Robert Mickle
Advisor John Seng
Senior Project Report
Winter-Spring 2013

Roborodentia Entry: Rob-ot

Table of Contents

1. Introduction

The goal of my senior project was to successfully compete in Cal Poly's annual Roborodentia Open House competition. This project allowed me to utilize my knowledge of microcontrollers, electronics, and motion control, as well as providing an opportunity to learn and apply the mechanical-design aspects of robotics. I entered the competition as a one-man team and all expenses were paid out of pocket. This task was interesting for me in that it was my first independent and large scale project (all previous projects were done in teams.) The basic object of the competition is to navigate the course (four by eight foot arena with masking tape grid) and collect and stack as many cat food cans as possible in three minutes.

2. Electrical Design

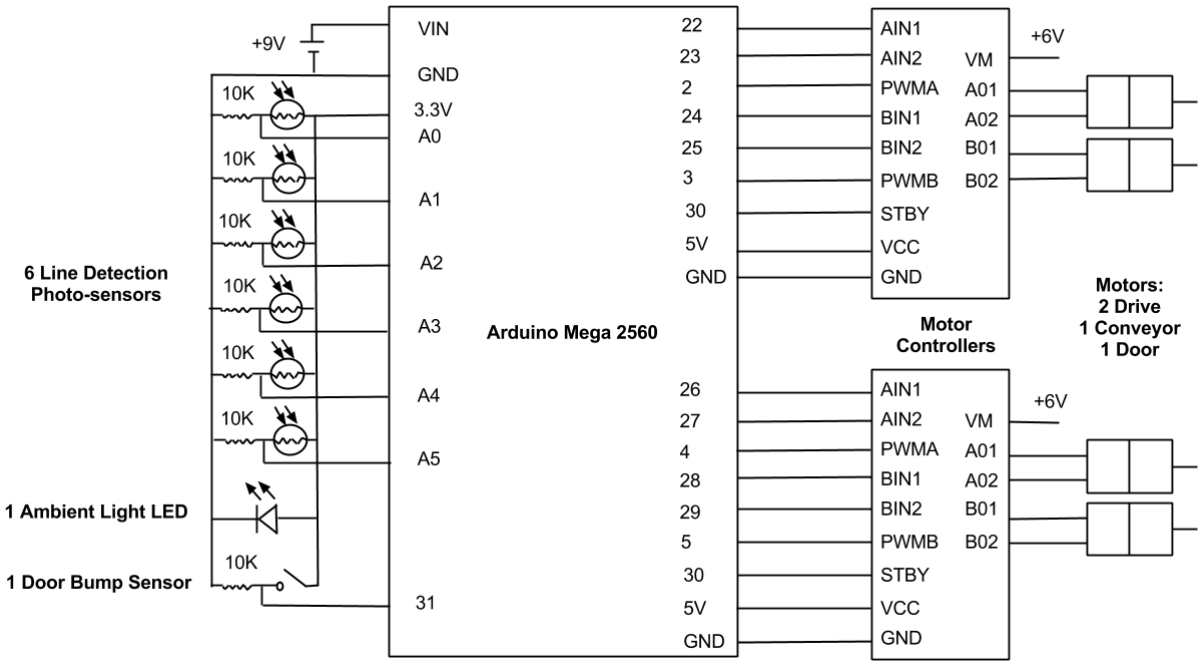

Figure 1. Electrical Schematic of Robot

Components Used:

(1) Arduino Mega 2560 Microcontroller
(2) Dual channel motor controller, 15V max, 1.2A average
(4) 100:1 geared motor, 140 rpm at 6V
(6) Photo resistor
(1) Red LED
(1) Bump sensor
(1) 9V battery
(4) 1.5V AA battery
(7) 10KΩ resistors

The Arduino was supplied with a 9 volt battery. The motor controllers were powered by 5 volts from the Arduino and the remaining photo sensors, bump sensor, and LED were powered by 3.3 volts from the Arduino. The motors were supplied with a separate dedicated 6 volts (4 1.5V AA batteries) that the motor controllers used to drive the 4 motors. The photo sensors were used for line following. An LED was required under the chassis to prevent surrounding light and shadows from affecting the line readings. The single bump sensor was placed to determine when the can-disposal door had reached the closed position. Image 7 shows the top view of the robot,

specifically highlighting the Arduino microcontroller and the breadboard containing the motor controllers, resistors, and the accompanying wiring. The majority of the confusing-appearing wiring results from connecting the microcontroller to the two dual channel motor controllers.

3. Mechanical Design

The base (Figure 1) of the robot consists of a plywood sheet that is reinforced on either side with aluminum angle brackets. The ramp (Figure 2) was constructed from plywood as well, supported by hinges on the base of the ramp (allowing for access to the components underneath) and threaded rods at the top of the ramp. The electronics platform (Figure 3) was supported by vertical threaded rods. All of the threaded rods and bolts were secured with washers and a combination of hex nuts (for parts that were to be permanently secured) and wing nuts (for parts that were frequently assembled and disassembled to access or adjust various aspects of the robot). The conveyer and can disposal system are constructed from Lego® Technic parts and powered by non-Lego motors.

The design goal was to have the cans funneled to the “grab wheels” (seen in Image 9) and subsequently up the conveyor track on the ramp. After reaching the top of the ramp, the cans drop into the designated shaft. Once a stack of three is achieved, the can-disposal door opens while the robot is moving forward, leaving a stack behind.

