Roborodentia Robot
(Amazon Prime)
Senior Project, Spring 2016

Alec Cheung
Advisor: Dr. John Seng

Table of Contents

Introduction

Roborodentia is an annual autonomous robotics competition sponsored and hosted by Cal Poly. In the 2016 competition, participants are to design a robot that scores the most points by gathering rings from marked supply pegs and placing them onto marked scoring pegs. For Roborodentia I designed, constructed, and programmed a robot, named Amazon Prime, to compete.

Problem Statement

In the competition the robot must abide by the following rules:

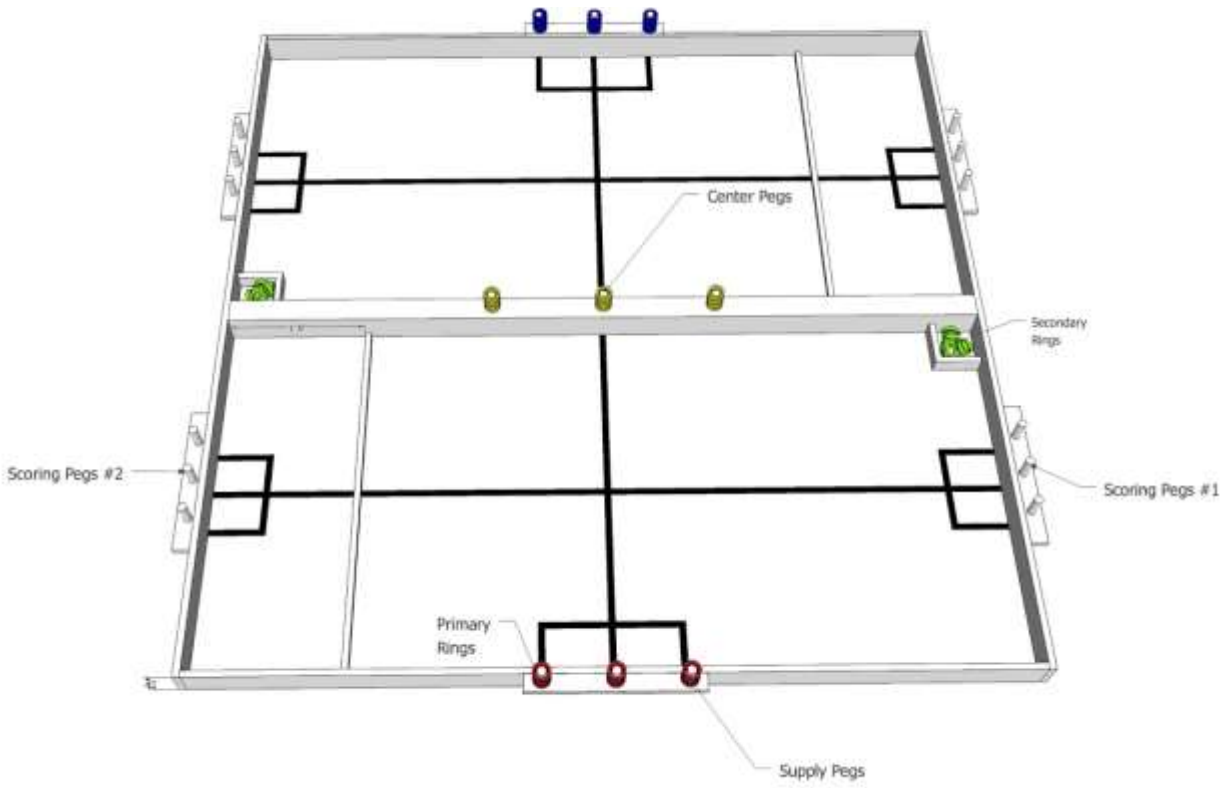
- it must be fully autonomous
- it must have a 12” x 12” footprint or smaller
- it can have a maximum height of 15”
- it cannot fly
- it cannot interfere with a competitor’s robot

The robot can score points by collecting rings from the supply pegs (1 point) or the secondary rings from the supply box (2 points) in the corner as labeled in **Figure 1** below. There are also yellow multiplier pegs that individually count as three points and multiply the value of the rings on the scoring peg by three.

Points are awarded to the robot after there are at least four rings on a scoring peg, at which point the rings will be removed to allow for more rings. If a robot manages to score more than four rings in a single collection, the additional rings will not be counted.

At the end of a 3 minute match, of the two participants currently competing, the one with more points will be announced the winner.

Figure 1: 2016 Roborodentia Course



For additional information on the rules visit:
https://docs.google.com/document/d/1rYSVdBPb6dNHQXfgRWYkEfzj00706zCozCtoAm_0bv4/pub

Software

The software for Amazon Prime is written in C on the Arduino Mega 2560.

Below in **Figure 2** is a block diagram of the overall architecture for the software:

