



ECE 100

Design Portfolio

Alan Palayil
Prof. Erdal Oruklu/12/6/2019



Table of Contents

Acknowledgements.....	4
Executive Summary	5
Problem Statement.....	6
Judging Criteria.....	6
Speed.....	6
Stability	6
Efficiency.....	6
Consistency	6
Constraints	6
Competition 0.....	6
Competition 1.....	6
Competition 2.....	6
Assumptions.....	6
Research and Investigation	7
Figure 1 Metasens.ic:	7
Alternative Solutions	9
Competition 0.....	9
Table 1 Metasens responses:.....	9
Competition 1.....	9
Table 2 Example of normalize motor speeds:.....	9
Figure 2 Normalize:	10
Figure 3 Lightfinal.ic:	10
Competition 2.....	11
Figure 4 PuckBot.ic:	11
Optimum Solution.....	12
Competition 0.....	12
Figure 5 HandyBug Design 0:	12
Competition 1.....	12
Figure 6 HandyBug Design 1:	13
Competition 2.....	13
Figure 7 HandyBug Design 2:	14
Construction/Implementation	15

Figure 8 Stages of Construction.....	15
Analysis and Testing.....	16
Table 3 Trials and Precision example:.....	16
Final Evaluation.....	17
Objective.....	17
Key Design.....	17
Future Work.....	17
Conclusion.....	17
Appendix.....	18
Code PuckBot.ic.....	18
References.....	20

List of Figures:

Figure 1 Metasens.ic:.....	7
Figure 2 Normalize:.....	10
Figure 3 Lightfinal.ic:.....	10
Figure 4 PuckBot.ic:.....	11
Figure 5 HandyBug Design 0:.....	12
Figure 6 HandyBug Design 1:.....	13
Figure 7 HandyBug Design 2:.....	14
Figure 8 Stages of Construction.....	15

List of Tables:

Table 1 Metasens responses:.....	9
Table 2 Example of normalize motor speeds:.....	9
Table 3 Trials and Precision example:.....	16



Acknowledgements

I might want to start by expressing gratitude towards Professor Erdal Oruklu taking his time in educating the ECE 100 course and being a prominent figure in the Electrical and Computer Engineering office at IIT. The value given to the student's inquiries and his quick responsiveness in both lectures and laboratories is really appreciated. I am appreciative for the various visitors and guests he had solicited to exhibit part of their life and the different opportunities that are awaiting on our future careers. Professor, I whole heartedly thank you.

I might also take the opportunity to offer my gratitude to my Teacher's Assistant, Mr. Rafael A. Perez. I would like to thank him for the numerous laboratory's sessions, where he would give us subtle indications about how to build our robots and code, and even broaden more help when inquired. I might likewise want to him for being such a motivation, a glorious understudy of the field of Electrical and Computer Engineering and one to gaze upward to. Much obliged to you, Rafael.

I can't resist the opportunity to demonstrate my appreciation to the all my past laboratory accomplices. From Competition 0, thank you Mr. Rezwan and Mr. Matthew O. Even though we didn't win, it was a joy working with both of you. From Competition 1, thank you Mr. Raymond and Mr. Wedge. The thoughts exhibited in this gathering were a long way past the basic level. At long last, from Competition 2, thank you Mr. Matthew G. and Mr. Matthew V. The ideas and hard effort contributed by both of you were what enabled us to be fruitful in this Competition. I am incredibly appreciative for having been able to learn about everyone a little, the opportunity to grow with each new encounter. I greatly appreciate it.

At long last, I might want to thank my folks for the innumerable years of help.



Executive Summary

The uses of mechanized autonomous robots combined with touch sensors demonstrate to include ample importance. From applications running from, for instance, the utilization for military field activities or navigations, robots with touch sensors are a viable mix in the present society. In this way, given a challenge to structure a successful robot, these applications give essentialness when the most relevant robot is required. The given proposition lives in exploring a self-assertive maze utilizing a LEGO robot named the "HandyBug" outfitted with servo motors, self-ruling reactive code, and guard sensors in a planned setting.

