Roborodentia Robot 2015

Travis Stuever

CPE 462: Senior Project

Advisor: John Seng

Spring 2015

Contents

List of Figures

1 Problem Statement

The 2015 Roborodentia event was a ring challenge that required robots to pick up rings from a horizontal three peg setup, as seen in Figure 1 and then drop these rings off on a vertical three peg setup which can be seen in Figure 2. The point values of the drop off pegs go from 2, 4 and 6 points per ring in regards to the height of the peg.

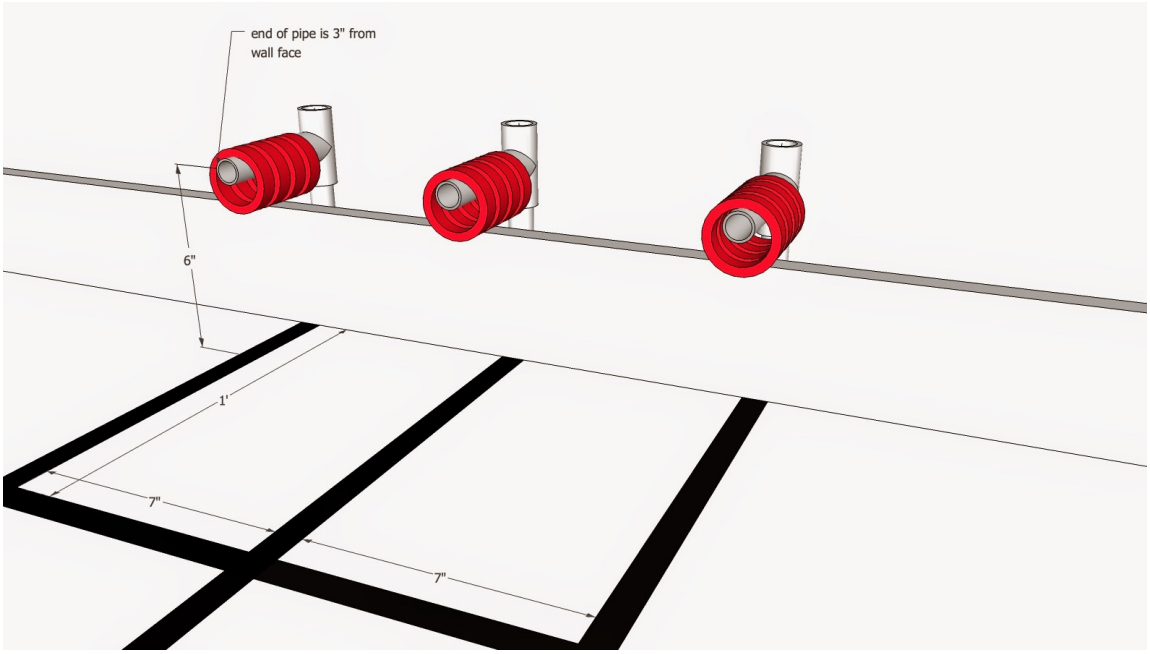

Figure 1: Pick Up End of Roborodentia Course.

There is also a middle peg on the course that multiplies the overall score of the contestant with the highest ring by 2, the design for my robot did not utilize this peg. There are also other ways to score besides the pegs but my design did not utilize these other approaches to scoring. Matches go for 3 minutes and the robot with the highest score at the end wins that round.

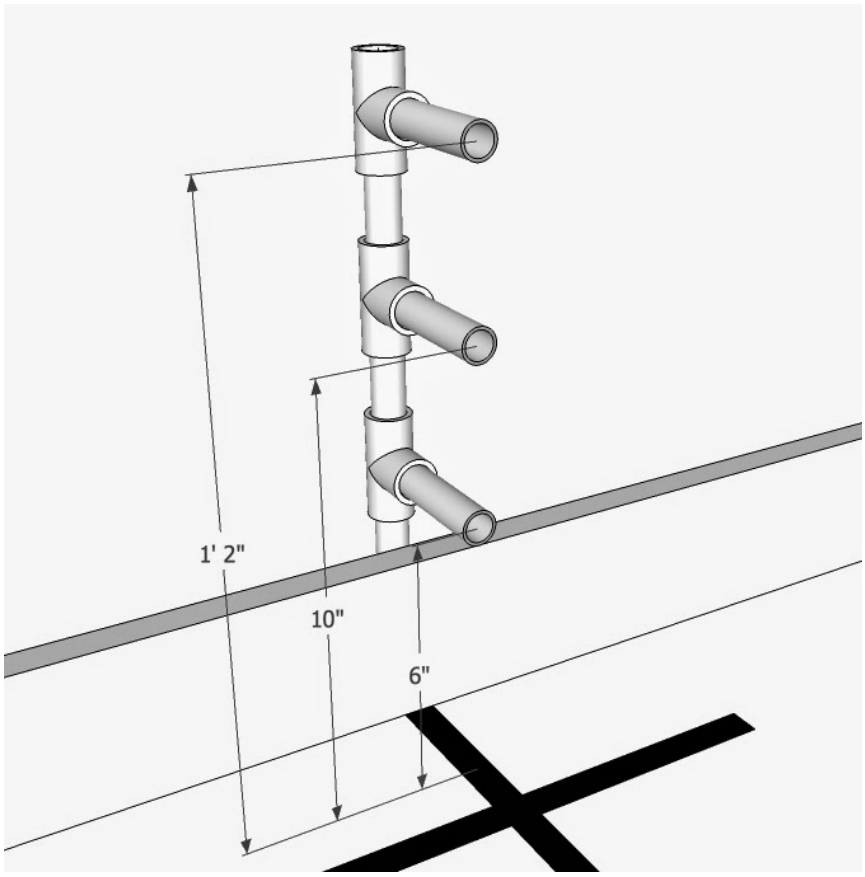

Figure 2: Drop Off End of Roborodentia Course.

2 Design

I approached the design of my robot with the mindset of trying to utilize the middle receiving peg and the lowest drop off peg. These two pegs are directly across from each other and I felt they would be the fastest way to score points. I also felt that it would be faster to score if I did not have to worry about picking up rings from the right and left pegs, but just the middle one. In regards to the higher drop off pegs, I felt it would require too much additional logic and hardware to attempt.

2.1 Line Following

Knowing that there would be a black line to follow from one side of the course to the other, I knew that I would have to implement some hardware and logic in order to follow this line. I chose to use 3 white LEDs coupled with three light dependent resistors (LDRs) on both the receiving and dropping sides of my robot in order to know when my sensors were over the black line or the white surface of the course.

