The Trials and Tribulations of the Group 16 Robot

A report on our preparation and results

Angus Burgess - Kieran Hislop - Daniel Bellefontaine - Prannat Jain - Alex MacCulloch

B00891717 - B00888752 - B00819770 - B00907785 - B00888773

CSCI 1108 - Experimental Robotics - Group 16 - Section 2

Faculty of Computer Science

an333955@dal.ca - kr912789@dal.ca - dbellefontaine@dal.ca - prannatj@dal.ca - cl496753@dal.ca

Abstract

Our final project found us coding a Thymio robot to let it make its way through an obstacle course without hitting any blocks, all while following a specific line. As well, we had to make sure the robot went a specific way around the obstacles without hitting them and then finding the line again. During the time we were given, we spent our lab days as well as extra time outside of class working on our code so that we could have the robot achieve what was needed of it. The results: our robot aced the 3rd course netting us a 100 score.

Introduction

Robotics is the science of perceiving and manipulating the physical world through computer-controlled devices"- Sebastian Thrun, Probabilistic Robotics

The rapid adoption of robotics in our everyday lives continues. Industries small and large are investing and betting on the potential that robotics has to offer. Automation and robotics promise to speed up complex tasks and help in the cost-effectiveness of industrial operations. The components shared among the variety of contemporary robots are, **sensors**, **controllers**, **actuators**. To familiarize us with all the components of a standard robot, our final project required us to compete in a competition that utilized all the components mentioned above.

Throughout the semester, we've been working on different labs that made us capable of coding with ASEBA language and creating some complex programs, all of which led up to our final project for the class. When it comes to said final project, we were given a timetable of when the different milestones were expected to be done by, so we would have an idea of what needed to be done, and when. The milestones helped in maintaining pace and kept us consistently working on the code. The final project: simply have our robot make its way through a course littered with obstacles without hitting any of them, all while detecting the white colored barcode in front of the robot, and then turning to go that way around the block, and ultimately finding the line again and continuing along the course. We were also given a time limit for the robot to complete the course being 90 seconds.

Knowing all that, our group got going on the project right away. The first day was fairly straightforward. We created a project charter and assigned duties, in terms of project submissions, note taking and we discussed our plan of action for programming the robot. Since we already had some code available from past labs that could help us, we chose to start with those sections aside from some minor adjustments to tighten it up. After that we would just have to write new code for the sections that we hadn't fully covered prior to this project, and once we had that all done, we would just have to combine all our code and make sure it all worked well together. If it didn't work, we had to troubleshoot it and make revisions. We set our sights high for our end goal so we could achieve the best grade possible. We wanted our robot to complete the course in under 90 seconds without hitting any of the hurdles. On our second day we didn't

encounter many issues as the code that was required for this milestone was already mostly finished from a prior lab. All we had to do was make some slight adjustments to allow the robot to follow the line much easier, which mainly consisted of us adjusting the speed target constant. We found that the robot's ability to follow the line suffered on sharp turns depending on how it approached the turn. Based on the speed, the robot is positioned differently depending on which side of the motor it's primarily powering. Setting the speed constant to 410 or 500 worked, but most other targets for our speed constant would not work. After we found the right speed and made sure it worked 100% of the time we submitted the code for our milestone and thus ended the first week of our project. For day 3 our objective was object recognition, making sure our robot could detect blocks in front of it and stop without hitting any of them. We did run into a few issues with the coding, however we managed to get the robot to stop where we wanted it to by the end of our class time. Once we started on barcode recognition, we split into two groups to tackle the task. 3 of us worked on getting the robot to read the barcode at angles as it wasn't always head on while the other two attempted to have the robot orient straight to make the barcode scan easier. However, we still had trouble with it, so we had to get together on a weekend to work on the project. During that we were able to get the line follow program, the controlled stop, as well as the sensor reading all working together. We also came up with our plan for getting our robot around the block without hitting it. Depending on the side it was supposed to go around we would have the robot pivot in that direction until one of the sensors closer to the back was activated. Upon doing so, the robot would then begin to make an arcing turn around the block that would utilize a slower inside wheel and a quicker outside wheel to round the block in the desired direction and then stop once the line is detected again. On day 6 we had gotten our 2nd milestone report back and learned that our code had more errors than we knew about. We spent most of that lab troubleshooting solutions so that we could fix those issues and have the robot read the barcode properly. We also modified the code from a suggestion from Angus to try adding some TRUE and FALSE statements which helped. Day 7 was more troubleshooting and improving of the code. By day 8 we were in trouble, in the state our code was, we knew it wouldn't be passing any trials no matter what we did to fix it, so we started anew. Alex during class and in their spare time, redid a bunch of the code while making it neater and easier to read in the process. Once they had a decent code set, they sent it to our google drive to allow us to access and work on it alongside them. During that lab we managed to get the new code reliably scanning and maneuvering around the 1st, 4th, and 5th obstacles, but we still had much work to do. We spent countless time outside of the lab time working on the code, trying to get it working and eventually Prannat managed to get the code working the way we wanted it to. He diagnosed this issue to be mostly arising from the fact that when our robot would complete a turn and return to the line, our line follow program would cause the robot to turn in the opposite direction of the tape which caused it to veer off the line every time it tried to rejoin the line after completing a turn. Specifically, we added a variable: "*prevturn*" which was updated with the information about the direction of the previous turn each time the robot read the barcode. This variable was used to call different line follow programs each time the robot completed a turn and tried to rejoin the line. After a few adjustments we had it working most of the time. Eventually, we were as ready as we could be for the competition. During the competition we were tested on 3 trial courses. While our robot didn't 100% make it through the 1st and 2nd courses the way it was supposed to, it did make it through the 3rd course without

