Coventry University

MASTER OF SCIENCE BY RESEARCH

An investigation into a novel landmine disabling manipulator

Grindley, Josef

Award date:
2016

Awarding institution:
Coventry University

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of this thesis for personal non-commercial research or study
- This thesis cannot be reproduced or quoted extensively from without first obtaining permission from the copyright holder(s)
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 15. Jul. 2025

An Investigation into a Novel Landmine Disabling Manipulator

A DISSERTATION SUBMITTED TO COVENTRY UNIVERSITY FOR THE DEGREE OF MASTER OF RESEARCH IN THE FACULTY OF ENGINEERING, ENVIRONMENT & COMPUTING

By

Mr Josef E GRINDLEY

Supervisor – **Dr Andrew Jason TICKLE**

September 2015

Abstract 1
Chapter 1 -Introduction 2
Overview 2
Motivation 2
Objectives 3
Original Contribution 3
Structure of thesis 4
Chapter 2 - Literature Review 5
Analysis of the problem 5
Current and contemporary technology 6
Analysis of target landmines 7
M14 anti-personnel mine 8
Type72A & Type72B anti-personnel mines 9
PROM-1 Anti-personnel mine 10
Analysis of tools and sensors the manipulator must possess 13
Implementation Strategy 15
Where to and how to use it 16
Chapter 3 - Outline of approach 17
Render safe strategy 17
Epoxy characteristics 17
Apparatus used 17
Method 17
Observation 18
Conclusion 18
M14 18
Type72A & Type72B 19
PROM-1 24
Conceptual design 26
Chapter 4 - Catia design and mechanical drive 27
Manipulator components 27
Structure 27
Drill 33
Air nozzle 39

Epoxy injector nozzle 45
Drive 46
Hydraulic 46
Pneumatics 46
Electrical 48
Discussion as to how the joints could be actuated 49
Joint 1 49
Joints 2 & 3 50
Joint 4 51
Joints 5 52
Table of recommended actuators 52
Chapter 5 - Electronics, FPGA sensor interfacing 53
Custom sensing solutions 55
Capacitance 56
IR reflectance sensors 56
Simulated hardware on FPGA 57
Pulse duration counter 57
CCD Reader 63
Wishbone interface and ZPU interaction 73
Chapter 6 - Kinematics and System Integration 75
Choice of integration system 75
Microsoft Robotics Developer Studio 75
V-Rep 76
ROS (Robot operating system) 76
ROS 76
ROS operation 76
ROS Redundancy 77
ROS Serial protocol 77
ROS Industrial 78
Importing design into ROS 79
Overall ROS integration 80
ROS Integration with Human interface devices 80
Kinematics and path planning 81
Chapter 7 - Conclusions and further work 82

Conclusion 82
Achievements and key contributions to the area 82
Weak points 84
Unresolved issues 84
Future work 85
Reference 85
Appendix 89
HDL Code 89
Wishbone counter 89
CCD Reader 93
CCD Reader Wishbone interface 93
CCD control latch 98
CCD reader 99
Vector logic converter 105
Optical encoder 106
Slit Disk 108

Abstract

In the field of land mine clearance there are two clear distinctions: civilian and military demining. The objectives, techniques and budgets vary substantially between them (McGrath, 2000). The substantially larger budget available for military mine clearance steers much research and development in this field. In contrast, civilian mine clearance generally sees little research and development specific to its organisational needs.

The objectives of military organisations are to clear safe routes through an area in the shortest time possible with the least casualties, this accepts however that some of the mines may remain active and result in casualties during the process (McGrath, 2000). Civilian demining however tends to have significantly different objectives; these are to completely clear an area of landmines and return the land to functional use.

Anti-personnel mines are designed to be inherently stable over time. However due to exposure to environmental conditions they can degrade. The result of such weathering varies between landmines, with some becoming inactive but others can become more unstable. This problem is exacerbated in civilian mine clearance, as this is generally done primarily by hand and not through mechanised means, placing the deminer at an elevated risk.

It's been claimed by Colin King that technological and mechanical solutions have been most use in the least number of situations. This implies that in order to make an impact in the area the problem must be thoroughly investigated. From the onset the project took a broad scope and started by investigating the operational needs of civilian mine clearance organisations. A large number of different land mine designs exist and so to keep the scope of the project achievable a set of three anti-personnel mines were chosen to be analysed. They were the most common, most likely to become unstable with time and most likely to remain active for the longest time. Contemporary strategies and technology used to remove them were analysed, this focused on the failings of mechanical clearance methods. The investigation then moved on to describe a robotic solution with a set of specific tools to be used to aid the removal process.

A mechanical manipulator design is presented as well as a set of proposed tools to tackle suspected devices. A selection of sensors used to produce odometry data where explored and some solutions demonstrated. Control of the manipulator was also explored as well as means of actuating the device. The projects overall contribution to the area of mine clearance is a presentation of a proposed technique in and a proposal to develop a robotic manipulator with a set of tools for the task. Due to the expansive nature of the field of study and time constraints a demonstration of concept model was not produced. However a set of recommendations for further work may with adequate time and resources, enable the concept to be fully realised.

Chapter 1 -Introduction

Overview

There is no doubt that remnants of war are a blight on the development of a nation particularly when found in areas of economic and social concern. Such examples include inhabited areas and agricultural or mineral rich land. Remnants of war take many forms but very commonly land mines. This prompted the motivation behind the research question posed. What are the limitations of current demining equipment and can those be tackled with a novel robotic solution?

A broad approach was required to answer this question. The first step was to gain an understanding of the demining process and the organisations conducting demining efforts. Then to choose which devices to concentrate on and finally come up with a solution to deal with the chosen devices. While undertaking an initial study of the demining industry, it was found that the money spent developing defence equipment greatly outweighed the money and effort spent making post conflict sites safe, in particular sites containing land mines. This project set out to first gain an understanding of how remnants of conflict, namely land mines, operate mechanically or electrically. The project then investigates how current robotic solutions are employed to deal with such devices and suggests a design and technique that could perhaps be more effective. Upon further investigation it was found that much of the technology used in the sphere of mine clearance had remained much the same, having built up in small increments with few major breakthroughs since their first appearance in the early 20th century.

Many of the technological solutions available were found to be prohibitively expensive, designed primarily for military use and only useful in the least number of situations (McGrath, 2000).

The most reliable methods of demining are manual methods that require a person to detect, uncover and remove a device, as opposed to mechanical methods that typically use heavy machinery and brute force to find and destroy mines. When undertaking this project, it was necessary to identify a means of comparing a proposed solution to an existing one. Early on it was decided that attempting to design a large mechanical solution was neither feasible nor an area of research that could deliver a high impact (Mikulic, 2013). The reasoning behind this decision was that machinery designed by organisations such as Aardvark Ltd they're mature products and as stated above, are most useful in the least number of situations. Due to this, the scope of the research was focused on how human deminers operated and how a robotic solution could be made to assist them. Of particular interest was how a suspected land mine could be dealt with when it was found. As such, the most appropriate comparison would be to compare the operation of a human deminer to that of a robotic manipulator.

Motivation

Currently there is a lack of investment in humanitarian demining equipment; this is apparently due to the huge costs involved in R&D activities as a result most equipment produced is developed to meet the needs of military organisations rather than humanitarian demining organisations. This project intends to assess the needs of humanitarian organisations and propose a robotic solution that helps meet them. The research undertaken indicates that most technology designed for the purpose has been suited to military demining. This often consists of machinery with mounted flails, rollers, or tillers (King, 1996).

Historically, such equipment was mounted on the front of ex-military equipment such as tanks, which caused a range of problems, for example the destruction of weak roads, irrigation systems and bridges (McGrath, 2000). Other more modern adaptations of these vehicles have been very similar, with additions such as remote operation to protect the operator, which has benefits such as reduced mental stress for the operator (Mikulic, 2013). Modern adaptations of such technology however have been very similar to previous technology with a lack of major innovation.

An obvious drawback of mechanical demining is the range of terrain these machines are able to operate in, as such vehicles are rarely able to function on rough or mountainous terrain. In addition, mechanical demining teams have always been accompanied by human deminers, as machines are unable to guarantee the destruction of mines (McGrath, 2000). Landmines are often damaged but may remain active, possibly in a more volatile state and have to be dealt with by human deminers afterwards.

The huge scope of the project and the time available meant that only some part of the overall manipulator could be developed. The decision was made to focus research efforts on the needs of humanitarian demining teams and that critically assessing those needs and suggesting solutions to meet them was a key focus of this project. Particular aspects where concentrated on summarised as a technique by which the target mines could be disabled, an overall description of the design of the manipulator and the electronics as well as sensor interfaces used.

Objectives

1. Research the needs of humanitarian demining organisations, in terms of their procedures and the limitations of contemporary equipment.
2. Pick three land mines and study the mechanisms by which they operate in addition to methods and procedures involved in rendering them safe.
3. Research how a robotic manipulator could be designed to render the three target mines safe, outline its basic design and determine a means of actuating it.
4. Outline a means of unifying the whole system including odometry data from sensors, human interfacing devices, kinematics and overall system control.

Original Contribution

- The proposal of a device that can excavate a suspected landmine and disable is in situ without the use of explosives, while utilising commonly available off the shelf components at significantly reduced cost compared to conventional mechanical clearance solutions.
- A wishbone based softcore peripheral was written in VHDL that can interface with a Toshiba TCD1304AP Linear CCD reader that automatically reads lines of pixels and stores them in RAM. Thus simplifying the process of obtaining data off the linear CCD and operations on a supporting processor.
- The development of a technique in which adhesive is injected into a mines firing mechanism as a means of disabling it, the use of such a technique in combination with a robot is a new concept.
- The combination of using FPGA's with Robot Operating System by bridging a softcore ZPU processor running on the FPGA with Robot Operating System, allowing a developer to utilise the advanced libraries available in ROS while at the same time taking advantage of the flexible reconfigurable nature of an FPGA. It should be noted that time constrains prevented the ROS

nodes from being written, however the code that creates a node on the ZPU does complete correctly meaning its only one step from implementation.

Structure of thesis

This thesis had been structured into 7 different chapters which follow the order in which the research work was carried out. Neglecting chapter 1 these chapters include:

Chapter 2, Literature Review

This chapter discusses the nature of the demining industry and the motivation behind the two key types of organisation involved in it, these being humanitarian and military. It goes on to discuss the mechanical equipment utilised by those organisations and there limitations. It also discusses the geographical distribution of a set of land mines and the motivation behind choosing the proposed target mines.

Chapter 3, Outline of approach

The process by which a landmine is removed is referred to as a render safe strategy, in this chapter the proposed render safe strategy is discussed. An analysis of the specified target landmines and a suggestion as to how to implement the render safe strategy on each is detailed. A key part of the approach is an outline of the basic concept behind the manipulator design; this is introduced in this chapter.

Chapter 4, Catia design and mechanical drive

The mechanical design and actuation aspects of the project are reviewed in this chapter. Outlined is the structure of the manipulator, proposed tools and the means by which they could be used and constructed. This chapter also discusses the means by which the manipulator could be driven, this outlines the explored methods that could be used such as hydraulic, pneumatic and electrically driven actuation.

Chapter 5, Electronics, FPGA sensor interfacing

A means of integrating the whole systems electronics and sensors is discussed in this chapter. Robotic systems require accurate odometry data to make relevant control decisions; this chapter also explores the means by which this data could be obtained. A novel way of integrating the proposed sensors is explored with the use of an FPGA and a softcore processor, taking advantage of the reconfigurable nature of an FPGA and the libraries of available to the softcore processor.

Chapter 6, Kinematics and System Integration

Unified control and user interaction is discussed in this chapter, there are several robotics packages available that can be used to take user and odometry input and produce control responses for the actuators. This chapter also discusses higher level control and consideration for kinematics.

Chapter 7, Critical appraisal, Conclusions and further work

This chapter discusses the strong and weak points of the project and suggests what further work could be done to move onto a demonstration of concept phase of the project.

Chapter 2 - Literature Review

Analysis of the problem

Rae McGrath (McGrath, 2000) discusses the failings of current equipment and concludes that the technology industry is unlikely to produce a single technological fix for landmine disposal. He does mention however that there is potential for development in areas such as detection and excavation (McGrath, 2000). As such there is an opportunity to develop a device that is in principle centred on excavation and the rendering safe of anti-personnel mines while neglecting detection. A device may have a positive impact that is portable and easily operated while composed of several different tools for excavation and the rendering safe of landmines.

One of the difficulties demining faced by organisations is the standard way in which mines are disposed of by using a small explosive charge (McGrath, 2000). As the volatility of mines can be variable, especially over time (United States Department of State and Office of Weapons Removal and Abatement, 2009), coupled with the occasional use of anti-handling devices 'booby-traps' makes detonation in situ the most favoured technique of removal in the interests of safety. This has various drawbacks most evident when the mines have a high metallic content. Upon detonation, possibly hundreds of fragments are often scattered over a wide area and this can cause false positives in the quality assurance phase of a mine clearance operation. Detonating in situ therefore can complicate the quality assurance phase of a demining operation, and in some cases can lead to a whole area requiring a costly rechecking process at the expense of the demining organisation.

The funding available to military organisations for mine clearance equipment is also comparatively large compared that of humanitarian organisations (McGrath, 2000) and this is a particularly important point to consider when developing landmine clearance equipment. Developers of military equipment typically expect at least 10 times the price for new technology compared with existing equipment (McGrath, 2000).

It is evident from the reviewed literature that funding available for military demining constitutes the vast majority of investment into the research and development of new mine clearance technology. Hence clearance technology is designed for use in the military sphere as this would often yield the most profit. As a result, most of the technology available is designed to suit the needs of military rather than humanitarian organisations. Most military demining consists of breaching (McGrath, 2000) in which equipment is designed to make a corridor from one area to another in order to allow people to pass through relatively safely. Breaching is a key requirement for military organisations; hence most demining machinery is tailored to suit this process and its associated requirements. Although humanitarian organisations have used this equipment successfully, as discussed by McGrath, there are inherent limitations such as weight and accessibility to remote areas. In addition, the equipment is unable to guarantee 100% removal so human teams must accompany mechanical demining teams. Drawbacks provide an opportunity for other technology to be developed to assist in ways that traditional machinery has so far been unable to do; some such examples of contemporary clearance machinery are listed in the figures 1, 2 and 3.

Some materials have been removed from this thesis due to Third Party Copyright. Pages where material has been removed are clearly marked in the electronic version. The unabridged version of the thesis can be viewed at the Lanchester Library, Coventry University

Figure 1, Flail configuration (Aardvark UK, 2015)

Some materials have been removed from this thesis due to Third Party Copyright. Pages where material has been removed are clearly marked in the electronic version. The unabridged version of the thesis can be viewed at the Lanchester Library, Coventry University

Figure 2, Tiller configuration (Mikulic, 2013)

Figure 3, Roler configuration (Mikulic, 2013)

Current and contemporary technology

Comparing present robotic tools, mainly designed for bomb clearance, it appears that a well-established basic footprint is used; this is typically composed of a robot with wheels or tracks and an arm with a gripper at the end. There is some variation, but for humanitarian clearance purposes they have limited usefulness without additional components being tailored to the task in hand. The task is an important consideration. Conventional clearance robots need to be able to inspect a device then either place a demolition charge, or move it, depending on the situation. Unfortunately there is limited information available on the design considerations that are taken into account in conventional bomb clearance robot design. However some information on tools they commonly utilise is avaleble. One such example is the water disrupter which can be fitted onto many conventional ordinance clearance robots such as the Wheelbarrow robot. The water disrupter is made up of a small explosive charge that is used to propel a water payload into a device; this is intended to damage its triggering mechanism before it can fire, in doing so rendering the device safe(Chemring, 2015).

There are a several examples of mobile robot that can perform basic manipulation tasks with water disrupters, ceramic cutters and grippers but there is not much else (AZO Robotics, 2015; Remotec, 2015). Further examination of the drawbacks of such equipment in terms of portability, cost and usefulness in humanitarian operations suggests that manipulation was a suitable area to research. As an area of research excavation and removal has not been as heavily covered as areas such as

detection, although detection has the highest demand for further research and development it is an area that little impact could be made in within the time scale of this project. Due to these factors it was decided to pursue excavation and manipulation as a subject topic.

Figure 4, Dragon Runner 20 (DR-20) (QinetiQ, 2015)

In terms of portability the wheelbarrow robot Mk9 in figure 5 ranges from 330kg - 350kg, meaning it cannot be transported to an area that is inaccessible to vehicles. Although the Dragon runner 20 robot designed by QinetiQ (QinetiQ, 2015) in figure 4 and weighing 9.07kg is portable in general, clearance robots are not, implying that there is a need for lightweight clearance equipment (QinetiQ, 2015a). Having analysed the technology, it was apparent that several conventional bomb and ordinance clearance robots already exist as mature products it was decided to centre the project on complementing these robots with other tools.

Figure 5, Wheelbarrow Mk 9, (Remotec UK Ltd & Northrop Grumman International, 2015)

Analysis of target landmines

It is necessary to analyse some typical landmines to determine both the mechanics of their operation and their vulnerabilities. The motivation behind these choices was centred on three properties.

- Most common.
- Most dangerous and likely to become unstable over time.

- Most likely to remain operational for the longest time.

After some consideration and close scrutiny it was decided to focus on the following three mines that fit the above criteria respectively.

- Type72A & Type72B
- PROM1
- M14

From this, a set of standard tools for the manipulator can be determined in addition to producing render safe procedures for each mine.

M14 anti-personnel mine

It was most notable that the M14 (US antipersonnel) in figure 6 appeared to be very resilient to environmental effects, this also applies to the Vietnamese copy MN-79. As limitations in the scope of the literature available are apparent however, this may not be the case in all circumstances. For example, it is stated that the M14 samples that came from Jordan where many other mines showed little deterioration compared to mines of a different design found in Cambodia. No M14 samples where recovered in Cambodia however, which could lead to false conclusions of their operational life expectancy in humid climates (United States Department of State and Office of Weapons Removal and Abatement, 2009). Also (United States Department of State and Office of Weapons Removal and Abatement, 2009) suggests that the high clay content and moisture content of soil is a key indicator of the rate at which any buried mines are expected to deteriorate.

The M14 is made mostly of plastic and is very difficult to detect. Its design is relatively simple with a Belleville spring applying pressure to the firing plate, which in turn provides the necessary reaction force to keep the mine from detonating prior to being triggered.

It is vulnerable through the top however, which is made out of plastic. It would probably be safe to gently drill the top of the mine, as the operating pressure is 88-156N (9-16kg) (King, 1996), and then use the proposed method of injecting oil, then epoxy resin, to jam the firing mechanism. The render safe procedure describes the mine as disarmed only when the detonator is unscrewed from the bottom, hence the mine would be only disabled and not disarmed following such a procedure (King, 1996). Drilling the top and not the side is advisable as this would allow the epoxy to settle between the spring and the firing plate, ensuring no leakage. This will jam the firing pin with a view to allow the removal of the detonator, should the deminer decide to disarm it at a later stage.

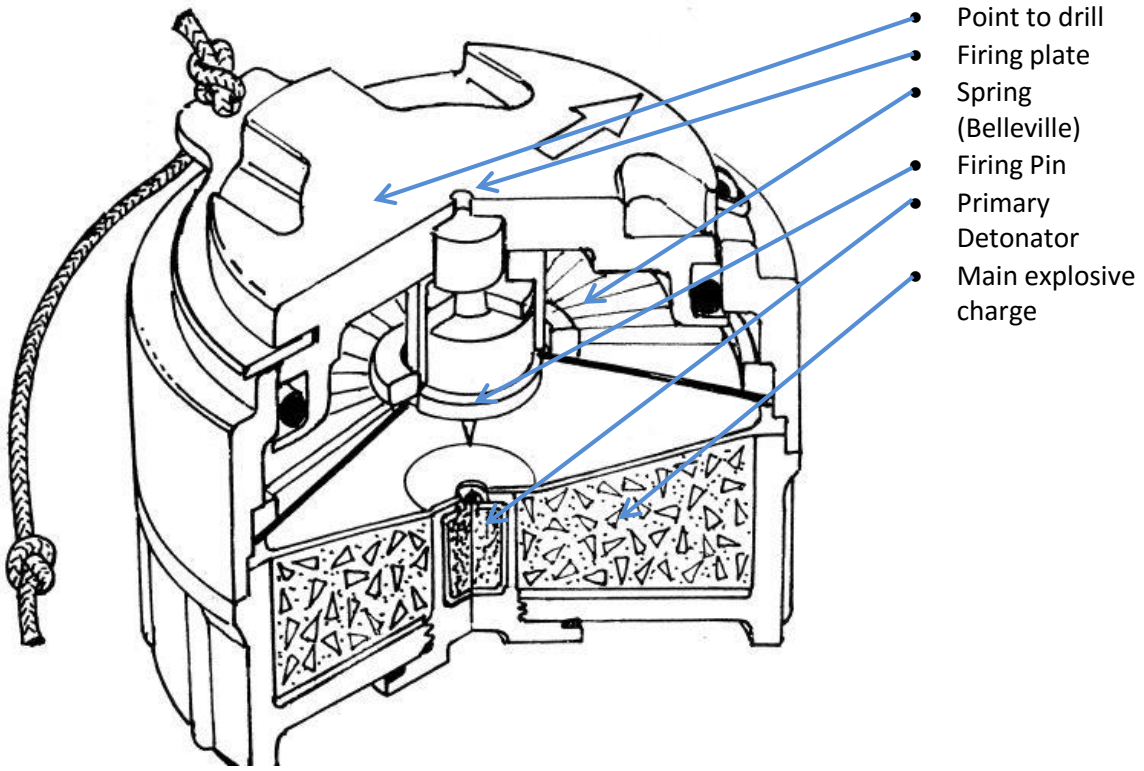

Figure 6, M14(US) MN-79 (Vietnamese copy) , (King, 1996)

Type72A & Type72B anti-personnel mines

The second target mine is the TType72 both A and B variations of the antipersonnel mine (note the illustration for version B version is located in figure 24) as it is so cheap and common. The Type72B is equipped with an anti-handling device which will detonate if tilted in excess of 10° (King, 1996). The information provided in shows that the Type72 is particularly prone to water damage where the firing pin often rusts as it is constructed of mild steel (United States Department of State and Office of Weapons Removal and Abatement, 2009).

