EEE3099S Milestone 2

Group Number	32
Date	22/08/2019
Hyde,Liam	HYDLIA001
Naidoo,Vikyle	NDXVIK005
Ignatius,Crustus	IGNCRU001
Nkwinika,Nyiko	NKWNYI001

1 WORK DISTRIBUTION

Table 1 Work distribution for Milestone 2

Task Description	**Person(s) of Responsibility**
<task>	<student number>, <student number>
Sensor 1 Introduction	HYDLIA001
Sensor 1 (Line Sensor) Circuitry	HYDLIA001, NDXVIK005
Sensor 1 Building and Testing	NDXVIK005
Algorithm 1	IGNCRU001
Compilation and editing	NKWNYI001

2 Sensor 1 Introduction

2.1 Aim

The aim of the sensor is to detect whether the robot is following the line, and feed information back to the micro to allow adjustments in direction, selection of routes and route optimisation.

2.2 Sensor 1 User Requirements

1) The sensor must have the ability to detect the difference between white and black on the board.
2) The sensor must be able to run off the available voltage.
3) The sensor must fit within the required space of 3cm by 3cm.
4) The sensor must be able to detect the end of the maze.
5) The sensor must be accurate and repeatable.

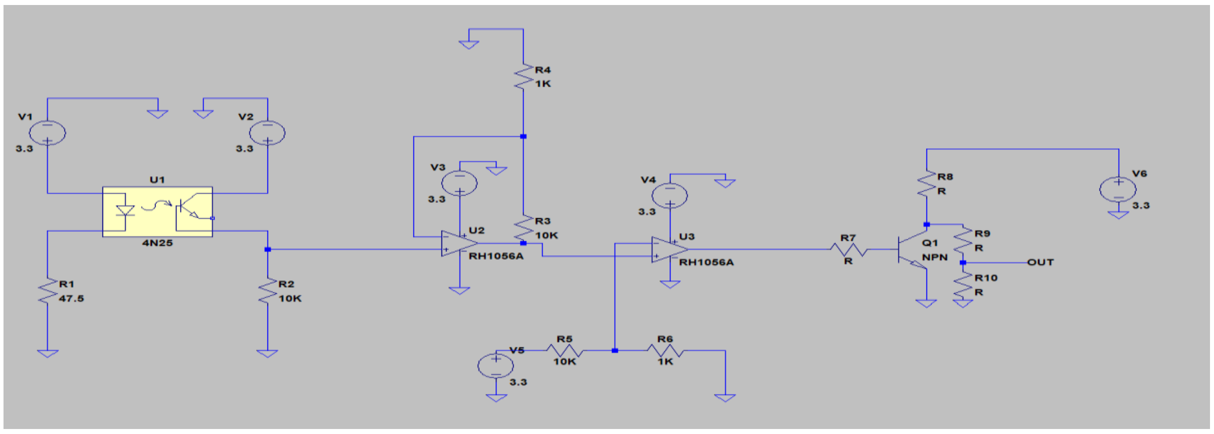
2.3 Sensor 1 Technical Requirements

1) The sensor must run off a variable voltage (7V to 12V)
2) The sensor must have a 3 pin molex header for VCC, GND and Digital output
3) The sensor must have a digital output. This means a level shifting circuit must be included.
4) The sensor should have two outputs. High between 2.8 and 3.3 V, Low between 0 and 0.5V
5) Should be able to detect the line from 5mm above the line to 30mm above the line
6) The result should be accurate, with no noise. The output needs to be a DC voltage level with no measurable noise.
7) Operate at shifting light levels. I.E. no detectable ambient light to extremely high levels of ambient light. IR sunlight should also be accounted for.
8) Sensor should operate using IR diodes.

2.4 Algorithm 1 Requirements

1) The robot must be able to receive positional information from the sensors (End, cross roads, turns, t junctions, end of tracks)
2) The robot must be able to receive directional information from the sensors (Drifting left, drifting right, reverse, forward, right turn and left turn)
3) The robot must be able to follow every available path on the board
4) The robot must be able to optimize the paths on the board and select the shortest path
5) The robot must be able to exit loops that it can get stuck in
6) The robot must be able to restart the track and follow the optimal path

3 Sensor 1 (Line Sensor) Circuitry

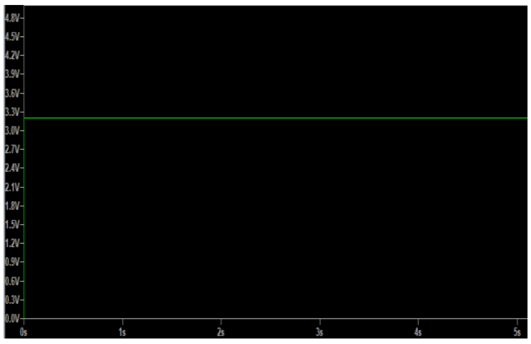
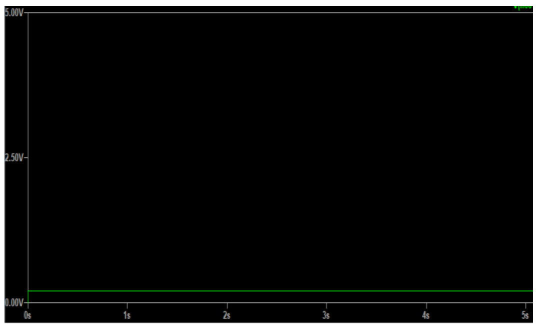


