradford	lhb
Ehren Foss	ehren	Dean Christakos	dean
Jon Lee	jonlee	Jim Hardison	hardison
Jan Malasek	jmalasek	Kailash Hiremath	kailash
Erica Peterson	astronut	Teresa Ko	tko
Manu Seth	manu	Paul Ojanen	pro
		Goutam Reddy	greddy
		Javed Samuel	javed
		Neil Sanchala	nsanch
		Benji Sterling	benster
		Nick White	nickw
		Mouser Williams	mouser

Figure 1.1: 2001 6.270 Staff and email list

in bulk, the prices of the tools will be lower than what you can find elsewhere. It is very important that you have a good set of tools to work with.

1.3 Electronic Communication

Due to the quick pace of the course, information must often be distributed quickly to large numbers of people. To accomplish this, 6.270 primarily uses electronic communication.

1.3.1 Mailing Lists

Email is the primary medium through which important announcements are sent. Since this information must often reach the entire class on short notice, participants are encouraged to check their email daily.

- **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 the email list to use if you have a question about the rules, contest proceedings, or the validity of your robot. The Rules Committee is comprised of a subset of the Organizers. This year's members are: Erica Peterson and Jan Malasek.

- **6.270-participants@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.

1.3.2 Zephyr Instance

The course uses a zephyr instance for 6.270-related discussion. It is meant to be a forum where participants can help each other with problems they are having. The staff does not officially monitor the discussion, but will often be online to help out.

To receive zephyrs on the instance, you should type at the Athena prompt:

```
zctl add message 6.270 "*"
```

To send a message to the instance:

```
zwrite -i 6.270
```

If you wish to remove yourself:

```
zctl delete message 6.270 "*"
```

1.4 Laboratory Facilities

During the course of constructing your robot, you will need access to the appropriate workspaces, tools, and computers. These will all be made available to you by the course.

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 contest tables for testing.

The lab will be open and staffed from 9 am to 11:45 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, you will be expected to be on your best behavior. Do not touch equipment not explicitly meant for 6.270 use and treat the lab staff with respect. Be aware that when the equipment desk workers are ready to close the lab, you should be going out the door. Abuse of the lab or its staff will not be tolerated.

1.4.2 Other Facilities

If you have the appropriate cable, you can also program your robot at most Athena workstations. You may not, however, solder, cut, or glue in the clusters, and you must be respectful of others when operating your robot, since robots can be quite loud. Violations of Athena etiquette will result in severe action by the 6.270 Organizers.

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. Unfortunately, due to the staff's limited amount of time, technical support for personal computers must take low priority with respect to other duties.

1.4.3 Etiquette

When working in the lab or at Athena, you will be expected to be respectful to those around you. The following guidelines should be adhered to 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.

3. *Tidiness.* Do not leave your stuff laying around lab. The lab will be crowded and people 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. *Locked Screens.* At Athena, do not leave your screen locked for more than 20 minutes. In lab, any computer with a locked screen will be logged off. Repeated violations will result in a loss of computer privileges.

5. *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 tolerated and will be dealt with severely. If the Organizers receive complaints about any team causing a disturbance in the Athena clusters, that team will be required to return its kit and will be thrown out of the course. Repeated violations in lab will be dealt with by the Organizers on a case by case basis.

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 you will be doing a lot of work anyway. 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.** Each team must create a web page for its robot before impounding. The page should present information about the robot suitable for display to the general public. It should focus on the overall design and strategy of the robot including an explanation of anything particularly clever or unique. Each individual desiring credit must help with the work.

- **Assignment Completion.** Four assignments will be handed out during the first three 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.

- **Completed Robot.** The team must "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.

These requirements are meant to be useful to both you, the class participant, and the instructors, who will be authorizing credit. You should have no trouble receiving credit if all of the requirements are satisfied. If you have any questions about your standing in the subject at any time, feel free to ask your instructor for feedback.

Please note that due to the scheduling constraints of the Registrar and the sanity of the Organizers, there is no leeway on any of the due dates. Please do not ask for extensions.

1.6 Schedule

The schedule of activities between the start of 6.270 and the evening of the contest is very tight. You will have to work steadily and with determination to produce a working robot by the end of the course. To assist you in this endeavor, a number of class meetings will be held to teach the course material. It is recommended that you attend as many of these sessions as possible.

- **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.

1.6.1 Important Dates

While it is important that you attend all of the scheduled 6.270 events, the following meetings are mandatory and should not be missed:

- **Parts Sorting Session** - *Sunday, January 6th, 2 pm, Room 38-600*
  Each team must provide one person-hour of manual labor to help sort out the kit parts. Usually, this session is a lot of fun as you get to meet other people in the class and see all the kit parts.

- **Lecture 1** - *Monday, January 7th, 10 am, Room 34-101*
  Each team must have at least 50% of its members present to claim the kit.

- **Contest, Qualifying Round** - *Monday, January 28th, 2 pm, Room 26-100*
  All robots must compete in the first round in order to qualify for the remainder of the contest. This event is open to the public. See Section 2.3 for more details on the contest rounds.

- **Robot Impounding** - *Wednesday, January 30th, 5 pm, Room 38-600*
  All work on robots must cease and robots will be impounded. During this time, robots will be inspected for rule violations.

- **Contest, Second Round** - *Thursday, January 31st, 11 am, Room 26-100*
  All qualifying robots will compete in this round. This event is open to the public. This round may be cancelled if few enough robots qualify for the competition. See Section 2.3 for more details on the contest rounds.

- **Contest, Final Round** - *Thursday, January 31st, 6 pm, Room 26-100*
  The main competition. Robots will compete until a winner is decided. This event is open to the public. See Section 2.3 for more details on the contest rounds.

- **Lab Cleanup** - *Friday, February 1st, 2 pm, Room 38-600*
  Attendance at this session is mandatory. Each team must provide one person-hour of manual labor to help clean up the lab, so we can return it to 6.111 the way we found it.

1.6.2 Syllabus

Lectures and workshops will be the primary way that material is taught, so attendance at these sessions is very important. Workshops run on the hour from 12-3 pm and 7-9 pm. The following topics will be covered:

- **Lecture 1 - Welcome**
  Monday, January 7th, 10 am, Room 34-101
  Introduction to 6.270, Contest Rules, Fundamentals of LEGOs, kit distribution.

- **Workshop 1 - Soldering**
  Monday, January 7th, and Tuesday, January 8th, Room 38-600
  In 6.270 you will have to solder several components for your robot. Learn the basics of soldering. During this workshop you will have the opportunity to solder your battery recharger.

- **Workshop 2 - Basic Techniques of LEGO Assembly**
  Tuesday, January 8th, and Wednesday, January 9th, Room 34-301
  Bracing is a key element in keeping your robot together during the contest. Find

out how to do it effectively and learn how you can use some of the more mysterious parts of your LEGO kit.

- **Lecture 2 - Code & Sensors I**
  Wednesday, January 9th, 10 am, Room 34-101
  The HandyBoard; Sensors & Actuators; Basic Robot Skills

- **Optional Lecture 1 - Crash Course in C**
  Wednesday, January 9th, 9 pm, Room 34-301
  Come to this lecture to learn and review the basics of C programming.

- **Workshop 3 - Code I: Basic Control and Robot Skills**
  Thursday, January 10th, Room 34-501
  In this workshop you will learn how the HandyBoard communicates with its sensors and actuators. Get an introduction to IC and learn fundamental ways to make your robot move and turn. If you feel you do not know much about C at all, be sure to attend the Crash Course in C Tuesday night.

- **Lecture 3 - Code & Sensors II**
  Friday, January 11th, 10 am, Room 34-101
  Finite State Machines; Using the Beacon

- **Workshop 4 - Motor Mounting and LEGO Gearboxes**
  Friday, January 11th, Room 34-301
  Learn how to properly make a LEGO gearbox that works smoothly and efficiently. Learn more about gear ratios, and how to connect your motors to your gearbox.

- **Workshop 5 - Servos, Sensors, and Shaft Encoders**
  Monday, January 14th, Room 34-503
  Your robot cannot do anything useful without some knowledge of its external surroundings. Your robot will also need mechanisms for which it can navigate swiftly and accurately around the contest table. Learn how these parts in your robotics kit can help your robot move the way you want it to move. Find out which sensors work best for your robot's needs, and how to keep accurately track of how far or at which angle your robot has been moving. Also discuss how to use these unique parts in some more creative ways.

- **Workshop 6 - Code II: Advanced Techniques**
  Tuesday, January 15th, Room 34-501
  The code for your robot plays a crucial role in your robot's ability to win. Learn the value of good coding practices by observing robots with identical hardware trying to do the same task in different ways. Learn about error recovery techniques and how to achieve robustness in your code and strategy.

A mock contest will be held in the 6th floor lab at 7 pm on Saturday, January 26th. This is a good way to familiarize yourself with the contest proceedings, as well as find any bugs in your robot as you compete against other teams. Even if your team's robot is not ready, you can observe the strategies other teams have used. This contest is optional and not required for completion of this course.

	6 Sunday	7 Monday	8 Tuesday	9 Wednesday	10 Thursday	11 Friday	12 Saturday
Lab Hours		9 am - 11:45 pm	9 am - 11:45 pm	9 am - 11:45 pm	9 am - 11:45 pm	9 am - 11:45 pm	Noon - 10 pm
			Assignment 1 Due			Assignment 1 Due	
10:00 am		Lecture 1 (34-101)		Lecture 2 (34-101)		Lecture 3 (34-101)	
11:00 am							
Noon		Workshop 1 (38-600)	Wkshps 1 (38-600) & 2 (34-301)	Workshop 2 (34-301)	Workshop 3 (38-501)	Workshop 4 (34-301)	
1:00 pm		Workshop 1 (38-600)	Wkshps 1 (38-600) & 2 (34-301)	Workshop 2 (34-301)	Workshop 3 (38-501)	Workshop 4 (34-301)	
2:00 pm	Parts Sorting (38-600)	Workshop 1 (38-600)	Wkshps 1 (38-600) & 2 (34-301)	Workshop 2 (34-301)	Workshop 3 (38-501)	Workshop 4 (34-301)	
3:00 pm							
4:00 pm							
5:00 pm							
6:00 pm							
7:00 pm		Workshop 1 (38-600)	Wkshps 1 (38-600) & 2 (34-301)	Workshop 2 (34-301)	Workshop 3 (38-501)	Workshop 4 (34-301)	
8:00 pm		Workshop 1 (38-600)	Wkshps 1 (38-600) & 2 (34-301)	Workshop 2 (34-301)	Workshop 3 (38-501)	Workshop 4 (34-301)	
9:00 pm				Crash Course in C (34-301)			
10:00 pm							
11:00 pm							
Midnight							

Figure 1.2: First week schedule

	13 Sunday	14 Monday	15 Tuesday	16 Wednesday	17 Thursday	18 Friday	19 Saturday
Lab Hours	Noon - 10 pm	9 am - 11:45 pm	9 am - 11:45 pm	9 am - 11:45 pm	9 am - 11:45 pm	9 am - 11:45 pm	Noon - 10 pm
						Assignment 3 Due	
10:00 am							
11:00 am							
Noon		Workshop 5 (38-501)	Workshop 6 (38-501)				
1:00 pm		Workshop 5 (38-501)	Workshop 6 (38-501)				
2:00 pm		Workshop 5 (38-501)	Workshop 6 (38-501)				
3:00 pm							
4:00 pm							
5:00 pm							
6:00 pm							
7:00 pm		Workshop 5 (38-501)	Workshop 6 (38-501)				
8:00 pm		Workshop 5 (38-501)	Workshop 6 (38-501)				
9:00 pm							
10:00 pm							
11:00 pm							
Midnight							

Figure 1.3: Second week schedule

	20 Sunday	21 Monday	22 Tuesday	23 Wednesday	24 Thursday	25 Friday	26 Saturday
Lab Hours	Noon - 10 pm	closed	9 am - 11:45 pm	9 am - 11:45 pm	9 am - 11:45 pm	9 am - 12 am	24 hours