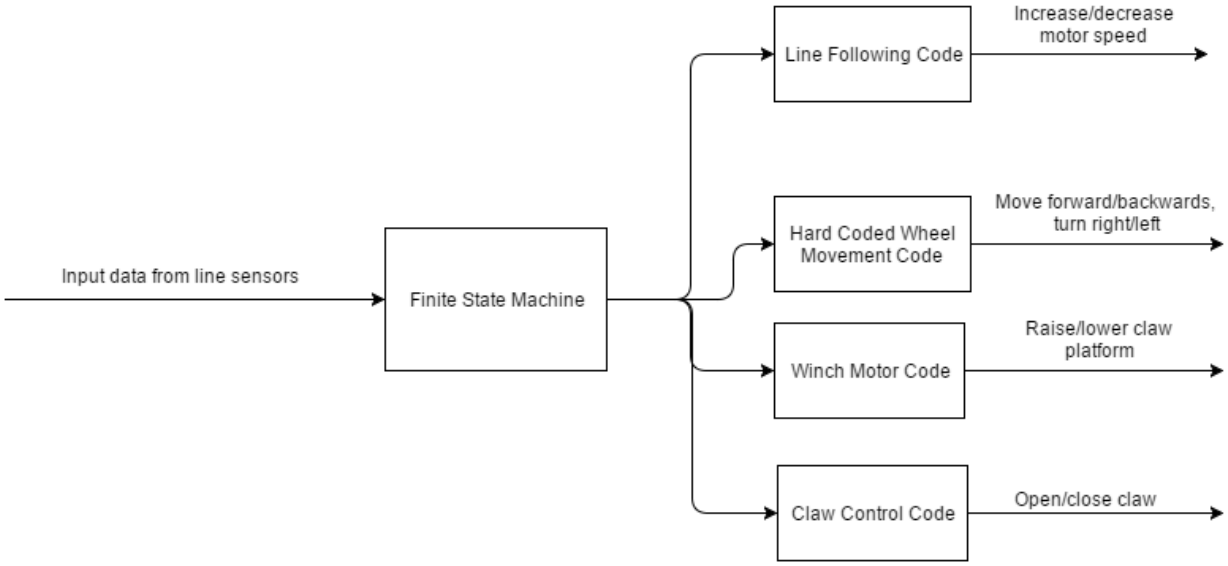


Figure 2: Block Diagram of Overall Software Architecture

Amazon Prime functions by reading input from the 5 IR sensors mounted on the underside of the robot. After computing the values, signals are sent to the wheel motors, winch motor, or claw servo to perform the appropriate actions according to the finite state machine.

Line Following

For the line following code, I used input from the sensors to determine the orientation of the robot with respect to the line. Since there are five IR sensors aligned in a row 2 cm apart from each other (illustrated in the Mechanical, Sensors section) to detect the line, it is possible to determine how far the robot has strayed from the line. Depending on which sensor hits the line, a correction constant is returned by the getPosition function as shown in **Figure 3**.

```
86  int getPosition() {
87    int pos = 0;
88    if (onLine(right_outer)) {
89      pos = 4;
90    }
91    else if (onLine(left_outer)) {
92      pos = -4;
93    }
94    else if(onLine(right_inner)) {
95      pos = 2;
96    }
97    else if (onLine(left_inner)) {
98      pos = -2;
99    }
100   return pos;
101 }
```

Figure 3: getPosition function that returns correction constant

This is then used to compute the corrected motor speed for each motor to readjust the heading of the robot. This is shown below in the lineFollow function in **Figure 4**.

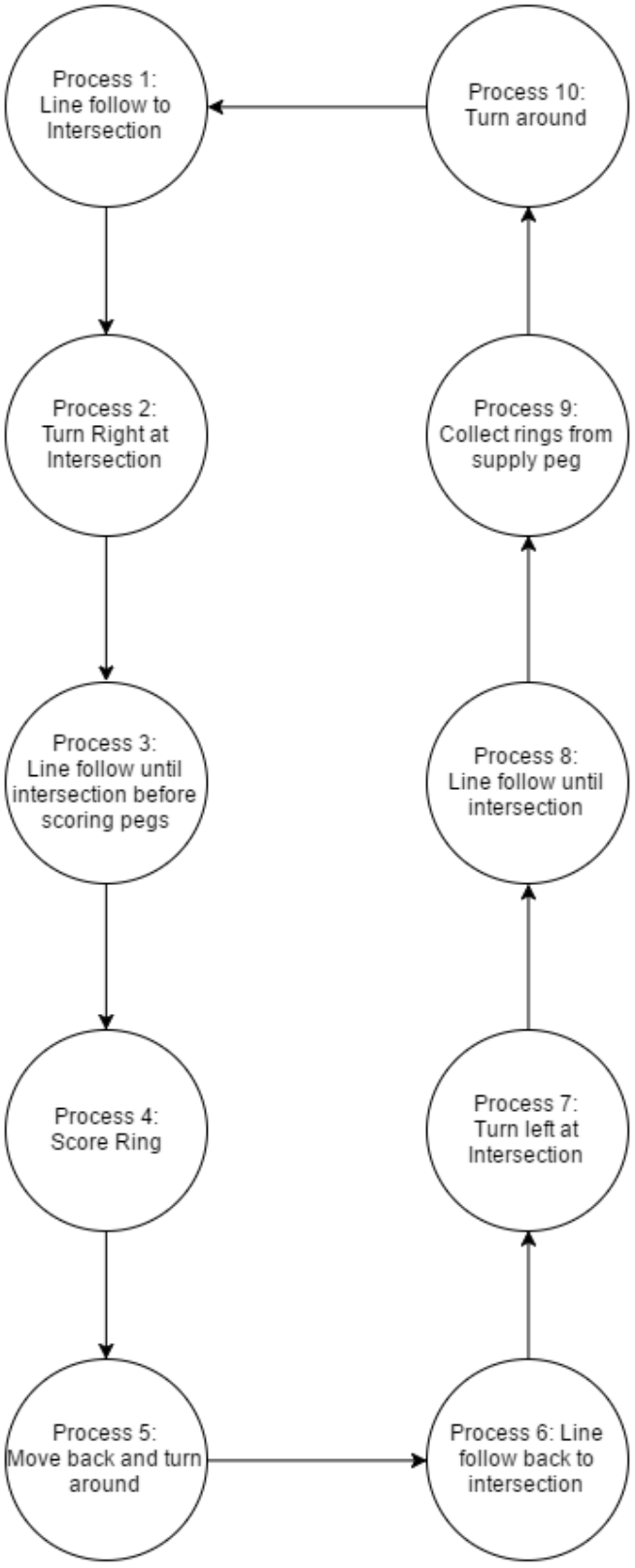
```
103 void lineFollow() {
104   error = getPosition();
105
106   int correction = kp * error;
107   int r_speed = def_speed - correction;
108   int l_speed = def_speed + correction;
109   if (r_speed > 128) {
110     r_speed = 128;
111   }
112   if (r_speed <-128) {
113     r_speed = -128;
114   }
115   if (l_speed > 128) {
116     l_speed = 128;
117   }
118   if (l_speed < -128) {
119     l_speed = -128;
120   }
121   roboShield.setMotor(r_motor, r_speed);
122   roboShield.setMotor(l_motor, -l_speed);
123 }
```

