HandyBug: Design Portfolio

Mete Morris

ECE 100-04

Illinois Institute of Technology

Introduction to the Profession I

Due Date: 12/8/14

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to offer my gratitude to Professor Erdal Oruklu important knowledge given in the lecture and our teaching assistant Andy Huang for his patience and assistance in the lab. I would also like to thank my teammates for Competition 0(Giancarlo Barillas, Aaron Xie), Competition 1(Sebastian Hall, Ying Tang Lu) and Competition 2(Shuei Aoki, Zhaoyin Qin) for their help in creating and constructing the robots Banana, Pomegranate and noName, and their friendship. Lastly I would like to thank my girlfriend Cady Rodney for proofreading my lab reports, giving me creative ideas by questioning my work and being supportive through whole semester.

Executive Summary

This executive summary summarizes the research and competition to build an autonomous robot which navigates through various mazes using touch sensors, in the shortest time possible. The maze that the robots competed was approximately $1.5m^2$ in area. The teams were only allowed to use one handy board, two motors and two sensors. Wheels that had diameters longer than 4.5 centimeters were not allowed. This robot can be helpful in many different areas from autonomous vacuum cleaners to a robot for rescue missions which requires high precision. Although many problems caused by the Handy Board halted the progress of the project at times, the team still managed to adapt strategies and get a fair result and more importantly experience.

Before the lab sessions, construction process of the HandyBug, programming in Interactive C and basic principles of Handy Board was researched. The team was obliged to construct the robot in 2 full lab sessions and a short pre-competition preparing session.

Two alternative methods were discussed. One utilized the code "avoid_abstract.ic" doing pre-determined maneuvers when a collision was detected by the two front touch sensors. The other one utilized the code "metasens.ic" which had the same basic principle as "avoid_abstract" but this time the HandyBoard kept track of the collisions made. Depending on whether repetitive collisions on either one of the sensors were detected, the Handy Board performed a predetermined maneuver to avoid getting stuck in a point for extended periods of time.

The optimum solution for this design should be a reliable program that would be able to make approximately 90 degree turns so that the robot would not collide with the same wall repeatedly and manage to get away from it in one maneuver. The robot also should be able to get through the maze in the shortest time possible so it would not get stuck in any point. Additional codes are required to prevent the robot getting stuck in a corner so the "metasens.ic" mentioned above would be the optimum solution.

The construction process was divided into two, software and hardware. In the first lab session sample test programs were installed to test the HandyBug. Changes to design were made according to the preliminary test. In the second lab the two alternative solutions mentioned above were installed and tested. Hardware changes were made such as using the vehicle with 3 wheels to get the best performance out of HandyBug. Due to various malfunctions of the Handy Board, the team's testing chances have been limited. The testing was conducted using a sample maze and recording the time it took to complete testing or getting to the various points in the maze.

At the end of the construction and testing process there was an unknown maze in which teams had 2 trials. In the first trial the team competed poorly completing two checkpoints out of thirteen due to Handy Board malfunctions preventing proper preparation for the competition. However, in the second trial, with a few changes in the code, the team managed to complete 10 checkpoints showing the greatest increase in performance compared to their counterparts.

The preparation for this competition added valuable experience to the team both with how to do construction and the testing process. The old HandyBoard and batteries caused many problems to but the team managed to handle all these problems and gained more experience than the other groups. While dealing with these problems, the team was able to adapt smartly and quickly, increasing the robot's performance from one trial to another significantly. With all this experience, if given a better HandyBoard and new batteries the team will be able to build perfectly functioning robots with similar set-ups.

Problem Statement

The goal of this project was to design and construct a robot using HandyBoards, Lego, light sensors and/or touch sensors. This robot will be able to successfully compete in a tournament of Mint Shuffle competition.