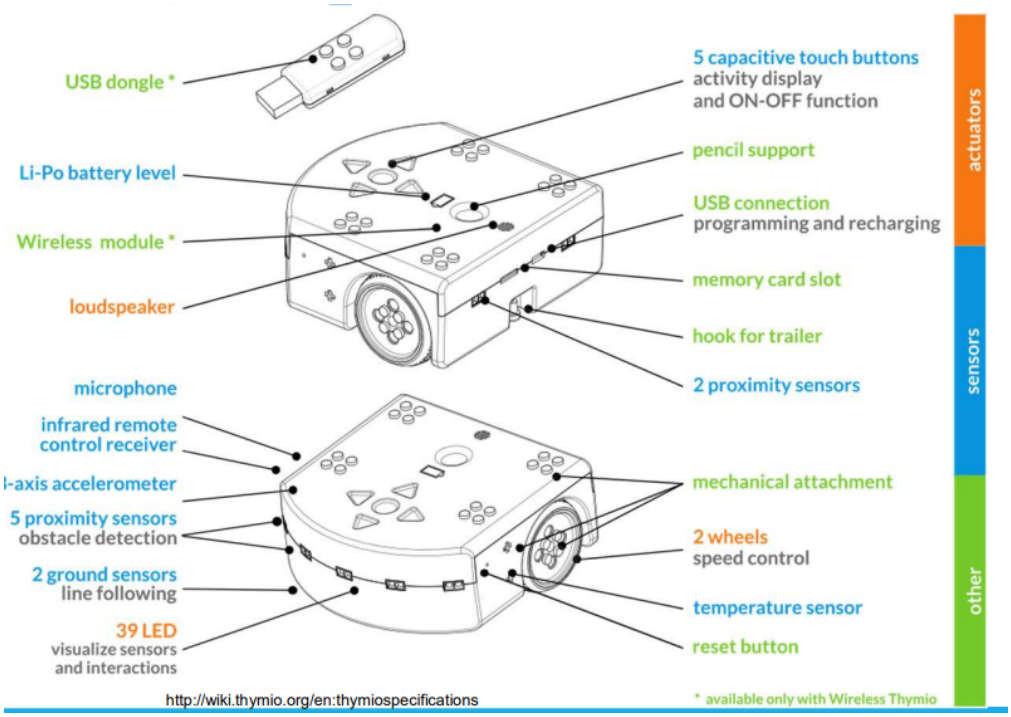
hitting any of the blocks and under the time limit that was set which netted us a 100 for the competition portion of the final project.

Background

In this project, we used two different programs: Hammer (our simulator for the robot) and ASEBA Studio (our compiler for the code). The course also required us to use the Thymio II robot which is programmed using the ASEBA language. The Aseba language syntax resembles popular programming languages, such as Pascal and Matlab. Semantically, it is a simple imperative programming language with a single basic data type (16 bit signed integers) and arrays (**http://wiki.thymio.org/en:asebalanguage**). Its simplicity makes it suitable for programming robots.