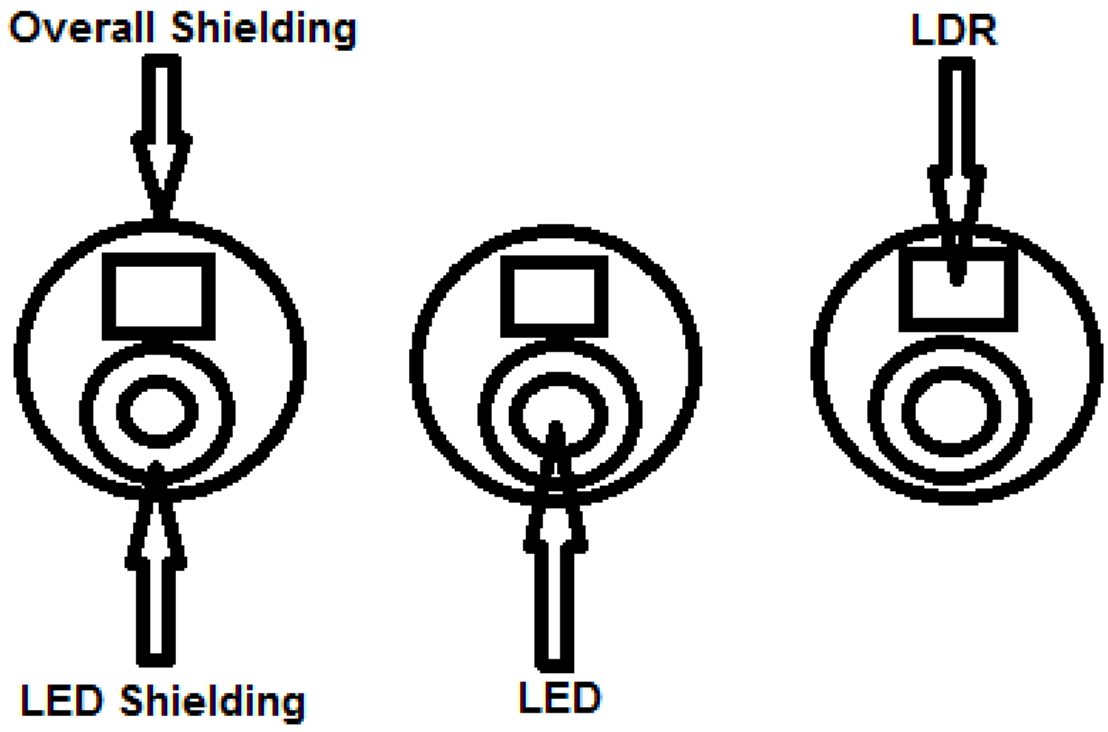

Figure 3: Line Following.

As you can see in Figure 3 there is a shield covering each pair of components so that the LED/LDR pairs do not affect the other LED/LDR pairs. Additionally, LEDs would be isolated from the LDRs by a blacked out shield which I thought

would isolate the light going to the LDRs. This means that all the LDR will see is the reflected light off the surface of the course. In Figure 4 you can see how the LEDs and LDRs would be ideally situated over the black line on the course.

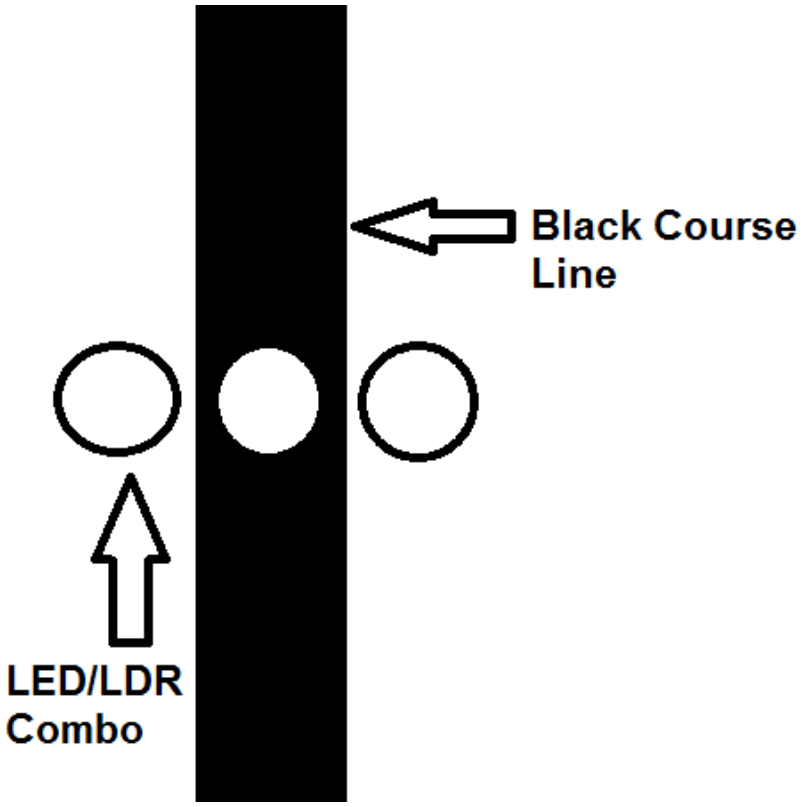

Figure 4: Line Follow Design.

2.2 Ring Grabbing

The next component of my robot that I had to think about was how to pick up and drop off the rings. I came to the conclusion that it would be ideal to have one servo control horizontal motion. This servo would be used to swing the rings from the receiving end of the robot to the dropping end of the robot.

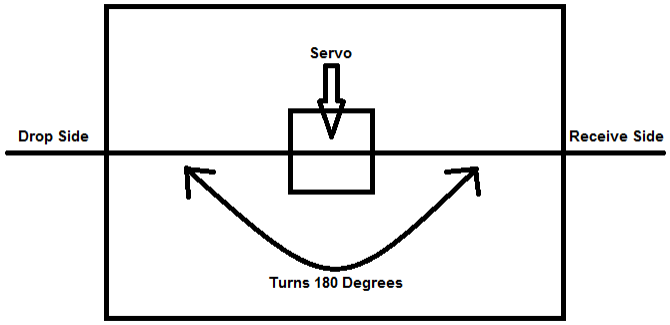

Figure 5: Base Horizontal Servo Overview.

As seen in Figure 5 the servo would be positioned in the middle of the robot and be able swivel from one side to the other. Figure 6 shows the vertical servo attached to the swivel servo by a shaft and how this servo will move up and down in order to pick up and drop off the course rings.

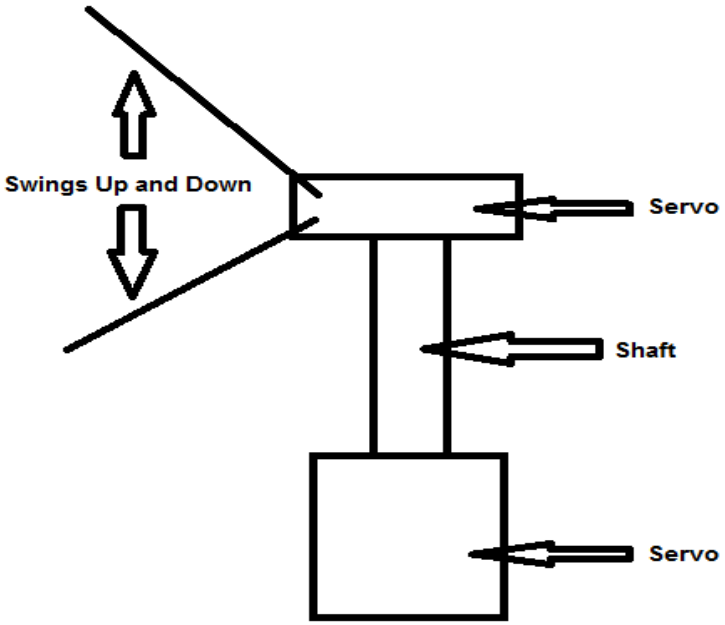

Figure 6: Vertical Servo Side View.

2.3 Other Components

The rest of my design revolves around the placement of the motors, caster wheels and the battery pack. As seen in Figure 7, the placement of all the components I have thought about fit nicely within a 9 inch by 9 inch piece of cardboard.