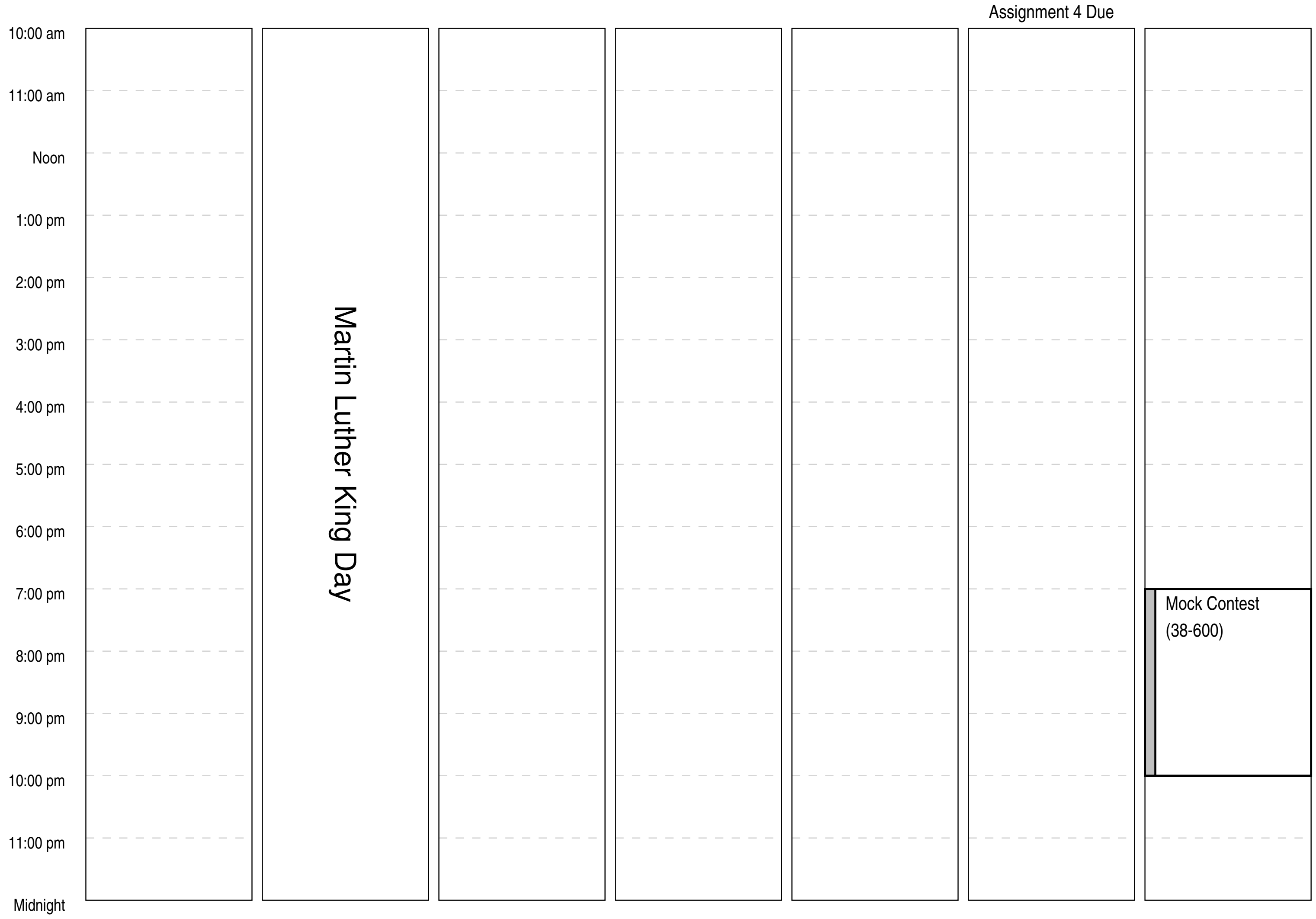

Figure 1.4: Third week schedule

	27 Sunday	28 Monday	29 Tuesday	30 Wednesday	31 Thursday	1 Friday
Lab Hours	24 hours	24 hours	24 hours	12 am - 5 pm		
10:00 am						
11:00 am					Contest, Round II (if needed) (26-100)	
Noon						
1:00 pm						
2:00 pm		Contest, Round I: Qualifying Round (26-100)				Lab Cleanup (38-600)
3:00 pm						
4:00 pm						
5:00 pm				Impounding (38-600)		
6:00 pm					Contest, Round III: Final Round (26-100)	
7:00 pm						
8:00 pm						
9:00 pm						
10:00 pm						
11:00 pm						
Midnight						

Figure 1.5: Fourth week schedule

Chapter 2

Chicken

We bring you now to the not-too-distant future. It is a future where the transformation of our society from agrarian to a modern industrial utopia has been fully achieved. Cities are massive, layered constructs, the pinnacle of efficiency and space usage. Their inhabitants must, however, be fed.

Plains states speckled with homely farmhouses have been replaced by vast rectilinear tracts of hydroponic grain, huge vats of nutrient slurry, millions of cloned cows that moo in unison, and acellular sheep composed entirely of wool fibers. Science, however, has not completely conquered the farm. One lone, proud animal remains.

The chicken.

Too fickle to conform to breeding programs, and too specialized to succumb to robotic replacement, the world's chickens still produce natural eggs, such that the omelette has become a symbol of The Way Things Used To Be. *The world's chickens all reside in a massive complex of pens in central Nebraska, fully automated.*

Unfortunately, a recent massive accident resulting from a programming bug has released untold millions of chickens from their pens. They peck and cluck at random, depositing their eggs in a scattered manner, rendering useless the automatic and immobile collecting machines.

It is up to you to produce the world's first generation of Cybernetic Chickens, capable of autonomously recognizing, handling, and depositing these eggs in the collecting machines before they go bad. You must realize that if these eggs go bad, they could unleash a cloud of noxious vapors worse than any chemical weapon.

Time is of the essence. Godspeed!

2.1 The Table

The layout of the contest table is shown in Figures 2.1 and 2.2. All measurements are guaranteed to be accurate to within $\frac{1}{4}''$ of the actual dimensions, though the only official measurements are those of the actual tables. The tables may also have seams, where sections of the table physically meet. Make sure your robot is capable of facing the imperfections of the board.

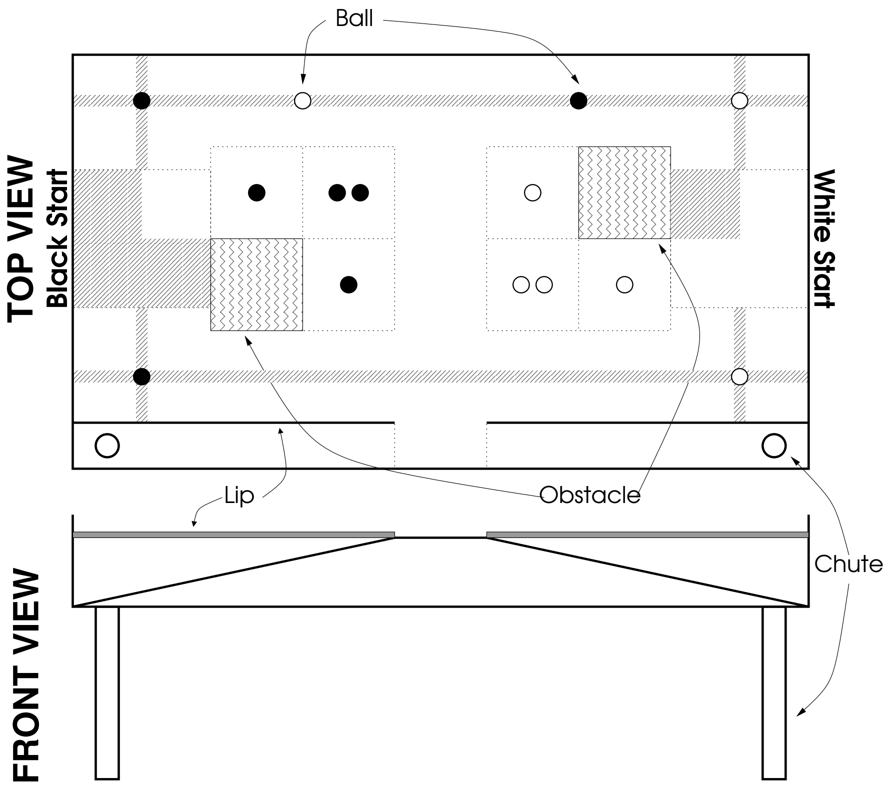

Figure 2.1: The 2002 playing field

The $8' \times 4\frac{1}{2}'$ surface base is surrounded by a $2''$ wall on all sides. The main playing field $8'$ long and $4'$ wide is surrounded by three of these walls. A $\frac{1}{2}''$-tall lip borders the fourth and front side. A $1'$ gap in the lip is located in the center of that side of the table. The gap leads to a trough, which peaks at the center, and slopes downward 9.3° toward the ends of the table, where two vertical $4''$-diameter chutes representing the collecting areas are located. A series of $2''$-wide black lines lead to and from various significant locations on the table, and can be used for navigation.

Fourteen balls are on the table, seven for each robot, as shown in Figure 2.1. The balls may be pushed around the table or into the troughs by the robots.

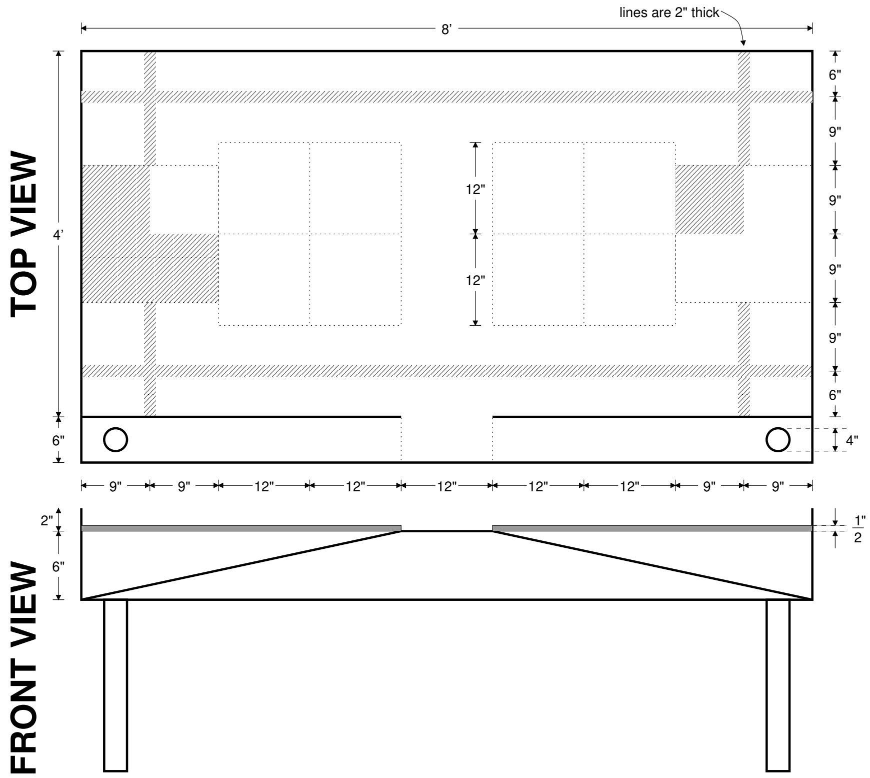

Figure 2.2: Measurements of the playing field

On each side of the table, there are four locations in which a 1′ flat square platform 2″ high may be placed. Contestants will choose the obstacle position for their opponents. Placement of the obstacle also determines ball placement, as shown in Figure 2.3.

The robots begin on either side of the board in an 18″ × 18″ square area with a start light in the center. The patterns on the starting areas assist the robot in determining which side it is on.

The balls located at various points on the table are field hockey balls, approximately 3″ in diameter.

2.2 Scoring

The score that each robot receives is determined by the final state of the contest table after the match has been played. The points are determined by the ball position in each tube. The lowest ball in a tube will be worth one point, the second lowest will be worth 2 points, the third lowest 3 points, and so on. A sample scoring is shown in Figure 2.4. The winner of the match is the one who received the most points.