Some reserved keywords of ASEBA language: *abs, call, callsub, do, else, elseif, emit, end, for, if, in, onevent, return, step, sub, then, var, when, while.* These constructs enable the creation of loops, if-else statements, arrays, events, etc. which allow for making complex programs.

The anatomy of the Thymio robot used in this final project:



With the help of work accomplished in lab 9 (object recognition), we had begun the base of what would later become our barcode reader for the final project after lots of modifications. A key difference here being that in the lab, our robot was in a static stopped position. One of the problems we spent the most time troubleshooting later in the project is partially related to this because the robot would often not be aligned properly in front of the block to obtain an accurate barcode reading. Other labs leading up to this gave us a foundation of tools for our completion

of the project, but they did not seem to make parts of the code that were as direct or substantial for our project.

A concept used in this final project is one of Markov Localization.
Markov Localization is an estimation that a robot makes to attempt to determine where it is in comparison to other objects surrounding it. We used this estimation to help our robot traverse the course in several different ways, and through several different methods such as acute observation, comparison, as well as process of elimination.

Program description

To start with, ASEBA language does not appreciate static numbers, it much prefers a range when dealing with if statements or they may not work. Even if the code states a range greater than “X” it still has a chance of not occurring. Or at least, that is what our testing showed us. We had already played with a “Line follow” subroutine within a lab previously so we all had a basic understanding of what kind of scope was required. All we needed to do now was have the robot traverse on the line and that part was done, the blocks were our next problem. In lab 9, we saw that we could read a block head on with no issue, but any angle would pose the challenge of comparing sensors. We checked out a range of inputs to see in what values a green block could be read and a white block could be read and put our range of values based off of that data. So we would compare horizontal sensors 1 and 3 of the Thymio robot, while making sure that each sensor was in a set range. This method of solving said problem was chosen because we knew at any point our sensors would give a different reading by the slightest change in their angle of attack. This balancing act of what angles could the robot encounter and to what extent of data could it read. We figured comparing two or three sensors rather than just one, would give us a better look on what kind of block we were reading. Since any angle other than a declared “tested” would provide a vastly different result. For an explanation: It's better to check sensor 1 to sensor 3 and see if the difference matches with that of other tested angles to see if the difference in measurements means we are at angle *A* or angle *B.* This idea caused us to look at our readings as a broader array of numbers than if we were dealing with a set program. Lets say a game run in java. As it should be stated a plethora of times, ASEBA and its robot can and will give you different outputs for each run of your program. Due to a realistic set of variables.

Note: Our state transition diagram can be found under the *Appendices* number *2*

Expanding on our state transition diagram, it first needs to be stated that the “TURN RIGHT” and “TURN LEFT " states are used in our path to move around the detected block and return to following the line. Now, when the program first starts it moves forward. If, the ground sensors detect a line then it will execute our line follow(“FORWARD” state). Now, if there is a block ahead of itself, firstly it will go into the “BLOCKED” state if horizontal sensors 2 or 0 reach certain threshold values and then secondly it will read what side the white barcode is on, indicating the direction the robot should turn around said block, and then enter the state of “TURN LEFT" or “TURN RIGHT” for a set amount of time before checking for a reading from its ground sensors meaning that there is a line to follow. Then repeat.

To explain our thinking on why these solutions came to such problems. We looked at this problem as how do we use what the robot sees to our advantage. We ended up just taking a short period of time out of our day to see what kind of data or robot would get depending on the angle and/or distance the robot was from the target block, and it was these tests that gave us the idea for test cases. If these amounts of data come up we must be at about “this” angle so we are under a set restriction that was previously set to work under “this” angle. We took the problem of if the white block is on the right, turn right, into the solution: If the robot is to the right of the block, turn for X amount of seconds depending on the turn needed to make. It was a constant, how do we make the data work for us? Are there patterns or commonalities in said data? Our entire solution to the task at hand can be simplified to an “if then” statement. Just manipulated for a vast array of data.

Using the concept of localization we developed our line follow program as follows. To begin, using the two ground sensors, we could determine whether or not the black line was seen or not by comparing its reflectivity to the normal backdrop. By doing so we could determine two things. First, whether or not the robot s was or was not on the line at any given moment of time. Second, we could determine where the line was in comparison to the robot by seeing which sensor detected the line. Now, since we knew that the robot would begin centered on the line to start with, we could use this knowledge, as well as process of elimination to allow the robot to stay on the line.