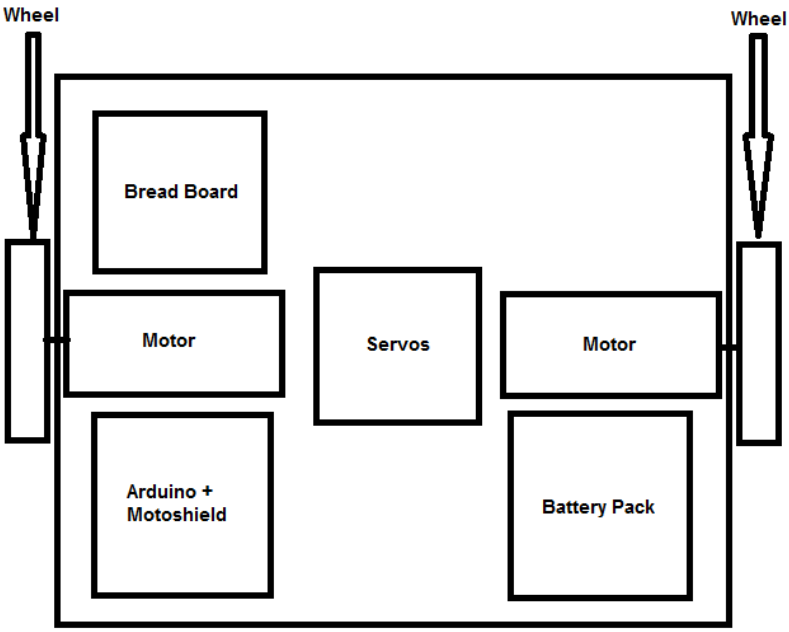

Figure 7: Top view of Robot Design.

The underside arrangement of the robot can be seen in Figure 8 and shows the two caster wheels and where the line sensing components will be located. The robot will be wholly supported by the two wheels attached to the motors and the caster wheels. The caster wheels are offset from the middle to balance and support the robot. This is the entire design behind the robot that I have created, but of course this is just the initial design and other improvements were added along the way.

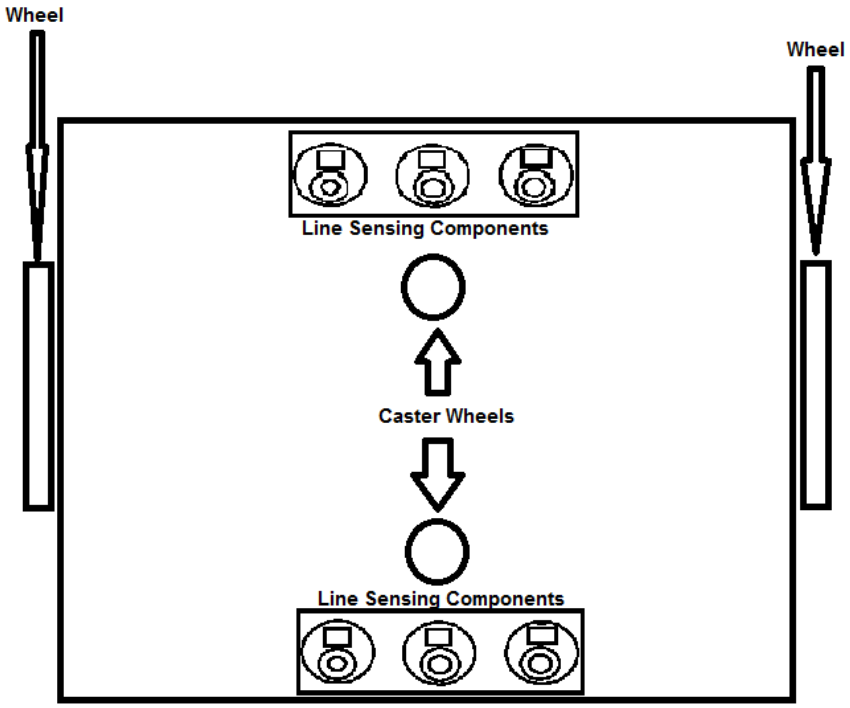

Figure 8: Bottom view of Robot Design.

2.4 Late Additions

Over the course this two quarter project, a couple of additions were made to the original design. One of the first additions was adding limit switches to the front and back of the robot to tell the robot to stop when pressed. Figure 9 shows where on the design these switches were added.

Another addition to the robot was an additional battery pack to power the arduino individually, and the original battery pack powering the motors and servos individually. Figure 10 shows the location of the battery packs on the robot.

The last addition to this robot was weight on top of the motors in order to give the attached wheels traction, I ended up using rolls of nickels.

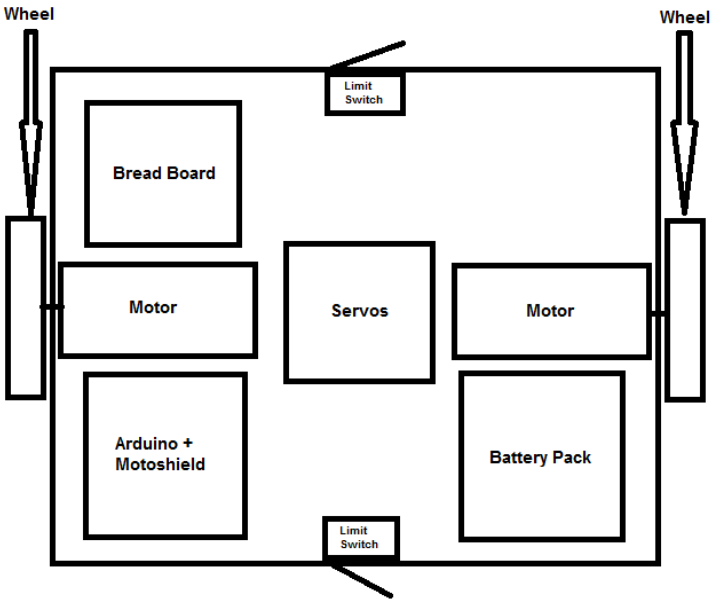

Figure 9: Top View of the robot with Limit Switches.

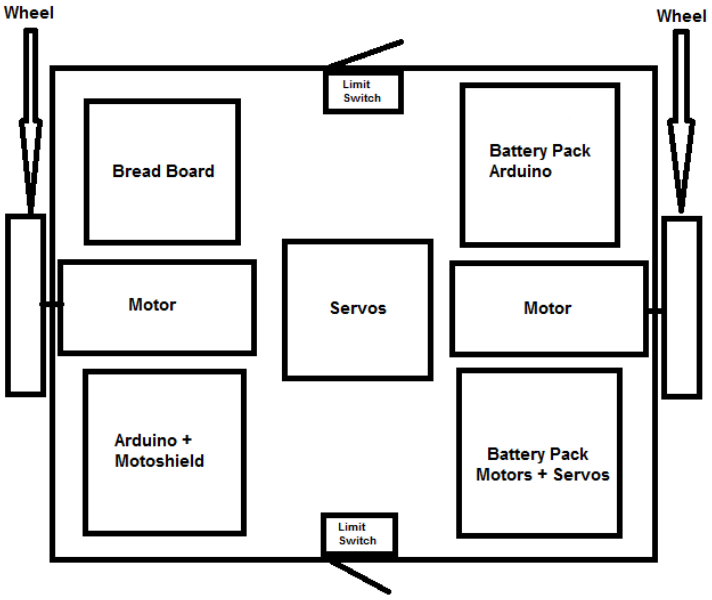

Figure 10: Complete Top View of the Robot.

