Mr. Robot: A Roborodentia Contestant

by

Stephen Berry

Jamie Nease

Senior Project

June 2011

Computer Engineering

California Polytechnic State University

Table of Contents

I. Introduction

Roborodentia is an annual robotics competition held at California Polytechnic State University, San Luis Obispo. Each year, students and alumni create small robots to perform different tasks using ping pong balls. This year's competition involved two robots going head to head in a 6' x 8' court. Each robot retrieved ping pong balls from two racks on either side of the court, and shot them into the opponent's net. Some balls were worth more points than others, and the robot with the highest score at the end of the three minute match won. The robots had to comply with the specifications outlined in the rules, which we will go into more detail about in the *Rules and Specifications* section.

As fourth year Computer Engineering students, neither of us had any experience in robotics prior to the competition, but had always been curious and interested. When the idea was proposed as a potential senior project, we decided it would be a great way to combine our software and hardware skills to produce a project that tested our computer engineering knowledge.

In the end, we produced a simple yet very durable robot which performed well in the competition. His name was Mr. Robot, and he exceeded our initial expectations. Out of a possible 108 points to be scored in a match, our robot consistently scored about 70 points. To give some perspective to the score, the first place team scored about 95 points in the final match, and the top four teams ranged from 75-95 points per match. We won two matches and lost two matches, though they were very close games.

After building the robot, we have learned that robotics involves a lot of mechanical engineering and physical labor. Building the physical board and components took a lot of

measurements and precision. We now have many ideas on how to make future robots better after learning the hard way how to assemble certain components.

In this report, we will talk about the rules and specifications of the competition as they are very important in our design considerations. Then we will discuss our initial design, the steps leading to our final design, the final physical and software design, the materials used, the process used to build the robot, what we would have done differently knowing what we know now, and finally we will conclude the paper.

II. Rules and Specifications

The competition had various rules to ensure a fair and exciting match, along with adding a challenge to the robot designers. The official competition court is shown below in *Figure 1*. Here you can see the racks in which the balls must be retrieved from, as well as the general size of everything. The orange balls in the troughs are worth 3 points. In the center of the court lays the super ball, which is worth an increasing amount of points as the competition carries on. In the beginning of the match the super ball is worth 3 points, and in the last 30 seconds its worth 15 points. No robot could cross the halfway point of the middle barrier, except in the 3" area surrounding the super ball.

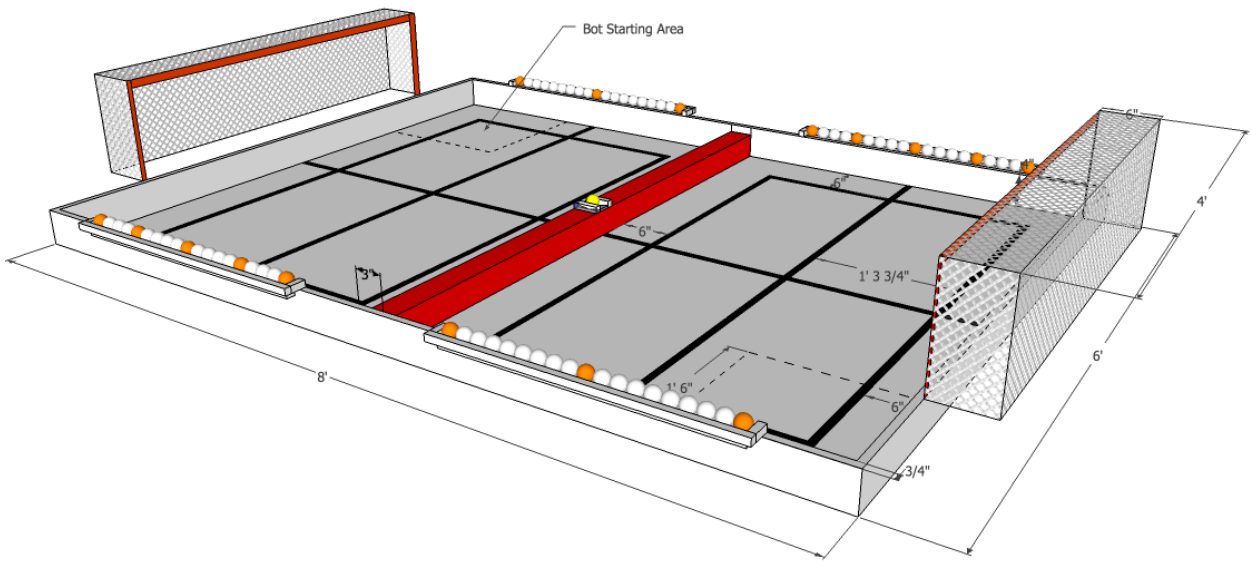

Figure 1. Official Competition Court

The robots must comply with size and assembly specifications as well. Each robot must

fit within a 12” x 12” x 12” box at the beginning of the match, but may autonomously expand and/or detach after it begins. The robots must be completely autonomous, and must turn on all the way with the press of one button at the beginning of the match. For robots with multiple disassembling parts, all parts must be continuously moving throughout the match.

The competition regulations state that during a match there is a possibility of two trough reloads. The left trough is reloaded after all balls have been removed from the trough. The right trough is reloaded only after all of the balls have been removed from it on the first round, and all of the balls have been removed from the left trough after its first reload. The final relevant regulation is that at any point during a match, a robot may block no more than 40% of the net.

III. Initial Design

Now that the competition has been described in detail, we can discuss our initial thoughts on the design of our robot. The main components of each robot were the retrieval system, the drive system, and the shooting mechanism. When we first started brainstorming, we were thinking of what the best possible robot could be. We wanted multiple robots that autonomously expanded, including a goalie-bot to block the opponent's shots. We looked at previous year's winning robots to get some inspiration.

For the drive system, we were looking into using omniwheels, which had been used by previous winners. Omniwheels are multidirectional, allowing the wheel to be powered forward, but, at the same time, there are small rollers on the edge of the wheel allowing it to move laterally. At first, we wanted to place them on the four corners of the robot similar to the design of a car, but with two wheels on opposite corners from each other facing perpendicular to the other two wheels.

Initially we didn't have many ideas for the retrieval system, other than what had been used in previous years. These systems used a rotating servo with a brush attached to sweep in balls. Unfortunately, this was the first time this particular rack system was being used in the competition. This meant we were going to have to get creative and come up with a new and clever idea to get the balls.

After having done research on previous robots, we noticed that most everyone uses two wheels for the shooter. This seemed to work pretty consistently, but we thought it might be interesting to try using compressed air instead to be original and possibly gain an advantage. We knew we'd need to make prototypes and do some research to decide which the best method was.

The two main ideas we considered for ball collection were to create a hopper to store

collected balls and then shoot from a single position on the court or to shoot the balls as they were being collected. Using a hopper would allow us to avoid dealing with shooting angles, but it would also require us to have enough space on the robot for an entire rack of balls. Shooting the balls as they are collected eliminates the need to store the balls, freeing up space for other functions such as driving or collecting. However, this method requires an aiming mechanism as each ball would be shot from a different location on the court. To accomplish this, we thought about using a rotating platform with the shooter mounted on top. This would allow the shooter to effectively change its angle as the robot moved.

IV. Intermediate Design

The one piece of advice that we got from everyone we asked was to keep it simple. Simplicity would lead to a more consistent and durable robot, as well as be easier to implement. Because of this, we abandoned our idea of having multiple robots. We decided to focus on getting a single robot to do everything it needed to do well, without having to deal with the complexity involved with detachable robots and multiple drive systems.

