

M4X01506

by Nicholas Evans

FILE	4057_NICHOLAS_EVANS_M4X01506_93631_1074655259.DOCX		
TIME SUBMITTED	26-APR-2018 10:26AM (UTC+0100)	WORD COUNT	2252
SUBMISSION ID	87512800	CHARACTER COUNT	11630

2018

Integrating Project

Nicholas Evans

M4X01506 - 1704662

5/4/2018

Table of Contents

1.0	Introduction	2
2.0	Building the Robot	2
3.0	Choosing the Sensor	3
4.0	Section One – Line Following	3
5.0	Section Two – Obstacle Avoidance	5
6.0	Section Three – Maze Navigation	5
7.0	Conclusion.....	7
8.0	References	8

1.0 Introduction

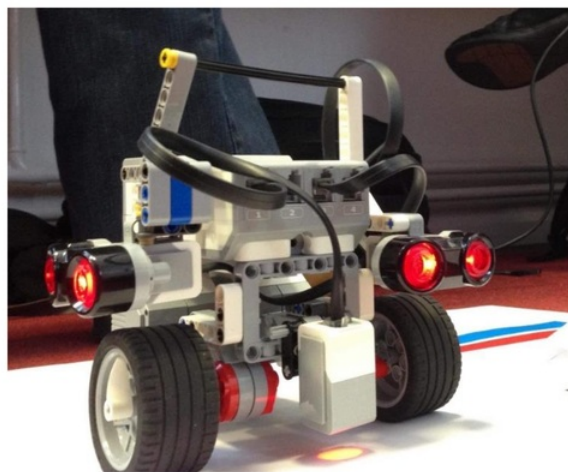
As a first-time user of Lego Mindstorms, it's not hard to become very impressed by the simplicity of how the programming page was laid out for you. Every command was within its own section and each sub section is colour coded, so locating exactly what you needed was a simple task. The sheer volume of support and enthusiasts online working with the software just proves how easy it is for anyone to pick up. Families and friends alike working together on a project than they can be proud of. Whether it is to create a fighting robot that that will put the fear into anyone else's Lego robot, or something less violent, like creating a bot that can colour sort a box full of crayons or even creating a 'bionic' arm that can move and grip just like a real one!

So, you can understand the excitement after being told to create a robot in a group that can complete a course that consists of 3 challenges. Line following, Obstacle avoidance and maze navigation. Each person overseeing the programming of each section.

2.0 Building the Robot

Considering how different each individual styles and ideas were within the group, deciding on using the basic design that was presented when the group were first introduced to Lego Mindstorms was something the group were all on board with. All that was needed was something that can keep its balance with whatever was decided to attach to it, considering the lack of knowledge on how and where the course would be set up like, the group felt that it would be the most practical approach.

Tests were conducted to review a few modifications to see if there was a way to improve the functionality of the robot. Instead of using the ball bearing as the rear wheel, a member of the group designed a swivel tyre in its place. Thinking it would allow the robot to move more freely, plus it looked great! Unfortunately, this was not the case. Due to the Lego pieces available, the team was very limited on what could be done to improve upon the idea as the piece had restricted the turning too much. However, attaching a 'handle' along the top of the robot so it could be safely carried to where it was needed without fear of any pieces breaking off. (learned the hard way).



3.0 Choosing the Sensor

This was a very vital decision to make as this was the most essential part that will get the robot around the course. A book by Terry Griffin [1] which was recommended, wrote how the robot was designed with all the attachments in place, had the ability to operate using up to five senses. The most popular solution for navigation though was by either touch using a pressure sensor or sight using a colour or infrared sensor.

Finding the pressure sensor can be very temperamental, due to the pieces available, the team were only able to create a flimsy attachment to hold it in place, which did not help as its main function was to collide with any obstacle in front to trigger the sensor, which usually caused it to either shift in place causing it's placement ineffective, or it would break off entirely. Not only that, it was found that it also added more coding to the loop as it would need to reverse away from that obstacle, just so it can turn to avoid it.

Unfortunately, being unable to get hold of an infrared sensor, it was decided that using an ultrasound sensor to allow the robot to 'hear' any obstacle and to follow it around would be best, preventing the need to bump in to anything in the first place. This turned out to work incredibly well. This enabled the team to adjust the distance on how far to register any obstacle to help avoid, and to even use it to keep within a certain distance away from a wall to follow. thanks to yet again to the simplicity of the programming itself.

For the line following, the only sensor that can really be used was the colour sensor. Thanks to its ability to differentiate not only colours, but also light reflectance from any surface it can see. Choosing a function was very difficult as the part could be very temperamental, it would unfortunately not see certain colours on times and this was incredibly frustrating.

4.0 Section One – Line Following

The first section of the course was to be programmed primarily by myself. Due to the nature of the section, the line being either red or blue, you would think that this section using the colour sensor would be rather simple. Thanks to the program, they allowed the coding of the sensor to be very easy. I could set it to read a specific colour or multiple if commanded.