3 Circuit Designs

There are a few different circuits for this robot. The first circuit is that of the LED which is a part of the line following mechanism. Figure 11 shows the circuit that all six line following LEDs share.

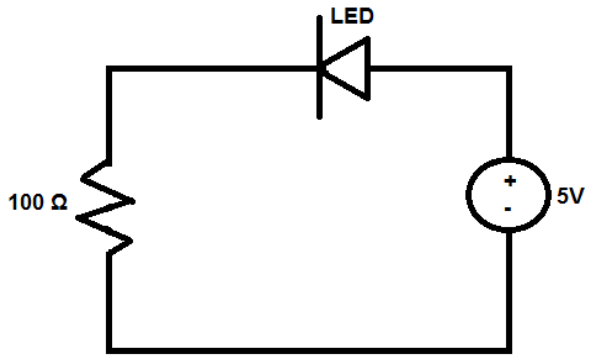

Figure 11: LED Circuit.

The pair circuit for the LED is the LDR circuit. Through the use of the LED and LDR circuits the line following components are made. Figure 12 shows the circuit that all six LDRs share.

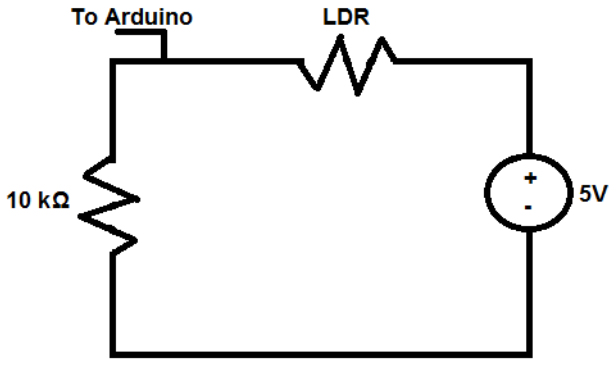

Figure 12: LDR Circuit.

The only other circuit within the robot is the limit switches circuit, which controls when the robot will stop. This circuit operates by having the switch either be at a zero voltage or a positive voltage according to whether it is pressed or not. Figure 13 shows the limit switch circuit.

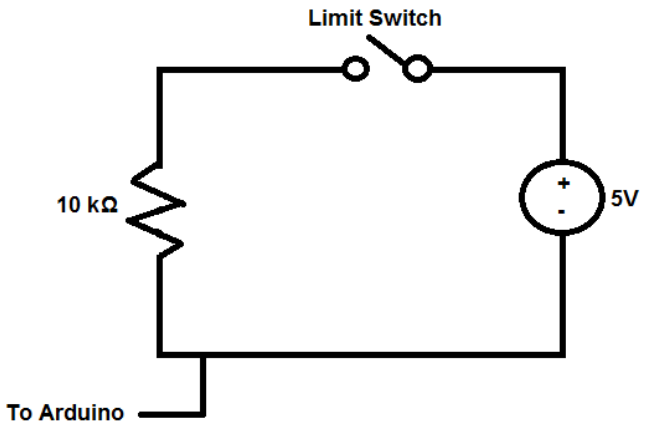

Figure 13: Limit Circuit.

4 Hardware

The high level block diagram for my robot can be seen in Figure 14. This high level view shows the basic connections between all the components of the robot.

The boxes that are labeled"Line Sensing Components" have 3 pairs of LDR and LED circuits, of which the three LDR circuits send individual data to the arduino through the motorshield. For simplicity, the data of all three of these LDR circuits was combined into one data line to the motorshield. The limit sensing boxes also contain their respective circuits which are also routed through the motorshield to the arduino.

The arduino battery pack first gives its power to the arduino and then it is broken out to the components that need it through the use of a 5 volt line to a breadboard with all the necessary circuits sharing from that supply. In the case of the servo and motor battery pack, the connections for those respective components are located on the motorshield of which the battery pack supplies the power. There are specific connections used for each of the components to the arduino and motorshield, this means that each unique connection to the arduino has a unique variable within the code. The code for the entire robot is located within the code section of the appendix. The code is commented and the variables are named in such a way that any reader with some micro controller background should be able to understand what is being done within each function.

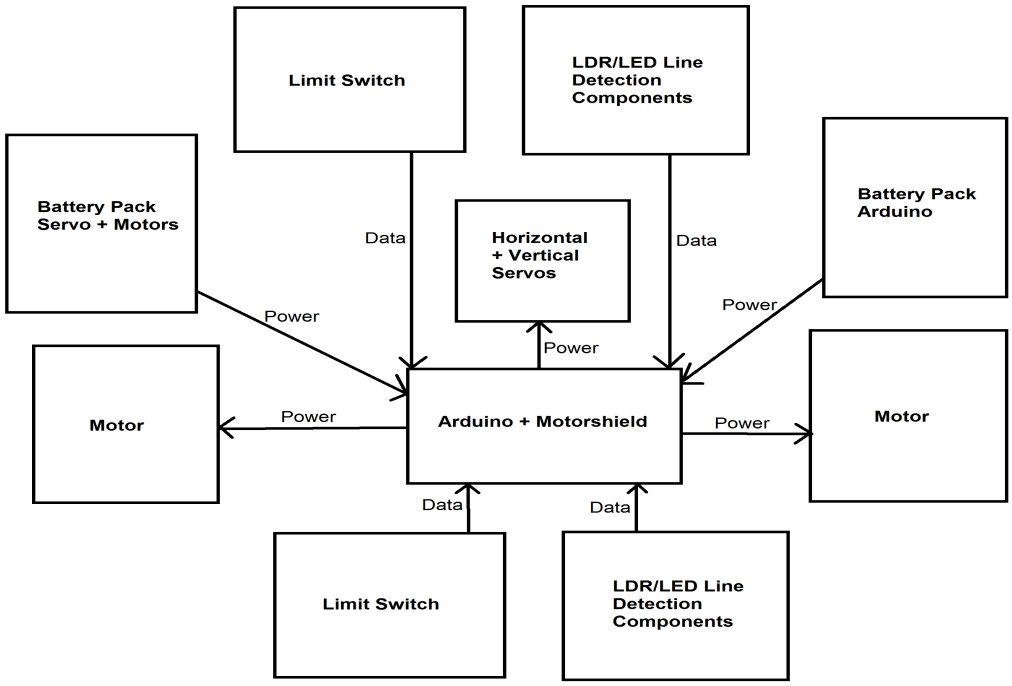

Figure 14: Block Diagram of All Hardware.

5 Use Cases

5.1 Line Following State

There are a couple states that my robot has when operating. One of the modes is line following. In this mode the robot tries to keep its sensors on the line according to three different states. The first state is when the robot is on the line and just needs to keep moving both motors at the same speed. This state can be seen in Figure 15.

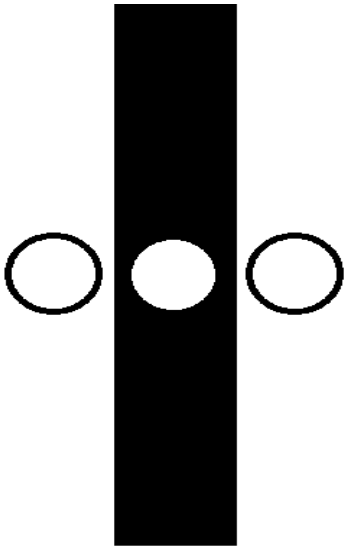

Figure 15: Line Following Sensors Are At Proper Locations.