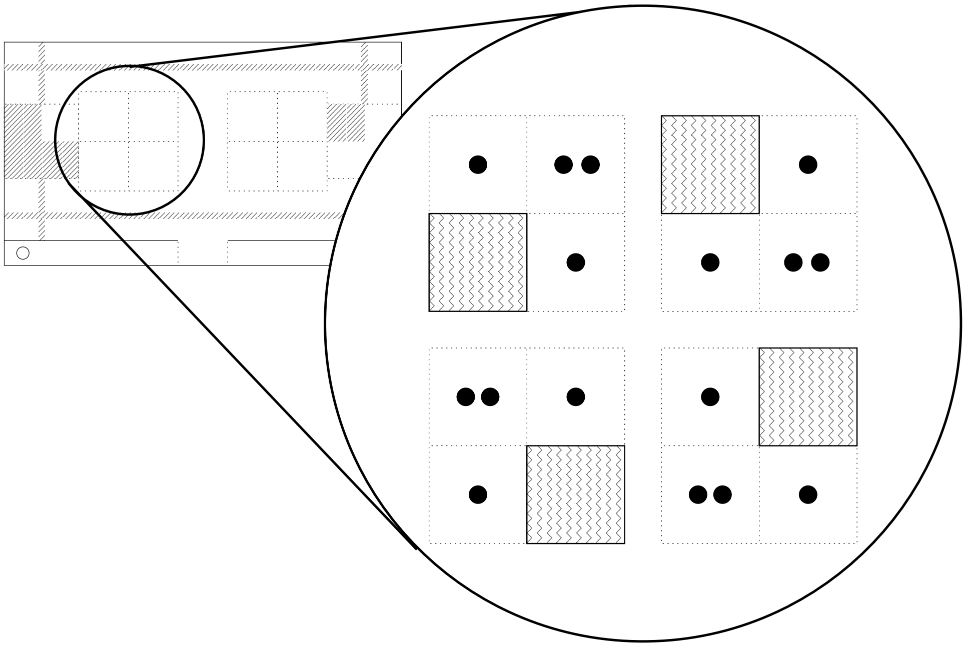

Figure 2.3: Possible Configurations with Obstacle

Balls left on the troughs or the playing field do not count. Any balls that overflow from the top of the tubes that are not in the tubes do not count. Each tube can hold up to 8 balls.

In the case that there is a tie, robots which at any point did something during the match that directly improved their score instead of only preventing the opposing robot from scoring receive a win, and those that did not receive a loss. Thus, in a tied situation, a double-win, double-loss, and a single win are all possible outcomes.

2.3 The Competition

The competition will be a double elimination tournament held over the course of two days. Robots compete head to head in successive rounds until they lose twice. When all but one robot has been eliminated, that robot will be crowned champion.

- **Contest, Qualifying Round.** The first round of the contest will also serve as a qualification round. If a robot demonstrates the ability to score points, regardless of whether it wins or loses, it will be allowed to proceed to the next round. If it does not, modifications may be made, and it will have two more chances in lab to run against an inert placebo. If it cannot score points against the placebo after two

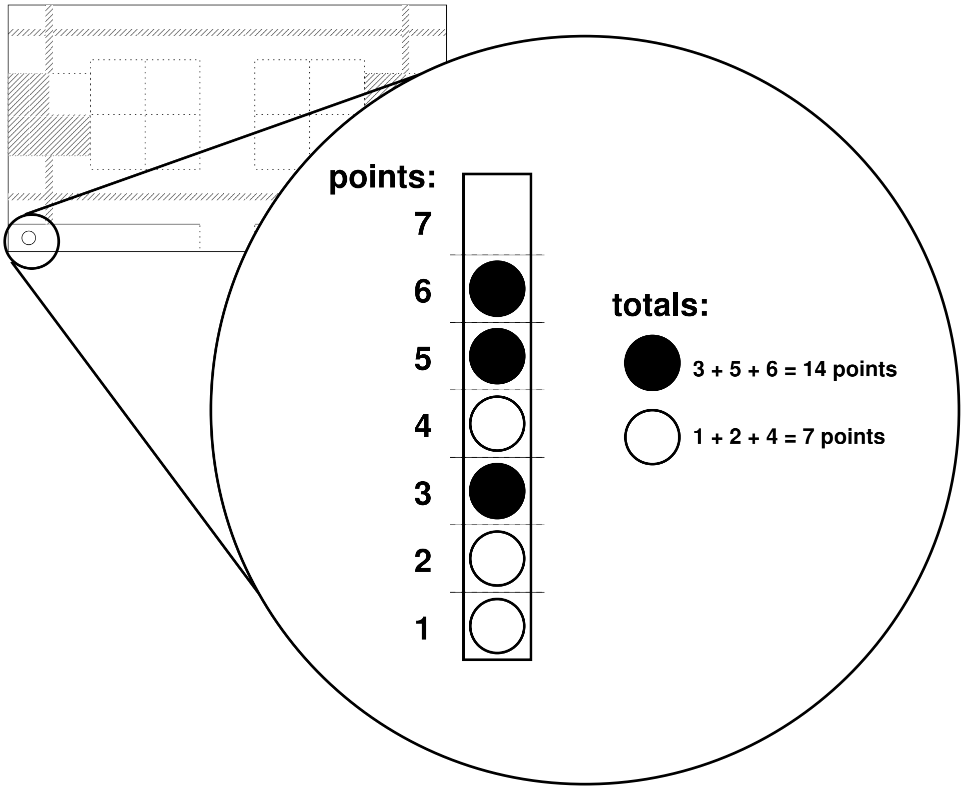

Figure 2.4: Calculating Points

tries, it will not qualify for the rest of the contest. Losses to opponents during this round do count towards a robot's elimination.

- **Contest, Second Round.** Only qualifying robots may compete in the second round. If a robot which has lost in round one also loses in this round, it will be eliminated from the competition. This round may be skipped if few enough robots make it past round one.

- **Contest, Final Round.** This is the main competition that everyone comes to see. Robots will compete until all but the winner has been eliminated. At the discretion of the Organizers, the final round of competition may be conducted in a round-robin format, ignoring previous losses.

2.4 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 about the legality of your robot, feel free to ask the Rules Committee for an official ruling before your robot is impounded.

2.4.1 Period of Play

1. Contestants will choose the obstacle placement for their opponent. A judge will place the block in the appropriate area and re-arrange the balls as necessary.

2. The judges will choose a random starting orientation from the four cardinal directions. The contestants will have 60 seconds to set up their robot, during which time the starting light will be blinking at 20 Hz.

3. After the minute has elapsed, the judge will turn off the lights. The contestants arm their robot (probably by pressing a button), and must step away from the table.

4. Robots may not supply power to their actuators at this point. If a robot does, it has false-started. If a robot false-starts twice, it forfeits that match and is replaced by a placebo.

5. The starting lights signal the beginning of the match by blinking on and off at 20 Hz for one second. The robots must turn on their IR beacons, and have 60 seconds to score points. Software to detect the signal will be provided.

6. The IR beacon must be on during the 60 seconds of the match. It must, however, be turned off at all other times. Any robot that fails to do this will be disqualified.

7. 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 his machine to forfeit. All robots must be controlled solely by their onboard computer.

8. At the end of 60 seconds, the robot must turn off electrical power to its actuators. In addition, the robot's IR beacon must be off, so as not to interfere with another match. Any robot that fails to do either will be disqualified. Software is provided to do this.

9. The match ends when all robots and game objects on the table come to a rest.

10. The robot that scores the most points will be the winner. In the event of a tie, the judges will award wins to robots that have changed the score in their favor, and losses to those that have not.

2.4.2 Kits

1. All kits contain the same set of components, although some parts may be colored differently in different kits.

2. Some parts in the kit are considered tools and may not be used on the robot.

3. Robots must be built only from the parts in the kit, except when explicitly allowed by other rules.

4. Teams may trade only functionally identical components. This includes trading identical LEGO parts of different colors and replacing broken components.

2.4.3 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. Only the IR broadcast beacon and sensors may extend higher. 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, their opponent's microprocessor board or infrared beacon.

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.

2.4.4 LEGO

1. Only LEGO parts may be used as robot structure.

2. A robot's structure may not be altered once the contest has begun, although 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.
   - 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.

4. LEGO pieces may not be joined by adhesive.

5. Lubricants of any kind are not permitted.

6. One layer of rubber band or tape may be applied to LEGO wheels and treads to alter the coefficient of friction. See Section 2.4.6 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 or LEGO. Double-check with an Organizer before using another material.

2.4.5 Software

1. A robot's program cannot be altered once the contest has begun.

2. In the event of a memory failure, a new copy of the robot's program may be downloaded between rounds. A computer will be available at the contest for this purpose.

3. A robot may not be told its orientation or given information about its opponent. It may only deduce this information after the match has begun.

2.4.6 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. The wooden dowel may be used only as a tower to mount the infrared transmitter and sensors.
7. The lead-acid batteries may only be used in the standard electrical configuration of three cells connected in series.
8. String may be used to convey force between moving parts (i.e. pulley systems) but may not be used for structural support.
9. Only thin rubber bands that are supplied by the organizers may be used. Only #16 and #32 rubber bands may be used.
10. Rubber bands may only be used to store energy and affect the motion of moving parts. Rubber bands cannot be used for structural support.
11. Each rubber band must be touching at least one lego piece, and lego pieces connected by a single rubber band or a chain of two rubber bands must move relative to each other.
12. Rubber bands and string may be used to stuff tires for stiffening purposes.
13. Rubber bands cannot be used for nets. Valid uses include damping external forces and storing energy. If you have an idea that uses rubber bands, be sure to run it by an organizer before implementing it.
14. Rubber bands may be used to add strength to tread.
15. Extraneous components may not be added to a robot for the purpose of adding weight.

2.4.7 Infrared Beacon

1. All entries must be transmitting modulated infrared (IR) light using the supplied IR beacon and software for the entire 60 seconds of the match, but at no other times.

2. Robots failing to meet the IR transmission specification, or in any way modifying or jamming their transmission during the round of play will be disqualified.

3. The beacon must be located between 17″ and 18″ above the surface of the playing field when mounted on the robot. The position of the beacon relative to the microprocessor board may not change during the round.

4. The beacon may not be intentionally obstructed. Because of this, robots may not extend above 16″ or lift objects above this height.

2.4.8 Placebos

Placebos are staff-designed “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 concerning the transmission of the IR beacon.

2.5 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 25 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 25 dollars.

2.5.1 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.

Several rules must be noted regarding trading of sensors:

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.

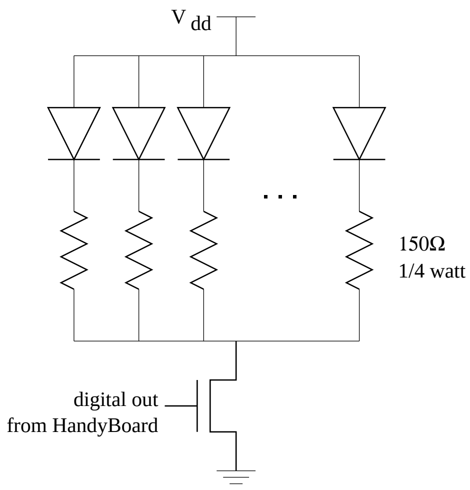

Figure 2.5: Power FET circuitry with LEDs

3. A broken sensor may be traded in for a new sensor of the same type at a cost of 5 dollar 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 25 dollar nor the 20 sensor point allotment.

2.5.2 25 Dollar Electronics Rule

A team may spend up to 25 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. Except for those given in the kit, batteries, motor driver circuitry, and microprocessors of any kind may not be used. Motor driver circuitry includes motor driver chips, relays, and power transistors. The only exception to this are power FETs that are supplied by the staff, which can be used to drive anything but actuators (the FETs may drive such things as LEDs and speakers). See Figure 2.5 for the circuit. Talk to a staff member if you are interested in doing this.

2. Resistors rated less than 1 watt and capacitors valued less than 100 μF may be used freely, without counting towards the 25 dollar total.

3. Contestants who add any non-kit parts to their robot must turn in a design report which includes a description of the modification, a schematic of all added circuitry,

and store receipts for the parts purchased. This design report must be turned in with the robot at impounding. Any robot found with added circuitry that has not be documented in this fashion will be disqualified.