Given the development of the maze, a progression of right-angled turns was often anticipated. This plan in the maze considers the HandyBug to explore its surroundings with the utilization of two touch sensors (one for left contacts and the other for right contacts). The HandyBug's capacity to explore the maze effectively relies upon both a powerful and proficient code, and hardware that is responsive and fast. On account of the HandyBug's design, this requires the utilization of a gear ratio that advances speed, and touch sensor placement that reacts to touch sensitivities. In like manner, the code driving the HandyBug must be reliable and persistent while having the capacity to distinguish when it is trapped. These limitations present requesting, yet energizing, challenges to be fruitful in the Competition.

The programming of the robot took place in stages. The first stage was to program the robot to go forward and backwards. The second stage was to enable the touch sensors to the Handy-Board, and when the robot hits an obstacle, it should change its direction and go the other way. The third and the last stage was to add intelligence to the robot, during which if the robot got stuck in a corner how could it remove itself from that situation. To tackle this stage, the team came up with three different solutions, the first solution was to make the robot use meta-sense to make a sharp 90 degree turn. This solution could only make the robot turn only in one direction. The second solution was to make a random turn. The third solution was to use both the meta-sense and random function. The program was tested during each stage and each solution was tested as well.

For executing complex tasking through small functions or steps. During any situation in which the solution became more unpredictable. Since the testing is in maze. After using the third solution as the approach to solve a complex problem through meta-sensing. The optimum solution allowed the robot to bump into the walls and avoid the obstacles properly without getting stuck in a corner. The code prevented the robot from turning the exact amount each time which result in it getting stuck, instead it made smaller turns which will allow it to slowly make its way out of the maze instead of using brute force.

During these three weeks the team constructed the robot. The robot was constructed using LEGO Technic blocks, 2 touch sensors, 2 servo motors, and Handy-Board. The LEGO blocks were used to keep the cost low for the prototype and since it's easy to modify the bricks. The team decided to use the third solution as the base of the algorithm for the robot. The robot was tested on a maze to check if the robot could solve the maze. The maze consisted of sharp right angle turns. The robot couldn't make it off the maze, and the team figured out the reason. The random function made the sleep time change which made the robot turn 180 degree or 360 degree turns. The team improved the code for future references by removing the random function and adding a 270 degree turn once the robot hit both the sensors consecutively getting activated.



Problem Statement

To build up an autonomous robot that will contend in different competitions, concentrating on the utilization of touch and light sensors.

Judging Criteria

Speed

How quickly the robot can complete the task due to the addition of time constraints

Stability

Stable structure means the parts don't tear off and the robot can complete its task intact.

Efficiency

The structure of the code or design of the robot determines the efficiency.

Consistency

The precision of the robot to successfully perform during the competition.

Constraints

Competition 0

Navigate a maze within 90 seconds using two touch sensors.

Competition 1

Follow a tape line and complete the maze within 90 seconds using two light sensors.

Competition 2

Compete in "Mint Shuffle X", a game which resolves in moving pucks either using both light and touch sensors, with an allotted time of 90 seconds. The robot must turn off within 90 seconds of starting.

Assumptions

To be as efficient and successful in all competitions as possible, several assumptions were made. These include generally constructing the robot with parts ensuring it to be fast, maintaining a constant state of charge in the battery pack, and ensuring a consist achievements in testing trials.

Research and Investigation

The utilization of autonomous robots can be amazingly gainful to the organization's prosperity – particularly with the proof from the research directed in the lab sessions. The focal unit of the HandyBug comprises of the HandyBoard, an automated controller that houses the Motorola 68HC11 chip, 32K of primary system memory, inputs for analog and digital sensors, output drivers for four DC motors, and a battery-powered battery pack.

[1] The HandyBoard can be customized in interactive C, a C-based language that was written specifically for educational robotic applications.

[2] The HandyBoard can likewise be outfitted with motors.

[3] To promote its capabilities, the HandyBug can exploit a variety of sensors, for example, touch and light sensors. The light sensors, for instance, are cadmium sulfide photocells, or ordinarily known as CdS photocells, whose resistance changes as per the measure of light it gets.

[4] When all joined, the HandyBoard builds up its own aptitude to cooperate with its surrounding conditions.