Figure 4: lineFollow function that sets motors to corrected speed

Finite State Machine

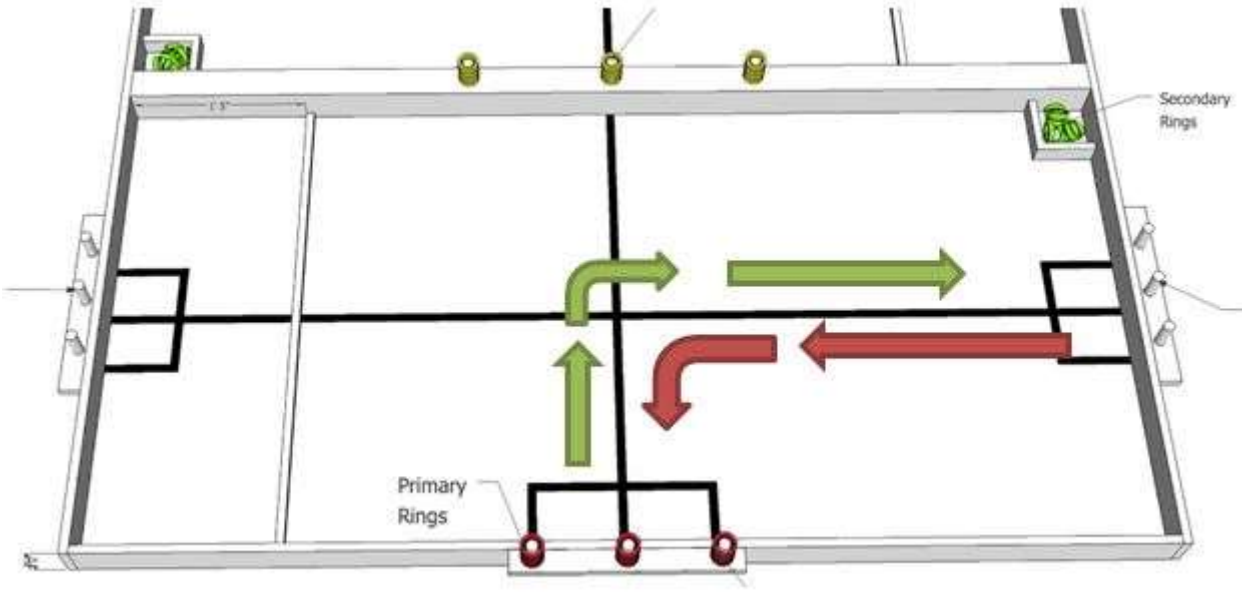
The finite state machine (FSM) is the controlling code of the robot. It handles line following/traversal of the course, along with ring acquisition and scoring. The FSM has 17 different states that can be abstracted into 10 general processes as shown in **Figure 5**.

Figure 5: Overall Summary of FSM



Controlled by the FSM, Amazon Prime will traverse the course as outlined in **Figure 6** below.

Figure 6: Amazon Prime Course Traversal



Intersection Detection

In the FSM, state changes depend upon either intersection detection or timed delays. Intersection is detected when 3 or more sensors are over the black tape, as shown in the code in **Figure 7** below.

Figure 7: Intersection Detection code

```
132 bool hitIntersection() {
133   int sum = 0;
134   if (onLine(right_outer) ) {
135     sum++;
136   }
137   if (onLine(left_outer)) {
138     sum++;
139   }
140   if (onLine(right_inner)) {
141     sum++;
142   }
143   if (onLine(left_inner)) {
144     sum++;
145   }
146   if (onLine(center)) {
147     sum++;
148   }
149   return sum >= 3;
150 }
```

Process 1, 3 and 8 – Line Following

Process 1, 3 and 8 are fairly similar as they are mainly line following, as explained previously, until reaching an intersection. The code for Process 1 is shown below in **Figure 8**. Code for Process 3 and 8 are identical to Process 1 with the exception that the state changes are different as expected.

Figure 8: Code for Process 1 – Line Following

```
177   if (state == line_follow) {
178     lineFollow();
179     if (hitIntersection()) {
180         state = middle_inter;
181     }
182   }
```

Process 2, 5, 7, and 10 - Turning

Processes 2, 5, 7 and 10 behave similarly as they all focus on turning Amazon Prime. The principle for turning relies on Amazon Prime's sensors. Although actual sensors may differ based on which direction the robot is turning, the principle remains the same. On a turn, Amazon Prime turns right until the left_outer sensor (from the perspective of looking at the robot) hits a line. Then it keeps turning until the right_outer sensor hits the line, at this point the robot turns in the opposite direction until the left_inner sensor hits the line. Now Amazon Prime is positioned sufficiently such that line following will function correctly. The code for Process 2 can be found below in **Figure 9**.