4. If a contestant 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 and be included in the design report described above.

5. A team may purchase extra servos for 10 dollars.

6. To replace a broken 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 25 dollar allotment.

Chapter 3

The Human Factor

Participating in a challenging activity can be either a rewarding or stressful experience. Whether it is the former or the latter, however, depends entirely on you. In 6.270, you will be faced with the challenge of building a functional robot in a short period of time, which is by no means an easy task. Accomplishing this will not only require technical expertise, but also the ability to motivate yourself and to contribute as a member of a team.

Since each person is different and has his own unique set of skills to offer, there is no one correct way to approach the course. This chapter, therefore, is meant to present some suggestions for dealing with the human aspects of the course. Whether you take this advice or develop your own approach is entirely up to you.

3.1 Survival Tips

When working on a large project, many human factors come into play. In order to effectively contribute, you must not only have the knowledge, but also the desire and ability to apply it. Remaining motivated for the duration of the task can be difficult, and participants often find themselves feeling burnt out and stressed. This stress results in fatigue, irritability, and poor performance which in turn leads to more problems and more stress. If you keep the following tips in mind, however, you will be able to minimize your stress and stay motivated:

- **Have fun.** The best way to remain motivated is to simply enjoy the experience and have fun. Beware of falling into the trap of thinking that your robot has to be the best. This course is not about winning or scoring a lot of points; It is about having fun and learning something in the process. If you simply keep a positive attitude and take the time to enjoy the course, you will find it to be a very rewarding experience.

- **Take care of yourself.** While skipping a few meals or pulling an all-nighter might seem like a good way to get some extra work done, in the long run, it tends to be counterproductive. Neglecting your body's needs will inevitably leave you tired and drained, making you much less productive and increasing your chances of catching an illness. If you eat and sleep on a regular schedule, you will find that you are healtier and more motivated.

- **Start early.** Building a robot takes longer than you expect, even when you take that fact into account. By starting early and following a reasonable schedule, you will allow yourself the time to get things done without the stress of working at the last minute. If you plan well, you can spend the last few days goofing around with your robot and making those little last minute adjustments instead of pulling all-nighters just trying to make the robot work.

- **Share your ideas.** Many people think that by keeping the design of their robot a secret they will gain a competitive advantage; however, this is usually not the case. When you are willing to share your ideas with others, others will be willing to share their ideas with you. Quite often, another team will be able to suggest an idea that you have missed or a solution that you have been unable to find.

- **Take a break.** If you find yourself arguing with your teammates or becoming frustrated over a problem, take a break and do something else. Getting away from the robot and your teammates for awhile will help you relax and allow you to collect your thoughts. The world has many experiences to offer and exploring some of them might be just what you need.

3.2 Teamwork

One of the most essential parts of any large project is teamwork. A person working alone will not have the time to learn and do everything necesssary to accomplish the task. A team, on the other hand, can draw upon the talents and manpower of all of its members, making it much more productive than an individual.

3.2.1 Planning

Before a team begins work on a problem, it has to develop a plan. Rushing ahead is likely to cause work to be duplicated or important tasks to be missed. Worse still, failure to plan ahead can lead to incompatibilities in parts that are supposed to fit together. When discovered too late, these errors can prove fatal to the project.

A good place to begin planning is to decide what the team is interested in accomplishing. Some teams are focused on winning while others just want to have a little fun. Still others are interested in the learning process and would prefer to spend more time on the parts that are most educational. It does not matter what goals a team sets for itself as long as all the members understand and agree with the overall vision. This will help coordinate the efforts of all the team members and provide direction for the project.

3.2.2 Brainstorming

Teams often employ the technique of brainstorming for generating potential solutions to a problem. During such a session, participants think aloud, suggesting ideas as quickly as they can think of them. Other members of the team can then use those thoughts to create new ideas of their own which they throw back to the group. When it works well, a team can combine the knowledge and creativity of all its members to generate solutions that an individual would not even consider. The following guidelines will help make a brainstorming session as effective as possible:

1. No squashing. Negative comments have no place in brainstorming. Insulting another person's ideas will cause them to be reluctant to offer further suggestions.

2. Don't hold back. The process only works if everyone shares their thoughts. Even the silliest idea can often inspire a great one.

3. Stay on topic. During the course of discussion, it is easy to wander off on a tangent. Focus on the problem at hand and avoid distractions.

4. Relax. Ideas flow more freely in a relaxed environment. Find a quiet, comfortable place where the team can concentrate on the task at hand.

3.2.3 Constructive Conflict

Teams composed of members who always agree with each other work quickly and efficiently but never produce the best solution. Instead, the teammates who disagree often are the ones that build the strongest teams. This may seem counterintuitive at first, but it turns out that conflict, if handled correctly, can be one of a team's greatest strengths.

Shouting at each other and throwing tantrums will certainly not accomplish anything, but calm, rational debate allows the team to view a topic from multiple perspectives. This not only allows the team to consider various possible solutions, but it also forces the issue to be examined in greater depth. Often, you will find that an idea that seems good at first will not hold up under the scrutiny of another teammate. Disagreements between teammates force the team to constantly reevaluate and improve the design and may even help generate new ideas.

In order to engage in rational debate, you must walk the fine line between strongly defending your position and being open-minded enough to consider other ideas. Debate is not about being right or winning the argument; It is about examining both sides of an issue, so that the team can choose the best course of action. It is very easy, during an intense debate, to forget that you are supposed to be participating in a productive task, so it is helpful to keep the following guidelines in mind:

1. Prepare a strong position and present it forcefully, but keep an open mind.

2. Allow others a chance to speak and listen attentively while they do.

3. Try to view the problem from multiple viewpoints, including the opposing one.

4. Do not take disagreement and rejection personally.

3.2.4 Friends and Enemies

Forming good relationships with your teammates is one of the primary lessons of 6.270. In past years, the ability to work well together has often been the most critical factor in a team's success or failure. Participants whose robots do not perform well often attribute their failure to poor team dynamics and arguments between teammates. Contest winners, on the other hand, usually attribute their success to their enjoyment of the course and the fun they had working together with their friends.

The relationships you form with your teammates are likely to continue long after the course is over. In the past, teams formed by complete strangers have left as very good friends, and unfortunately, good friends have left the course no longer speaking with one another. Remember that your teammates are human, and your actions affect not only the project, but also the people you are working with. Putting in the extra effort to work well with your teammates will pay off both in the contest and for a long time afterwards.

3.3 Implementation

Building a robot is usually more work than an individual can handle on his own, so it is necessary to work as part of a team. Everyone should help out by providing part of the labor necessary to design and implement the robot, but in order to do this, the work needs to be divided up in some fashion.

3.3.1 Division of Labor

Each person brings a different set of strengths to the team, so many teams opt to divide the work into a number of subtasks, each of which becomes the responsibility of an individual team member. In this *specialist* approach, each person works on one area of the project and becomes an expert at it. The most common division in 6.270 is into hardware, software, and LEGO construction, but as long as the work is divided along clearly defined abstraction barriers, the communication needed to organize the team is small. This tends to be very efficient, especially for teams whose members come into the course with varying backgrounds and interests, though it tends to lead to a very narrow learning experience for the individual.

Another popular division of labor is the *generalist* approach, where every member of the team shares equally in all aspects of the implementation. This allows each individual to have a say in every part of the design and to gain an overall understanding of the process. It also requires that the teammates work in close proximity which can lead to a more fun and relaxed experience. Because of the amount of coordination needed between teammates, though, a great deal of time will have to be spent on organization and communication. This makes the implemenation less efficient, but can often lead to a better learning experience.

3.3.2 Debugging

Debugging can be a long and tedious process, so it is important to follow good design practice to minimize the number of bugs you will have to fix. Regardless of how careful you are, though, mistakes are inevitable and debugging will be necessary. As a general rule, it will take longer than you think to debug, so it is always better to allocate too much time for debugging than too little.

Occasionally, you will run into a bug that just seems to elude you. In these cases, instead of banging your head on the wall, you should. Have a teammate review your work and search for the bug. It may be that you are using a bad assumption or that you are continually missing the same mistake. When this happens, a teammate can bring a fresh perspective to the problem which might yield the answer.

Some teams take this debugging philosophy even further. No person on the team ever debugs his own work. Instead, each person gives their work to another teammate and that person debugs it. This way, each part of the project benefits from the input of at least two people.

3.4 Contest Tips

Everything always goes wrong at the worst possible time, which in 6.270, is contest night. There is nothing more heartbreaking than having your robot not work because of some small oversight. To help minimize the chances of such unfortunate occurrences, follow the tips below when preparing for the competition:

1. When making practice runs with your robot, try to avoid helping it. During actual competition, you will not be able to touch your robot when it does something wrong.
2. Practice your calibration routine in lab, so you can do it quickly and accurately at the contest. You must be able to complete your routine within a fixed time limit.
3. The lighting conditions at the contest will be different from those in the lab. Make sure that your light sensors are well-shielded and can be calibrated to work under different conditions.
4. Be aware of how your proximity will affect the calibration of your robot. When you lean over your robot, you can cast shadows or cause reflections which could affect the measurements of your sensors.
5. Develop a checklist for preparing your robot to compete. Between rounds you should examine your robot and repair anything that has broken.
6. Bring a repair kit to the contest. This should include a fresh set of batteries and a replacement for any part that tends to wear down or break during operation.
7. Have fun.

Chapter 4

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.

4.1 Soldering

Soldering is a method of creating electrically conductive connections between electronic components. A special type of metal, called *solder*, is melted onto the joint and allowed to harden. This forms a bond between the components which joins them both structurally and electrically. A soldering iron is extremely valuable for constructing electronic circuits, but as with any tool, you must begin by mastering the skills necessary to use it.

Aside from the sensors and actuators, your team will be required to solder the beacon, the expansion board for the Handy Board, and the battery recharger for the expansion board. It is crucial that you solder the circuitry cleanly and correctly.

4.1.1 Safety

Soldering is not a dangerous activity, but if you do not respect the soldering iron, it can does have the potential to cause harm. While you work, it is important to observe the following safety rules:

1. Keep the soldering iron tip away from everything except the point to be soldered. The iron is *hot* and can easily damage parts, cause burns, or even start a fire.

2. Keep the soldering iron in its holder when not in use. Never wave the soldering iron around or hand it to another person. If someone else wants the iron, place it in its stand and let him pick it up from there.

3. Never assume that a soldering iron is cold. Always check the iron before you pick it up.

4. Do not touch a joint immediately after soldering it. It takes a moment for the solder to cool back down.

4.1.2 Technique

Before you begin any work with the soldering iron, you should assemble all of the tools that you will be using. The ones that you will require include a soldering iron, stand, solder, and a damp sponge. You may also wish to have a set of helping hands, cutters, and wire strippers available if needed for the task.

Once you heat up the iron, the first thing to do is *tin* it. Wipe the tip on a damp sponge to clean it and then immediately melt some fresh solder onto it. This gives the tip a protective coating and also helps improve heat transfer. You should tin the tip again each time you use the iron or when it has been sitting idle for awhile. A properly tinned iron should have a shiny silver tip, so if it ever becomes dull or dirty, it needs to be tinned again.

With a properly tinned iron, you are ready to solder. Firmly secure the parts to be soldered with a set of helping hands. Heat the two surfaces by inserting the tip of the iron into the point where they touch each other. The solder should be applied to the joint, *not* to the iron directly. This way, the solder is melted by the joint, and both metal surfaces are heated to the temperature necessary to bond chemically with the solder. When done properly, the solder will be drawn into the joint to fill the connection. Figure 4.1 shows the results of good and bad soldering technique. The following pointers can be useful in honing your skills:

1. If the solder does not melt easily into the joint, applying a small amount of extra solder to the iron will improve heat transfer.