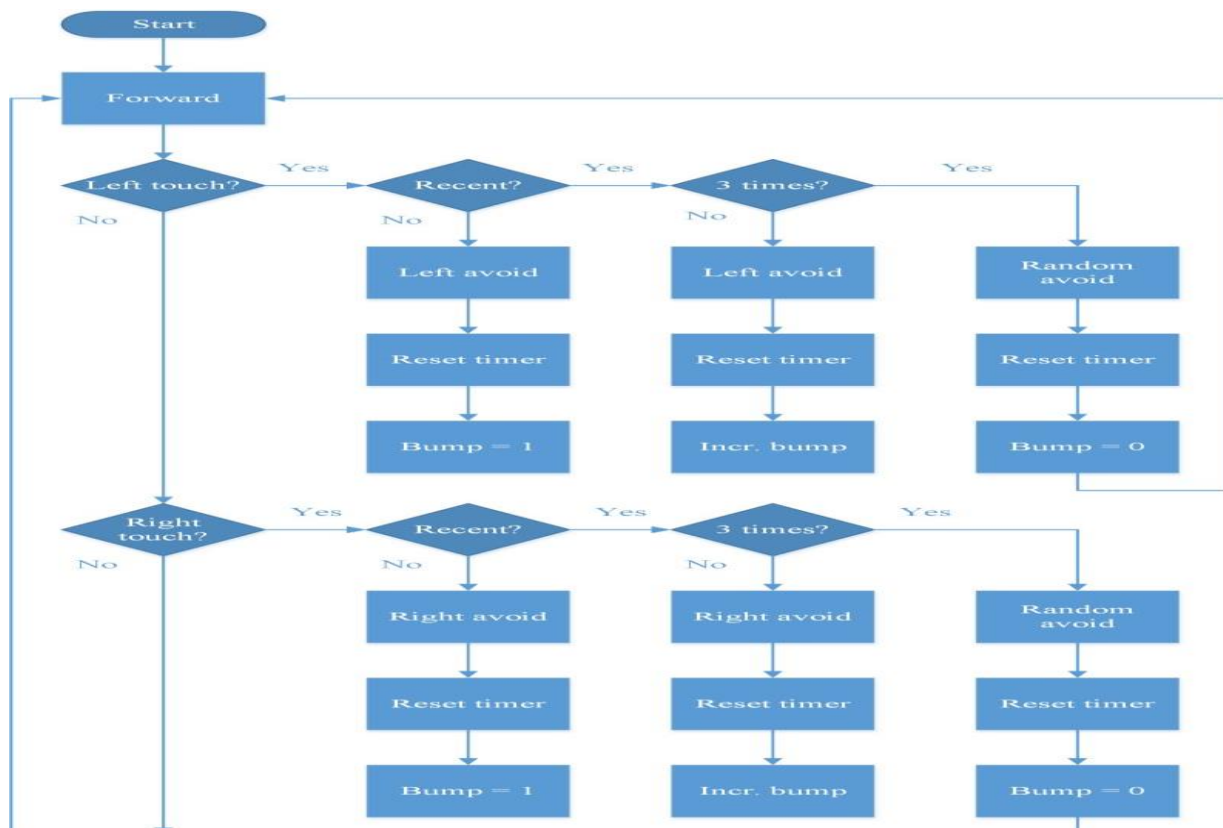
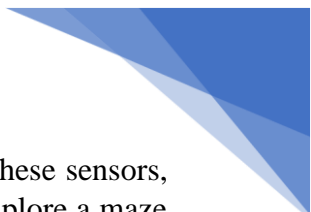


Figure 1 Metasens.ic: metasens.ic enables the HandyBug to systematically avoid walls to its left and right and initiate a “random avoid” after a set number of bumps accumulate. (Martin 2001, p. 91)



Competition 0 introduced a point by point see in the capacities of touch sensors. These sensors, combined with prescient and fitting intelligent C code, enabled the HandyBug to explore a maze organized by right-calculated dividers. For instance, metasens.ic is a technique that spotlights on utilizing timers to decide how to the HandyBug should move about in the maze. On the off chance that the HandyBug touches a divider to one side in a genuinely late measure of time (seconds), at that point the robot ought to maintain a strategic distance from that specific divider by moving right. This cycle reiterates itself until the HandyBug escapes from the maze. Specifically, the utilization of the touch sensors given the assignment is genuinely reasonable. They are dependable, compelling, and proficient in the route of completing the maze, and considerably more fundamentally they offer financially sharp response for the organization's prosperity (these sensors can be completed effectively and inexpensively).