List of Criteria

- Push at least two pucks per game to opponents area
- Do not cross to the opponent's territory
- Stop and Beep at the end of 90 seconds
- Robot must remain in tact
- If opponent pushed pucks, push at least one back

List of Constraints

- Only 1 HandyBoard and 2 motors per team
- Max of two touch sensors and two light sensors per team.
- One battery pack per team
- Robot cannot be bigger than the starting box

List of Assumptions

- Battery life is infinite(no power loss)

Research and Investigation

To better understand the competition, the credentials and necessary information for the competition were researched using the course website [2]. The credentials provided by the course website [2] outlined the rules and showed the instructions about the competition. Each team is allowed to design freely using light and touch sensors but the robot must be smaller in size than a box sized $10inch^2$ at all times. The usual length of the game is 90 seconds. Any interference with the robot during the game will cost the team 1 point. If a robot crosses over to an opponent's side the team is penalized 6 points. There are 3 pucks on either side of the game field. Each puck that is pushed to the opponent's side will give the team 1 point. If a team's robot pushes back a puck that the opponent pushed to their side, the team gains 1 point. Each Lab session competition is constructed like a league where a win is worth 3 points, a draw is worth 1 and loss is worth 0. The team with the highest amount of points wins.

Previous research for building touch and light sensor robots are also going to be required for this project. Research done for touch sensors (for competition 0) mainly utilized the textbook “Robotic Explorations” [1]. Construction process of the HandyBug, programming in Interactive C, creating methods, randomness function, and meta-sensing was introduced in Chapter 2: Sections 2, 3, 6 and 7 [1]. In addition to this valuable information, sensors and ports on HandyBoard were also included in these chapters.

Research for light sensors (for competition 1) were done in a broader manner including both Robotic Explorations [1] also online based sources [3]. In Chapter 2: Section 4, Braitenberg Vehicles were introduced. Further research was conducted using the World Wide Web and more information about Braitenberg Vehicles 1, 2a, 2b, 3a, 3b and 3c was found. These are all variation of positive and negative feedback from light and connections.

Due to the experience from competitions 0 and 1 with touch and light sensors, the team decided to take an approach without any sensors. In competition 1 the team used touch sensors to navigate through a maze and it was observed that if the bumpers are reliable, touch sensor are very good however light sensors are even more reliable. Due to complexity of the maze and random placement of the pucks mentioned in course and website [2] the team decided to use no sensors for the major part of the program and only use them to make the existing code more reliable.