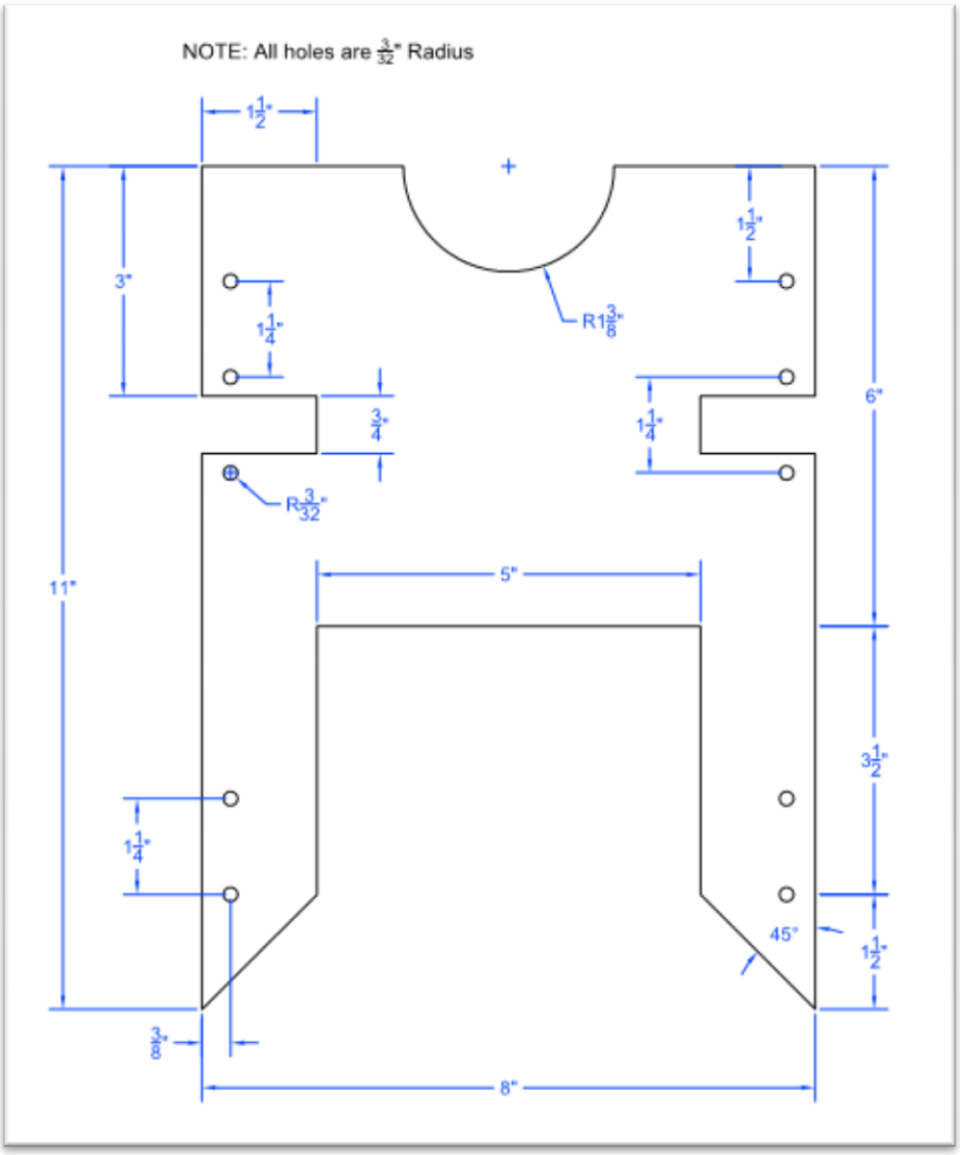

Figure 2. Chassis Base

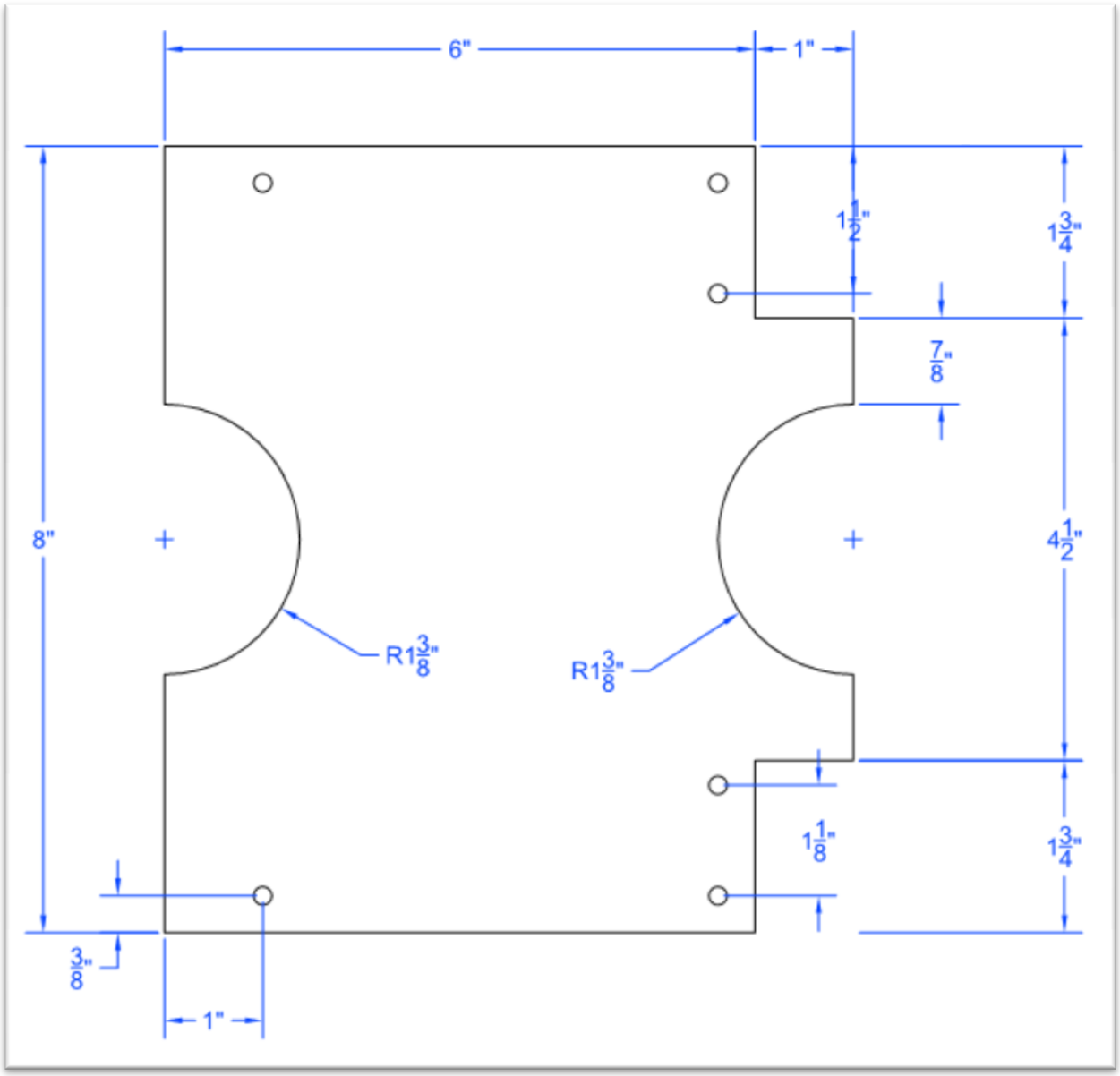

Figure 3. Ramp

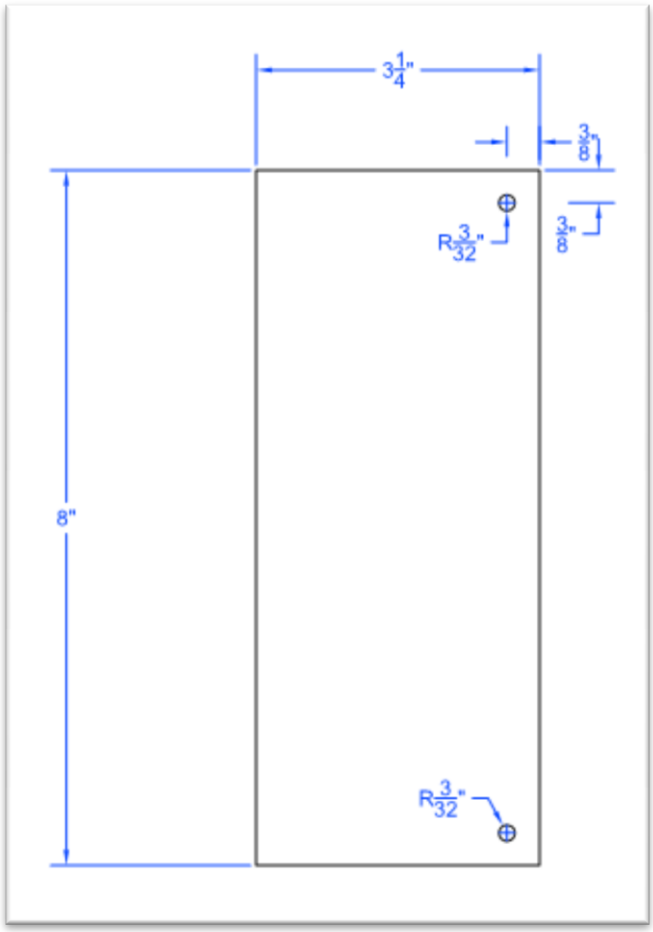

Figure 4. Electronics Platform

4. Code Structure and Analysis

The code can be better understood if the strategy used in the contest is discussed as well. The initial and final decision on how the robot would acquire the cans was that the robot would follow a strict script (no AI), using the grid on the course to navigate. The code was developed modularly and iteratively as electrical parts were received and physical construction began, leading up to final testing and modification on the course.

The first step was writing simple functions to get a single wheel spinning in a desired direction. After basic microcontroller control of the motors was attained, functions were developed that included parameters for wheel direction and speed. Having this basic functionality, a library of functions was developed in order mobilize the robot. Functions included moving the robot forward and reverse, turning about its axis (wheels spinning in opposite directions) and turning in a sweeping motion (have a single wheel moving), and finally making it stop.

Next step in the development was developing reliable line following code. This involved reading from analog input channels and determining the course of action given those values. The analog input returns an integer spanning from 0 to 1023. When the robot is initially turned on it takes readings from the sensors and these values are saved and used for all comparisons. A threshold value is then set. If the current reading is greater than the initial value plus the threshold value, the robot needs to compensate because it has deviated from it desired course. The physical sensors were placed side by side, and once established on a line, would be used to detect if the robot was driving straight. If the left sensor detected a significant change, the micro-controller would turn off of the right motor for one cycle of the detection loop and this would continue until the robot was back on track.

Having adequate motion and detection code, testing could begin and the desired path for the robot to take could start being developed. This consisted largely of expanding the library of useful functions. Several of the final functions required up to eight parameters, but this prevented duplication of code and effort. This library provided to be sufficiently robust and towards the end of development, the tweaking and fine tuning came from modifying small issues such as moving and turning speeds, as well as establishing of the “hard-coded” values used in the sensitivity of the line detection.