Significantly more in this way, Competition 1 concentrated on the capacities of light sensors, explicitly their capacity to increase the HandyBug to follow a tape way. These sensors have a most extreme light estimation of 16, and least of 194.

[5] They can likewise be modified for use from multiple points of view, however most eminently include: normalize, an ability proposed by Fred G. Martin to change over light values read into suitable motor speeds, and basically testing a scope of light values that are then coded to adjust motor speeds.

[6] While normalize is effective for line following, there are focal areas for substitute alternative; for example, the capacity to be increasingly swifter while following the line. The after effects of Competition 1 appeared, in any case, that the normalize function was a viable method for following the tape way.

Alternative Solutions

Competition 0

Metasens.ic presents a successful and orderly approach to explore the maze. It utilizes legitimate if-else statements to decide the direction and where the HandyBug should turn. The points of interest lie in its free nature. For instance, regardless of where the HandyBug is put inside the maze, it will eventually discover out its way. Nonetheless, the disadvantages include the inability to complete the maze in the time given. This is on the grounds that the HandyBug has the plausibility of pivoting and leaving through the entry passage.

Wall Hit	Frequency	Response
Right	Not recent	Turn left
Left	Not recent	Turn right
Right	Recent – 4 times	Turn randomly
Left	Recent – 5 times	Turn randomly

Table 1 Metasens responses: An example of responses initiated in metasens.ic. If the HandyBug hits the same wall 5 times it should turn randomly to find a new path.

WallFollow.ic is a case of thought that uses the idea that in the event that one is following a divider, either right or left, at that point in the long run they will arrive at the end of the maze expecting it has just an entry passage and exit. This strategy has almost a 100% achievement rate as long as the HandyBug begins at the entry passageway and the maze has just one exit. Prime focal points of this strategy stay remains in its dependable and ensured method. The disadvantages lies in not just certain beginning conditions, for example, starting at the passageway, yet in addition which divider is picked. For instance, on the off chance that the correct wall has a left length to the completion, at that point the left divider, anyway the right divider is picked, at that point this significantly expands completing time.

Competition 1

Level of Brightness	Value Read	Motor Output (Normalize)
Dark	194	0
Very Bright	16	100
Somewhat Bright	45	70

Table 2 Example of normalize motor speeds: Values read by the sensors are changed over into proper motor speeds given by the normalize function. These values additionally differ in their degree of splendor. For this situation, the darkest worth read yields an engine speed of 0 while the most splendid yields an engine speed of 100.

Lightsens.ic crosses a tape-path by reading light values affected by the tape and setting suitable motor speeds. It uses *normalize*, a condition to change over sensor readings to motor speeds.

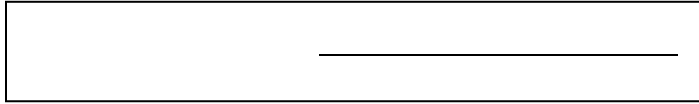


Figure 2 *Normalize*: The *normalize* function defined by Fred G. Martin (Martin 2001, p. 84).

This is exceptionally accurate as long as the light values coded are precise to the sensor's readings. In any case, the precision of the readings modifies with the lighting of the room. For instance, if the most extreme light coded is 14, anyway certain lighting of the room changes this incentive to 35, at that point the HandyBug won't execute as planned. Values additionally change depending upon whether the sensors are protected or shielded (a technique to acquire as exact a perusing as could reasonably be expected). Basically, these variables enormously decide on the achievement of the line following.

