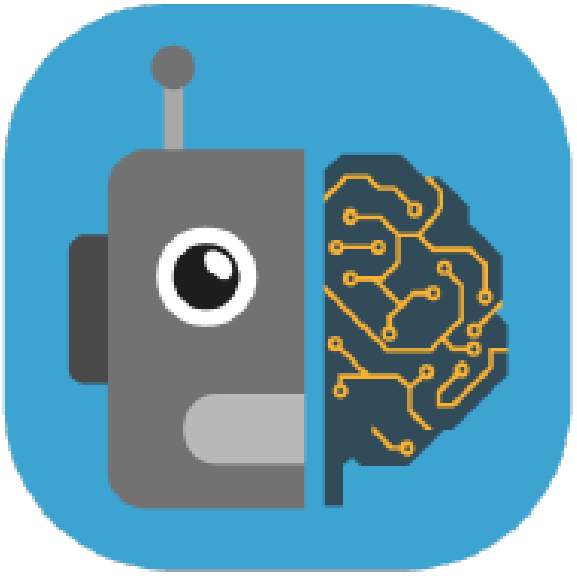

Fontys University of Applied Sciences

Affordable Manipulator With FPGA-driven FOC

Final progress report

Limpens, Jordie J.J.N.
Marinov, Mihail M.M.
Mindiola Romero, Ray R.D.
Sminia, Sean, S.M.
Strik, Leonardo L.S.

7-1-2020

Preface

This document is the final report of a project about the design of an affordable 6 degree of freedom manipulator arm, which was part of the Adaptive Robotics minor at Fontys University of Applied Sciences. The goal of the project was to design an affordable robot arm, that could be used for educational purposes. It was motivated by the fact that currently available robot manipulators are expensive and although Fontys has a few, close to project deadlines the demand for them is too big. An affordable, in-house produced manipulator arm would not only help meet the student's demand, but, since students will have access to the design documents and source code, will also allow them to improve it and modify it while studying robotics.

The project was worked on in the period from March 2020 until June 2020. Unfortunately, the project coincided with a global pandemic, which caused all work on the project to be remote, thus nothing could be physically made nor tested. This limitation has shifted the initially planned goals and end product of the project. Despite the difficult period, the group has put effort in order to finish as much as possible in order to provide a solid groundwork for the completion of the initially intended product.

The group members would like to express significant appreciation to Peter Klijn, who was our tutor for this project and helped us greatly. We also are thankful to Pablo Negrete, who created the project, gave us tips and acted as a client for the project's purposes. We are thankful for the advice given by Esmaeil Najafi, which was also very helpful for our project. Lastly, we also wish to acknowledge the help provided by Edwin Insuasty from MathWorks with modelling and designing the motor controllers of the robot arm.

This report is mainly intended for the students who will continue working on this project, as well as the teachers who will assess our work. The document is divided into different chapters for different fields of engineering in order to facilitate finding the information that the reader is seeking. In the end recommendations for further development are listed.

Contact details

Below, the contact details for all group members, the client, and the tutor are presented.

Group members:

Name	Student number	Email	Phone number
Ray Mindiola	2588005	r.mindiolaromero@student.fontys.nl	+31 6 49 674 116
Sean Sminia	2718383	s.sminia@student.fontys.nl	+31 6 29 027 316
Leonardo Strik	2909332	l.leonardo@student.fontys.nl	+31 6 36 348 027
Mihail Marinov	3225666	m.marinov@student.fontys.nl	+31 6 40 901 311
Jordie Limpens	3312976	j.limpens@student.fontys.nl	+31 6 27 122 022

Client:

Name	Pablo Negrete Rubio
Email	p.negreterubio@fontys.nl
Phone number	0885074731

Tutor:

Name	Peter Klijn
Email	p.klijn@fontys.nl
Phone Number	0885070657

Summary

Robotics can be prohibitively expensive and complex, preventing robotics students from learning about them. The goal of this project was to design an affordable, easy to use robot arm that would help solve these problems. The project was split into four parts: the electrical design, the mechanical design, the Field Oriented Control (FOC) implementation, and the simulation. Each of these projects' parts will be discussed in order below. For the electoral design, there were three main decisions: Using two different power supplies, a low and high voltage one, to eliminate interplay , the use of separate FPGA drivers for each motor, to minimize interference and cables through the arm, and the application of the I2C protocol to communicate sensor and motor data. In regard to the mechanical design, there were two important decisions: The use of aluminium extrusion, to minimize weight and complexity, and the application of pre-built joints, to minimise cost and complexity. The FOC implementation within Simulink and the exported VHDL code seemed to behave accordingly, although sufficient testing lacked due to time constraints. In order to minimize complexity and cost, pre-configured full system kits were chosen for the FOC solution. In regard to the simulation, Unity was chosen as a simulator. The interfacing between ROS and Unity is done through ROS Bridge and ROS Sharp. Joint state control and readout has largely been implemented, although only a few sensors are. The user interface for the control of the robot was implemented in the ‘TKinter’ Python framework, because of its cross-platform applicability. In general, most parts of the problem were delayed due to the circumstances, and therefore this document should mostly be used to guide future design, rather than as a full-fledged design document.

Table of contents

1. Introduction 1
1.1. Report reading guide 2
2. Project description: 3
2.1. Project goals 3
2.2. Use case 3
2.3. Requirements 3
2.4. Organization 3
3. Electrical design 5
3.1. Requirements 5
3.2. Design decisions 5
3.3. Hardware components 6
3.4. Schematic design 6
3.5. Future steps 10
4. Mechanical design 12
4.1. Manipulator frame 13
4.2. Manipulator joints 13
4.3. Mechanical design requirements 14
5. Field Oriented Control implementation 15
5.1. Structure of the FOC model: 15
5.1.I. The input: 15
5.1.II. The FOC: 16
5.1.III. The motor: 17
5.2. Variable input for the model: 18
5.3. Model limitations: 18
5.4. VHDL import: 19
5.5. FOC alternative methods 19
5.5.I. Logic IC and amplifier 19
5.5.II. Power logic IC 20
5.5.III. FOC control board 20
5.5.IV. Full system 21
5.5.V. Advantages/disadvantages 21
5.5.VI. Recommended solution 22
6. Simulation 23
6.1. Simulator of choice 23
6.2. ROS Communication 23

6.3. Unity Robot Simulation 24
6.4. The User Interface 25
6.5. Future Work 26
7. Conclusion 27
7.1. Electrical design 27
7.2. Mechanical design 27
7.3. FOC implementation 27
7.4. Simulation 27
8. Recommendations 28
8.1. Electrical design 28
8.2. Mechanical design 28
8.3. FOC implementation 28
8.4. Simulation 28
9. References 29

Table of figures

Figure 1: V-model illustration, taken from (RMS, 2017). 3
Figure 2: SCRUM method illustration, taken from (Gonçalves, 2018). 4
Figure 3 Motor Drivers 7
Figure 4 ADC 8
Figure 5 Motor Encoders connectors 8
Figure 6 Negative voltage generator 9
Figure 7 JTAG interface 9
Figure 8 Part of power supply 10
Figure 9 Tool power supply 10
Figure 10: An early-stage concept design for the manipulator. The individual degrees of freedom are marked and numbered. 12
Figure 11: The aluminium extrusion which was chosen for the manipulator arm. 13
Figure 12: the robolink D high-end robotic joint size 30 from igus (igus). 13
Figure 13 - FOC model structure 15
Figure 14 - Input block 15
Figure 15 - FOC subsystem 16
Figure 16 - FOC block 16
Figure 17- PWM Generator block 17
Figure 18: Motor Subsystem. 17
Figure 19: Motor speed (blue) compared to the target speed (yellow). Y axis is RPM, X axis is time. A load is added at 0.5 seconds. 18
Figure 20 MC58420 from Magellan (Performance Motion Devices, Inc., n.d.) 19
Figure 21: TMC4671-ES from Trinamics. (Trinamic, n.d.) 20
Figure 22: An integrated FOC motor control board by Odrive 20
Figure 23 EVALKIT-ROBOT-1 from STMicroelectronics (STMicroelectronics, n.d.) 21
Figure 24: Architectural overview of the ROS communication structure. 23
Figure 25: A picture of the GUI. On the left are the joint controls, and on the right the manipulator visuaisation. 25
Figure 26: Velocity controller block inside the FOC subsystem. 32
Figure 27: Current controller block inside FOC subsystem. 32

Table of tables

Table 1: The additional requirements for the design. 14
Table 2: FOC Solution comparison table. 21
Table 3: The table of requirements, split into ROS, Cost, Safety, Size, Production and Design requirements. 31

1. Introduction

Robotics is an ever-growing field of education, where many students learn valuable technological skills. However, this type of education can be gated by the high cost of robotic platforms, which allows only few of these units to be purchased. This has been experienced by the project group in their own minor, where gripper model testing was greatly complicated by the small number of available arms. Furthermore, robot platforms can be complex and closed source, preventing students from learning more about their inner workings.

As part of the minor Adaptive Robotics, and in cooperation with Pablo Negrete Rubio, a project was formed, with the intent of solving these problems. Pablo expressed his desire for a small form factor robotic manipulator arm that could be mounted on mobile platforms. With the intent of making the arm ROS compliant integration with mobile platforms should be made easy.

As this project is part of the adaptive robotics minor, the project length is set to end on the 3rd of July, 2020. The deliverables of the project include this report with its design files and an ending presentation as well as a prototype. Further Two specific learning goals were set by Pablo, FOC drive and ROS compliance. These learning goals were written as must in the requirements list. In order to achieve a functional prototype, and maintain as low a price-point as possible, pre-manufactured components are used, instead of designing parts from scratch.

Besides the design the scope of the project is important. The design was envisioned to be portable and to be used by students and researchers. This gives a different design philosophy as functionality and technical documentation becomes more important. Matters such as marketability and designing for specific use-cases is outside the scope of this project. This includes not creating variants or investigating certifications at this point in time.

Due to the impact of the Corona virus regular operation could not continue. For the project this meant not having physical access to the Fontys building or supplies. Therefore, the project deliverables were altered to a virtual prototype in the form of a simulation and the design and validation files being changed to design and recommendations. In this document each component of the design has its own recommendation as to why it was selected but no physical tests associated with it. The simulation is the only section in which a test case was tested.

The limitations of working from home and having everything being virtual have presented challenges. It has also prevented a lot of testing and has allowed for little progress into researching production as well as user interaction. While a design could be made from the recommendations alone, thorough scrutiny should be applied as the lack of physical interaction might have influenced the design.

1.1. Report reading guide

The information within the report will be structured in the following manner:

- The first chapter, introduction, contains background information surrounding the project, defines the project scope of the project.
- The second chapter, project description, contains the goals of the project, the projected use case, defines the requirements of the project and describes the organization techniques used in the project.
- The third chapter, electrical design, addresses the electrical design decisions of the affordable manipulator arm. It contains the progress achieved in the project regarding this area of the project, including some recommendations on how to continue with the design.
- The fourth chapter, mechanical design, addresses the electrical design of the affordable manipulator arm. It contains the progress achieved in the mechanical aspect of the affordable manipulator arm. Includes hardware choice suggestions for the future.
- The fifth chapter, field-oriented control implementation, goes over the experimental approach taken to introducing an FOC into an FPGA. It goes into detail on the approach used to accomplish the task. It also goes over possible, comparable alternatives that should be considered.
- The sixth chapter, simulation, contains the progress pertaining the simulation of the project. Including choice of environment, communication with ROS (Robot Operating System) and it addresses the progress made creating a graphical user interface for this simulation.
- The seventh chapter, conclusion, contains the results achieved in the project.
- The eighth chapter, recommendation, contains all the points of interest for the improvement of the results achieved and for the continuation of the project in the near future.