Firstly, if no line was detected on either sensor, the robot was centered on the line, and thus could drive straight without worry of going off the line. If the robot then detected as it was driving a sensor detected the line, it would either pivot left or right to allow the robot to stay on course. By checking these conditions multiple times a second, the robot could make slight, precise movements and thus allow it to stay on the line throughout the course.

Our second most used use case for localization was to determine whether or not a block was detected. This process was a fair bit more complicated than the line follow as not only were there more sensors to compare to, as well as more ways the robot would need to check and avoid the block, distance and angle of approach were added into the mix as well. To allow our robot to pass this section, we needed to break down the events that could happen for our robot.

To start, distance was one of the most important items to nail down as it would allow us to determine the robot’s clearance to get around the block, as well as if it was close enough to properly read the barcodes of the block. To solve this, we first took readings that the sensors would tell us as it was different distances from the block. We did this for both black and white colors, then once an average was found for each distance, we determined the best distance for the robot to be at in order to read the blocks and then promptly get around them the best. Once found, we ensured this would work for all sensors, and from there we were ready for the next task.

Next was angle, which was by far the most time consuming and test heavy part of the project as it was required for the robot to navigate around the block, required for ensuring it read the barcode correctly, and required for the robot to get back on track once around the block. To solve this, we did several tests to figure out what angles the robot was approaching blocks at, and from there gathered reading from the sensors to gather trends. From these trends, we could allow our robot to use Markov Localization to determine roughly what angle it approached the block, and from there what actions to take thereafter.

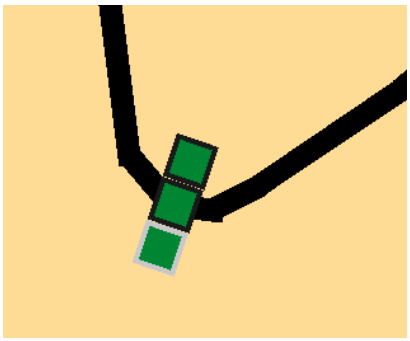
As stated and discussed, this is how we used Markov Localization within our program to allow our robot to succeed in its mission. No doubt without such an ability, the robot would not have been able to complete such a course, and for that, we are glad we were able to use this ability to our utmost advantage, in completing the course as desired.

Results

The final competition for our group was held on 29th March 2022, at the Goldberg in room 134. We were given 10-15 minutes to test our code along with our robot's sensors and motors to make sure everything was working as intended. This test period proved to be extremely useful for us, because we discovered that the barcodes on the duplo blocks had been replaced by newer and cleaner versions which affected our threshold values. We had to decrease our threshold values by a value of 100, which corrected the issue.

On **trial 1** we achieved a score of 90, 10 marks being deducted because our robot turned in the wrong direction around a block because of incorrectly reading the barcode. This can be attributed to the inability of the robot to successfully orient itself in front of the barcodes to read the barcodes.

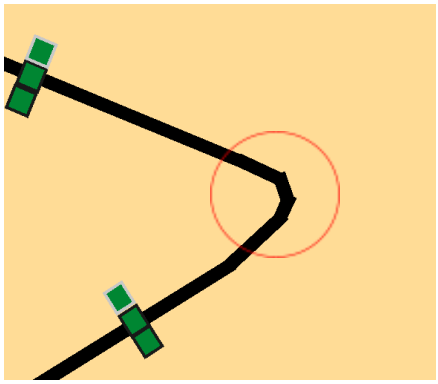
The block which resulted in a 10 point reduction in our score:



It successfully navigated through the rest of the blocks in trial 1 though, and turned in the correct direction for each barcode; and did so in exactly 90 seconds.

Trial 2 was the most challenging because its sharp turns gave the robot some troubles, primarily due to a higher than ideal speed of our line follow program.

The turn that caused our robot to go off course:



We eventually decided to abort our run in trial 2 and proceed on to trial 3.

