**FPGA CONTROLLED MAZE SOLVING ROBOTIC ARM**

Thomas Edmonds

May 2024

**Abstract**

This project examined the idea of using an FPGA as the main controller for a robotic arm. The idea behind this is to prototype a self-contained arm controller that can quickly respond to its environment with minimal outside control or input.

**Acknowledgements**

**Contents**

**Glossary**

FPGA: Field Programmable Gate Array

**1 Introduction**

Robotic arms are often used in industry to complete repetitive tasks such as assembling components or packing items into boxes, but what if the box is off centre or the component is upside down? With typical dumb robotic arms, the component would be welded upside down and the item would be put next to the box or in the wrong place.

This project hopes to provide a solution to these problems by creating a prototype robotic arm controller that can react and adapt to its surroundings on the fly with little to no external control. As a proof of concept, the arm should be able to use the colour data from a photo to map a maze, Q-Learning to solve it and inverse kinematics to move through it.

An FPGA is used for three main reasons. Firstly, they are significantly faster than microcontrollers particularly for complex mathematics due to their parallel processing capabilities. Secondly, they are far more flexible, where additional peripherals or microcontrollers may have been required to solve new problems or meet requirements as they appeared, an FPGA can do pretty much anything on its own. Lastly, while this project could have been completed easier and faster using a microcontroller the results wouldn’t have been as good/useful as the controller would have been slower to adapt/respond to its environment.

**2 Project management & Planning**

ASK YOURSELF QUESTIONS

Breakdown of each task,

Rough overview of each meeting

Risk assessment

Gannt chart?

Maybe mention practice fixed point module writing?

Why I chose the arm (mention problems later)

**3 Initial arm build**

Setting up the arm, Issues with noise/feedback overloading Arduino

Isolator circuit design/build/testing

Why I chose magnetic instead of optical isolators

**3.2 Late arm build**

Building base board

Issues with servos

Issues with arm (faulty servo’s etc)

Issues with pen (too firm/not flexible enough)

Replacing arm

**4 Early FPGA work**

Finding replacement power supply/cable?

Basic testing

PWM and PWM testing

**5 Machine learning attempts**

Don’t know how much to put here, probably just a basic outline of how it would have worked with limited details.

**6 Q-Learning**

Final stretch, discus how the Q-Learning did and didn’t work.

Discus potential issues and solutions. Duplicated below?

Using parallel capabilities maybe could have calculated the new-Q for all 4 actions at once

**7 Overall testing**

Issues with arm (faulty servo’s etc)

Issues with pen (too firm/not flexible enough)

Replacing arm

Manual control of servo angles modules

Failure to move arm using Q-learning, solutions and fixes. Duplicate?

**8 Conclusion**

Summarise everything could maybe be merged with overall testing depending on how much stuff fits/doesn’t fit.

**Recommendations**

With more time would have been doable too much work for one person in the time frame

Lack of peer to review of work due to lack of FPGA experienced technicians/peers in the lab.

With hindsight would have reduced scope of project, replaced arm with remote controlled buggy/car and used a bigger floor-based maze. Removes kinematics, use of MATLAB and worrying about how to communicate kinematics to the FPGA, reduced number of tasks/required skills makes project significantly more manageable.