Figure 9: Code for Process 2 - Turning Right

```
188  else if (state == right_turn_1) {
189    if (onLine(right_outer)) {
190      roboShield.setMotor(r_motor, 0);
191      roboShield.setMotor(l_motor, 0);
192      state = right_turn_2;
193    }
194    else {
195      roboShield.setMotor(r_motor, 20);
196      roboShield.setMotor(l_motor, 20);
197    }
198  }
199  else if (state == right_turn_2) {
200    if (onLine(left_outer)) {
201      roboShield.setMotor(r_motor, 0);
202      roboShield.setMotor(l_motor, 0);
203      state = right_turn_align_left;
204    }
205    else {
206      roboShield.setMotor(r_motor, 20);
207      roboShield.setMotor(l_motor, 20);
208    }
209  }
210  else if (state == right_turn_align_left) {
211    if (onLine(right_inner)) {
212      roboShield.setMotor(r_motor, 0);
213      state = approach_supply;
214    }
215    else {
216      roboShield.setMotor(l_motor, 25);
217    }
218  }
```

Process 4 - Scoring

Process 4 handles the positioning and dropping of the claw. Once Amazon Prime is at the intersection in front of the scoring pegs, it will move forward for a constant amount of time to position the claw above the scoring peg. After moving, the controlClaw function (shown in **Figure 10** below) is called to release the claw to drop the rings into the scoring peg. **Figure 11** below shows the code for Process 4.

Figure 10: Code to control claw

```
152 void controlClaw(int c_state) {
153   if (c_state == open_claw) {
154     roboShield.setServo(claw_motor, 90);
155   }
156   else if (c_state == close_claw) {
157     roboShield.setServo(claw_motor, -100);
158   }
159 }
```

Figure 11: Code for Process 4 - Scoring Rings

```
225   else if (state == align_claw) {
226     moveStraight(forward, 15, 22);
227     delay(315);
228     halt();
229     roboShield.setMotor(winch, -20);
230     delay(1300);
231     roboShield.setMotor(winch, 0);
232     controlClaw(open_claw);
233     state = retreat;
234   }
```

Process 9 – Collecting Rings

Process 9 handles collecting the rings from the supply pegs. Once at the intersection in front of the supply pegs, Amazon Prime moves forward so that claw is at the rings, closes the claw, and sends a signal to the winch motor to lift the claw with the rings. **Figure 12** below is the code for process 9.

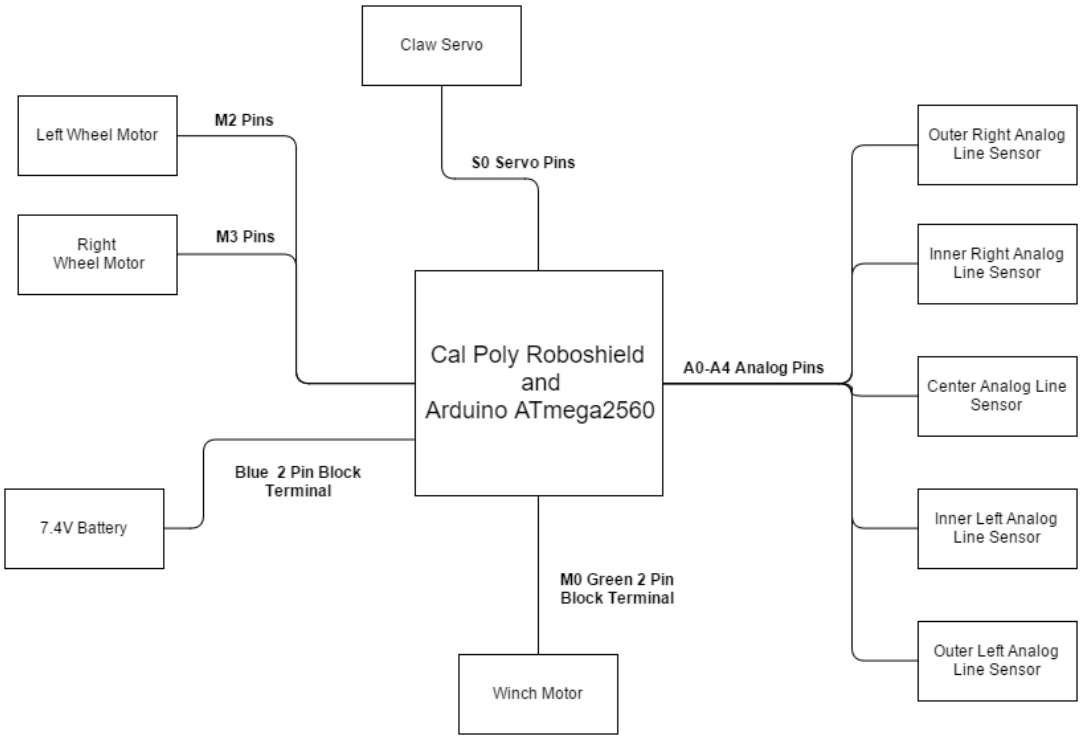
Figure 12: Code for Process 9 - Collecting Rings

```
else if (state == score) {
  int saved_speed = def_speed;
  def_speed = 17;
  lineFollow();
  if (hitIntersection()) {

    state = turn_around_reset;
    moveStraight(forward, 20, 22);
    delay(315);
    halt();
    delay(500);
    controlClaw(close_claw);
    halt();
    roboShield.setMotor(winch, 40);
    delay(3000);
    roboShield.setMotor(winch, 0);

  }
  def_speed = saved_speed;
}
```

Hardware

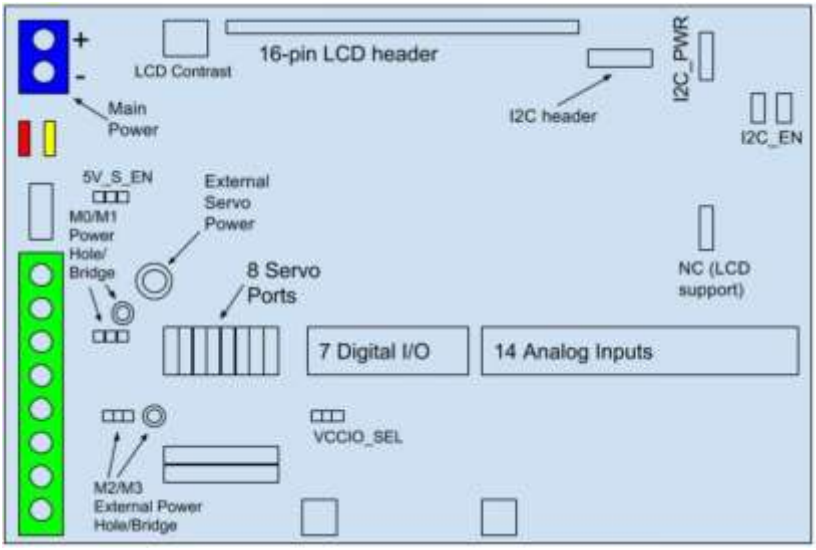
Below is the high level hardware architecture block diagram in **Figure 13**.