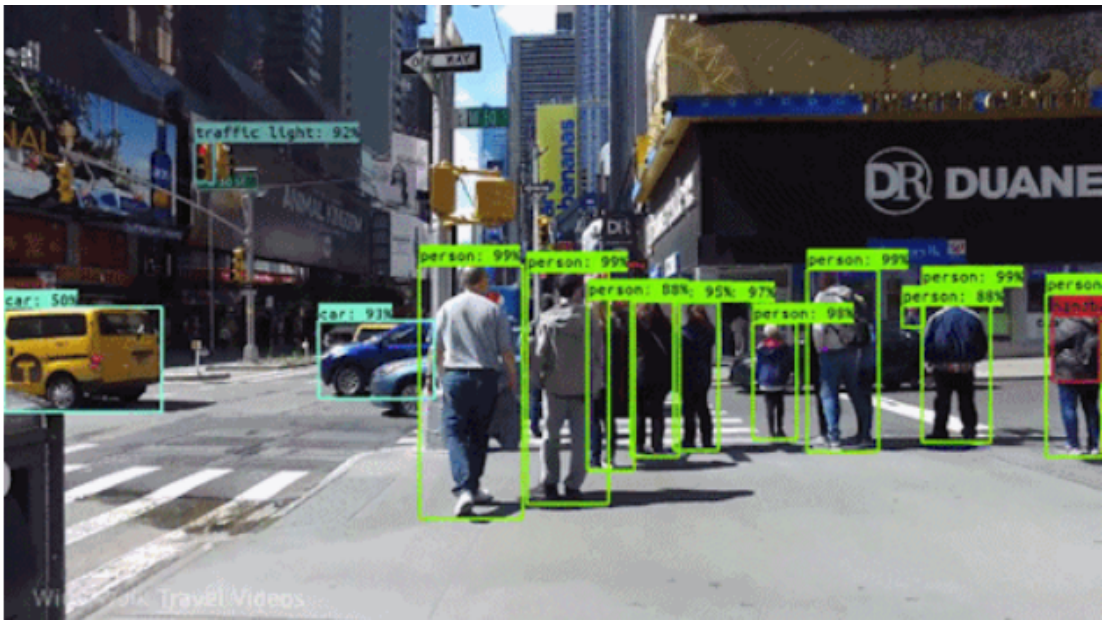
Our robot fared the best and scored a 100 on **trial 3** during the final competition. It completed trial 3 flawlessly and relatively quickly, mainly because the positioning of the blocks was quite favorable which enabled accurate reading of the barcodes. The course was completed in under 90 seconds with plenty of seconds to spare.

Conclusion and Potential Improvements

Robotics is one of the more interesting aspects of computer science and this final project provided us with invaluable experience and knowledge. Competing in this project required teamwork, co-operation, and most importantly hard work.

Our final competition code worked well and helped us achieve an A+. If we had more time to work on our project we would firstly improve upon our line follow program. The thymio robot struggled to navigate sharp turns at high motor speeds and frequently lost the line upon hitting these turns. The second thing we would like to work on is our failure recovery program. We had originally created a recovery program, but it didn't work out as well as we would have liked it to and it would have needed a lot more work if we were to achieve the result we wanted from it, so we chose to not include it in our final code. We instead found an easier, working alternative. Thirdly, during the test turns at the Goldberg we observed our robot's turning radius was quite high. After taking a look at the robots of other contestants and observing their robots while making a turn, the difference in the turning radius was quite noticeable. This did not affect the performance of our robot in any of the trials mentioned above but could potentially be an issue on a different course.

The concepts used in this final project have wide-ranging real world applications and are already being implemented by technological giants such as Tesla, Google, Waymo, etc. One such application is real world object recognition using computer vision:



Thanks to advances in artificial intelligence and innovations in deep learning and neural networks, this field has been able to take great leaps in recent years and has been able to surpass humans in some tasks related to detecting and labeling objects.(Ilija Mihajlovic April,2019)

References

1. *Thymio & Aseba.* The Aseba language - Thymio & Aseba. (n.d.). Retrieved April 6, 2022, from **http://wiki.thymio.org/en:asebalanguage**

2. Brodsky, A. (2017). *User's Guide to Thor's Hammer A Thymio II Simulator and Visualizer*.

3. Everything You Ever Wanted To Know About Computer Vision. (April,2019). Retrieved April 6, 2022, from **https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e**