Alternative Strategies

Alterative Solution 1

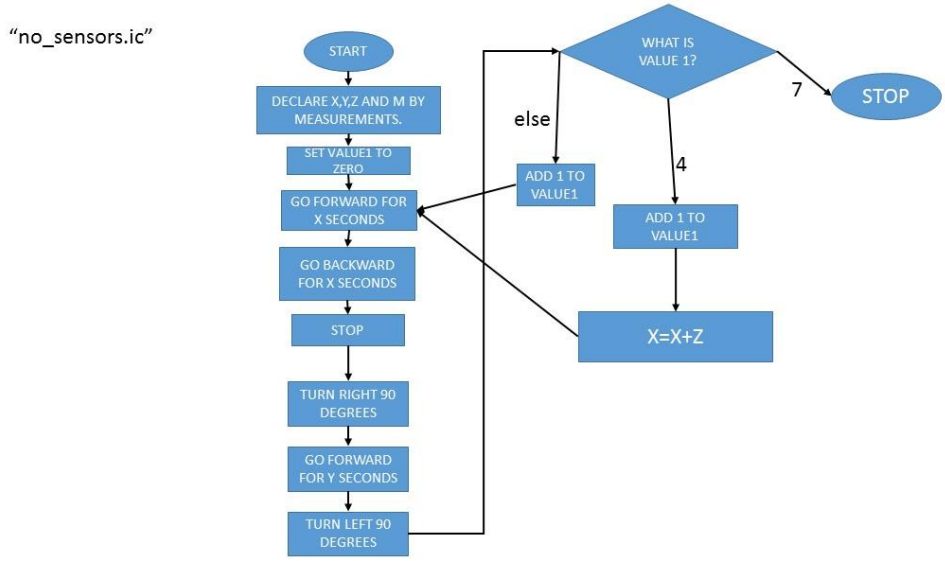


Figure 1:"no_sensors.ic" flowchart

The shape of the arena is consists of two sections of different lengths for each team and a wall surrounding the arena. The robot uses no sensors and is preprogrammed to perform all its actions. In the flowchart above, "X" symbolizes the time to reach the short end of your side of the arena, "Z" is the time difference between times required to reach shorter side of the arena and the long side. What the robot does is simple and completely predetermined. The robot goes forward for the time required to reach to the end of the designated side of the arena, then it moves all the way back. It performs a 90 degree turn to the right, goes forward for a bit to align itself with a new path, turns 90 degrees to the left to align straight with the end of the arena and performs this function again. When the required number of maneuvers (represented as value 1) is met, the maneuver above is performed to complete a section of the same length in the arena, the robot adjusts itself for the new length of the field and repeats this process again. When the required number of times for the robot to finish the arena is reached the robot stops. This way the robot covers the whole maze and has a high chance of pushing pucks and a low chance of crossing opponent's territory, and also stops at the end of 90 seconds.

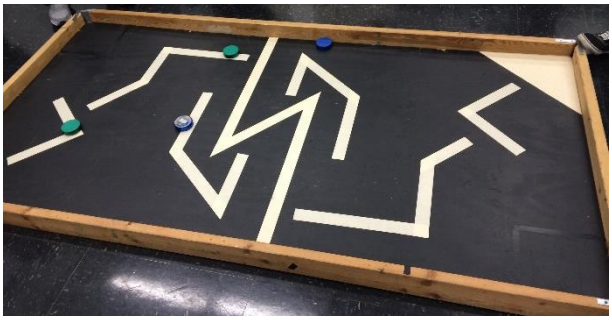
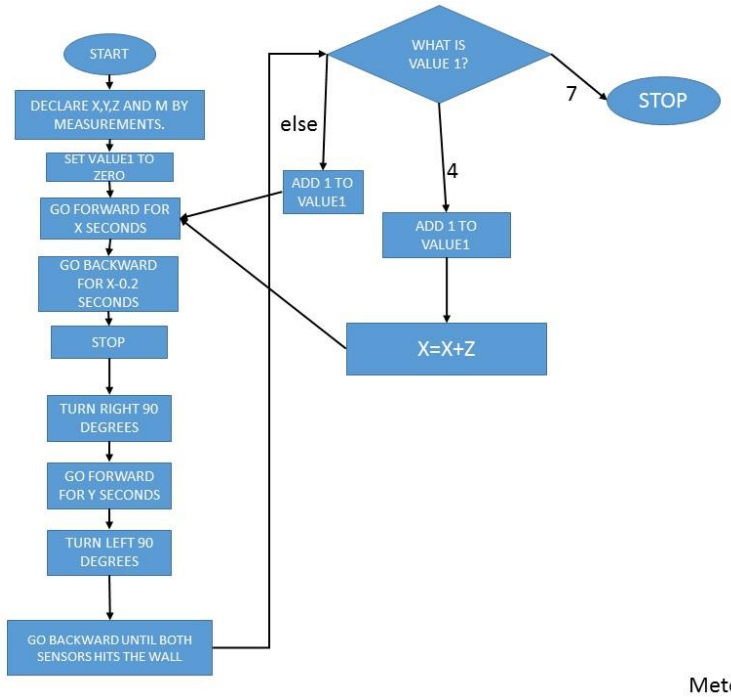


Figure 2:The shape of the arena

Alternative Solutions 2

"no_sensorstouch.ic"



Mete Morris

Figure 3:"no_sensorstouch.ic" flowchart

This solution is same in principle with the “no_sensors.ic” mentioned above but there are some additional features. The robot has two touch sensors and bumpers at the back and the code is adjusted so that these sensors could be used for additional stability.

The robot does the exact same maneuvers but this time after going forward and coming back the robot leaves a bigger distance between itself and the wall. After this it travels straight backwards with high power and collides with the wall going back until both sensors are pressed. This makes sure the robot is facing straight to the opponents field and is travelling in a straight line.