Figure 13: High Level Hardware Architecture Block Diagram



Amazon Prime uses an Arduino in combination with a Cal Poly Roboshield as the microcontroller. A diagram of the Roboshield can be found in **Figure 14** below.

Figure 14: Roboshield Pinout



Additional information on the Roboshield can be found at http://roboshield.github.io/roboshield/index

Mechanical

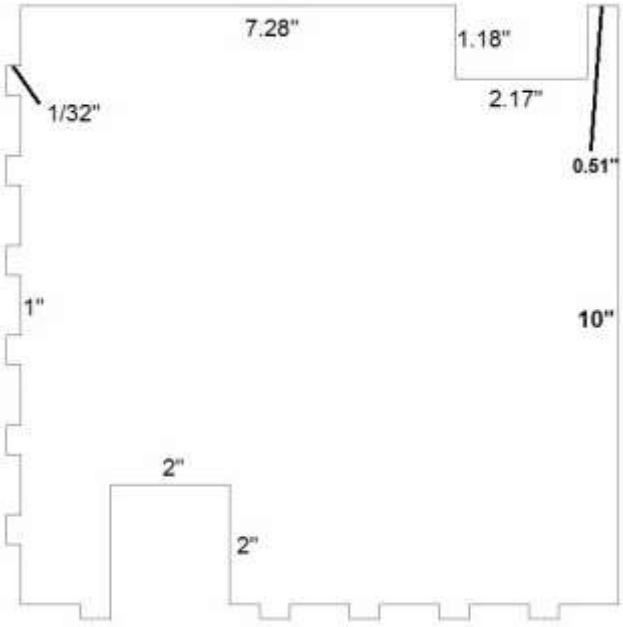
Amazon Prime is designed to be a simple, functional box robot with two motors in the back for movement, a ball bearing in the front center for support, and an elevator system for claw lifting/lowering. Below in **Figure 15** is the completed Amazon Prime.

Figure 15: Complete Amazon Prime

Frame

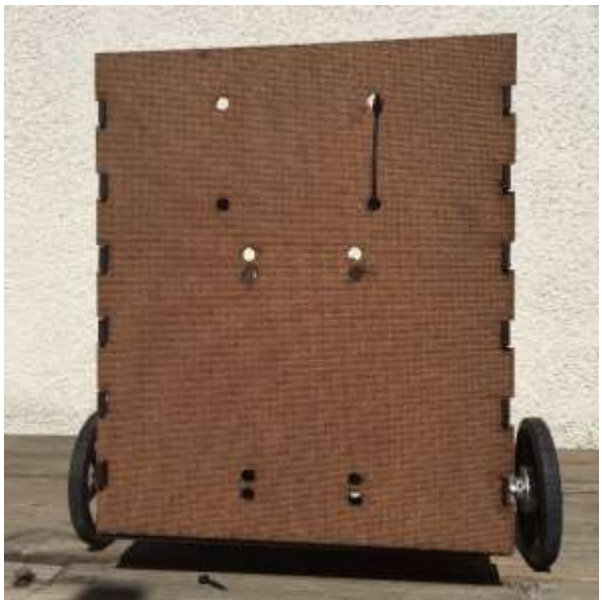
Amazon Prime's frame is a simple box design with puzzle-piece edges, allowing the walls of the frame to be glued together. The frame and all wooden parts are made from 1/8" thick laser cut hardboard. The left and right side walls have squares cutout for the wheel motors and the winch motor that handles lifting or lowering the claw. **Figure 16** below is the right wall.

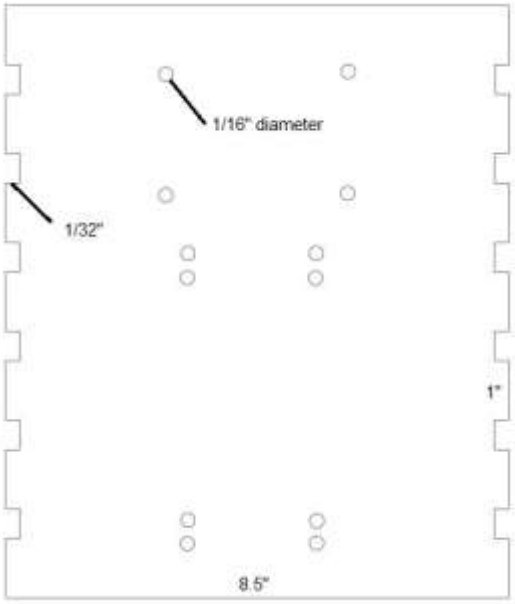
Figure 16: Amazon Prime's Right Wall (Left) and Mechanical Drawing (Right)



The back wall has holes that are used to attach the Arduino and Battery using cable ties. Below, in **Figure 17**, are Amazon's back wall and a mechanical drawing.