2. Project description:

This chapter will contain all relevant details about the minor project, including the vision for the project and the project approach and organization. This first part of the chapter will go over the goals of the project. The second section will describe the use case of the project. In the third section, the requirements drawn will be presented. In the fourth and final section of this chapter, the planning and organization of the project are overviewed.

2.1. Project goals

The ultimate goal of the project is to create a design for a miniature robotic manipulator, upholding an interest of keeping the cost of production as low as possible. The creation of such design would then be applied for the education sector, targeting specifically students and researchers, for them to freely experiment and work with robotic arms without concerns of availability in the future.

The design recommendations produced by the project will be evaluated and analysed, creating future project opportunities for the affordable manipulator to fully come into fruition.

2.2. Use case

The projected use case would be that of a miniature robotic arm, which should interface with ROS (Robot Operating System) or directly into its control board. Where it can be used in a similar manner as to a Universal Robot series manipulator, with the ability to taken home with relative ease. It should be able to accomplish movement with 6 degrees of freedom.

2.3. Requirements

The full table of requirements can be found in Appendix 1. An important remark that must be kept in mind, is that these requirements were agreed upon with the intention of producing the final design. However, unfortunately, the world underestimated the severity of the global pandemic caused by the coronavirus. This had severe consequences on what was able to be accomplished by the project group, and as such, the scope of the project was ultimately reduced to the design of the arm.

2.4. Organization

During this project, a combination of the V-model and SCRUM method was used. The V-model is an abstract model of the system development lifecycle. The V-model is split on two sides: the left side of the V-model systematically outlines the user requirements and system design; at the neutral bottom point lies the integration of the system; the right side focuses in the verification and testing of the project. An illustration of the V-model can be found in Figure X.

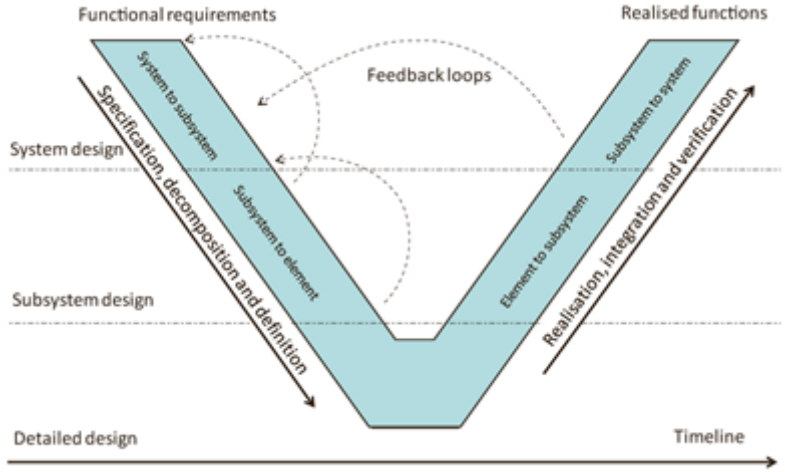

Figure 1: V-model illustration, taken from (RMS, 2017).

Affordable Manipulator With FPGA-driven FOC - Project description:

Upon agreement and request of the client, the SCRUM method was chosen. The SCRUM method splits the whole assignment into small, timed sets of tasks called “sprints”. There are small meetings at the end of the sprint deadlines to check progress and discuss results achieved within those sprints. Within the following figure, an illustration of the SCRUM method can be found.

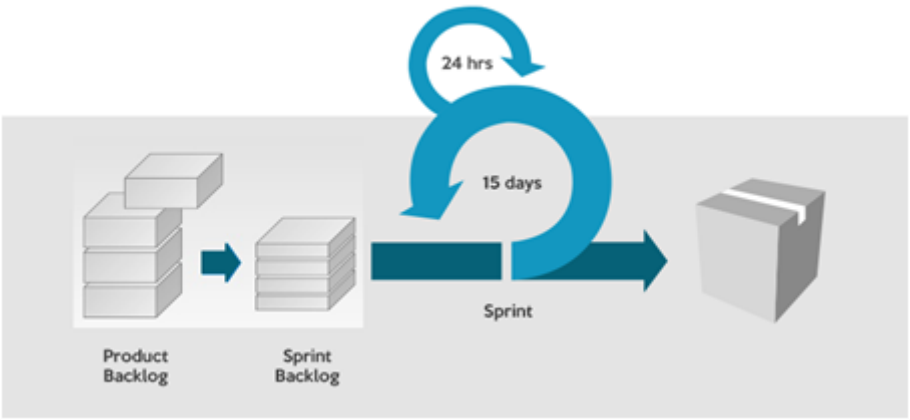

Figure 2: SCRUM method illustration, taken from (Gonçalves, 2018). .

The use of these tools proved themselves invaluably useful when faced against the current world situation. It allowed the team members to stay well in touch with each other and allowed an appropriate handling of the oncoming issues and challenges that would have been otherwise difficult to address given the current global pandemic crisis.

3. Electrical design

The robot arm is a machine with specific power supply requirements. The problem comes not from the high-power consumption, but from the fact that motors instantaneously switch from requiring zero power to requiring hundreds of watts, and vice versa. In this case there are six 100-Watt motors which should be able to move at once. Moreover, there are also components which use a lower voltage rail such as the motor controllers or main microcontroller. This low-level voltage line must be stable regardless of the motor's usage in order to keep the robot arm safe and precise.

The manipulator is designed to be interchangeable with the Universal Robots UR5 for tool interfacing. In other words, the manipulator should have the same tool power supply and logic voltage levels. The robot must be able to use the normal power grid and be plugged into a normal power socket and it must also adhere to the current electrical safety standards.

The 6 motors will cause EMI and therefore the robot's electrical systems should be designed to be able to withstand noisy environments.

3.1. Requirements

- The robot must use the standard 230V 50Hz power grid.
- The robot must feature safety system for short circuit, overload, over voltage, over temperature, EMC protection
- The power supply must have power factor correction
- The power supply should be able to continuously deliver power to 6 x 100 W motors.
- The low-level voltage rail should be independent from the high-level voltage line
- The tool power supply should provide 24V and at least 600mA current
- The tool I/O must use 24V
- The communication with the main controller and other systems should be able to withstand EMI
- Cables should fully fit inside the arm, with no outside connections.

3.2. Design decisions

Because of the requirement for a stable low voltage line and the fact that the high voltage rail might become unstable when motors start or stop it was decided to use 2 separate power supplies. The low voltage line will be 5V and the high level one will be 48V.

In order to minimize the amount of cables inside the robot arm the motor drivers will be inside the axes, next to the motors and the main controller will communicate with them through the I2C protocol. In this way there will be only 5 cables that run through the arm, moreover the signal will be more noise resistant compared to the situation where the motor drivers will be inside the base and the connection from them to the motors will run through the robot arm.

The tool I/O will also be controlled through the I2C lines. This is possible since the tool does not require a big I/O bandwidth and the I2C standard bandwidth will be fast enough to accommodate both all of the motor controllers and the tool I/O as well. The tool power supply will step down the 48V voltage line to 24V using a switching mode DC/DC converter as it would be more power efficient compared to the other option of using a linear power regulator.

A requirement of the project was to implement the field-oriented motor control on an FPGA. There were two options to achieve that. Either there had to be one or multiple FPGAs in the base of the robot or each motor controller had to have a separate FPGA. The problem with the first approach was that the amount of wires necessary to connect from the motor controller board to the FPGA will

be so big, that it would be impossible to fit all the cables for the 6 motors together inside the arm. The option to use some communication protocol for communication with the motor controller boards through less cables would have been unfeasible as it would require things to be send sequentially which would eliminate the whole point of using an FPGA. Therefore, the approach with an FPGA per motor controller was chosen.

3.3. Hardware components

For the main power supplies a MEAN WELL RSP-750-48 (Mean Well, 2019) and a MEAN WELL RS-35-5 (Mean Well, 2011) were chosen for the 48V and 5V lines respectively. They fit all the requirements for safety features and the RSP-750-48 also has active power factor correction. It can continuously deliver 750 W and has active cooling. Thus, it would be able to power the motor drivers for extended periods of time. Unfortunately, something that cannot be remotely tested or calculated is how well it behaves with the transient load of the motors.

The low voltage power supply can deliver up to 35W, which would be enough for all the systems that use 5V or less. Moreover, those systems do not cause the same problems with transient loads so the power supply will handle it well.

Because each motor controller board has an individual FPGA, an FPGA with less pins and less logic gates could be chosen. The Spartan-3A Xilinx XC3S50A-4VQG100C (Xilinx) was a suitable choice because of multiple reasons. Fist it got a suitable price for the goal. It also has a package with pins on the side instead of a ball grid array. This makes it much easier to solder it by hand, which is helpful for this in-house project. Moreover, it is from Xilinx, which was already used previously in the minor, thus the students were already familiar with its software.

The ADC is ADC0820BCN (TI), it was chosen because it has a suitable package and it can measure as much as 400kSPS, which is enough for the FOC motor controller.

The OpAmps used for conditioning the current for the ADC are MCP602SN. They are SMD, cheap, work with a 5V power supply and have a good bandwidth and voltage slew rate for the purpose (MCP602SN datasheet).

The power MOSFETs are BUZ10 (ST), they have a suitable power dissipation, maximum current, maximum drain source voltage, maximum gate source voltage and are of a suitable price.

3.4. Schematic design

Because the free version of EAGLE does not allow designs with more than 2 sheets of a single schematic the schematic snippets will be cut from the big schematic.