The other line following states that the robot can be in are too far left and too far right. In each of these respective states the robot tries to correct its course by disabling one motor and keeping the other motor running in order to get back on the black line of the course. Figure 16 and Figure 17 show the states where the robot is off course and must be adjusted back into the good state shown in Figure 15.

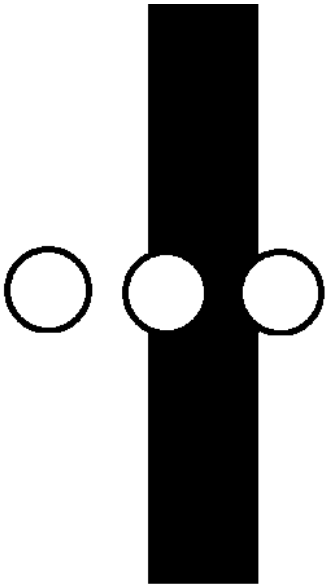

Figure 16: Line Following Sensors Are Too Far Left.

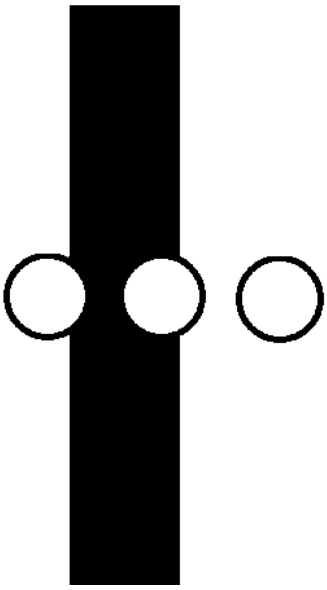

Figure 17: Line Following Sensors Are Too Far Right.

5.2 Ring Pick Up and Drop Off States

The other states that this robot can enter is when the robot is picking up and dropping off rings. In the pick up ring state, the robot should look like Figure 18 which is showing the point of view of the ring grabbing arm. From this state the robot does a pick up maneuver as shown in Figure 19 and Figure 20. After these two stages the robot reverses its course and swings the center shaft around for the drop off procedure.

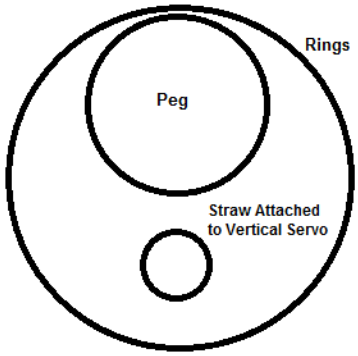

Figure 18: Servo Level View of Initial Pick Up State.

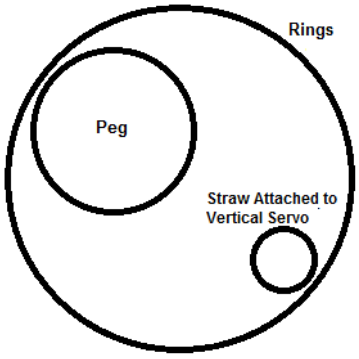

Figure 19: Horizontal Servo Swivels To Obtain Rings.

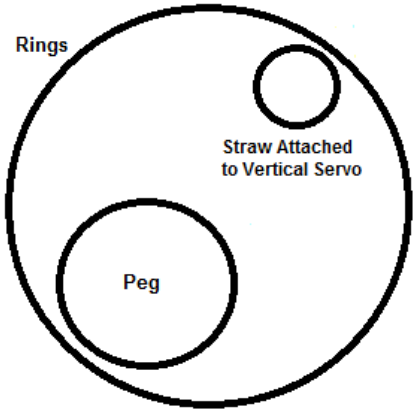

Figure 20: Vertical Servo Moves Up To Obtain Rings.

After grabbing the rings and moving across the surface of the course the robot will encounter its drop off states. As shown in Figure 21 the robot has all the rings over the bottom drop off peg. The next two states as shown in Figure 22 and Figure 23 show how the robot will drop off the rings.

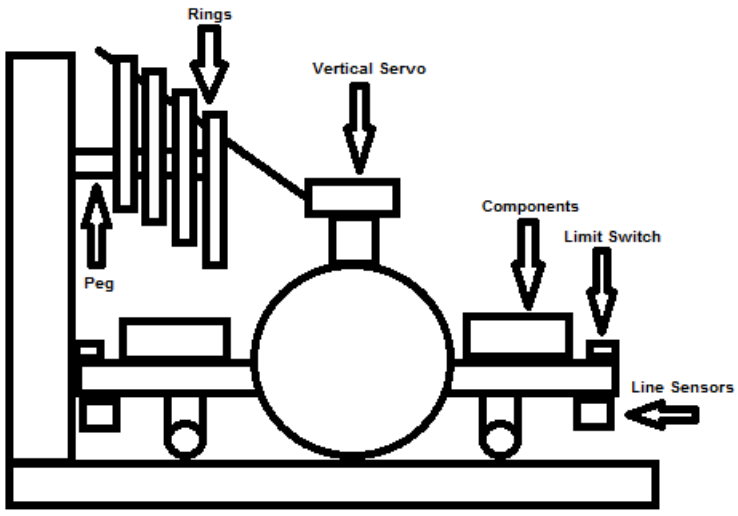

Figure 21: Side View of Initial Drop Off State.

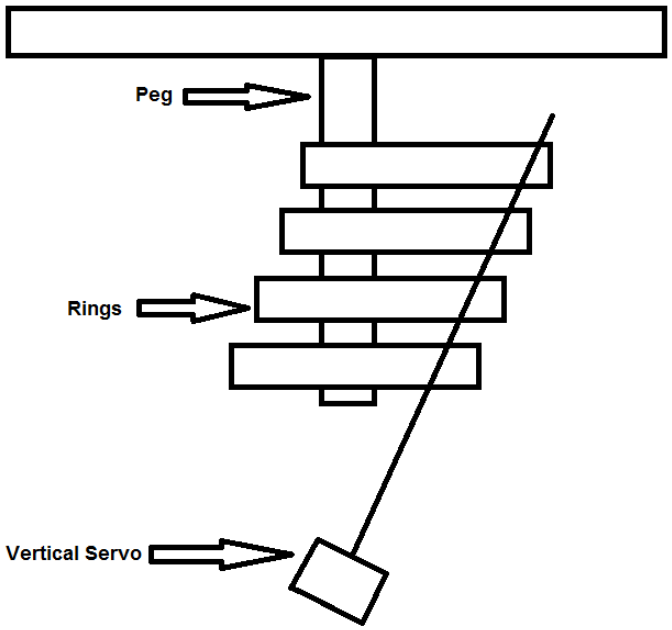

Figure 22: Horizontal Servo Swivels To Push Rings Against Peg.

