Real Robots 2-Eyed Phototaxis

Contents

[Introduction 1](#_Toc8124616)

[Method 1](#_Toc8124617)

[Results 2](#_Toc8124618)

[Discussion 2](#_Toc8124619)

[References 3](#_Toc8124620)

# Introduction

The objective of this investigation will be to develop a robot that is capable of moving towards light with only 2 sensors and 2 wheels. To achieve this, I will be using a lego mindstorm EV3 and programming it by hand in RobotC. My main focus in this investigation is to look at the pitfalls of hand coding a robot to solve a simple task. I will be comparing the troubles that have arised in a previous experiment done on NXT robots in order to determine which problems are frequently occurring across experiments when programming real robots and how best to minimise their effects. (Poikselka, Vallivaara, & Roning, 2015). While we spent time optimising the design of the bot, this was not a main focus of this investigation so future investigations have room for improvement through a more thorough optimisation process of the robot. The robot followed a simple design and the main focus of optimising the robot itself was on sensor placement and angle

# Method

In building my robot, I have opted for simplicity as the primary focus. A simple build can be beneficial in reducing potential part failure as well as identifying problems that occur with the build (Poikselka, Vallivaara, & Roning, 2015).

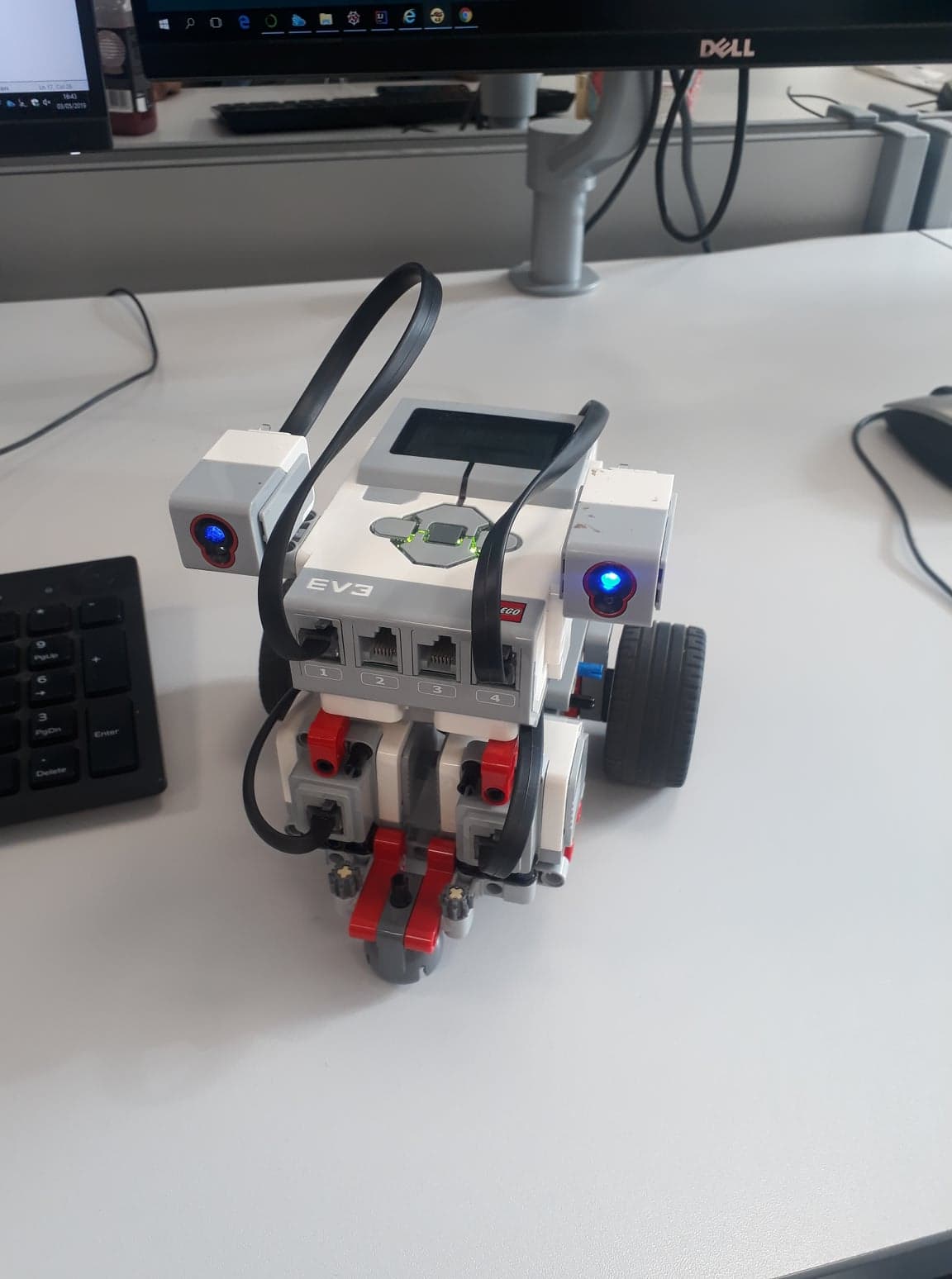


Figure 1

The final build used for the investigation

Figure 1 shows the build used to implement the code for the robot. Each sensor was placed on the sides with positioning determined through initial preliminary tests. While position may not be optimal, these preliminary tests ensure the bot is able to function correctly with the current placement. Further tests showed that sensor placement and angle had a noticeable effect on the performance of the robot.

After testing multiple solutions of hand coding reactions for the agent, the final implementation was also the simplest. Below is a pseudo code example of the implementation.

In the final implementation we simply recreated an ‘aggressor’ Braitenberg vehicle in order to achieve the robots task of following the light.

The success of the implementation was tested in 2 environments: a strong light source in which it was expected to seek out and move towards, a single light source that moved where the bot was expected to follow the light source.

While determining the performance of the robot, at the start of each test it was placed in a different starting position in order to ensure its responses could be repeated.