Along the same lines, we modified our idea of having the shooter mounted on top of a rotating platform. This would require a much more powerful servo and take up more space on the robot. Instead, we decided to attach a small piece of wood to a servo that could angle the shots dynamically as the robot moved.

After constructing a prototype for the shooter made out of wood, we realized how efficient it would be to use two un-geared DC motors with wheels attached to shoot the balls as seen in *Figure 2*. During our tests, the shooter was able to propel a ping pong ball roughly twice as far as the length of the court, more power than was needed and with consistent accuracy. At this point, we decided not to pursue a compressed air shooter due to the impressive results we obtained.

Figure 2. Shooter

When debating ball storage versus an immediate shot we had to take our entire design into consideration. After deciding on the piece of wood to angle our shot, our robot then had the ability to shoot from almost anywhere on the court with some code to aim it. Therefore we decided against the hopper idea and instead pursued a method of directing the balls toward the shooter immediately after being collected. This simply meant that we needed a ramp from each side of the robot leading to the front where they could be shot from. A gradual downward slope throughout the ramp was needed in order to ensure the balls would not get stuck.

When finalizing our design, we still had not given much thought to the ball retrieval system. The rack that held the balls was different than any of the previous years, so we knew we would have to think of something clever to get them. Since the robot had to move and collect

balls without disturbing the rest of the balls on the rack, we thought a system similar to the Hungry Hungry Hippos game would work well. We would have some sort of mechanism to reach out over the rack and scoop back a certain number of balls, while leaving the rest in place. This meant we would need an opening in the ramp big enough to hold those balls on the robot as they slid down to the shooter.

As we planned the design of Mr. Robot, we were mostly thinking about the physical side. However, we did consider a few software techniques that involved physical pieces. There are essentially three ways that the robot can move through the course efficiently. They could either use timers, switches, or sensors. In line with keeping our design simple, we thought timers would be the easiest method to employ. We dealt with sensors in previous classes and knew that they could be difficult to interface with. However, timers turned out to be more difficult to use than we had thought. As the batteries lost charge, the wheels would move slower, resulting in unreliable timing for the robot moving across the court. To solve this, we mounted lever-switches to all sides of the robot that could detect when a wall was hit. We left the timers active, except with a longer wait time, as a back-up in case the switches failed.

V. Final Design

Our final design consists of a 12” x 12” piece of expanded PVC as the baseboard, with four omniwheels on the bottom. The wheels are centered on their respective sides, two facing one direction, and the other two facing perpendicular as can be seen in *Figure 3*. The wheels are glued to small full-rotational servos, which didn’t prove sturdy enough under pressure. We later added four L-brackets screwed to the base board and to the wheel to give some support.

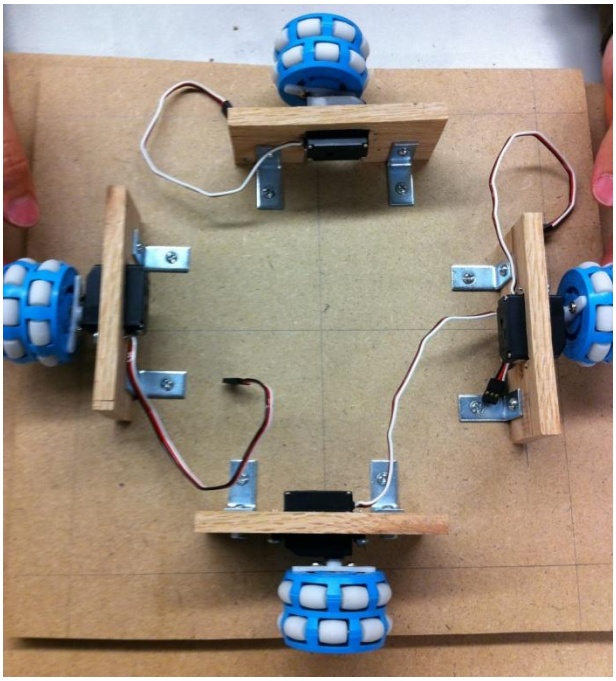

Figure 3. Initial Drive System

The shooter consists of a piece of expanded PVC cut out to make room for two un-geared DC motors. Foam wheels are attached to these motors slightly closer together than the diameter of the ping pong balls to ensure that they get sucked through. The board holding them up stood almost vertical, but was tilted slightly backward to allow lift in the balls being shot. It was also angled slightly to the right, so that the shot coming from the left side of the court didn’t need any

deflection to be aimed correctly. We soldered a circuit to the back of the shooter to enable the motors to be turned on and off using the Polybot board. This is a simple circuit that consists of a power source for the motors, two diodes to protect against reverse current, and a relay that is controlled by the Polybot board to complete the circuit, effectively acting as the on-off switch. The circuit can be seen in *Figure 4*.

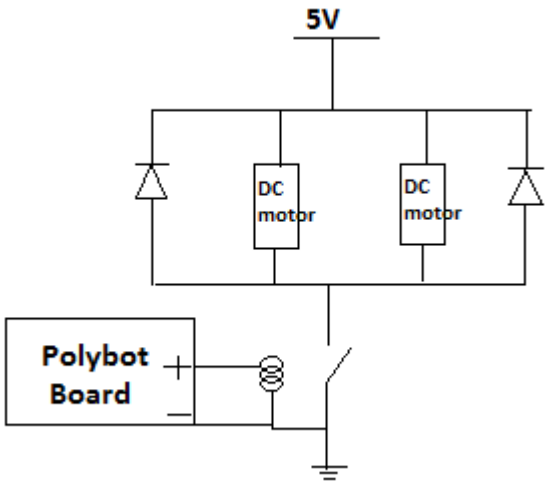

Figure 4. Circuit Diagram

Our aiming device that deflects shots coming from the right side of the court is made of a piece of wood, which is mounted to a small, 90-degree non-continuous servo. The servo stood out of the base board, and the piece of wood was glued to the top. The device was angled to the right at the beginning of the match to ensure it would not interfere with shots fired from the left side of the court, as the shooter was already positioned for shooting correctly from that side. When the robot moved to the right side of the court, the device was positioned to deflect shots from that side into the goal. Because of this deflection, the balls would lose some of their energy and not always make it into the goal, however, the majority of the time they would either make it

or bounce in so we decided to stick with the system.

The ball retrieval system ended up being two metal rakes attached to small lever arms which are attached to continuous rotation servos. These rakes were balanced on a piece of wood so that the rotation of the servos and the small lever arms would cause the rakes to mimic the motion of the hippos in the Hungry Hungry Hippos game. This motion can be described as having the rakes move up and out, then down and back, sweeping the balls inward with the motion. We designed the rakes themselves to be wide enough to collect two balls at once in order to improve speed and efficiency.

A simple ramp was used to connect the systems together. As balls are collected, they are placed on a piece of sheet metal that that leads to the shooter from either side. The ramp is sloped downward so that gravity takes over once the balls have been retrieved. There are walls on both sides of the ramp, made of either Expanded PVC or wood that prevent the balls from getting off course. You can see the ramp and walls in *Figure 5*.