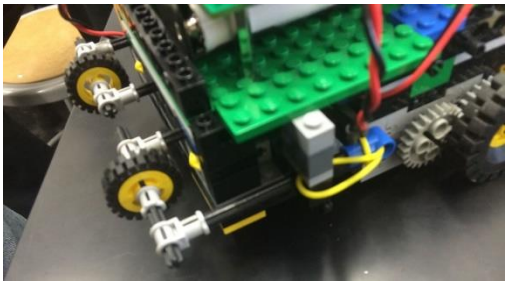


Figure 4: Bumpers and touch sensors placed at the backside of the robot

This helps the robot to stay on track as there are no sensors used for finding a path and everything is pre coded. This also decreases the chance crossing to opponent’s side under unexpected conditions.

Except from the touch sensors, there were not any changes in the robots hardware compared to alternative solution 1.

Alternative Solution 3

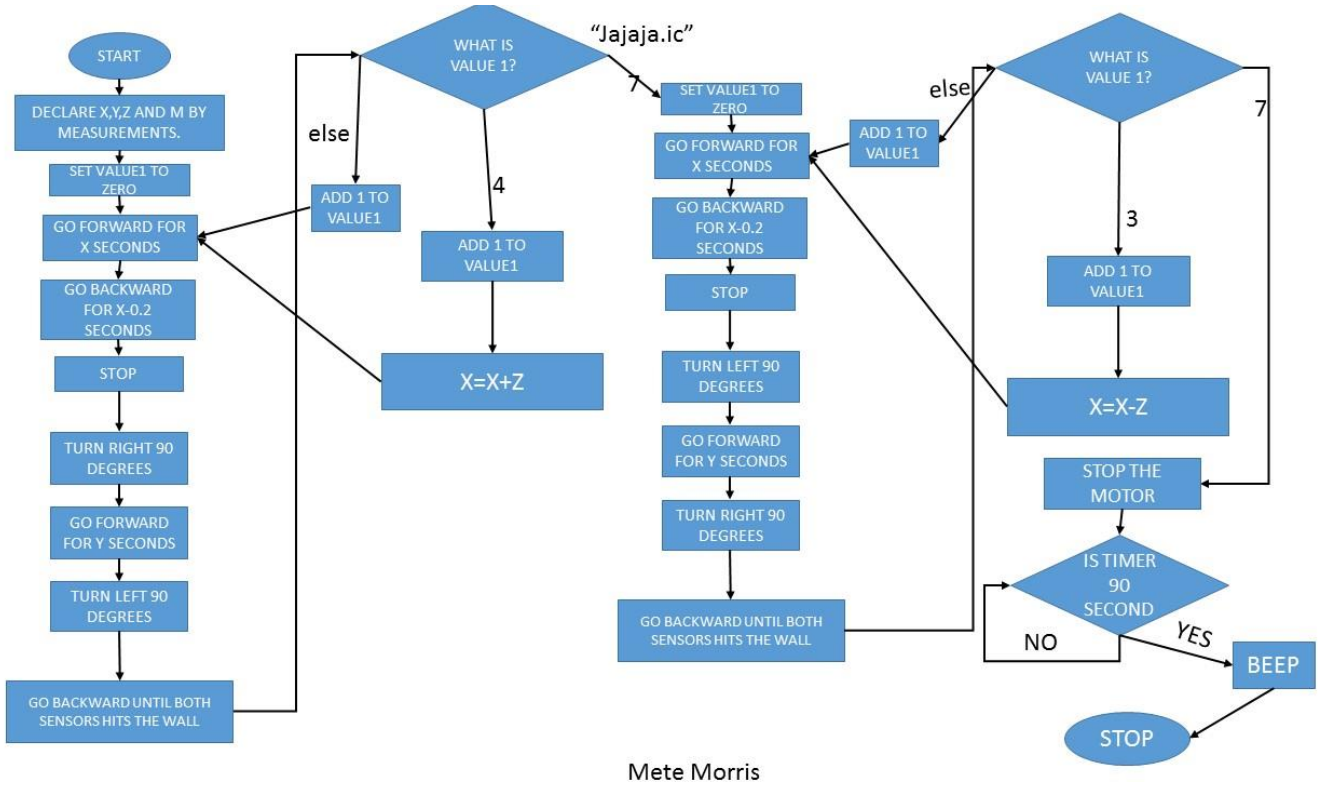


Figure 5: "jajaja.ic" flowchart

This solution follows the same basic principle as alternative solution 2("no_sensorstouch") but this time instead of covering the whole maze once the robot tries to cover the whole maze twice. The team came up with this idea due to the realization that going once through the maze does not average 2 pucks a game. This way some of the pucks that might have been missed by the robot while going forward through the maze can be pushed back while going backwards.