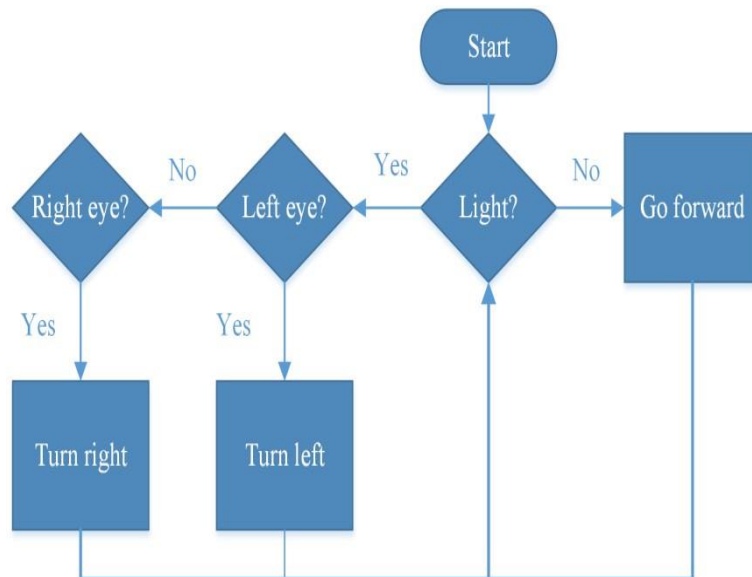


Figure 3 *Lightfinal.ic*: *Lightfinal.ic* makes use of thinking by following the ground instead of the tape.

Lightfinal.ic utilizes irrational speculation to navigate the tape-path. That is, rather than following the light tape-way legitimately, it rather pushes ahead when detecting the darker area surrounding the tape. This has various advantages, for example, the capacity to cross the tape-path more rapidly and diminishing sensitiveness to readings impacted by the lighting of the room. It isn't without disadvantages, nonetheless. Some of which requiring the HandyBug to begin over the tape the HandyBug moves ahead as long as the ground stays dim. Issues also happen with sharp angled turns, as the sensors must be put outwardly of the HandyBug to suit the width of the tape. On the

off chance that the HandyBug attempts to make a turn and the two sensors read light values, at that point the HandyBug must have the option to modify its direction.

Competition 2

Combined.ic is a case of utilizing the past code, for example, `metasens.ic` and `lightfinal.ic`, to build up a code that will be effective in Competition 2. This demonstrates to be very testing given the extent of the trials for this challenge and requires the utilization of extremely unpredictable if-else proclamations combined with strategies that differentiate each segment of the game board. For instance, the first method would utilize both light and touch sensors to get the main puck into the objective, trailed by another strategy that moves different pucks to the adversary's side of the field, and afterward guards the isolating line. This demonstrates to be perplexing and hard to test with the measure of lab time given.



Figure 4 PuckBot.ic: A flowchart describing the simple nature of PuckBot.ic. Here, the HandyBug will follow the series of steps until 90 seconds is reached, where it will then stop.

PuckBot.ic displays an altogether unique method for making progress in Competition 2. This strategy centers exclusively around the utilization of hard coding (a technique that uses a fixed information and data to get a particular output) the HandyBug to move to every puck in a precise and systematic way. The best advantage of this method has is its consistency and productivity. A few trials were finished where a similar result was accomplished in various trials. The disadvantage, be that as it may, shows up if the pucks positions and stage direction changes along with changes on the board, at which the strategy turns out to be totally futile. In any case, the given rules and principles of Competition 2 this was not an issue.

Optimum Solution

Competition 0

The ideal answer for this challenge is metasens.ic because of its ability to be as free as it could be allowed. The algorithm utilized enables the HandyBug to explore the maze inside within 90 seconds as the sensors are activated when contacting the dividers. This is critical in light of the fact that the absence of sensor initiation could bring about the HandyBug pushing ahead until a timer advises it should make an arbitrary turn. Basically, metasens.ic impacts the HandyBug to be brisk with its route. It also serves to be effective in navigation. That is, the HandyBug can finish the maze, anyway being consistent is an issue. A few trials were finished where the HandyBug was not able complete the maze, for the most part by leaving through the entry, anyway numerous trials sensor activation was an issue. This shouldn't imply that the solution isn't ideal, anyway with additional time a refined the code for exact navigation can be created.

