

MDarzins - M4X 01506

by Martin Darzins

FILE	19452_MARTIN_DARZINS_MDARZINS_- _M4X_01506_241917_1073722244.DOCX (3.83M)		
TIME SUBMITTED	01-MAY-2020 03:48PM (UTC+0100)	WORD COUNT	5669
SUBMISSION ID	125857688	CHARACTER COUNT	30499

University of Wales

Trinity Saint David

School of Applied Computing

Title: Research into Robotics in a Lego Mindstorms setting



Module: Integrating Project

Code: M4X 01506

Student: Martin Darzins

Student no.: 1909722

Programme: Foundation Year (Computing and Electronic Streams)

Lecturer: Sue Maw

Due Date: 7th May 2020

1 ACKNOWLEDGEMENTS

Due to the coronavirus situation and various governmental restrictions placed on fellow Applied Computing students, the author acknowledges contributions from the UWTSD Swansea facility members and the star of 1956 film 'Forbidden Planet', Robby the Robot.

2 TABLE OF CONTENTS

1	ACKNOWLEDGEMENTS.....	2
3	INTRODUCTION.....	6
3.1	AIMS.....	6
3.2	OBJECTIVES	6
3.2.1	<i>Overall Robotic Task.....</i>	<i>6</i>
4	COMPONENT 1 – REPORT.....	7
4.1	GENERAL ASSUMPTIONS	7
4.2	LINE FOLLOWING.....	7
4.2.1	<i>Assumptions and Concerns</i>	<i>8</i>
	• <i>Boolean (Zig-Zag) Line follower</i>	<i>8</i>
	• <i>Multi-sensor line follower.....</i>	<i>9</i>
	• <i>Proportional line followers</i>	<i>9</i>
	• <i>PID line follower.....</i>	<i>10</i>
4.2.2	<i>Line Follower Conclusions/Summary.....</i>	<i>11</i>
4.3	OBJECT AVOIDANCE.....	12
4.3.1	<i>Assumptions and Concerns</i>	<i>12</i>
4.3.2	<i>Object Avoidance Equipment</i>	<i>12</i>
	• <i>Colour sensor</i>	<i>12</i>
	• <i>Touch sensor.....</i>	<i>13</i>
	• <i>Infra-red method (IR).....</i>	<i>13</i>
	• <i>Ultrasonic method (US)</i>	<i>13</i>
4.3.3	<i>Design Build options.....</i>	<i>13</i>
	• <i>Bulldozing the Obstacle (Robodoz3r)</i>	<i>14</i>
	• <i>Forklifting the Obstacle (Bobb3e).....</i>	<i>14</i>
	• <i>Grabbing/Lifting the Obstacle</i>	<i>14</i>
	• <i>Climbing Over the Obstacle (Mars Rov3r)</i>	<i>14</i>
	• <i>Avoiding the Obstacle.....</i>	<i>15</i>
4.3.4	<i>Object Avoidance Conclusions/Summary.....</i>	<i>15</i>
4.4	3D MAZE NAVIGATION	15
4.4.1	<i>Assumptions and Concerns</i>	<i>15</i>
4.4.2	<i>Maze Navigation Methods.....</i>	<i>16</i>
	• <i>Random Mouse.....</i>	<i>16</i>
	• <i>Pledge</i>	<i>16</i>
	• <i>Trémaux.....</i>	<i>16</i>
	• <i>Dead-End Filling.....</i>	<i>16</i>
	• <i>Shortest Path</i>	<i>17</i>
	• <i>Recursive.....</i>	<i>17</i>

•	<i>Maze-Routings.....</i>	<i>17</i>
•	<i>Wall Following (Left-or-Right hand rule).....</i>	<i>17</i>
•	<i>Line Following.....</i>	<i>18</i>
4.4.3	<i>3D Maze Navigation Conclusions/Summary.....</i>	<i>18</i>
4.5	<i>OVERALL SUMMARY</i>	<i>18</i>
5	COMPONENT 2 – FLOWCHARTS / PSEUDO CODE	20
5.1.1	<i>Requirements Analysis</i>	<i>20</i>
5.1.2	<i>MoSCoW.....</i>	<i>21</i>
5.2	<i>FLOWCHARTS</i>	<i>22</i>
5.3	<i>PSEUDO CODE.....</i>	<i>23</i>
5.3.1	<i>Line Follower</i>	<i>23</i>
5.3.2	<i>Object Avoidance</i>	<i>24</i>
5.3.3	<i>Maze Navigation.....</i>	<i>24</i>
6	REFERENCES.....	26
7	REPORT PLANNING METHODOLOGY.....	27
Figure 1 -	Typical Robot Course Layout	6
Figure 2 -	Gear Down set-up	7
Figure 3 -	Typical Line Follower Robot.....	7
Figure 4 -	Colour Sensor on line.....	8
Figure 5 -	Calibrating Colour Threshold Values	8
Figure 6 -	2 Sensor Robot	9
Figure 7 -	Multi-Sensor Robot.....	9
Figure 8 -	Typical Proportional line programme code.....	9
Figure 9 -	More complex Proportional code.....	10
Figure 10 -	Typical more complex PID code.....	11
Figure 11 -	Example of Colour Sensor shielding	12
Figure 12 -	Colour Sensor	12
Figure 13 -	Touch Sensor.....	13
Figure 14 -	Infra-Red Sensor (IR)	13
Figure 15 –	Ultrasonic Sensor (US)	13
Figure 16 -	RoboDoz3r	14
Figure 17 -	Bobb3e.....	14
Figure 18 -	Grabber/Lift	14
Figure 19 -	Mars Rov3r.....	14
Figure 20 -	Typical Line Follower & Object Robot design.....	15
Figure 21 -	Author Designed Robot (Robb3e).....	19

Table 1 - Requirements Analysis (Set-Up).....	20
Table 2 - MoSCoW Haves	21
Table 3 - Typical Visio Flowchart Symbols	22
Table 4 - Flowchart Overview (Visio extract).....	22
Table 5 - Line Follower Mindstorms code extract	23
Table 6 - Object Avoidance Mindstorms code extract	24
Table 7 - Typical Wall Follower Mindstorms code.....	25

3 INTRODUCTION

3.1 Aims

The aim of this report is to provide a report covering research into Robotics in a Lego Mindstorms setting, showing how an EV3 Lego Mindstorms robot can successfully negotiate and surmount various problems encountered following a designated line path, encompassing both an obstacle and 3D maze.

3.2 Objectives

The purpose of the assignment is to show the understanding and evaluation process of a problem; to conduct relevant research; formulate requirements and constraints along with analysing the issues and producing logical solutions in order to reach the Red block finishing point, from a designated starting position.

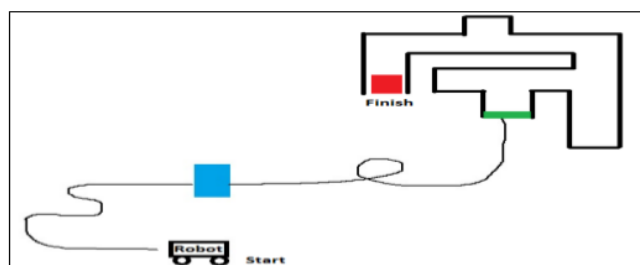
Split into 2(two) main components (Report and Flowcharts/Pseudo code), the report will incorporate both components and detail the following;

- Introduction and problem to be solved.
- Research covering the main key aspects of:
 - Line following (Black line)
 - Obstacle avoidance (Blue block)
 - Negotiating a physical 3D Maze (starting at Green tape / ending at Red block)
- Strengths and weaknesses of the various methods identified, along with a conclusion as to which are determined to be the most suitable.
- The Flowcharts and Pseudocode will show the logic of how the EV3 robot intends to satisfactorily negotiate the course with sufficient detail to enable a program to be installed and complete the course.