There were also some hardware changes with the robot. The robot has two clutches in front to help the team grab pucks better. These clutches have been equal in length but with the new design the clutch on the right side of the robot is shorter which helps the robot to grab the pucks better. In addition to this the team added rubber bands to hold the motor sensors plugs in their ports as they sometimes got dislocated due to the robot clashing with the wall fast to activate the sensors.

The last addition to this robot was the timer. The code for the timer was found using the textbook [1] and it was installed so that the robot beeps after 90 seconds as it is supposed to. Timer was not needed to stop the motors as the robot completes the whole program under 90 seconds.

Optimum Solution

The optimum solutions for this competition was alternative solution 3. The team aimed to push at least two pucks per game, avoid crossing to the opponent's side and stopping and beeping at the end of 90 seconds. All three codes which derived from each other have proven to be fairly consistent with not crossing to opponent's side but added touch sensors in solution 2 and 3 made sure that the robot does not cross to the opponent's side, so this makes them superior to solution 1. Both solution 2 and 3 averaged around pushing 2 pucks per game to opponents field but solution 3 had the higher average and by doing two rounds it also pushed the pucks the opponents has pushed to team 1's side preventing the opposition from getting any points. All three solutions stopped before designated 90 seconds but solution 3 was the only one to beep at 90 seconds.

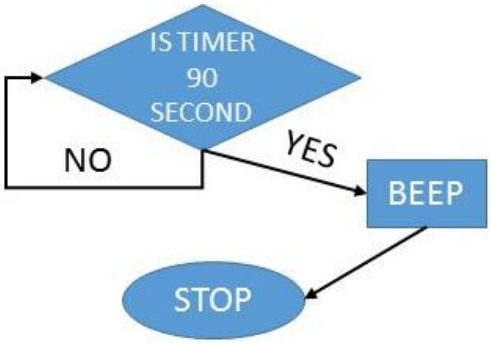


Figure 6: Timer code in "jajaja.ic"

With all these software advantages in mind, the hardware improvements also made solution 3 superior. Rubber bands prevented the cables falling out of their sockets which was a big issue with alternative solution 2.

The clutches in solution 1 and 2 sometimes pushed some pucks away from the robot causing some complications. In solution 3 the robot grabs the pucks much better thus enhancing the performance of the robot.