Figure 5 below shows the basic flow of the code that resides on the microcontroller. After the start up and initialization of variables, the robot waits for the switch to be flipped in order to start the competition. Once flipped, the conveyor starts and the robot proceeds to go through the directional script and move about the course. As can be seen in the code (section 7.2) there are many delays placed in the code. This was to compensate for the very slow movement of robot and the conveying mechanism.

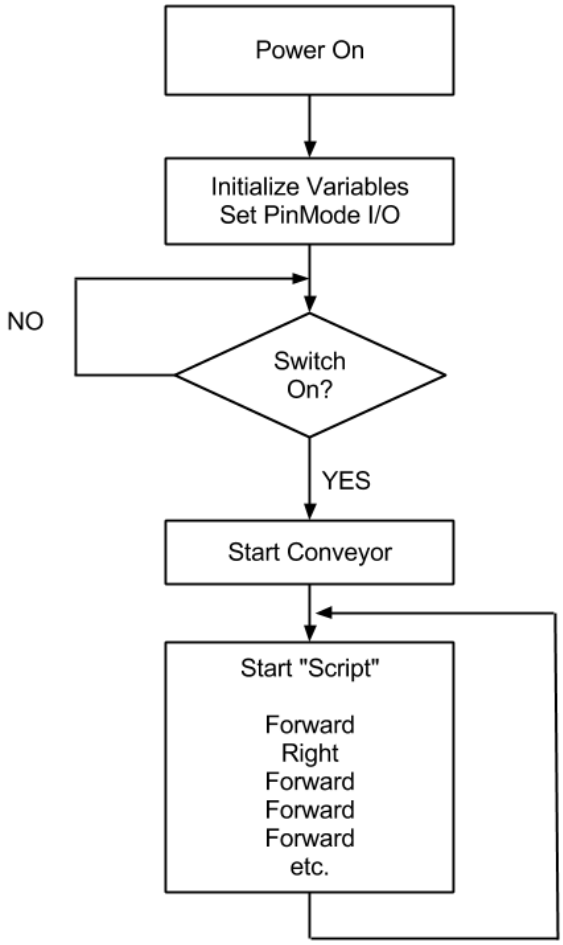

Figure 5. Software Block Diagram

5. What I Would Have Done Differently

When posed with the question, "What would you have done differently?" the responses that come to mind fall into two distinct categories: small design changes and large design changes.

Small design changes are considered "easy" tweaks or improvements while the overall functionality of the robot remains the same. The first improvement would involve swapping out all of the photo sensors with infrared emitting and detecting sensors. Infrared sensors are much more reliable and are not as easily affected by ambient lighting. The photo sensors that were used occasionally generated false positives when encountering shadows in the environment, which were then interpreted as dark lines. Another change involving the sensors is the placement and spacing of the sensors on the robot. After further research and peer input it appears that it is much more effective to have the sensors "straddle" the lines that are being followed as opposed to "sitting" on top of a line. Apparently this makes for smoother line following and is less prone to error. The next improvement involves upgrading the motors that were used for driving the robot as well as powering the conveyor and disposal door. Although the motors worked, they were much too slow for the competitive environment. Originally they were appealing because of their small size, high torque, and low power consumption. The tradeoff was speed. It took approximately fifteen seconds for a can to make its way up the conveyor. Although multiple cans could be on the ramp at the same time, the time it took was far too long for a three minute competition. The drive motors also proved to be very slow when navigating the course. Between the various motors, there was a considerable amount of time wasted either navigating from can to can, or having to wait for a can to reach the top of the ramp.

Large design changes are considered changes that, in hindsight, would drastically alter the functionality of the robot. If I were to go back in time and redo this competition, the design would try to pick up as many cans as possible. The design that was implemented was only capable of creating stacks of up to three cans. Many of the other contestants this year went for designs that tried to stack as many as possible and were able to score more points, while my design limited itself to fewer maximum points per stack. A physical design improvement would be to create a large robot. My entry was roughly eight inches wide and eleven inches long, when it was allowed to be up to thirteen inches square at the start of the competition and could expand to fifteen inches square. I believe a larger robot would allow for a wider can gathering ability as well as create more space for mechanical and electrical components. The next change would involve exploring the possibility of implementing AI and decision making. My robot's success was limited by the fact that it was very slow, had sensor issues and was following a set script. This was a major issue in the event that it needed to utilize a restart. The robot would have to placed back in the designated starting location and have to follow the script all the way back to the point where it failed and then continue on past that, which wasted valuable completion time. While there is nothing inherently wrong with following a set script, it allows for terrible failure if the robot happens to get off course as it doesn't have means to correct itself and will just continue with the script using erroneous environment data that is treated as the expected input.

6. Conclusion

In the end the robot was able to compete and successfully stack cans as designed. However, there were some issues with its performance. As mentioned above, the robot was slow to move around the course and to get the cans to the top of its ramp. Another problem encountered was that the cans in the competition were painted with a glossy finish, unlike the practice cans used. This caused slippage on the way up the ramp due to the conveyor not being able to grip the competition cans as well as tackier practice cans. The robot also needed to be restarted in both of its matches because it veered off course and was unable to correct itself. In hindsight there were several small improvements that could have been made that would have drastically improved the robots performance. However, the robot was only designed to stack a max of three can. Even on a perfect run, it would have not made it far in the tournament, losing to the robots that managed to generate stacks of seven or eight cans. This project provided an excellent opportunity to apply all of the knowledge gained up until this point in my college career, as well as explore and learn about the mechanical side of robotics.