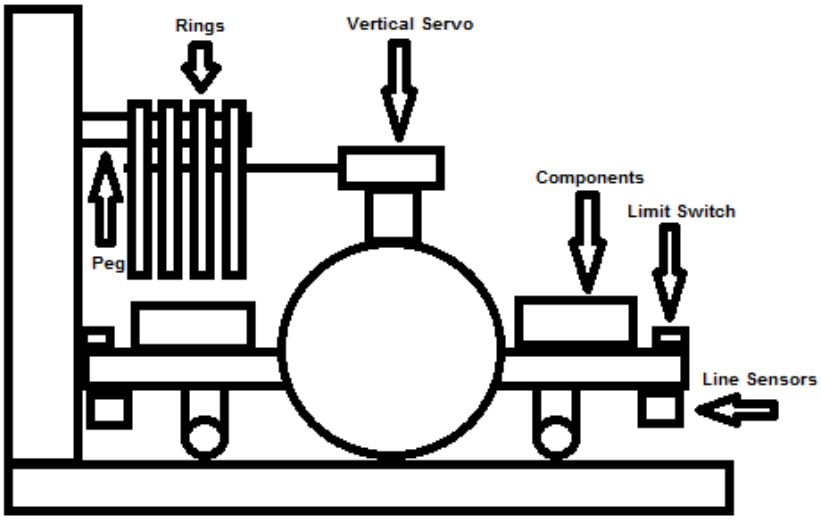

Figure 23: Vertical Servo Moves Down To Drop Rings Onto Peg.

There are some basic processes the robot is going through in the above three states. The first state is arrival, Figure 21 displays the robot perpendicular with the drop end of the wall with the rings positioned above the drop off peg. The robot then enters a swiveling state which moves the ring grabbing arm over to the side until the farthest ring out is against the peg, this is shown in Figure 22. Lastly, the robot lowers its ring grabbing arm and swivels back under the peg effectively dropping the rings onto the peg as shown in Figure 23. After this final maneuver the robot begins following the line back to the receiving end of the course.

All the above states make up the entirety of the operations the robot can perform while on the course.

6 Software

While the entirety of the software used is in the appendix at the end of the document, it would be best to discuss the way the software is structured. Because the code is being run on an arduino platform, the code was written in a way that allowed for continuous operation, or for as long as the arduino is powered. All of the variables used within the software have been declared as globals so that functions need not pass around data.

There are two functions every arduino program needs in order to operate and these are setup and loop. In setup, the initial states of the serial line, analog to digital converter, motors, servos, and a few other things are initialized. In addition to these initializations, the line sensing components are calibrated for the line that

they should be following. The sensors are given data from the LDR/LED combo to use for their calculations in determining the robot's position on the line. It is important in this setup phase that the robot has both ends of its chassis centered above the line it will be following with the sensors on the bottom of the robot positioned as shown in Figure 15.

Within the loop function the robot determines its direction, either towards the receiving end or the drop end, and beings following the line according to this choice. This section of software is accessed at start up and whenever the robot has dropped off or picked up rings.

All other functions within the software are tailored for either the robot picking up rings or dropping off these rings. Because of this setup there are duplicates of almost every function that are tailored for either of the two states. For example, there is a line follow state for going towards the receiving side and another for going to the drop off side. These two states are needed because of the direction the motors need to spin when going in different directions and also to allow for easier transitions to other side specific states.

Most every function has one specific task that it must perform, this helps with modularity in the code. With certain tasks encased in just one function, changes can be made very easily to certain actions without affecting other parts of the code. One last thing to mention about the software written for this robot is that some helper libraries/functions were used to control the motors and servos more easily.

To see the specifics of the code written for this robot see the code section within the appendix.

7 Completed Robot

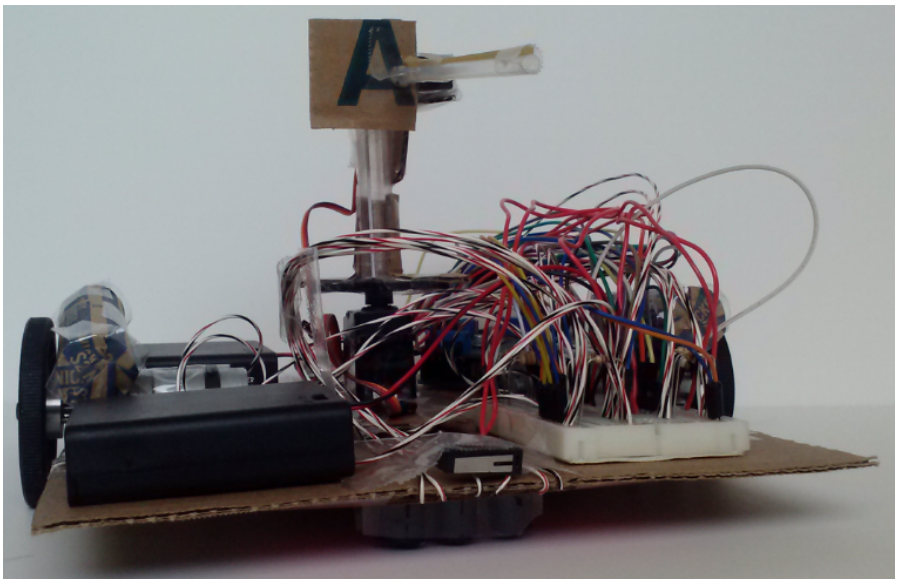

Figure 24: Front View of Complete Robot.

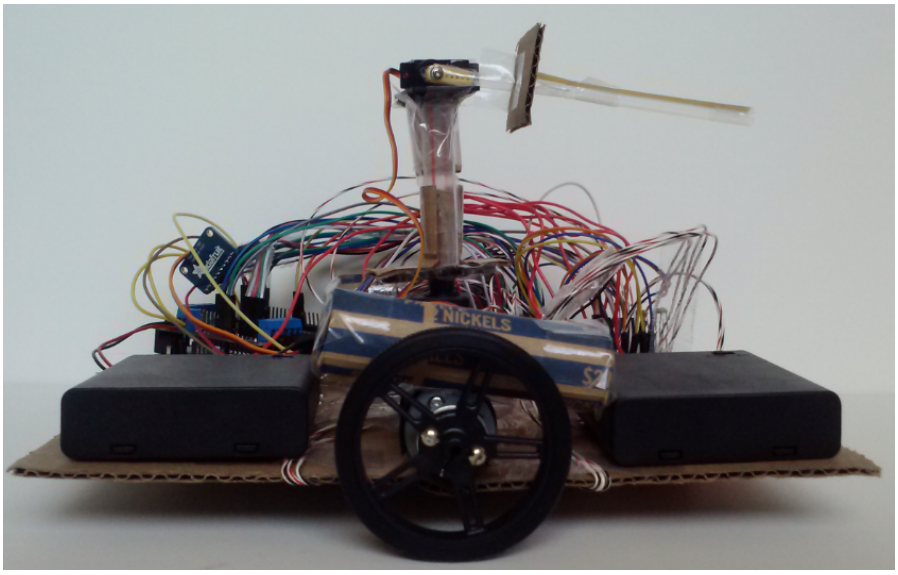

Figure 25: Side View of Complete Robot.

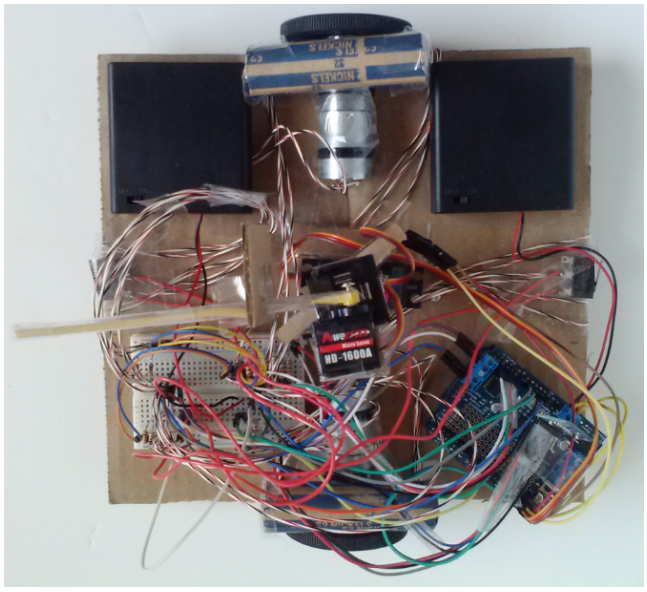

Figure 26: Top View of Complete Robot.