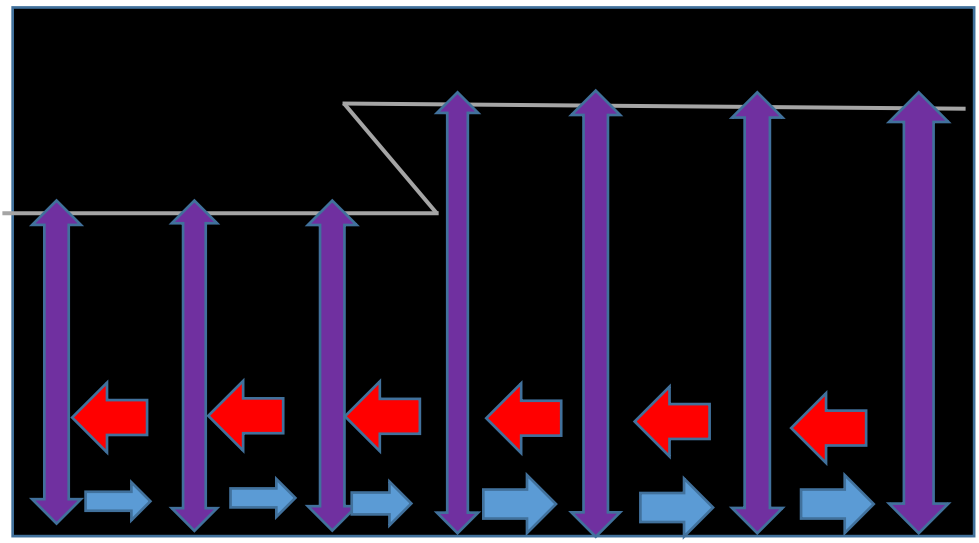


Figure 7: The motion of "jajaja.ic". Blue arrows represent motion done while going forward. Red arrows represent motion done while going backwards. Purple arrows represent motion done while both going forwards and backwards

Construction and Implementation

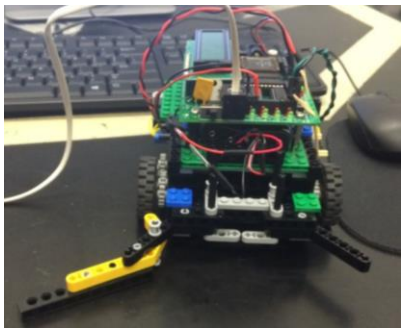


Figure 8: Front view of the robot

The robot was three wheeled with two wheels up-front and one wheel in the middle in the back. The robot was front wheel drive which helped the robot to turn much more precisely. The clutches in front of the robot helped the robot to get pucks and the advantages of one clutch being shorter has been discussed in optimum solution.

The main maneuver of the robot as shown in figure 9, is to go forwards and come backwards, turning left at first, going straight and then right at 90 degrees and repeating this entire motion again. Using this cycle, the robot covers the whole arena. There are different sections of the maze with different lengths. As shown on page 22 of the appendix, the robot adjusts going forward and backward time according to number of maneuvers done. The number of maneuvers required to complete the sections are pre-measured. When the robot finishes one round in the maze, it starts going backwards (also shown in figure 7). The only thing that changes going backwards is the direction of the turning. In figure 7 it shows that the robot is first turning right before going straight but while going back through the maze it turns left first.

The most challenging part of this code is the variables (listed in page 18 in appendix). The robot solely runs on variables which determines how long the robot should go straight in different sections, number of maneuvers in different sections, sleep time required to make 90 degree turns. When these are measured correctly the robot circulates through the whole maze and stops.

There also is a timer for the robot which makes the robot beep after 90 seconds. There are methods for going forwards, backwards, turning right, left, and finally there is method which tests whether both touch sensors are pressed or not.

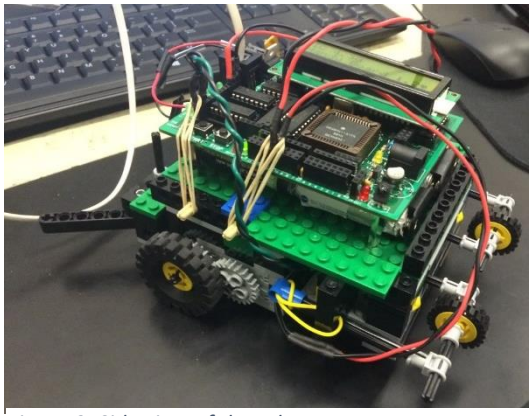


Figure 9: Side view of the robot

```
while(completecheck<=nummanouver)
  {
    forward();
    sleep(timetakenlong);


    backward();
    sleep(timetakenlong-offsetback);


    right_rotate();
    sleep(turnvalueturn);


    forward();
    sleep(turnvaluestr8);


    left_rotate();
    sleep(turnvalueturn);

    straight();

    completecheck++;
}
```

Figure 10: main general maneuver of "jajaja.ic"

Analysis and Testing

Due to most of the robot motion being pre written and dependent on variables, there was an excessive amount of testing required. The robot needed to turn 90 degrees in certain parts of the program so the team measured the sleep time for turning 90 degrees. After the team measured this time, they also measured the time required for the robot to complete different sections in the arena (table 1). The average of these values were used in the code. The team first estimated the number of maneuvers for each section and after a few trials these values were determined as well.