7. Appendix

7.1 Photographs

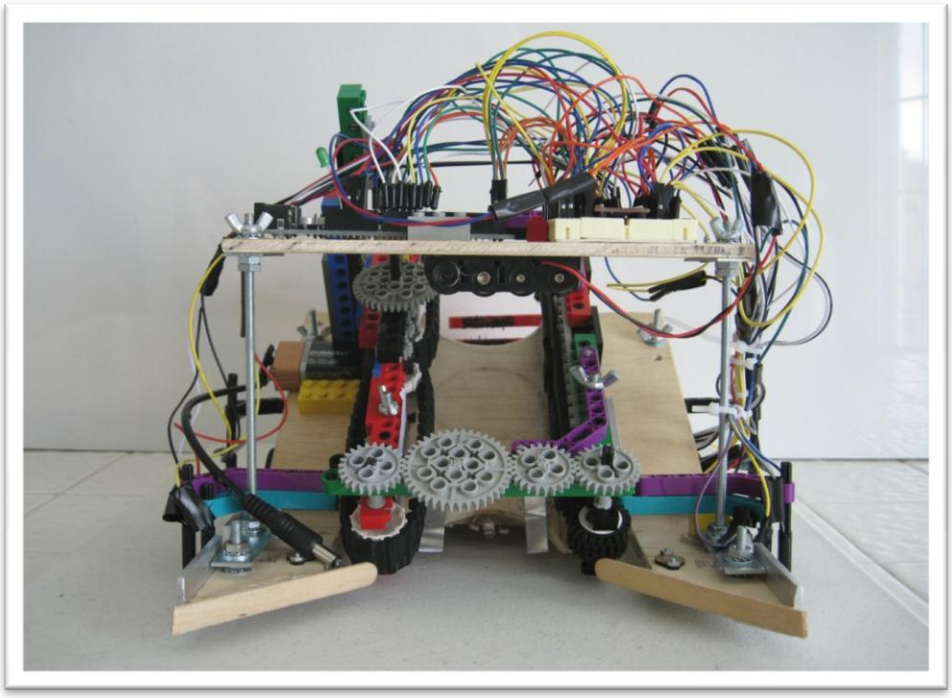

Image 1. Front View

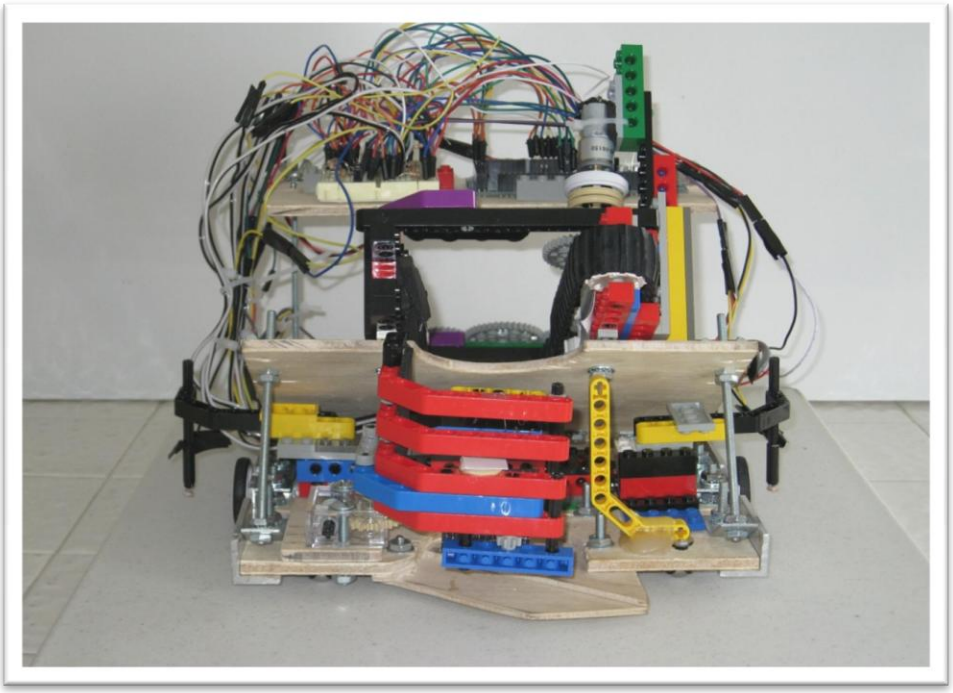

Image 2. Rear View

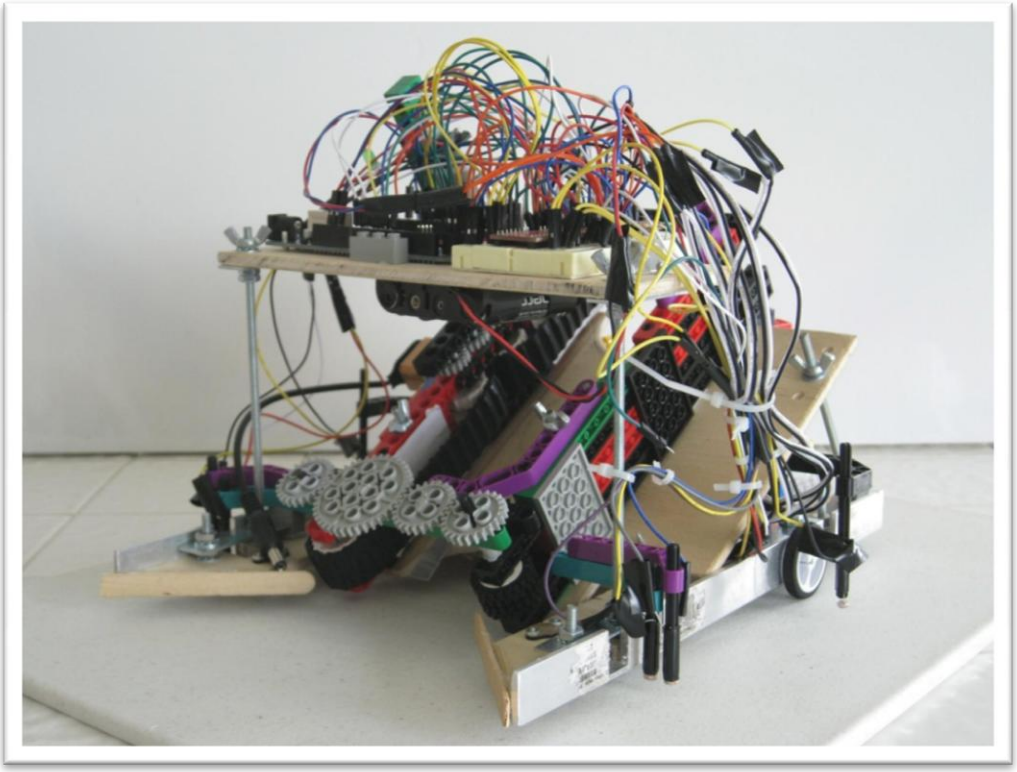

Image 3. Front Left View

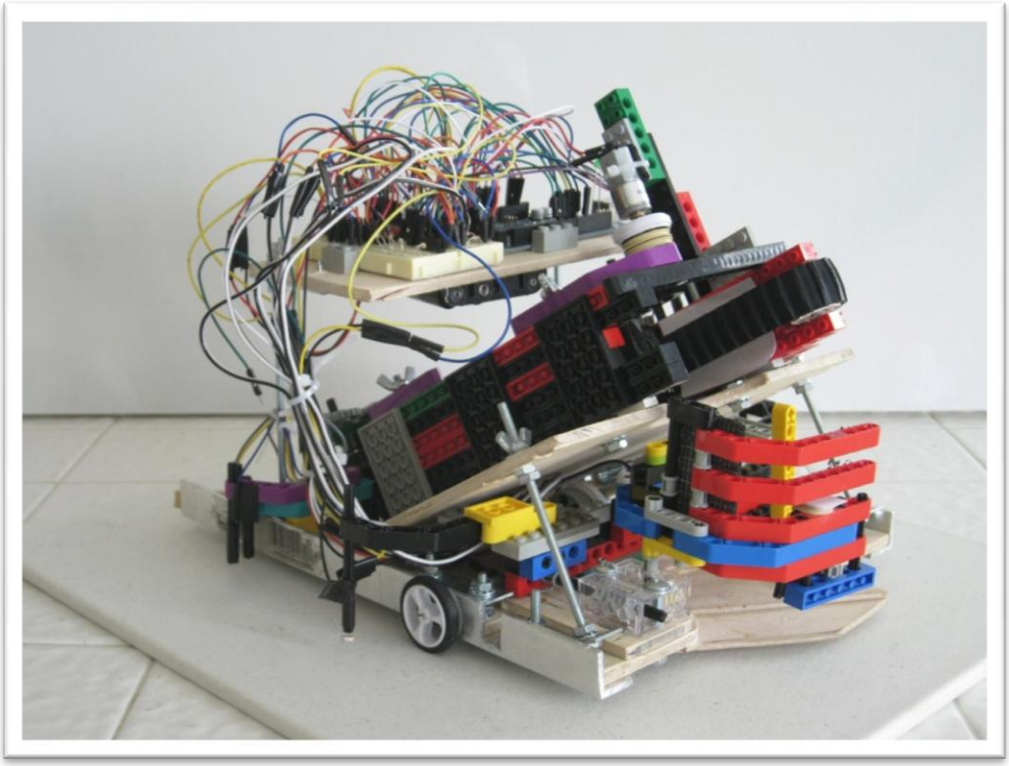

Image 4. Rear Left View

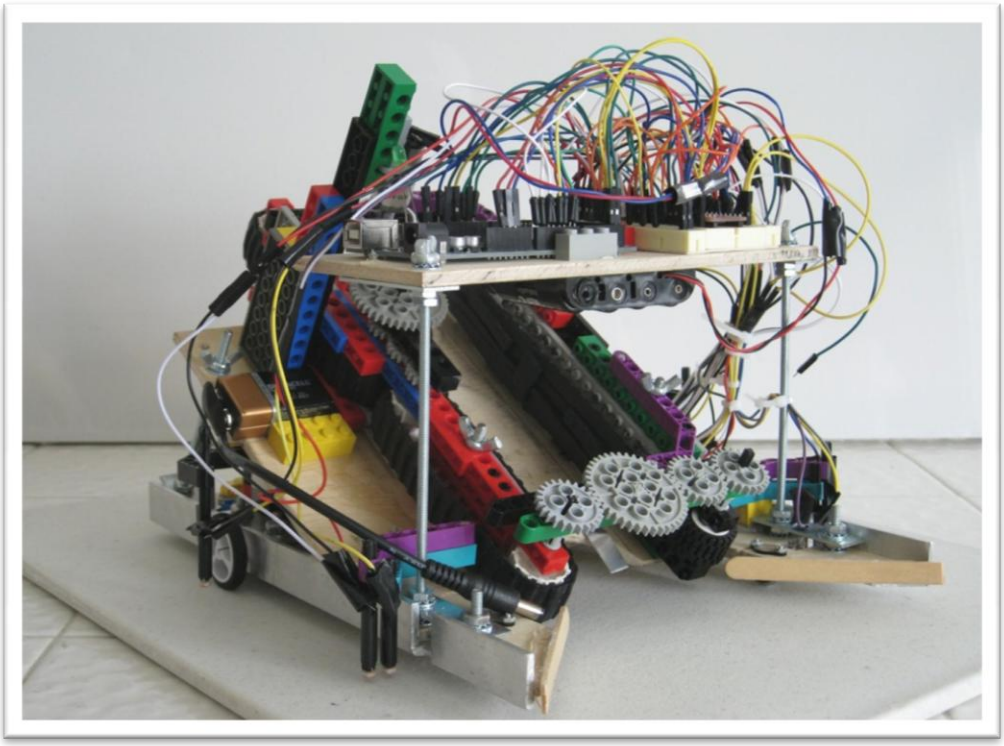

Image 5. Front Right View

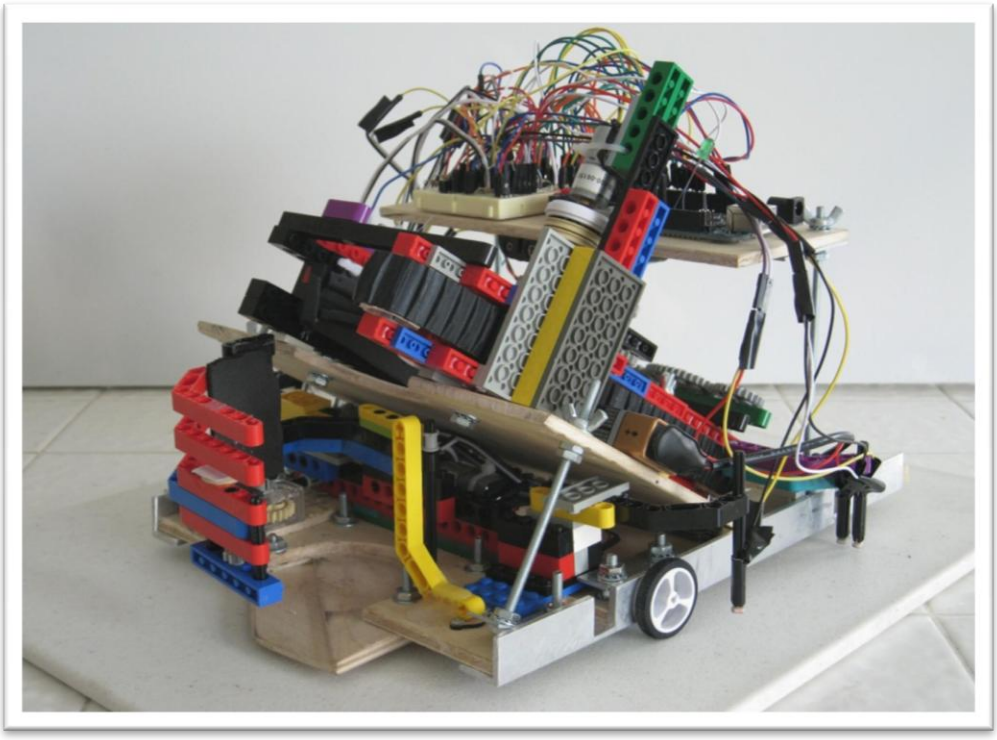

Image 6. Rear Right View

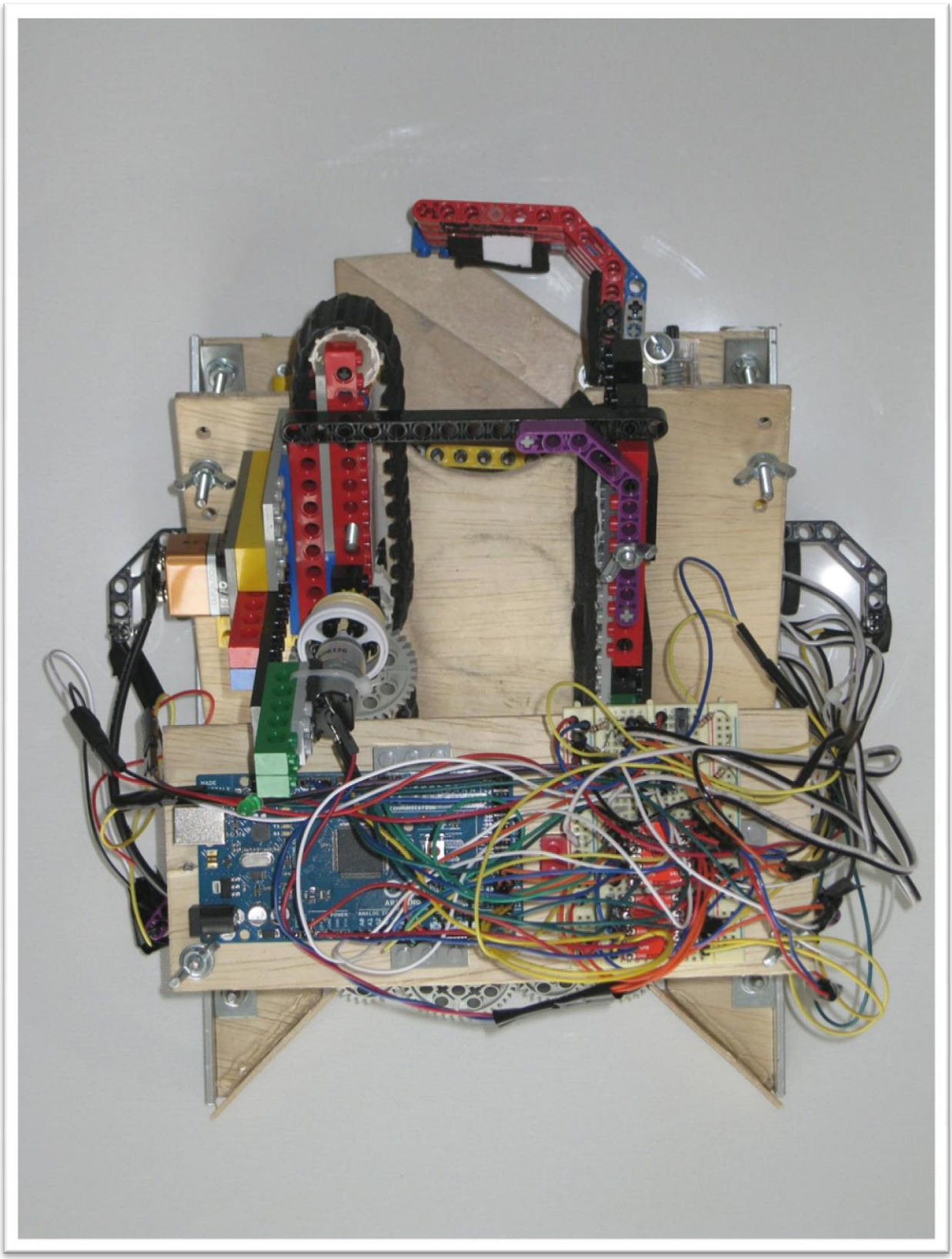

Image 7. Top View

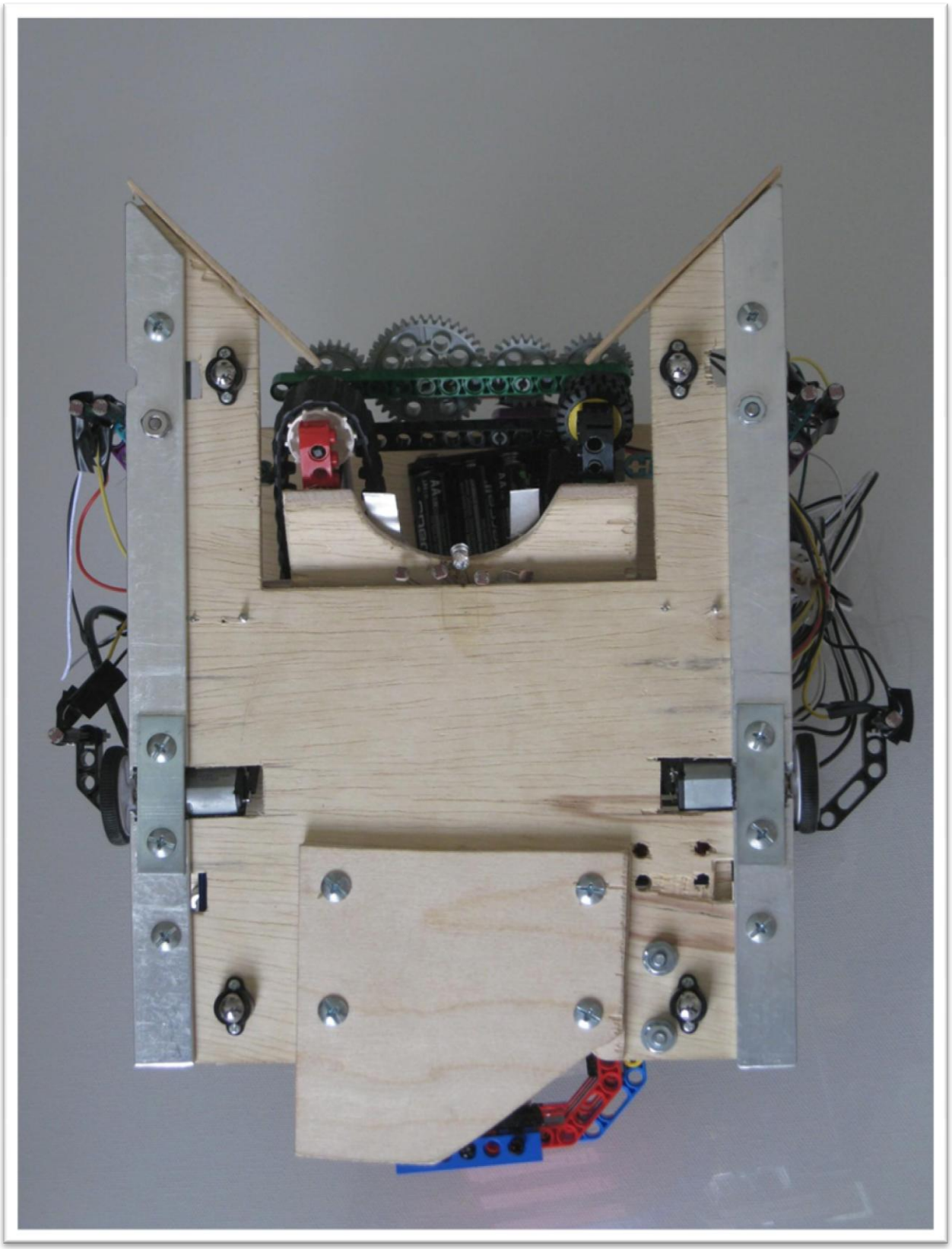

Image 8. Bottom View

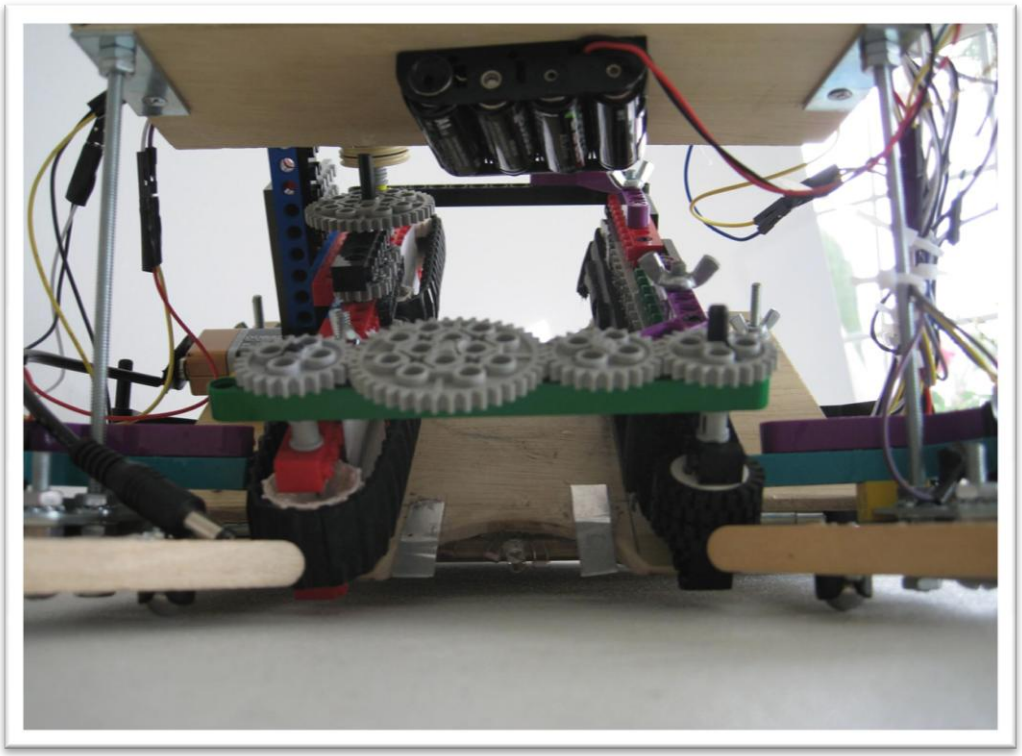

Image 9. Front Inside

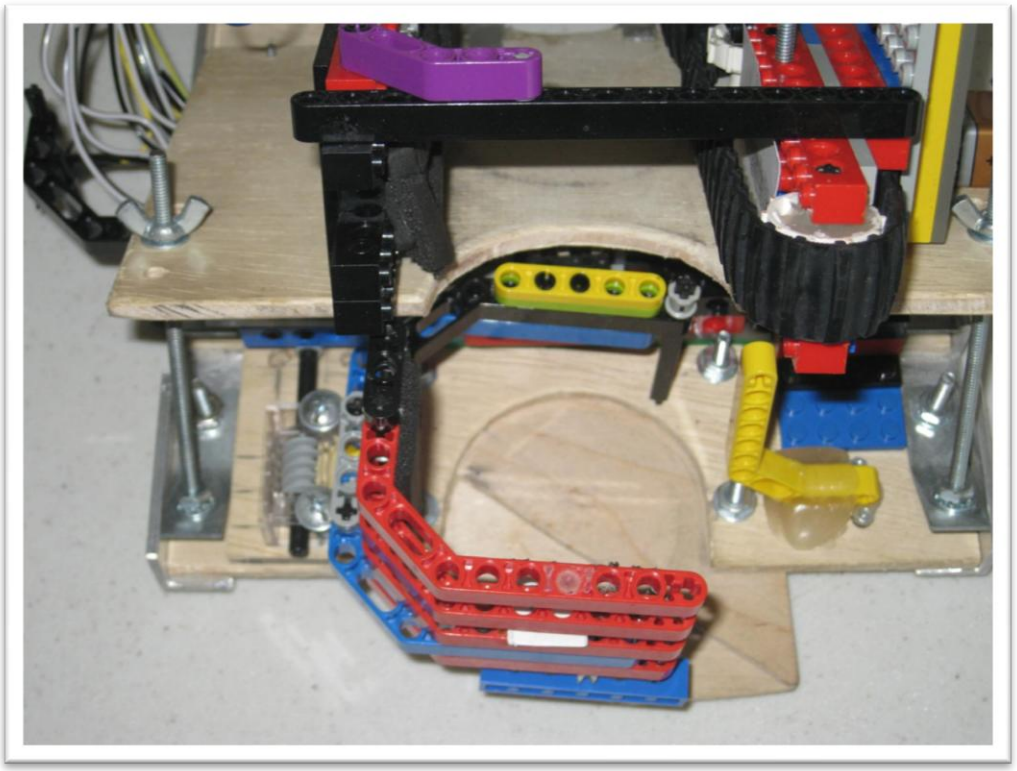

Image 10. Rear Inside, Can Receptacle

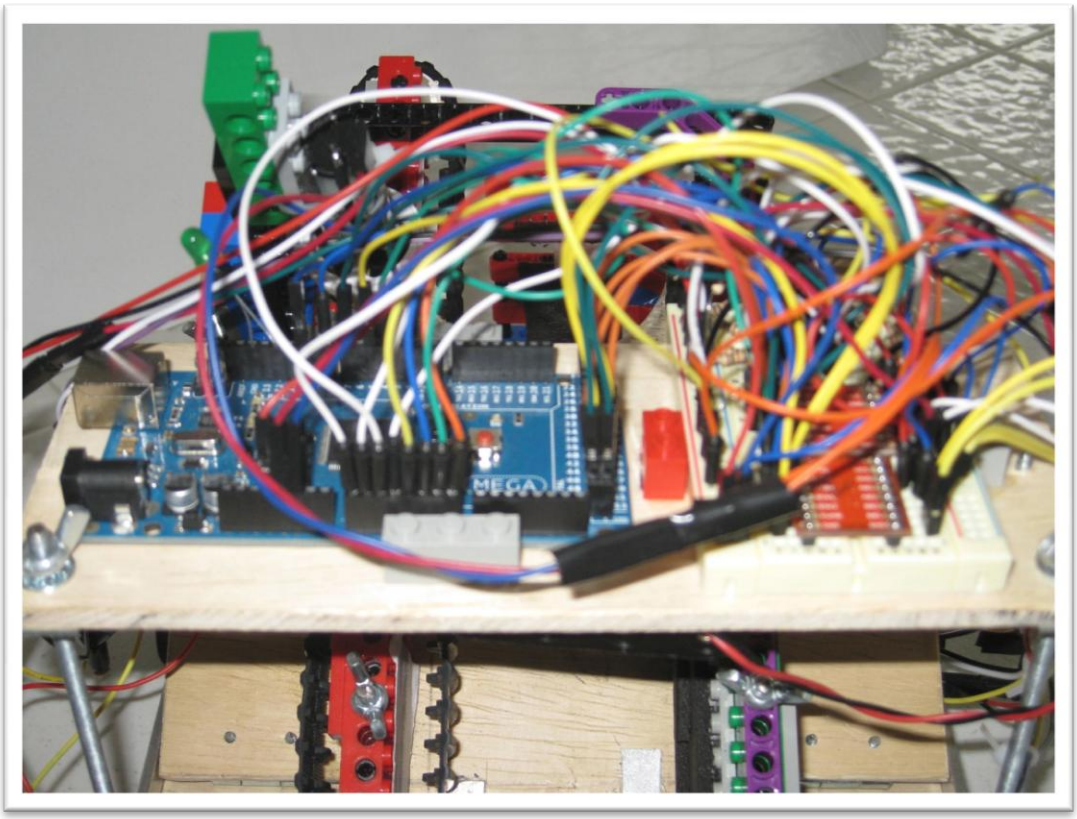

Image 11. View of Micro Controller and Electrical Components

7.2 Code

```
/*Author: Robert Mickle
  Project: CPE Senior Project April 20th 2013
  University: Cal Poly San Luis Obispo
  Advisor: Professor Seng
*/

//test LED
int blinky = 12;

//right side photo sensors
int RFI = A0;
int RFO = A1;
int RB = A2;
int RFIV,RFOV,RBV,initRBV,initRFIV,initRFOV = 0;

//left side photo sensors
int LB = A5;
int LBV,initLBV = 0;
//center outside sensors
int OL = A4;
int OR = A3;
int OLV, ORV, initOLV, initORV = 0;
//center photo sensors
int CL = A6;
```

```
int CR = A7;
int CLV,CRV,initCLV,initCRV = 0;
//standby
int STBY = 10;
//Motor 1 A Righty
int PWMA = 3; //Speed control
int AIN1 = 52; //Direction
int AIN2 = 53; //Direction
//Motor 2 B Lefty
int PWMB = 5; //Speed control
int BIN1 = 50; //Direction
int BIN2 = 51; //Direction
//Motor 3 C Conveyor
int PWMC = 2;
int CIN1 = 48;
int CIN2 = 49;
//Motor 4 D Door
int PWMD = 4;
int DIN1 = 46;
int DIN2 = 47;
//KILLSWITCH
int ONOFF = 7;
int LED = 13;
int switchState = LOW;
//photo resistor threshold

int threshold =75;
int del = 4000;
int pace = 210;
int beginPace = 180;
int turnPace = 192;
int beginTurnPace = 180;
int avgCenter = 0;

long time = 0;

void setup(){
  //standby
  pinMode(STBY, OUTPUT);
  //motor 1
  pinMode(PWMA, OUTPUT);
  pinMode(AIN1, OUTPUT);
  pinMode(AIN2, OUTPUT);
  //motor 2
  pinMode(PWMB, OUTPUT);
  pinMode(BIN1, OUTPUT);
  pinMode(BIN2, OUTPUT);
  //motor 3
  pinMode(PWMC, OUTPUT);
  pinMode(CIN1, OUTPUT);
  pinMode(CIN2, OUTPUT);
  //motor 4
  pinMode(PWMD, OUTPUT);
  pinMode(DIN1, OUTPUT);
  pinMode(DIN2, OUTPUT);
  //On Switch and LED
