CSCI 1108 – Experimental Robotics Project Report Group 5

ABSTRACT:

This project is rudimentarily a robot race which has the primary aim of line following along with obstacle avoidance on a fixed path made out of black tape which had five hurdles made of green LEGO blocks. Thymio Robot was used for this project and it was coded in the ASEBA programming language, written in the ASEBA studio. To epitomize the results, the robot had two perfect runs; 45 seconds each. While it lost the line in the third run which was a result of experimenting to incorporate a faster line following mechanism to decrease the time that the robot takes to complete the track.

INTRODUCTION:

Robots need to make sense of their surroundings in order to act in any environment. For this project, the environment is predefined as a track having definite number of hurdles and set turns which enables us to make the program specific and more efficient for the given track. The lighting and the surface are also pre-set to normalized white room lighting and a smooth textured yellowish-brown table. This project has been designed as an Olympic racecourse with the above set of constraints between several teams participating from this section of the course as well as the other section. It was presented on the 18th of October 2019.Three attempts were given and each of the attempt was recorded and graded by Head Teaching Assistant – Mitchell Kane.

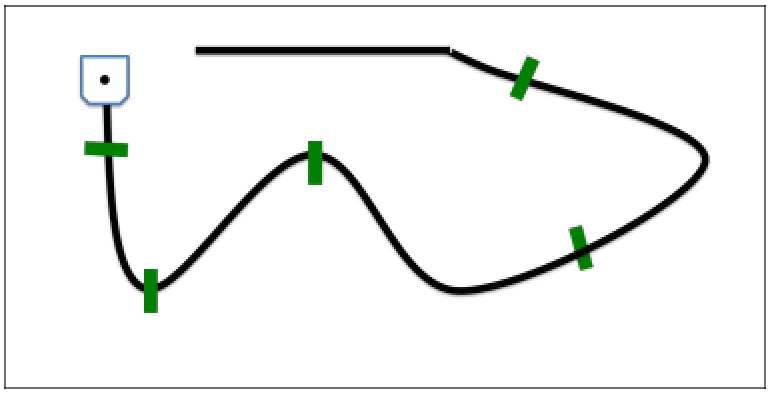
The Robot that was used is called Thymio. As line following and object avoidance are the tasks, two ground sensors (0 and 1) were used for black tape(line) following, and five out of the seven horizontal proximity sensors were used for object detection. The robot has two wheels each powered by a motor which can be calibrated by using the physical robot or by coding itself. ASEBA programming language is being used in this project and the platform used is called ASEBA studio.

The main problems encountered and the solutions we determined during the project were:

- The blocks placed close and made the turns sharper.

    - Solution: Make use of different-sided turns to eliminate touching blocks.
- The increase in speed in order to counter the competition but missed the line due to lack.
    - Solution: Make use of both the ground sensors instead of one for line following making it more effective.
- After turning, robot touches line at 90 degrees which makes it lose the line sometimes at greater speeds.
    - Solution: Adjust the turning speed such that the angle decreases.

BACKGROUND:



Retrieved from
https://dal.brightspace.com/d2l/le/content/100094/viewContent/1466617/View

On 24th October 2019.

The above track was supposed to be followed by the robot. We have used the ASEBA programming language which is specifically designed THYMIO robot. It is a very simple language only having 16–bit integer as its only data type. It follows a basic structure:

- Variable Declarations.
- Initialization Code
- Event Handlers.
- Sub-routines.

While the constants are declared in its area in the ASEBA studio.

The THYMIO robot has two ground proximity sensors, seven(five in front and two on back) horizontal proximity sensors, led lights on the top along with physical touch buttons for forward, backward, left and right, and two motorized wheels.

This project was only possible by aggregating the knowledge acquired from the earlier labs present on Brightspace and this statement is especially pertinent to Lab 8 and Lab 9.

Lab 8: Modelling Real World(STD) was described in four sections:

- Simple Stop and Go Program.
- Simple Line Follower.
- Combining the above two: A Stop and Go line follower.
- Better Line Follower.

Lab 9: Dealing with Failures described how to define failure modes, identify them, deal with them via recovery mechanism, and acknowledge those defeats when implied by hardware shortcomings.

Combining the above two labs helped achieve the goal of completing the track while avoiding the obstacles and following the line efficiently.