The Type72A is made of plastic with an operating force of 45-92N (5-10kg), it is more susceptible to overpressure than the M14 mine, due to the large surface area of the firing plate. Its firing pin is located underneath the Belleville spring, drilling it through the top therefore may be less effective. A better suggestion would be to drill through the side, at the point where the tapping is located, and then inject oil followed by epoxy underneath the spring, preventing the firing pin contacting the detonator. Disarming the mine is possible only done when the detonator is unscrewed from the bottom, similar to the M14 (King, 1996). As the main detonator and booster are located below the area where the epoxy will settle and set, the Type72A can be still be fully disarmed afterwards. Some investigation has gone into how to tackle the Type72B located in figure 24 but it is inconclusive as yet, one suggestion is to drill the area where the battery would be located in an attempt to damage and discharge it and thus render the mine safe, but this cannot be conclusively done without testing a real device.

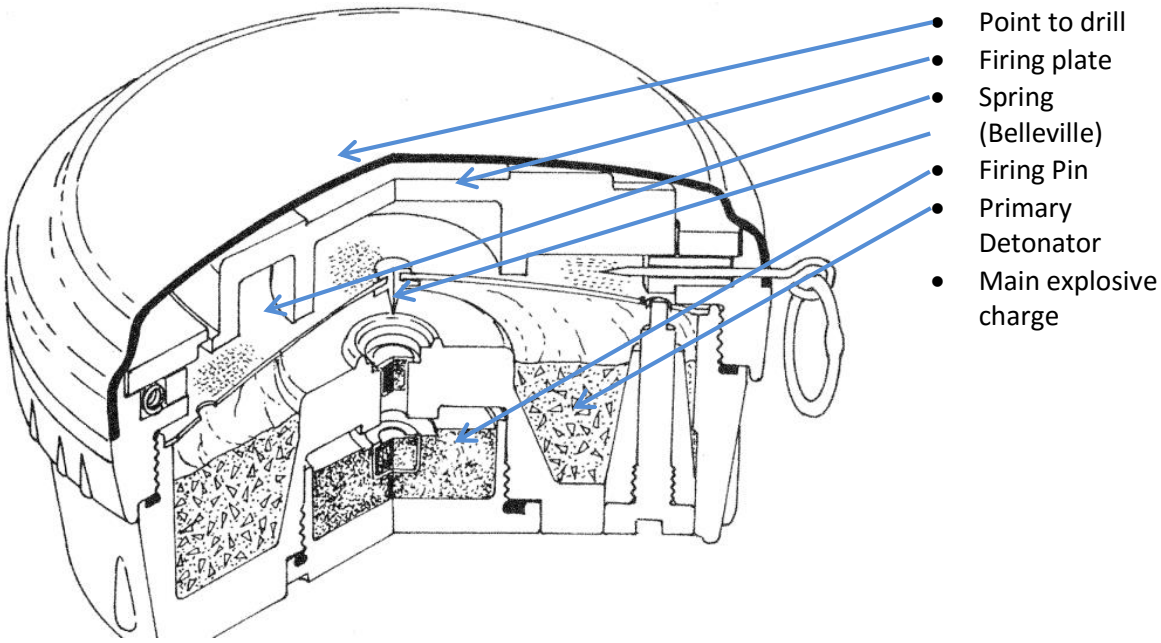

Figure 7, Type72Amechanical fuse (China), (King, 1996)

PROM-1 Anti-personnel mine

The third target mine was determined both by anecdotal evidence and qualitative analysis of information provided by deminers in the Balkans. There is also technical documentation (King, 1996). The PROM-1 (former Yugoslavian bounding anti-personnel mine) varies from the others mainly because it fires out of the ground and explodes at approximately head height, therefore it is dangerous for approximately 20 meters (Military Periscope, 2015). Anecdotal evidence from Balkan deminers has been discussed (United States Department of State and Office of Weapons Removal and Abatement, 2009) where the mine has been rumoured to trigger at lower pressures as it ages. This study largely discards that particular evidence as it is hard to prove, however this may be an oversight as it may not have been properly considered. King describes the operation of the PROM-1 in some detail including useful technical information such as weight, trigger pressure and a detailed diagram (King, 1996). The diagram shows the way in which the spring is responsible for forcing the firing pin into the detonator when triggered in addition to maintaining a constant reaction force against any pressure applied to the external trigger of the mine. This can be seen in figures 8-12. It is evident in the diagram that if the spring were to deteriorate, the reaction force would decrease as a result, implying that the deminers in the Balkans may well be correct about this mines' increased instability as it ages. As stated in the render safe procedure for this type of mine, it should be destroyed in situ. Disarming should not be attempted due to its highly volatile nature, though detonation this would result in the scattering of metal fragments (King, 1996).

According to King, the PROM-1 cannot be fully disarmed as its detonator is located inside the mine where it cannot be reached (King, 1996). The firing mechanism of the PROM-1 is relatively complex, figure 8 shows a scaled up version of the detonator. From the pictures in the literature and by using the size of a person's thumb in the image figure 10 as a reference, it was determined that the outer casing which is likely made out of steel was approximately 2mm thick. By having a rough guide to the dimensions of the components the detonator could be reconstructed to make analysis easier, this is illustrated in figures 11 &12.

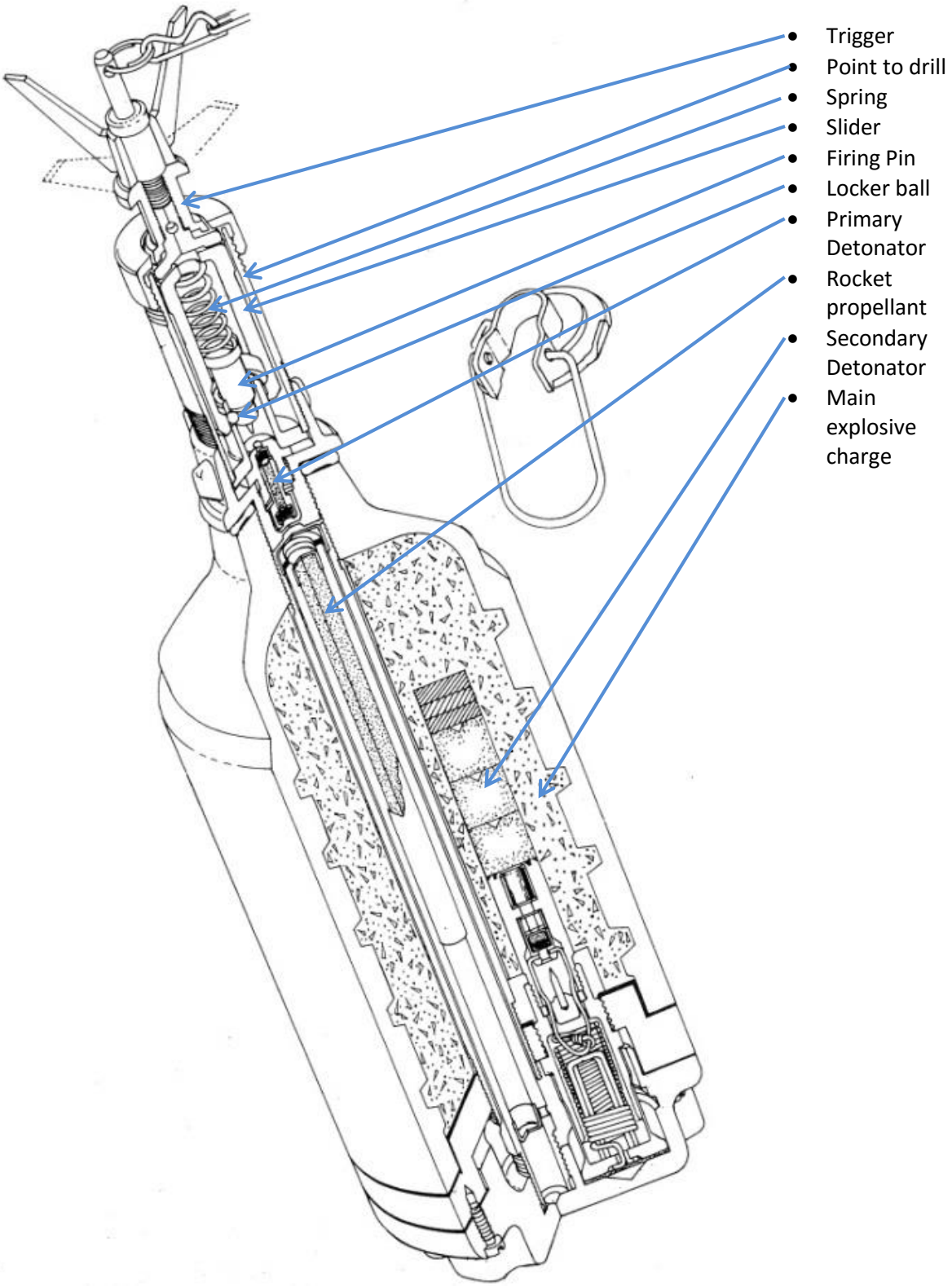

Figure 8, PROM-1 main body (Former Yugoslavia), (King, 1996)

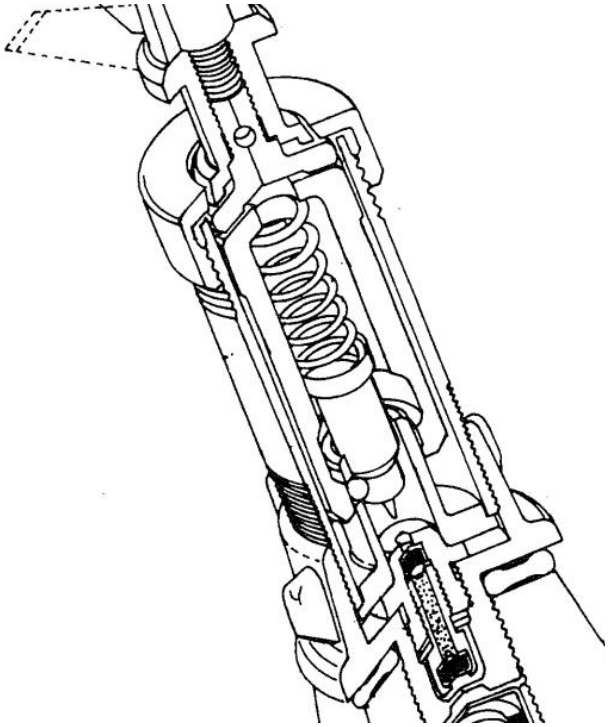

Figure 9, Scaled image of PROM-1 firing mechanism (King, 1996)

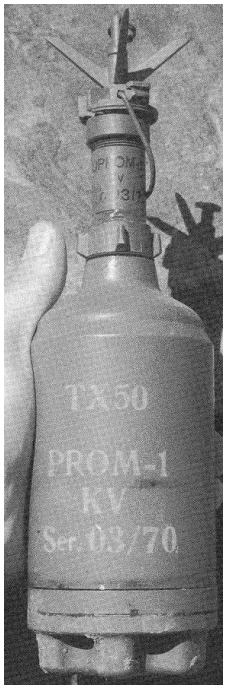

Figure 10, Photograph of PROM-1 (King, 1996)

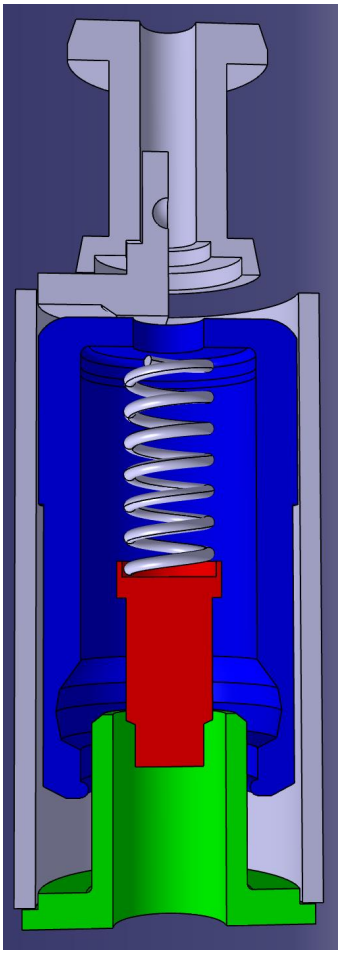

Figure 11, Illustration of PROM1 (Grindley, 2014)

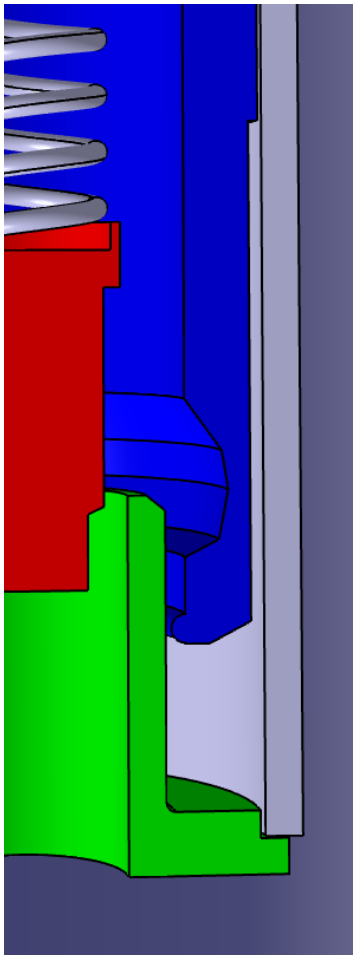

Figure 12, Illustration of PROM1 zoomed view of target area (Grindley, 2014)

Analysis of geographical distribution of target mines specified they can be found in the following countries (King, 1996)

1. M14 and MN-79 (copy)
    - Cambodia
    - Vietnam
2. Type-72
    - Angola
    - Cambodia
    - Kuwait
    - Mozambique
    - Somalia
3. PROM-1
    - Angola
    - Iraq
    - Namibia
    - Former Yugoslavia

The information provided in is approximately 16 years old, but despite its age, the information is reliable and it is unlikely that the distribution of these devices has changed greatly since publication (King, 1996). There may be an exception with Afghanistan however, where the distribution of black market arms may mean one or more of the three target mines may also be found.

When investigating the selection of the target mines, information on the degradation of some samples was available. Several different test cases on the degradation of land mine samples found in Jordan, Cambodia, Angola and the Falklands Islands are discussed in the (United States Department of State and Office of Weapons Removal and Abatement, 2009). The study indicates that the soils chemical properties such as acidity can have a significant effect on the aging of mines. The literature indicates that in Cambodia for example, the soil is more acidic than in Jordan. It illustrates cases in which landmines recovered from Jordan were in considerably better condition than those recovered in Cambodia. This information was useful in selecting the target mines, but when determining the type of tools the manipulator must possess, the density of the soil is of a greater concern than its acidity. The proposed manipulator must be able to operate in the countries listed, where the soil type ranges from the wet loosely packed, or clay like soil found in Cambodia, to the loose sand or arid, hard baked soil found in Angola or Afghanistan.

Analysis of tools and sensors the manipulator must possess

In most cases it would be necessary for the manipulator to dislodge soil fragments and transport them to an adjacent area, therefore a tool to dislodge and move earth is essential. A number of design considerations must be taken into account. This is usually done with a trowel similar to a standard garden tool (King, 1996). When operating a trowel on a robot however, the way in which the pressure data is obtained and used as feedback must be considered in order to prevent over pressure, which can cause the operator to unwittingly detonating the mine. To avoid this issue it was decided that such a tool was inappropriate for use on the manipulator. Another less abrasive option was needed. The proposed solution was to specially develop a vibrating wire brush to dislodge the earth, then to transport it with high pressure air. This is arguably the best option for excavation because the manipulator is far less likely to be put into a position where it can inadvertently trigger a mine through over pressure.

An air nozzle will be used to move the dislodged earth; which raises a number of questions as to how to provide high pressure air on demand while considering the device's weight constraints. Product searches have so far shown that a small diaphragm pump similar to that found in a mobile car tyre pump should be sufficient to provide the high pressure air needed. When used in conjunction with a small pressure tank, enough compressed air could be stored to allow short blasts to clear the dislodged earth. Alternatively, a rotary centrifugal pump or a ducted fan with an appropriate hose and nozzle could be more suitable for lower pressures or higher rates of flow requirements. The nozzle requires two degrees of freedom with respect to both pitch and yaw. This means the air nozzle could be operated when the rest of the manipulator is stationary, or it could operate independently when doing other tasks. To simplify the feedback, a red laser diode will be installed behind the nozzle and used to pinpoint its orientation and a camera will then be used to provide feedback to the operator and control.

The drill should use an air turbine from a dentist drill rather than an electric motor to turn the shaft, due mainly to the high rotation speed of the air turbine. It is necessary to keep any forced vibrations generated by the drill as far away as possible from the resonance frequency of any components in the firing mechanism, or in a higher mode. Though the resonance frequency of the components is unknown, it is suspected that they would tend to be a rather low frequency. The turbine chosen in this project is rated at 400,000 RPM (6666.7Hz) unloaded.

The proposed method of disabling the landmines involves using an abrasive rotating instrument to drill the mine in a specific location stipulated in the render safe procedures, then to inject mixed epoxy resin into the hole to jam the firing mechanism.

In order to move any device, a set of grippers will be needed. These can be simple, as other more complex tasks, such as uncovering and disabling the device, are done by other tools on the manipulator. Grasping the target mine or another object is key in the disposal process, however the means of grasping objects is well researched and documented and has been omitted from this project.

Another consideration is what actions should be taken when the mine has been excavated. From talking to an ex weapons disposal expert from the Royal Air Force (Mr Paul Moore), it was stressed that the most important thing to do after the mine has been uncovered is to identify it. It was made clear that without positive identification it would not be typically safe to proceed, as no render safe procedure could be determined. For this task, it is necessary to have cameras placed at specific locations on the manipulator allowing the operator to identify the device and also to provide feedback about the manipulator's position and orientation to the operator. As with grippers, installing cameras onto the manipulator would be a relatively straight forward task, especially as many of them are USB based. ROS (Robot Operating System) simplifies makes this task as it, has a huge repository of drivers for hardware support, so it was considered to be further work (Willow Garage, 2012).

It was also noted in the conversation that the process of providing feedback was considerably more complex than simply installing an array of cameras onto the manipulator and providing video feedback through a monitor. One of the issues raised in conversation with Mr Paul Moore was the way in which camera sensor data was fed back to the operator. The way in which cameras and other sensors integrate with each other is of paramount importance for the usability of the device because if done correctly, it may allow additional features to be added simply by modifying the software.

Several other types of sensor are available, of which the most promising device concerning this project is the Leap Motion. This is a human interface device which uses two IR cameras to produce stereo disparity; this is then used to produce very accurate distance data to within approximately 0.01mm. This is intended to be used as a human interface controller, allowing the user to have a very accurate

and intuitive method of controlling the actions of the manipulator. It is suggested that with the help of a specially designed pen, controlling the manipulator would be similar to using a paintbrush. 3D scanning of the environment around the device would be a useful means of providing an operator with feedback, this can be done with the use of 3D scanning devices similar to the Microsoft Kinect, though this is to be considered as further work in this project. The means of doing this was touched on but not in great detail, one suggested technique involves the use of point clouds generated using libraries available in Robot Operating System (ROS). Despite the increased complexity the major advantage of having the environment in front of the manipulator characterised in 3D would be allow the manipulators controller by virtue of collision detection libraries in ROS to automatically prevent the user from inadvertently touching the mine or surrounding area. This would allow for a greater level of safety when using the manipulator and would be considered a key feature.