```

```
  pinMode(LED, OUTPUT);
  pinMode(ONOFF, INPUT);
  pinMode(blinky, OUTPUT);
  //get white reading
  initLBV = analogRead(LB);
  initOLV = analogRead(OL);
  initORV = analogRead(OR);
  initRFIV = analogRead(RFI);
  initRFOV = analogRead(RFO);
  initRBV = analogRead(RB);
  initCRV = analogRead(CR);
  initCLV = analogRead(CL);
}

void loop(){
  //CHECK ONSWITCH
  switchState = digitalRead(ONOFF);
  while(switchState == LOW){
    digitalWrite(LED, LOW);
    stopAll();
    delay(100);
    switchState = digitalRead(ONOFF);
  }
  digitalWrite(LED, HIGH);

  //COMMENCE
  //start track forever
  startTrack();

  time = millis();

  marchStart(CL, CR, initCLV, initCRV, beginPace, threshold);
  delay(250);
  initCLV = analogRead(CL);
  initCRV = analogRead(CR);
  delay(250);

  marchStart(LB, RB, initLBV, initRBV, beginPace, threshold);
  delay(del);

  //turnL(beginTurnPace, CL, initCLV, threshold);
  turnR(beginTurnPace, CR, initCRV, threshold);
  delay(50);

  march(CL, CR, LB, initCLV, initCRV, initLBV, pace, threshold);
  delay(del);
  march(CL, CR, LB, initCLV, initCRV, initLBV, pace, threshold);
  delay(del);

  turnL(turnPace, CL, initCLV, threshold);
  delay(del);

  march(CL, CR, LB, initCLV, initCRV, initLBV, pace, threshold);
  delay(del-1000);

  turnL(turnPace, CL, initCLV, threshold);
```

```
  delay(400);

  openDoor();

  march(CL, CR, LB, initCLV, initCRV, initLBV, pace, threshold);
  closeDoor();
  delay(del);

  march(CL, CR, LB, initCLV, initCRV, initLBV, pace, threshold);
  delay(del);
  delay(del);

  turnL(turnPace, CL, initCLV, threshold);
  delay(del);
  delay(del/2);

  openDoor();

  marchToWhite(CL,CR,LB,RB, initCLV, initCRV, initLBV, initRBV, pace,
threshold);
  closeDoor();
  delay(50);

  openL(164, RFO, initRFOV, threshold);
  delay(150);

  initRFOV = analogRead(RFO);
  initRFIV = analogRead(RFI);

  delay(250);

  march(RFI, RFO, LB, initRFIV, initRFOV, initLBV, pace, threshold);
  delay(50);

  march(RFI, RFO, LB, initRFIV, initRFOV, initLBV, pace, threshold);
  delay(del);

  march(RFI, RFO, LB, initRFIV, initRFOV, initLBV, pace, threshold);
  delay(del);
  marchTime(RFI, RFO, initRFIV, initRFOV, pace, threshold, 1500);
  delay(del);
  rightArc(turnPace, LB, initLBV, threshold);
  delay(del);
  delay(del);

  turnR(turnPace, CR, initCRV, threshold);
  openDoor();