3.2.1 Overall Robotic Task

The Robot will be required to negotiate the course as generally specified in figure 1 below, starting at a pre-determined position, it is required to follow the marked **Black line**, negotiate the **Blue block**, re-join the line, enter (at **Green tape** spot) and navigate the 3D maze finally reaching the **Red block** finishing position. At this point the Robot stops and plays a fanfare announcing the end of the course trial task.

Figure 1 - Typical Robot Course Layout



4 COMPONENT 1 – REPORT

As this task encompasses 3(three) distinction aspects of; Line following, Obstacle avoidance and Maze navigation, each will be investigated individually and all incorporated into the 1(one) EV3 Robot design.

4.1 General Assumptions

Floor surface conditions and area under which the Robot is to perform, allow free and unhindered Robot wheel turning and movement accessibility, thus catering for the size of the designed Robot.

Once the task activity commences, the Robot designers/builders are not allowed to re-adjust Robots' position on the course, but must program the Robot to make its own changes in order to complete the activities.

Whilst there may be reasonable time restrictions placed on the completion of this activity, there is no need to have a Robot designed for speed. Greater control of the Robot's torque and speed will assist the sensor's ability to accurately determine the line threshold values, so the Lego Gear Bot design (incorporating the gear down) cog method has been utilised from the Lego type 45544 set Education edition components [1].

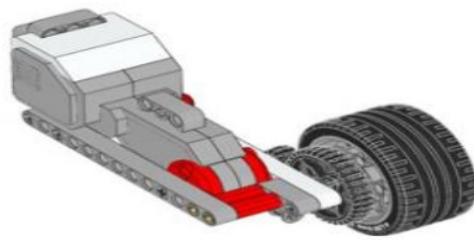


Figure 2 - Gear Down set-up

4.2 Line Following

The Task is for the designed EV3 Robot to follow the Black line and reach the entrance/start of the 3D Maze (Green tape) destination. A number of differing methods can be adopted to enable the Robot to follow a line utilising Light/Colour Sensors, the main examples explored below cover; Boolean, Multi-Sensor, Proportional and PID line following methods.

Figure 3 - Typical Line Follower Robot



4.2.1 Assumptions and Concerns

The line to be followed is Black in colour and of sufficient density and line/size thickness (ideally at least 3cm wide) to enable it to be recognised by the EV3 colour sensor, as too small a line will not be read accurately by the sensor.

The line needs to be much darker (measurable contrast) than the background to enable the sensor to be able to distinguish between them, hence why the background light setting environment is important when selecting either Colour, Reflective Light or Ambient Light Intensity modes.

For the line following task, the Colour sensor will be utilised selecting the Reflective Light Intensity in Compare mode, along with determining the threshold values of between 0 – 100, with 0 being very dark (i.e. not reflecting light well) and 100 being very bright (i.e. a lot of light being reflected).

The starting position of the Robot/colour sensor is pre-determined as the initial condition and will follow either the Left or the Right edge of the line, this edge reading will ideally give a reading of 50, but may differ subject to scenario light setting conditions.

- Boolean (Zig-Zag) Line follower

This method is the simplest to deploy with adjustments just being corrections left or right, however, as it follows the line edge (ether left or right side) it has no straight case situation and will always zig-zag. It does not adapt its turning radius, will always keep to the programmed turning setting established and therefore will not cope well with line direction changes.

Figure 4 - Colour Sensor on line

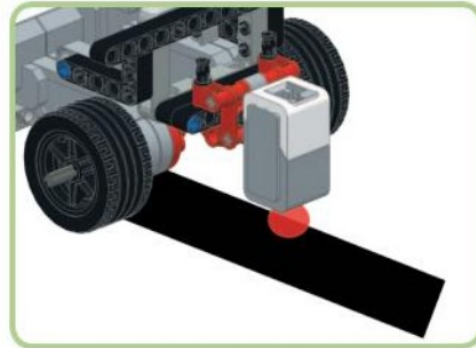
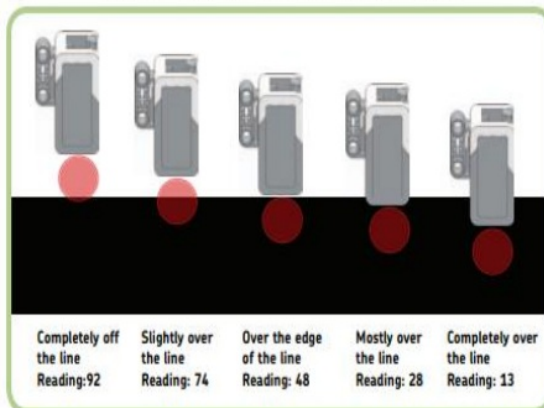
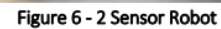
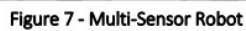


Figure 5 - Calibrating Colour Threshold Values



This simple line follower utilises the Switch block to determine whether it turns left or right and Colour sensor in Reflective Light Intensity mode to constantly adjust the turns in comparison to the light sensor pre-determined threshold reading. This Switch block itself is contained within a Loop block so that the programme continues until stopped.

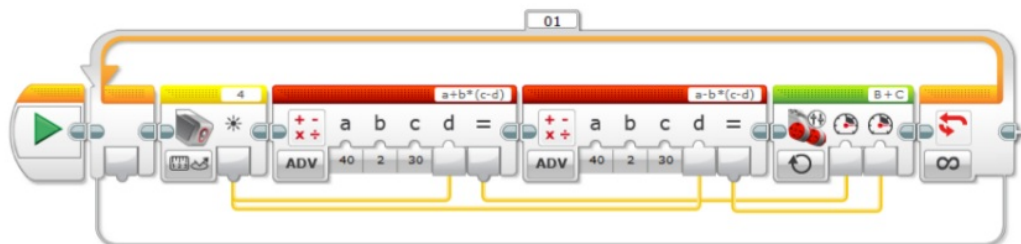
- With 2 (or more) sensors placed either side of the line edges, this type of line follower will now include the straight-line case situation. It is still relatively simple to create and less likely to lose the line when compared to the (1 one sensor) zig-zag line follower.



- Proportional line followers

Even though there are additional mathematical functioning and data wires to complete, it is still relatively simple to programme and is adjustable, offering smoother line following capabilities, with steering being able to adapt relative to error. It is less likely to lose the line and is better at adapting to a more complex curved line with tighter turns.

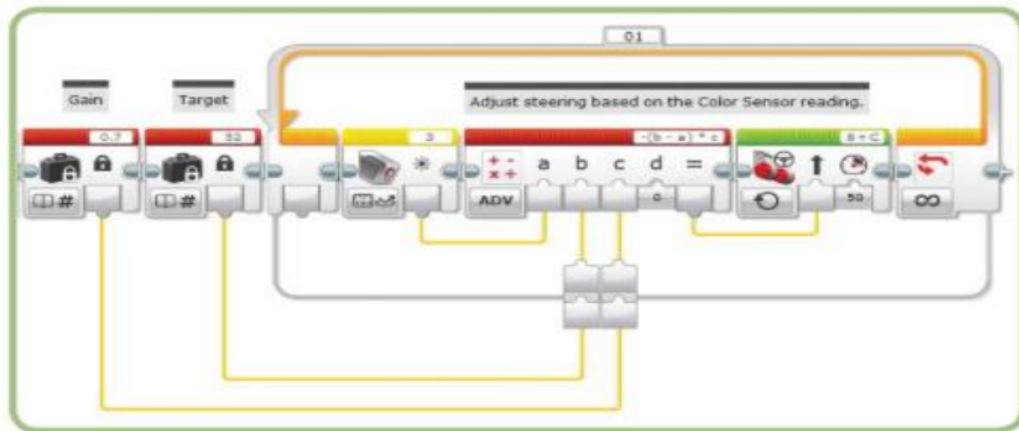
Figure 8 - Typical Proportional line programme code



The example programme above uses a Colour Sensor block set to Measure the Reflected Light Intensity to take a light reading and then feeds this value into the Math blocks. The Math Blocks are configured to evaluate these formulas: Left motor = $a + b \times (c - d)$, Right motor = $a - b \times (c - d)$, where: a = default motor power, b = gain, c = target value and d = sensor reading. The difference between the target value and the sensor reading, $(c - d)$ in the above formula example is known as the error [2].