Implementation Strategy

The project requires a number of sensors and actuators to be connected to one system as well as an intuitive means of commanding it to perform desired actions, furthermore its control must be also integrated into the same system. To add to the complexity, the proposed manipulator must make use of inverse kinematics to compute joint positions and feedback for its control system. In addition, image capture and processing capabilities are also needed as well as IP networking capabilities, as proposed; it may also be possible to implement SLAM which would allow far superior user feedback and potentially safer control.

Such a task is potentially huge if started afresh. Some investigation was therefore carried out into readily available systems that could deal with such large scale integration. To this end, the following systems where found.

- V-rep Robot simulator
- Microsoft Robotics Studio
- ROS (Robot Operating System)

Each system was closely examined, with pros and cons to each. V-rep for example (Coppelia Robotics, 2014) it an user friendly piece of software which supports many different programing languages from typical languages such as C/C++ and python to less ubiquitous ones like Luna script . It includes physics engines such as the Bullet, Vortex Dynamics, Newton Dynamics and ODE each is easy to switch between and invoke the simulation. It does however require a PC or Apple Mac to operate; this eliminates its usability in this project as the control system should ideally be located on the manipulator itself. V-rep can also integrate with ROS was concluded to be a good platform to develop and simulate robots in. However as a piece of software used as an overall controller handling odometry data, kinematics calculations and producing control responses in real time to hardware it was not suitable.

Microsoft Robot Developer Studio was discounted quickly for similar reasons, it requires a Windows based environment to operate. Though more recent versions of Windows can operate on ARM instruction sets the complexity in implementing it and proprietary limitations that come with Microsoft systems such as drivers and access to the IP stack lead to the software was discarded.

It was decided that Robot Operating System referred to in this document as ROS was the best choice. ROS is a thin client that runs on a publisher subscriber basis, originally designed to run on Ubuntu Linux. It consists of nodes and services, a node for example could be a piece of code that interfaces with a sensor and publishes the data to a control loop service, or another node that controls an actuator. The system itself allows for tasks to be modularised and if needed, distributed over a number of different systems including computers in other geographic locations with different

operating systems. The following is an example of a major advantage of using ROS. If image processing or a supervisory task needed more processing power than was available on the manipulators controller, the tasks could run as a service on another computer with minimal extra coding or configuration. Such is ROS's ability to modularise tasks. Many open source nodes and services have already been written to make the process of control and manipulator path planning far easier. Specific packages and libraries have been written in ROS for sensor integration, SLAM, open computer vision support, MoveIt (Willow Garage, 2012) for path planning and control through ROS industrial, to name just a few.

Where to and how to use it

As there are many other robots available that have tracks or wheels and an arm as illustrated in figure 28, redesigning such devices may not yield adequate novelty to justify the project, therefore analysing what current technology does not have would be more appropriate. Typically when robots approach a suspected device, an end effector is used to interact with it and the surrounding area, usually consisting of a gripper with two 'fingers'. The dexterity of such grippers does depend greatly on the control strategy implemented to utilise it. However should a robot such as the QinetiQ Dragon Runner 20 be used in landmine removal the grippers would be a key limiting factor. As such the proposed device should be able to be fitted to an existing robots arm as well as used with the proposed tripod design in figure 28, this could compliment the usefulness of current designs.

The tripod based design that can be found in figure 28 was influenced by human deminers as human deminers are the best suited for the task of demining. The pose and stance of a human deminer kneeling down and using their hands to inspect and disable a device should ideally be mimicked. The tripod design allows the centre of gravity to be shifted towards the rear of the robot allowing the manipulator to lean forward while maintaining stability. Another advantage is weight and portability, assuming the choice of drive and material used to build structure has adequate strength. It should be portable and could be carried into mountainous regions for example where conventional robots struggle to operate.

An additional advantage of using the manipulator compared to a human deminer is that it could be operated at night with a sufficient light source. The use of ROS makes the manipulator geographically independent as long it has internet connectivity. ROS works primarily through IP meaning it could potentially be operated from anywhere in the world (Willow Garage, 2012). This has another key advantage depending on how the demining team operate. Usually the area a demining team is working in is split up into sections of trenches, each deminer works in a trench. If team member discovers a suspected device they stop working, mark the location it and leave the trench for the team leader to inspect and deal with. This usually involves uncovering it and placing an explosive charge with a view to detonate it at end of the working day. Having the team leader uncover, inspect and place a demolition charge is delicate and potentially time consuming work. The proposed solution could operate at night allowing the team to carry on through the day without the need to stop could be very beneficial.

Chapter 3 - Outline of approach

Render safe strategy

The proposed technique of disabling the mines is to drill a hole in a specific location on the firing mechanism, then to inject light oil into it. The purpose of the oil is to fill the cavities in the firing mechanism and eliminate any air bubbles, as these may impede the flow of the epoxy mixture into the firing mechanism. Another function of the oil is to impede any mechanical movement related to the device being triggered. A chamber filled with oil would be expected to act like a piston filled with oil, this helps to make doubly sure the mine does not trigger when it is being worked on. Finally, the cavity would be then injected with epoxy resin which is denser than the oil, so it settles at the bottom of the firing mechanism. Once the epoxy is set, the mine should be safe to remove, assuming no anti-handling device is present.

Epoxy characteristics

Within the time constraints of the project the characteristics of the epoxy to be used were not extensively researched. The proposed technique was to inject oil prior to the epoxy and then displace the oil with the epoxy resin. This is done under the assumption that the epoxy resin would settle at the bottom of a firing mechanism and jam it. Initially an experiment was conducted to see if the method would be effective. The main concern was that the epoxy would mix with the oil and would not displace the oil as anticipated. An experiment was carried out to determine if this was the case and that whether or not the epoxy would set.

Apparatus used

- Epoxy resin listed as (bisphenol-A-(epichlorhydrin) and Buty 2,3-epoxypropyl ether).
- Plastic tube with small hole cut in the side approximately 3mm diameter.
- Standard plastic syringe.
- Light bicycle oil.

Method

1. Cut small hole approximately 3mm in diameter into the lower side of the plastic container.
2. Add oil into plastic container, until it is almost past the hole and replace the red top, seen in figure 13.
3. Pull plunger out of the syringe and add both the epoxy resin and hardener.
4. Briefly mix the epoxy resin and hardener and replace the plunger.
5. Turn syringe with the injecting nozzle facing upwards, allow the epoxy to settle at the plunger's side, seen in figure 14.
6. Press plunger to force air out of syringe.
7. Inject epoxy resin through the hole in the container, seen in figure 15.
8. Observe how the epoxy resin interacts with the oil, this is illustrated in figure 16.
9. Leave standing for 30 minutes, as advised by the instructions shown in figure 17.

Figure 13, Light oil in container

Figure 14, Mixed epoxy resin in syringe

Figure 15, Epoxy resin being injected into container

Figure 16, Epoxy resin and oil in container

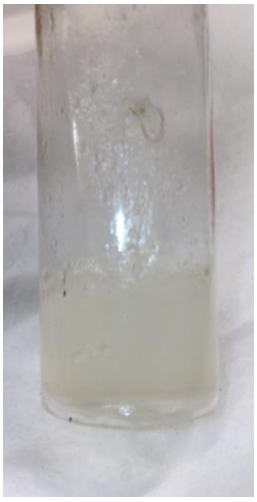

Figure 17, Epoxy resin after 16 hours

Observation

The epoxy resin quickly settled at the bottom and did not appear to mix with the oil. It is clear from figure 16 that there is a small amount of oil underneath the set epoxy resin. After 30 minutes the oil was removed from the container and the state of the epoxy resin was examined. It was found to be quite plastic in nature and deformed easily when inspected with a stirring implement.

Conclusion

The epoxy was affected by the presence of the oil and did not harden as it would do under normal circumstances within the 30 minutes, as specified by the instructions. However it is likely that it in this state though it is not fully hardened and has some compressibility, it would likely prevent the firing pin of a mine from contacting the detonator or at least with any notable velocity. More research needs to be done on the type epoxy and lubricant used prior to the injection of the epoxy. Such as determining if the epoxy should be oil or water based and if a more suitable lubricant could be used that did not adversely affect the strength of the hardened epoxy, or indeed a different adhesive altogether. It should be noted that air bubbles were present in the mixed epoxy resin, allowing them to escape before the resin began to harden was not feasible due to the rate at which it hardens this may have affected the stiffness of the hardened resin. The instructions state that it should be left to set up to 16 hours to fully harden seen in figure 17, this was done and there were some differences in its properties but it was not substantial. In conclusion this technique would likely work as a method of jamming the firing mechanism of a land mine, however further work must be done on the specifics of the epoxy resin and lubricant used.

M14

The Type14 mine, similar to the Type72A, has a Belleville spring to push the firing pin into the detonator when triggered and two cavities above and below the spring (King, 1996). Unlike the Type72A however, there are no holes in the spring that would allow the oil and epoxy to transit from the top chamber to the bottom. Visible in figure 18, access to the bottom cavity is particularly difficult with this type of mine, as the base of the spring is located just above the explosive charge making it very difficult to access. The firing pin is held by the Belleville spring and cannot fire without its movement. The outer rim of the firing plate (top section) is protected by an o-ring that prevents water ingress and directly below that, there is a lip that holds the o-ring in place below which there is the

top cavity then finally the spring. If the space between the spring and lip beneath the o-ring was to be filled with hardened epoxy then the mine would be unable to fire. As such, the proposed method of disabling the firing mechanism would be to drill, the firing plate half way between the centre and the side and inject oil then epoxy resin. This method is an outline of the proposed technique however until it is tried in practice, the finer details cannot be determined. One such unknown is whether or not the oil will work its way past the firing pin and settle below the Belleville spring, or if disabling the M14 this way requires oil to be injected at all.

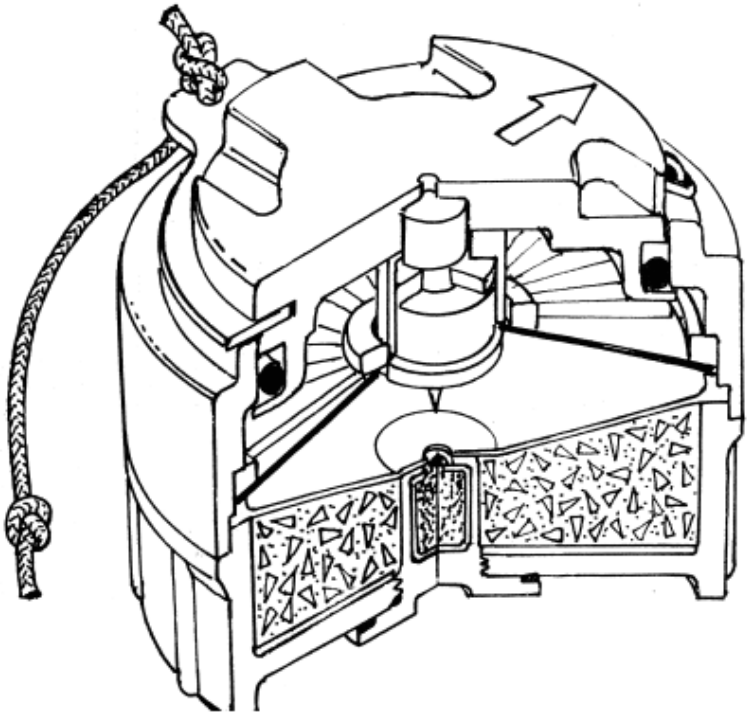

Figure 18, Type 14, (King, 1996)

Type72A & Type72B

The Type72A is mechanically operated and mostly composed of plastic, the Belleville spring is one of the few metal components that make up the mine (King, 1996). Figure 12 shows that there are two chambers above and below the spring, with the detonator housed in the lower of the two. It is desirable that the lower chamber be filled with epoxy to prevent the mine triggering. The Belleville spring can be seen in figure 20 (United States Department of State and Office of Weapons Removal and Abatement, 2009) which also shows the spring from a dismantled mine. There are three clearly visible holes that could be utilised to allow the oil and epoxy resin to transit from the top to the bottom chamber with ease. The proposed render safe procedure for this mine therefore is to drill the plastic top of the mine and first inject oil to fill the air cavities, wait for a given time then inject the epoxy. Again how long it will take for the oil to transit from the point it is injected and fill the cavity below is unknown, the delay in between injecting oil and then the mixed epoxy should be determined by experimentation.

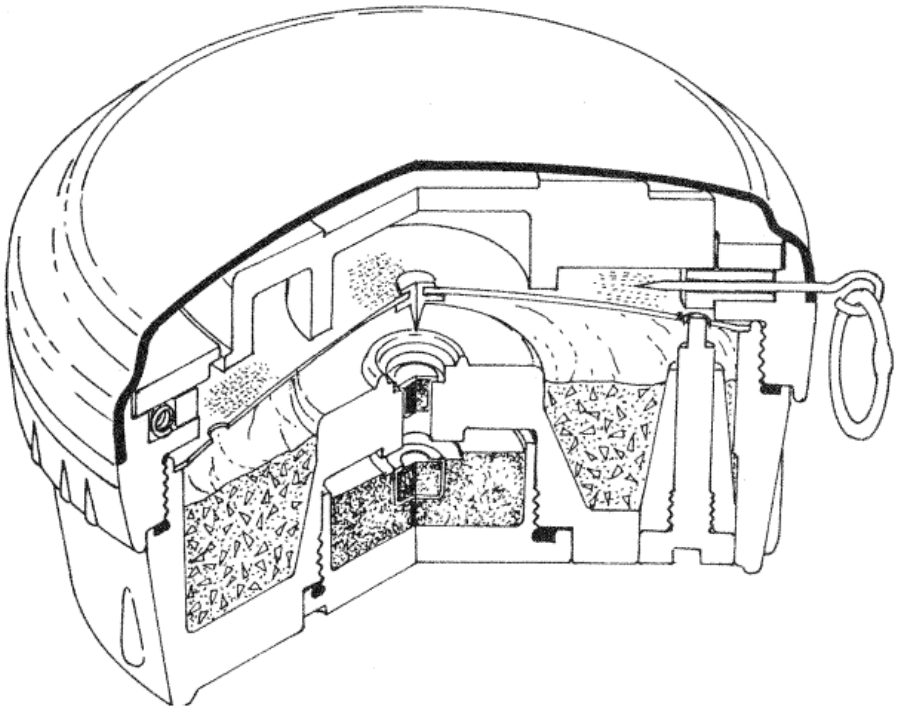

Figure 19, Type72A, (King, 1996)

Figure 20 Type72A disassembled (United States Department of State and Office of Weapons Removal and Abatement, 2009)

The Type72B is the electronically fired version of the Type72A in figure 24. According to (King, 1996) it can by triggered by either tilting the device through an angle more than 10°, though according to (De Wolf, 2008) applying a sudden acceleration to the mine will also trigger it.

The operation of the Type72b is described in detail in (De Wolf, 2008), as well as details of components used in its makeup. Figure 21 also shows a schematic of its operation (De Wolf, 2008) also describes its mode of operation. Unlike mechanically fired mines the Type72B is triggered by a tilt switch that closes if the device is tilted beyond 10deg of its vertical position at rest, it is also activated by sudden acceleration. Although (De Wolf, 2008) provides an adequate description of its operation, the circuit was simulated in Simulink using the Simscape toolbox to confirm the papers assertions and better understand the device. The simulation can be seen in figure 22 and its results in figure 23. The electronics in the circuit comprise of 4 NAND gates 3 of which are configured as inverters and 2 NPN bipolar junctions cascaded together for higher amplification. As described in (De Wolf, 2008) the mine has effectively two switches featured in its design. The trigger that is a tilt switch described above and the arming switch that is closed when the arming pin is removed this is not featured in figure 21, though has been included in the Simulink simulation. The replicated circuit was made to be as close to the Simulink simulation described (De Wolf, 2008), the same labels and naming conventions were also used.

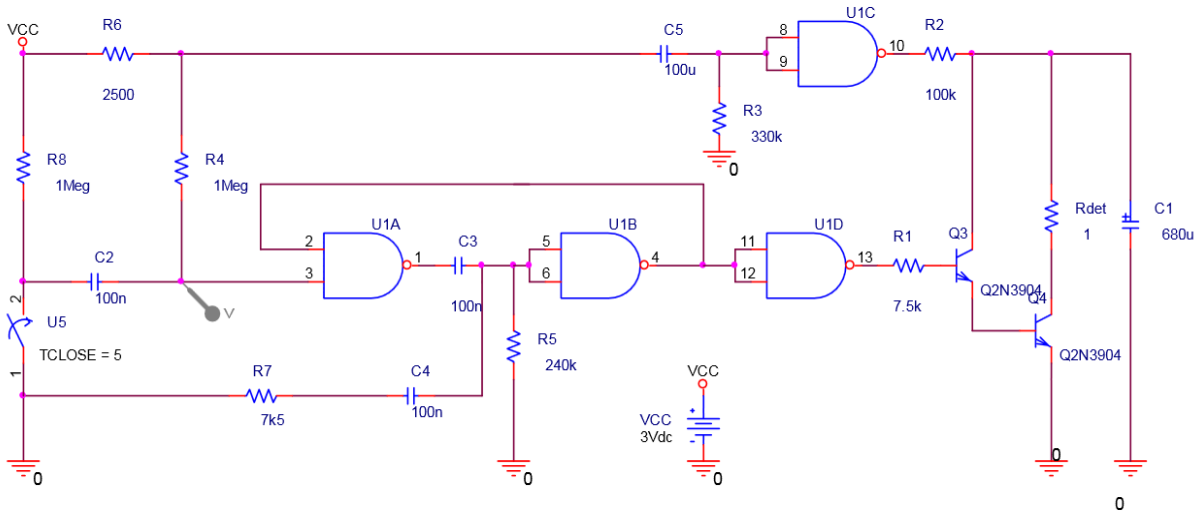

Figure 21, Schematic of operating circuit for Type72B (De Wolf, 2008)

Note that the cascaded bipolar NPN transistors Q3 & Q4 are responsible for detonation of the mine. The capacitor marked C1 in figure 21 has been renamed as C_{det} for clarity in figure 22, is responsible for holding the charge required to fire R_{det} the detonator that can be seen in both figure 21 and 22. If the input of Q3 is high at any time R_{det} will be triggered, assuming that C_{det} is charged this allows for a safety feature to be added.

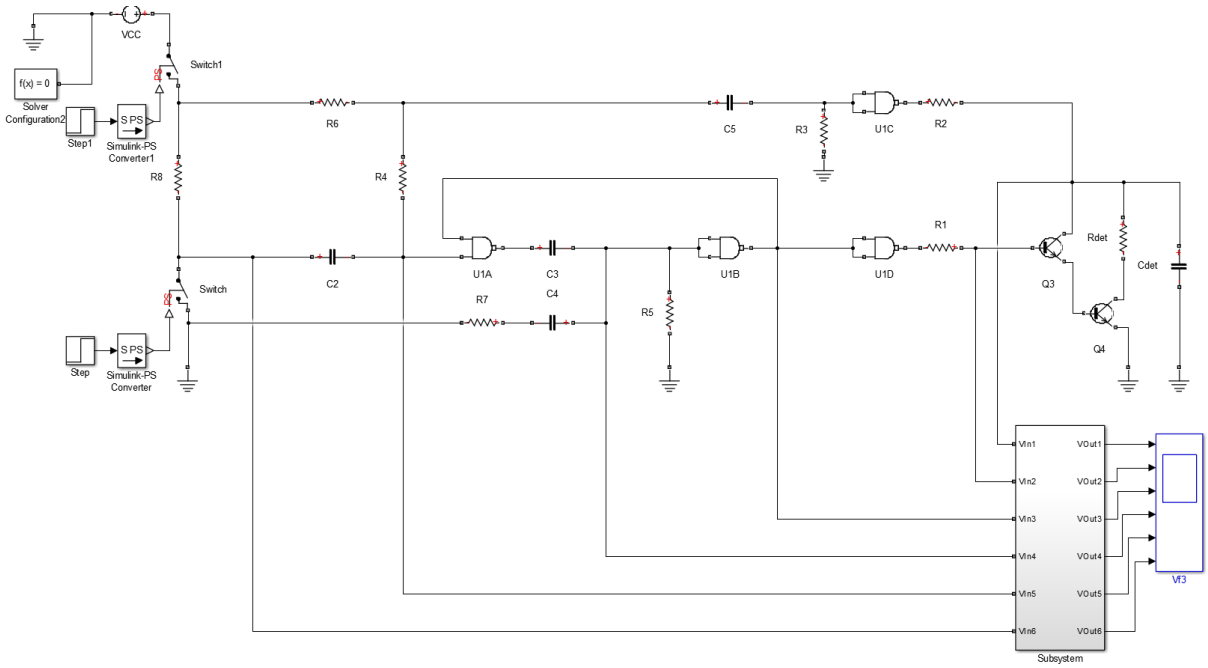

Figure 22, Simulink simulation of Type72B operation (Grindley, 2015)

The simulation in figure 22 was setup to run for 270 seconds with two triggers the first at 3 seconds which simulates the firing pins removal. This first event can be seen in scope 1 of 6 in figure 23 which represents the voltage between C_{det} and ground, as the firing pins removal is actioned the voltage signal drops slightly and begins to rise after approximately 30 seconds then continues to rise exponentially. This delay is due to RC constants in the circuit and designed to prevent the person laying the mine from immediately triggering it after removing the triggering pin. This safe initialisation of the device is controlled by capacitor C5 and inverter U1C as well as resistors R3, R2 and R6. When the safety pin is removed C5 begins to charge initially allowing current to pass which creates a voltage drop across R3, as UC1 is configured as an inverter its output is held low which prevents Cdet from charging. This happens until C5 charges and the voltage drop across R3 falls below the threshold of the inverter UC1. Once this occurs the output of UC1 becomes high and Cdet begins to charge putting the mine into an armed state. From the Simulink simulation is was found that Cdet is almost fully charged at approximately 350 seconds making its RC time constant approximately 87.5 seconds. As such 3 time constants was 262.5 seconds so 270 seconds was chosen as the simulation run time and the second instance that simulates the trigger was set at 265 seconds.