References:

- Programming learnt: http://wiki.thymio.org/en:asebalanguage
- Line following: Lab - 8
  https://dal.brightspace.com/d2l/le/content/100094/viewContent/1457647/View
- Solving failures: Lab – 9
  https://dal.brightspace.com/d2l/le/content/100094/viewContent/1465312/View

Program Description:

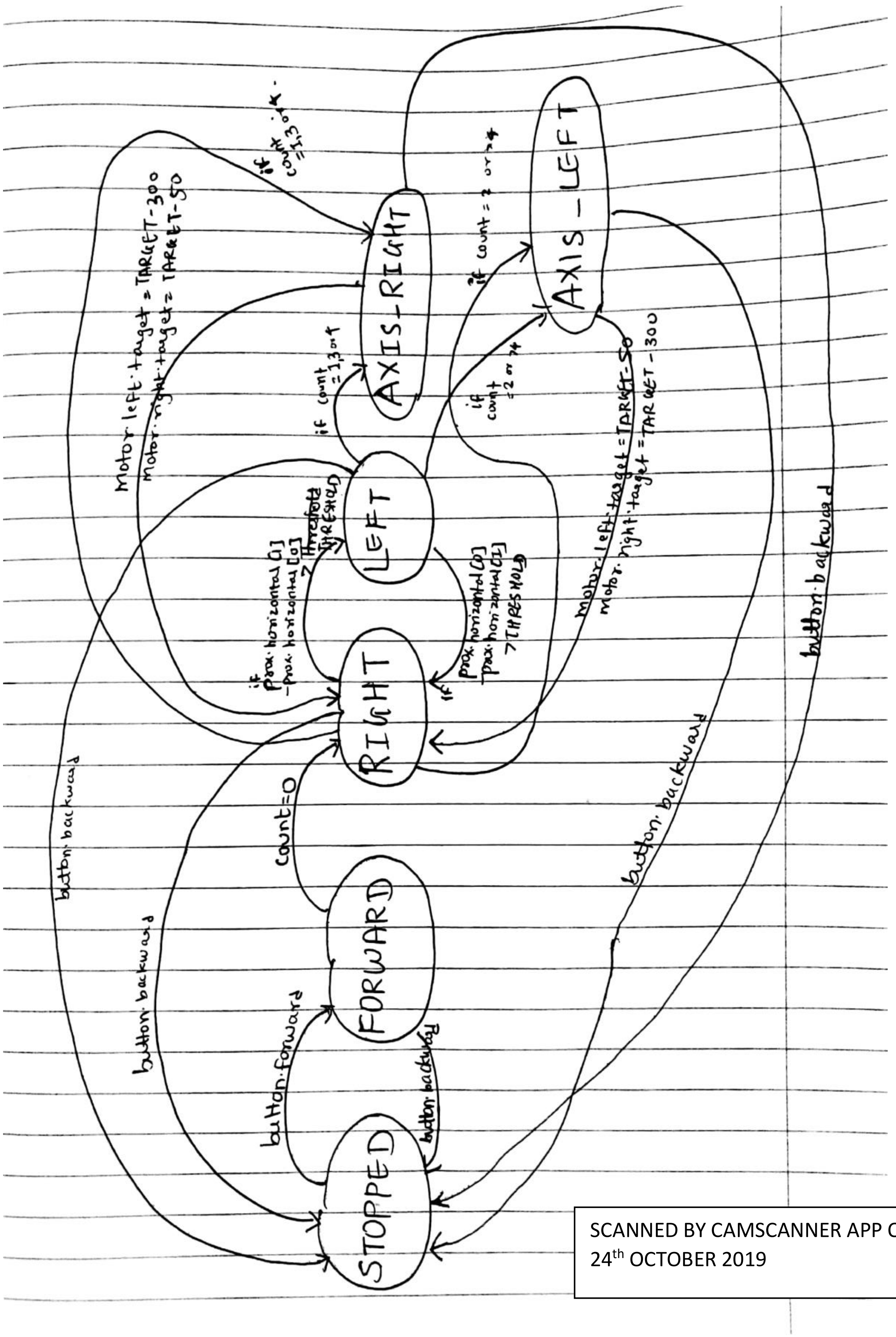
The program is executed in the following steps:

1. First variables are declared which include state, max, min, mean, count = 0 and constants defined are STOPPED = 0, FORWARD = 1, LEFT = 2, RIGHT = 3, TARGET = 500, THRESHOLD = 250, HORIZONTAL_THRESHOLD = 2800, AXIS_LEFT = 4, and AXIS_RIGHT = 5.
2. When the forward button is pressed, the robot moves forward as the speed of both its motors is set to TARGET, and the state is declared as RIGHT along with declaring count to 0.
3. When the robot detects the line by one of its ground proximity sensors, it utilizes both the sensor readings and checks which one is greater and puts the constraint of their difference to be more than the given THRESHOLD to move across the yellow desk without confusing it as the black tape.
   Execution: When the robot detects a line using its 1st sensor, the state is declared to LEFT then the conditional code is executed which makes it go to the right, by decreasing the speed of the left motor to zero which changes the state to RIGHT. Inversely, now when the 0th sensor detects the line, the state is changed to LEFT by decreasing the left motor speed to zero. And so, on it will continue to follow the line.

4. Now when a block appears in front of the robot, the robot makes a right or left turn axially (according to count) and when it detects the line, it continues the line following program. Execution: When the block appears, math.stat native function assigns the maximum value detected by the sensors in max. If max is greater than the HORIZONTAL_THRESHOLD, it increases count(to make specific turns), now it checks the count and accordingly makes a right(state = AXIS_RIGHT) or left(state = AXIS_LEFT) turn axially by setting the motors to inverse speeds. This turn is executed until there is nothing in front of the sensors and then it makes an arc by decreasing the speed of left motor to make a left arc or the right motor to make a right arc and setting the state to RIGHT. When the 0th sensor detects the line again, the program reverts to the line following the code.