Appendices

1. Code used in the final competition:

```
#* File: Hurdle
Date: March, 2022
Author: CSCI 1108: Experimental Robotics.
Purpose: FINAL PROJECT
Description: A program that avoids hurdles and follows a line.
*#

<!--list of constants-->
<constant value="0" name="FORWARD"/>
<constant value="1" name="STOPPED"/>
<constant value="470" name="TARGET"/>
<constant value="2" name="BLOCKED"/>
<constant value="350" name="EDGE"/>
<constant value="2600" name="THRESEHOLD"/>
<constant value="3" name="TURNL"/>
<constant value="4" name="TURNR"/>
<constant value="690" name="ROTATE"/>
<constant value="5" name="RETURN"/>
<constant value="1700" name="FRONTCHECK"/>
<constant value="1075" name="SIDECHECK"/>
<constant value="1" name="WHITE"/>
<constant value="0" name="BLACK"/>
<constant value="30" name="LEFT"/>
<constant value="40" name="RIGHT"/>


var reading1 = BLACK #LED color initialization
var reading2 = BLACK
var reading3 = BLACK
var prevturn = 0
var timer0 = 0
var timer1 = 0
var followsafe = 0

var check = 0
var state = STOPPED # State declarations

motor.left.target = 0 # reset motors
motor.right.target = 0

timer.period[0] = 0
timer.period[1] = 0 #timer initialisation
```