Frustratingly though, I found the robot would just ignore any line and wander off, or just proceed to go in a straight line. I decided to test to see how reliable the sensor was by coding it to carry on moving in a straight line until it saw either a red or blue coloured strip I placed on the floor, for it to then stop. After conducting this test 10 times, it recognised the colour blue 5 times and the red only twice with the other attempts not picking anything up at all. Due to the inconsistency of this, I decided to test out it's other function, light intensity. This enables the sensor to read how much light is being reflected off the surface.

The colour code allowed myself to see exactly how much intense the reflection was on each surface thanks to a number scale. It showed me using the number scale how much of a difference each colour would reflect, which then enabled me to code the sensor to only follow a specific line by how much light it saw by inputting the number into the colour code. How this worked was, it would follow the specific light intensity I coded by going back on itself if it saw the light reflectiveness increase thanks

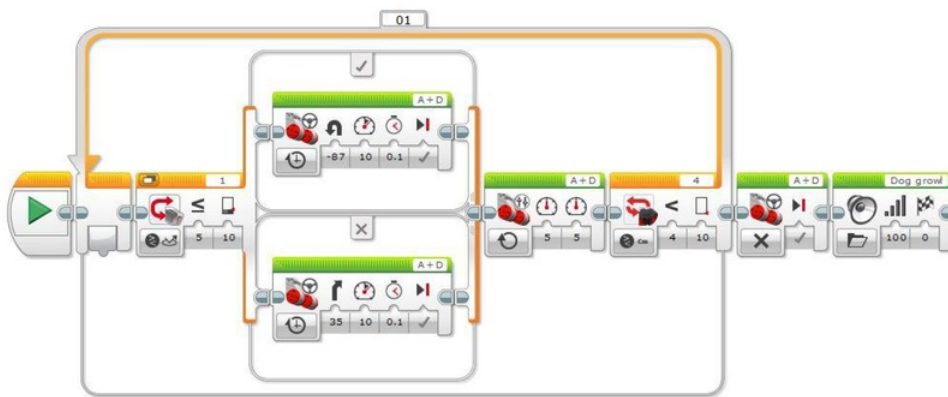
to the side of the colour being white. This worked wonders! Only problem I had left was how to get the robot to take the turning better, as it seemed to be unable to turn quick enough to follow the line properly, resulting it wondering off again looking for the line.

To solve this was a simple task but time consuming, as the only method in fixing this issue was to not only change the speed of the robot to allow for more time to turn, but also needed to investigate the speed in each individual wheel. I needed to do this so if the line turned left, I would want the left wheel to either slow down enough allowing the right wheel to carry the momentum to turn the robot in time, or to have that wheel stop completely.

There was one thing I didn't consider. Even though in tests the robot followed the line perfectly, due to using the light intensity setting on the sensor, the lighting in each room, thanks to the sun being out or even the placement above lights, effected the light reading. Causing the robot to just wander off. Thankfully, due to the simplicity of the coding design, it was a simple matter to adjust the numbers accordingly with a little bit of testing to make sure.

There are a few negative points I experienced that I've failed to mention. I whole heartedly believe that the coding software used for this project is very simple and user friendly, however, I do find the coding to be far too unreliable in practice. Having tested the coding of the robot for weeks, I find that the coding, even though it would work brilliantly on a number of trials consecutively, it would still randomly not work and appear to do its own thing far too often without reason. The sensors themselves are also just as unreliable. Even swapping out the thought 'defective' sensors, the same problems seem to occur. It can become incredibly infuriating as there would be no reason to affect the results harshly as it does. Looking on forums online though, I find that all these issues are very much commonplace whilst using the software and the Lego Mindstorms parts themselves.

Here is the resulting code used:



5.0 Section Two – Obstacle Avoidance

This section was mainly focused on by Anthony Smith. His job was to code the robot to have the robot stop from colliding with a wooden block which was put down on the line blocking his path and navigate itself around that said obstacle until it relocated the line, for it to then continue along the course.

As mentioned in the sensors section, the group tested the touch sensor a handful of times, but it was very transparent that it would be to be unreliable. The Lego piece attached would either break off, or the sensor would not register the block and proceeds to push it out of the way! So, decided on the use of ultrasonic.

The way this worked in coding is very similar to the colour sensor whilst using light intensity. The sensor would transmit a soundwave, which it would then receive the waves back. Using the length of time each wave to comes back to determine the distance of that object.

This worked wonderfully! The robot would stop at a distance just far enough to allow the robot to turn on its axis without bumping into anything, allowing the robot to navigate (at speed) around the obstacle!