Figure 17: Amazon Prime's Back Wall (Left) and Mechanical Drawing (Right)

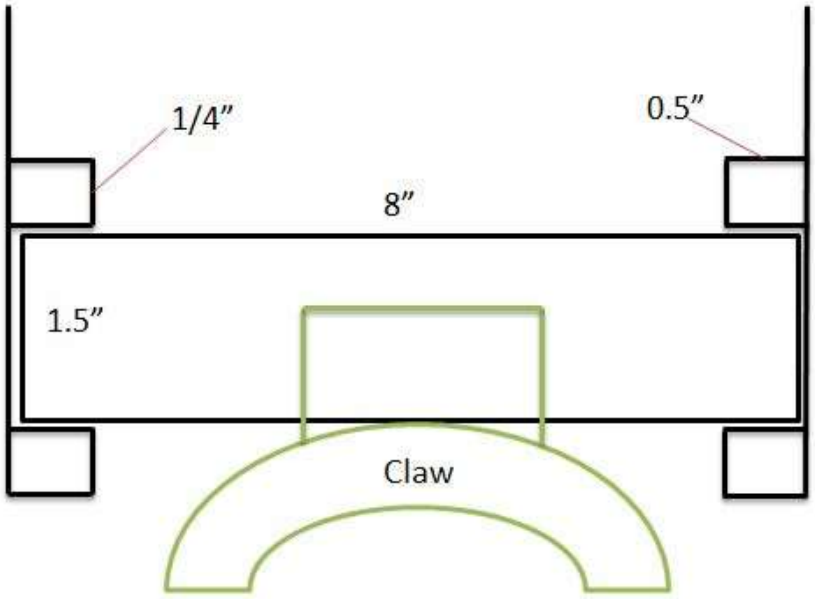




Lift

The lift design consists of a rectangular platform with a claw attached to the center. The platform is held in place by two vertical railings on each end. Four lines of twine are attached to the corners of the platform connecting it to the winch motor that raises and lowers it. **Figure 18** below shows the front view of the lift and a mechanical drawing of the top view without the winch motor.

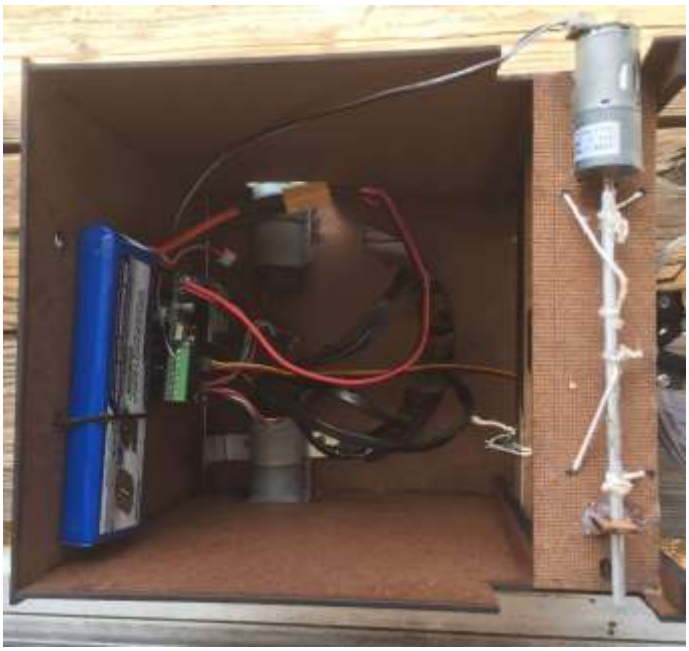
Figure 18: Lift Front View (Left) and Top View Mechanical Drawing (Right)

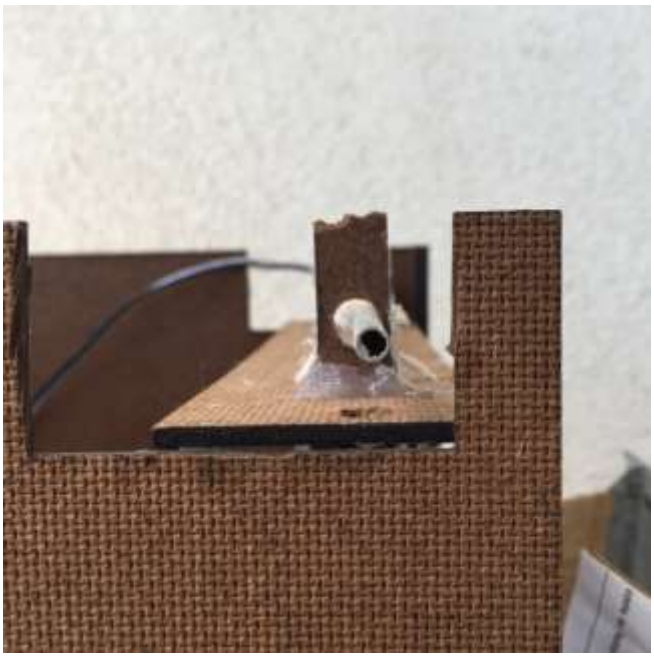


Winch

The winch is the driving force of the lift. It consists of a hollow 1/16” metal rod attached to the end of a motor powered by the Roboshield. The metal rod is held up by the motor and a piece of wood with a hole. Twine from the lift platform is also attached to the metal rod allowing the motor to wind the lift up or down. **Figure 19** below is a top view image of the winch and a right side view of the metal rod being held up.