Differing programmes are available, where a = Colour sensor reading, b = Target value and c = Gain value

Figure 9 - More complex Proportional code



In either case, all require extra time to determine the Error and Steering values, as well as additional time spent on fine tuning. Should more than 1 sensor be utilised then extra equipment costs are also involved, as set 45544 only contains 1 colour sensor [3], but overall, they are an improvement on both the Zig-zag and Multi-sensor options.

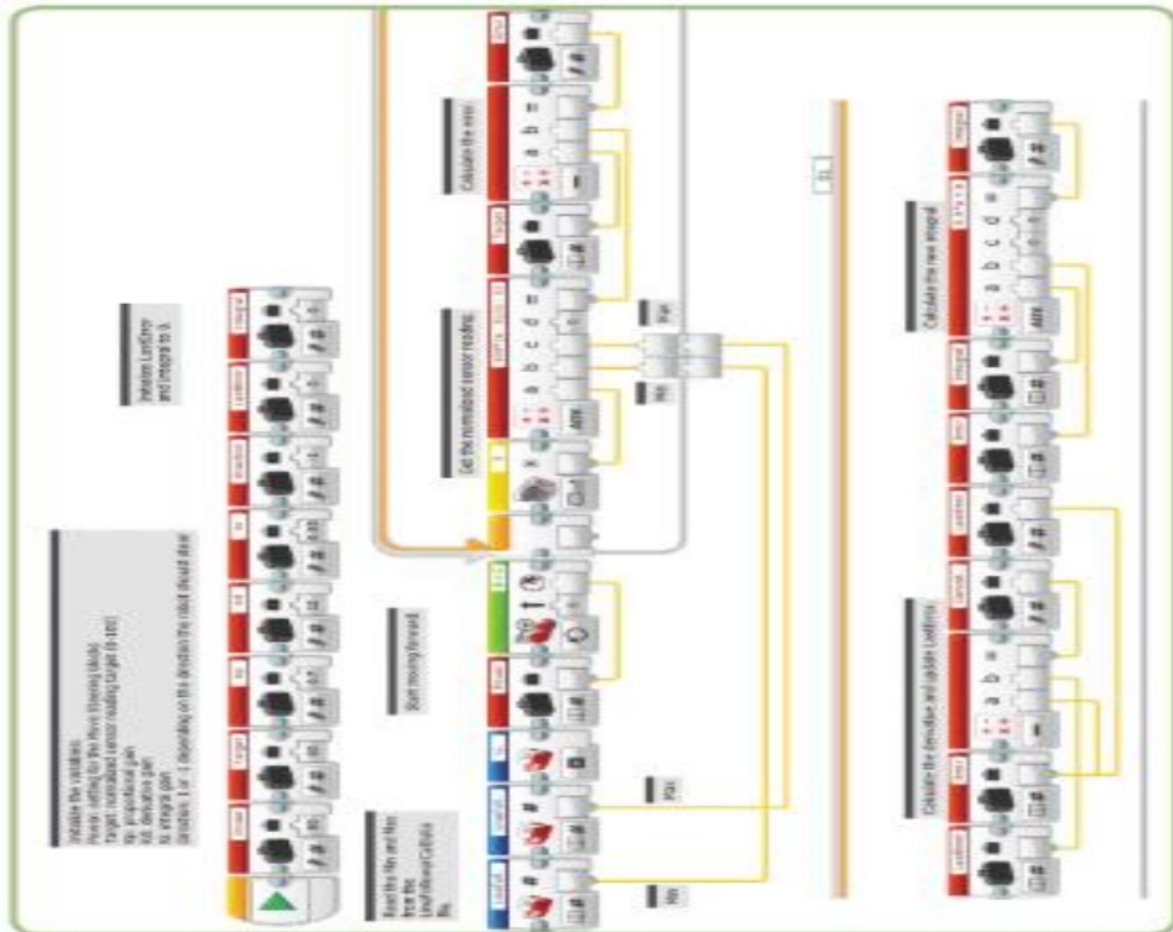
- PID line follower

These Proportional Integral Derivative (PID) line followers are the most advanced and complex line followers, they read its overall trend over time and predicts what's its own next error is going to be. These attributes make it the most accurate, advanced, smoothest, adaptable, adjustable and reliable line follower from all those identified above. The proportional element adjusts the steering according to its current distance; integral value detects any drifting by the Robot over time and the derivative determines whether the Robot is travelling along a curve or straight line via the latest measurement comparisons.

This PID capability is a control system algorithm with feedback mechanisms to predict errors by comparing output variables to a reference value and applies corrections on proportional, integral and derivative terms. This ability to predict and minimise error in an output variable makes the line follower move smoothly along and closer to the line with greater pace and responsiveness to line change directions [4].

But, as can be seen from the extracts of a typical PID programme below, it requires a high degree of mathematical understandings and encompasses large, unwieldy, complex programming. These in turn require more time to determine and fine tune the various K_p , K_i , and K_d arbitrary values [3, 5].

Figure 10 - Typical more complex PID code



4.2.2 Line Follower Conclusions/Summary

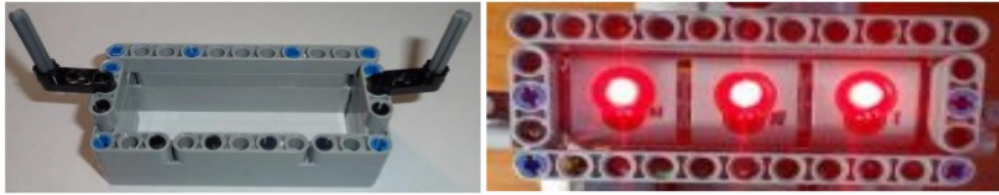
Of the 4(four) generic types of colour/light sensor centric line follower methods, detailed above, it is clear that the PID line follower is the most advantageous method to utilise.

However, dependent upon the Users technical/programming ability and time available, the much simpler Zig-zag/Boolean method will also complete the task, albeit with much less finesse.

This leaves the Proportional line follower method as probably the most utilised option, as it offers a number of the desired benefits with some additional programming effort.

In either case, additional shielding against ambient light could be a consideration for the Colour/Light Sensor, as this will improve (make more accurate) the various threshold readings [6].

Figure 11 - Example of Colour Sensor shielding



4.3 Object Avoidance

As part of the line following exercise, the Robot is required to negotiate a stationary Blue block and continue back on (re-join) the line path in order to arrive at the 3D Maze (Green tape) entrance point.

To avoid an object you must first be able to detect it, three main methods of detection are either by; utilising an ultrasonic sensor (US); an infra-red sensor (IR) where both determine distance, or via the Touch sensor which physically touches the object to determine its presence [7].

When utilising either US or IR there are a number of differing design build options that can be adopted to enable the Robot to avoid the Blue block obstacle, the main examples shown below cover; Moving (Bull dozing), Carrying (Fork lifting), Grabbing/Lifting, Climbing-over (Rov3r) and Avoiding the obstacle.