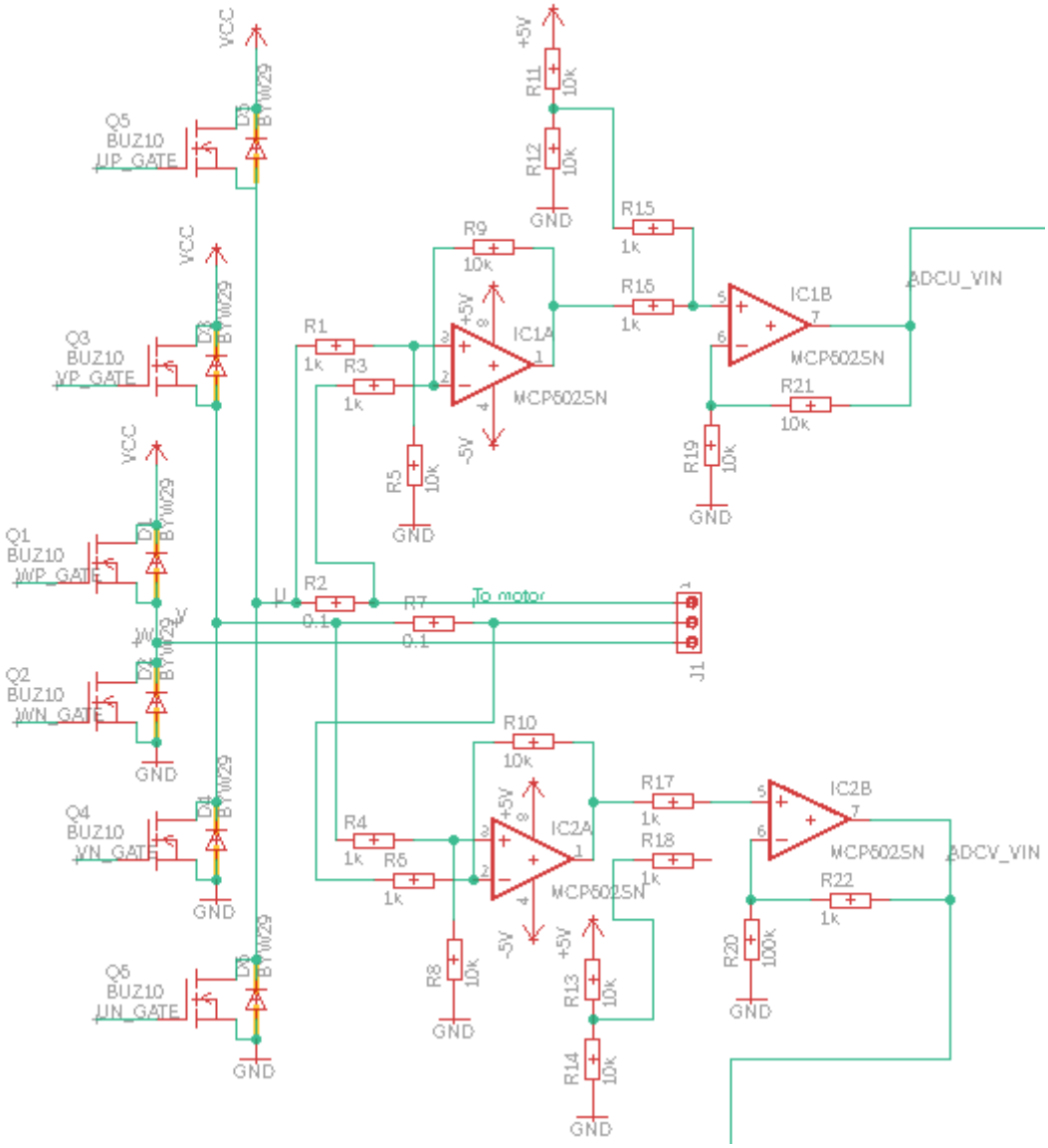

Figure 3 Motor Drivers

The motor drivers (Figure 3 Motor Drivers) use a standard architecture for controlling 3 phase AC motors. There are also 2 current sensing resistors (R2, R7). The voltage over them is measured by the first OpAmp and then it is added to 2.5V and amplified to better fit in the 0 – 5V range of the ADC.

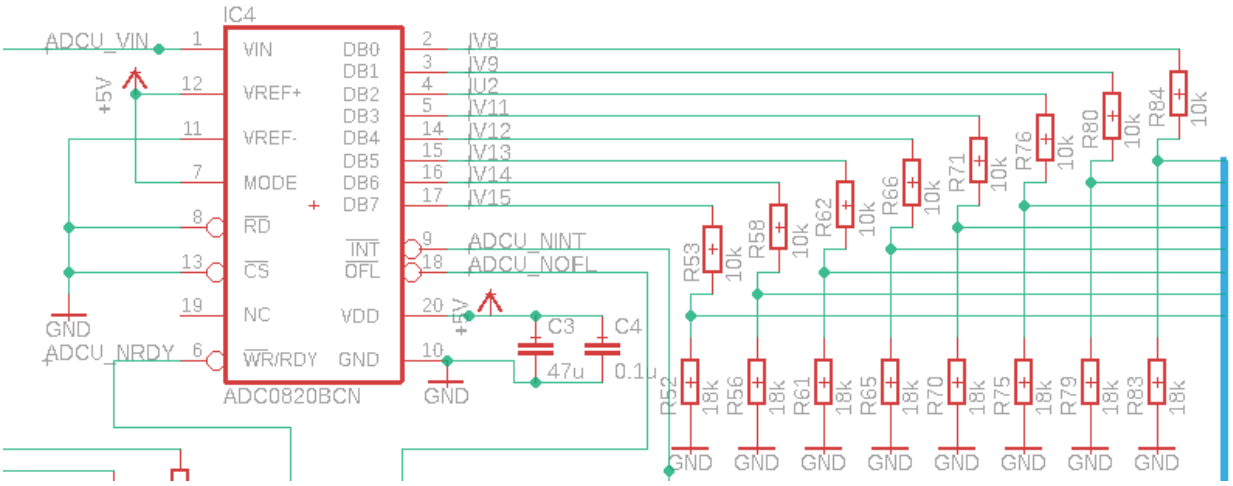

Figure 4 ADC

The ADC (Figure 4 ADC) uses 5V logic levels, so it is necessary to step down the voltage from 5V to 3.3V so that it can be read on the FPGA. This can be done either by an integrated logic level shifter, but in this case, it wasn't necessary to do bidirectional voltage level transfer. On the bright side, the ADC accepts any signals higher than 1.8V as a logical high so shifting the voltage the other way around is not necessary. On the other hand, this is currently not necessary as in this configuration the ADC measures automatically all the time, so there is no data going from the FPGA to the ADC.

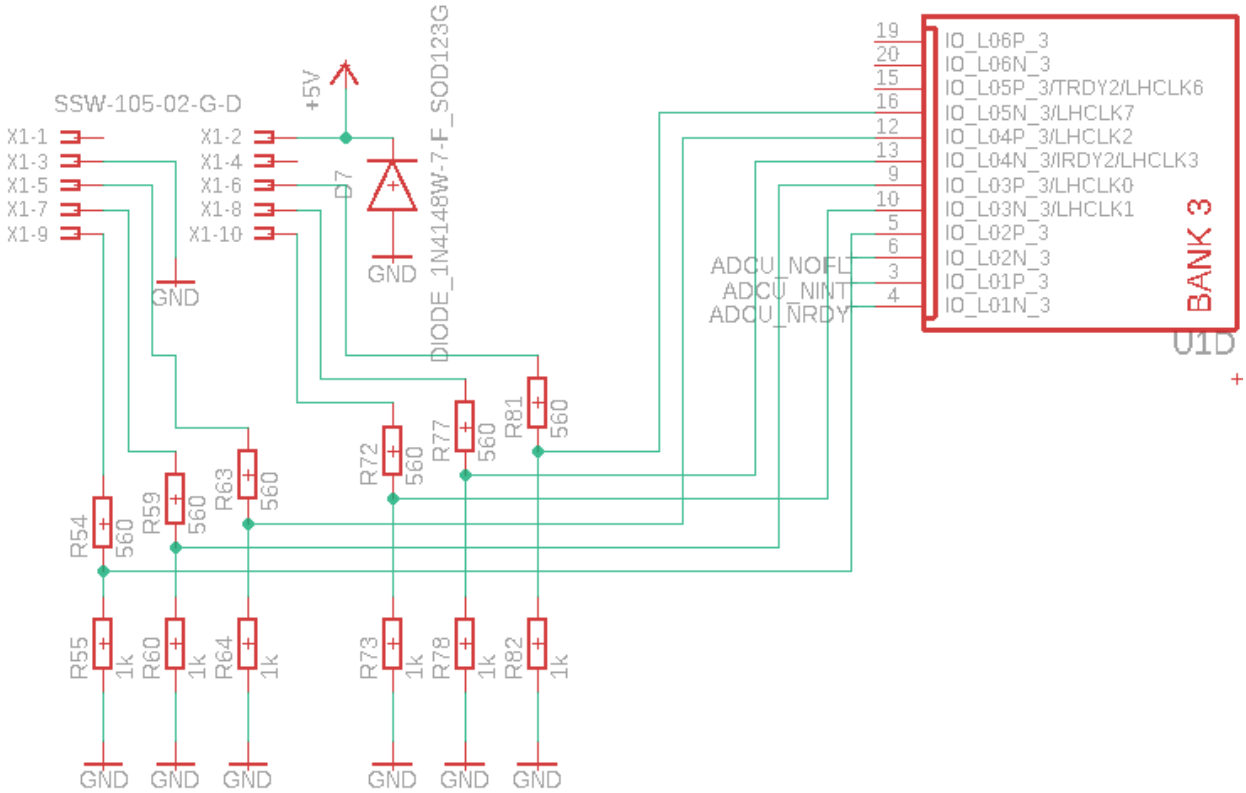

Figure 5 Motor Encoders connectors

Another component that attaches to the FPGA are the motor encoders (Figure 5 Motor Encoders connectors). They also work on 5V, so the same voltage level shift is necessary. For the same reason that the data in unidirectional, it is easier to use resistor voltage dividers instead of an integrated voltage level shifter.

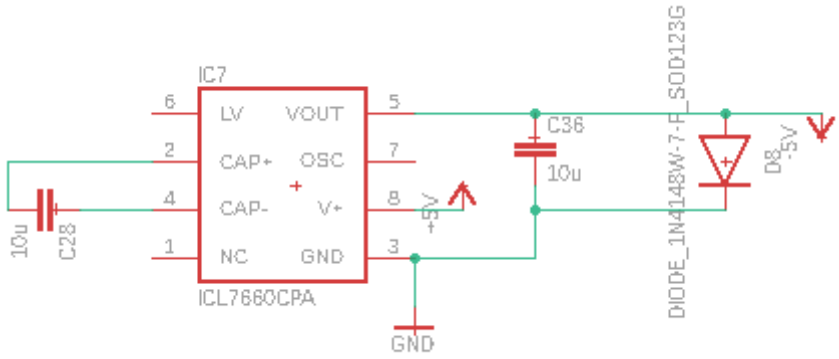

Figure 6 Negative voltage generator

This negative voltage generator (Figure 6 Negative voltage generator) is necessary to measure the voltage over the current sensing resistor for the motor current.

There is also the JTAG interface for programming the FPGA (Figure 7 JTAG interface). It attaches to an external programmer so that the FPGA can be programmed while in circuit. There is also an indicator LED, to show when the FPGA is ready.