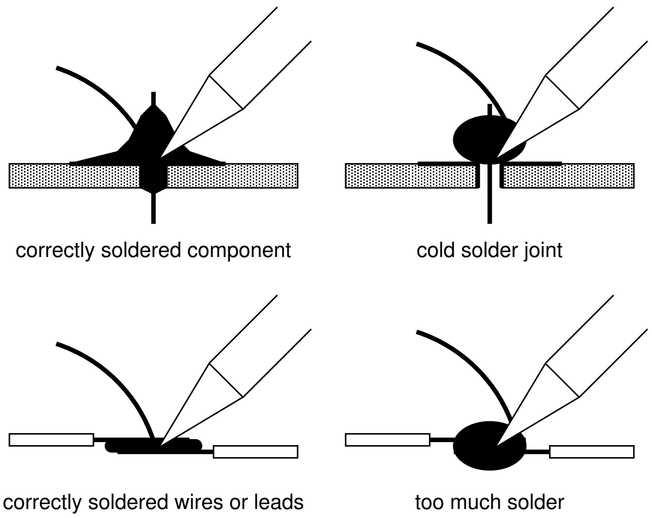

Figure 4.1: Good and bad soldering technique

2. If a ball of solder begins to collect on the tip of the iron, use a damp sponge to wipe it off.
3. Do not hold the iron against the joint for more than about 8 seconds. Many electronic components can be damaged by excessive amounts of heat.
4. When working with stranded wire, it helps to tin the end of the wire. This holds the strands together and improves heat transfer. Apply heat with the soldering iron and let the solder flow between the stands.
5. When attaching wires, remove as little insulation as possible to make the joint. Exposing too much wire is likely to cause short circuits.
6. Never use the iron to melt anything but solder. Impurities will damage the iron and cause it to smoke. If the tip becomes dirty, it can be cleaned by melting generous amounts of solder onto it.
7. Be sure to get rid of any extra solder left on the solder tip.

A *cold solder joint* occurs when an air bubble or other impurity has entered the joint during cooling. This is most commonly caused by an attempt to paint the solder onto a joint by applying it to the soldering iron directly. The solder does not flow properly into the joint, causing it to ball up and have a dull appearance. These joints are brittle and make poor electrical connection. To fix such a joint, heat it with the soldering iron until it melts into the joint properly. If the cold solder joint reappears, remove the solder and then try again.

4.1.3 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.

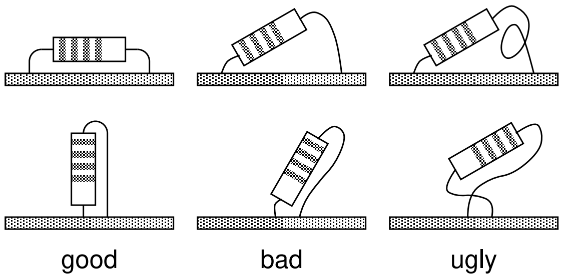

Figure 4.2: Axial component mounting

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

4.1.4 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.

4.2 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.

4.2.1 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 4.3 shows the meaning of the colors. A few examples should demonstrate how to read a resistor:

- *brown, black, red*: 1,000Ω or 1kΩ

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 4.3: Resistor color code

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

4.2.2 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 4.4:

- **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.
- **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.

4.2.3 Capacitors

Capacitors are available in a variety of types and values:

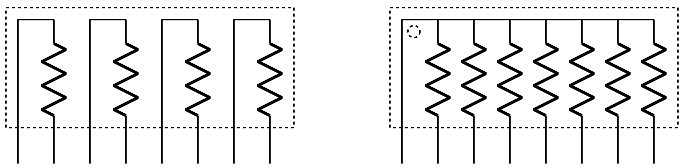

Figure 4.4: Resistor pack internal wiring

- **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 μsymbol acts as a decimal point like $4\mu7$ for a 4.7μF value. Small capacitors (1.0μF or less) have their values printed in picofarads ($1,000,000p$F $= 1\mu$F). 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.

4.2.4 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 4.5 shows how to identify the leads.

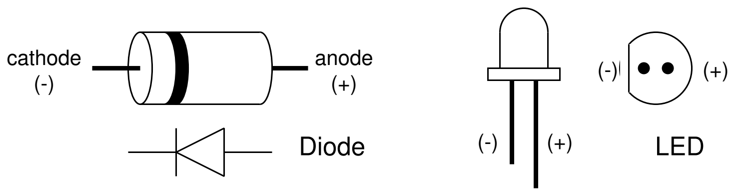

Figure 4.5: Identifying diode leads

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.

4.2.5 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).

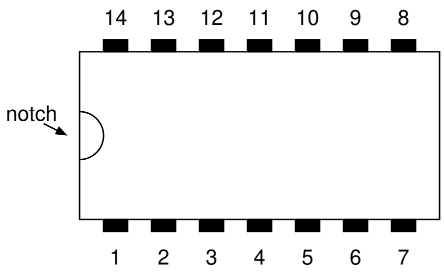

Figure 4.6: Top view of a 14-pin DIP

A marking on the component identifies pin 1 of the component's circuit, as shown in Figure 4.6. 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.

4.3 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.

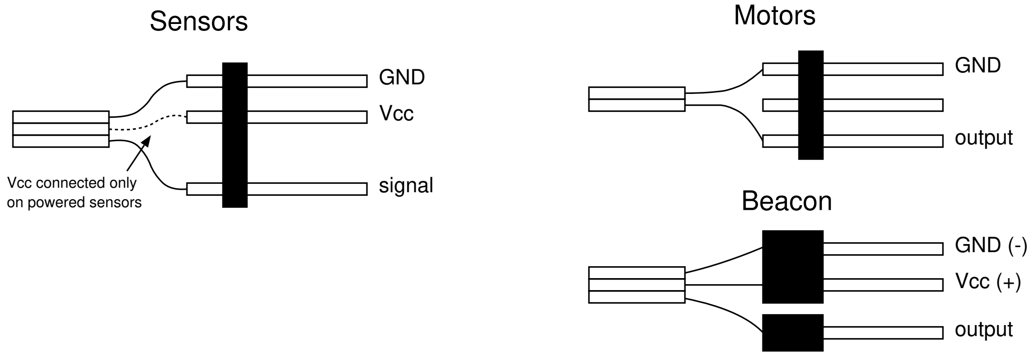

Figure 4.7: 6.270 connector standard

In order to keep connectors from being plugged into the wrong port, different types of components are built with different connectors. Figure 4.7 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. Remove just enough insulation from the ends of the wires to allow them to be soldered.

3. With your fingers, twist the threads of each individual wire end tightly.

4. Slip each wire through a $\frac{1}{4}''$ length of $\frac{1}{16}''$ heat shrink.

5. 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.

6. Solder the wires to the connector.

7. 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.

8. If heat shrink is unavailable, hot glue may instead be used to strengthen and insulate the connection.

   (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.

4.4 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) If the motor comes with a metal gear on its shaft, remove it with a pair of wire strippers. Place the jaws of the strippers between the motor and gear and squeeze. When the strippers close, the bevel in the cutters should pry the gear off.
   (b) Shrink a piece of $\frac{1}{16}''$ heat-shrink tubing onto the motor's shaft. Then shrink two additional $\frac{1}{8}''$ pieces around that. An 8-tooth LEGO gear should now fit snugly over the tubing.
   (c) Push an 8-tooth gear over the head-shrink tubing. Make sure that it goes all the way on and that it is aligned correctly.
   (d) Cut off any excess tubing which sticks out from the end of the gear.

2. Legoize the motor

   (a) Construct the jig shown in Figure 4.8. 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.

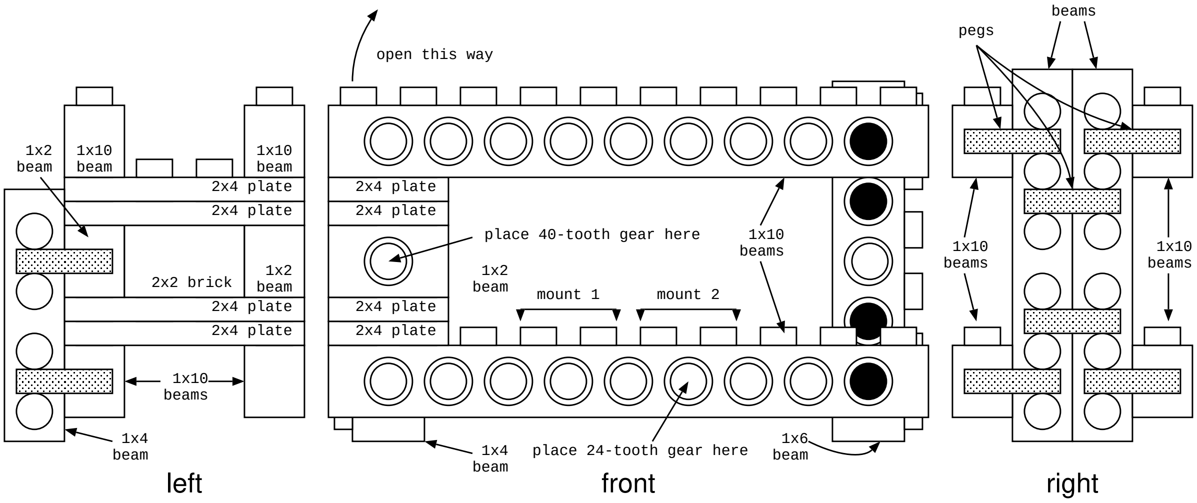

Figure 4.8: Jig for motor assembly

(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.

(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.

4.5 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").

4.6 The Handy Board and Expansion Board

The "brain" of your robot is Fred Martin's Handy Board, a controller based on the Motorola 68HC11 processor. This year, Fred has designed a modified version of the Handy Board's expansion board, which makes it well-suited for use with 6.270. Features of the system (Handy Board + 6.270 Expansion Board) include the following:

1. A dual rechargable battery system—on-board NiCd batteries preserve program memory, off-board Hawker Cells power actuators.

2. Twenty-one analog inputs and nine digital inputs.

3. Three digital outputs, one of which will be used to control the infrared beacon.

4. Six bi-directional motor ports, four with speed control.

5. Six servo ports, one of which will be used to power the infrared beacon. (You can have at most three servos in 6.270, anyway.)

6. An LCD screen for debugging output.

All wiring diagrams found in these course notes should be compatible with the inputs and outputs of the HandyBoard. More information on the features and usage of the Handy Board can be found in *The Handy Board Technical Reference*, included with your course notes. If you have further questions, please talk to a member of the staff!

The controller can be programmed using Interactive C (IC), a language designed specifically for use in robotics. IC programs are compiled into pseudo-machine instructions (pcode), which are interpreted by a pcode interpreter installed on the Handy Board. For information on IC, see the *The Interactive C Manual For the Handy Board*, included with your course notes.

Instructions for assembling the expansion board are in Appendix B.

Chapter 5

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.

5.1 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 5.1. 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.

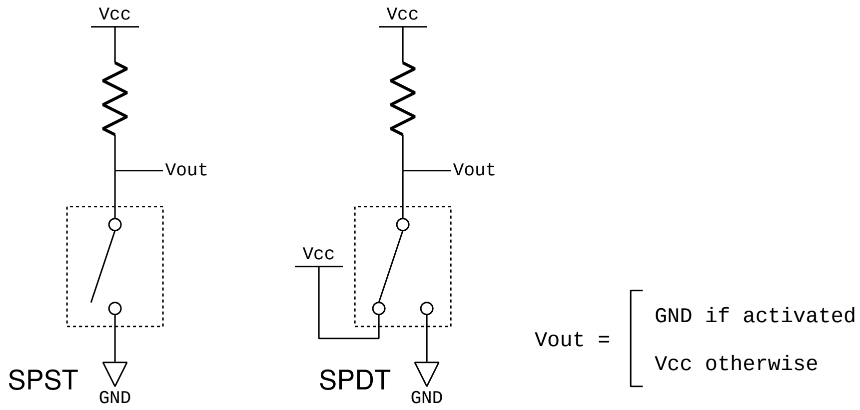

Figure 5.1: Digital Sensor Circuit

5.1.1 Switches and buttons

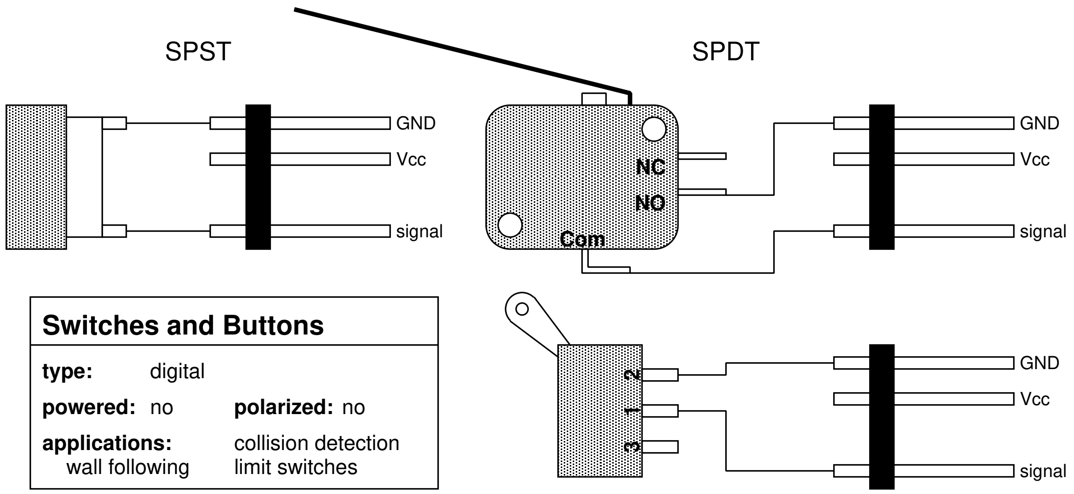

Figure 5.2: Switches and buttons

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 5.2. 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.