4.3.1 Assumptions and Concerns

That the stationary Blue block is positioned on the Black line, being constructed sturdily and stable enough to be moved, carried or driven over, without impairing the Robot's functions.

The Block is able to be re-positioned, handled and be of suitable size to be recognised by the EV3 Colour and/or Ultrasonic sensors.

It has been established that Turning degrees will vary depending upon Robot weight, speed employed and any time constraints allocated to turn, therefore trialling the Robot is essential so as not to rely solely on theoretical designs.

Furthermore, by utilising the US sensor compare distance function and adding Loop to programme will allow Robot to perform task consistently and Switch commands to determine different scenarios.

4.3.2 Object Avoidance Equipment

- Colour sensor

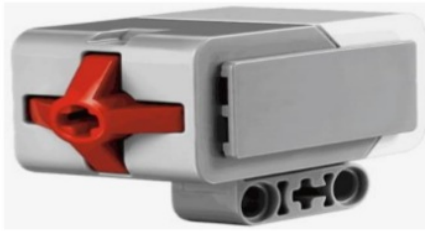
This sensor measures the colour or brightness of light entering the front window of the sensor, when used in; Colour, Reflective and/or Ambient Light Intensity modes, it helps determine the item. Whether this be the Black line (Line Follower), Green tape (Maze entrance), Blue Object (Avoidance) or the Red Block (Maze end).



Figure 12 - Colour Sensor

- Touch sensor

Figure 13 - Touch Sensor



The Touch sensor relies solely on contact with the object and is totally unaware of the obstacle until contact is made.

- Infra-red method (IR)

The IR method measures distance by sending out an infra-red signal and detecting the reflection of the signal bouncing back off an object. It also has the ability to operate in Beacon mode, which can detect an infra-red beacon and will calculate the beacon's proximity and heading/angle based on the signal hitting the sensor. Furthermore, in Remote mode, it can detect a button (or combination of buttons) being pressed on a remote-control device, which can be transmitted into the programme. This remote-control option is only available via IR not US.

Figure 14 - Infra-Red Sensor (IR)



- Ultrasonic method (US)

Figure 15 – Ultrasonic Sensor (US)



The US method measures distance by sending out high-frequency sound waves that will bounce off an object and measure how long it takes for the sound to reach back to the sensor. Amongst its abilities the US sensor has a Listen mode in which it can 'listen out' for other US sensors. Other benefits are that it is; more accurate than IR when measuring distance; is

less susceptible to interference and has a larger measuring range of up to 255 cm, as opposed to 100 cm for IR.

4.3.3 Design Build options

Of the 3(three) methods the superior design options will utilise either the IR or US sensor to detect the object, possible build design options for these methods are;

- Bulldozing the Obstacle (Robodoz3r)
Probably the simplest design method to adopt, but object may still sit directly in front, so object would need to be pushed from the side to remove it from the line.



Figure 16 - RoboDoz3r

- Forklifting the Obstacle (Bobb3e)



Figure 17 - Bobb3e

The Object itself would need to have the facility for it to allow forks to pick-up.

- Grabbing/Lifting the Obstacle
With no details as to size of object, the grab arm size would be difficult to determine.



Figure 18 - Grabber/Lift

- Climbing Over the Obstacle (Mars Rov3r)



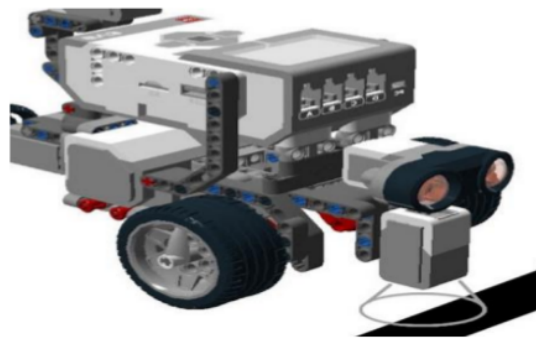
Greater effort required in programming the multi-suspension and Robot build more complicated.

Figure 19 - Mars Rov3r

- Avoiding the Obstacle

Build design for avoidance will cater for all sizes of obstacle as no physical contact is required with object and will also cater for the requirements of 3D Maze completion.

Figure 20 - Typical Line Follower & Object Robot design



4.3.4 Object Avoidance Conclusions/Summary

The Touch method is the least accurate as it does not measure the distance between Robot and object, relying on trial and error repetition touching, to determine the object position.

Utilising the IR method will be more accurate as it does measure the distance between Robot and object, however, the most accurate method of avoiding the object will be through the use of the US sensor method which measures more accurately the distance from object.

4.4 3D Maze Navigation

Similarly, following on from the Line Following and Object Avoidance exercises, the Robot is now required to negotiate a fixed 3D (physical) Maze from the (Green tape) entrance point on through to its end, signified by reaching a Red Block, where it announces the end to the entire task assignment.

There are a number of differing methods to navigate a 3D maze, the most popular (for Lego Mindstorms) being Wall Following potentially utilising Touch, US, IR, Gyro and/or Light (Colour) sensor options. Other methods that could be utilised inside the maze with no advance information of the maze include; Random Mouse, Wall/Line Follower, Pledge and Trémaux. As opposed to methods that require the entire maze to be recognised at once which are; Dead-End Filling, Shortest Path, Recursive and Maze Routing.

Due to the absence of a line within our 3D Maze, specialised (non-Lego) equipment and technical programming ability, the options explored in this assignment relate solely to the Wall Following, not the Line Following or the more complex methodologies.

4.4.1 Assumptions and Concerns

The Green tape maze entrance lies directly at the end of the Black line being followed and of sufficient colour density and line/size thickness to enable it to be recognised by the EV3 colour sensor.

The 3D maze design size (both width & height) is sufficient to allow free and unfettered Robot access and (if required) will be adjusted to suit the size of the Robot designed and suitable to be recognised by the EV3 Ultrasonic sensors.

The 3D maze design will only include straight lines and/or 90 Degree turns, no curves, but may have 'dead-ends' and will only have 1(one) open starting entrance (Green tape) and 1(one) designated finishing position (Red block) and suitable for it to be recognised by the EV3 colour and Ultrasonic sensors. Both these start/finish positions are on the outside edge of the maze, not contained inside the maze itself.

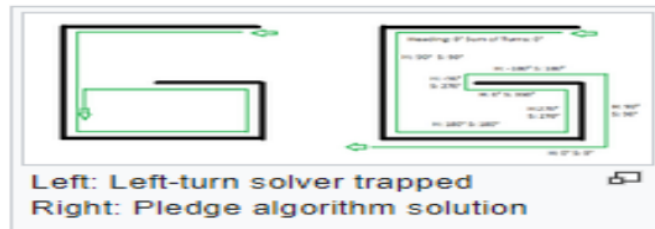
4.4.2 Maze Navigation Methods

- Random Mouse

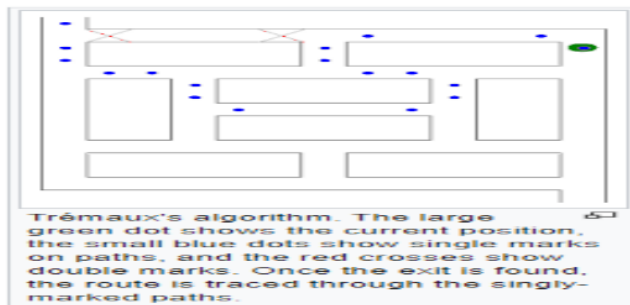
This method follows a passage until a junction is detected and then randomly decides the next direction to follow. Utilised by a simple robot it will eventually find a solution, but will be exceptionally slow [8].

- Pledge

The Pledge algorithm, designed to circumvent obstacles, requires an arbitrarily chosen direction to go toward, which will be preferential [9].