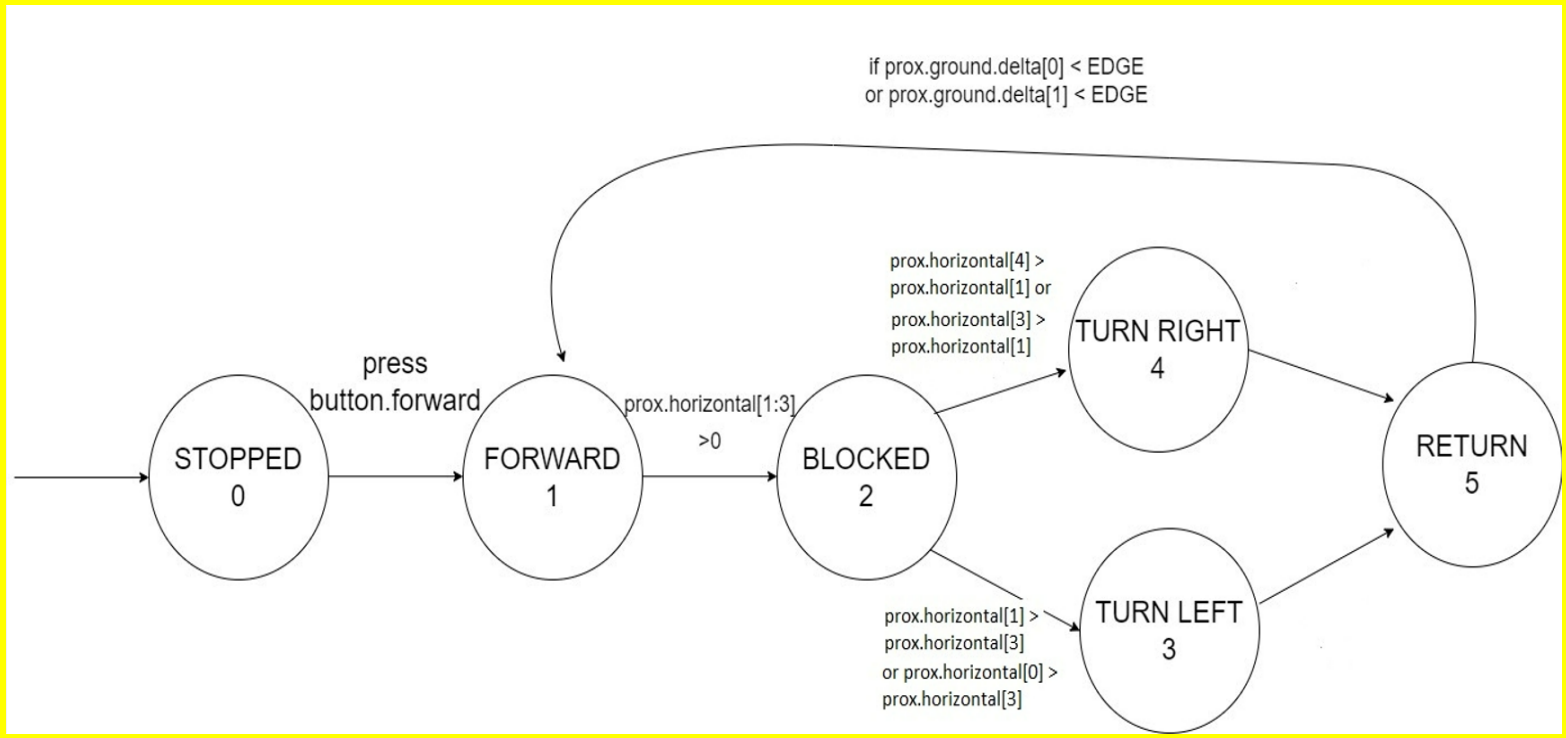
```
call leds.circle(0,0,0,0,0,0,0,0) #sets lights back to "off"

onevent button.backward # on backward button press

	state = STOPPED # transition to STOPPED state
	motor.left.target = 0 # stop motors
	motor.right.target = 0
	timer.period[0] = 0
	call leds.circle(0,0,0,0,0,0,0,0)
onevent button.forward # on forward button press

	state = FORWARD
	call leds.circle(0,0,0,0,0,0,0,0)
	prevturn = LEFT
onevent prox

	if state != BLOCKED and state != TURNL and state != TURNR and state != STOPPED and
state != RETURN then

		callsub lineFollow #line follow pragram is called
	end
	if state == BLOCKED and state != TURNL and state != TURNR and state != STOPPED then

		callsub wall #wall is detected
	end

	if state == RETURN and (motor.left.target == TARGET or motor.right.target == TARGET)
and state != STOPPED then # these line get robot back onto line follow after turn
		timer.period[1] = 0
		if prevturn == LEFT then #checking for the line after a left turn
			if prox.ground.delta[0] < EDGE then
				callsub lineFollow
				check = 1
			end
			if prox.ground.delta[1] < EDGE then
				callsub lineFollow
				check = 2
			end
		end
		if prevturn == RIGHT then #checking for the line after a right turn
			if prox.ground.delta[1] < EDGE then
				callsub lineFollow
				check = 3
			end
			if prox.ground.delta[0] < EDGE then
				callsub lineFollow
				check = 4
			end
		end
	end
sub lineFollow #line follow subroutine
	timer.period[0] = 0
```

```
        timer.period[1] = 0
        state = FORWARD

        if prevturn == LEFT then #returning back to the line after a left turn

                if (prox.ground.delta[0] >= EDGE) then
                        motor.left.target = 300 # turn left
                        motor.right.target = 300/6
                end
                if (prox.ground.delta[1] >= EDGE) then
                        motor.left.target = 300/6 # turn right
                        motor.right.target = 300
                end
        end
        if prevturn == RIGHT then #returning back to the line after a right turn
                if (prox.ground.delta[1] >= EDGE) then
                         motor.left.target = 200/6 # turn right
                         motor.right.target = 200
                         timer.period[1] = 0
                end
                if (prox.ground.delta[0] >= EDGE) then
                         motor.left.target = 200 # turn left
                         motor.right.target = 200/6
                         timer.period[1] = 1500
                end
        end
        if (prox.ground.delta[0] < EDGE) and (prox.ground.delta[1] < EDGE) and state !=
RETURN then
                 motor.left.target = 300 # go forward
                 motor.right.target = 300
        end
        if prox.horizontal[2] > 0 or prox.horizontal[1] > 0 or prox.horizontal[3] > 0 or
prox.horizontal[0] > 0 then
                state = BLOCKED
                reading1 = BLACK #reset LEDs after completing a turn
                reading2 = BLACK
                reading3 = BLACK
        end
sub wall

        if prox.horizontal[2] > 0 and prox.horizontal[2] < THRESEHOLD then
                 timer.period[1] = 0
                 motor.left.target = 300 # go forward
                 motor.right.target = 300
        end

        if prox.horizontal[2] > FRONTCHECK or prox.horizontal[0] > SIDECHECK or
prox.horizontal[4] > SIDECHECK and state != TURNL and state != TURNR then
                 motor.left.target = 0 # stop
                 motor.right.target = 0
        if prox.horizontal[1] > prox.horizontal[3] or prox.horizontal[0] > prox.horizontal[3]
or prox.horizontal[2] > 1900 then
```

```
                state = TURNL
                motor.left.target = -TARGET # rotate left to position the robot for a turn
                motor.right.target = TARGET
                prevturn = LEFT
                reading1 = WHITE
        elseif prox.horizontal[3] > prox.horizontal[1] or prox.horizontal[4] >
prox.horizontal[1] or prox.horizontal[2] > 2000 then
                state = TURNR
                motor.left.target = TARGET # rotate right to position the robot for a turn
                motor.right.target = -TARGET
                prevturn = RIGHT
                reading3 = WHITE
        end
        if prox.horizontal[0] > 500 then #take a smaller turn if block is not very near
                timer.period[0] = 900

        elseif prox.horizontal[4] > 500 then
        timer.period[0] = 950 #take a larger turn if the block is near
        else
                timer.period[0] = ROTATE
        end
end

callsub identify #switch on LEDs

onevent timer0
        if state == TURNL then
                motor.left.target = TARGET # turn left
                motor.right.target = (TARGET*5)/11
                state = RETURN
        end
        if state == TURNR then
                motor.left.target = (TARGET*7)/15 # turn right
                motor.right.target = TARGET
                state = RETURN
        end
sub identify
        if reading1 ==1 and reading2 == 1 and reading3 == 1 then
                call leds.circle(32,32,0,0,0,0,0,32)
        elseif reading1 ==0 and reading2 == 1 and reading3 == 1 then
                call leds.circle(32,32,0,0,0,0,0,2)
        elseif reading1 ==1 and reading2 == 0 and reading3 == 1 then
                call leds.circle(2,32,0,0,0,0,0,32)
        elseif reading1 ==1 and reading2 == 1 and reading3 == 0 then
                call leds.circle(32,2,0,0,0,0,0,32)
        elseif reading1 ==0 and reading2 == 0 and reading3 == 1 then
                call leds.circle(2,32,0,0,0,0,0,2)
        elseif reading1 ==1 and reading2 == 0 and reading3 == 0 then
                call leds.circle(2,2,0,0,0,0,0,32)
        elseif reading1 ==0 and reading2 == 1 and reading3 == 0 then
                call leds.circle(2,32,0,0,0,0,0,2)
        elseif reading1 ==0 and reading2 == 0 and reading3 == 0 then
```

```
            call leds.circle(2,2,0,0,0,0,0,2)
        end
```