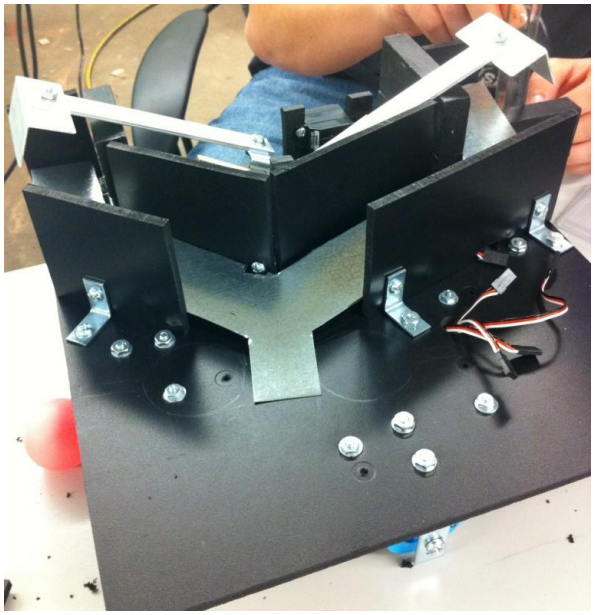

Figure 5. Ramp

The Polybot board is mounted near the front of the robot. During testing, we noticed that the board would reset if it was hit by a ball. The same was true of the battery packs that were initially mounted in the front. To fix this issue, we moved the battery packs to above the ramp and behind the shooter so they would be protected. We then enclosed the Polybot board with walls made of Expanded PVC which prevented any contact with balls. The final design of the robot can be seen in *Figure 6*.

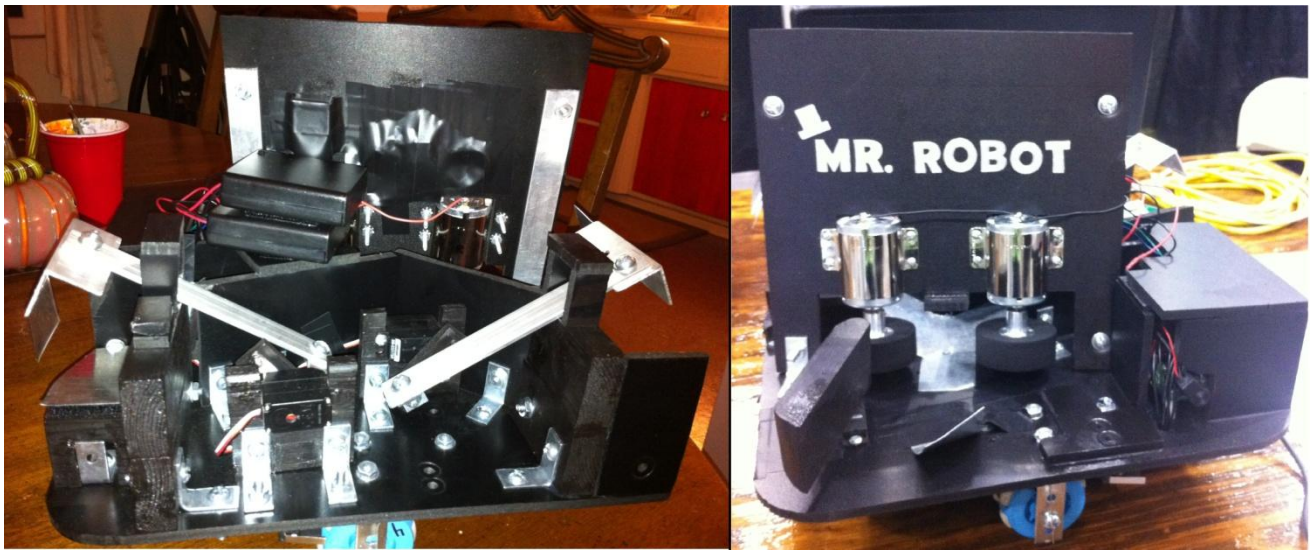

Figure 6. Back and Front of Mr. Robot

The wiring for our robot was relatively simple. The two servos attached to the rakes for ball retrieval had their wires routed beneath the ramp to the Polybot board, avoiding any potential interference. All of the other servos were even easier to wire due to their location on the bottom of the robot. We drilled a hole next to the Polybot board where all other servo wires could be routed from the bottom up to the Polybot board. We used electrical tape to keep the wires in place under the robot. The shooter is mounted next to the Polybot board and the wires

are located above. Our control circuit is mounted on the back of the shooter board so the wires could easily connect to the Polybot board. The lever switches were a last minute add-on and were mounted on the bottom of the base, centered on each side near the wheels. Luckily, the wiring for the switches was able to be routed exactly the same as the servos for the wheels.

The software was written in C using given libraries for the Polybot board which uses an ATMega 644 microcontroller. The software algorithm can be seen in the flowchart in *Figure 7*. The full software code base can be seen in *Appendix A*.

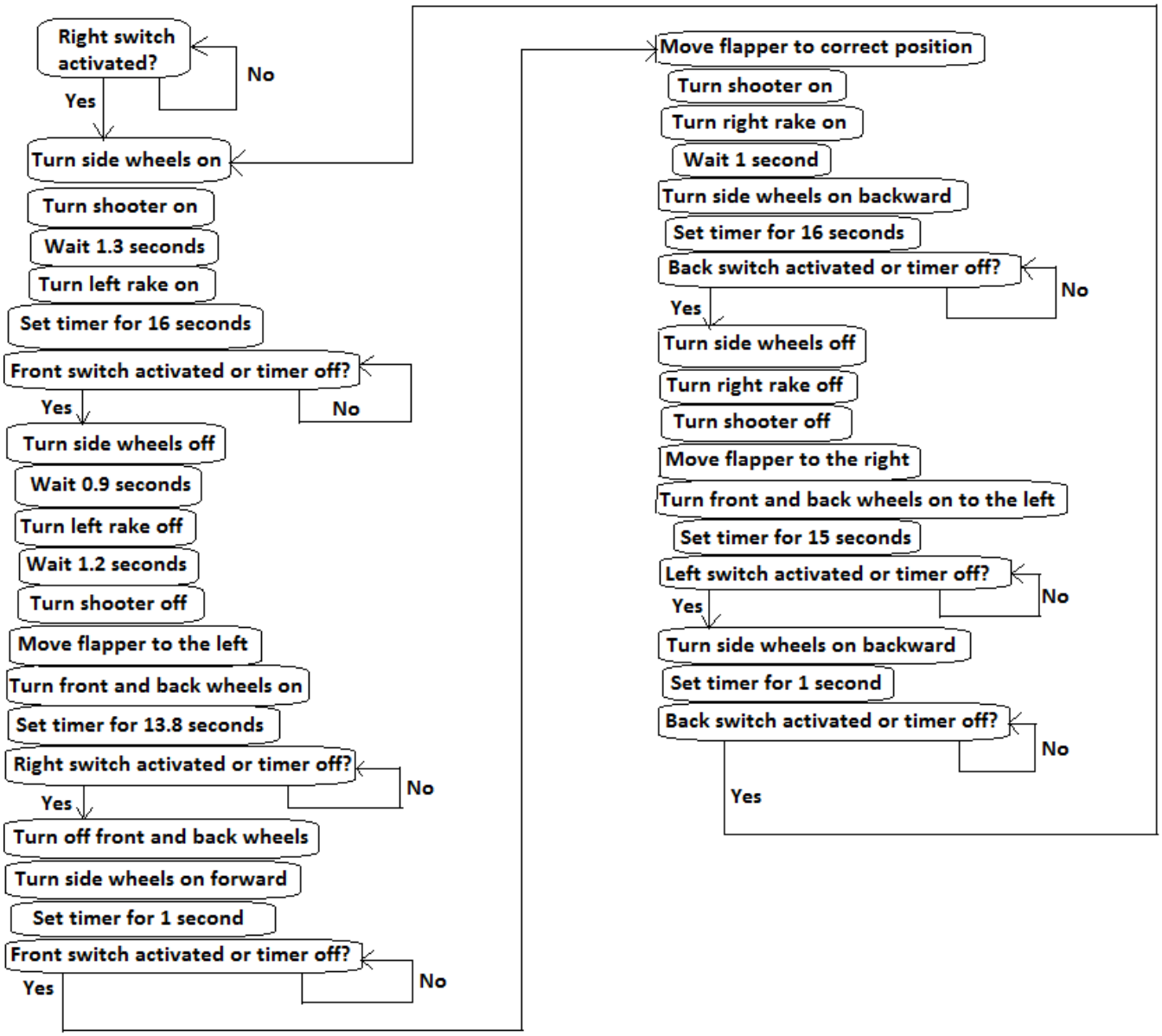

Figure 7. Software Flow Chart

VI. Materials Used

When choosing materials our main concerns were strength, cost and weight. We ordered expanded PVC after it was recommended to us for its high strength-to-weight ratio. We were told it could be cut like wood. While waiting for it to get delivered, we decided to build a prototype base board out of wood since it was cheap and easy to get. Once we got the material delivered, we used it for the base board, the shooter, and the walls of the ramp leading to the shooter.