- Trémaux



The Trémaux algorithm marks a path and nodes throughout a maze as they are visited, this path from a junction is either unvisited, marked once or twice. It will find the exit for all well-defined mazes, but not in the shortest time [10].

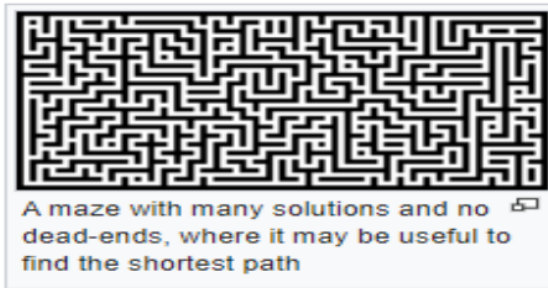
- Dead-End Filling

This algorithm solves mazes by filling all dead ends, leaving only the correct ways unfilled. It is not useful if started inside an unknown maze since this method looks at the entire maze at once. The method is to first find all of the dead-ends in the maze, and then "fill in" the path from each dead-end until the first junction is met [11].



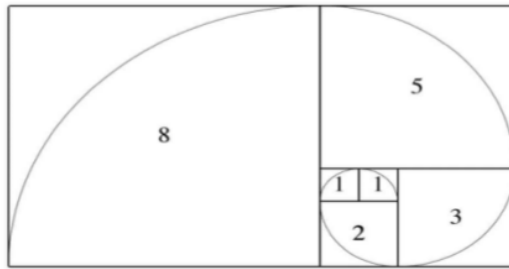
- Shortest Path

A shortest path algorithm may be required for a maze containing multiple solutions, with a number available to find the shortest route, many originating from Graph Theory, with main proponents being Dijkstra and Bellman Ford [12].



- Recursive

With a total view of the maze, a recursive algorithm can determine how to get to the end by calling itself with smaller/simpler input values and which obtains the result for the current input by applying simple operations to the returned value for the smaller/simpler input. Recursive algorithms divide the problem into sub-problems of the same nature. The result of one recursion is the input for the next recursion [13].



Because each term after the second term is the sum of the previous two terms, the Fibonacci sequence is recursive. As shown, the first two terms are 1 and 1. The third term is the previous two terms added together, i.e. $1+1=2$.

- Maze-Routings

The maze-routing algorithm overhead method finds its way between any two locations of the maze and works for any grid-based maze. It can detect when there is no path between the source and destination. Requiring 4 variables to determine the path and detect an unreachable location, it will not to establish the shortest path [14].

- Wall Following (*Left-or-Right hand rule*)

The Wall Following method is the best-known rule for traversing mazes, by keeping one (either Left or Right) hand in contact with one wall of the maze, it is guaranteed not to get lost and will reach a different exit if there is one; otherwise, the algorithm will return to the entrance having traversed every corridor next to that connected section of walls at least once.



The Wall follower method only works on mazes without tunnels, bridges or floating walls and must have a start and end point at the outer edges of the maze (i.e. not in the centre). Only if the above conditions are met will the Left or Right wall following method eventually succeed to find the exit, albeit not necessarily via the shortest (or quickest) route.

- Line Following

This method relies upon the route being established on the floor of the maze, therefore does not meet the task criteria and is not considered.

4.4.3 3D Maze Navigation Conclusions/Summary

Whilst all the methods detailed above will (with varying degrees of success) meet the task set in the assignment, it is clear that only the Line and Wall Follower techniques will work satisfactorily, in light of the current equipment, time and user programming ability constraints. Of these 2(two) 'Following' methods only the Wall Follower will succeed, within the set parameters of the physical 3D maze, as no line is present to follow.

Utilising either (Left or Right hand) Wall Follower method incorporating any of the previously detailed, Boolean, Sensor, Proportional and PID programming options, will accomplish the task within a reasonable time frame. But again, the same Line Follower order analysis of accuracy/speed and preference applies with Boolean being the least complex/simplest/slowest thru to the PID being most complex/accurate/fastest.

With Gear Down movement controlling speed, by further adding combinations of Touch, US, IR, Gyro and/or Light (Colour) sensors will assist in the Wall Following method, making the wall and object easier to detect (Touch/US/IR), accuracy of Robot Turns/adjustments (Gyro) improved and identification of Object colour (Light/Colour) determined.

4.5 Overall Summary

The set assignment tasks of; Line Following, Object/Colour Detection, Object Avoidance, Re-joining Line, 3D Maze Navigation and Sound Projection requirements, can be achieved by incorporating all the required sensors into the 1(one) Robot design.

The simplest programming options are to utilise Boolean (Line Follower) and Wall Follower (Object and Maze), including the various sensors methods;

- Light/Colour sensor to be adjustable (to 90 & 180 degrees) in order to read line (180) and object (90) via incorporating medium motor to move upward/downwards.
- Speed/Torque control via Gear Down methods from 2 large wheel motors.
- 2(two) Ultrasonic sensors, one fixed on right of Robot (Right-hand method) and the other pointing directly ahead (for Object & Wall Follower) to identify Objects and Maze.

- 1(one) Gyro sensor to measure rotational motion and more accurately determine degrees turned (for Object and Maze).

The author-designed 'Robb3e the Robot' is shown below as a possible design and is hoped to be trialled and videoed as an additional credit to the set assignment, in due course.

Figure 21 - Author Designed Robot (Robb3e)



Length (CSdown) = 22 cm Height = 12 cm

Length (CSup) = 22 cm Width = 17 cm

The author has chosen to build the Robot utilising Large Motor Gear Down method, for overall greater speed/torque control. For the Line follower aspect, the PID line follower is the most accurate, but for the reasons previously stated above, author will utilise Boolean (Zig-Zag) and Light/Colour sensors, in trialling. Similarly, for judgements made earlier, by ruling out the Touch and Line Following methods for the Object avoidance and Maze tasks, the author would choose combinations of Ultrasonic, Light/Colour and Gyro sensors to afford greater accuracies.