2. State transition diagram of the program:



3. Overall Competition results: Arrow indicates our result.

Trial 1				Trial 2				Trial 3				Grade
% Course	Barcodes	Touched	Time	% Course	Barcodes	Touched	Time	% Course	Barcodes	Touched	Time	
100	5*	2	NA	100	3	1	NA	100	5	0	64	A+
100	5	0	72	20	1	0	NA	80	4	1	NA	A+
100	4	1	NA	80	3	2	NA	100	5	0	86	A+
100	5*	1	60	20	1	0	NA	100	5	0	90	A+
100	5*	1	90	20	1	0	NA	100	5	0	90	A+
100	5	0	95	100	5	0	90	80	4	0	NA	A+
80	5	0	NA	80	3	1	NA	100	5*	1	90	A
100	5	4	NA	100	1	2	NA	100	5*	1	NA	A
100	5*	1	NA	20	1	1	NA	20	1	1	NA	A
20	1	0	NA	40	4	0	NA	100	5	0	105	A
80	3*	0	NA	20	1	0	NA	100	5	1	NA	A
80	5	0	NA	80	4	1	NA	100	4	1	NA	A
100	5	0	103	20	1	0	NA	60	2	0	NA	A
100	5*	1	NA	20	1	0	NA	80	4	1	NA	A
100	5	2	NA	100	2	3	NA	80	4	1	NA	A-
60	3	1	NA	60	3	1	NA	100	3	2	NA	B+
80	2	4	NA	100	3	5	NA	100	5	5	NA	C+
100	4*	2	NA	20	20	1	NA	100	2	2	NA	C+
100	3	2	NA	20	1	0	NA	80	2	2	NA	C+
80	4*	1	NA	20	0	0	NA	40	2	0	NA	C+
100	2	4	NA	20	1	0	NA	80	4	2	NA	C+
100	3*	2	NA	20	1	0	NA	80	3	0	NA	C
0	-	-	-	0	-	-	-	0	-	-	-	70 (simulator)
0	-	-	-	0	-	-	-	0	-	-	-	70 (simulator)