3.1 Description and Circuit Explanation

Circuit components: As LTSPICE doesn't have all the components that we will be using in our circuit, some equivalent components had to be used.

1) This is the simulation for the LED’s and Photodiodes.
1) This is an amplifier/buffer
2) This is a comparator, to detect between black and not black.
3) This is a level shifter to convert to digital signals

3.2 Simulation Result and Discussion

Output when Diode is “detecting black”	Output when Diode is “not detecting black
	

3.2.1 Discussion

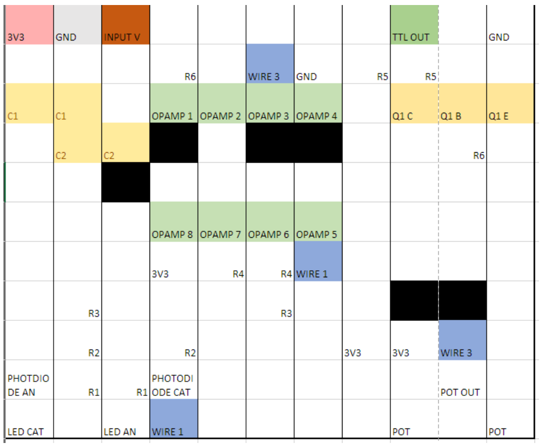
Our circuit outputs values within range of the functional requirements. 0.2V above ground can be attributed to the level shifter circuit. 3.2V is when the circuit is high and this can be attributed to drops across components in the circuit. A more ideal circuit is not possible, as even rail to rail op amps would cause slight voltage differences at the base of the level shifting circuit, creating a voltage above 0 and below 3.3. While these values are not a problem, it is useful to use the internal pull up and down resistors in the STM32 to create the ideal input, to make coding easier.

3.3 Bill of Material

Item #	Distributor Part Number	Manufacturer Part Number	Manufacturer	Description	Quantity
1	whitelab	MCP1700	Texas Instruments	voltage regulator: 7v to 3v3	5
4	654-8895	SFH 213 FA	OSRAM Opto Semiconductors	photodiode 750-1100nm	5
5	whitelab	TSAL6100	VShay	infrared led 940nm	5
6	661-0530	LM324N	Texas Instruments	opamp	10
	whitelab		-	resistor 1k, 10k 47	30
	whitelab	2N2907	-	PNP transistor	5
	whitelab	2N2222	-	NPN transistor	5

4 Sensor 1 Building and Testing

4.1 Veroboard Design



4.2 Testing Procedure

1 check if each module is giving desired output as it should from the circuit diagram.
2 if not correct output
3 identify fault
4 fix fault
5 go to 1
6 connect all modules together
7 if correct output from level sensor
8 circuit working
9 else identify fault
10 go to 7