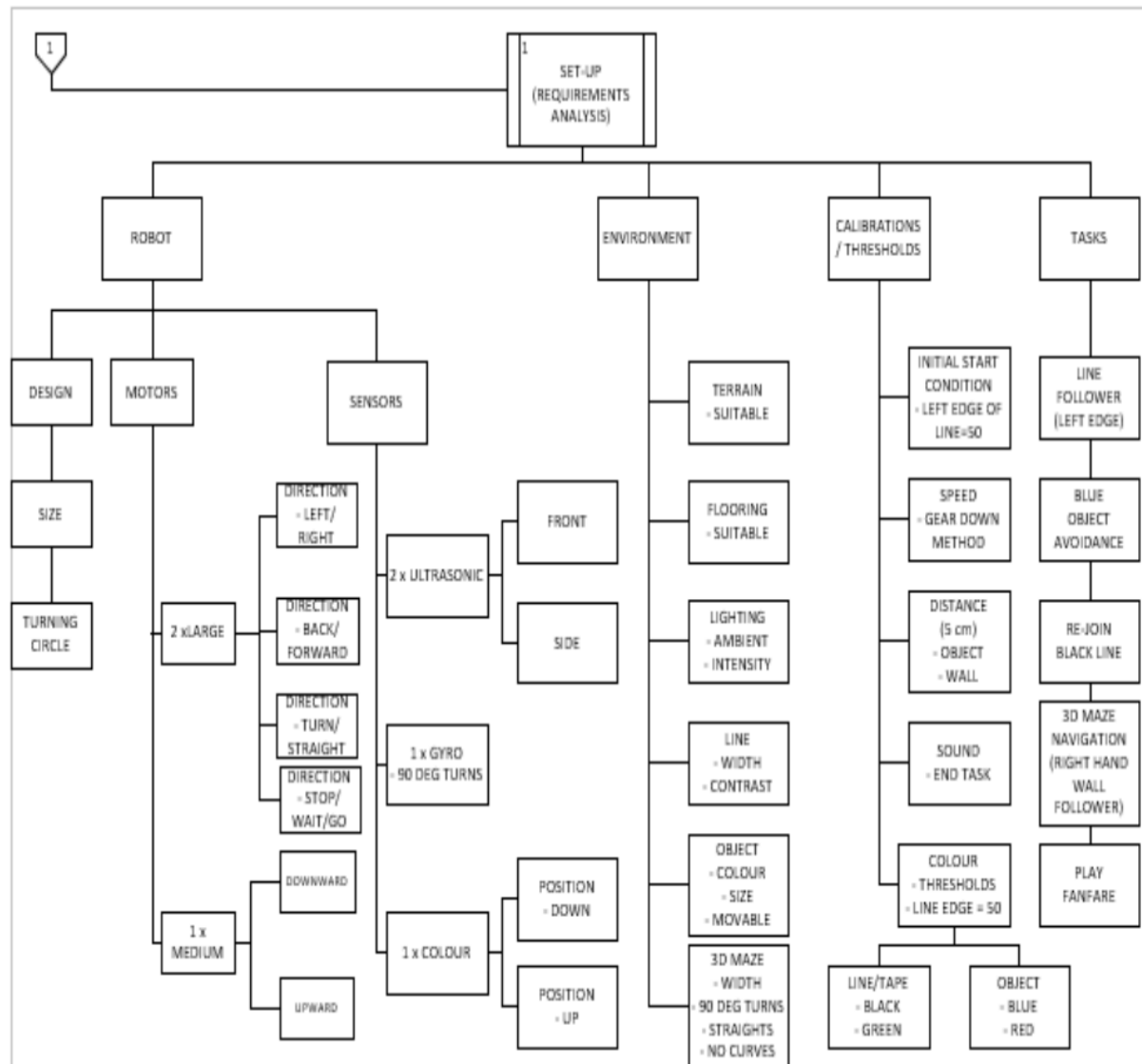
5 COMPONENT 2 – FLOWCHARTS / PSEUDO CODE

5.1.1 Requirements Analysis

To provide a computer-based solution, a list of requirements/features (Set-Up) have been established to enable further breakdown into greater detail. A hierarchical chart is shown below to detail how the solution has been broken down into smaller modules, utilising Decomposition and Modularisation techniques. These elements are not placed in any priority or sequence order, they are to aid in the further flowchart/programming elements that follow.

The (Set-Up) figure below shows the various overview elements of the Robotic task requirement.

Table 1 - Requirements Analysis (Set-Up)



5.1.2 MoSCoW

- Definition and Terminology

The **MoSCoW** method prioritises the Requirements Analysis listing to distinguish between wants and needs and categorises them as; Must (**M**) have features which are critical, Should (**S**) have such important features, Could (**C**) desirable/nice to have features and Won't have yet (**W**) because not currently appropriate needs. Based on these task requirements the following aspects have been determined as;

Table 2 - MoSCoW Haves

MUST Haves:	Heading
To encompass suitably sized & designed Lego Mindstorms robot	Robot
To include appropriate Motors & Sensors	Robot
Allow suitable directional movements (Left, Right, Stop, Go, Forward, Backward etc)	Robot
Cater for differing lighting conditions	Environment
Suitable Line and Maze entrance tape width & colour contrasts	Environment
Objects need to be recognised by Robot	Environment
Maze needs to be 3D physical, straight walls, 90 degree turns and dead-ends, no curves	Environment
Maze size needs to cater for Robot design to allow turns and identify walls/objects	Environment
To be a Lego Mindstorms computer-based solution	Calibrations
Initial Start Condition - Left Edge of Line	Calibrations
Determine Colour Thresholds for Line, Background, Objects, Wall	Calibrations
Determine Wall & Object finder distance	Calibrations
Robot to stop at end of final task	Calibrations
Incorporate Loops & Switch commands into programmes	Calibrations
Determine Robot Turning circle	Calibrations
Determine Colour Line readings for Large Motor movements (Stop, Go Left, Straight,	Calibrations
Determine US Object distance readings for Large Motor movements (Stop, Go Left,	Calibrations
Determine US Object distance readings for Medium Motor Colour Sensor movements (Up,	Calibrations
Complete tasks of Line Following, Blue Object Avoidance, Re-Join Line	Task
Complete tasks of Find Green Maze Entrance, Navigate Maze	Task
Complete tasks of Find Red Object, Play Fanfare and End Task	Task
Capable of following a line which has a circular section	Task
Blue Object encountered needs to be Avoided, Grabbed, Lifted, Climbed, Discarded	Task
Blue & Red Objects need to be identifiable	Task
Play sound at final task completion	Task

SHOULD Haves:	Heading
Line needs to be suitable size width and colour contrasted with background	Environment
Blue & Red Object need to be rectangular shape and movable	Environment
Cater for differing terrains & floor conditions	Environment
Robot speed & course time restrictions	Calibrations
Robot Large Motors designed with Gear Down	Calibrations
Select Wall Follower Algorithm (Right-Hand) method	Calibrations

COULD Haves:	Heading
Incorporate additional actions (Spinning etc) at task points	Calibrations
To prohibit user intervention once task commences	Task

WON'T Haves:	Heading
Allow different design/programming methodologies to solve tasks set	Robot
Maze to have different floor levels	Environment
Allow multiple methods of task completion	Calibrations
Play sounds throughout each task objective	Calibrations
Allow EV3 screen to display actions	Calibrations
Repeat entire task goin back from Finish to Begin	Task
Remember the quickest route thru Maze	Task

5.2 Flowcharts

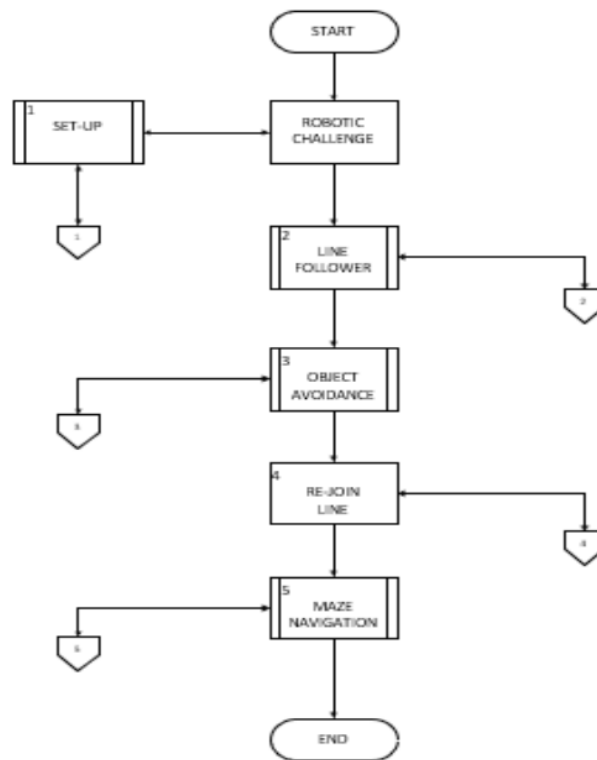
Flowcharts visually show the steps to take, utilising connected symbols to show the sequence of events, it is a diagram that represents a set of instructions, which normally use standard symbols to represent the different types of instructions. These symbols are used to construct the flowchart and show the step-by-step solution to the problem [15].

Table 3 - Typical Visio Flowchart Symbols

	Process		Decision
	Subprocess		Start/End
	Document		Data
	Database		External Data
	Custom 1		Custom 2
	Custom 3		Custom 4
	On-page reference		Off-page reference

The separately attached Visio flowchart files provide an algorithm for the proposed EV3 Robotic challenge solution, an overview of the main sub-processes is extracted below.