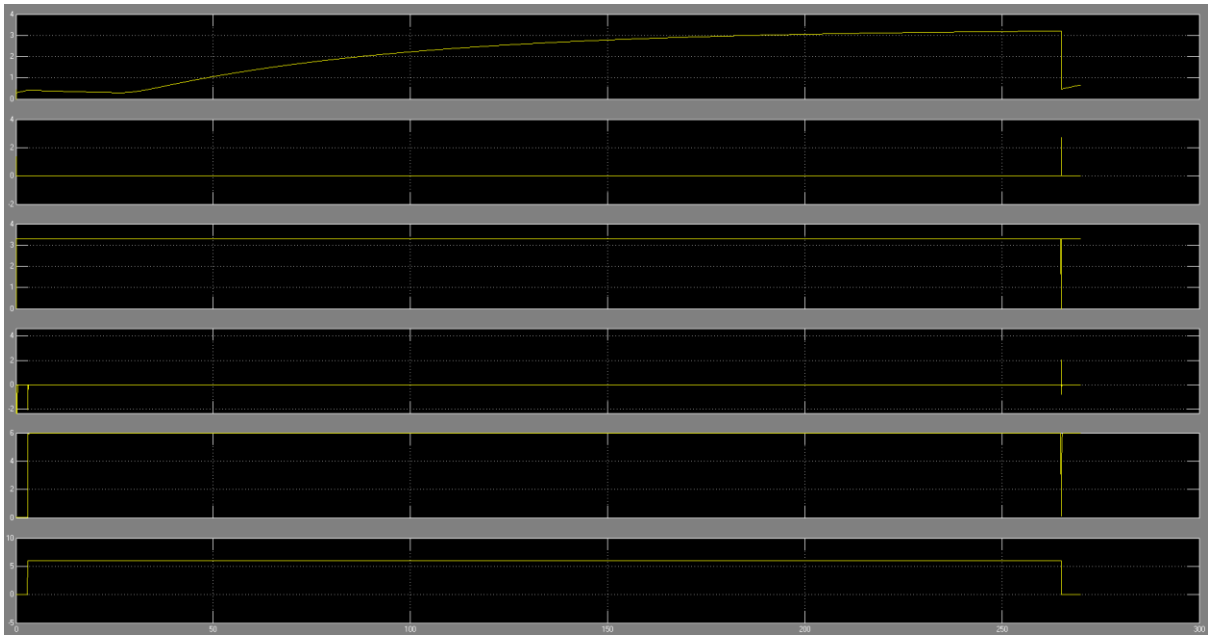

Figure 23, Simulation of Type72B (Grindley, 2015)

The part of the circuit responsible for triggering Q3 is effectively a mono stable configuration. In its armed but stable state the output of U1D seen in plot 2 of 6 in figure 23 is logic low meaning its input it logic high. In this state U1B outputs logic high plot seen in 3 of 6 figure 23 inferring that in the mines stable state its input is logic low seen in plot 4 of 6 in figure 23, its output is also used as feedback to the input of U1A. When the trigger switch is closes it pulls down the voltage at the switches side of C2 plot shown in 6 of 6 in figure 23, due to the capacitor only allowing transient current to pass this momentarily pulls down the voltage to one of the inputs of U1A seen in plot 5 of 6 in figure 23, making its output logic high. Due to C3 acting as a filter U1B and the draining effects of R5 U1B receives logic high as an input. Close inspection of the Simulink simulation in figure 23 shows that this disturbance settles after approximately 0.6 seconds however is more than long enough to force the input of U1B to logic high and its output logic low. The feedback from the output of U1B resets the configuration assuming the switch that started the process returns to an open position. U1D is switched into a logic high state when it receives a logic low input from U1B, this pushes the cascaded bipolar junction configuration into saturation in turn draining $\mathrm{C_{det}}$ into $\mathrm{R_{det}}$ which detonates the mine.

The way in which the device holds this charge means that damaging or disabling the batteries that provide power to the circuit may not immediately disable the device and render it safe. A & Type72B are visually identical which implies the render safe procedures should ideally have some commonality, as it will be difficult, if not impossible, to determine which type the operator is working with.

A small experiment was carried out to assess how a 3V watch battery would perform if a hole was drilled into it. It was found that it was unable to deliver current after such damage, therefore it is expected that the batteries in the device would perform in the same way. The device however uses two batteries, meaning the most likely that both must be drilled to achieve the same effect. In addition to this, as the device is symmetrical, locating the position of the battery can only be done if the location of the firing pin hole is found first. Once this is done, the position of the battery should be directly opposite and slightly off centre as detailed in figure 24.

Another possible solution may be to drill the tilt switch where the firing pin is positioned in the hope that this would force the switch into a permanently open state, though success depends on the switches modes of failure. This would prevent the device from firing, though it may be difficult to determine if the switch had been drilled correctly and was in an open state. More research must be undertaken on this device in order to determine the best procedure to render it safe.

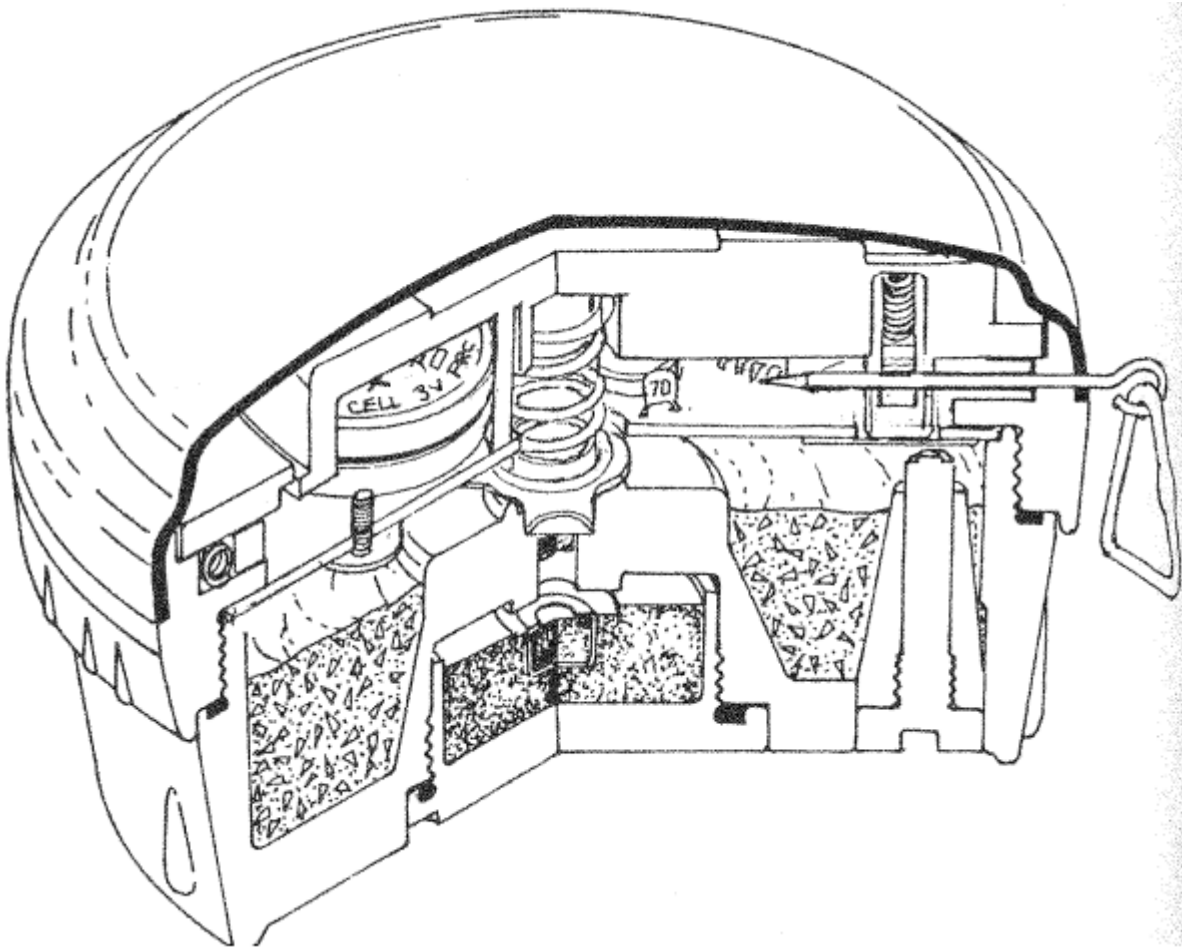

Figure 24, Type72B (King, 1996)

PROM-1

The PROM-1, as described in the literature review, has a complex firing mechanism. For the purposes of further analysis, it was decided to build a 3D model of the mechanism. The 3D model can be found in figure 25. Only the critical components of the firing mechanism are illustrated in the 3D diagram. The top part has a locking ring (not illustrated) that holds the mechanism in position and prevents water ingress. When it is disturbed, its movement forces the part shown in blue downwards. There is a locking ball located in a hole in the part shown in green. Though not illustrated in the 3D diagram, it is included in the figure 25. The downward movement of the part shown in blue makes the locking ball fall into the cavity also shown in blue, allowing the firing pin shown in red to be forced down into the detonator. If the part shown in blue was jammed and unable to move downwards, this would prevent the firing mechanism from activating.

The proposed render safe procedure to tackle this device would be to drill a hole in the side of the trigger at the point where the part marked in blue narrows slightly shown in figure 25 which should provide a channel for the oil and epoxy to flow down into the cavity below. The literature review states that the original image from (King, 1996), also shown in figure 10 features a person's thumb holding the mine which was used as a reference to determine its approximate dimensions. From this it was estimated that the outer case is 2mm thick. In this case it is particularly necessary that light oil should be injected first, followed by the mixed epoxy. Its low viscosity would allow it to seep down and force out all the air in the cavity below, leaving it full of oil. This should be done as the epoxy is considerably more viscous than the oil and may impede the flow of air out of the cavity. Injecting the epoxy first would risk it setting in the channel rather than at the bottom of the cavity, which may leave the device in a precariously active state.

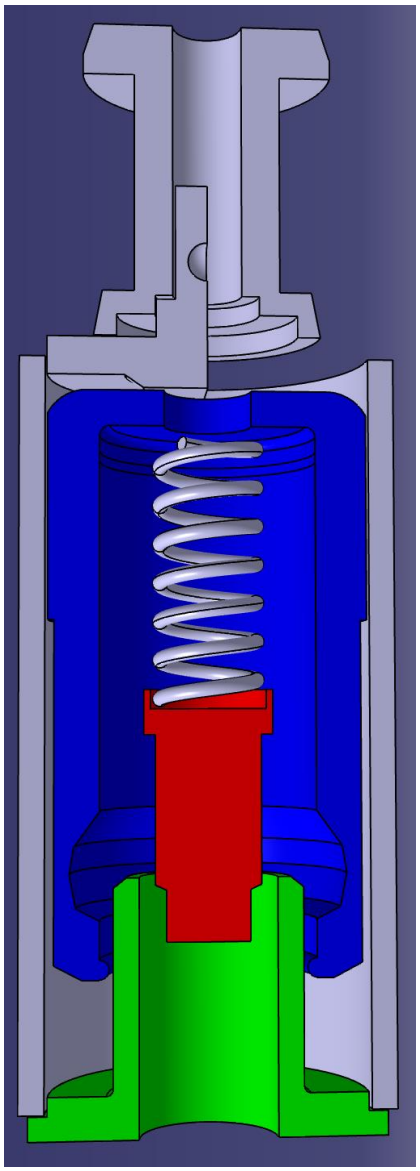

Figure 25, Illustration of PROM1 firing mechanism (Grindley, 2014)

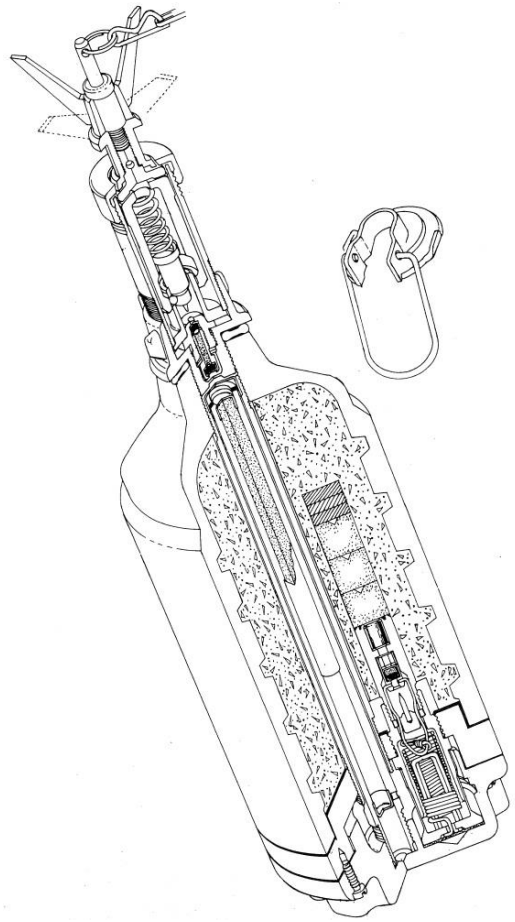

Figure 26 PROM-1 Cutaway (King, 1996)

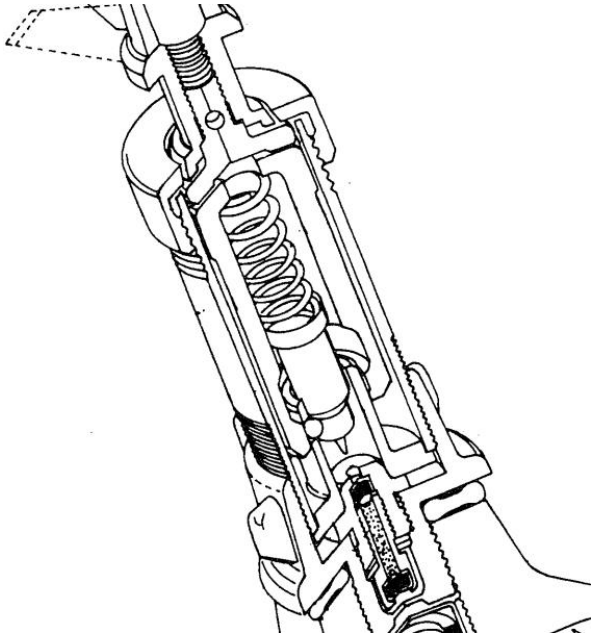

Figure 27 PROM-1 trigger close up (King, 1996)

Conceptual design

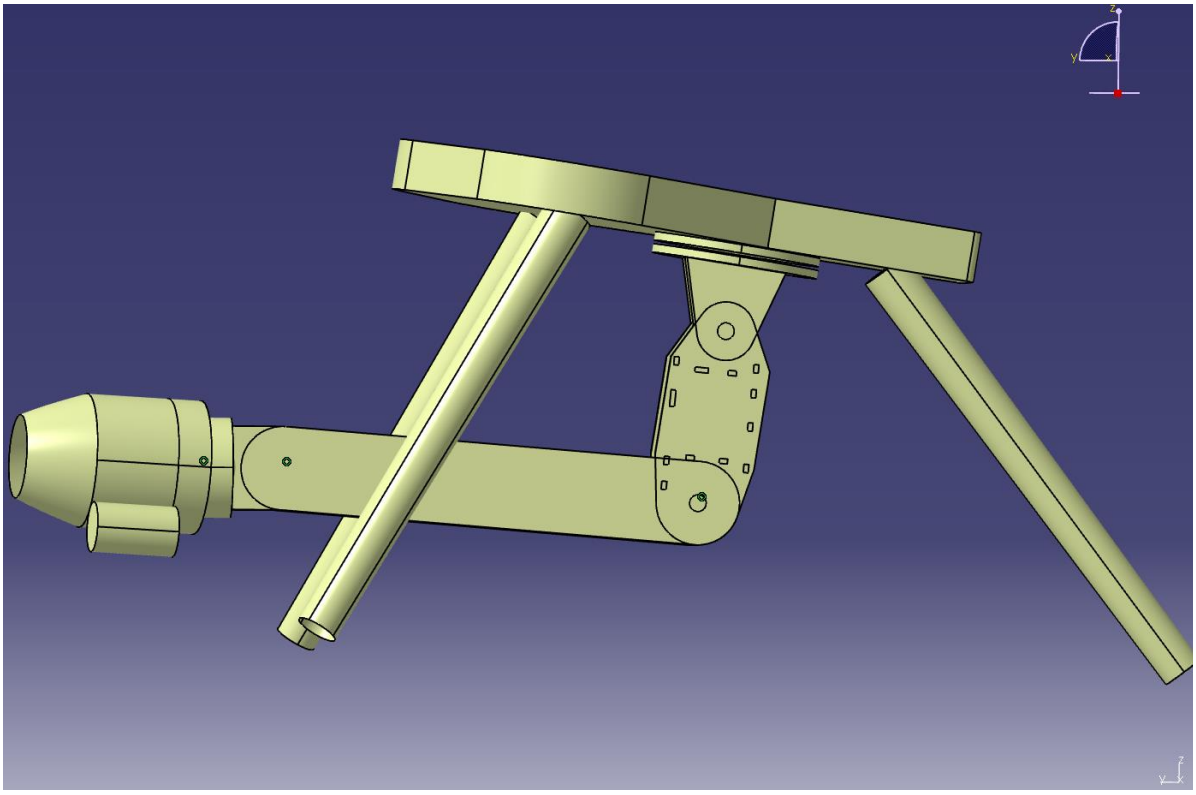

Figure 28, Conceptual view of all manipulator structure (Grindley, 2015)

Chapter 4 - Catia design and mechanical drive

Human deminers are the most effective at the task of demining compared to any mechanical method. As such the structure of the manipulator should have some similarity in terms of dexterity and freedom of movement. The end of the effector is connected to the rest of the structure with a ball joint (joint 5), this gives pitch and yaw as two degrees of freedom in a similar way that a wrist does for a human had. On a human arm the forearm allows the hand to roll or rotate about the x-axis and y-axis, this was neglected to keep weight down as roll is not necessary to perform the proposed task. Though the grippers may benefit from this extra degree of freedom its implementation was considered further work. The next joint is a revolute joint (joint 4) rotating about the x-axis, this is not a feature found in a human arm but was implemented as there is a need to lower the end effector to inspect underneath a device. Being able to inspect underneath is important because typically this is where anti-handling (booby traps) devices are installed. The long linkage that connects to the next joint (joint 3) may be adapted to extend or contact in a prismatic fashion, allowing the end effectors pose to be moved forward a greater distance than is now possible, though this was considered further work.

The Final linkage is connected with two revolute joints either side (joint 3 and 2), this linkage allows the end effector to be shifted forwards and backwards this requires another two degrees of freedom provided by the revolute joints. Finally a revolute joint connects the arm to the base of the manipulator, allowing it to swivel about the z-axis giving another degree of freedom (joint 1).