Figure 19: Winch Top View (Left) and Right Side View (Right)

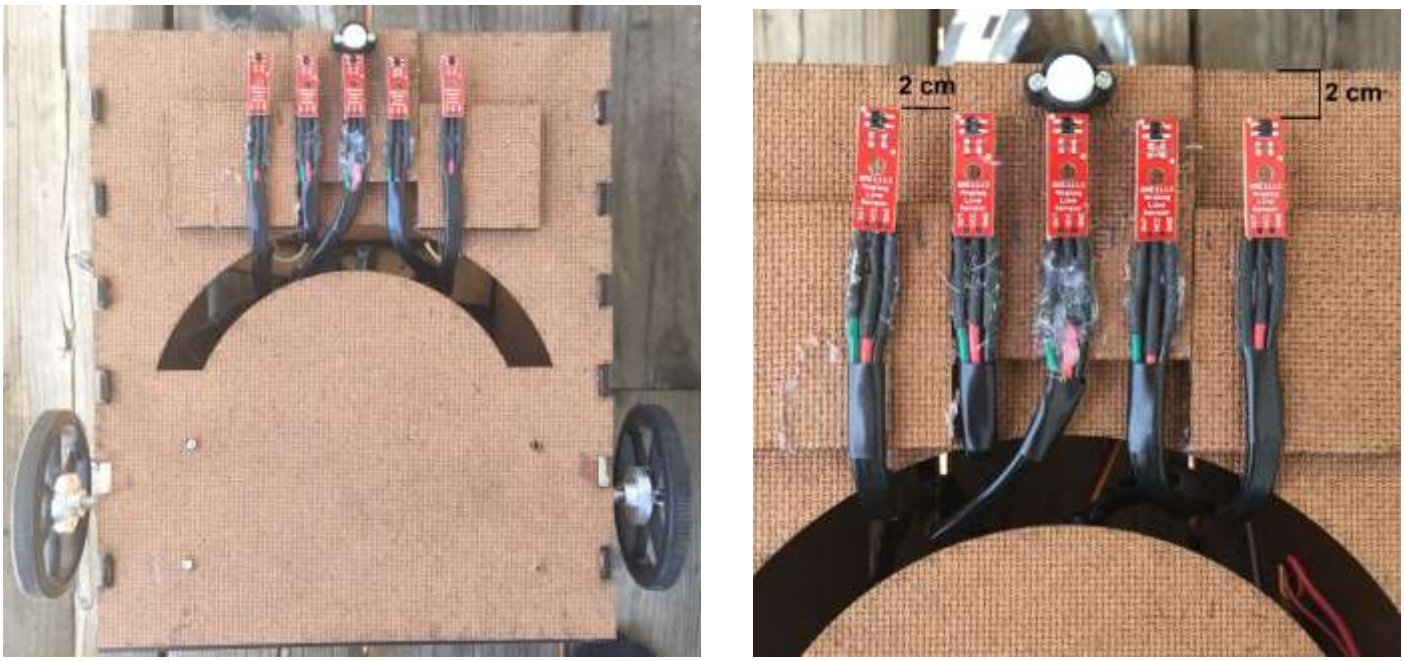




Sensors

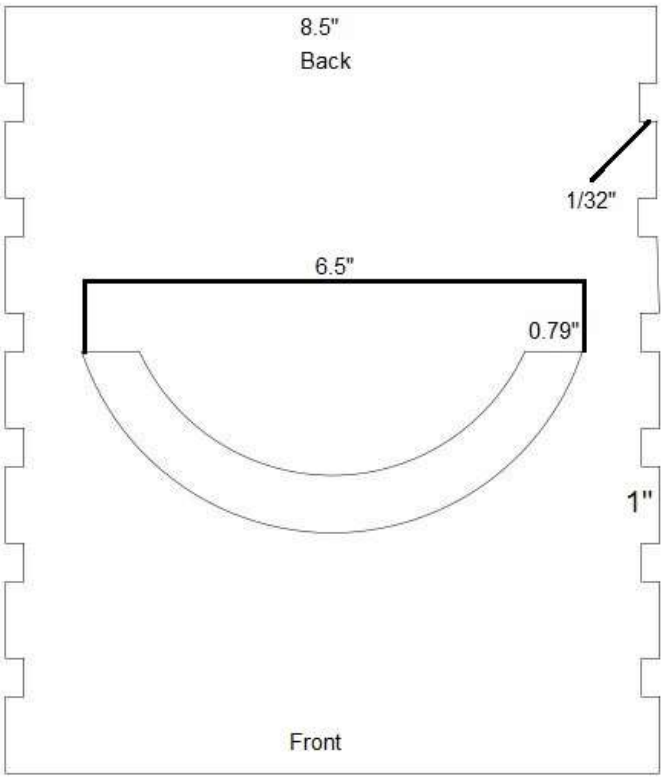
Amazon Prime has 5 analog IR sensors aligned 2 cm apart from each other in a row. They are mounted 2 cm below the front through the semicircular opening onto the underside of the robot on top of 1/8” thick squares of hard board. The 1/8” thick squares of hard board positions the sensors closer to the ground allowing them to read more accurate input.

Figure 20: Sensors Full View (Left) and Zoomed In View (Right)



Below, in **Figure 21**, is a mechanical drawing of Amazon Primes bottom.

Figure 21: Amazon Prime Bottom Mechanical Drawing



Budget and Bill of Materials

In the end, there was no hard budget for the robot, but I tried to keep the price as low as possible. However, as it is shown here, the number of components required increased the total price, even though most of components were relatively cheap.

Table 1: Bill of Materials

Parts	Vendor	Quantity	Price	Total
Standard Gripper Kit A - Channel Mount	SparkFun	1	9.95	9.95
SparkFun RedBot Sensor	SparkFun	6	2.95	17.70
SunFounder Mega 2560 R3 ATmega2560-16AU	Amazon	1	17.99	17.99
Hitec 31311S HS-311 Servo	Amazon	1	10.28	10.28
Turnigy 5000mAh 2S 20C Lipo Pack	Amazon	1	24.49	24.49
XT60 Drone Connectors 5 Pairs	Amazon	1	3.07	3.07
Integy RC Hobby C23212 LiPo Voltage Checker + Warning Buzzer	Amazon	1	4.75	4.75
SGT KNOTS Non-Skid Safety Tape	Amazon	1	8.95	8.95
Slideez Lubicrant	Amazon	1	8.94	8.94
Torpedo 4 oz. Fishing Sinker	Amazon	1	4.59	4.59
Ball Caster	Cal Poly Robotics Club	1	5.00	5.00
Roboshield	Cal Poly Robotics Club	1	30.00	30.00
Metal Gearmotor 37Dx52L mm with 64 CPR Encoder	Cal Poly Robotics Club	2	15.00	30.00
Pololu Wheel 80×10mm Pair	Pololu	1	9.25	9.25
Pololu Universal Aluminum Mounting Hub	Pololu	1	7.95	7.95
Pololu Stamped Aluminum L-Bracket Pair for 37D mm Metal Gearmotors	Pololu	1	7.95	7.95
DC motor	Radioshack	1	5.99	5.99
Titebond III Ultimate Wood Glue	Home Depot	1	5.97	5.97
Hardboard Tempered (1/8 in. x 2 ft. x 4 ft.)	Home Depot	8	4.97	39.76
Steel Safety Plate (5 in. x 8 in.)	Home Depot	6	1.57	9.42