Table 4 - Flowchart Overview (Visio extract)



5.3 Pseudo code

Pseudocode is an informal high-level description of the operating principle of a computer program or other algorithm. It uses the structural conventions of a normal programming language, but is intended for human reading rather than machine reading [16]. Written in plain English using shortened statements, pseudo code is the informal language used to plan out the logic (algorithm) for a problem. It enables the problem to be broken down into more manageable steps, making it easier for a programmer to understand and to convert into a computer solution. It typically covers statements regarding; prompting for input, setting variable values, displaying output, along with simple calculations and processes.

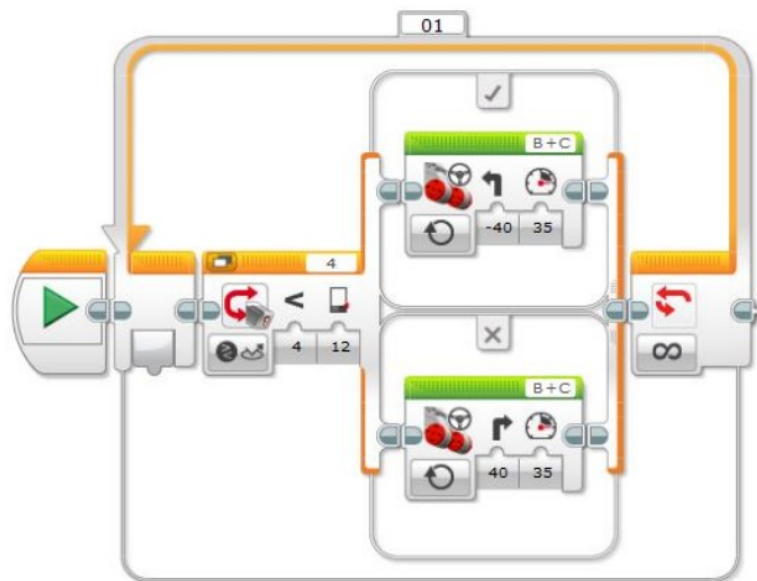
5.3.1 Line Follower

With Colour Reflected Light Intensity threshold readings set to compare mode and being determined as; White >50, Line Edge = 50 and Black <50 along with initial condition of Robot being positioned on left hand side of line edge, then;

```
begin loop
  wait for start button to be pressed
  if Colour sensor reads greater than 50 then
    use Move Steering Block to Go Forward and TURN Right
  else if Colour sensor reads less than 50 then
    use Move Steering Block to Go Forward and TURN Left
  else
    use Move Steering Block to Go Forward and Straight
  end if
loop unlimited
```

- Extracts from Author's own Lego (Robb3e the Robot) Mindstorms code:

Table 5 - Line Follower Mindstorms code extract



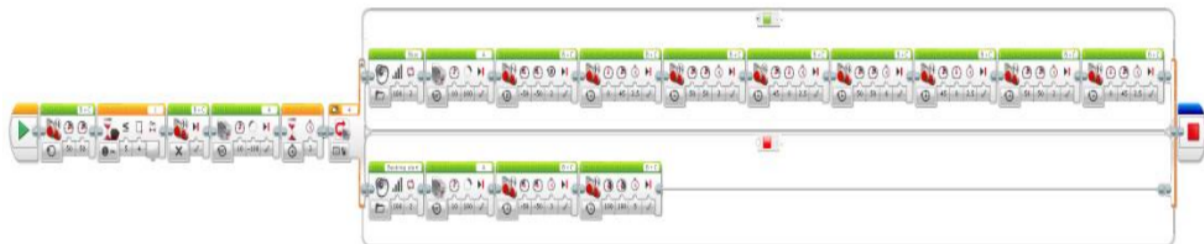
5.3.2 Object Avoidance

With Front Ultrasonic object identification distance threshold set at 5cm and Colour Reflected Light Intensity threshold readings set to determine Blue and Red, then;

```
wait for start button to be pressed
if Front Ultrasonic sensor detects Object at 5 cm away then
  use Move Steering Block to STOP motors
  use Medium Motor Block to reposition Colour sensor Horizontal to floor
  wait for Colour sensor to detect colour of Object
  use Medium Motor to return Colour sensor to Vertical position
  begin loop
    if Colour sensor reads BLUE then
      use Move Steering Block to TURN Left 90 degrees
      use Move Steering Block to Go Forward 3 rotations
      use Move Steering Block to STOP
      use Move Steering Block to TURN Right 90 degrees
      use Move Steering Block to Go Forward 3 rotations
      use Move Steering Block to STOP
      use Move Steering Block to TURN Right 90 degrees
      use Move Steering Block to Go Forward 3 rotations
      use Move Steering Block to STOP
      use Move Steering Block to TURN Left 90 degrees
    else if Colour sensor reads RED then
      use Sound Block to play Fanfare
    else
      use Move Steering Block to Go Forward and Straight
    end if
    Colour sensor detects Black line
  loop unlimited
```

- Extracts from Author's own Lego (Robb3e the Robot) Mindstorms code:

Table 6 - Object Avoidance Mindstorms code extract



5.3.3 Maze Navigation

With Both Front and Side Ultrasonic sensors (US) object identification distance threshold set at 5cm and Colour sensor set to Horizontal position and with Reflected Light Intensity threshold readings set to determine Blue and Red, then the Wall Follower (Right-hand) method is deployed, with initial Robot position placed facing inward to maze at entrance point;

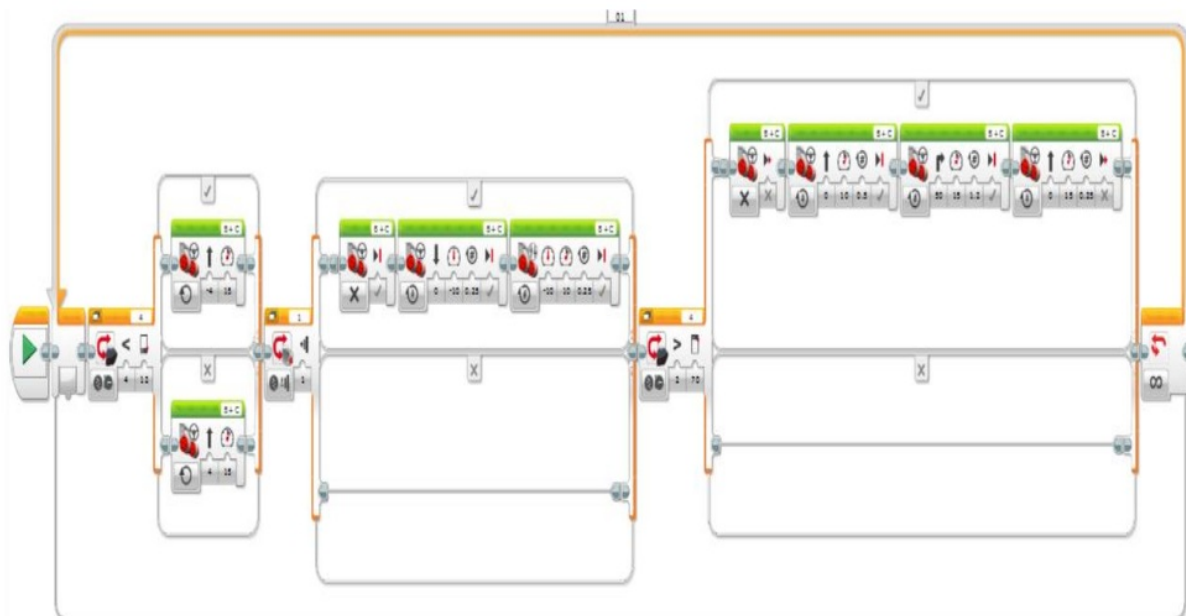
```

wait for start button to be pressed
begin loop
  if Side US sensor does NOT AND Front US sensor does NOT detect wall then
    use Move Steering Block to Go Forward and TURN Right
  else if Side US sensor DOES detect wall AND Front US does NOT detects wall then
    use Move Steering Block to Go Forward and Straight
  else if Side US sensor DOES detect wall AND Front US DOES detect wall then
    use Move Steering Block to STOP
  begin loop
    if Colour sensor reads RED then
      use Sound Block to plat Fanfare
    else
      end if
  loop unlimited
  use Move Steering Block to TURN Left 90 degrees
  use Move Steering Block to Go Forward and Straight
  else Side US sensor does NOT detect wall AND Front US sensor DOES detect wall then
    use Move Steering Block to STOP
  begin loop
    if Colour sensor reads RED then
      use Sound Block to plat Fanfare
    else
      end if
  loop unlimited
  use Move Steering Block to TURN Right 90 degrees
  use Move Steering Block to Go Forward and Straight
end if
  Colour sensor detects RED
loop unlimited