The legs of the tripod base are fixed though the means of fixing them is considered further work, this may be done using a friction breaking system though linear actuators utilising a worm gear would be the likely preference as there typical failure mode makes them stop working but not collapse. In addition whether or not they should be designed so they can extent and retract should also be considered, as this would also increase the manipulators usability.

Manipulator components

Structure

The design of the structure should be easy to produce and relatively inexpensive. Initially the choice of material was aluminium, due to its strength and workability. The main drawback however is that aluminium is typically expensive to mill on CNC machines. The time required can also be considerable, especially when milling an intricate design.

Inspiration for the construction of the structure came from a QinetiQ Dragon Runner 20 robot seen in figure 4 on display in an exhibition at the National Army Museum in London (QinetiQ, 2015). Although little documentation on its construction and specifications is available in the public domain on inspection of the robot on exhibition it was clear that most of the structure that made up the arm was constructed using carbon fibre. Figure 4 shows a Dragon Runner 20 robot, the two linkages that make up the arm are constructed from two sections. The first is rectangular and the second circular each constructed out of carbon fibber, at the time this was inspected on exhibition the thickness of the material was not observed and recorded. However the drive chain employed in the Dragon Runner 20 to move the joint in the centre of the arm was driven by a chain similar to a bike chain with the motor driving it located inside the main body of the robot. The use of a chain in the driveline for joint actuation was an interesting means of keeping the weight of the arm low while maintaining a ridged interconnection, it was also clear why a rectangular configuration was chosen to house it. Using this same design on the manipulator was considered however time constraints meant it could only be implemented as further work. Carbon fibre was seen as an ideal solution, as expensive processes such

as CNC milling are not necessarily needed. Though carbon fibre is relatively expensive it can be laser cut, a process that is both fast and cheap depending on the availability of a laser cutter. It should be noted that this is not without its drawbacks, laser cutting carbon fibber can reduce its strength around the edges where it is cut. Water cutting is recommended as a better solution but it is considerably more expensive, which is a design trade off it should be considered.

A price of aluminium billet however is reasonably priced compared to carbon fibre. For example a 70mm x 145mm x 10mm piece of Aluminium billet costs £6.30 including vat available from (Metalmania UK, 2015). In comparison, carbon fibre with similar dimensions 3mm x 210mm x 148mm is available for £32.34 from (Easy Composites, 2015). Although it is clear that carbon fibre is considerably more expensive, it can be laser cut quickly and easily, which makes its fabrication much cheaper, assuming a laser cutter is available for free, or at a low cost.

In order to make it both easy to assemble and rigid, it was decided to make a carbon fibre structure that would slot together. Figures 29 to 42 show the process by which a solid link can be made into a structure using carbon fibre that can be slotted together then held together with a nut and bolt arrangement.

Figure 29 shows the chosen section of the manipulator. All of the elements in figure 29 are structures and the link shown in purple has been made into a solid for the purpose of making the struts.

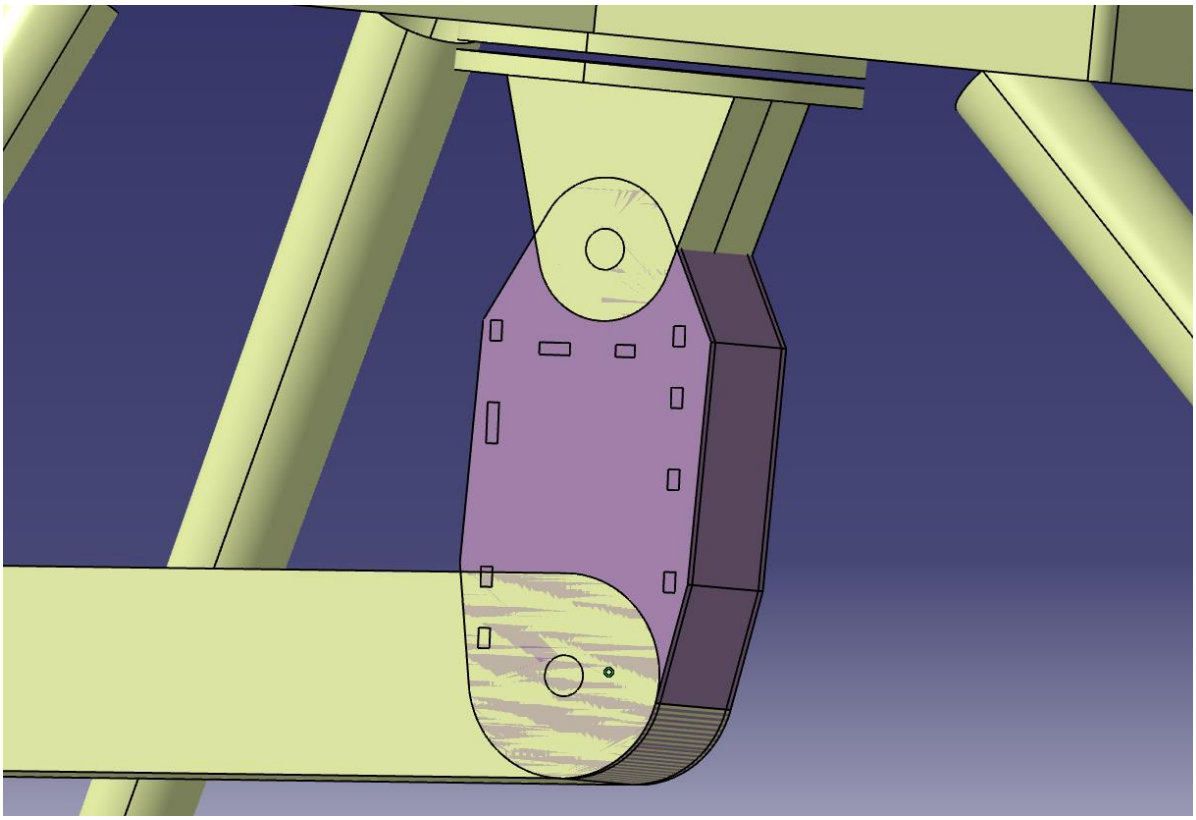

Figure 29, Demonstration of manipulator structure with segment selected (Grindley, 2015)

In this illustration a section of the manipulator was chosen to demonstrate the how it could be made into a rigid structure through assembling it as a number of carbon fibre struts.

Determining the exact dimensions and material properties required to ensure the structure was strong and durable enough to last in the field was not possible due to time constraints. This would require a more mature design and the use of complex analysis tools like finite element analysis. In the absence of a more detailed analysis a 3mm thick carbon fibber was recommended as a best guess, it is also the widest gauge commonly available that is reasonably priced. First the link was isolated from the rest of the design, in order to make it easier to work with.

A plane is then inserted and the results of its intersections with the link are used to produce an outline of the strut which is then extruded to the specified 3mm thickness.

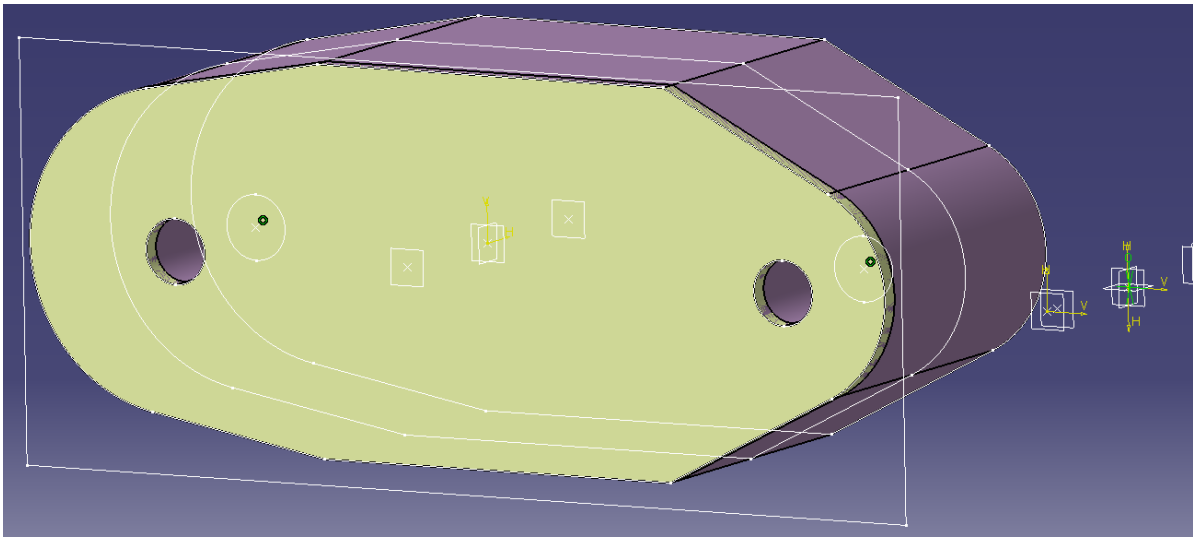

Figure 30, Segment with plain (Grindley, 2015)

The same is done for the opposite side, given two structures that form the base for the rest of the struts.

The struts connected along the top and bottom were then drawn in, note that a jigsaw arrangement was used with protrusions that allow the struts to be firmly slotted into each other. The outline is then extruded to produce a structure this can be seen in figure 31.

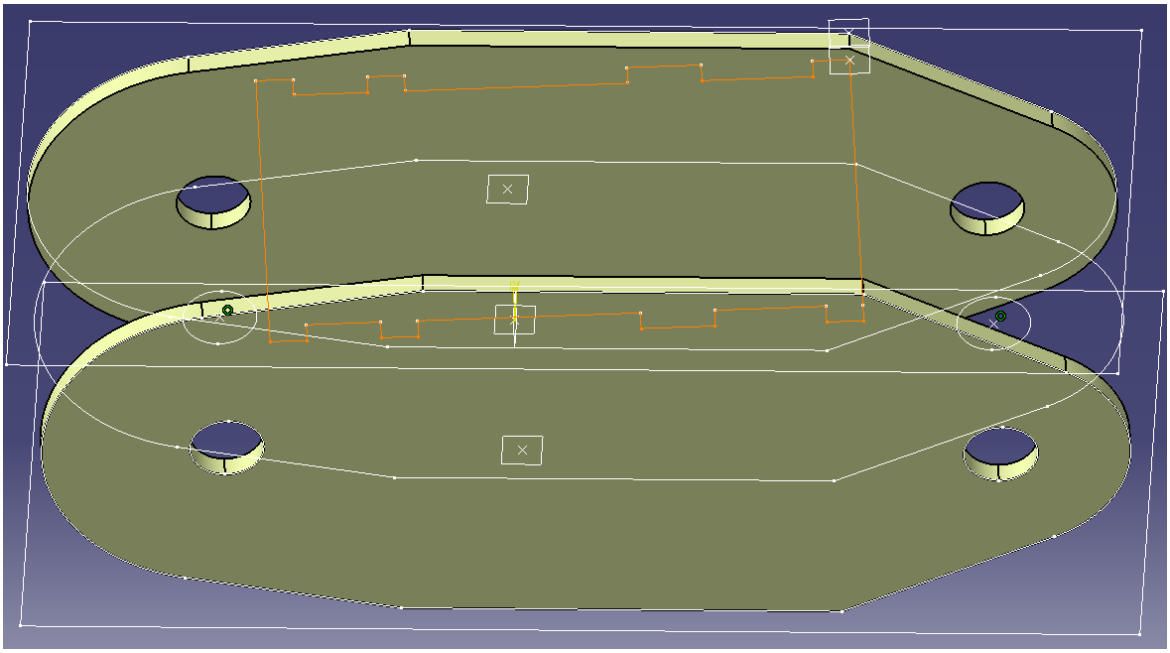

Figure 31, Top of structure (Grindley, 2015)

The same is done for the bottom of the link, connecting both of the sides together.

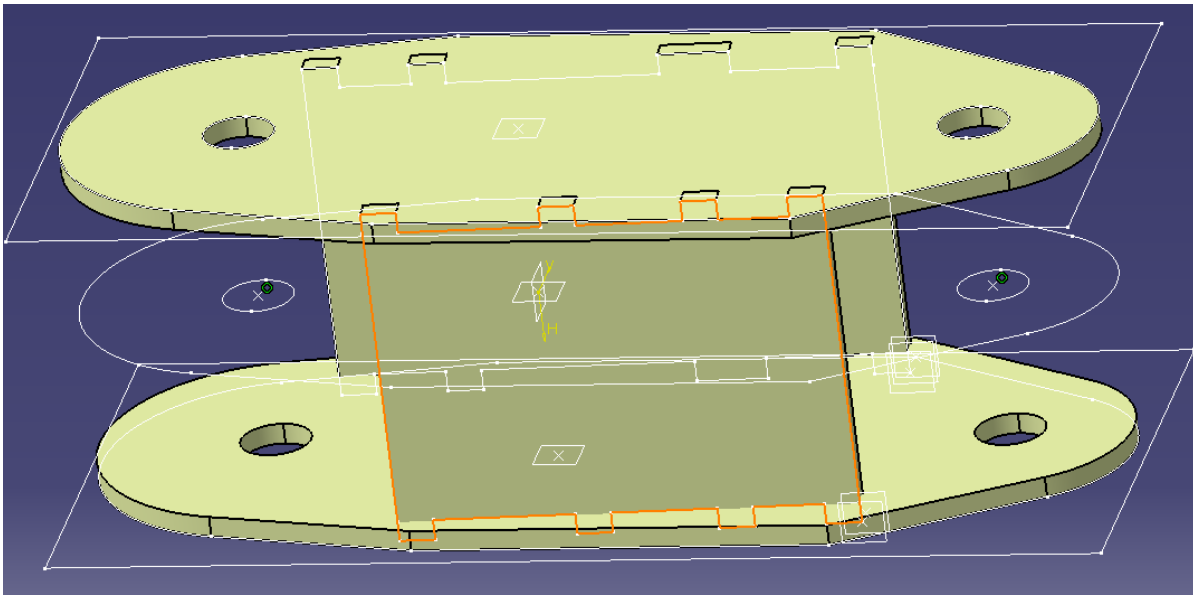

Figure 32, Structure design process (Grindley, 2015)

At this stage the structure affords some rigidity, especially in terms of the manipulators movement in regards to pitch, though it is still weak if forces are applied along the yaw and roll axis. To mitigate against this structures were applied centrally to the front and rear of the link; this can be seen in figures 33 & 34.

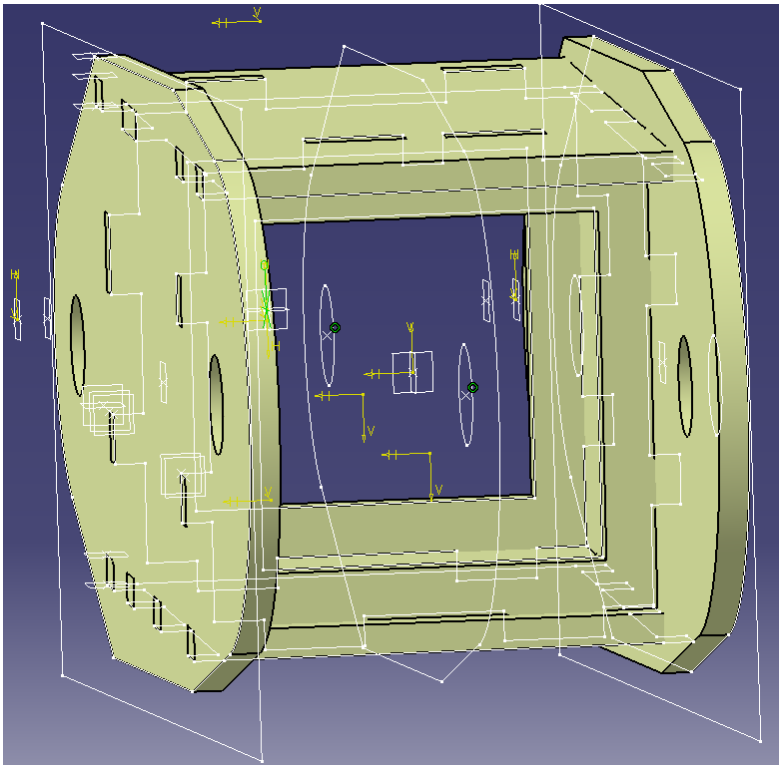

Figure 33, Structure design front (Grindley, 2015)

Finally four extra struts are added in order to distribute the load on the side struts.

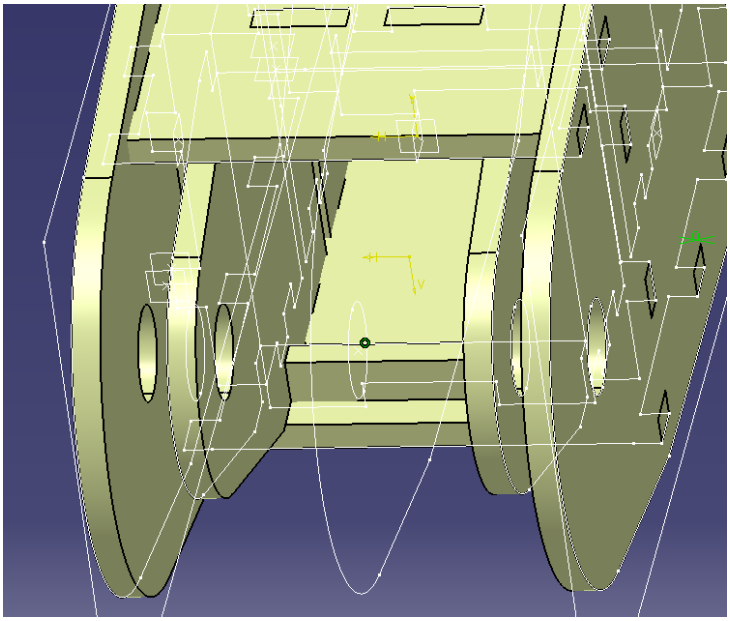

Figure 34, Finished structure (Grindley, 2015)

Each strut includes protrusions that make it fit with the rest of the design, though at this stage they exist as protrusions only and relevant cuts into the rest of the structure are added at a later stage.

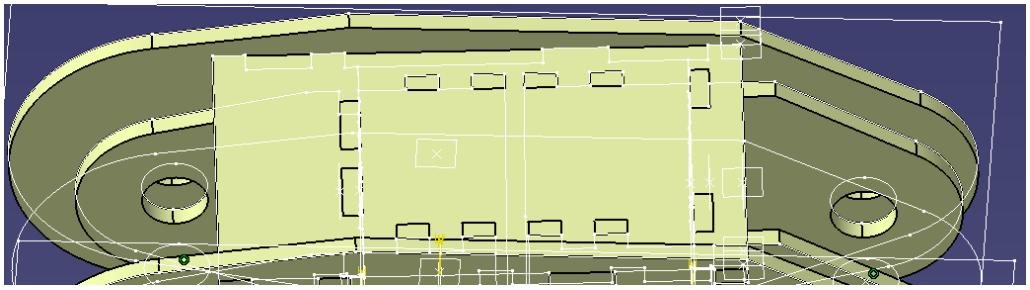

Figure 35, Top of finished structure (Grindley, 2015)

The next part of the process involves extracting the struts and making them into solids so that slots can be cut into them to complete this part of the design process.

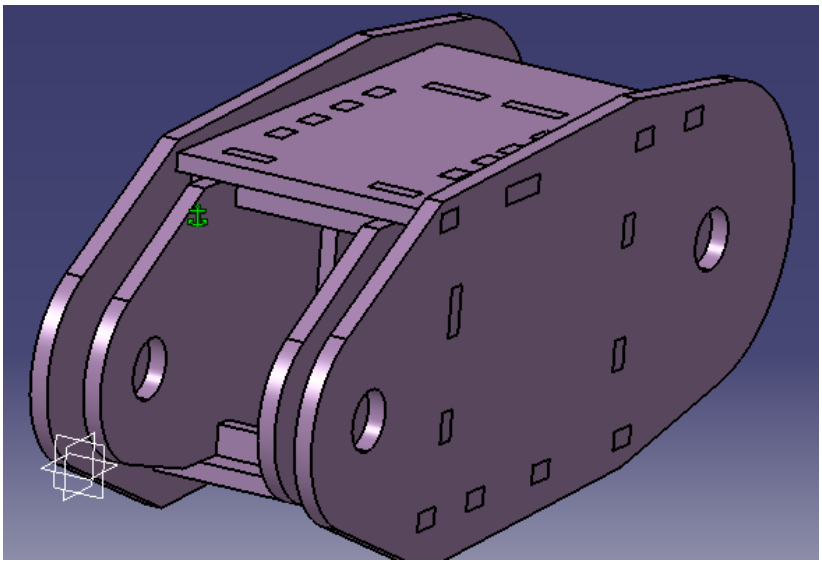

Figure 36, Finished solid link (Grindley, 2015)

Binary subtraction of the solids was used to make the indentations required for the struts to slot together, though the figures above show these indentations only as intersections because they do not yet exist as solids at this stage. This is illustrated in figure 37 below; the remaining solid includes the indentations needed to slot the design together.