Twine	Cal Poly Campus Store	1	7.99	7.99
Hollow Metal Rod 1/16”	Cal Poly Campus Store	1	2.99	2.99
Hot Glue Gun	Cal Poly Campus Store	1	5.99	5.99
Hot Glue Sticks	Cal Poly Campus Store	1	2.99	2.99
			Total:	**281.90**

Lessons Learned

Moving Components

The most important thing I have learned in creating Amazon Prime is to prototype a lot before deciding on a final design. The most temperamental piece of the robot was definitely the moving platform that raised and lowered the claw. I had to make five to six modifications before getting to a design that was reliable enough to use in the competition.

Originally, I had the platform supported by two strings in the center of the left and right edge. The strings would be wound up by a motor acting as winch to lift the platform. However, this did not work because the platform leaned forward due to the weight of the claw. After some trial and error, I decided to switch to four strings at each corner of the platform to provide stabilization. This almost resolved the problem but the left edge of the platform would often get caught on the side wall of the robot due to friction. Once the edge was caught it would completely destabilize the platform from the track, making it impossible to lift or lower the platform. In order to fix the problem, I had to place a fishing weight to overcome the friction with the wall.

Although the final claw lifting mechanism was mostly reliable, I would have preferred to have switch to another guaranteed mechanism, which is why it is important to leave a lot of time to test the mechanical design, especially moving parts. This was even more important for me due to my lack of mechanical design experience.

Mechanical Design

I think the most important thing I have learned in relation to the competition is to measure everything properly and to take more advantage of the mechanical design to solve the problem. The biggest limitation in scoring points was the actual dropping of the rings over the peg. Amazon Prime uses time delays, and constant motor speeds to move the claw directly over the scoring peg. However, this was unreliable because sometimes the claw was not forward enough or it was too far forward, so the rings would miss the scoring peg.

It would have been more reliable if I had designed the robot with a bumper on the front of the robot that would displace the claw enough for it to be properly centered over the scoring peg. Using this method, I could have just made Amazon Prime move forward until it hit the end of the wall, at which point the claw would be aligned by the bumper.

Mechanical design and measurement was also a problem in the claw lowering the procedure for collecting rings from the supply peg. I had placed wooden stoppers at the bottom of the platform tracks to limit how low the platform could descend. However, my measurements were not very exact, so the lowest point was actually too tall, causing the claw to often only be able to collect three rather than four rings. Unfortunately due to time constraints, I was unable to readjust the wooden stoppers, so Amazon Prime was not performing optimally.

Proper Voltage Control

An annoying factor that I had not considered was the voltage of the battery. This is actually important because it has a direct impact on the motor speeds. This means that the motor speeds will differ when the battery has been drained, even by a relatively minor amount, from testing.

It would have made testing, line following, and time delayed movements much simpler if there was a way to control the voltage to the motors, to create more reliable and repeatable movements.

Complexity versus Simplicity

During the beginning of the design process I had wanted to create an extremely complex robot that had many moving parts. However, after more consideration, I went with the opposite and decided upon Amazon Prime's current simple design. I thought that I could supplement a simple design with more complex software, but some limitations were too big to overcome given the hardware.

Although the design was functional, it was apparent that the maximum points that it could score was far less than some of the other robots in the competition. I feel that I should have been more ambitious in trying to score points, although I do admit that Amazon Prime's success was due to its reliability and simplicity.

I learned that a simplistic design may not always be the best, but neither is a complex one. I have found that it is difficult to find a balance between a simple and complex design.

Use Reliable Version Control

A surprising problem that I did not expect was version control. In a project where I worked alone, I did not think that robust version control software like git or svn was necessary. Instead I used Microsoft's OneDrive to sync Amazon Prime's code between my laptop and my desktop computer. Unfortunately OneDrive has been proven to be unreliable and destructive in certain cases.

On one occasion, OneDrive completely overwrote my code with a blank file. When I was uploading code to the Arduino, my laptop had run out of power and shutdown. This caused all my code to be wiped from the Arduino IDE, so when I restarted my computer OneDrive automatically synced the blank file to cloud storage.

If I had used git or svn to maintain my files, this would not have been a problem because they require manual commits for files to be saved. Using proper version control would have saved a lot time and effort.

Conclusion

Amazon Prime did much better than I expected. It managed to get third place and most of the features were successfully implemented. The only drawback was that it was not as ambitious as the first and second place robots, so it had a lower range of maximum points that it could score. However, I do feel that its relatively simple design did contribute a lot to how well it performed.

From participating in Roborodentia, I have come to learn that designing a robot, even a simple one, is a lot more complicated than expected. You have to account for mechanical, electrical, and software related issues and as a Computer Science major, I can definitely say that physically creating a robot provides a whole new set of problems that is not found in programming or software design. It has shown me some of the limitations of software and how hardware can supplement software to solve the problem.

Appendix A: Code

The full code can be found in the following GitHub Repository:

https://github.com/AlecChng/AmazonPrime