```

- Typical Lego Mindstorms Wall Follower code [17]:

Table 7 - Typical Wall Follower Mindstorms code



6 REFERENCES

- [1] Lego Mindstorms Education (2020) *Make It Move Up An Incline - Gear Down*. Available at: <https://education.lego.com/en-us/lessons/ev3-dep/make-it-move-up-an-incline#Planitem0> (Accessed: 29 April 2020)
- [2] Torok, Rob. (2019) *Line Following and Proportional Controls*. Available at: <http://www.legoengineering.com/line-following-and-proportional-controls/> (Accessed: 29 April 2020)
- [3] Griffin, T. (2014) *The Art of Lego Mindstorms EV3 Programming*. p.153
San Francisco: No Starch Press Inc. ISBN 978-1-59327-568-6.
- [4] Prassath, H. (2017) *Arduino UNO driven PID Line Follower*. Available at: <https://www.engineersgarage.com/contributions/arduino-uno-driven-pid-line-follower/> (Accessed: 29 April 2020)
- [5] Builderdude35 (2016) *Comprehensive Comparison of EV3 Line Followers*. Available at: <https://www.youtube.com/watch?v=P5OCE0xwhvo> (Accessed: 29 April 2020)
- [6] Dimitrios, V., Panagiotis, V., Eleni, K., Themistoklis, P. (2017) *The Line Follower Robot 'Samianator'*. Available at: http://robotickyden.cz/2017/robots/img/2017r_0131_L.pdf (Accessed: 29 April 2020)
- [7] Turnbull, G. (2020) *Lego Mindstorms EV3 Programming 101: Object Detection*. Available at: <https://www.funcodeforkids.com/lego-mindstorms-101-object-detection/#Vs> (Accessed: 29 April 2020)
- [8] McGough, J. (2018) *Introduction to Mobile Robotics. 4 -Navigation/Mazes*. Available at: <http://roboscience.org/book/html/Navigation/Mazes.html> (Accessed: 01 May 2020)
- [9] Klein R., Kamphans T. (2011) *Pledge's Algorithm - How to Escape from a Dark Maze*. Available at: https://link.springer.com/chapter/10.1007%2F978-3-642-15328-0_8 (Accessed: 29 April 2020)
- [10] Snapp, R. (2014) *Puzzles, Games & Algorithms*. Available at: http://www.cems.uvm.edu/~rnsnapp/teaching/cs32/notes/tremaux_rules.pdf (Accessed: 29 April 2020)
- [11] cs4all.org (2020) *Maze Activity Using Maze Solving Algorithms*. Available at: <http://www.cs4all.org/files/Maze%20Activity.pdf> (Accessed: 29 April 2020)
- [12] Abdelnahi, O. (2020) *Shortest Path Algorithms*. Available at: <https://www.hackerearth.com/practice/algorithms/graphs/shortest-path-algorithms/tutorial/> (Accessed: 29 April 2020)
- [13] CareerRide.com (2016) *Data structure – What is an recursive algorithm?* Available at: <https://www.careerride.com/Data-structure-recursive-algorithm.aspx> (Accessed: 29 April 2020)
- [14] Bose, A. (2019) *Maze Routing – Lee's Algorithm*. Available at: https://amitrajitbose.github.io/blog/maze_routing_lee_algorithm/ (Accessed: 29 April 2020)
- [15] BBC Bitesize (2020) *Flowcharts*. Available at: <https://www.bbc.co.uk/bitesize/guides/z3bq7ty/revision/3> (Accessed: 28 March 2020)
- [16] WhatIs.com (2005) *pseudocode*. Available at: <https://whatIs.techtarget.com/definition/pseudocode> (Accessed: 01 May 2020)
- [17] Medium.com (2019) *Lego Maze Solving Tribot v1.0*. Available at: <https://medium.com/kidstronics/maze-solving-tribot-v1-0-a29963f8742a> (Accessed: 29 April 2020)

7 REPORT PLANNING METHODOLOGY

This Integrating Project report assignment was completed by adhering to the following the activities;

- Read and understand the report brief and Task Summary.
- Make use of provided Moodle lecture notes, presentations and video recordings.
- Make plan (Gantt chart) including time management – planning/research most time consuming, cater for other exams.
- Create report format and style – use Moodle template example.
- Follow structure
 - Abstract/Acknowledgements
 - Introduction
 - Main Body
 - Conclusion (if required)
 - References
- Define the problem and purpose of the report – decide assignment title, aims & objectives.
- Define audience – Lecturer/Tutor (Sue Maw/Gaynor Thomas).
- Brainstorm (Mindmaps) and Determine ideas to include.
- Research and gather information (Task Worksheet, library, internet etc).
- Collect, sort, evaluate organise material.
- Complete references (as you go) during all researching phases
- Analysing my research/material.
- Prepare outline drafts.
- Design, build and programme own prototype Robot to test theories researched (Robb3e the Robot)
- Write the report, check spelling and grammar.
- Peer Review – (Terri & Sue Maw)
- Incorporate peer comments into final redraft.
- Submit final report via Turnitin.
- Separately submit Visio flowcharts via Moodle

FINAL GRADE

90/100

GENERAL COMMENTS

Instructor

Firstly I really wish I'd assigned some marks for a cool robot on the front sheet. Robbie is indeed an inspiration to us all.

Project Introduction 10% 9

The Introduction is well structured and shows a systematic approach to planning the solution. The detailed analysis of Assumptions was a great feature.

Line Follower 25% 23

A comprehensive analysis of the problem with drawings and code modules to support your designs.

Block avoidance 25% 22

Another very comprehensive analysis with many examples provided. Section 4.3: in Academic English try to avoid "you must". "It must be possible to detect" is a possible rewording in this style. It can seem awkward to phrase at first but check out the phrasing in textbooks and journals as well as the content.

Maze 25% 24

Excellent research into maze solving algorithms demonstrated. It's a tough job to read widely and to write a short explanation putting together your research but you nailed it here! Another small point on academic English - if you (correctly) use "the author" every time you would like to say "I" then it can get repetitive. Instead, mix it up with the passive voice - "the algorithm was designed".

Overall summary 15% 12

A good outline of what has been achieved. A tip here is to check the marking scheme and be sure to include content in proportion to the marks as you could have included a little more on what you would have done differently and perhaps referred back to some of the theory covered earlier. What is included here is very

good!

Assessment Component 2 – Coursework - 40% 90

You've covered this topic in triplicate with flowcharts, pseudocode and a video of your robot in action. I can't argue with the video evidence to show me that the algorithm works.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27