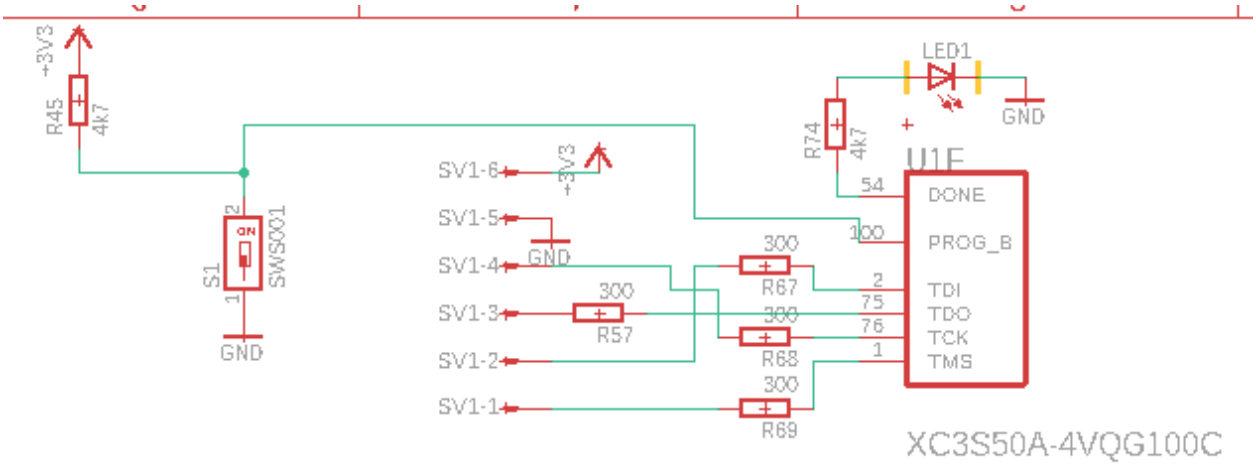

Figure 7 JTAG interface

The last thing is the power supply for the FPGA (Figure 8 Part of power supply). It can use different power supplies for each I/O bank, but in this design, this is not necessary as all the lines use 3.3V. The core of the FPGA uses 1.2V which was also done using a linear voltage regulator. There are numerous capacitors in order to make the voltage line stable and remove harmonics. The capacitors were used based on the design of the development board of the same FPGA (Xilinx, 2007). There are also 2 similar configurations on the left of the U2G block but were not included in the screenshot for

readability.

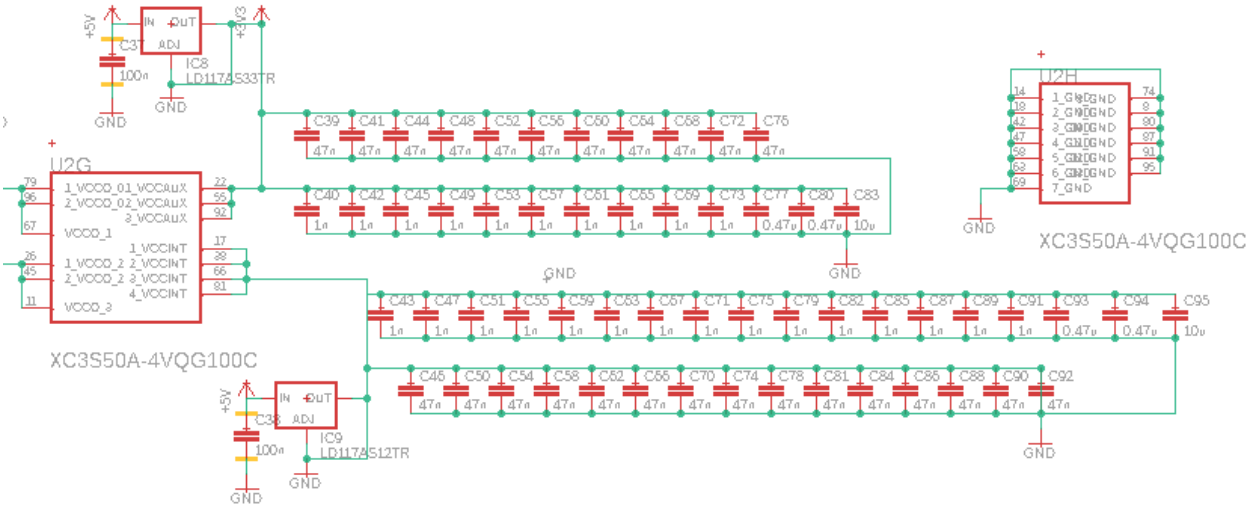

Figure 8 Part of power supply

The tool power supply (Figure 9 Tool power supply) uses a buck converter to step down the voltage. This design actually allows currents higher than the one that the UR5 uses, but a higher maximum than the requirement is not a problem. So this design is feasible.

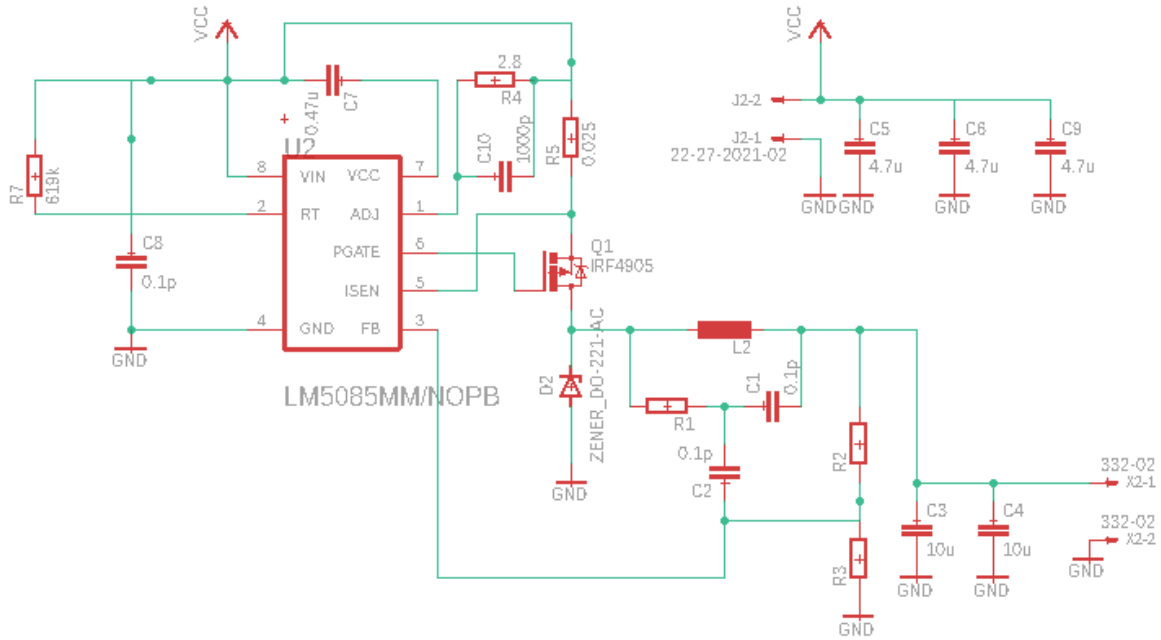

Figure 9 Tool power supply

3.5. Future steps

First and foremost, the circuit board must be built as a prototype and tested, whether all voltage levels are as expected.

When the circuit board is verified to behave as expected, a test script should be made, to start the motors and move them with full power before stopping again in order to monitor if the power rails are stable during usage. If that is not the case either the motor control algorithm should be changed

in order to make the motor startup more gradual, or additional capacitors can be added in the robot base, if the power line is unstable, but still very close to being within limits.

The board layout is not done, as this depends a lot on the real-life test of the circuit board. For example, if the motor driver MOSFETs get very hot, a heat sink should be integrated, or the MOSFETs can be thermally coupled to the case of the robot arm in order to spread the heat over a bigger area. The temperature of the other components should be also monitored.

To make the board layout compact, but still serviceable in Fontys 1206 SMD capacitors and resistors are best to be used. They are a good compromise between space, and serviceability. Not all components can be SMD, as some of the capacitors for the power supply cannot be found in an SMD package.

The 48V power cables that run inside the arm should have a core area of at least $4mm^2$ to cope with the continuous current if the motors are continuously on.

The tool I/O has to be finished because of time constraints it was not been done. It can be either done using a small microcontroller to set latches and read inputs. Another option is to use an already done GPIO I2C controlled expansion board. In both cases the voltage level will have to be shifted, to and from 24V.

4. Mechanical design

This chapter will detail the mechanical design. Unfortunately, due to time constraints, the mechanical design was not finished. The choices and design decisions that were made will be detailed below.

Of the set requirements, those that were relevant during the mechanical design are those concerning the payload weight, size and production of the arm. The arm can be no longer than 710 mm, must be able to take a payload of 200 grams, and must be able to be produced relatively simply, for less than € 1500 in larger numbers. A concept design for the manipulator is shown in Figure 10. This figure also shows the degrees of freedom. Also shown in the concept model is the base, the size of which is not definite yet, due to some uncertainty on the electrical front. However, a base with some cooling facilities needs to be included in the design.

Figure 10: An early-stage concept design for the manipulator. The individual degrees of freedom are marked and numbered.

4.1. Manipulator frame

The material chosen for the manipulator arm itself is aluminium extrusion, specifically a 50*50 mm type 2E L one, shown in Figure 11. This allows for simple assembly of the manipulator, and is readily available for a low price. This profile also has covered channels running through it, which allows the cables to be hidden, as per the requirement, and helps prevent electromagnetic interference.

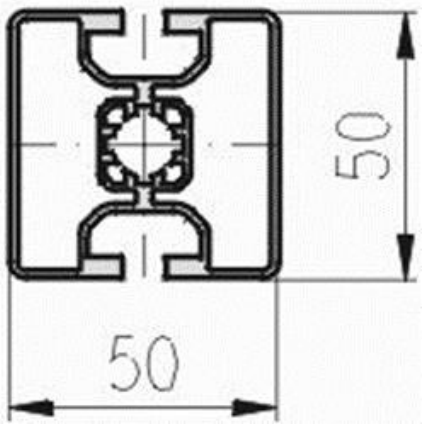

Figure 11: The aluminium extrusion which was chosen for the manipulator arm.

4.2. Manipulator joints

During the design process, the design of the joints quickly proved to be the most challenging. The fact that the cables needed to be hidden means that the joints have to be able to have the cables pass through them. This means that the bearings required have to be extremely large, and therefore costly, upwards of €150. Secondly, the high motor speeds requires significant reduction. In order to prevent motor backdriving and achieve this reduction, a worm gear redactor was chosen. Sourcing worm gear redactors proved that these, also, are relatively costly, and can cost upwards of €300 each. Since both of these components alone would cost upwards of €450 per joint, which would exceed the budget, a more affordable solution had to be found. To provide joints that satisfy the requirements at a reasonable cost, Igus robolink joints were found. The robolink D highend robotic joint size 30, which is shown in Figure 12 has a width of 45 mm (igus), which matches the aluminium extrusion, and is therefore the component that was chosen for the joints. These joints cost €288, which presumably could be lowered when scaling up production.

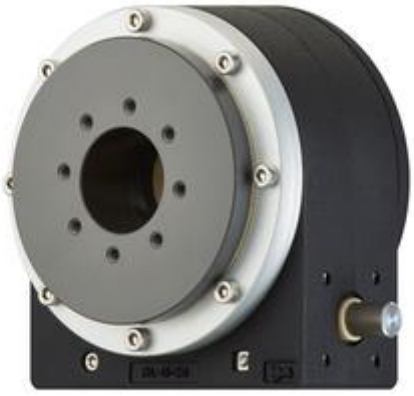

Figure 12: the robolink D high-end robotic joint size 30 from igus (igus).

4.3. Mechanical design requirements

In order to aid further mechanical design, the recommendations that result from this project are summarized as a set of requirements, which can be used in addition to the main project requirements in the plan of approach. These requirements are categorized in Must, Could, Should and Would requirements, and can be found in Table 1.

Table 1: The additional requirements for the design.