   We chose this solution as the arc takes less amount of time to reach the line in comparison to other options like square and others. Other than that, we made the choice of making specific turns as to take up the maximum advantage of the predefined path. One program used by a fellow team (team 3) was incorporated with a recovery mechanism which is lacking in our program. This can be a problem when the robot loses the line. Line following was much slower in our case when compared to some teams (team 1) due to use of only one motor in turning during line follow.

STATE TRANSITION DIAGRAM


STOPPED
FORWARD
RIGHT
LEFT
AXIS_RIGHT
AXIS_LEFT
button.forward
button.backward
button.backward
button.backward
button.backward
button.backward
count=0
if prox.horizontal[1] -prox.horizontal[0] >THRESHOLD
if prox.horizontal[0] -prox.horizontal[1] >THRESHOLD
if count =1,3 or 4
if count =1,3 or 4
if count = 2 or >4
if count =2 or >4
motor.left.target = TARGET-300
motor.right.target = TARGET-50
motor.left.target =TARGET-50
motor.right.target =TARGET-300

SCANNED BY CAMSCANNER APP ON 24th OCTOBER 2019

RESULTS:

Our program executed perfectly scoring a time of 45 seconds both the times. Keeping into consideration that we got a perfect run and so a perfect score for twice, we tried to increase our speed in the third attempt but it made several mistakes as it started to miss the line and at a point in the start, it lost it and moved randomly through the track. But since we were able to find the reasons that caused the errors, we will be able to change that in the future. Relative to other programs, we did well because our program made the second fastest timing: 45 seconds. The reason that we were able to make it well, because we tried different kinds of logic like square, sensor-based movement and ARC which led to the conclusion of ARC being the fastest and it covered the most route in this track as well. While devising the logic for the project and after we finished the code there were several mistakes, but we examined them step by step, solved them by hundreds of runs with changed variable values, and collecting data from them enabled us to choose the most suitable combination of the variables. On the day of the competition, we saw a program being completed in 37 seconds and recognised out main failure which was: Slow Line following.

CONCLUSION AND FUTURE WORK:

Our program performed what it asked of it perfectly. We made the second fastest time in the competition. Overall, in the first and second attempts, we made a perfect run with 45 seconds acc, and in the third attempt, we lost the line in an attempt to increase the speed. We failed to grasp how to increase the speed of the most basic code, line follow. But now that we have acknowledged it, we will be able to make a program that is much more efficient in line following decreasing the time down to around 30 seconds.

We are not quite satisfied with our results, and because of this we decided to incorporate efficient ways and shortcomings in our code:

- Faster Line Follow:
    - By using both the left and right motors we can increase the amount of distance covered between two scans of the black tape/line.
- Recovery Mechanism:
    - Incorporate a recovery mechanism using timers, which makes the robot go axial round and then find the line again.

APPENDIX

```
CONSTANTS:
STOPPED = 0
FORWARD = 1
LEFT = 2
RIGHT = 3
TARGET = 500
THRESHOLD = 250
HORIZONTAL_THRESHOLD = 2800
AXIS_LEFT = 4
AXIS_RIGHT = 5

CODE:
var state = STOPPED
var max
var min
var mean
var count = 0

motor.left.target = 0
motor.right.target = 0

onevent button.forward
state = FORWARD
motor.left.target = TARGET
motor.right.target = TARGET
state = RIGHT
count = 0

onevent button.backward
state = STOPPED
motor.left.target = 0
motor.right.target = 0

onevent prox
if state != STOPPED then
if count <= 5 then
if state == LEFT and (prox.ground.delta[0] - prox.ground.delta[1] > THRESHOLD)then
state = RIGHT
motor.left.target = 500
motor.right.target = 0
```

```
elseif state == RIGHT and (prox.ground.delta[1] - prox.ground.delta[0] > THRESHOLD) then
state = LEFT
motor.left.target = 0
motor.right.target = 500
end
end

end

call math.stat(prox.horizontal[0:4],min,max,mean)

when max >= HORIZONTAL_THRESHOLD do
count++
if count == 1 or count == 3 or count == 4 then
motor.left.target = 0
motor.right.target = 0
state = AXIS_RIGHT
motor.left.target = TARGET
motor.right.target = -TARGET
elseif count == 2 or count > 4 then
motor.left.target = 0
motor.right.target = 0
state = AXIS_LEFT
motor.left.target = -TARGET
motor.right.target = TARGET
end

end
when max < 1 and state == AXIS_RIGHT do
motor.left.target = 0
motor.right.target = 0
motor.left.target = TARGET - 300
motor.right.target = TARGET - 50
state = RIGHT
end
when max < 1 and state == AXIS_LEFT do
motor.left.target = 0
motor.right.target = 0
motor.left.target = TARGET - 50
motor.right.target = TARGET - 300
state = RIGHT
end
```