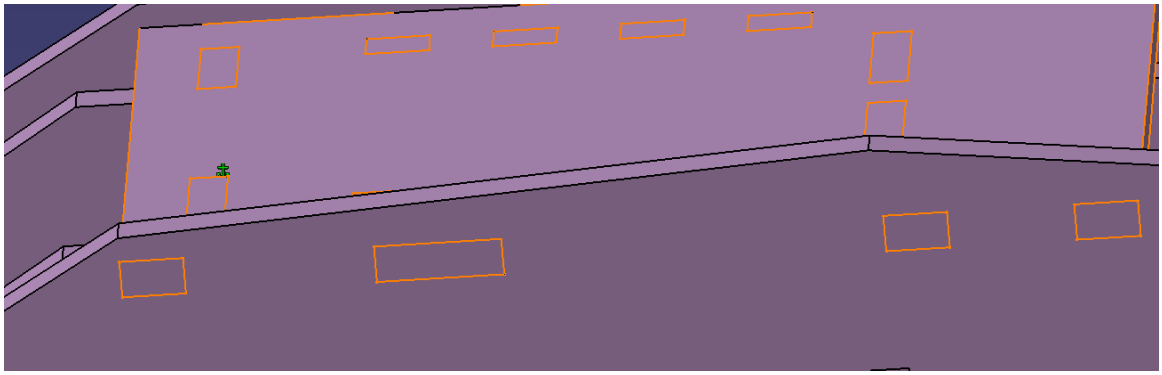

Figure 37, Top of solid showing cuts (Grindley, 2015)

Figure 42 shows an exploded view of the design, which can be slotted together. The final stage is to make holes for the nuts and bolts used to hold the assembly together.

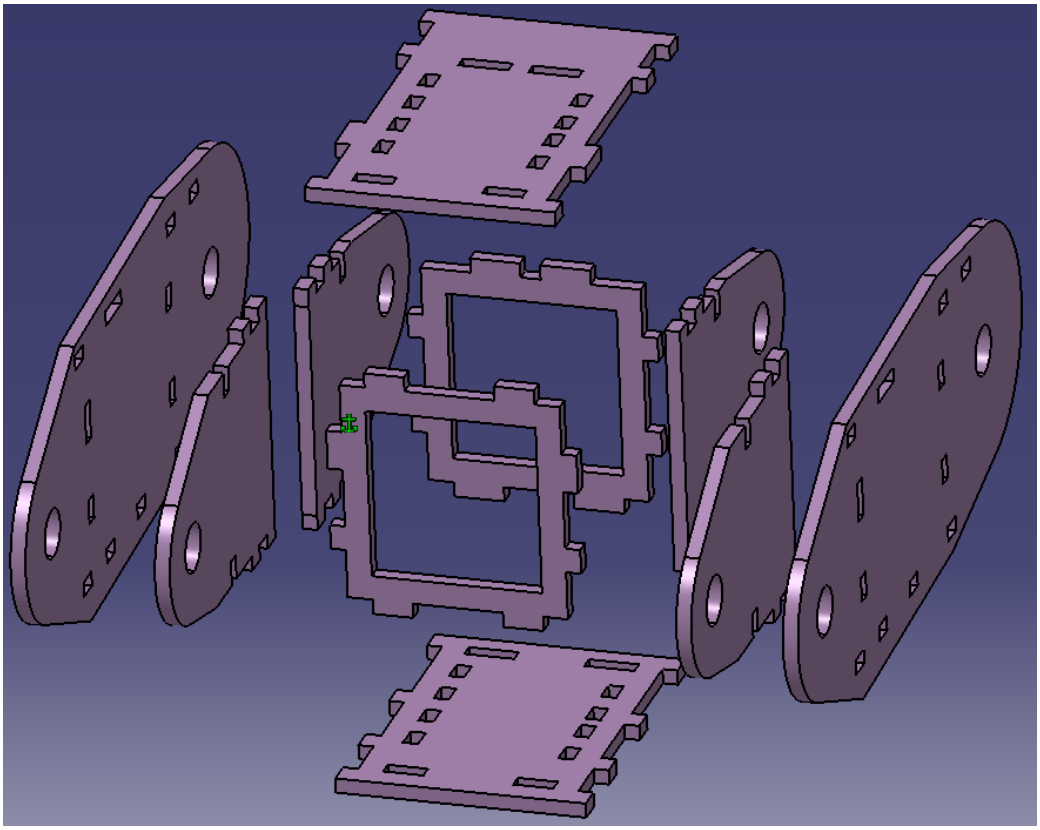

Figure 38, exploded view of link (Grindley, 2015)

The design however is incomplete without provision being made for drive and for sensors for control. Space for motors for the mechanical drive of the joints has been left in the centre. Though at this stage it was not clear as to the dimensions of the final drive actuators used would be. This will be considered further work.

Drill

One of the key components of the design was the drill. Early on in the project it was decided that the risk of disturbing components within a target mine due to a resonance effect from a drill posed too great a risk. Inadvertent triggering of a device is unacceptable, therefore ruling out a conventional drill driven with an electrical motor with a relatively low rate of rotation (approx. 50,000 RPM). An air turbine that could reach speeds of 400,000 RPM was a far better solution particularly as they are also relatively inexpensive and are mass produced for the dental industry. Another advantage of air turbines is their low mass compared to an electrical motor. As a result, both the motor that produces the compressed air and the cylinder to provide extra capacity can be shifted towards the rear of the manipulator, allowing for extra stability.

In order to determine if it was feasible an air turbine was purchased to better understand its operation, they are driven using compressed air typical of the order of 240kPa. As the motivation behind having a high frequency drill was the assumption that it would be less likely to cause resonance effects a higher frequency turbine was chosen. Dental drills come in two common forms the high frequency type is for dental work and the lower frequency types for cosmetics seen if figure 43, initially a dental drill was chosen and a set of fixtures where produced to mount the turbine inside.

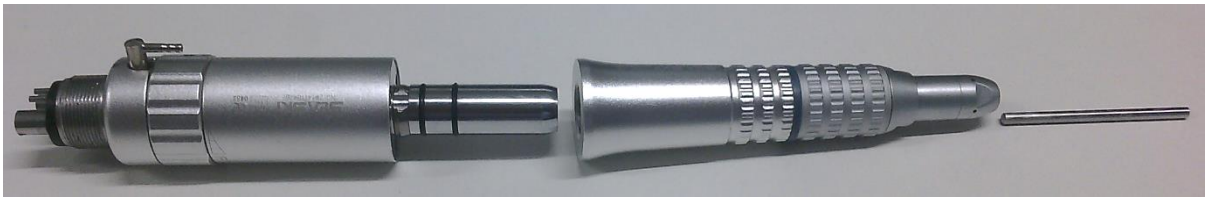

Figure 39, Cosmetic air drill (Grindley, 2015)

A small holder to mount the air turbine inside was designed for experimentation; this can be seen in figures 47 to 50. The holder was designed to fit the dimensions of the air turbine cassette visible in figures 44 to 46 and initially it appeared that the seals required to connect the turbine to the mount could be achieved with a small amount of silicone. After the holder was 3D printed it was apparent however that the dimensions specified in Catia were not those that were produced by the 3D printer and that if this strategy was to be more effective the holder must be reworked to make the turbine fit. In due course it became apparent that it was unfeasible to use this holder and another strategy was required.

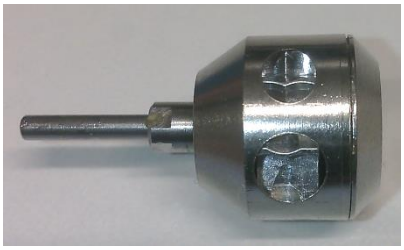

Figure 40, Air turbine cassette, (Grindley, 2015)

Figure 41, Air turbine cassette inside (Grindley, 2015)

Figure 42, Air turbine (Grindley, 2015)

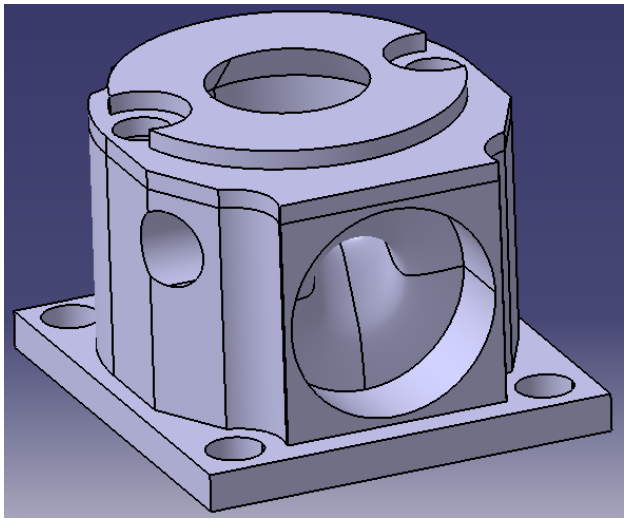

Figure 43, Turbine cassete holder (Grindley, 2014)

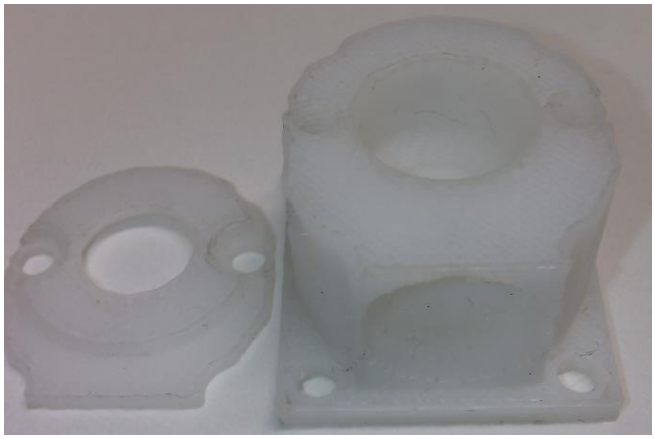

Figure 44, Rapid prototyped drill (Grindley, 2014)

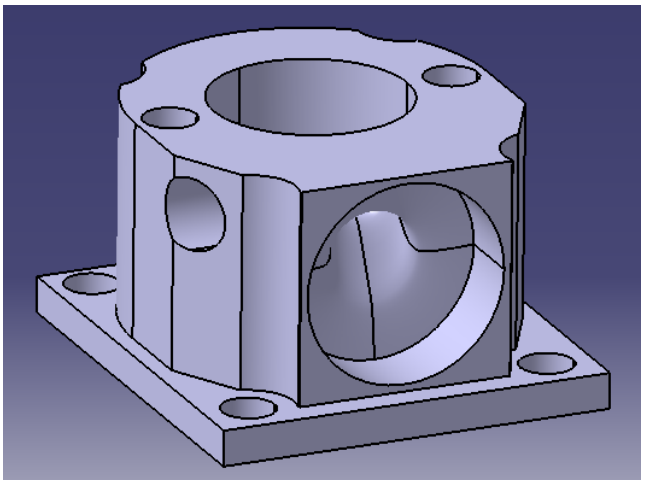

Figure 45, Side view of drill holder (Grindley, 2014)

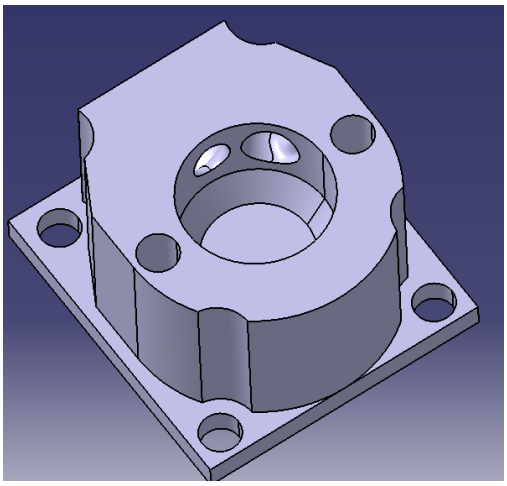

Figure 46, Top view of drill holder (Grindley, 2014)

The issue experienced with the holder was that the turbine in its case did not fit as expected. It was also not guaranteed that the seal would not leak between the holder and air turbine's intake. After some examination of the turbine it was decided that removing the casing altogether and placing the turbine in a specially designed housing would be the best option.

The specification of the housing were

- It must have input and output airline connectors compatible with a 4mm male stud pneumatic fitting with an M5 thread.
- Space shall be made behind the air intake and outtake for the airline connectors.
- It must have a 3mm diameter hole that has a direct line of sight with a single turbine blade, for an IR LED.
- It must have a 3mm diameter hole with a direct line of site with a turbine blade illuminated by the IR LED for a photo diode.
- The turbine housing should include space for the O-rings accompanying the turbine.
- The housing must have a point at which three rods can be placed allowing the housing to move along the Y –axis (positive Y being the forward facing direction when viewed from the side).
- A hole to attach a piston to control its forward and backwards movements shall be included.

- A top that can be sealed with an O-ring or with silicone shall be included to allow for the insertion and removal of the turbine.
- M1.6 screw holes should be included with space for the nuts to fix the top onto the holder.

Figures 51 and 52 illustrates the drill holder at the front (Left) shaft with space for 4mm connectors in the centre and holes for rods and piston to control the drill to the rear (Right). Illustrated in figure 51, 53, and 54 are the drill as designed in Catia and produced on a rapid prototyping machine in ABS plastic figure 52. Not that in figure 51 the top that holds the turbine in place is heled in place with nuts and bolts and sealed with impact adhesive glue.

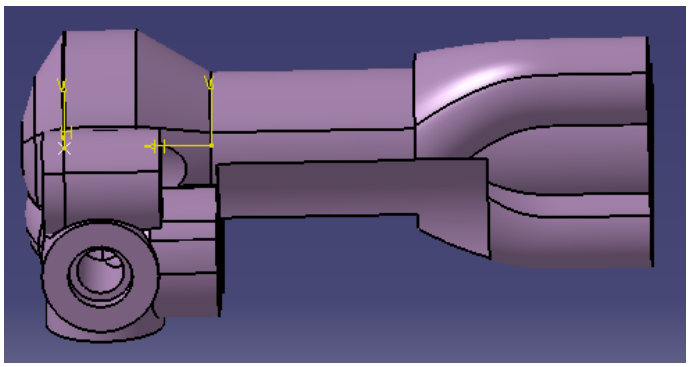

Figure 47, CAD drawing of air drill side view (Grindley, 2014)

Figure 48, Rappid prototyped air drill in ABS side view (Grindley, 2015)

The Figure 53 illustrates the three holes for the rods with a hole for the controlling piston in the centre. The air intake is visible to the right and the outtake is visible to the left note the holes are made to suit a M5 thread.

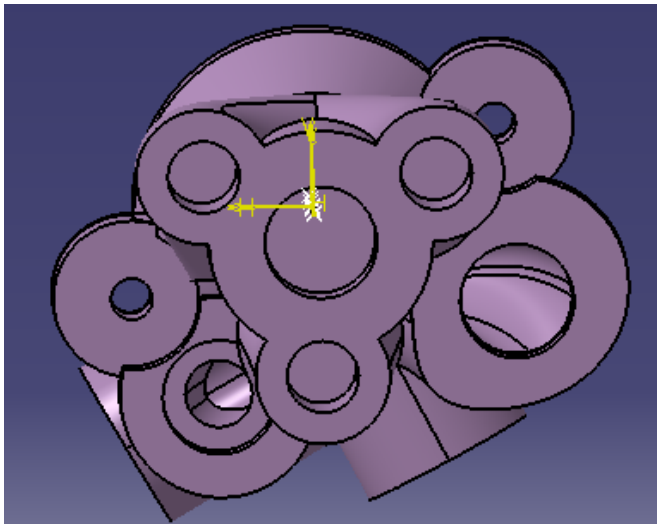

Figure 49, CAD drawing of air drill back view (Grindley, 2014)

Figures 54, 55, 56 and 57 show the chamber that houses the turbine, the holes to the side are M1.6 size to fix the top to the rest of the assembly. The left figure shows the drill holder with its top on and right figure shows it without the top.

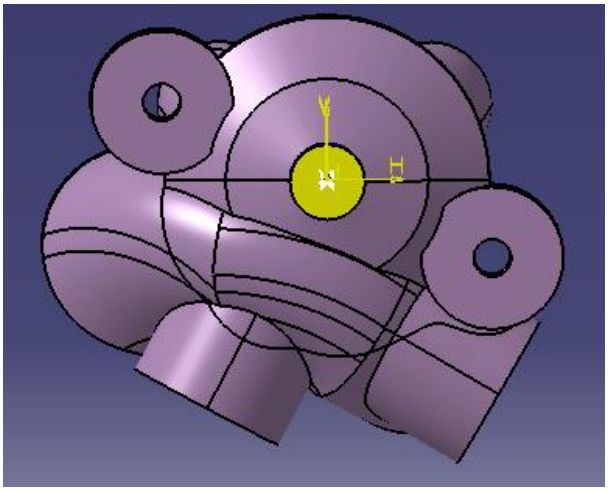

Figure 50, CAD drawing of Air drill with top (Grindley, 2014)

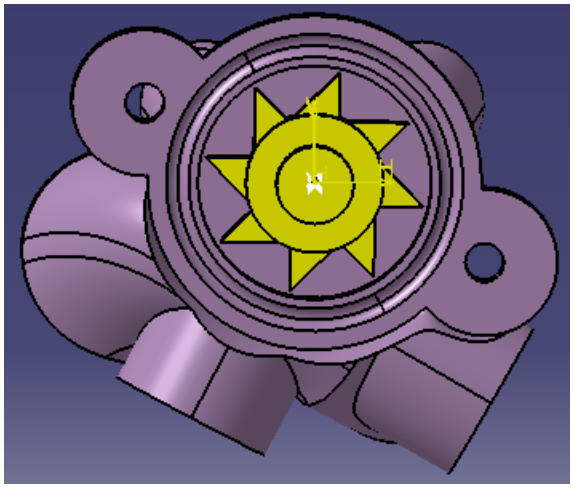

Figure 51, CAD drawing of air drill with no top (Grindley, 2014)

Figure 52, Rappid prototyped air drill head with top on (Grindley, 2015)

Figure 53, Rappid prototyped air drill head with turbine exposed (Grindley, 2015)

Visible in the figure 58 & 59 are the air intake and outtake. These have been positioned at a 30° angle to each other respective of the centre of the chamber; this is exactly the same as the positioning on

the original turbine housing. In the bottom of the chamber, the hole for the bearings is visible as well as the groove designed to fit its accompanying O-ring. Figures 52 and 59 illustrate the rapid prototyped drill holder produced in ABS plastic. Note the same angle was applied to the air intake and outtake as the turbine cassette that it originally came in, this was 30°.

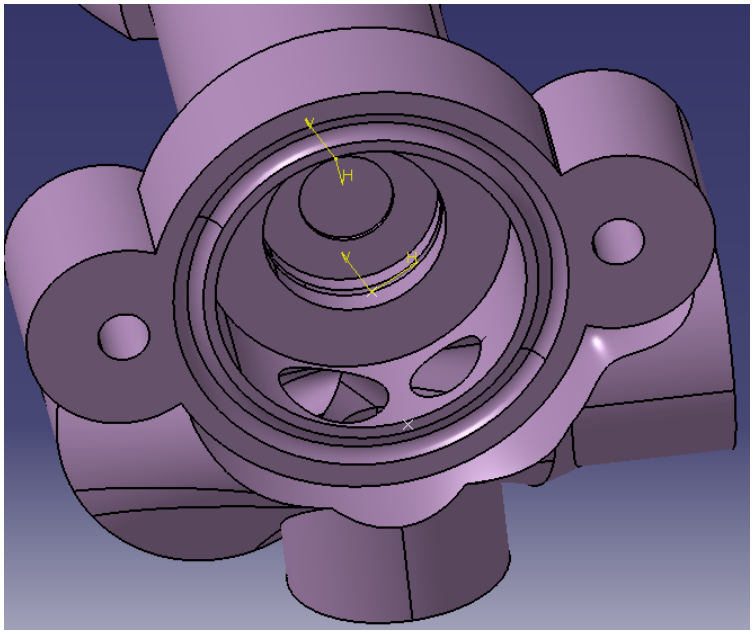

Figure 54, CAD drawing of air drill without turbine (Grindley, 2014)

Figure 55, Rappid prototype of air drill without turbine (Grindley, 2015)

The figures 60 & 61 show the countersunk holes designed to fit the IR LED figure 60 and IR photo sensitive diode figure 61. The positioning of each hole is specific so that each time a blade passes the field of view of the photo diode it will greatly reduce the light received. The amount of light received by the diode will vary due to the ambient temperature and life cycle of the LED and photo diode, as such the light levels that trigger logic 1 on the receiving circuit should be tuneable to account for this. Figure 62 and 63 show the rapid prototyped drill piece in the same orientation and the design in figures 60 and 61.