Nr.	Requirement	MSCW
1	A hollow base to house the electrical components such as power supplies and circuit boards, with adequate cooling must be provided.	M
2	The base should be weighted enough to offset the arm plus payload.	M
3	Complete pre-fabricated robot joints should be used in order to minimise production cost and complexity.	S
5	The possibility of using different motors and joints for different degrees of freedom, according to the load. This would minimise cost.	S
5	In order to allow the cables to be hidden in the arm, and minimise weight, the arm segments should be hollow.	S
6	In order to allow the cables to pass through the joints, bearing with a large enough inner diameter should be used.	S

5. Field Oriented Control implementation

The project requirements include the implementation of an FOC within a FPGA board to achieve the control of smaller, yet still powerful motors, to be used in the design of the arm. In order to do so, it was decided that a using Xilinx Vivado would present multiple obstacles in the future of the project; specifically, in regards to complex functions such as the Park transform, and the inverse Park transform. As an alternative, MATLAB Simulink models can be exported to HDL, which then in turn, can be synthesised within FPGAs. This method was chosen for the project, because of the flexibility to tune the model to motor specifications and controller requirements it offers.

5.1. Structure of the FOC model:

The model, which is shown in Figure 13, is divided into subsystems to simplify the process of modifying the model, and make it clearer to understand, for the continuation of the project in the future. As it stands, the current model is divided into 4 subsystems: Input, FOC, Motor and Visualization.

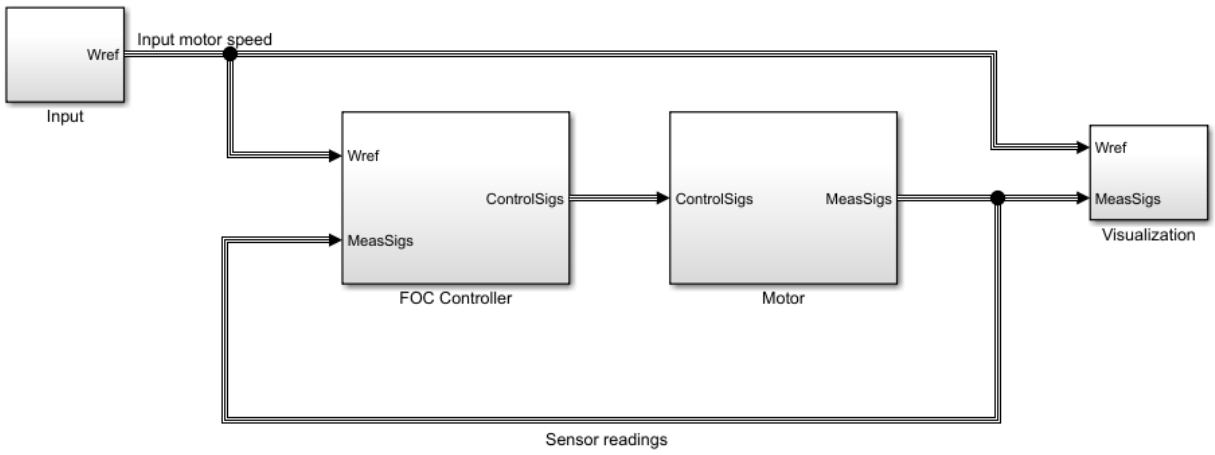

Figure 13 - FOC model structure

Below, a brief overview of each block and its functions will be given:

5.1.I.The input:

The input subsystem, shown in Figure 14, contains the generation of the input for the model, which is the desired rpm of the motor. This value is then transformed into radians per second (rad/s) before being exported out of the block.

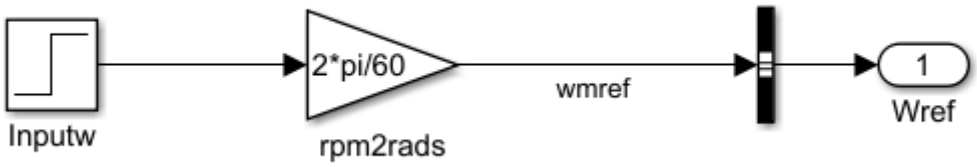

Figure 14 - Input block

5.1.II.The FOC:

The FOC subsystem is further divided for clarity, it is comprised of the input speed block and sensor block, which decouples the input signals and sends them to the FOC block. the motor signal generation, which acts as a PWM generator, taking the control signal from the FOC and transforming it into voltages for the motors. This is visualized in Figure 15.

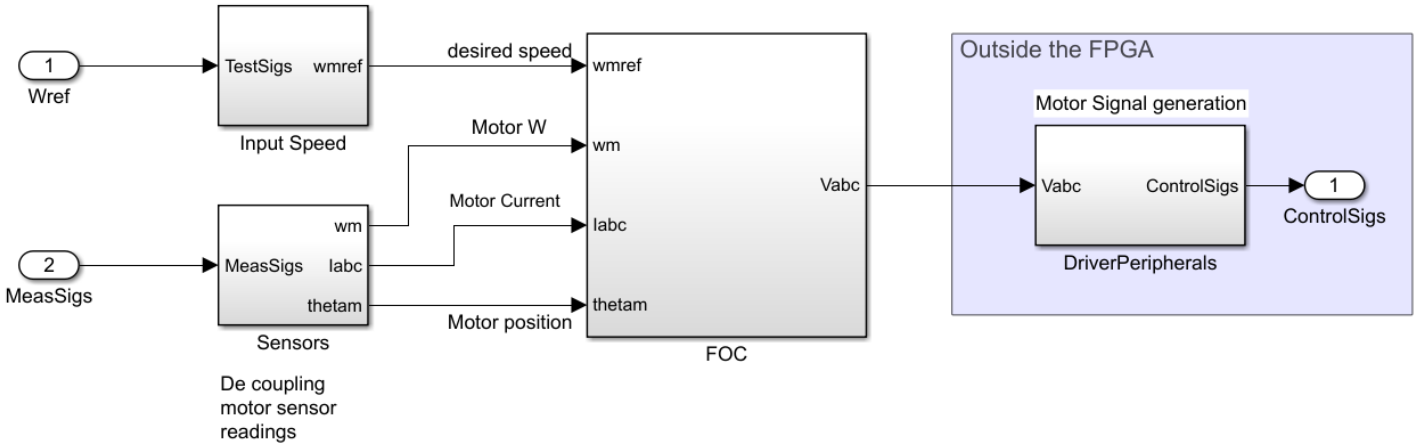

Figure 15 - FOC subsystem

Inside the FOC block, the desired speed and measured speed are taken into a standard PI controller, which can be seen in Appendix 2, the control signal output of the velocity PI is then used as a reference for the current controller. The current controller uses PI controllers for the current components given from the Park Transform (Id and Iq), which can be seen in Appendix 2. The FOC block can be seen in Figure 16.

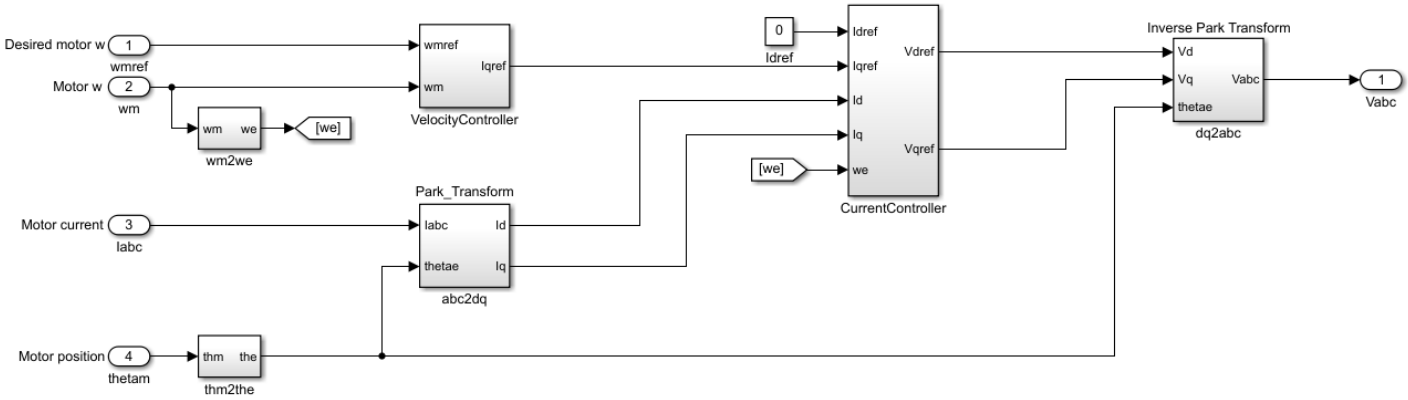

Figure 16 - FOC block

The output signal from the FOC is taken into the PWM generator, which outputs 6 gate signals for the control of the motor. This part of the model is external to the FPGA and the output is directly connected to the motor subsystem. It is illustrated in Figure 17.

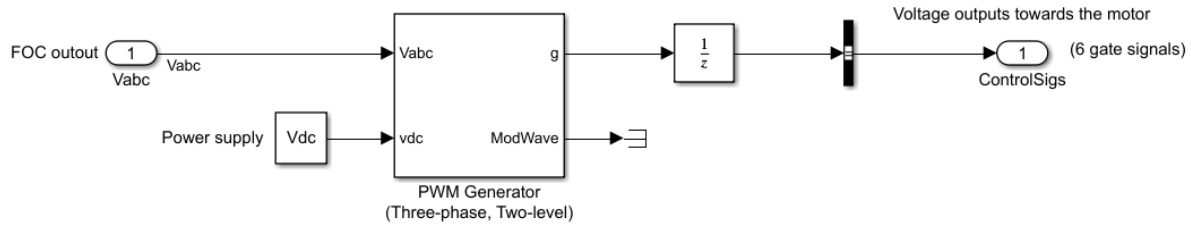

Figure 17- PWM Generator block

5.1.III.The motor:

The motor subsystem, shown in Figure 18, is comprised of a modelled inverter, which represents a circuit of six Insulated-gate bipolar transistors (IGBT) to convert the control gate signals from the FOC into the Phase voltages for the motors. The current and phase voltages are measured, and the current measurements are used for the controller. The motor block has been modelled as a permanent magnet synchronous motor. The parameters of the modelled motor have been adjusted to match the selected motor as closely as possible. Finally, there is a sensor that measures the motor position, speed, and torque, and a load attached to the motor to verify the integrity of subsystem.