5.2 Resistive Analog Sensors

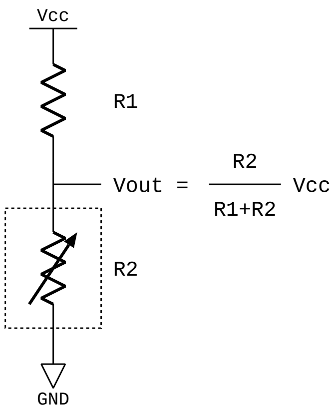

Figure 5.3: Resistive Analog Sensor Circuit

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 5.3. 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.

5.2.1 Potentiometers

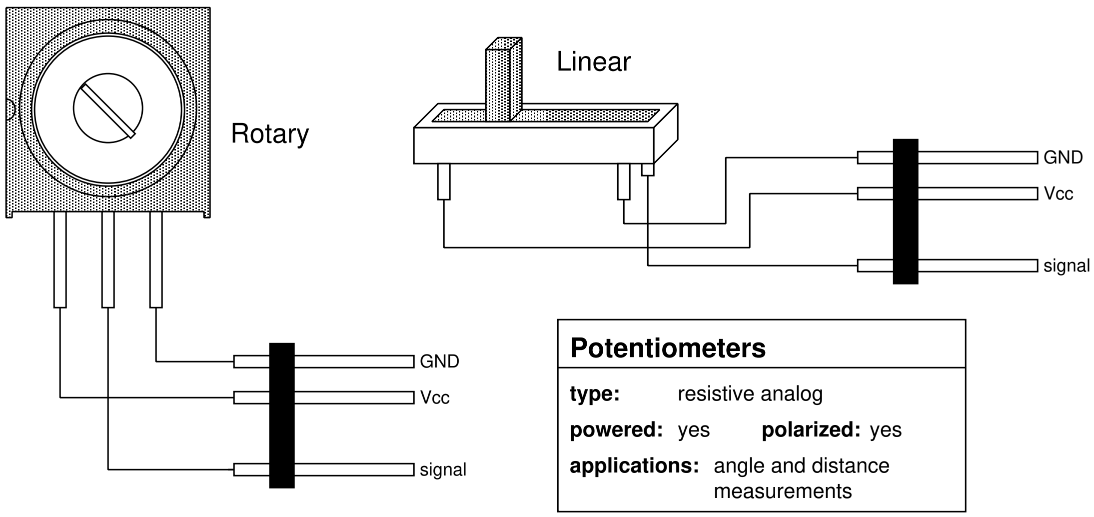

Figure 5.4: Potentiometers

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 5.4, 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 5.4.

5.2.2 CDS Cell

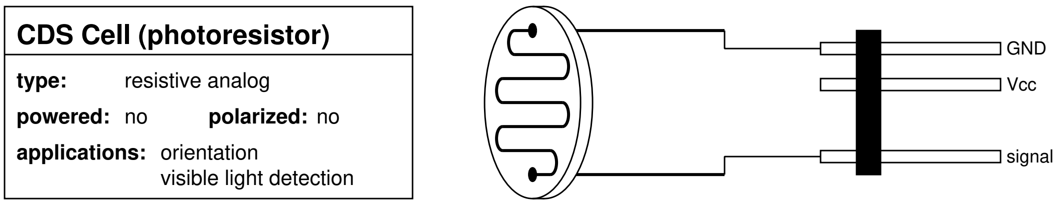

Figure 5.5: CDS Cell

The CDS Cell is a nonpolarized device which responds to visible light. It contains a chemical whose electrical resistance decreases as more light hits it. The output of the sensor is an analog voltage which corresponds to the intensity of the light hitting it. Lower light levels yield higher output values.

The CDS Cell can operate over a large range of light intensities. In absolute darkness, the resistance is around 1MΩ and in direct sunlight, the resistance falls to around 1kΩ. In indoor conditions where your robot will most likely operate, the resistance will vary from around 10kΩ to 100kΩ. The 47kΩ pullup resistors built into the sensor ports are ideally suited to this range and will yield good results with the proper shielding.

Photoresistors are probably the easiest light sensors to build and use. Since they are responsive to white light they can be used for detecting visible light sources external to the robot or measuring ambient lighting conditions. These same properties also mean that they must be well-shielded in order to return usable values.

5.2.3 LED and Photoresistor

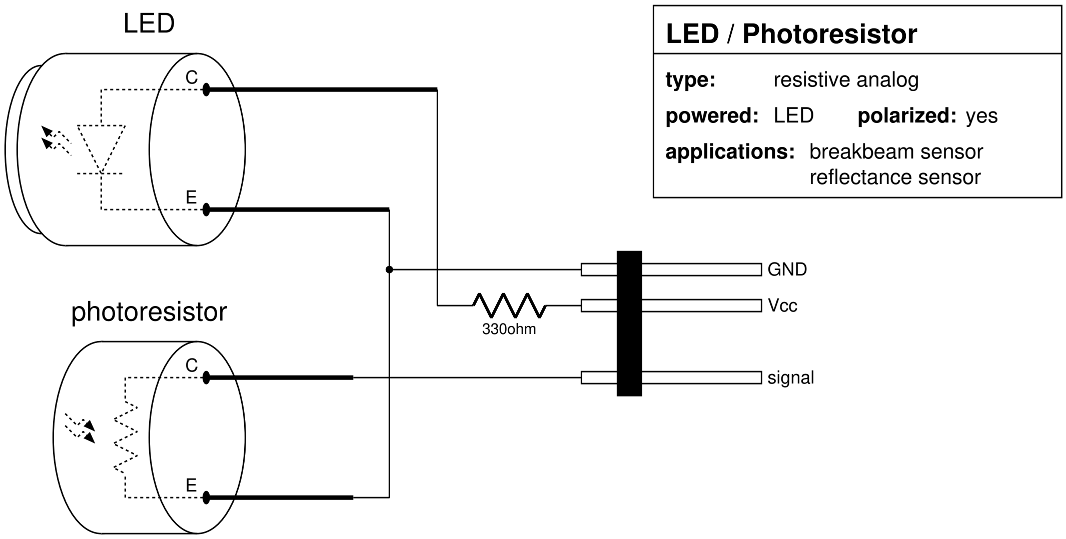

Figure 5.6: LED and Photoresistor

Measuring the color of the table is a common task that needs to be accomplished by the robot. In order to do this, some light sensor and a source of light to illuminate the table are needed. The source may be ambient light that comes from above the table and around the room, but this may not be enough to guarantee consistent readings because the light source is dependent upon a varying table environment.

So, it is better for the robot to have its own light source. An LED, mounted next to a well-shielded photoresistor, can make a spot of light on the table that is significantly brighter than the ambient light. Consequently the brightness of the light will be fairly constant across the table, and discerning colors will be easier.

Be sure to hook the LED to the connector correctly, as shown in Figure 5.6. The longer lead on the LED is the collector and should be connected to power through a 330Ω resistor. The shorter lead goes to ground.

Because the analog sensor ports are powered continuously when the robot is on, the LED will also be on during the entire 60-second match. This is not necessary. In fact, the LED can be plugged into a motor output to conserve on-board battery power. Any number of LEDs can be plugged into a single motor port.

A unique use of this sensor with the LED plugged to the motor port is to measure the color of the table by taking the *difference* of two light measurements, one with the LED on and one with it off. In this case there are two numbers instead of one, and a more reliable reading of the surface color can be expected. By computing the difference of these two values, the approximate amount of LED light that was reflected from the surface is being measured. By comparing the difference to a threshold, the robot can discern between different colors at more than six inches away from the table.

The digital outputs can also be used for light measurements as well, but if you wish to try doing this, be sure to talk to Paul Grayson (`pdg@mit.edu`).

5.3 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 5.7 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 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.

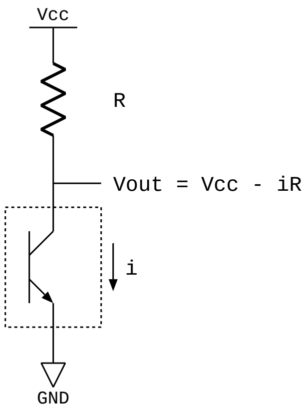

Figure 5.7: Transistive Analog Sensor Circuit

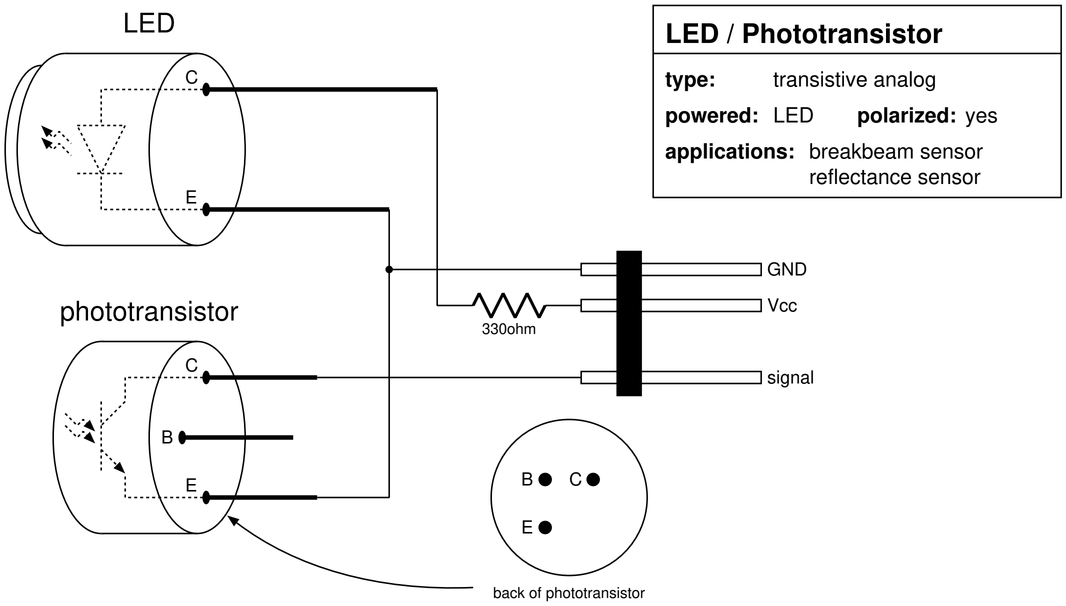

Figure 5.8: LED and Phototransistor

5.3.1 LED and Phototransistor

The phototransistor's output characteristics are perfect for use with the 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. Note that the longer lead on the LED is the collector and should be connected with the 330Ω resistor to power. It also may be easier to solder the phototransistor if the base lead ("B") is either bent or cut off.