We bought several different types of wood, varying in thickness, to use for the components of the robot. We used a thick piece of wood for the aiming device and for the walls that the rakes slid over. We also used wood to house the servos for the wheels and the rakes. The wood used for these components had to be the exact thickness of the servo so that it was held securely.

When considering the material to be used for the ramp, we wanted something that would be thin and light so that we could create a steep incline to ensure the balls would not get stuck on their way to the shooter. We decided to use sheet metal, as it could be easily cut to make the shape we needed, and was thin enough that we could make as steep an incline as necessary.

The rakes needed to be made out of something durable and with low friction so that they could easily slide over the walls when retrieving the balls. The ends of the rakes needed to be somewhat thin so that they could slide behind the balls on the rack before dragging them in, without knocking them off the rack. We chose to use thin steel bars for the arms of the rakes, and an L-shaped piece of steel for the ends

Everything was held together using different sizes of machine screws, washers, and nuts. As more and more components were added to the board, it got more difficult to find space for the

screws. With the smaller components we used super glue, because it was all that was necessary and we needed the board space. A full list of our parts can be seen in *Appendix B*.

VII. Building Process

Building the robot required many different tools and a detailed plan. After getting a rough idea of our design, we created a basic sketch of the layout we wanted, which can be seen in *Figure 8.* The sketch allowed us to draw the components at half scale and choose where to place each component without having to manually test where each component would fit during the building process.

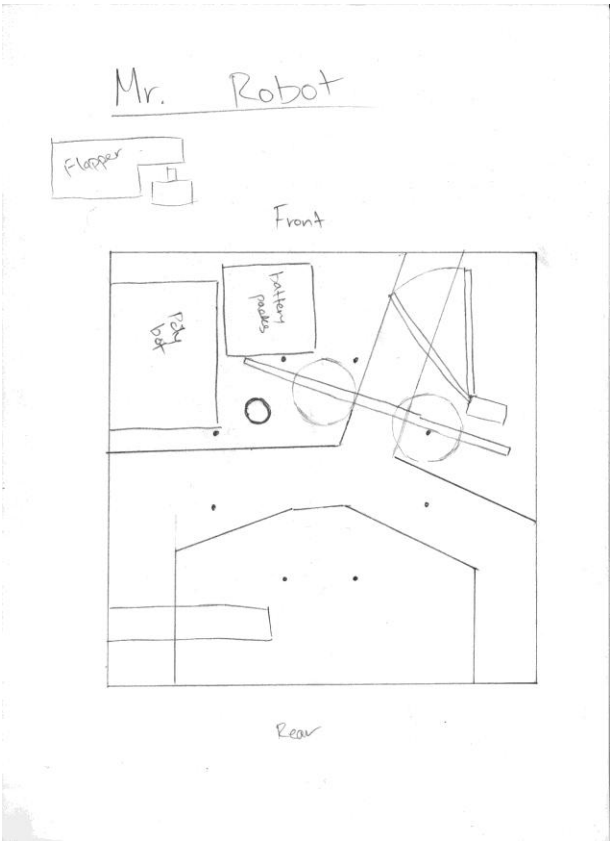

Figure 8. Initial Sketch

We started the process by creating the drive system on the base board. We used a jigsaw

to carve out pieces for the servos to be secured to, and later attached the wheels with glue. We used L-brackets and screws to attach the servos to the bottom of the board, and a ruler to make sure that each wheel was placed at the center of its respective side.

Once the drive system was built, and before attaching the wheels, we built and mounted the retrieval system. This took a lot of trial and error and measurements to ensure that each piece of the system was the perfect length to collect the balls from the rack. We had to cut out several different lengths of pieces for the small arm and for the long arm of the retrieval to test at first. Once we saw how close it was to being correct we could make exact measurements for the final pieces. We built each component of the retrieval and tested it to ensure it was correct before mounting it to the board. We used the same method to mount the servos to wood and to the board as we did for the drive system, but it was tricky attaching the two lever arms together. We needed them to be attached like a bolt, so that they stayed together but were also able to slide over each other without much friction. We ended up using a capped nut that we glued to the screw so that it was relatively loose.

Once the retrieval system was mounted, we traced out where the ramp was going to be. Then we traced the exact image onto a 12” x 12” piece of sheet metal and cut out the ramp. Next, we cut out pieces to create the walls that would encompass the ramp and that provided a way for the rakes to scoop the balls properly. The wall that supported the rake from the retrieval system was a thick piece of wood that we had been using during testing. Since it was working so effectively we decided to keep it. We mounted the pieces to the board using L-brackets. We created the rest of the wall by cutting out pieces of expanded PVC using a jigsaw into small rectangular pieces that could be pieced together to create the shape we wanted. We mounted

most of them using 1-2 L-brackets, and a couple with super glue if there wasn't enough space to use an L-bracket.

Next we created the shooter board. We cut out the necessary piece of expanded PVC into the shape we wanted using a jigsaw, and drilled small holes in the board for the DC motors to be mounted. We also attached two strips of steel on either side of the back of the shooter board for stability. We tested the angle at which the shooter should be positioned to provide an accurate shot from the left side of the court. We did this by creating a small part of the court at home and testing different angles until we were happy with the shot we were getting. Once we had the angle set, we mounted the shooter board to the base board using L-brackets and screws. We had to bend the L-brackets so that the shooter was angled upward at the appropriate spot for the shot we desired.

Finally we mounted our aiming device to the front of the board. First we tested where it should go and what angle it should be at to aim the ball effectively from the right side of the court. Then we cut out a piece of the base board big enough to fit the servo snugly. We mounted the servo using small screws similar to how we mounted the other servos. Then we glued a servo connector piece to the bottom of the wood piece used for the aiming device, and glued it onto the geared end of the servo.

Once the main components of the robot were in place we began testing everything. This is when we realized the glue holding the wheels on the bottom was not strong enough to support the weight of the robot. We had to attach L-brackets to the outside of each wheel and to the base board to make it sturdy enough. In order to do this we had to take several components off of the board, and perform a major operation to get everything back in place correctly.

Near the end of the project we mounted the Polybot board and battery packs. We mounted the Polybot board using a drill and provided screws, and we mounted the battery packs with Velcro. We also created housing for the Polybot board since it was at the front of the robot where it could get hit by balls. We did this by cutting out walls and a roof from the expanded PVC and gluing it all together. Pictures showing each step of the build process can be seen in *Appendix C*.

VIII. Key Learnings

Looking back on our project, there are a lot of things we would have done differently knowing what we know now. When we first started this project we did not have any experience with robotics and only had experience in one class with microcontrollers. As we worked on the project we quickly realized that our methods could be greatly improved and we could have saved an enormous amount of time.