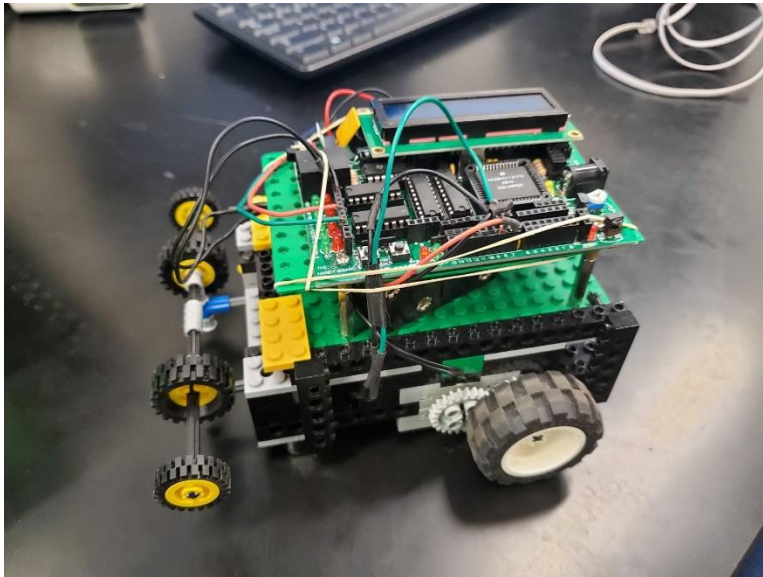


Figure 5 HandyBug Design 0: Competition 0 HandyBug with bumper to retain hits.

Competition 1

Lightfinal.ic presents itself to be the ideal solution for Competition 1. In spite of a large number of the inconveniences introduced before, the arrangement is very powerful. Since the surface territory to peruse dark values is greater than that of the light from the tape, the HandyBug moves a lot quicker than in lightsens.ic. This solution is additionally impressively effective and reliable; to such an extent that few trials were fruitful when moving along the tape-path. Be that as it may, issues emerged when crossing tape situated in sharp angles. This was expected, particularly since the situation of the sensors were fixed (roughly 3 ½ "), anyway lab time restricted the amount of

trials. Given additional time, conditions could be executed to deal with sharp angles with better accuracy, in-general expanding consistency for future trials.

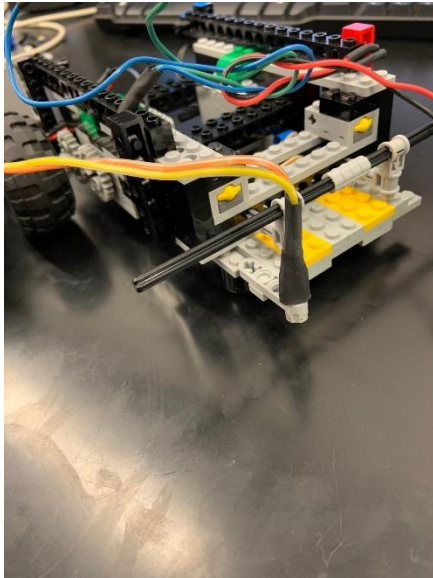


Figure 6 HandyBug Design 1: Competition 1 HandyBug with custom bumper retaining light sensors.

Competition 2

Competition 2 demonstrated to be the most energizing yet, basic, of all the ideal solutions exhibited up to this point. PuckBot.ic is the ideal solution for Competition 2 due to its consistency, effectiveness, and speed. Various trials were done in the lab and the greater part were effective. This present solution's splendor lies in its straightforwardness – it comprises of just the motors and sleep functions that control every steep. Given the intricacy of this Competition and considering the field doesn't change altogether, coding with sensors is a suitable choice, however an answer like PuckBot.ic is substantially more effective.

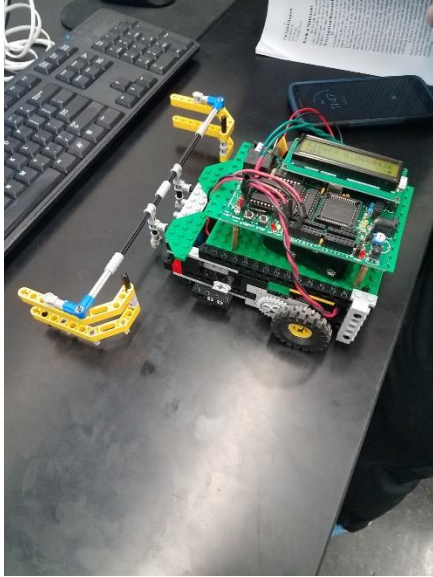


Figure 7 HandyBug Design 2: Competition 2 HandyBug with claws to retain pucks.