  march(RFI, RFO, LB, initRFIV, initRFOV, initLBV, pace, threshold);
  closeDoor();

  delay(10000);

 march(CL, CR, RB, initCLV, initCRV, initRBV, pace, threshold);
 delay(del);
 march(CL, CR, RB, initCLV, initCRV, initRBV, pace, threshold);
```

```
 delay(del);
 turnR(turnPace, CR, initCRV, threshold);
 delay(del);
 delay(del);
 delay(del/2);
 openDoor();
 march(CL, CR, RB, initCLV, initCRV, initRBV, pace, threshold);
 closeDoor();
 delay(del);
 turnR(turnPace, CR, initCRV, threshold);
 delay(del);
 march(CL, CR, RB, initCLV, initCRV, initRBV, pace, threshold);
 delay(del);
 march(CL, CR, RB, initCLV, initCRV, initRBV, pace, threshold);
 delay(del);
 delay(del);
 turnR(turnPace, CR, initCRV, threshold);
 openDoor();
 march(CL, CR, RB, initCLV, initCRV, initRBV, pace, threshold);

}

void marchTime(int L, int R, int initL, int initR, int speed, int threshold,
int time){

  int LV, RV = 0;
  boolean exit = false;
  int ignoretime = 0;

  while(!exit){
    ignoretime++;
    LV = analogRead(L);
    RV = analogRead(R);

    forwardR(speed);
    forwardL(speed);

    if(abs(LV - initL) > threshold){
      stopR();
    }
    if(abs(RV - initR) > threshold){
      stopL();
    }
    if(ignoretime > time){
        stopR();
        stopL();
        exit = true;
    }
    delay(1);
  }
}

void openL(int speed, int S, int initS,int threshold){
  int SV = 0;
  boolean exit = false;
```

```
  while(!exit){
    SV = analogRead(S);
    reverseL(speed);
    if(abs(SV - initS) >threshold){
      stopL();
      exit = true;
    }
    delay(2);
  }
}

void rightArc(int speed, int S, int initS, int threshold){
  int SV = 0;
  boolean exit = false;
  while(!exit){
    SV = analogRead(S);
    forwardL(speed);
    if(abs(SV - initS) > threshold){
      stopL();
      exit = true;
    }
    delay(2);
  }
}

void marchToWhite(int L, int R, int LB, int RB, int initL, int initR,int
initLB, int initRB, int speed, int threshold){

  int LV, RV, SV, LBV, RBV = 0;
  boolean exit = false;
  boolean exitR = false;
  boolean exitL = false;
  int ignoretime = 0;

  while(!exit){
    LV = analogRead(L);
    RV = analogRead(R);

    forwardR(speed);
    forwardL(speed);

    if(abs(LV - initL) > threshold){
      stopR();
    }
    if(abs(RV - initR) > threshold){
      stopL();
    }
    if((abs(LV - initL) > threshold)&&(abs(RV - initR) > threshold)){