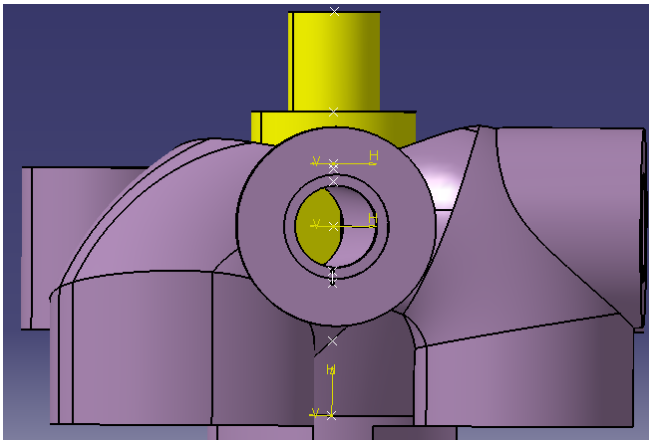

Figure 56, Hole for IR LED (Grindley, 2014)

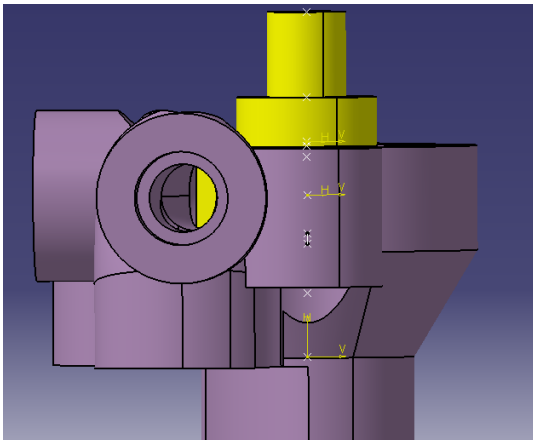

Figure 57, Hole for photo diode (Grindley, 2014)

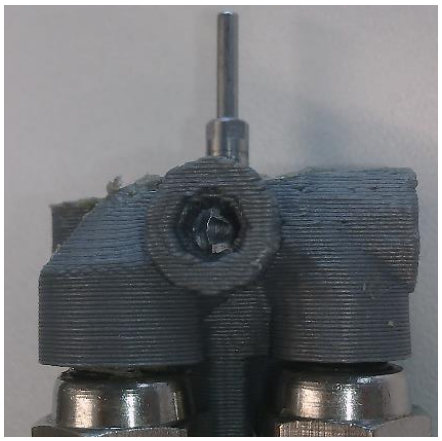

Figure 58, Rapid prototyped air drill top view (Grindley, 2015)

Figure 59, Rapid prototyped air drill side view (Grindley, 2015)

Air nozzle

The air nozzle is necessary to move earth away from a suspected mine. Typically, when a deminer is uncovering a suspected mine they would use a trowel or a similar instrument to move away loose soil from around the device (Keeley, 2003). While using a trowel or similar instrument is a simple task for a human deminer it is much more challenging for a robot. A robotic solution to this problem is complex because a human deminer has the sensory perception and special awareness needed to ensure the correct use of a tool such as a trowel. In contrast, developing a manipulator that has a comparable level of perception and control is not feasible within the scope of the project; therefore a more suitable solution was needed. The Wheelbarrow robot developed by Remotec (AZO Robotics, 2015), (Remotec, 2015) for example can be fitted with a large rotary centrifugal air pump suitable for moving sand and other loosely packed types of earth. This approach to uncovering a device was seen as a good solution, though some further considerations were needed to be made for this implementation. For example, the source of the air should be considered; a small diaphragm based air compressor similar to that found in a portable car tyre pump was a potentially cost effective solution when used in conjunction with a small air cylinder as described earlier. Such a configuration could provide short high pressure bursts of air rather than a continuous flow, like a rotary centrifugal pump or a ducted fan. The system would be accompanied with pressure sensors for feedback and a number of electrically actuated solenoid valves. This may have some key advantages; compressed air of 0.7M Pascal for example may dislodge soil with more ease than air of a lower pressure and a higher

volumetric rate of flow. This was also justified by the low cost and availability. Some of the examined air cylinders within the operating range of cheap compressors operate at a maximum pressure of 0.7M Pascal (SMC USA, 2015). Also, a compressed air source requires ducting with a lower diameter than would otherwise be needed for a source with a higher flow rate. The end of the manipulator itself needs a way of directing the air at the target area independent of the direction the manipulator is facing, although still within the same arc as the forward facing direction of the manipulator. As stated above, two degrees of freedom are needed (pitch and yaw). Such freedom of movement requires additional actuation, though actuators that deliver a lower force can be used, as the nozzle has minimal load bearing properties compared with the joints that make up the manipulator. It was concluded that actuation can be achieved by the use of high torque servos with cables to transfer the load. Steel brake cables as used on bikes are a suitable way to couple the nozzle to the servos, though they would have to be placed under tension to stretch them before use. This was justified by the low weight of the solution and low cost of its components.

Feedback was also considered because the direction in which the nozzle is pointing is difficult to determine by knowing the position of the servos alone. In figure 64 there is an illustration of the intended method of determining the position from which the air nozzle would be blowing air. The red line has been added to simulate the laser described earlier, though its width is exaggerated. In order to determine its position, a small camera will be placed on the manipulator that has the same arc as the air nozzle. A typical red laser has a wave length of approximately 650nm (Arima Lasers, 2015). The colour spectrum of the image seen by the camera can be converted from RGB to HIS colour space (Gonzalez, 2007) as the hue and intensity of the laser will remain constant within a small range, making it easy to detect. Knowing the ranges of hue and saturation will allow the system to determine the exact point within the arc that the nozzle is facing and this can be used as feedback to the servos that control the direction. This method does not resolve the direction directly but rather the intersection position of the air nozzle's forward facing direction with the ground. This has the potential to enable the operator to specify an arc within which to aim the air nozzle and have the control system automatically blow in air over the whole plane, as demonstrated with the white line in figure 64.

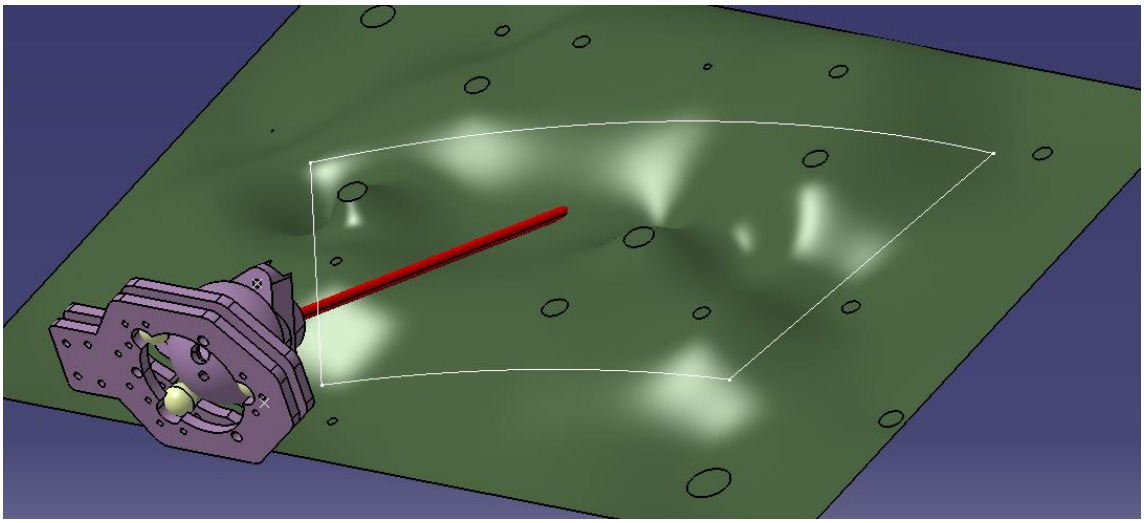

Figure 60, Illustration of air nozzle concept (Grindley, 2014)

When considering the materials used in the construction of the nozzle however there were few options available. The design of the nozzle itself has intricacies that for the purpose of the

demonstration of a concept model were only feasible with a 3D printer. Considering that the only stresses to which it is subject to are compressive stresses respective of the reaction forces of the air pressure and the shear stress due to tension in the control lines, ABS was considered an acceptable material to use. As it had adequate strength and its use meant that a rapid prototype machine could be used to produce it. The holder however needed to withstand both the shear stress due to the control lines and the force due to air exiting the nozzle and as such the required material would have to be considerably more durable under these conditions. Carbon fibre of a 3mm thickness was chosen for this. Two sheets connected together would make a structure that could distribute the forces and hold the ball bearings fixed in place, thus allowing the nozzle to rotate.

The specification of the nozzle were

- 2 degrees of freedom (pitch and yaw)
- Takes a 12mm threaded air nozzle to direct flow
- A standard M5 threaded pneumatic connector at the back
- 3D printed design using rapid prototyping machine
- Laser cut carbon fibre design for holder
- Space for laser pointer to determine direction of the nozzle
- Holder should be screw mountable
- Holder should have three grooves cut to allow ball bearing's to fit
- Mounts for control cables should be included

Several designs were experimented with, though the example in figure 65 was decided to be the best choice.

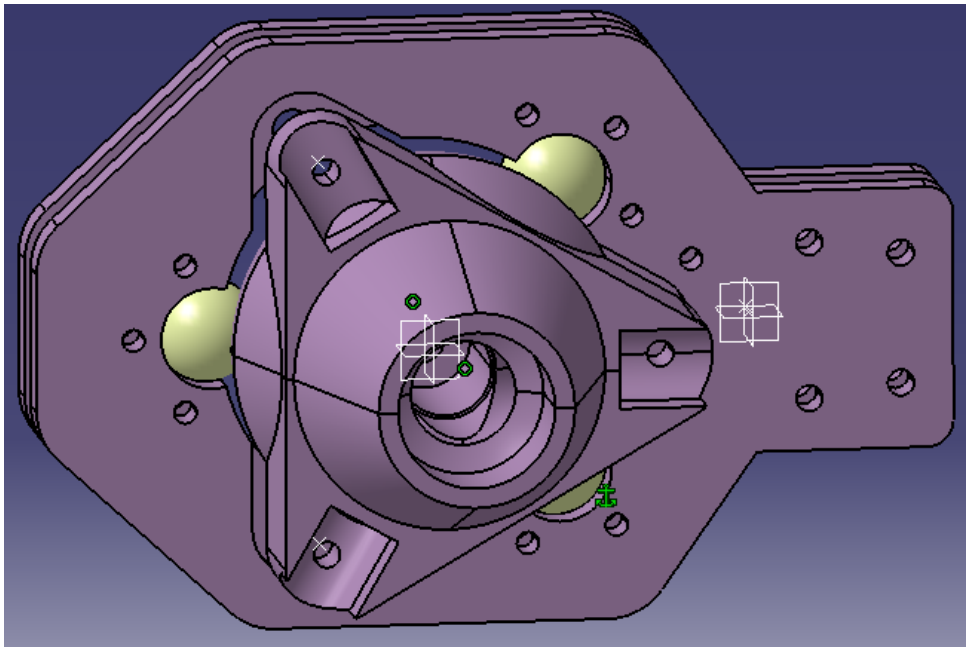

Figure 61, Illustration of air nozzle (Grindley, 2014)

In order to allow the movement of the nozzle along the two desired axes (pitch and yaw) a simple yet effective solution was to use ball bearings of 8mm in diameter held in place by two plates as shown in figures 66 & 67. The plates were to be screwed together with 1.6M screws and nuts, with three holes

surrounding the three ball bearings and two more closer to the base mount of the holder. Spaces were made to allow the nozzle to rotate within the desired arcs.

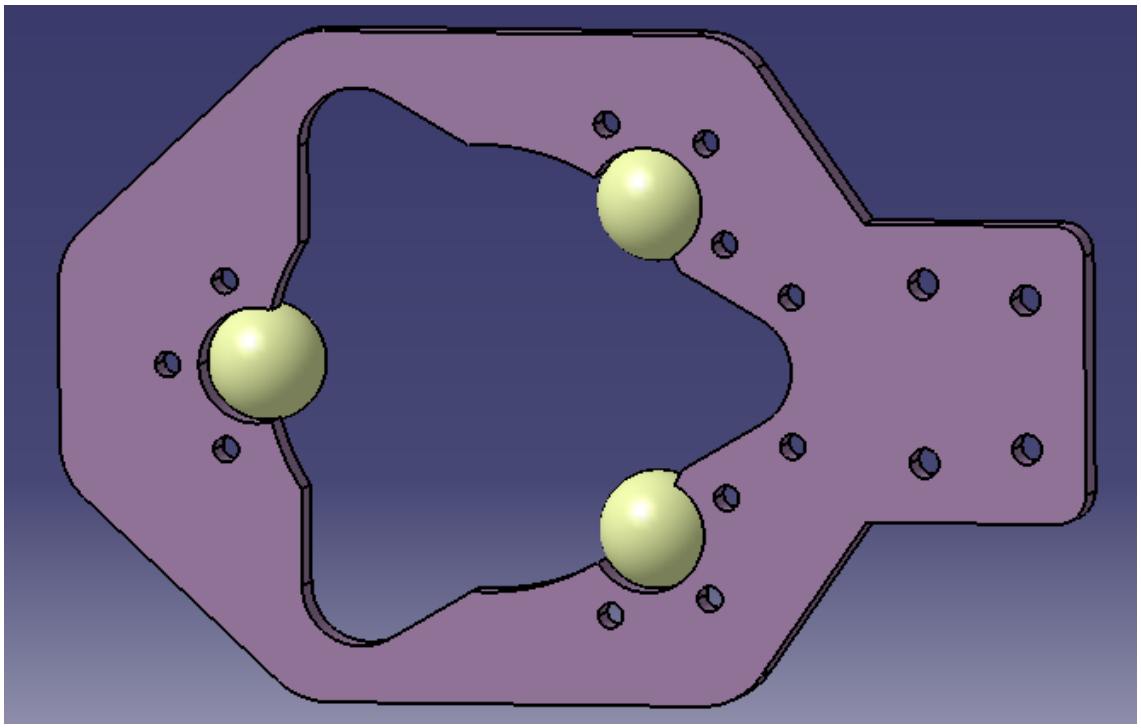

Figure 62, Front of nozzle holder (Grindley, 2014)

The back plate has a similar design to the front as the distance between the back and front plate was designed to be 3mm. The holes for the ball bearings were made specifically so that these dimensions would be constant. Three additional holes of 3mm in diameter can be seen for the control lines which run to the main body of the manipulator. As this was a demonstration of concept model, wear on the holes was not considered, though for future versions a means of allowing the lines to move without rubbing on the holder will be considered. This would likely entail another small configuration of ball bearings with another back plate to hold them in place.

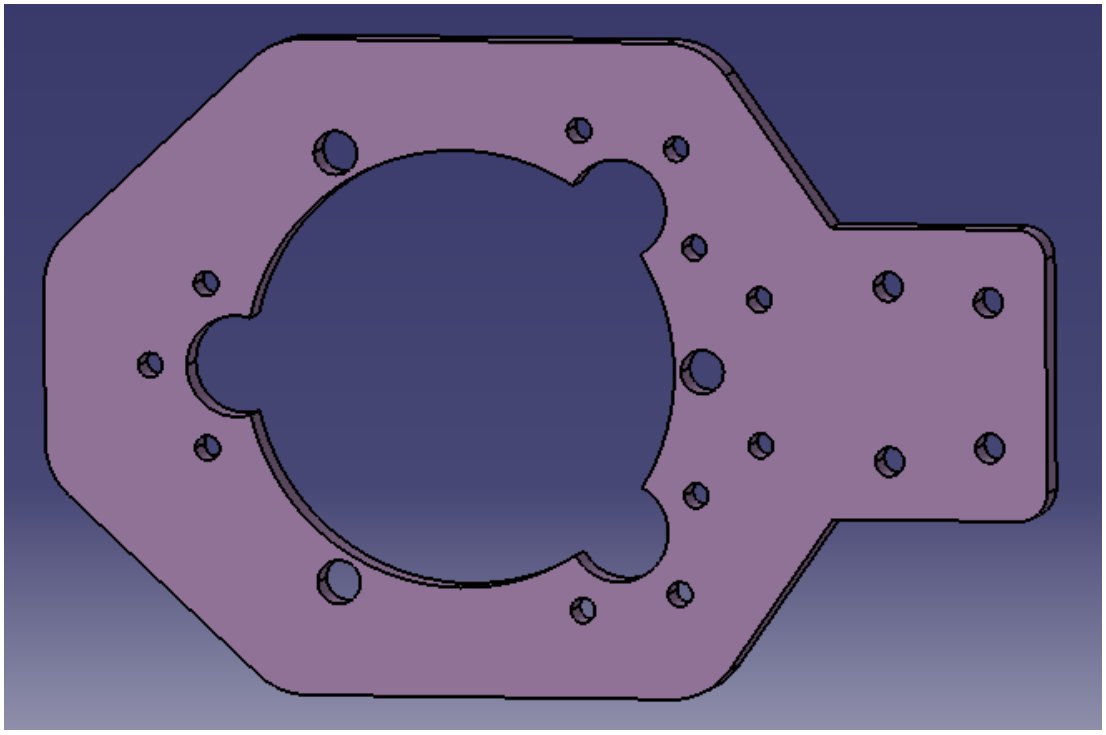

Figure 63, Back of air nozzle holder (Grindley, 2014)

The nozzle itself is spherical at the base to enable it to rotate on the ball bearings and it has a triangular configuration for the control lines towards the front. The grooves and holes cut into the three corners are for the fittings to secure the control lines; these are made to fit a nipple from the brake of a vesper scooter, as they are readily available and fit for purpose. Towards the rear of the nozzle there is a space for the laser pointer to fit and a small hole that leads to the back for its wires to run. The laser should be fixed inside with epoxy resin to prevent the air leaking through the hole for the wire.

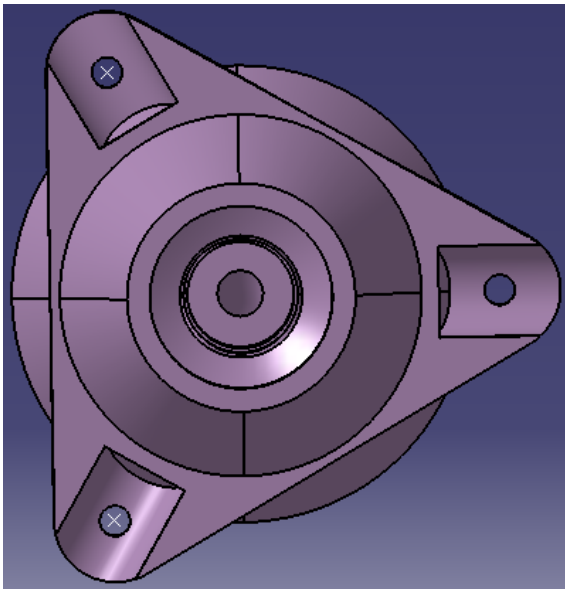

Figure 64, Front of air nozzle (Grindley, 2014)

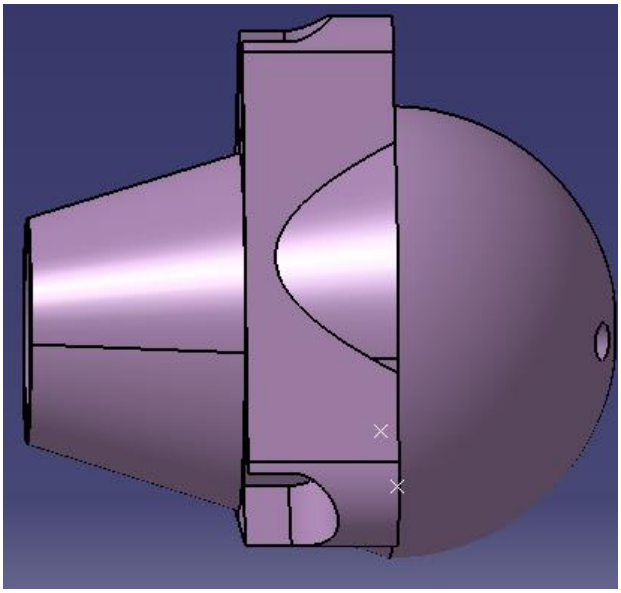

Figure 65, Side of air nozzle (Grindley, 2014)

The rear of the nozzle shows the M5 connecting port for the air supply and also the hole through which the wire for the laser would run. The M5 connecting port will self-tap when the pneumatic connector is screwed in.