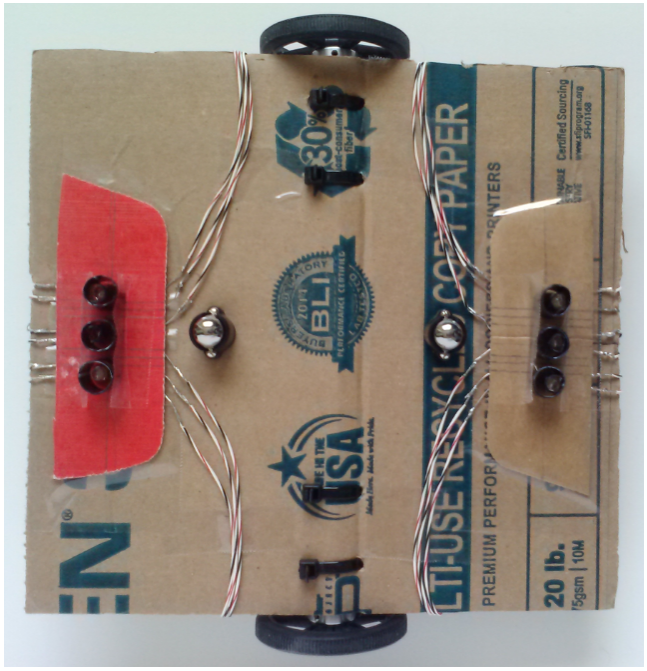

Figure 27: Bottom View of Complete Robot.

8 Lessons Learned

There are many things that I have learned over the course of this project. One of the biggest things that I have learned is that cardboard is not a suitable chassis for any kind of small to medium sized or larger robot. The weight of all the components on the cardboard caused significant bend in my robot that I feel may have effected the operation of the robot. Certain components were also effected by the bendable nature of the cardboard. For instance, the servos in the center of the robot were not sitting flush to the cardboard and had some lean to them that had to be compensated for during testing of the robot.

Weight distribution on the robot was another concern that I had to consider over the course of this project. Due to the cardboard chassis and the amount of items being placed on the robot itself there were some weighting issues that arose. Because of uneven weight on each motor, the rotations of each motor had to be tailored to match each other during operation. However, the added weight of the rings during operation of the robot sometimes caused the tailored rotation speeds to become ineffective.

Another thing that I learned was how inaccurate the LDR/LED combos were in detecting the black course line. I would have chosen an actual component that did this for me rather than build my own if I had known the inaccuracy when I designed the robot.

Wire management was another thing of interest during the construction of this robot. As more and more things were added to the robot, the amount of wires used became a mess to the point where it was hard to determine where any of the wires

went. Buying shorter cables or cutting my own more precisely would have made the presentation and mechanical troubleshooting aspects of the robot much easier.

Overall this project was a great experience in learning how to build a robot for a specific task. I had never built a robot before and it was a good feeling to see it do what it was designed to do during the competition. Even though there were some flaws in the final implementation, the basic functionality was there and I am proud of what I have designed and built.

9 Appendix

9.1 Bill of Materials

Description	Store	Cost	Running Total
Arduino Uno (1)	Arduino Store	$20.00	$20.00
Adafruit V2 Motorshield (1)	Adafruit Store	$19.95	$39.95
White LEDs (6)	Sparkfun Store	6 * $.95	$45.65
Photoresistors (6)	Sparkfun Store	6 * $1.50	$54.65
Motors (2)	Pololu Store	2 * $19.95	$94.55
Micro Servo (1)	Pololu Store	$5.95	$100.50
Standard Servo (1)	Pololu Store	$9.95	$110.45
4xAA Battery Pack (2)	Radioshack	2 * $2.49	$115.43
Castor Wheel (2)	Pololu Store	2 * $1.99	$119.41
Motor Mount Pair (1)	Pololu Store	$6.95	$126.36
12-Bit ADC (1)	Adafruit Store	$9.95	$136.31
Zip Ties Pack	Amazon Store	$2.06	$138.37
Limit Switch (2)	Pololu Store	2 * $.75	$139.87
Batteries AA 24 Pack (1)	Sparkfun Store	$10.24	$150.11
100Ω Resistors (6)	Sparkfun Store	6 * $.25	$151.61
10kΩ Resistors (6)	Sparkfun Store	6 * $.25	$153.11
MF/MM/FF Wires (Multiple)	Amazon Store	$5.29	$158.40
Small Straw (1)	Anywhere	Varies	$158.40
Large Straw (1)	Anywhere	Varies	$158.40
Rubber Band (1)	Anywhere	Varies	$158.40
Computer (1)	Use Personal	Varies	$158.40
CardBoard 10”x10” (1)	Anywhere	Varies	$158.40
Tape Roll (1)	Anywhere	Varies	$158.40

Table 1: Bill of Materials

The above Bill of Materials represents all items needed to build the robot described. Some items are so basic, like cardboard, that the price is listed as “Varies” because I am assuming the reader already has some of these items in their possession, or can easily get the item(s) at a low or free price.