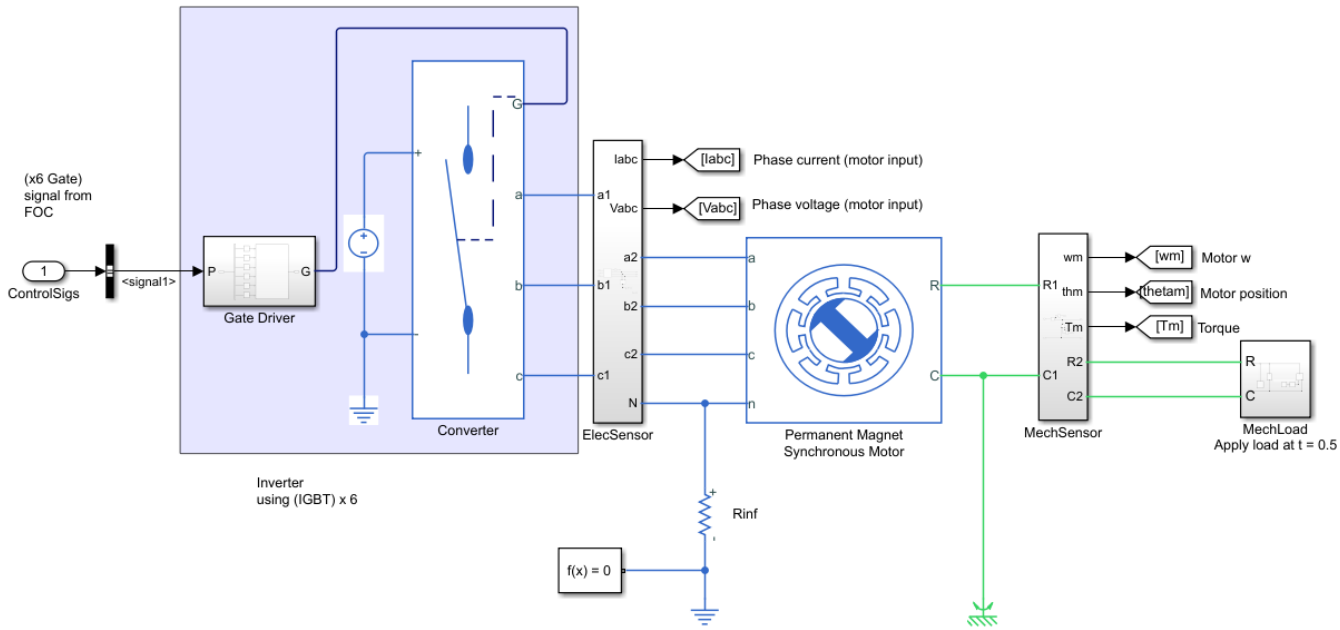

Figure 18: Motor Subsystem.

5.2. Variable input for the model:

Attached to the Simulink model file, is a MATLAB Script file which contains a list of simulation parameters that can be modified. In order to successfully run the simulation, it is required to run the parameter script file before the model file. In order to use the simulation file with another, similar motor, the values must be modified, and the controllers need to be tuned again.

The parameters required to be filled in should be found within the component datasheets and/or provided at request from the manufacturer.

5.3. Model limitations:

The model, as it stands, requires the tuning of three different PI controllers, which can be challenging. MATLAB Simulink's auto tuner function cannot be used for this, as it is incompatible with the use of PWM signals and continuous time functions across the whole model.

It is possible to overcome this issue by using the System Identification toolbox (Mathworks, n.d.), to linearize the parts of the model that include the controllers. But the license for the toolbox was currently unavailable and due to time constraints, it was not able to be acquired in time.

Despite that, the model is able to behave as intended, but if changes need to be made to the parameters of the model, then the tuning of the controllers must be addressed. The motor model behaviour is shown in Figure 19

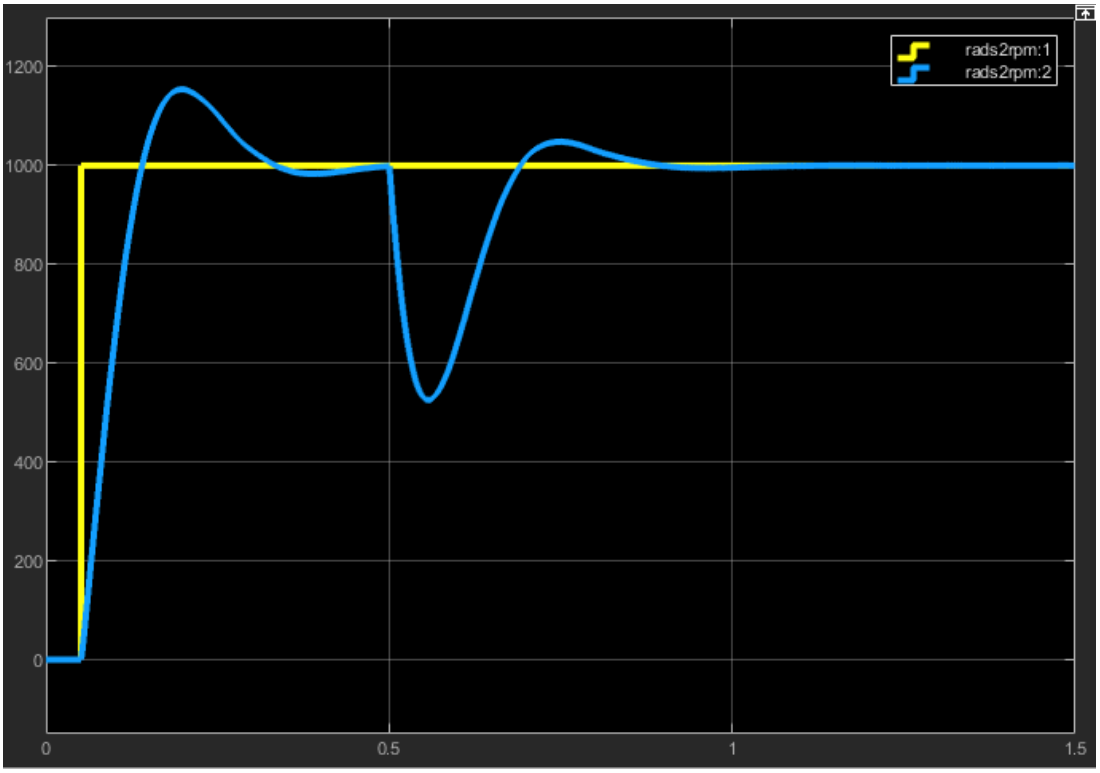

Figure 19: Motor speed (blue) compared to the target speed (yellow). Y axis is RPM, X axis is time. A load is added at 0.5 seconds.

5.4. VHDL import:

The FOC block is able to be exported into VHDL, and the code successfully compiles on Xilinx's Vivado software. Unfortunately, testing was not able to be performed due to lack of hardware and time constraints.

Given the current limitations of the approach taken in this project, it is advisable to also consider possible alternatives of FOC control that would lead into similar or more desirable results.

5.5. FOC alternative methods

There are Figure 20many alternative FOC solutions within the industry. Below is a brief description of several of these alternatives, along with their strengths and weaknesses. Below that is a comparison table that details on what criteria the method used in the project was chosen.

5.5.I.Logic IC and amplifier

Magellan's MC58000 series offers solutions in the form of an Integrated Circuit (IC) that handles the FOC logic , paired with a digital amplifier to power the motors, shown in Figure 20. This system is highly configurable as both the logic type and its tuning can bet set separately from the power delivery. (Performance Motion Devices, Inc., n.d.) The digital amplifiers regulate both the input and output voltage and can even correct for wire losses. However, these systems are designed for high-end use which is outside of the budget range for the project.

Figure 20 MC58420 from Magellan (Performance Motion Devices, Inc., n.d.)

5.5.II.Power logic IC

Trinamics's TMC4671 Series, shown in Figure 21, offers solutions for the smaller motors which the project requires (Trinamic, n.d.). It combines an amplifier and the FOC logic onto a single IC. Due to the size of the system it is only suited for small/ low-power motors. The ICs power input and output can be scaled to allow high power usage, the circuitry required for these changes can be found in the user guide provided by the manufacturer.

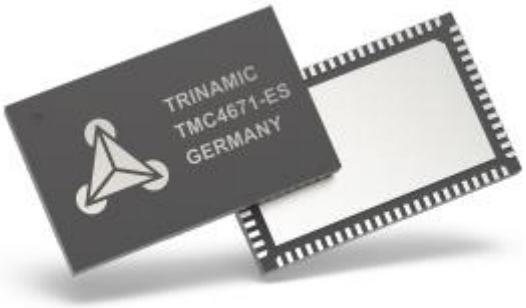

Figure 21: TMC4671-ES from Trinamics. (Trinamic, n.d.)

5.5.III.FOC control board

This option is a self-contained FOC motor control board. An example of one from Odrive is shown in Figure 22. These boards contain both a power circuit and a logic IC for the FOC calculations. These kinds of boards come pre-built and only require wiring to power it and connect the motor. These boards come in many variants. Boards vary in operating ranges, tune-ability, customization, frequency and more. Therefore, selection of the right control board to suit the motor is crucial in this case.

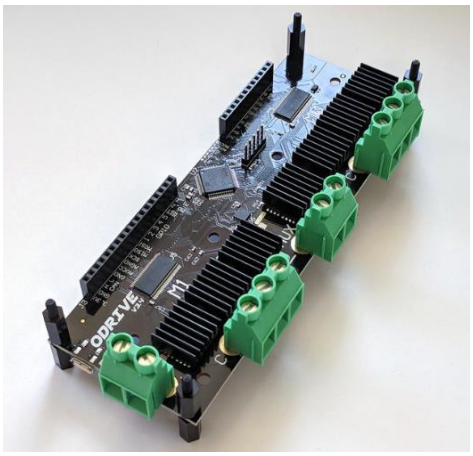

Figure 22: An integrated FOC motor control board by Odrive.

5.5.IV.Full system

The difference between a full drive system and the above mentioned options is that the former includes a pre-selected motor. By having a pre-selected motor, the freedom of customization is compromised. This makes mass manufacturing easy and allows for a consistent product, that is easy to use. The biggest downside of this solution is the limitation of the packages. They are limited to a small range of motors and operating ranges. An example of this, would be the EVALKIT-ROBOT-1 from STMicroelectronics, which is illustrated in Figure 23. It is an evaluation kit that includes motor and board. The motor included in the kit is the same as the one selected for the design, making it perfectly compatible with the design produced.

Figure 23 EVALKIT-ROBOT-1 from STMicroelectronics (STMicroelectronics, n.d.)

5.5.V.Advantages/disadvantages

To compare the different solutions, Several evaluation categories were created. Each category is then assigned an importance, in the form of a weight. It is assumed that motor specifications are available for each solution, and that each solution provides adequate levels of precision and torque, when making this comparison. Then, each option is ranked 1-4 on each category, where 1 is best and 4 is worst. These ranks are multiplied by the weight of their respective category, and added. The option with the lowest resulting score is the most preferable. To be clear, Tune-ability is the option to change the FOC behavior by tuning the internal PID. Configurability is the width of the margins for operation.

Table 2: FOC Solution comparison table.

	Cost	Ease of use	Tune-ability	size	configurability	Weighted scores
Weight	3	3	1	2	1	
Logic + amplifier	4	3	2	4	1	32
Power Logic	2	4	1	1	2	21
FOC control board	3	2	3	3	3	27
Full system	1	1	4	2	4	18

According to this ranking method, the Full system looks like the best option. This option is very cost efficient as the packages contain motors at an acceptable price. The ease of plug and play allows for quick integration with few problems. The biggest drawback of this option is the limitation of not being able to customize the solution. However, by selecting the right system this should not be an issue. The arm has a set requirements list (comes in a single version), which means the FOC solution does not need to be customized for multiple variants of the arm. Therefore tune-ability and configurability are not needed for this arm.

5.5.VI.Recommended solution

In this case, the full system was selected to be the best option. STMicroelectronics sells FOC kits that fit the requirements. After selecting the full system, a list of requirements in terms of power, size and torque was laid against the offerings found at STMicroelectronics. These kits are full standalone FOC drive systems that fit both the power requirements and precisions requirements of the arm. An affordable FOC was found at STMicroelectronics. The full name of the kit is EVALKIT-ROBOT-1. The current recommendation is to utilize this kit as the FOC drive solution for this case. The affordability and ease of implementation in this use case make it the recommendation to be used for the arm.