The most significant change we could have made would have been to completely finish our drive system before continuing to the rest of the robot. Having a solid base from the beginning would have saved us a lot of time and effort. After realizing that the servos could not support the robot's load, we had to completely dismantle the robot in order to install extra supports. Also, testing a completed drive system without other components would have made tweaking the angles and heights much easier. After seeing other robots in the competition, we also realized how much easier and efficient it would have been to use DC motors and a motor controller for the drive system instead of servos. The DC motors had much better adapters available so our extra supports would not have been needed. They also have better torque so we wouldn't need to worry about stalling when the robot ran into a wall.

We are very happy with how our ball retrieval system turned out. It is extremely effective and original. However, due to its complexity, it took up a large portion of the robot and a lot of time to design and build. A simpler design, such as a rotating brush, would have taken up less space and still gotten the job done. The extra room could have allowed us more space for shooting and drive system components. We also would have included a sweeping device that would ensure that all the balls were removed from the rack.

In the beginning, we attempted to use timers to control the drive distance across the court.

Eventually we switched to using lever switches because the timing would change as the batteries discharged. The lever switches ended up being much easier to use and write code for. Luckily we were still able to use our timer code as a backup for when the switches failed or the robot was stuck on a ball. However, if we were to start over again we would use switches from the beginning.

Instead of screws and L-brackets we would have used glue or some other method to secure the components on the robot. The screws and L-brackets took up much more space than necessary. It ended up being very difficult to position the pieces so that the screws would not interfere with other components.

Our aiming device was just a piece of wood attached to a servo. Because of the small connection, it was not very sturdy. This caused the wood to wobble resulting in an inconsistent shot. To fix this problem we could create a mechanism that secures the wood from both the top and the bottom so that it remains in place as the balls bounce off of it. This way the balls would not lose as much energy as they are deflected when shot from the right side of the court.

Although we included a recovery algorithm in case the robot got stuck, it still cost us points when it occurred. During the matches, if a pile of balls would build up in the corner our robot would get stuck and our switches would never be pressed. Our recovery algorithm used timers to reset itself and get back on track, but most of the time it would miss the first few balls on the rack. A simple deflection device around the wheels would have allowed our robot to make its way through piles of balls instead of stalling.

Overall, most of our problems were caused by not sticking to our original plan, to keep it simple. Especially nearing the end of the project when our robot had decent functionality, we began to think about more complex methods and add-ons. These were all great ideas but we did

not have the time to implement and test them completely and actually ended up abandoning them when they did not work. A better way to approach the project would have been to perfect the simpler ideas we had before moving on to more complex methods that were not absolutely necessary.

IX. Conclusion

We learned a lot about robotics throughout the course of this project. Although we made many mistakes, we have learned from each of them and plan to enter this competition again in the future with our new found knowledge. Mr. Robot ended up doing quite well in the competition and we are pleased with the results. We learned that sometimes the simplest solution can also be the most effective. As computer engineers it is important to have experience in robotics, and this knowledge will help us with future work even if it isn't directly related to our jobs.

Appendix A – Code Base

test.c

```
#include "globals.h"
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

int main(void) {
   initialize();

   print_string("Mr. Robot");
   next_line();
   print_string("Roborodentia");
   delay_ms(1000);

   //Press right switch to start
   while (digital(1));
   delay_ms(500);

   //course using switches
   while (1) {
      forward();

      right();

      backward();

      left();
   }

   return 0;
}
```

utility_switch.c:

```
//Various utility functions for the PolyBot board
```

```
#include "globals.h"
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

void right_rake(void);
void left_rake(void);
void flapper_hide(void);
void flapper_left(void);
void flapper_right(void);
void switch_timer(u16 input, u16 time);

//returns status of the SW1 button
//returns 1 when the button is pressed
//returns 0 when the button is not pressed
u08 get_sw1(void) {
  return (s08)PIND >= 0;
}

//turns the relay driver port on
void relay_on(void) {
  sbi(DDRB,4);
  sbi(PORTB,4);
}

//turns the relay driver port off
void relay_off(void) {
  sbi(DDRB,4);
  cbi(PORTB,4);
}

//turns the led on
void led_on(void) {
  sbi(DDRA,4);
  sbi(PORTA,4);
}

//turns the led off
void led_off(void) {
  sbi(DDRA,4);
 cbi(PORTA,4);
```

```
}

//returns the value of a digital input
//parameter ranges from 0 to 7
u08 digital(u08 num) {
  u08 port_c;
  u08 port_b; //PB2 & PB3 are part of the multiplexed bus

  cli();
  //DDRC = 0;
  make_bus_input();
  cbi(PORTD,4); //enable digital input latch
  _delay_loop_1(1);
  port_c = PINC;
 port_b = PINB;
  _delay_loop_1(1);
  sbi(PORTD,4); //disable digital input latch
  sei();

  if (num < 2)
    return (((port_b>>2) & _BV(num)) != 0);
  else
    return ((port_c & _BV(num)) != 0);
}

//stop moving
void stop_moving(void) {
  clear_screen();
  lcd_cursor(0,0);
  print_string("Stopped");
  //turns off all servos for wheels
  servo_off(1);
  servo_off(2);
  servo_off(3);
  servo_off(4);
}

//move forward, collect balls and shoot
void forward(void) {
  //position flapper
  flapper_left();
```

```
  //turn on shooters
  clear_screen();
  lcd_cursor(0,0);
  print_string("Relay On");
  relay_on();

  //move forward
  clear_screen();
  lcd_cursor(0,0);
  print_string("Moving Forward");
 set_servo_position(1,152);
  set_servo_position(3,97);

  //turn on rake
  delay_ms(1300);
  left_rake();

  //stop when a wall is hit or timer expires
  switch_timer(2,16000);
  stop_moving();

  //turn off rake
  delay_ms(900);
  servo_off(6);

  //turn off shooters
  delay_ms(1200);
  clear_screen();
  lcd_cursor(0,0);
  print_string("Relay Off");
  relay_off();

  //hide the flapper before moving across
  flapper_hide();
  delay_ms(200);
}

//move backwards, collect balls and shoot
void backward(void) {
  //position the flapper
```

```
  flapper_right();
  delay_ms(100);

  //turn on the shooters
  clear_screen();
  lcd_cursor(0,0);
  print_string("Relay On");
  relay_on();

  clear_screen();
  lcd_cursor(0,0);
  print_string("Moving Backward");

  //turn on right rake
  right_rake();

  //move backward
  delay_ms(1000);
  set_servo_position(1,105);
  set_servo_position(3,148);

  //stop when a wall is hit or timer expires
  switch_timer(4,16000);
  stop_moving();

  //turn off rake
  servo_off(5);

  //turn off shooters
  clear_screen();
  lcd_cursor(0,0);
  print_string("Relay Off");
  relay_off();
}

//move right
void right(void) {
  //move right
  clear_screen();
  lcd_cursor(0,0);
  print_string("Moving Right");
```

```
  set_servo_position(2,78);
  set_servo_position(4,184);

  //stop when a wall is hit or timer expires
  //while (digital(1));
  switch_timer(1,13800);
  stop_moving();

  //Move forward to ensure we are in the corner
  if (digital(2)) {
    set_servo_position(1,147);
    set_servo_position(3,95);
    switch_timer(2,1000);
  }
}

//move left
void left(void) {
  //move left
  clear_screen();
  lcd_cursor(0,0);
  print_string("Moving Left");
  set_servo_position(2,230);
  set_servo_position(4,93);


  //stop when a wall is hit or timer expires
  //while (digital(3));
  switch_timer(3,15000);
  servo_off(4);
  delay_ms(100);
  stop_moving();

  //Move backward to ensure we are in the corner
  if (digital(4)) {
    set_servo_position(1,103);
    set_servo_position(3,146);
    switch_timer(4,1000);
  }
}
```

```
//collect balls on right side
void right_rake(void) {
  //start right rake
  clear_screen();
  lcd_cursor(0,0);
  print_string("Right Rake");
  set_servo_position(5,93);
}

//collect balls on left side
void left_rake(void) {
  //start left rake
  clear_screen();
  lcd_cursor(0,0);
  print_string("Left Rake");
  set_servo_position(6,95);
}

//move flapper out of the way
void flapper_hide(void) {
  clear_screen();
  lcd_cursor(0,0);
  print_string("Flapper Left");
  set_servo_position(0,158);
}

//position flapper for robot on left side
void flapper_left(void) {
  clear_screen();
  lcd_cursor(0,0);
  print_string("Flapper Left");
  set_servo_position(0,0);
}

//position flapper for robot on right side
void flapper_right(void) {
  clear_screen();
  lcd_cursor(0,0);
  print_string("Flapper Right");
  set_servo_position(0,134);
}
```

```
//perform all initialization
void initialize(void) {
  DDRC = 0x00;

 sbi(DDRD,4);
  sbi(DDRD,5);
  sbi(DDRD,6);

  sbi(PORTD,4);
  cbi(PORTD,5);
 cbi(PORTD,6);

  sbi(DDRB,0);
  sbi(DDRB,1);
  cbi(PORTB,0); //set E low
  cbi(PORTB,1); //set RS low

  sbi(PORTD,7); //enable pullup for switch

  //initialization for each system
  //commenting out unused systems increases overall performance
  servo_init();
  //adc_init();
  lcd_init();
  //motor_init();

  write_control(0x0C); //turn on lcd
  delay_us(160);
}

//waits for either digital[input] to trigger or timer to expire
void switch_timer(u16 input, u16 time) {
  u16 delay_counter = 0;
  u16 max_delay = time;

  while (1) {
    delay_ms(1);
    delay_counter++;

    //the switch was pressed
```

```
    if (!digital(input)) {
      break;
    }

    //exceeded the timeout value
    if (delay_counter > max_delay) {
      break;
    }
  }
}
```