Time taken to Complete Short(s)	Time Taken to Complete Long(s)
2.80	3.25
2.95	3.35
2.85	3.35
2.90	3.25

Table 1: Time in seconds for reaching the end of designated area in different sections:

Sleep time for turning(s)	Result
0.5	Turns less than 90 degrees
0.6	Turns more than 90 degrees
0.55	Turns approximately 90 degrees

Table 2: Different sleep time for turning and their effects

Table 1 and Table 2 shows the results obtained in the first lab. The team has used many different HandyBoards and battery packs and these values changed as a result, but differences were usually minimal. The first lab mainly consisted of testing these values and assembling the robot.

In the second lab after some small changes were done on code with variables, the team focused more on testing. Due to observations the team added touch sensors in the back and installed the final solution “jajaja.ic”. The improvements in hardware and software from alternative solutions 1&2 to alternative solution 3 boosted the robots performance. The team mainly tested the robot with putting pucks in different positions and letting the robot run and score as many pucks. The results for some of these test drives are shown in table 3.

Trial	Result
1	1 puck in two in the line
2	2 pucks in 1 missed
3	3 pucks in
4	3 pucks in
5	2 pucks in 1 on the line
6	3 pucks in

Table 3: Test trials for team 1

Before the competition in class, the team added the timer for beeper to the robot. During the league competition the robot and the code was extremely successful going unbeaten against all teams. The only lost point was due to battery drainage which prevented the robot from pushing forward enough. The team finished the class league in first place and went on for class finals. During the semi-finals the robot collided with team 2's robot on the border and went out of its path and crossed over the line. The robot is very vulnerable to collisions.

Match	**Result**
Against Team 2	Won Scored 2 pucks
Against Team 6	Won Scored 2 pucks
Against Team 5	Won Scored 3 pucks
Against Team 4	Drawn Scored 0 pucks
Against Team 3	Won Scored 2 pucks
Lab Semi Final	Lost, Crossed over due to collision

Table 4: Class tournament results for Team 1

The robot qualified for overall finals between lab sessions. It performed extremely well winning all three final games and winning the competition. The tense semi-final competition has shown an extreme performance where the robot scored 3 pucks and also pushed 3 of the opponents pucks back.

Competition Stage	**Result**
Quarter Final	Won Scored 2 pucks
Semi- Final	Won Scored 3 pucks, pushed 3 pucks back
Final	Won Scored 2 pucks

Table 5: Final tournament results for team 1

There are not too many future improvements besides making the robot more resistant to collisions. This way, the sole weak point of the robot could be addressed.

Final Evaluation

Averaging two pucks a game, consistently not crossing to opponent's side and stopping and beeping before 90 seconds was the team's main objectives. The team and robot were very successful in these aspects and managed to win the competition while performing very well.

The key design features are the unsymmetrical clutches which enhances the grabbing pucks, touch sensors and bumpers placed at the back of the robot that help the robot remain in a straight line, the front wheel drive which enhances turning and the code which makes the robot go through the maze twice.

The robot performed very well in all competitions only losing one game and drawing one out of 9 games. Regarding future improvements, the main upgrading would be the sensitivity of the robot towards collisions. The only game the robot lost was due to a collision on the line. Additional touch sensors could make sure the robot stays on path after colliding.

The robot was highly successful and it was extremely consistent. The team blended in very well and with hard work managed to win the competition with an impressive display on the finals. With additional funding and touch sensors the robot could be resistant to collisions and would be able to find the path easier.