      while(!(exitL && exitR)){
        LBV = analogRead(LB);
        RBV = analogRead(RB);

        forwardR(3*speed/2);
        forwardL(3*speed/2);

        if(abs(LBV - initLB) > threshold){
```

```
          stopL();
          exitL = true;
        }
        if(abs(RBV - initRB) > threshold){
          stopR();
          exitR = true;
        }
        delay(2);
      }
      exit = true;
    }
    ignoretime++;
    delay(2);
  }
}


int marchStart(int L, int R, int initL, int initR, int speed, int threshold){

  int LV, RV, CL, CR, CLRet, CRRet, ret= 0;
  boolean exitL= false;
  boolean exitR = false;
  while(!(exitL && exitR)){
    LV = analogRead(L);
    RV = analogRead(R);

    forwardR(speed);
    forwardL(speed);

    if(abs(LV - initL) > threshold){
      stopL();
      exitL = true;
    }
    if(abs(RV - initR) > threshold){
      stopR();
      exitR = true;
    }
    delay(2);
  }
}

//turn robot around axis to the right
void turnR(int speed, int sensor,int initS, int threshold){
  int S = 0;
  boolean exit = false;
  int ignoretime = 0;

  while(!exit){
    ignoretime++;
    S = analogRead(sensor);
    forwardL(speed);
    reverseR(speed);
    if(ignoretime > 120){
      if(!(abs(S - initS) > threshold)){
        stopR();
        stopL();
```

```
        exit = true;
      }
    }
    delay(2);
  }
}
//turn robot around axis to the left
void turnL(int speed, int sensor,int initS, int threshold){
  int S = 0;
  boolean exit = false;
  int ignoretime = 0;

  while(!exit){
    ignoretime++;
    S = analogRead(sensor);
    forwardR(speed);
    reverseL(speed);
    if(ignoretime > 120){
      if(!(abs(S - initS) > threshold)){
        stopR();
        stopL();
        exit = true;
      }
    }
    delay(2);
  }
}

void march(int L, int R, int S, int initL, int initR,int initS, int speed,
int threshold){

  int LV, RV, SV = 0;
  boolean exit = false;
  int ignoretime = 0;

  while(!exit){
    LV = analogRead(L);
    RV = analogRead(R);
    SV = analogRead(S);

    forwardR(speed);
    forwardL(speed);

    if(abs(LV - initL) > threshold){
      stopR();
    }
    if(abs(RV - initR) > threshold){
      stopL();
    }
    if(ignoretime > 100){
      if(abs(SV - initS) > threshold){
        stopR();
        stopL();
        exit = true;
      }
    }
```

```
    ignoretime++;
    delay(2);
  }
}