Appendix B – Parts List

Parts	Quantity	Price
GWS EM400 Motor	2	11.98
2 in Diameter Lite Flite Wheel 2pk	1	4.39
Astroflight 1/8 in aluminum prop adapter	2	10.9
FXA317 (2052BX CAT-TRAK) Trak Wheels	4	29.92
Crest RS-540 Cermag Motor	2	29.98
Mini Aluminum Hub set of two - 3mm bore	1	8.49
Battery Holders w/ Switch	2	3.98
Heat Shrink	1	3.49
Electrical Tape	1	2.99
Wire - 3 Color Pack	1	6.99
Expanded PVC Plastic Sheet (Black (6mm))	3	17.25
Servos	11	128.55
Jumper Cables 6" M/F - Pack of 10	2	7.9
Corner Braces	3	17.83
Wood	3	18.1
Screws	19	22.42
Velcro	2	6.93
Duro Superglue	1	1.97
L metal	2	20.04
Metal Bar	1	5.72
Washers	4	4.72
Clevis Pin	2	3.02
Hitch Pin	1	0.59
Sheet Metal	1	6.97
Relay	1	4.69
Switches	6	18.64
Nuts for Push-button Switches	4	2.12
Total Tax	11	14.73
Total Shipping	6	42.84
	Total	458.14