6. Simulation

One of the goals of the project was to create a digital twin of the robot and an interface for controlling both the real robot and the digital twin. Both robots also had the requirement of being fully controllable via ROS.

6.1. Simulator of choice

For the simulation the Unity platform was chosen as Gazebo ran quite poorly and had rendering problems. So, it was decided to look at other simulation systems and Unity ended up as the final choice. Unity is also a great option as it also gets used a lot in game design and for other programs. It is also a good fit for artificial intelligence. Unity has a nice programming interface for managing objects in the simulation.

An early choice was to try to make all control to still use ROS under the hood. The Graphical User Interface (GUI) would not require any ROS knowledge, but it will still communicate with Unity using ROS topics.

6.2. ROS Communication

The first thing to go into for the simulation is the communication between ROS and Unity as all control of the robots will be done over this system.

In order for communication to happen between the Unity simulation and the ROS system there are 2 ways of communicating:

- A native ROS extension that offers direct communication with ROS for Unity or the programming language that Unity uses (C#)
- A communication bridge system that bridges between ROS and a different communication type that can be made to work with Unity

The first option, though available, was a very out of date and not supported library, so in order to use this, it had to be extensively tested and likely even fixed or rewritten.
For the second option there already is a system build for ROS that allows communication in a different way. Another company also already started to work on implementing this specific system in Unity itself so the communication was for a big part already handled.

So in the end a RosBridge system was used, that exposes the ROS system over a standard web-socket (ROS Org, 2017). For Unity RosSharp was used (ROSSharp, 2019), which is a library created for Unity that uses the RosBridge system for communication with ROS. An overview can be found in Figure 24.

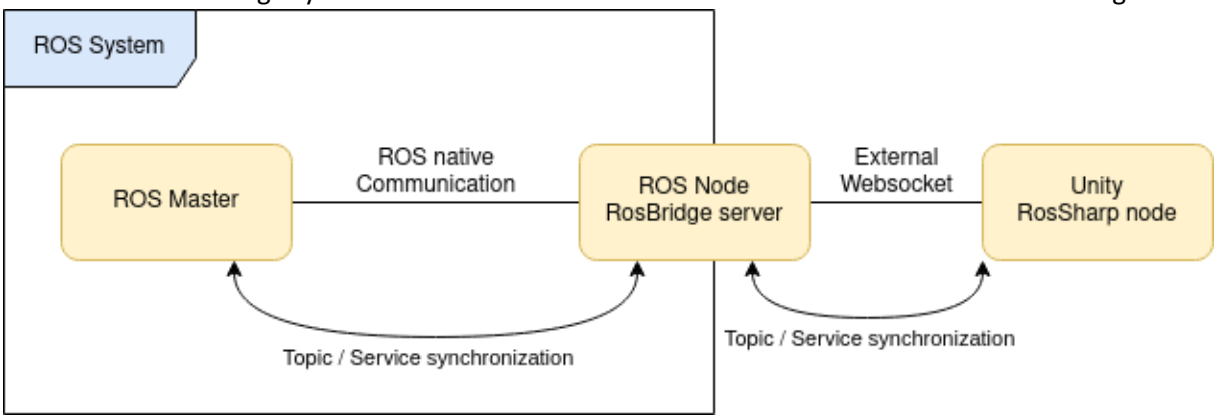

Figure 24: Architectural overview of the ROS communication structure.

For a more in-depth look into the different options and approaches that can be taken and how the possibilities of bridging the ROS communication between the ROS system and Unity were researched (ARMinor 2020 P1, n.d.).

6.3. Unity Robot Simulation

As discussed in the ROS Communication section, the simulation is using the RosSharp package from another company to bridge between ROS and Unity (ROSSharp, 2019). This package already has the pub-sub and service/request interfaces working out of the box. Also, most basic message types are included.

One of the more important parts in working with robots in ROS is that that the robot model is shared between the various components that need to drive or take its properties into account. For the simulation the model itself has to be imported with all the limits etc. To accomplish this. RosSharp has a built in URDF transfer utility. this way it is very easy to transfer a robot model to Unity. Doing the reverse should also be possible tough this functionality was never used or tested.

Probably the most important feature is being able to read out and control motors or joints. Readout of the joints was something that was already implemented but had some issues in the reporting that had to be fixed. There are still some issues as there are some issues with the way Unity reports the angles, But it now also has a way of calibrating its zero point.
The writing of the joint states had to be completely rewritten as the implementation by RosSharp only sets absolute angle values as opposed to controlling the velocity and effort as one would want when simulating actual behavior. The old absolute setting is still available for when one would like to use the system purely as a viewing application where it would mimic exact positions from another robot. This was the way the original system was intended at first.

One of the other parts required for getting joints working was a ROS Controller so that other ROS systems can send commands to the Unity simulation. This was quite a difficult task as documentation was quite sparse on how one would actually needs to implement these controllers.

The last set of functionality that would be needed inside of the Unity simulation is sensors. The RosSharp library already has sensors for lidar and regular camera’s, other would have to be build when needed. the camera also needs to be redesigned as the current implementation takes to much processing on higher resolutions.

6.4. The User Interface

The goal of the robot arm is to be an educational substitute to the Universal Robots UR5. Therefore, the user interface is designed to have similar basic functionality to the UR5, namely indicators and controls for each axis, real time robot visualization, moving the end effector linearly, control I/O. Of course, all of that should ideally be used to write a program that can run on the physical robot arm.

The GUI, which is shown in Figure 25, uses the Tkinter Python framework. This framework was chosen as it can work on Windows, Linux and also Raspbian and it also allows freedom into making custom complex widgets. Moreover, the basic widgets available in the framework are intuitive and easy to work with. Tkinter is also supported in python 2, which is the version that is currently used by the most recent ROS release (Melodic) as the time of doing this project. This makes the interface between the UI and ROS simple.

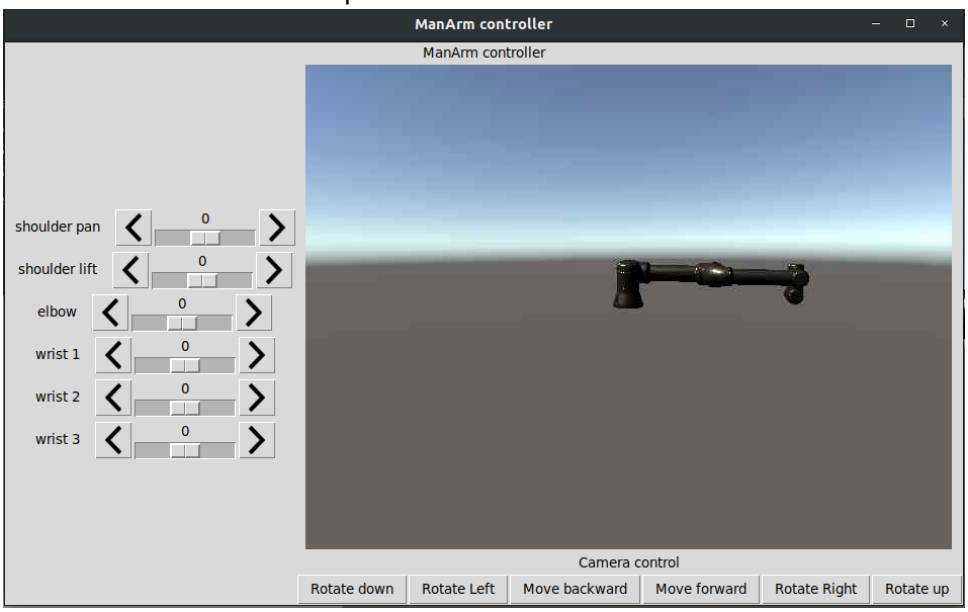

Figure 25: A picture of the GUI. On the left are the joint controls, and on the right the manipulator visualisation.

The GUI's development is progressing, but as its development started late, it is still in an early stage. Currently the only fully working part is the real time robot visualization. In the GUI there is a live view of the robot's simulation, in order to visualize the current position and movements of the robot. It was achieved by setting a unity camera to publish compressed images to a ROS topic. The GUI has controls to move the camera around as desired in order to best visualize the robot at all possible poses.

The GUI also has indicators and controls for the axes' current positions. The axes control is achieved through a ROS topic, which uses the Joy message type. In this way it is possible to have independent control of each individual joint. The indicators still do not work properly. They are correctly converted from radians to degrees, however so far, the current rotation of the arm is not shared from unity to the GUI. In other words, there is no way to tell whether the arm has rotated 90 degrees clockwise or 270 degrees counterclockwise from its reset position. There are also cases where wrong radians are reported to the ROS topic, which also cause wrong values in the GUI.

The other parts of the UI are still under development. After the functionality is done, the GUI would be designed so that it looks good is also useable in different screen resolutions and aspect ratios.

6.5. Future Work

As mentioned before. This system is by no means complete or finished. For the Unity simulation, sensors and joint support will have to be improved. Especially the sensor support has to be looked into so that more types of sensors become available for simulating. At the time of writing this report having full ros_control (the hardware bridge between ros and the robot) and moveit (the movement planning and execution system) support is being finished, but probably could use another revision to make it nicer.

The GUI also has to still be finished, so that it has all the basic features available in the UR5 simulator, moreover the goal is to be able to make a robot program on one system and then save it into a file. This file should later be put in the robot controller in order to execute the program.

A fully featured user-interface that would be compatible with any robot (Unity simulation, real or otherwise) would be a complete project on its own, especially when full scripting and sensor support would be included in it.

7. Conclusion

The goal of this project was to design an affordable manipulator arm, with FOC implemented on FPGAs. However, due to the global pandemic, communication and cooperation was made a lot harder, and the project suffered for it. Because of this, the project goal was adapted, and this report now serves to describe the progress that was made, in order to facilitate continuation of this project. Below, a set of recommendations that derive from this project are presented for each project part.

7.1. Electrical design

A set of requirements were derived for the electrical design of the affordable manipulator, alongside a list of components that have been selected to comply with the requirements and produce a practical design that can be implemented into the future. This design includes a detailed schematic that connects the different selected components together.

7.2. Mechanical design

A basic proof of concept was created for the mechanical design of the arm, complacent with the requirements established. The design could not be finalized due to time constraints; however, a list of hardware was devised. The list is meant to be used in the final design of the robotic arm to be used in conjugation with a list of requirements to produce the visualized design.

7.3. FOC implementation

A Field oriented controller was designed and created with the intention to be implemented and used a FPGA board, a MATLAB Simulink model was created with the intention to export the designed controller for FPGA applications. The results are promising, but the tools required for the tuning of the controller were not available at the time of writing. The model successfully was exported as a VHDL file. It proves that, in theory, this is a feasible approach. Due to the constraints of the pandemic and lack of time, the generated file could not be validated, and it is likely to have issues. To compensate for this, alternatives were explored and suggested that would have comparable performance for the project to continue if the generated file approach is deemed to costly or not feasible.

7.4. Simulation

The simulation of the robot has been achieved in Unity. This was accomplished using the open source RosSharp package from Siemens to successfully bridge ROS and Unity. The simulation is achieved by the importation of the robot models using URDF files. The control of the arm within the Unity environment, controlling joint speeds as it would be done in reality by rewriting parts of RosSharp. There are issues on join angle reports that are being evaluated and addressed. There is limited functionality with the Sensors that can be used with the Simulation, these being a camera and a lighter.