9.2 Code

```
/*
Project: Senior Project
Purpose: Roborodentia 2015
Author: Travis Stuever
Advisor: John Seng
*/

/*
Layout
This end Rec
_________________________
|Battery|Bumper |Arduino|
|_______|_______|_______|
|Motor L|Shaft  |Motor R|
|_______|_______|_______|
|Battery|Bumper |Board  |
|_______|_______|_______|

This end Drop
*/

/*Includes*/
```

```
#include <Wire.h>
#include <Adafruit_MotorShield.h>
#include "utility/Adafruit_PWMServoDriver.h"
#include <Servo.h>
#include <Adafruit_ADS1015.h>

/*Globals*/
const int Aone = A0; //LDR define
const int Atwo = A1; //LDR define
const int Athree = A2; //LDR define
const int Afour = A3; //LDR define
int test = 0; //Used just for testing
int RecOneMid = 0; //Correction values
int RecTwoMid = 0;
int RecThrMid = 0;
int DropOneMid = 0;
int DropTwoMid = 0;
int DropThrMid = 0;
int DropThree = 0; //Analog Reading
int DropTwo = 0; //Analog Reading
int DropOne = 0; //Analog Reading
int RecThree = 0; //Analog Reading
int RecOne = 0; //Analog Reading
```

```
int RecTwo = 0; //Analog Reading
int Dir = 0; //1 for Drop end, -1 for receive end
int SPos = 0; //Shaft Position
int KPos = 0; //Keeper Position
int LinesPassed = 0; //Count solid lines passed
int RecEdgePin = 8; //Digital pin to connect the Receiving bumper sensor
int DropEdgePin = 7; //Digital pin to connect the drop bumper sensor
int RecEdge = HIGH; //Turns to 1 if Receiving edge is pressed
int DropEdge = HIGH; //Turns to 1 if Drop edge is pressed
Servo Shaft; //Name Implied, pin 10
Servo Keeper; //Name Implied, pin 9
Adafruit_MotorShield AFMS = Adafruit_MotorShield(); //MotorShield
Adafruit_DCMotor *RightMotor = AFMS.getMotor(4); //Right Motor
Adafruit_DCMotor *LeftMotor = AFMS.getMotor(3); //Left Motor
Adafruit_ADS1015 ads; //ADC

/*Everything Else*/
void Grab(void) { //Grab rings
  for(int i = SPos; i < 19; i++) { //Move Right
     Shaft.write(i);
     delay(15);
  }
  SPos = 18;
```

```
  delay(500);
  for(int i = KPos; i < 22; i++) { //Move up
     Keeper.write(i);
     delay(15);
  }
  KPos = 21;
  delay(100);
  Dir = 1;
}

void Drop(void) { //Drop rings
  for(int i = SPos; i < 181; i++) { //Move Right
     Shaft.write(i);
     delay(15);
  }
  SPos = 180;
  delay(1000);
  for(int i = KPos; i > -1; i--) { //Move down
     Keeper.write(i);
     delay(15);
  }
  KPos = 0;
  delay(1000);
```

```
  for(int i = SPos; i > 168; i--) { //Move back to center
     Shaft.write(i);
     delay(15);
  }
  SPos = 167;
  Dir = -1;
}

void SwingRec(void) { //Moves shaft to correct receive position
  for(int i = SPos; i > -1; i--) {
     Shaft.write(i);
     delay(15);
  }
  SPos = 0;
  delay(100);
  TipRec();
}

void SwingDrop(void) { //Moves shaft to correct drop position
  for(int i = SPos; i < 168; i++) {
     Shaft.write(i);
     delay(15);
  }
```

```
  SPos = 167;
  delay(1000);
  TipDrop();
}

void TipRec(void) { //Moves Keeper to correct initial position
  for(int i = KPos; i < 5; i++) {
     Keeper.write(i);
     delay(15);
  }
  KPos = 4;
  delay(100);
}

void TipDrop(void) { //Moves Keeper to correct initial position
  for(int i = KPos; i < 40; i++) {
     Keeper.write(i);
     delay(15);
  }
  KPos = 39;
  delay(100);
}
```

```
void calibrate(void) {
  delay(1000);
  RecEdge = digitalRead(RecEdgePin);
  DropEdge = digitalRead(DropEdgePin);
  while(RecEdge == HIGH && DropEdge == HIGH) {
    RecEdge = digitalRead(RecEdgePin);
    DropEdge = digitalRead(DropEdgePin);
    Serial.println("Waiting for calibration");

    Serial.print("D1: "); //One on Drop
    DropOne = analogRead(Athree);
    Serial.println(DropOne);

    Serial.print("D2: "); //Two on Drop
    DropTwo = analogRead(Atwo);
    Serial.println(DropTwo);

    Serial.print("D3: "); //Three on Drop
    DropThree = analogRead(Aone);
    Serial.println(DropThree);

    Serial.print("R1: "); //One on Receive
    RecOne = ads.readADC_SingleEnded(0);
```

```
    Serial.println(RecOne);

    Serial.print("R2: "); //Two on Receive
    RecTwo = ads.readADC_SingleEnded(1);
    Serial.println(RecTwo);

    Serial.print("R3: "); //Three on Receive
    RecThree = analogRead(Afour);
    Serial.println(RecThree);
  }
  RecOneMid = ads.readADC_SingleEnded(0) - 10;
  RecTwoMid = ads.readADC_SingleEnded(1) + 10;
  RecThrMid = analogRead(Afour) - 10;
  DropOneMid = analogRead(Athree) - 10;
  DropTwoMid = analogRead(Atwo) + 10;
  DropThrMid = analogRead(Aone) - 10;
  RecEdge = HIGH;
  DropEdge = HIGH;
  delay(1000);
}

void FollowRec(void) { //Follow line towards drop end
  RightMotor->setSpeed(155);
```

```
LeftMotor->setSpeed(160);
while(RecEdge == HIGH) {
  RecOne = ads.readADC_SingleEnded(0);
  RecTwo = ads.readADC_SingleEnded(1);
  RecThree = analogRead(Afour);
  RecEdge = digitalRead(RecEdgePin); //Stops while loop if bumped

  if(RecOne <= RecOneMid && RecThree <= RecThrMid) { //pass line
    RightMotor->run(FORWARD);
    LeftMotor->run(BACKWARD);
    delay(15);
    RightMotor->run(RELEASE);
    LeftMotor->run(RELEASE);
  } else if(RecOne <= RecOneMid) {
    LeftMotor->run(BACKWARD);
    delay(15);
    LeftMotor->run(RELEASE);
  } else if(RecThree <= RecThrMid) {
    RightMotor->run(FORWARD);
    delay(15);
    RightMotor->run(RELEASE);
  } else {
    RightMotor->run(FORWARD);
```

```
      LeftMotor->run(BACKWARD);
      delay(15);
      RightMotor->run(RELEASE);
      LeftMotor->run(RELEASE);
    }

    //Swing the shaft to the correct spot and lower or heighten the keeper
    if(LinesPassed == 100) {
       delay(1000);
    }
    LinesPassed++;
  }
  RecEdge = HIGH;
}

void FollowDrop(void) { //Follow line towards receiving end
  RightMotor->setSpeed(155);
  LeftMotor->setSpeed(160);
  while(DropEdge == HIGH) {
    DropOne = analogRead(Athree);
    DropTwo = analogRead(Atwo);
    DropThree = analogRead(Aone);
    RecOne = ads.readADC_SingleEnded(0);
```

```
RecTwo = ads.readADC_SingleEnded(1);
RecThree = analogRead(Afour);
DropEdge = digitalRead(DropEdgePin); //Stops while loop if bumped

if(DropOne <= DropOneMid && DropThree <= DropThrMid) { //pass line
  RightMotor->run(BACKWARD);
  LeftMotor->run(FORWARD);
  delay(15);
  RightMotor->run(RELEASE);
  LeftMotor->run(RELEASE);
} else if(DropOne <= DropOneMid) {
  LeftMotor->run(FORWARD);
  delay(15);
  LeftMotor->run(RELEASE);
} else if(DropThree <= DropThrMid) {
  RightMotor->run(BACKWARD);
  delay(15);
  RightMotor->run(RELEASE);
} else {
  RightMotor->run(BACKWARD);
  LeftMotor->run(FORWARD);
  delay(15);
  RightMotor->run(RELEASE);
```

```
      LeftMotor->run(RELEASE);
    }

    //Swing the shaft to the correct spot and lower or heighten the keeper
    if(LinesPassed == 100) {
       SwingDrop();
       delay(1000);
    }
    LinesPassed++;
  }
  DropEdge = HIGH;
}

void setup(void) { //Sets everything up
  Serial.begin(9600);
  ads.begin();
  AFMS.begin();
  RightMotor->setSpeed(120);
  LeftMotor->setSpeed(120);
  pinMode(RecEdgePin, INPUT);
  pinMode(DropEdgePin, INPUT);
  Shaft.attach(10);
  delay(1000);
```

```
  Keeper.attach(9);
  delay(1000);
  Keeper.write(0);
  delay(1000);
  calibrate();
}

void loop(void) {
  if(Dir == 1) { //Drop
    FollowDrop();
    delay(1000);
    Drop();
    delay(1000);
  } else { //Receive
    FollowRec();
    delay(1000);
    Grab();
    delay(1000);
  }
  LinesPassed = 0;
}
```