Construction/Implementation

All through every one of the competitions a typical box-molded theme was applied for the structure of the HandyBug. Two wheels were utilized in all the competitions including Competition 2. Competition 0 considered two servo-motors and two touch sensors, while Competition 1 took into consideration two servo-motors and two light sensors. Competition 2 considered the utilization of the two servo-motors, and both types of sensors, anyway for the development of PuckBot.ic the sensors were ignored as their utilization was superfluous. Rather, just the motors and modified bumpers (for holding pucks) were used.

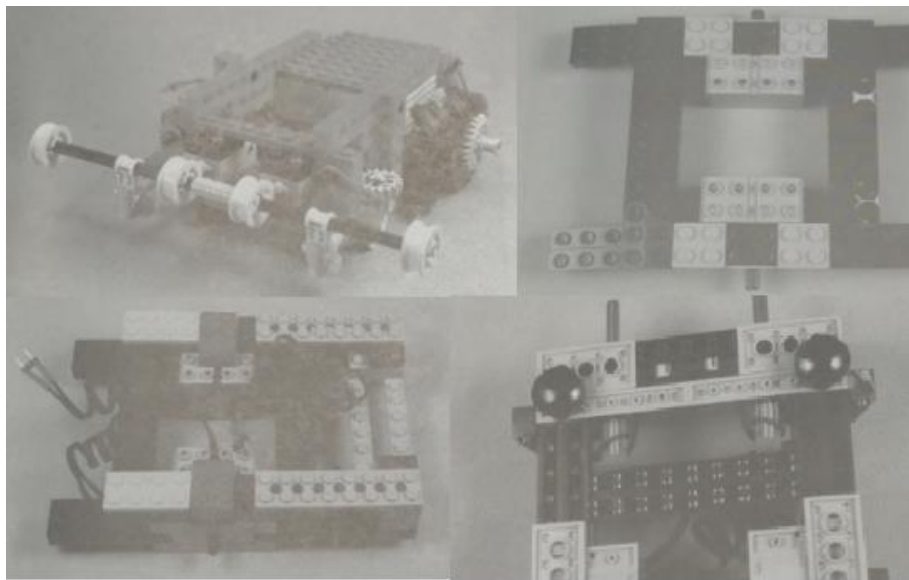


Figure 8 Stages of Construction: The different stage made throughout the lab sessions taken from Robotic Explorations.

Programming the HandyBug with PuckBot.ic was somewhat straightforward. After testing certain values, the code was refined given a particular motor strength and retested for precision. Thusly, the motor and sleep contained the vast majority of the solution. Figure 4 shows a case of the code that was organized. In this specific case, the HandyBug is to push ahead from 40 power for 3.0 seconds. Regarding the Competition, the HandyBug moves sufficiently only to push the puck in position retaining to the first puck. Consistency was able to be keep up along the lines in light of the fact that the HandyBug would move to a similar position if the batteries were genuinely charged and it begun from the home box.

Analysis and Testing

Testing methods for Competition 2 was tedious, anyway exact in nature. Subsequent to composing the code for the HandyBoard, stages were made to separate every technique and task. After that, the code was written partially along with testing. Stage 1, for instance, includes recovering the first puck and moving it into the goal. In the wake of testing the motor/sleep functions, which may drive the HandyBug excessively far or not exactly enough, the procedure of refinement was started to make the beginning and halting points as exact as could be allowed. Table 3 shows how accurate the code is obtained through consecutive testing.

Trials	Motor Speeds (Both)	Sleep Functions	Outcome
1	40	3.0	Too far
2	40	2.0	Too close
3	50	2.3	Almost
4	65	2.1	Exact

Table 3 Trials and Precision example: A case of how precise is acquired through testing methods for Competition 2. In spite of the measure of preliminaries, precision is expanded the more preliminaries are finished. This successfully makes trial exhibit the same similar but not consistent, overall giving almost consistent results.