Here is the resulting code used:



6.0 Section Three – Maze Navigation

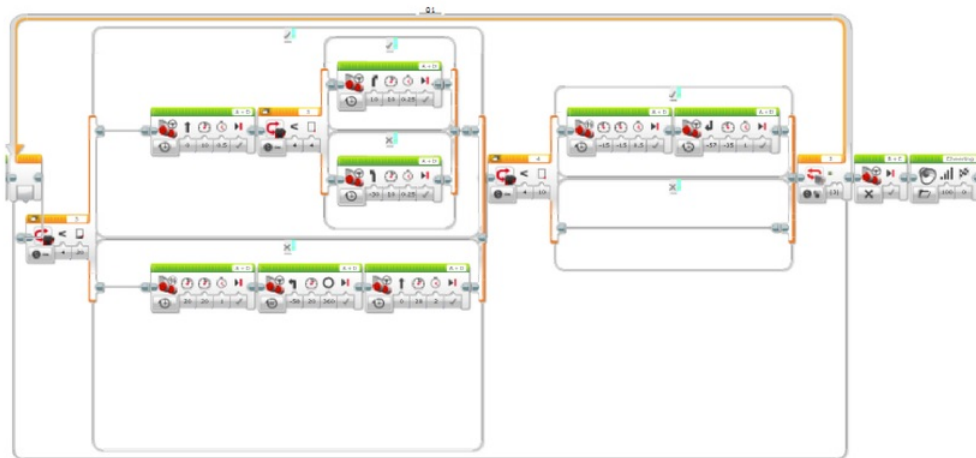
This final section was mostly tackled by Rebecca Ryall. It was suspected that this section was going to be the toughest complete. It would require everything that has been coded for the previous sections, but to work simultaneously! Thanks to all the experimenting beforehand for the previous parts, knowing that using the ultrasound sensor worked great, attached one to the side of the robot.

Unlike the sensor at the front, which would move away from an obstacle, its purpose was to find the wall, and using its soundwaves, placing itself at a consistent distance from the wall and follow it.

So, in effect, have the front ultrasonic sensor looking for a wall blocking the path. Once it comes across such a wall, it will reverse slightly then turn to the right by 90 degrees, allowing the right sensor to read a follow the wall until such a wall blocked the robot again.

This was how it worked right up until the end where the robot needed to sense the colour green in the middle of it all to indicate the end of the maze and the completion of the course. Thanks to the unreliability of the sensor though, this was always a hit or miss. But thankfully, it worked more often than not.

Here is the resulting code used:



7.0 Conclusion

This project really has given me a lot to think about. You would think that with it being mainly considered an expensive children's toy franchise, one would think that given what we were asked to accomplish using the very simple programming software they provide for their Lego Mindstorms kits, the task would be relatively simple. And on a certain level, it was. As I have said throughout the report, the coding side is very simple and also relatively fun to use thanks to its ease of access and layout. Everything on screen is very basic and easily understood by a glance. Each code is colour coded unto colour sub groups that makes sense. The click and drag system just shows that simplicity was at the heart of the designers of this wonderful bit of software.

The Lego pieces themselves were exactly what you would expect them to be, strong, sturdy and with enough imagination, can be used to build incredible looking robots. The motors themselves and all of the available sensors are very clever and allows any user to be able to build whatever they can imagine at a grander, more engineered scale!

With all that being said, it still has a lot of problems that unforgivably makes the whole experience infuriatingly stressful if used for anything other than as a hobby or purely for fun. The programming side itself sometimes require what I can only explain as a 'break' in the code if you want it to do more than just one thing in your coding. Take the obstacle avoidance for example. Instead of the robot being able to turn to avoid the block once it sees it, we had to have the robot stop before it could proceed to following the rest of the line of code.

But all this being said, it was a very enjoyable experience, and as a group, it was very interesting seeing just how different we took our ideas on coding the robot and putting them into practice.

Here is our coding in full:



8.0 References

[1] T. Griffin, in *The Art of Lego Mindstorms EV3 Programming, 1st Ed. No Starch Press*, 2014.

FINAL GRADE

70/100

GENERAL COMMENTS

Instructor

Assessment Component 1 (Report):

Introduction: Quite well stated, but would be improved by including a diagram - it's quite a visual assignment, so it would help the reader

4/5

Background Research: You haven't really included a section on background research, you've gone straight into the design. You do hint at the knowledge gained throughout but without directly referencing it. For future project-based assignments, it's worth keeping a note of anything you've done in this regard as it's very easy to forget it later! (which I think has happened here)

7/15

Project Plan: There doesn't seem to be one? From the assignment brief "You need to include a plan of how you will allocate your time in completion of the project." ie tasks against time - any tasks going on at the same time, time for writing up etc

Design Process: Quite well explained, with a screenshot of the final robot, might have been nice to see previous incarnations - again, this is part of the design process, don't forget to document throughout. You allude to the code and how you programmed it, show a screenshot of that section to illustrate your words.

31/40

Evaluation and results: I think you could have been a bit more positive in your evaluation, as a good implementation was achieved and many problems were overcome! 22/30

Assessment 1 Total: 64%

Assessment Component 2 (Demo):

Technical knowledge/skill 17/20

Analytical and Problem-solving skills 17/20

Quality of result 46/60

Good overall, transitions and end point negotiated, some minor deficiencies in overall negotiation of obstacles. A good implementation overall.

Assessment 2 Total: 80%

Just a few parts missing from the documentation, a good project overall, well done.

OVERALL MARK FOR MODULE: 70%

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9