//forward for certain time at certain speed
void forwardT(int speed, int time){
  forwardR(speed);
  forwardL(speed);
  delay(time);
  stopR();
  stopL();
}
//reverse for certain time at certain speed
void reverseT(int speed, int time){
  reverseR(speed);
  reverseL(speed);
  delay(time);
  stopR();
  stopL();
}

//left motor forward at certain speed
void forwardL(int speed){
  move(2,speed, 1);
}
//right motor forward at certain speed
void forwardR(int speed){
  move(1,speed,1);
}
//left motor reverse at certain speed
void reverseL(int speed){
  move(2,speed, 0);
}
//right motor reverse at certain speed
void reverseR(int speed){
  move(1,speed, 0);
}
//start the conveyor track
void startTrack(){
  move(3,255,0);
}
//stop the conveyor track
void stopTrack(){
  stopC();
}
//open the door
void openDoor(){
  move(4, 255, 0);
  delay(2300);
  stopD();
}
//close the door
void closeDoor(){
  int contactor = 8;
  int buttonState;
```

```
  int lastButtonState = LOW;
  long lastDebounceTime = 0;
  long debounceDelay = 20;
  boolean exit = false;

  while(!exit){
    move(4,255,1);
    int reading = digitalRead(contactor);
    if (reading != LOW) {
      exit = true;
    }
  }
  stopD();
}
//stop everything
void stopAll(){
  digitalWrite(STBY, LOW);
}
//stop right motor
void stopR(){
  digitalWrite(AIN1, LOW);
  digitalWrite(AIN2, LOW);
}
//stop left motor
void stopL(){
  digitalWrite(BIN1, LOW);
  digitalWrite(BIN2, LOW);
}
//stop conveyor motor
void stopC(){
  digitalWrite(CIN1, LOW);
  digitalWrite(CIN2, LOW);
}
//stop door motor
void stopD(){
  digitalWrite(DIN1, LOW);
  digitalWrite(DIN2, LOW);
}
//move function called in order to move the motors
void move(int motor, int speed, int direction){

  digitalWrite(STBY, HIGH); //disable standby

  boolean inPin1 = LOW;
  boolean inPin2 = HIGH;

  if(direction == 1){
    inPin1 = HIGH;
    inPin2 = LOW;
  }

  if(motor == 1){
    digitalWrite(AIN1, inPin1);
    digitalWrite(AIN2, inPin2);
    analogWrite(PWMA, speed);
  }
```

```
  else if (motor == 2){
    digitalWrite(BIN1, inPin1);
    digitalWrite(BIN2, inPin2);
    analogWrite(PWMB, speed);
  }
  else if(motor == 3){
    digitalWrite(CIN1, inPin1);
    digitalWrite(CIN2, inPin2);
    analogWrite(PWMC, speed);
  }
  else {
    digitalWrite(DIN1, inPin1);
    digitalWrite(DIN2, inPin2);
    analogWrite(PWMD, speed);
  }
}
```