During the testing procedure numerous issues happened, particularly with structure of the HandyBug. There were quite a few mistakes for structural design. For instance, when running, the wheel or few LEGO pieces would precipitously tumble off. This was combined with various hardware issues, for example, flawed battery packs that, when moved a specific way, would totally restart the Handy Board's system, and motor cables that would not appropriately connect into their ports. For example, the decrease in the batteries' capacity would endanger the Handy Board's code integrity. After about an hour or so of testing, batteries would in the end lose their full condition of-charge – ultimately decreasing motor's execution. This was exceptionally hard to see at first, until a difference in batteries caused the group to acknowledge execution was normal once more. Even more so, the battery's discharge made an adverse impact during testing, as the HandyBug didn't move the extent that it should have. This causes the accuracy that was endeavored to acquire lose its reliability.

The aftereffects of Competition 2 hushed up promising, be that as it may. Our group only one challenge by pushing more pucks over the field without intersection. At last, inconsistency in the HandyBug's activities were what ascribed to the almost success.

Later on, improvements can be made to make the HandyBug altogether autonomous and taking care of issues that emerge. Nonetheless, this would possibly be essential if the states of the game are not steady or fixed.



Final Evaluation

Objective

To build up an autonomous robot that will contend in different competitions, concentrating on the utilization of light and touch sensors.

Key Design

The key design and plan features for Competition 2 included hard coding the IC code and actualizing it into the HandyBug without the utilization of sensors. This arrangement demonstrated to be the best and credited to the achievement of Competition 2.

Future Work

The plan and design presented is viable, effective, stable, and predictable, be that as it may if the course were to change the HandyBug would not perform so well. Given this, there is opportunity to get better. A genuine powerful robot will have the option to be fruitful in the Competition without the guide of hard coding. Future enhancements would incorporate programming the HandyBug to have the option to totally navigate the course and move pucks to their suitable situations with the utilization of sensors.

Conclusion


To finish up, with the additional finances in the research, testing, and laboratory time will help initiate the solidification of the structure of the HandyBug. The organization's prosperity will be guaranteed with design and planning that will be exhibited in the competitions to come.



Appendix

Code PuckBot.ic

```
int LEFT_MOTOR= 3;
int RIGHT_MOTOR= 0;
int LEFT_SENSOR = 15;
int RIGHT_SENSOR = 12;
int bumpedBool = 0;
float _timer;
void main ()
{ int i =0;
  while (1)
  { if (start_button ())
  { reset_timer ();
    while (timer () <15.0)
  { printf ("timer=%f \n", timer ());
    motor (LEFT_MOTOR, 50); //forward 1st
    motor (RIGHT_MOTOR, 50);
    sleep (2.2);
    motor (LEFT_MOTOR, -35); //back
    motor (RIGHT_MOTOR, -50);
    sleep (2.9);
    motor (LEFT_MOTOR, -40); //turn
    motor (RIGHT_MOTOR, 40);
    sleep (.95);
    motor (LEFT_MOTOR, 85); //forward 2nd
    motor (RIGHT_MOTOR, 100);
```



```
sleep (1.46);
motor (LEFT_MOTOR, -70); //backwards
motor (RIGHT_MOTOR, -100);
sleep (1.44);
motor (LEFT_MOTOR, 40); //turn
motor (RIGHT_MOTOR, -40);
sleep (.05);
motor (LEFT_MOTOR, 100); //forwards 3rd
motor (RIGHT_MOTOR, 100);
sleep (1.8);
motor (LEFT_MOTOR, 5); //curved turn
motor (RIGHT_MOTOR, 100);
sleep (.65);}
motor (LEFT_MOTOR,0);
motor (RIGHT_MOTOR,0);
break;}}
void reset_timer ()
{ _timer = seconds ();}
float timer ()
{return (seconds () - _timer);}
```



References

1. Martin, Fred G. 2001. Robotic Explorations: A Hands-On Introduction to Engineering. New Jersey: Prentice Hall.
2. Oruklu, Erdal. 2017. ECE 100 Lecture Notes. Chicago: Illinois Institute of Technology, Electrical and Computer Engineering Department.