# Results

During most of the early versions of our final solution, we consistently got unexpected results. Only when we started using the simplest implementation possible did we realise that one of our 2 sensors was constantly off by 5 units of measurement. This was tested and confirmed. This issue of imperfect sensors mirrors technical challenges identified in previous experiments on NXT systems and represents one of the potential pitfalls of using this system for experiments (Poikselka, Vallivaara, & Roning, 2015).

Problem occurred when measuring the success of each implementation. Varying light levels and lack of a non-noisy controlled environment posed an issue in terms of quantifying success of each build. Success was hard to measure due to different responses in seemingly the same scenarios. We attempted to counter this by determining the success of each implementation visually as opposed to using a mathematical fitness function. We spent large amounts of time determining success of builds in order to ensure that the final build was our most capable.

Using the final implementation, the robot was able to continuously seek and follow a light source.

# Discussion

While we were able to transfer the same style of robot from simulation into reality, the biggest problem we had was the variance in the readings of the light sensors. We were able to compensate for the variance in the end by hand. The simulation we used previously just did not account for this, while it is possible to have a simulation account for high amounts of noise, when hand crafting an agent, a pitfall is trying to determine these variances and account for them. It is worth noting that once we had accounted for the variance then the same principles we found in simulation did transfer. This agent and its task did have a low complexity, in a more complex environment it would be much harder to determine and account for the variance experienced in the real world by hand.

We found through our earlier attempts that the higher the complexity of our agent, the more room for error in its logic. The most successful implementation was by far the simplest. I believe this is an important lesson we learned from this investigation.

Hand coding a responsive agent is possible, however, pitfalls do come with it, such as over engineering, not accounting for noise/variance and time consuming testing processes. Bridging the gap between simulation and reality is an important step in streamlining the process and helping remove the pitfalls that come with humans hand engineering responsive agents.

# References

Lew, N. W., Horton, T. B., & Sherriff, M. S. (2010). Using LEGO MINDSTORMS NXT and LEJOS in an Advanced Software Engineering Courses. *IEEE Conference on Software Engineering Education and Training.* Pittsburgh: IEEE.

Poikselka, K., Vallivaara, I., & Roning, J. (2015). Evolutionary Robotics on Lego NXT Platform. *International Conference on Tools with Artificial Intelligence (ICTAI).* Vietri sul Mare: IEEE.