Appendix C – Pictures

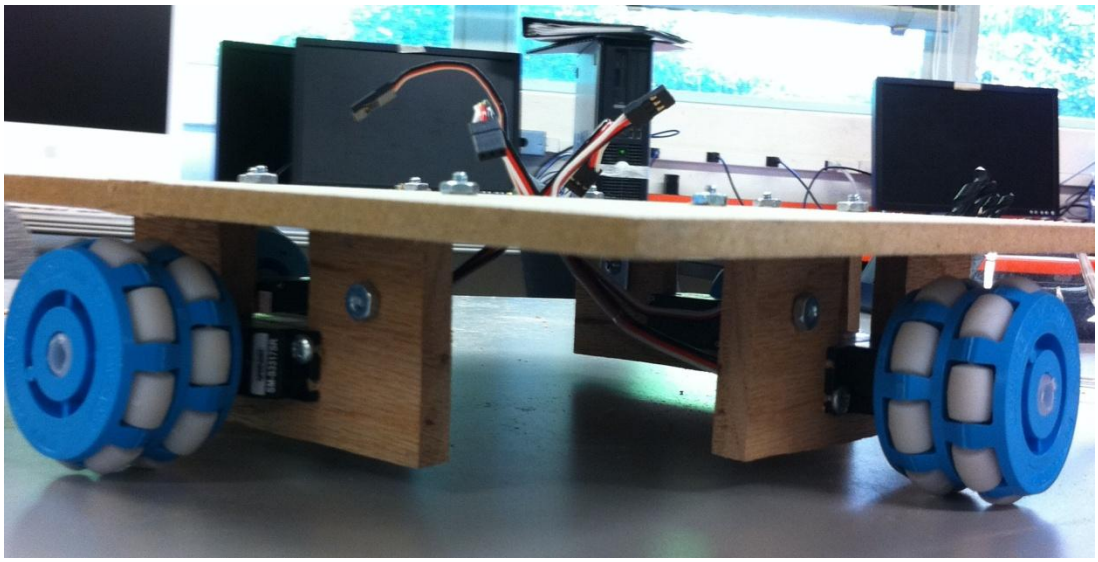

Figure 9. Original Drive System & Base Board

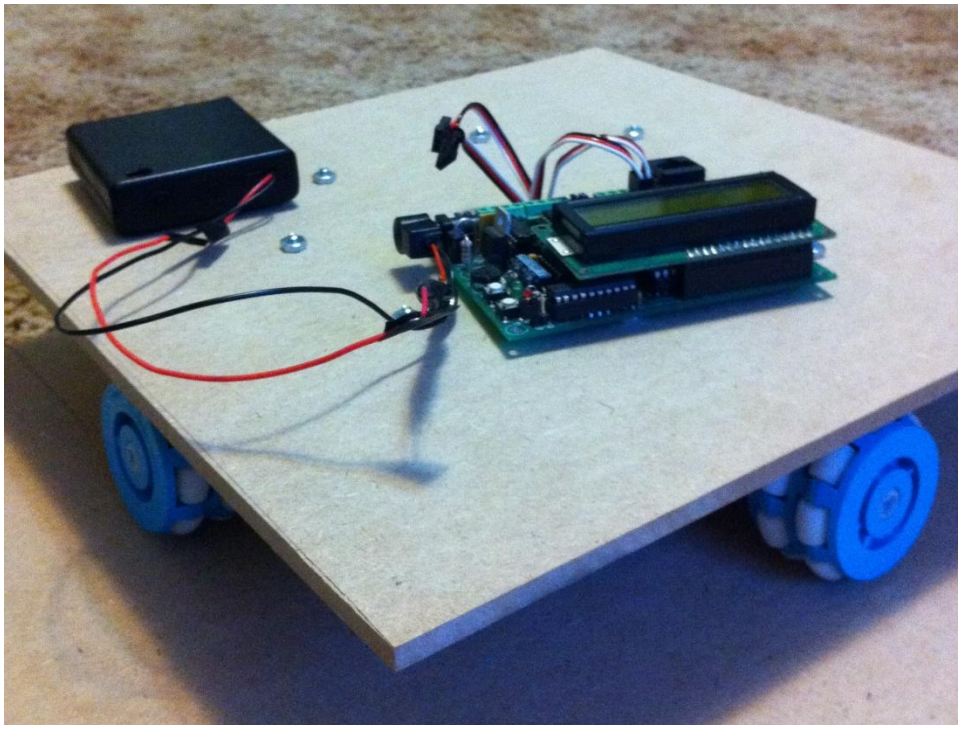

Figure 10. Prototype Base Board

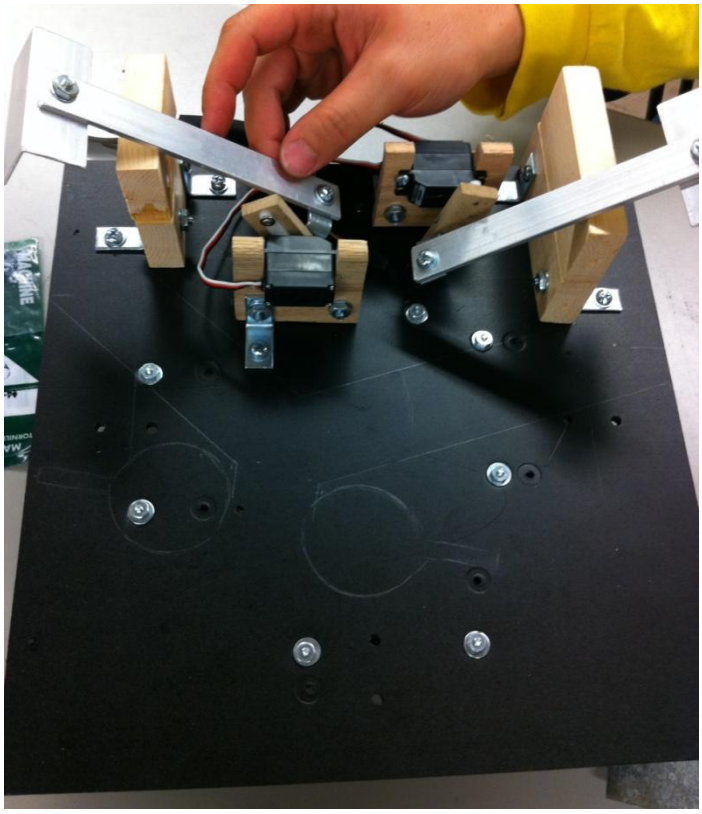

Figure 11. Final Base Board & Retrieval System

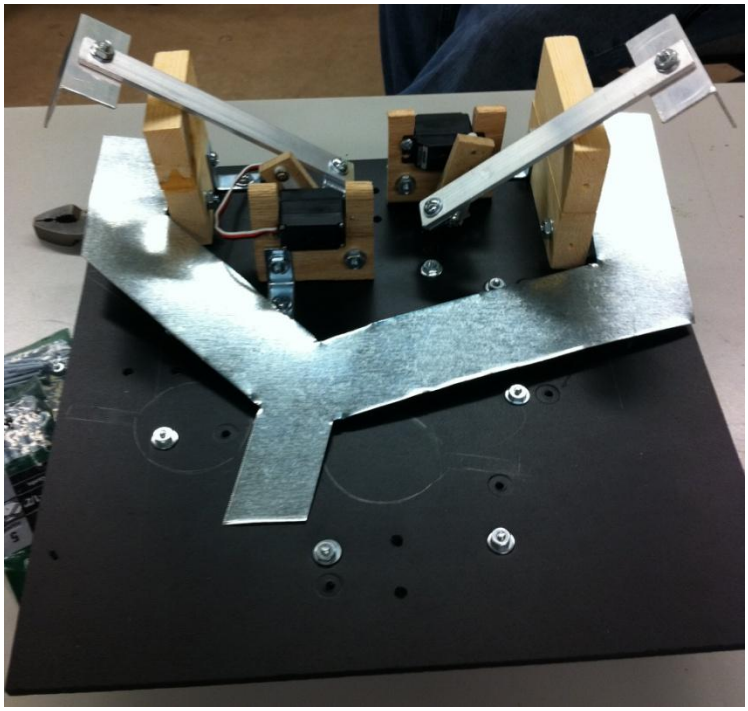

Figure 12. Ramp Cutout Added

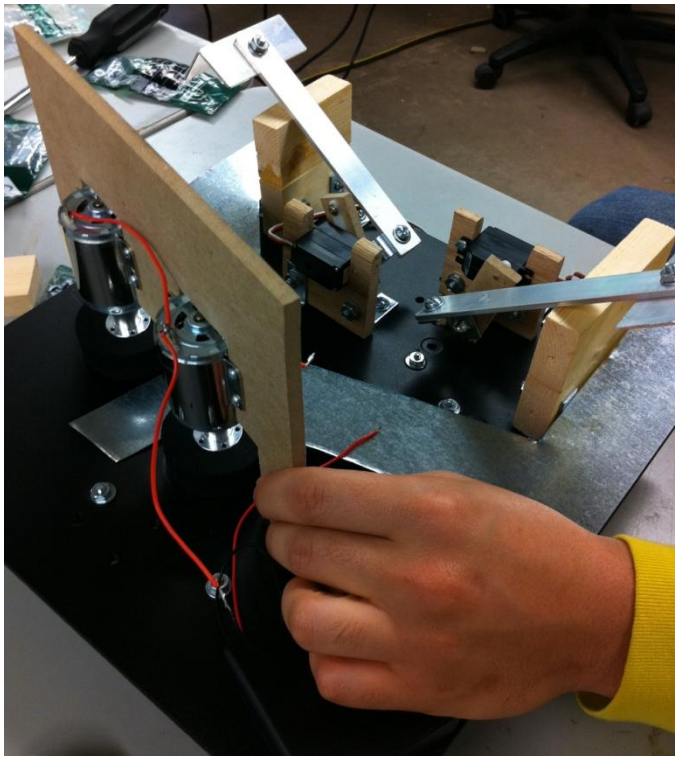

Figure 13. Prototype Shooter Added

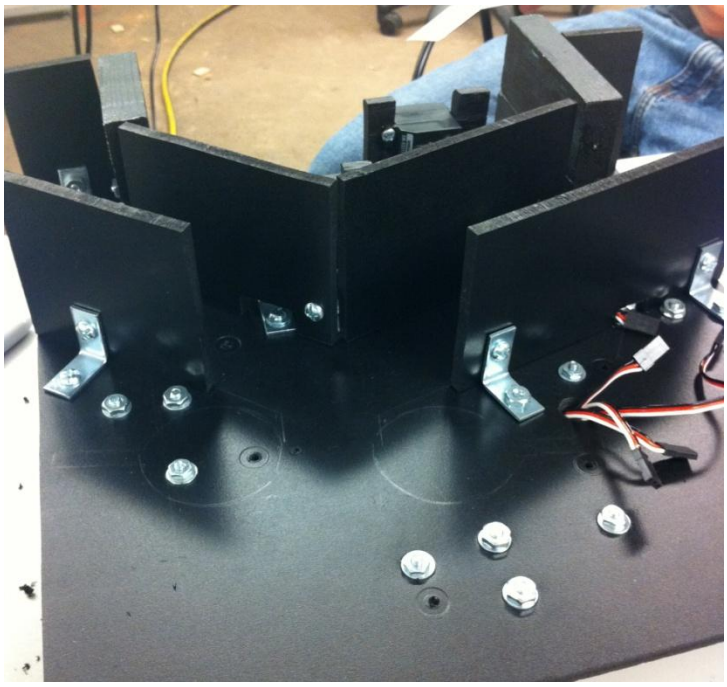

Figure 14. Walls Added

Figure 15. Flapper Added

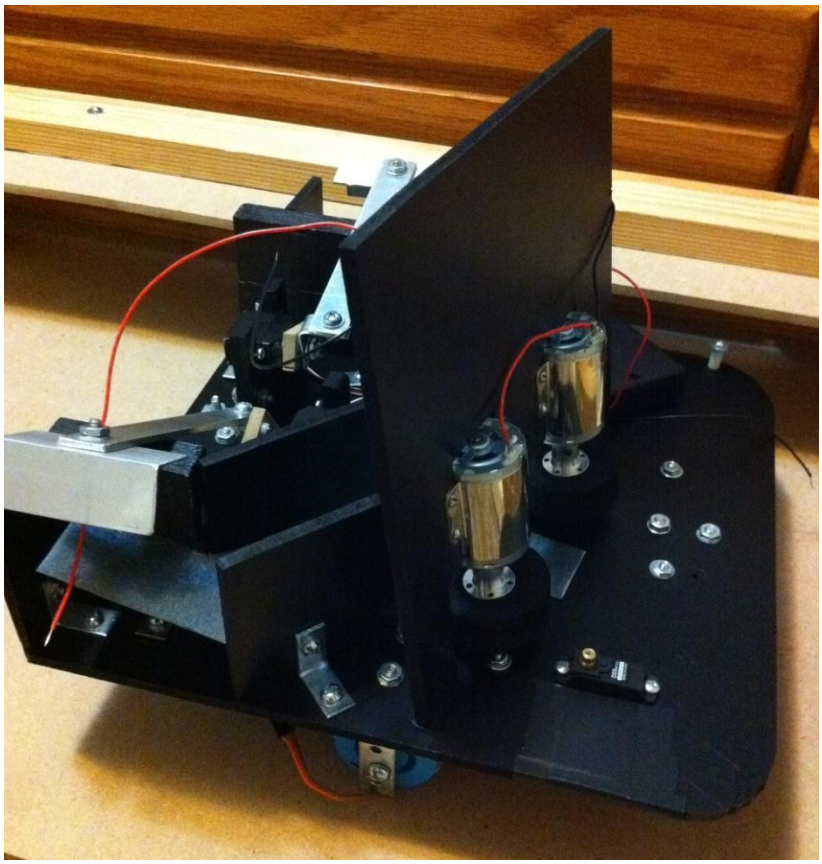

Figure 16. Shooter Added

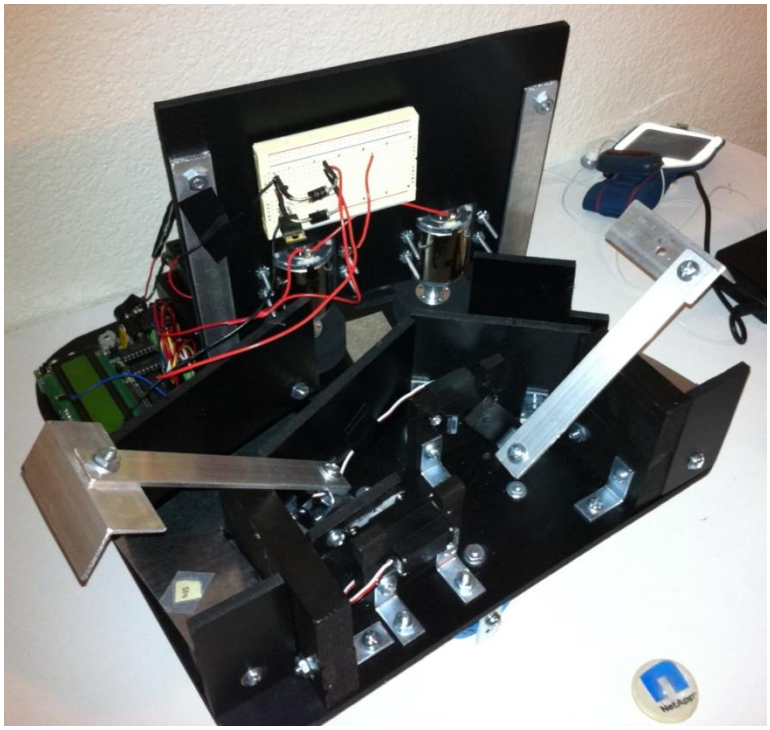

Figure 17. Original Circuit

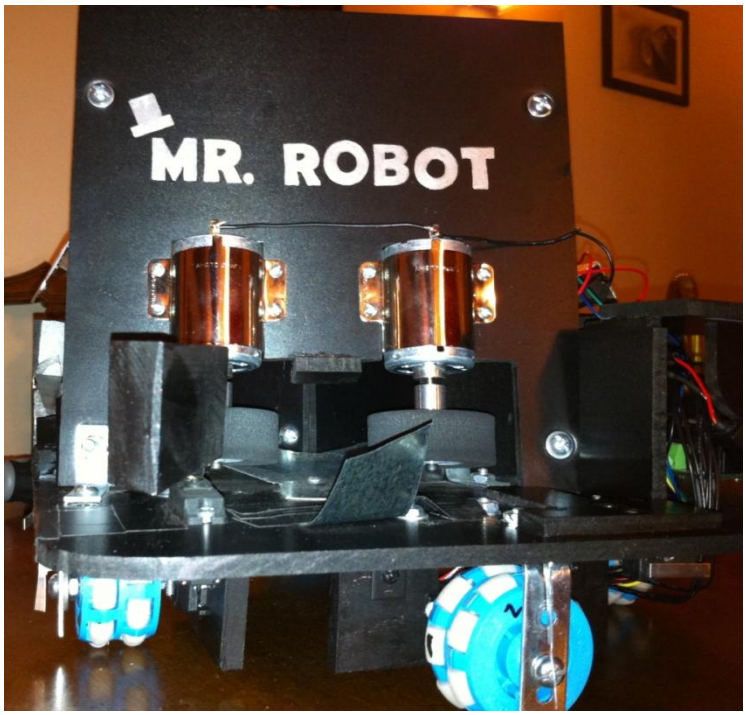

Figure 18. Final Design

Figure 19. Final Design from the Back

Appendix D – Analysis of Senior Project Design

Summary of Functional Requirements

Our robot has the ability to autonomously collect ping pong balls, maneuver around half of an 8’ x 6’ court and shoot the ping pong balls into a net on the opposite side

Primary Constraints

The Roborodentia competition imposed a set of constraints that we had to follow. It set the maximum size of our robot to be a cubic foot when a match starts as well as a 3 minute time limit. The competition also laid out the objectives that defined the functions our robot needed to perform.

Economic

- Original estimated cost of component parts: ~ $400
- Actual final cost of component parts: $458
- Additional equipment costs: $140
- Bill of materials can be found in *Appendix B*
- Original estimated development time: ~100 hours
- Actual development time: ~140 hours

Environmental

The environmental impact of our senior project was small. It includes the waste from parts and packaging that will not be reused by future teams. It also required power to use our drills and saw, as well as batteries in order to operate.

Manufacturability

Our project will not be manufactured on a commercial scale. There were no major challenges while constructing our robot other than finding the right materials or parts.

Sustainability

- The only thing required to maintain our project would be fresh batteries and the occasional tightening of screws.
- Our project impacts the sustainable use of resources with its requirement of constant battery change.
- Lower power DC motors and servos would improve the sustainability of our project.
- Upgrading to lower power components would require a relatively significant amount of money because those components would cost more than the ones in place now.

Ethical

The only ethical issue involved with our project is the issue of cheating in the Roborodentia competition. Unfair advantages in code or hardware that violate the competition specifications would fall in this category.

Health and Safety

Our product by itself does not affect the well-being of society. During the manufacturing of our project, we had to use numerous power tools which require caution when operating to avoid injury.

Social and Political

There are no implications at the social or political level for our project.

Development

We used a new microcontroller, the ATMega 644, for the first time as well as interfacing with motors and servos.