4.3 Result

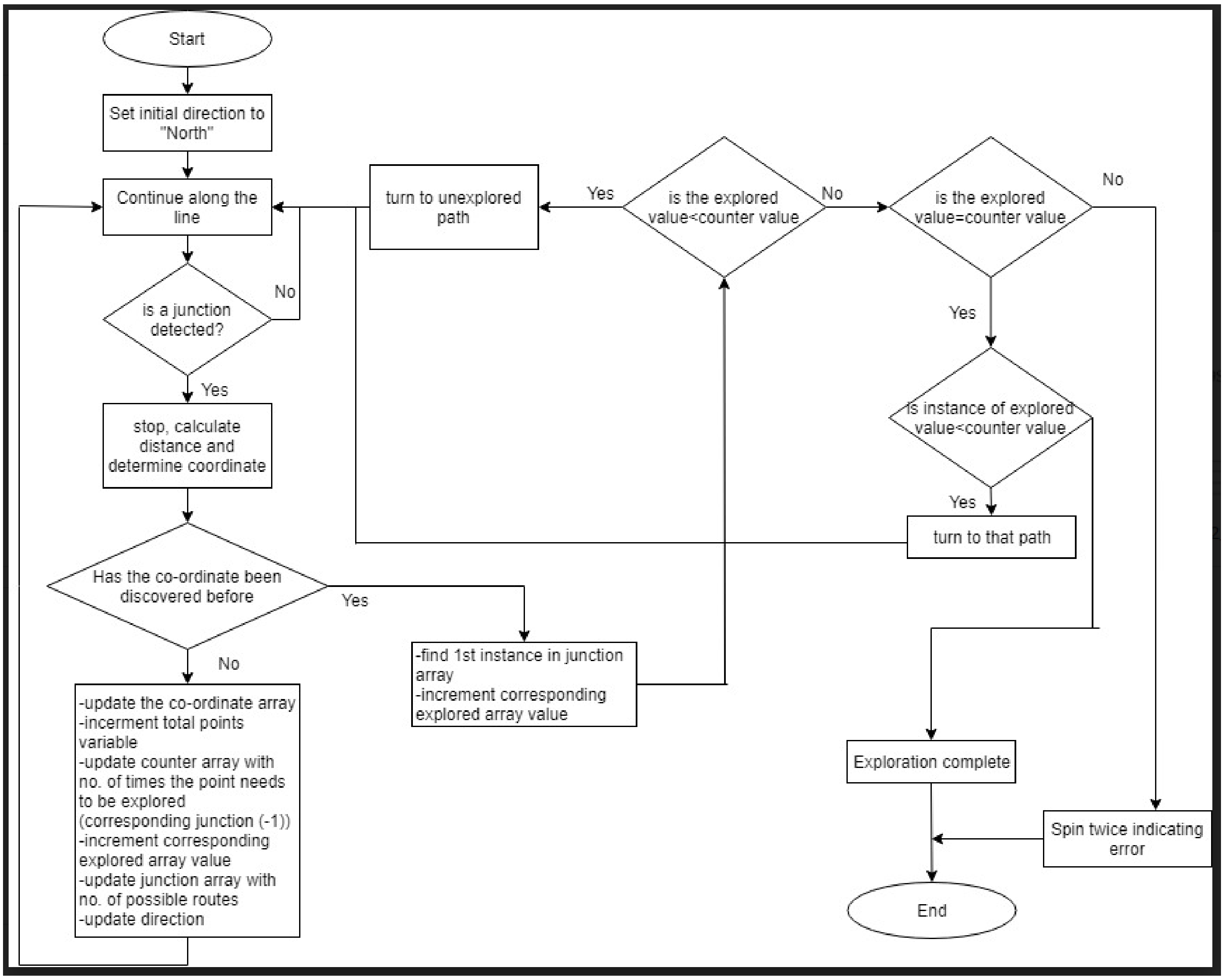
Final sensor circuit on Vero board was working correctly. If white surface detected: output low. If black surface detected: output high.

5 Algorithm 1

5.1 Approach and Explanation

An algorithm is required to enable the robot to learn the maze by following a black line and find the shortest solution to exit the maze (at which point it should stop and an LED should blink) which is indicated by a black circular patch of 150mm diameter. Once this is done, the robot the robot is returned to the starting position and should complete the maze as quickly as possible when the button is pushed again. There are several maze-solving algorithms were considered. This includes the 'random mouse algorithm', which solves the maze by taking a random turn every time a junction is encountered. However, this is a very laborious method and is highly likely to fail as the learning process is also a timed event. The second consideration was a 'wall following' algorithm where the robot commits to only turning in a pre-determined direction (such as only left wherever possible). This is a more efficient than the 'random mouse algorithm' but since the maze is not a simple one (it may contain loops and multiple solutions), it does not necessarily ensure that the shortest path has been taken. This led to the 'shortest path algorithm'. This algorithm maps out the entire map by exploring every single path on the maze along with the distances between junctions. This is done with the use of 4 arrays (a **2D-coordinates array** which stores (x, y)-coordinates, a point counting array which counts how many time a point has been explored, a **junction array** which counts how many times the point needs to be explored based on the nature of the junction (T-junction, crossroads, etc.), a **counter array** which counts how many explorations are still required and **an explored array** which keeps track of how many of the paths in a junction have been explored). Two counting variables are also needed to keep track of the total points discovered and the direction that the robot is traveling in (using a cardinal system with 'North' defined as the staring direction). Once the entire maze is explored, the algorithm returns the maze in a graph format where a shortest path finding algorithm (e.g. Dijkstra's Algorithm) can be applied to solve for the shortest path. This method is elaborated on in the flowchart that follows.

5.2 FLOWCHART


Start
Set initial direction to "North"
Continue along the line
is a junction detected?
No
Yes
stop, calculate distance and determine coordinate
Has the co-ordinate been discovered before
Yes
No
-update the co-ordinate array
-incerment total points variable
-update counter array with no. of times the point needs to be explored (corresponding junction (-1))
-increment corresponding explored array value
-update junction array with no. of possible routes
-update direction
-find 1st instance in junction array
-increment corresponding explored array value
turn to unexplored path
Yes
is the explored value<counter value
No
is the explored value=counter value
No
Yes
is instance of explored value<counter value
Yes
turn to that path
Exploration complete
Spin twice indicating error
End