Appendix

```
//Code of competition winners in ece 100 competition 2. Tunable parameters
are listed below.

//These code is aimed to movepucks in a special arena

//Authors: Mete Morris, Shuhei Aoki, Zhaoyin Qin

//declaring motor numbers and ports

//last edited: 12/03/2014

int RIGHT_MOTOR= 0;

int LEFT_MOTOR=  3;

int LEFT_TOUCH = 10;

int RIGHT_TOUCH = 11;

//these are all tunable parameters

float timetakenlong = 2.07;

float difference    = 0.58 ;

float turnvaluestr8 = 0.31;

float turnvalueturn = 0.62;

float offsetback    = 0.39;

int   nummanouver   =    3;

int   completecheck =    0;

int   isstraight    =    0;

int i = 0;

int nummanouverback = 3;


//declaring methods for turn so that it will be easier

//right turns

void right_rotate()
```

```
{
    motor(LEFT_MOTOR,100);
    motor(RIGHT_MOTOR,-100);



}
//left turns
void left_rotate()
{
    motor(RIGHT_MOTOR,100);
    motor(LEFT_MOTOR,-100);
}
//going forward
void forward()
{
    motor(LEFT_MOTOR,50);
    motor(RIGHT_MOTOR,50);
}
//going backwards
void backward()
{
    motor(LEFT_MOTOR,-50);
    motor(RIGHT_MOTOR,-50);
}
//stopping
void stop()
```

```
{
    off(LEFT_MOTOR);
    off(RIGHT_MOTOR);
}

//code for touch senors making sure  the robot is straight
void straight()
{
    isstraight = 0;
    while(isstraight==0)
      {
        if(!digital(LEFT_TOUCH))
          {
            motor(LEFT_MOTOR,-80);
        }else{
            off(LEFT_MOTOR);
        }
        if(!digital(RIGHT_TOUCH))
          {
            motor(RIGHT_MOTOR,-80);
        }else{
            off(RIGHT_MOTOR);
        }
        if(digital(RIGHT_TOUCH)&&digital(LEFT_TOUCH))
          {
            isstraight=1;
```

```
        }



    }

}
//main function


void main()
{



    while(completecheck<=nummanouver)
      {
        forward();
        sleep(timetakenlong);



        backward();
        sleep(timetakenlong-offsetback);



        right_rotate();
        sleep(turnvalueturn);
```

```
        forward();
        sleep(turnvaluestr8);



        left_rotate();
        sleep(turnvalueturn);


        straight();


        completecheck++;
    }



    while(completecheck > nummanouver && completecheck<=((nummanouver*2)-
2))//number of manouvers needed for long part obtained by experiment
      {
        forward();
        sleep(timetakenlong+difference);//this is where robot moves into a
new section so going forward value changes



        backward();
        sleep(timetakenlong+difference-offsetback);



        right_rotate();
```

```
    sleep(turnvalueturn);


    forward();
    sleep(turnvaluestr8);



    left_rotate();
    sleep(turnvalueturn);


    straight();


    completecheck++;
}
//going back


while(i< nummanouverback)
  {
    i++;
    forward();
    sleep(timetakenlong+difference);



    backward();
    sleep(timetakenlong+difference-offsetback);
```

```
    left_rotate();
    sleep(turnvalueturn);

    forward();
    sleep(0.40);



    right_rotate();
    sleep(turnvalueturn);

    straight();

    completecheck++;
}
i = 0;
while(i<nummanouverback)

  {
    i++;
    forward();
    sleep(timetakenlong);

    backward();
    sleep(timetakenlong-offsetback);
```

```
        left_rotate();
        sleep(turnvalueturn);

        forward();
        sleep(0.40);

        right_rotate();
        sleep(turnvalueturn);

        straight();

        completecheck++;

    }

    stop();
    //below is the beeper for the timer
    while(1){
        if(seconds() >= 90.0){
            tone(700.0,4.0);
            stop();
        }
    }
    //sleep(5.0);

}
```

References

[1] Martin, Fred G. 2001. Robotic Explorations: A Hands-On Introduction to

Engineering. New Jersey: Prentice Hall.

[2] Flueck, Alexander J. 2012. ECE 100 [online]. Chicago: Illinois Institute of Technology, Electrical and Computer Engineering Department, 2005[cited 30 August 2012]. Available from World Wide Web;

(http://www.ece.iit.edu/~flueck/ece100)

[3] Thornton, Chris. "AI Lecture: Braitenberg Vehicles." *AI Lecture: Braitenberg Vehicles*. University of Sussex, n.d. Web. 10 Nov. 2014.