The proof of concept for a user interface for the control of the robot in this simulation was created, meant to reassemble the Universal Robot Series user interface. It was accomplished through the use Tkinter Python framework to ensure a comfortable array of platforms that should satisfy a broad number of possible users. Currently the GUI has a live visualisation of the robot, and indicators for each axis' position, but those indicators are inaccurate in edge cases. The GUI is still in development at the time of writing.

8. Recommendations

In order to continue the project and improve upon that which has been previously devised and explained, the following recommendations are to be considered. The recommendations are presented per project part.

8.1. Electrical design

A circuit board prototype must be built and tested, to validate the schematic design. Then a stress test should be conducted, having the motors run at maximum capacity before suddenly stopping to monitor the stability of the power rails while in operation. Depending on the results of the test, the motor control might have to be adjusted for a more gradual change of speed. Additional capacitors can be added at the base of the robot with the power line is stable, but close to acceptable limits. The board layout is incomplete and requires prototyping and testing. It is recommended to use SMD components as much as possible to keep the board layout compact. The 48V power cables that supply the motors should have a core area of at least $4mm^2$ to handle the continuous current flow if the motors are constantly in use. The I/O ports for the robotic arm was not addressed in this project, but it is recommended to either use a microcontroller solution or a GPIO I2C controlled expansion board.

8.2. Mechanical design

It is recommended to use the suggested joint solutions in the final design of the arm, whilst also exploring into different motors and joint designs for each join to optimize the design. Further consideration must be taken on the final design to account for the fitting of electrical components. It is also recommended to keep in mind the stability of the design at all times.

8.3. FOC implementation

The exported VHDL file must be validated, and the PI controllers used within MATLAB Simulink must be tuned. It is recommended to acquire System Identification Toolbox from MathWorks to accomplish this task. From the alternatives explored, it is recommended to use the EVALKIT-ROBOT-1 from STMicroelectronics.

8.4. Simulation

The Simulation and GUI show promise in its progress, however, they are still incomplete. Future work is required in the simulation in the form of better sensor support to fully achieve the vision of a digital twin for the imported robot. Further polish will be required in the bridge between ROS and Move-it to unsure a better simulation experience. The GUI is incomplete and still in development, but even after the current proof of concept is realized, another project dedicated to this component will be required to fully achieve the vision desired.

9. References

ARMinor 2020 P1. (n.d.). *ARMinor-2020-Maniputaltor-with-FOC-and-FPGA*. Retrieved from GitHub.

Gonçalves, L. (2018). *What is SCRUM methodology, everything you need to know.* Retrieved from Luis-Goncalves.

igus. (n.d.). igus data sheet robolink D rotary axis asymmetric. igus.

Mathworks. (n.d.). *System Identification Toolbox*. Retrieved from Mathworks: https://nl.mathworks.com/products/sysid.html

MCP602SN datasheet. (n.d.). Retrieved July 1, 2020, from http://ww1.microchip.com/downloads/en/DeviceDoc/21314g.pdf

Mean Well. (2011, August). *RS-35-5 Datasheet.* Retrieved July 1, 20, from http://www.farnell.com/datasheets/2286956.pdf

Mean Well. (2019, July 29). *RSP-750-48 Datasheet.* Retrieved July 2020, 1, from https://static6.arrow.com/aropdfconversion/4d6f3e8da53163c515bb68ac2a8bdc66b79f3d7e/rsp-750-spec.pdf

Performance Motion Devices, Inc. (n.d.). *Magellan ® Motion Control IC*. Retrieved from Mallegan: https://info.pmdcorp.com/hubfs/Resources/Documentation/Magellan/magellan-motion-control-ic-users-guide.pdf?hsLang=en

RMS. (2017). *Mechatronics*. Retrieved from RMS Acoustics & Mechatronics.

ROS Org. (2017, October 23). *rosbridge_suite*. Retrieved from ROS.org: http://wiki.ros.org/rosbridge_suite

ROSSharp. (2019, December 20). *ROSSharp -README*. Retrieved from GitHub: https://github.com/siemens/ros-sharp/blob/master/README.md

ST. (n.d.). *BUZ10 Datasheet.* Retrieved July 1, 2020, from https://nl.mouser.com/datasheet/2/389/stmicroelectronics_cd00000673-1204298.pdf

STMicroelectronics. (n.d.). *Compact reference design kit for robotics and automation based on STSPIN32F0A*. Retrieved from EVALKIT-ROBOT-1 Data brief: https://www.st.com/resource/en/data_brief/evalkit-robot-1.pdf

TI. (n.d.). *ADC0820-N datasheet.* Retrieved July 1, 2020, from https://www.ti.com/lit/ds/symlink/adc0820-n.pdf?ts=1593611581772

Trinamic. (n.d.). *DC Motor Control with TMC4671*. Retrieved from Trinamic.com: https://www.trinamic.com/fileadmin/assets/Support/Appnotes/AN025-DC_Motor_Control_with_TMC4671.pdf

Xilinx. (2007, August 21). *Spartan 3A development board schematic*. Retrieved from https://www.xilinx.com/support/documentation/boards_and_kits/s3astarter_schematic.pdf

Xilinx. (n.d.). *Spartan-3A FPGA Family:.* Retrieved July 1, 2020, from http://www.farnell.com/datasheets/1911287.pdf

Appendices

Appendix 1 : Design requirements 31
Appendix 2 : Additional FOC subsystems 32

Appendix 1: Design requirements

The MoSCoW method was used to draw requirements and assign a priority to the requirements based on the desires and vision of the client. In Table 3 the technical requirements will be listed.

RX = requirement X MH = Must have SH = Should Have NH = Nice to have

Table 3: The table of requirements, split into ROS, Cost, Safety, Size, Production and Design requirements.

Name	Requirement	Priority
ROS Requirements		
R1.1	Packages for navigation, manipulation and simulation must be provided	MH
R1.2	The software must approach real time control as much as possible.	MH
Cost Requirements		
R2	The production cost of the arm must be under 1500 EUR in the context for mass	MH
R3.1	The movement flow of the robot must reasonably smooth, without jittering or mayor vibrations.	MH
R3.2	The precision of the arm must be within 1mm.	SH
Safety Requirements		
R4	The robot must have an emergency stop system.	MH
Size Requirements		
R5.1	The arm must be less than 710 mm in size (extended arm length).	MH
R5.2	The arm should be as small as possible.	SH
Production requirements		
R6.1	The arm should be producible with the facilities within Fontys using materials accessible to students.	MH
R6.2	There should be no visible cables or parts on the finalized design.	MH
R6.3	The project files must be open source.	MH
Design requirements		
R7.1	The arm must be able to move freely within 6 degrees of freedom.	MH
R7.2	An FPGA must be used.	MH
R7.3	The FPGA should control the motors of the arm using field-oriented control.	MH
R7.4	The implementation of kinematics must be ROS independent.	MH
R7.5	A raspberry pi4 must be used (with the possibility to upgrade if the need is justified).	MH
R7.6	Kinematics should be implemented within the low-level hardware.	SH
R7.7	Brushless DC motors or AC asynchronous must be used for the design and prototyping of the arm.	MH
R7.8	The arm must be able to carry a payload of 0.200g (excluding the gripper).	MH
R7.9	The end mount designed must be able to host sensors or end effectors.	MH
R7.10	The materials used should be a combination of aluminium extrusions, laser cut pieces and 3D printed pieces.	MH
R7.11	The arm should feature the colour palette of Robohub.	NH

Appendix 2: Additional FOC subsystems

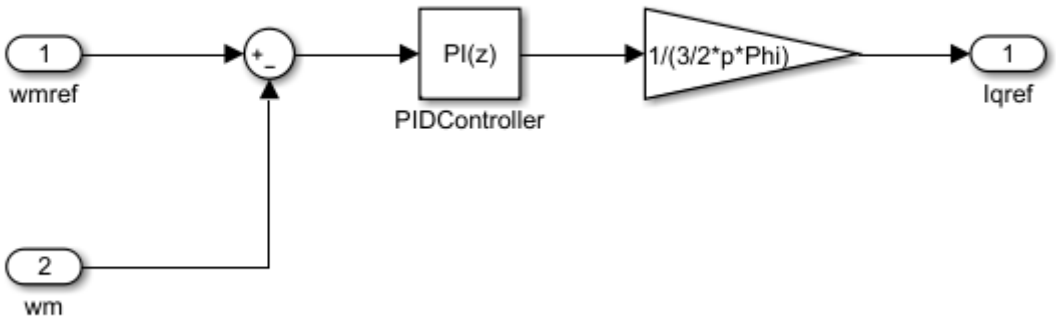

Figure 26: Velocity controller block inside the FOC subsystem.

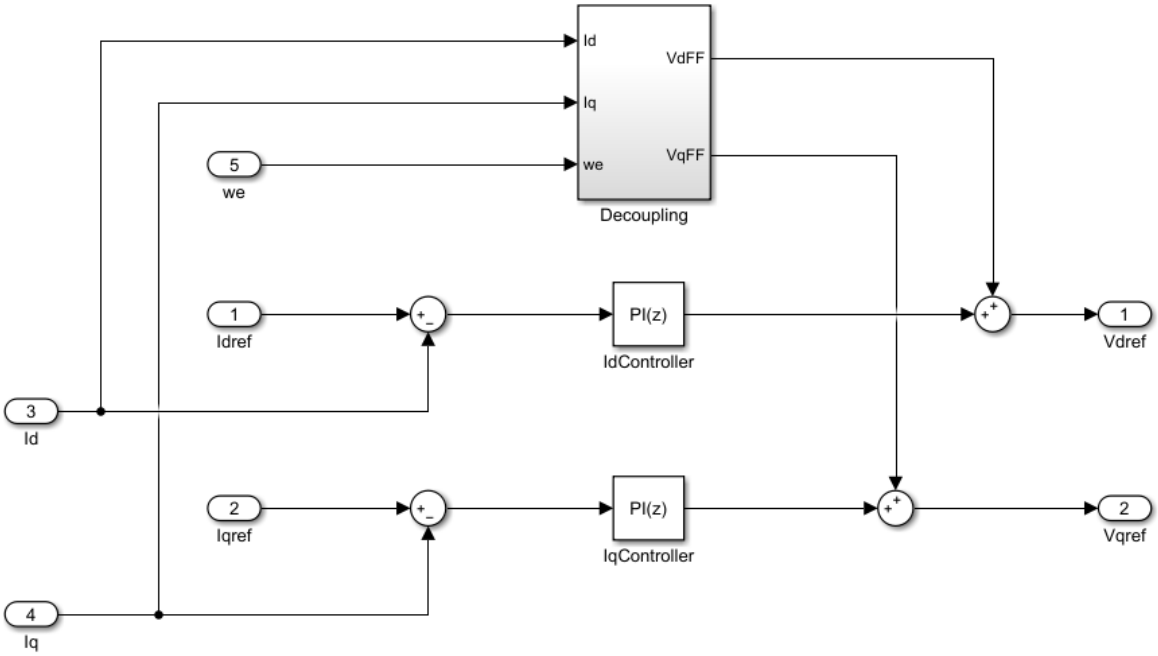

Figure 27: Current controller block inside FOC subsystem.