Because the phototransistor is so sensitive, it is helpful to wrap some black heat shrink around it. This is especially useful when looking for light in a particular direction.

Once again, realize that the LED does not have to be perpetually on during the 60 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.

This sensor is particularly convenient when working with LEGO because it is 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.

5.3.2 Breakbeam Sensor Package

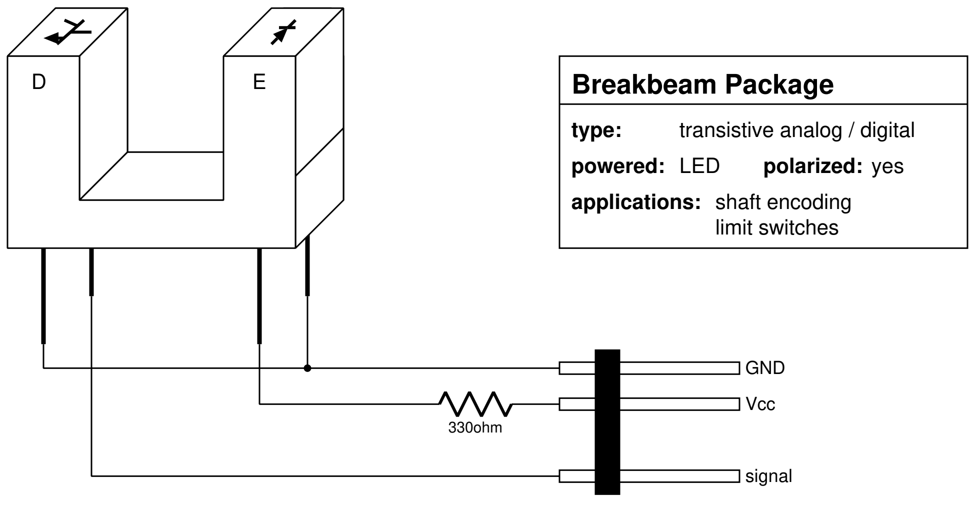

Figure 5.9: Breakbeam sensor package

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 5.9. 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 opague 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.

5.3.3 Sharp Distance Sensor

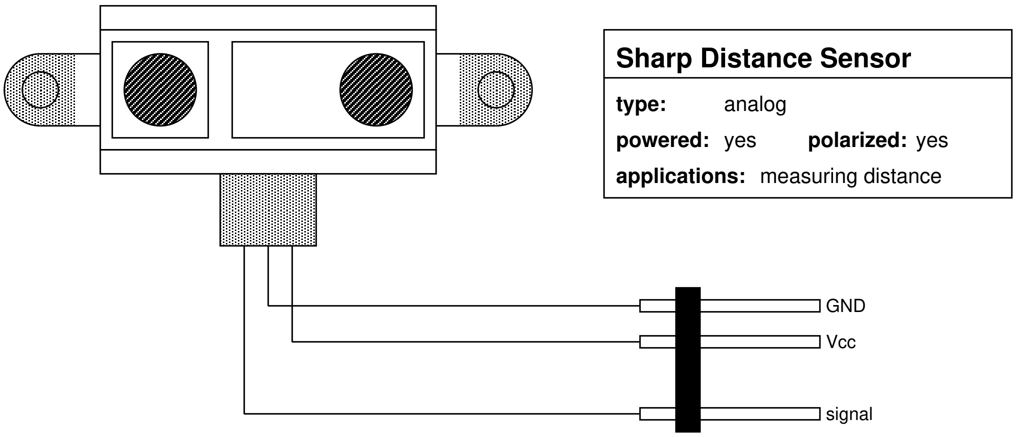

Figure 5.10: Sharp Distance Sensor

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. Besides the infrared beacon, this sensor is the only one which can detect things more than a few inches away—it has a useful range from about $3''$ 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.

Chapter 6

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.

6.1 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.

6.2 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.

6.2.1 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 6.1.

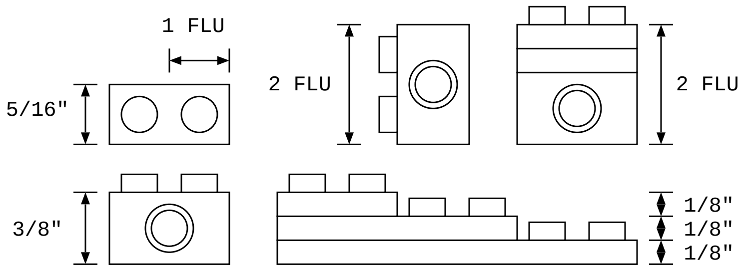

Figure 6.1: LEGO Dimensions

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.

6.2.2 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.

6.3 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 6.2. The concept is simple but important for building an effective robot.

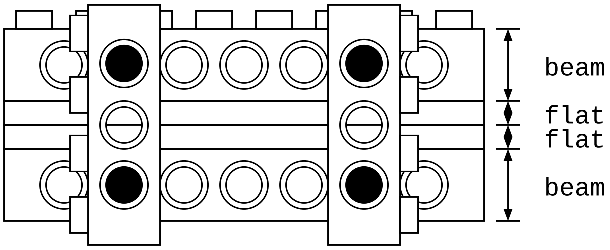

Figure 6.2: A Simple Braced Structure

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).

6.3.1 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.

6.4 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 shown in 6.3. These are the most efficient and easiest to use of the bunch because their

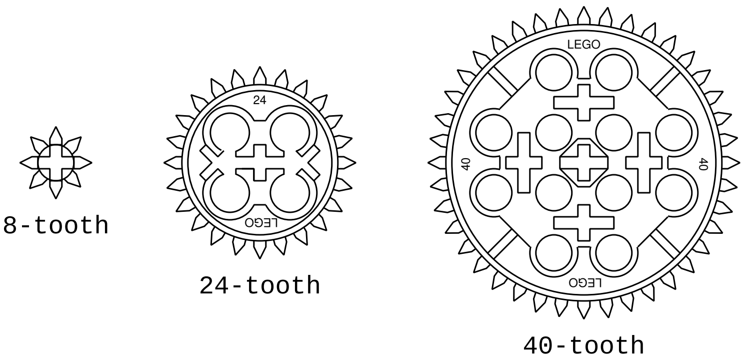

Figure 6.3: LEGO Gears

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.

6.4.1 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 6.4, 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.

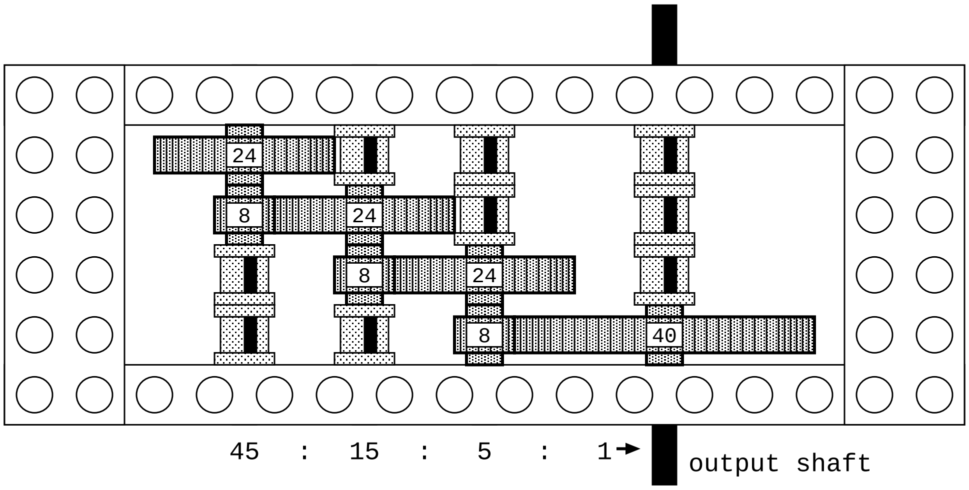

Figure 6.4: A LEGO Gearbox

6.4.2 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.

6.4.3 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.

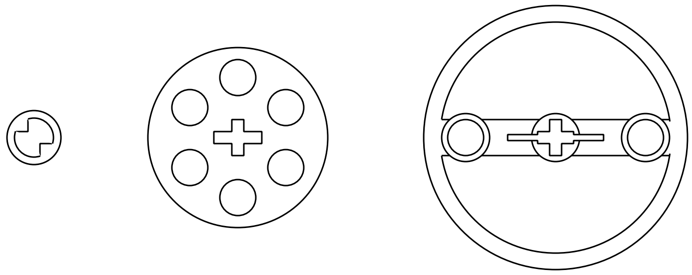

Figure 6.5: LEGO Pulleys

Pulley systems work in a similar manner and can be built using the pulley wheels shown in Figure 6.5 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.

6.4.4 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 awhile, 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.

6.5 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 6.6 shows the three most popular drive arrangements.

6.5.1 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

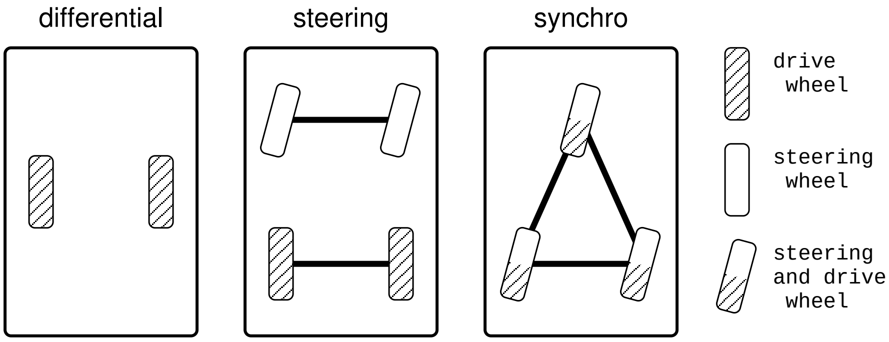

Figure 6.6: Popular Drive Arrangements

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.

6.5.2 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.

6.5.3 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 instructinos 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.

6.5.4 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 7

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.

7.1 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.

7.1.1 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 7.1.

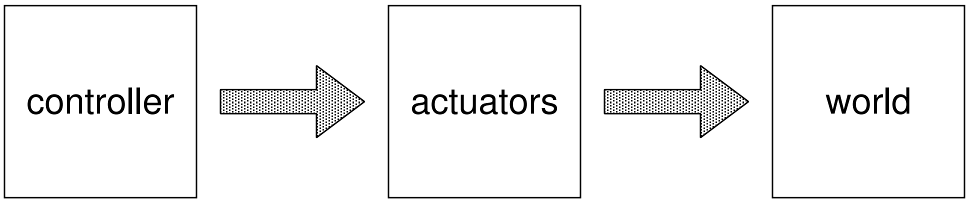

Figure 7.1: Open loop information flow

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 awhile, error begins to build up. Even the smallest errors will eventually build up to a point that the robot becomes lost.

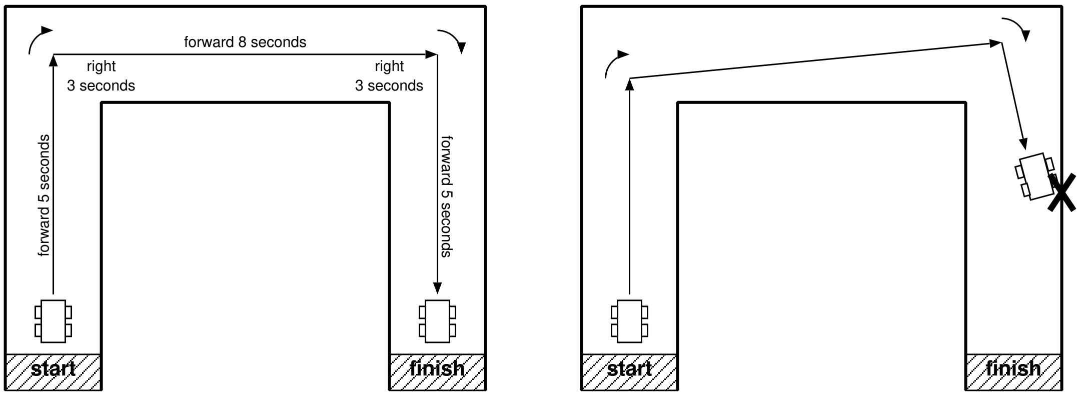

Figure 7.2: A robot trying to navigate with open loop control

Consider, for example, the path the robot must take to navigate the course in Figure 7.2. 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.

7.1.2 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 7.3.

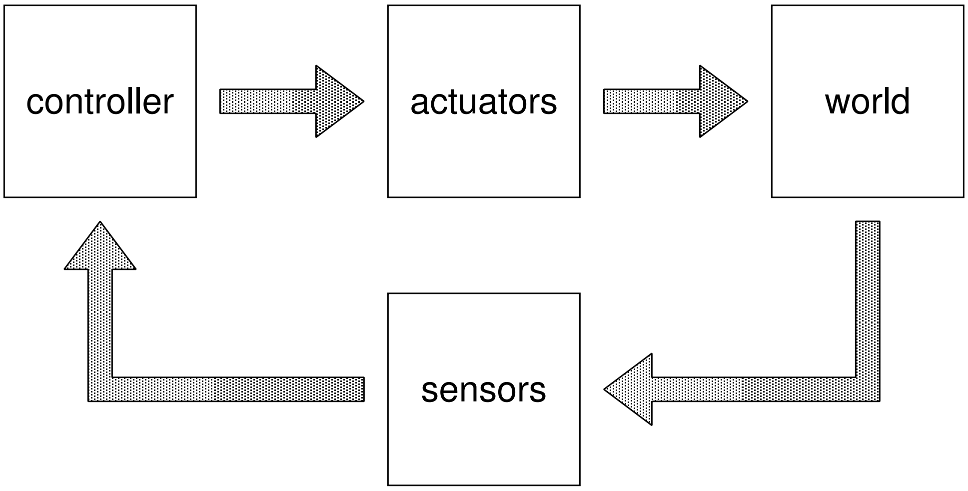

Figure 7.3: Closed loop (feedback) information flow

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 7.4 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.
4. Otherwise, drive forward.

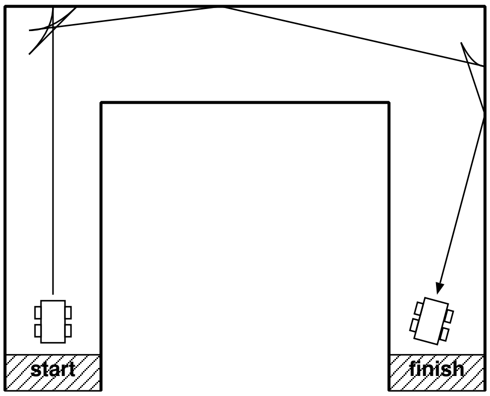

Figure 7.4: A robot using feedback to navigate

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.

7.1.3 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 7.5. 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 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

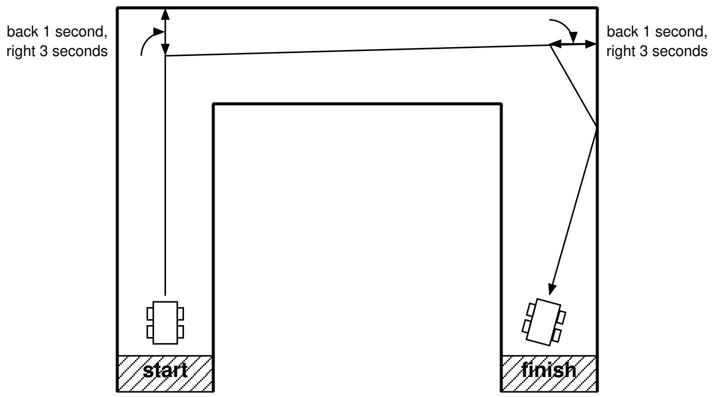

Figure 7.5: A robot combining open loop and feedback control

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.

7.2 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.

7.2.1 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.

7.2.2 Bouncing Switches

Most mechanical switches are afflicted with a phenomenon called *bouncing*. Wires are not ideal and instantaneous conductors, although they are often modelled that way. Instead, when a circuit is broken, the electricity continues to flow for a few microseconds. Since it has no place to go when it reaches the break, it bounces back and forth along the wire a few times before it settles down. This causes voltage levels to rise and fall, causing the sensor reading to transition a number of times (often dozens). A similar effect also causes the switch to bounce when the circuit is reconnected.

In most circumstances, this does not cause a problem, but it can come into play for operations like counting the number of times a button is pressed. If sampling is performed too quickly, the computer can measure multiple counts when the button is pressed just once. This problem can be solved in software by ignoring other transitions that occur for a small amount of time after the first transition is detected. It can also be solved in hardware by the addition of appropriate circuitry, but usually the software fix will be sufficient.

7.2.3 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.

7.3 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.

7.3.1 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 7.6, 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.

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.

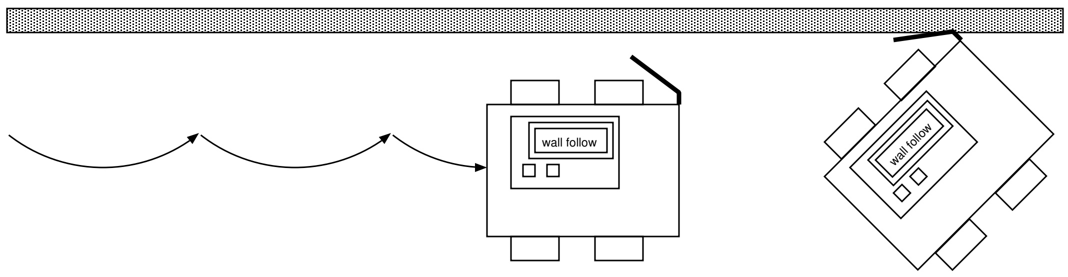

Figure 7.6: Wall following and a jammed robot

7.3.2 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.

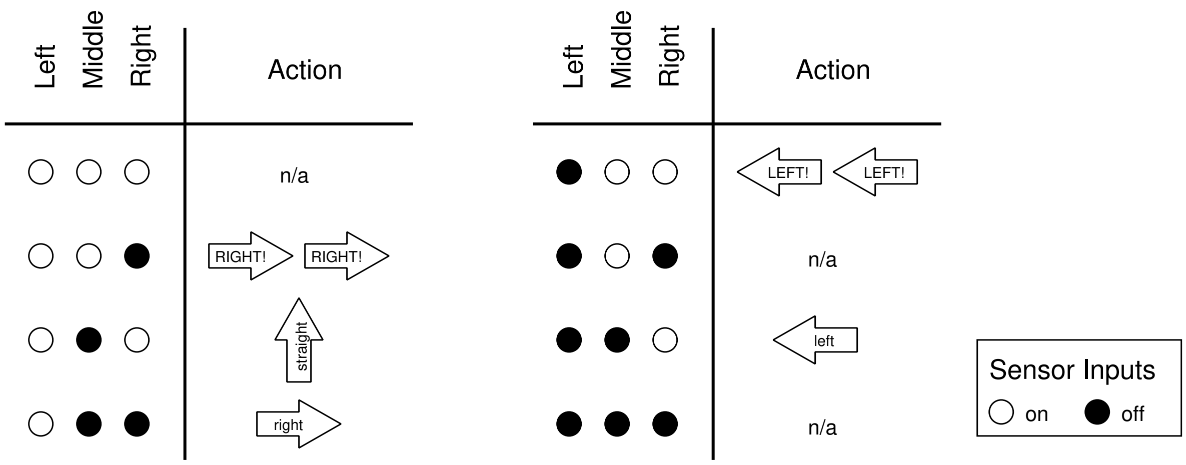

Figure 7.7: Line following with 3 reflectance sensors

Figure 7.7 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.

7.3.3 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.

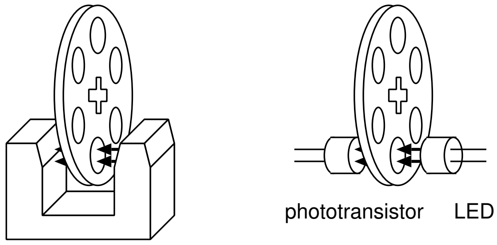

Figure 7.8: Shaft encoding using a LEGO pulley wheel

A shaft encoder is constructed from a breakbeam sensor and a Lego pulley wheel as shown in Figure 7.8. 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.

7.4 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.

Appendix A

IC commands for 6.270

The Interactive C Manual For the Handy Board describes the basic set of commands you will have access to on the Handy Board. In addition, to make use of the 6.270 expansion board, you will need to use several extensions to the standard IC library, described in this appendix.

A.1 Expansion Board: Motors, Analog Inputs, Digital Outputs

There are two motor ports on the expansion board, numbered 4 and 5. You can use them just as you use the motors on the main board:

```
fd(4);
motor(5,-100);
/* etc... */
```

Since the expansion board motors are not speed-controlled, all positive speeds and all negative speeds are equivalent to 100 and -100, respectively.

The expansion board add 16 analog inputs, labelled 16–31. They are used just like the analog ports on the main board:

```
analog(23);
```

When the expansion board is in use, ports 1 and 2 on the main board can't be used.

There are three digital outputs: `do0`, `do1`, and `do2`, which correspond to ports 8, 7, and 6 in software respectively. We recommend connecting the infrared beacon to `do0`, so there are two digital outputs left, that may be used to control LEDs, speakers, or other devices. To turn the three digital outputs on and off, use the following commands:

```
set_digital_out(p);
clear_digital_out(p);
```

where p is replaced by whatever port you wish to control.

A.2 Expansion Board: Servos

The expansion board has the ability to control up to six servos. To turn on the servos, run the following command:

```
enable_servos();
```

To turn off servo power, run this:

```
disable_servos();
```

The function `servo(int port, int period)` sets the servo to an angle that depends on `period`, which can vary from 0 to 4000.

A.3 Start-light Sensing and IR Beacon Code

Connect your IR beacon's control line to port `do0` on the expansion board and its power line into a servo output. You can then start and stop the beacon with the following commands:

```
ir_transmit_off();
ir_transmit_on();
```

The IC libraries include code to detect the blinking light which signals the start of the match[1]. Call the procedure `start_machine(int port)` (with the port number of your starting light sensor) when the light is blinking and your robot is ready. The code will walk you through the starting routine, consisting of the following steps:

1. The number displayed on the LCD is the intensity of the blinking light, as measured by your light sensor. Position your robot so that this is maximized. Any value that is consistently above zero will work (try it!), but you may want to get it up above five just to be sure.

2. Announce "Ready for off" and wait until the light goes off.

[1]If you are interested in the source code for the program that controls the table, you can find it at `/mit/6.270/Public/src/table/`. We will be updating this as necessary.

3. Arm your robot by pressing the START button while the robot is over the unlit starting light.

4. Announce "Ready to GO" and step back.

5. When the light comes on, the robot will go!

6. In sixty seconds, the robot will stop!

The following is the basic outline for a program that follows the rules of 6.270. You can find a copy of this program at `/mit/6.270/Public/src/startstop/test.c`. Note that you must not use the function `reset_system_time()` in your code or you will confuse the robot about when to stop.

```
#define START_LIGHT_PORT 31

void main() {
  /* don't transmit yet! */
  ir_transmit_off();

  /* align with start light and go! */
  start_machine(START_LIGHT_PORT);

  printf("GO!!!\n");

  /* Your code goes here... */
}
```

Appendix B

Expansion Board Assembly

While the main board comes fully assembled, you will have to assemble the newer expansion board yourself. If you are not familiar with soldering, please attend the soldering workshop to learn the basics, and if you have any further questions, no matter how small, please contact a member of the staff before you proceed. We'll be much happier if you come to us before you've made a mistake than afterward. That said, building the expansion board should not be much of a challenge for anyone, and we recommend that you get started on it as soon as possible, and not fall behind. The circuit board may look complicated, but do not despair! Many of its parts (such as circuitry to make it compatible with LEGO Mindstorms sensors) are optional and not needed for 6.270. For reference, a complete, assembled Handy Board is pictured at

`http://www.mit.edu/~6.270/images/expbd.jpg`

Although missing LEDs and the sockets for some ICs the board is functional. Also visible is the beacon power/signal cord, correctly connected to the expansion board.