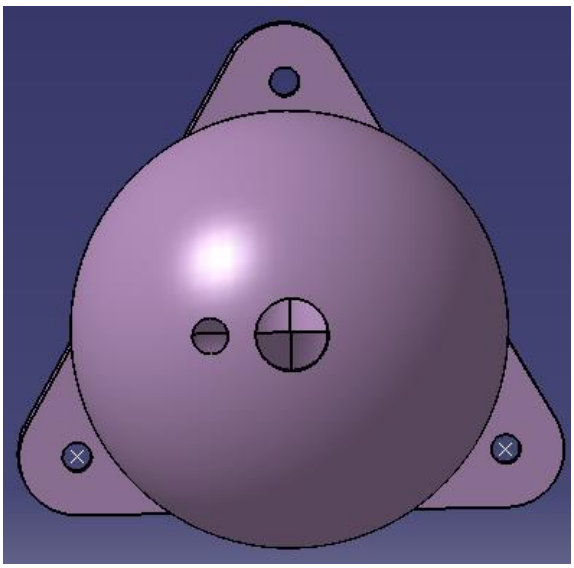

Figure 66, Back of air nozzle (Grindley, 2014)

The cutaway in figure 71 shows the inside of the nozzle depicting both the two passages for the air to flow down and the space for the laser. It is notable that this design is best suited to 3D printing, as other methods of fabrication, such as milling, would be expensive and likely require the design to be made from two or more pieces. The front of the nozzle has a screw in attachment to be interchangeable to produce different spay patterns to suit the circumstance. There are two spaces for the air to flow around which provides a low conductance path for the air to flow through. The paths are larger than the diameter of the airline, adding some capacitance into the system.

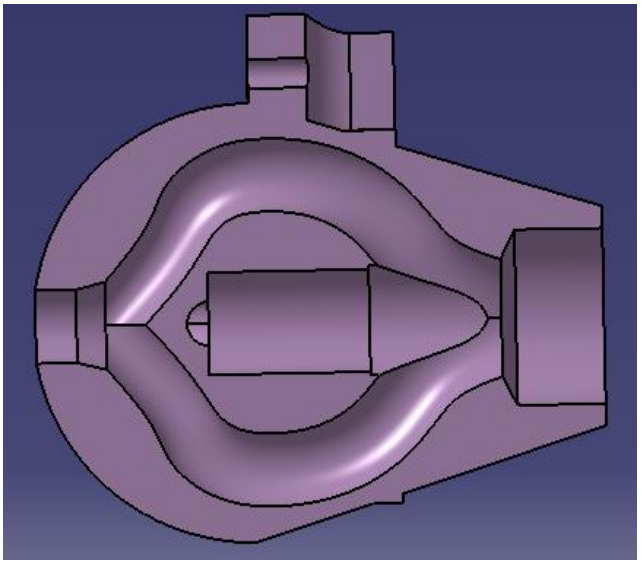

Figure 67, Air nozzle cutaway (Grindley, 2014)

Epoxy injector nozzle

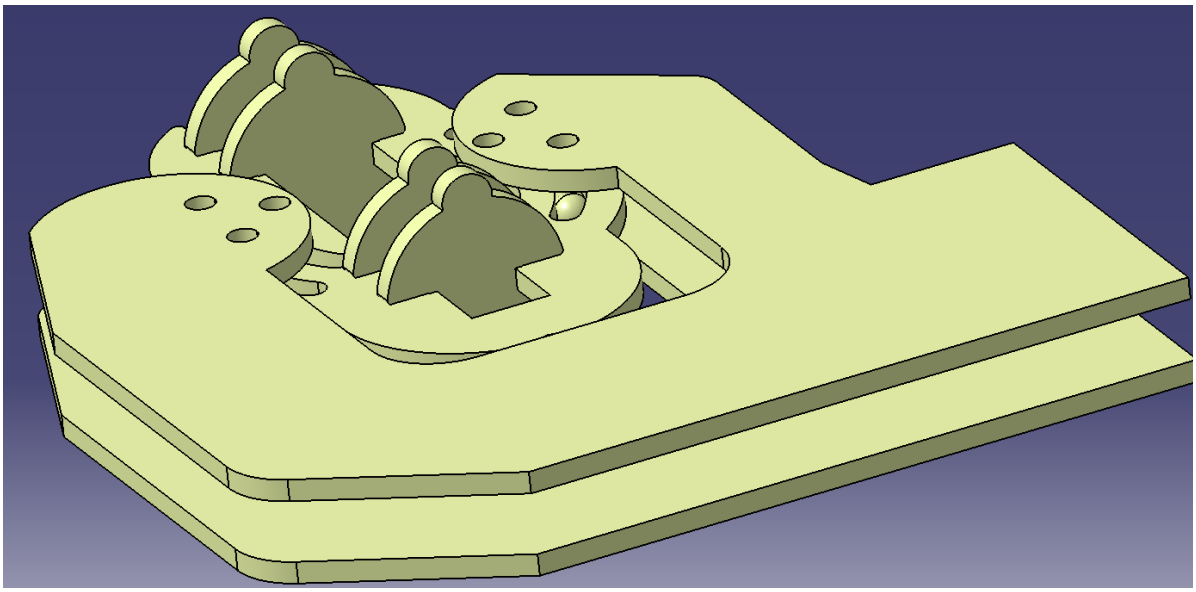

Figure 68, Epoxy injector concept (Grindley, 2014)

Figures 72 and 73 demonstrate the basic concept behind the epoxy injecting nozzle tool. Although it is incomplete, it is intended to demonstrate that its construction can be made from carbon fibre lazed into shape. In a similar way to the air nozzle, all bearings are introduced into the design to facilitate rotation about one axis (yaw). The rotation should be done with a small linear actuator or servo, connected to the back of the rotating section.

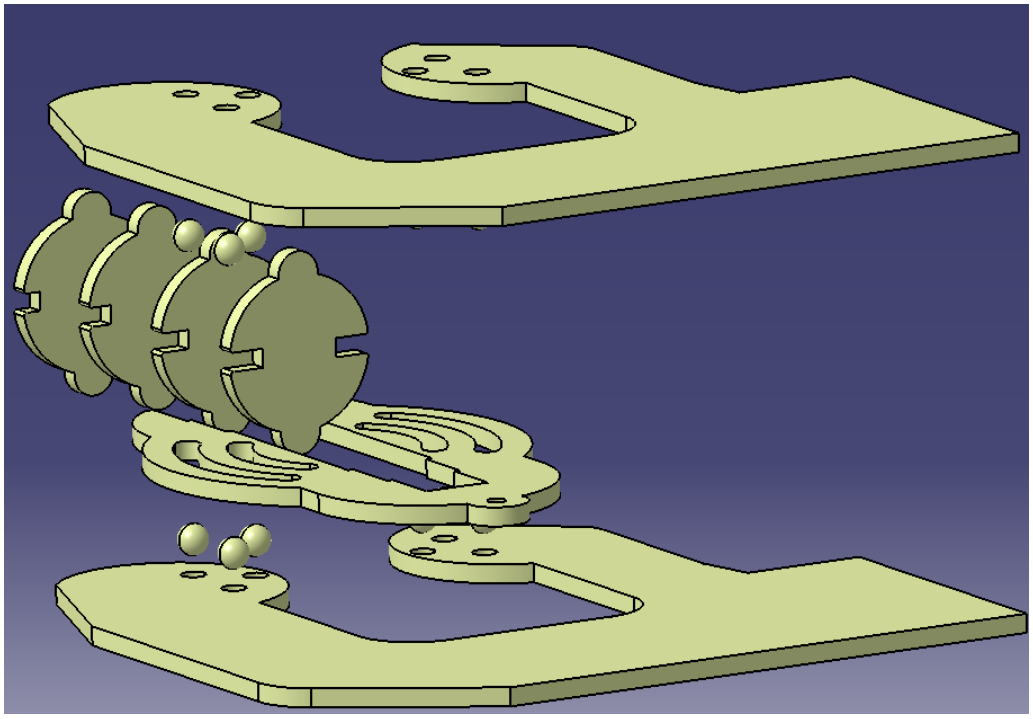

Figure 69, Explosion of epoxy injector concept (Grindley, 2014)

Drive

Given that failure of either the drive or the electronics is a possibility, is important to have a drive mechanism that does not allow the device to collapse in such an event. Both hydraulics and pneumatics and electrical drive were explored as the main method of actuation for the joints that make up the arm. There were some advantages of doing so as this would allow for the main bulk of the drive system to be kept towards the rear of the robot. This would reduce the mass towards the front of the robot, allowing for a faster and more stable response from the arm, in addition this will help stabilising the structure by shifting its mass closer to the centre of gravity.

Hydraulic

It was found that hydraulic systems of actuation or load bearing structures would have a limited applicability. This is because actuators tend to be more expensive and heavier mainly due to the cost and complexity of the required seals pump, actuation valves and hydraulic fluid. The incompressible nature of viscous fluid however removes the potentially explosive characteristics of using pneumatic actuation in comparison. In addition, the relationship between the force applied via the hydraulic fluid on one side and the reaction of the actuators end effector on the other is linear, assuming the connecting hoses are tied down succinctly in order to minimise hysteresis effects. This simplifies the control aspects in comparison to pneumatic systems. Due to the actuator's lack of controllability in the event of a leak, as it does not have a brake, its use was ruled out for the actuation of any load bearing joints. It may however be useful for the precision control required when operating the drill.

Pneumatics

Pneumatic actuators have pistons for example that are able to react quickly due to the low viscosity of air, assuming any hysteresis in the air lines is negligible.

The control valves used to control them are typically solenoid based, with the control of the solenoid being achieved by regulating the current into the solenoid. As the current regulates the magnetic field and the open state of the valve, it can be regulated to allow the air pressure into the actuator to be regulated; this can be done using PWM. However this is none linear and as the piston draws air from the supply the pressure of the air supply changes as the piston moves. As the reaction of the piston depends of the pressure applied at both its ends this could be another source of none linearity further complicating the control.

It should also be noted that its rated operating pressure range is 0.15 to 0.8MPa, though how much pressure will be needed to operate the manipulator will not be clear until the design is more mature. It is possible therefore that its rating will be insufficient.

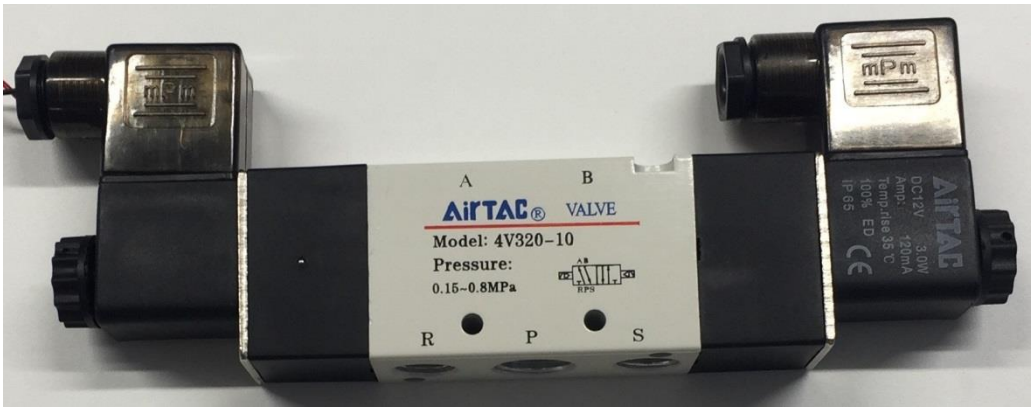

Figure 70, Generic pneumatic electrical actuation valve (Grindley, 2014)

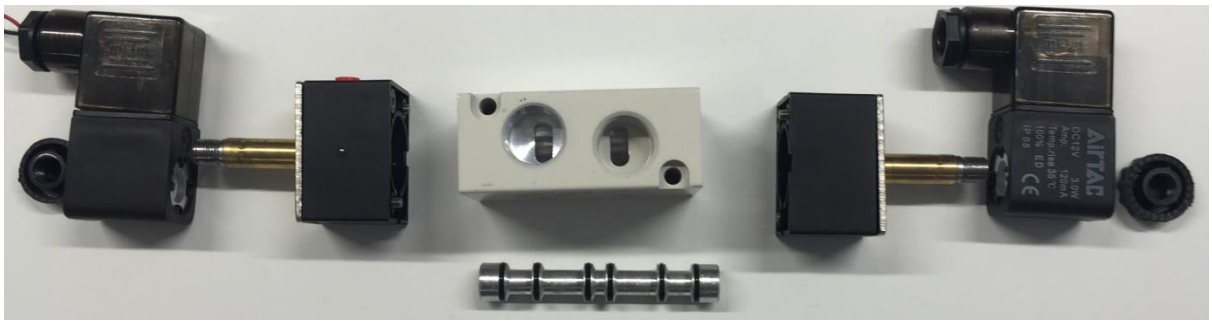

Figure 71, Generic pneumatic electrical actuation valve explosion (Grindley, 2014)

It can be seen in figures 74 & 75 that the operation of the valve relies on solenoids. The flux produced in the coils of the solenoids controls two valves which can be manually adjusted to regulate the conductance of the air path. In the centre there is a bar that moves depending on which solenoid is active, the relay and the bar both introduce lag into the valve's response. The valve also needs an air pressure differential between its input and output to actuate, so the rate of response is also a function of that pressure differential. The response of the valve to the control inputs is a concern however, in order to control the joins of the robot and stay stable, a suitable control loop frequency is needed. The valve would likely be suitable for a control loop operating at 50 Hz for example, though there were concerns relating to the slew rate of the pneumatic actuation system as a whole, so was discounted from the project at this stage. This was reinforced by the additional cost in terms of price of additional components such as the compressor, airlines, valves and storage capacity as well as the cost in terms of weight.

Altering the pressure on one or on each side produces a resultant force at the end of the actuator respective to the pressure differential. This is opposed to 1 way actuators that use a spring to produce a restoring force. The 2 way arrangement is a concern with respect to its modes of failure, as if one side should experience a leak, the pressure differential could change rapidly resulting in a rapid and potentially uncontrollable force being applied by the actuator. Pneumatics also requires both a pump and spare capacity, which added extra weight in addition to control aspects. As such the use of two way pneumatic actuators for load baring joints was dismissed over concerns that if one side experienced a leak the resultant uncommented actuation could compromise the overall safety of the system.

Electrical

Overall electrical actuation was deemed the most suitable for the system, largely due to price, size and availability. Firstly, control of electrical motors is relatively simple and in comparison to pneumatics or hydraulics there is no need for supporting hardware such as a pump, air hoses, electrically controlled valves and a breaking system. Electrical motors for the drive of load baring joints would require gears for high torque, that creates potential issues related to the meshing of the gears and hence accuracy. This is a key disadvantage of electrical motor driven actuation though the advantages of having the drive outputted directly to a joint outweighs the disadvantages of hydraulics and pneumatics, as the slew rate is reduced and the additional hardware required is minimal.. A geared DC brushed motor can be driven with H-Bridge for example. Such a simple configuration could allow for high speeds control loops, assuming minimal meshing in the gears connected to the motor, low motor time constant, low inertia and low H-Bridge switching time. This compared to a pneumatic system, as discussed above may have far greater time constants and other factors such as hysteresis in the air lines which may make there actuation more sluggish and nonlinear this is a key justification in the decision to use electrical drive.

The variety of self-contained electrical actuators available able to suit most instances in which they are needed, in the case of the load baring joints of the manipulator worm gear motors are most applicable. This is due to their failure modes; if the control electronics fail and electromechanical torque is lost then it would not lead to a collapse of the structure unlike pneumatic of hydraulic actuation if a leak were to occur.

A disadvantage of using electrical drive options is the way in which the mass of the motor is present at the location of the joint. With several joints making up the robot, this could make it heavy towards the front.

Electrical actuators pose issues such as weight to the forward end of manipulator and the meshing of gears which may produce a nonlinear response. Also, if a worm gear is used in the actuators construction, it will lock if the actuator fails, unlike hydraulic or pneumatic systems that could collapse without a breaking mechanism. When the whole system is taken into account, electrical actuators are inexpensive and elf-contained, unlike hydraulic and pneumatic actuators that require a pump or compression method in addition to associated control. This ultimately was the reason that electrical motors were chosen as the means of actuating the load baring structures in the manipulator.

Discussion as to how the joints could be actuated

Joint 1

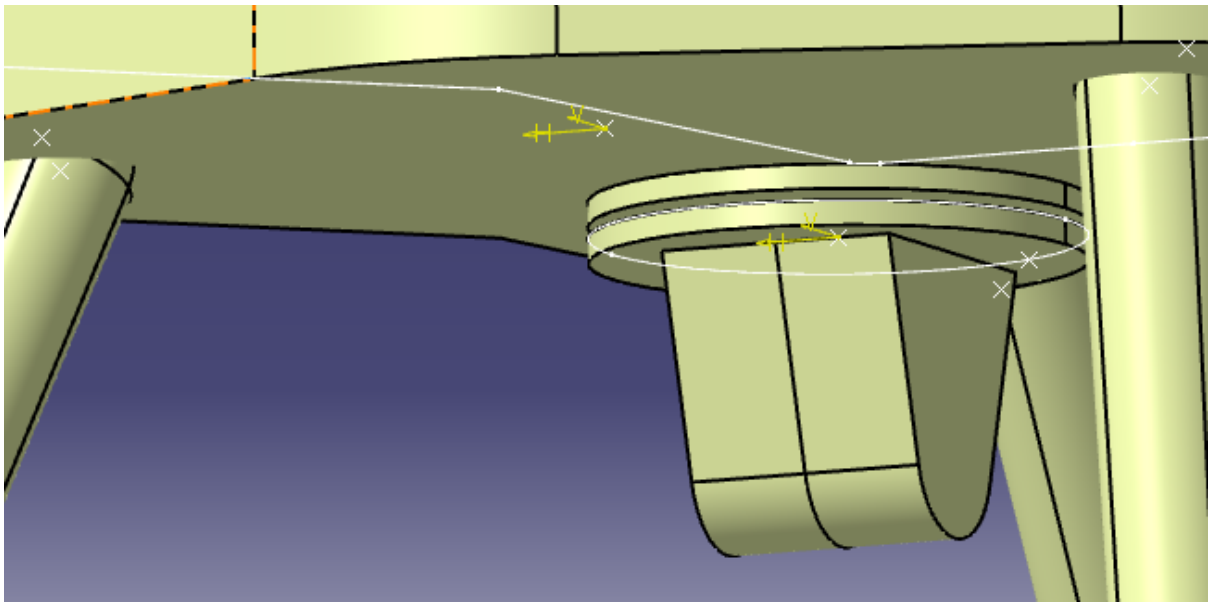

Figure 72, Joint 1 (Grindley, 2015)

Joint 1 shown in figure 76 shows a revolute joint, it is not required to fully rotate, but only move through an arc as the legs of the tripod also seen in figure 76 prevent it from doing so. The load in this case would be transferred through the bearings and a central shaft that would connect the two structures and allow them to rotate, time constraints prevented them from being characterised. As the loading does not transfer directly through the actuator using one that produces a lower torque, it is acceptable for it to be relatively low power compared to the others.

The mounting for the motor should be on the extremity of the two disks that are visible in figure 76, allowing the motor to generate the highest torque relative to the centre of the two disks about which the rotation occurs. The means of coupling the motor should be either done with a large cog arrangement or by using a set of gears and a chain similar to that used on a pushbike and Dragon Runner 20 robot (QinetiQ, 2015) in order to maintain a stiff interconnection. The motor recommended for this is a Power Window Motor (actuator number 3) which can produce a rated torque of 2.9Nm; its characteristics are displayed in table 2. Note the actuator requires feedback, as the motor does not have a means of resolving its position by itself the use of the CCD reader and rotary optical encoder should be used to provide it this is described in the chapter Electronics, FPGA sensor interfacing. Two more examples of suitable motors are listed in table 2, each with advantages and disadvantages of use. The High Torque DC Servo Motor in table 2 opiates as a geared servo motor with high torque, this has key advantages such as no backlash and an internal optical encoder to determine positon. This would simplify integration into the overall system; it's also light weight at 350g. Notable also is the Servo city, 1 RPM Gear Motor shown in table 2 (actuator number 5), it uses a brushed DC motor but is lightweight for the torque produced at a weight of 140g. Any of the proposed motors should be fit for the task.

Joints 2 & 3

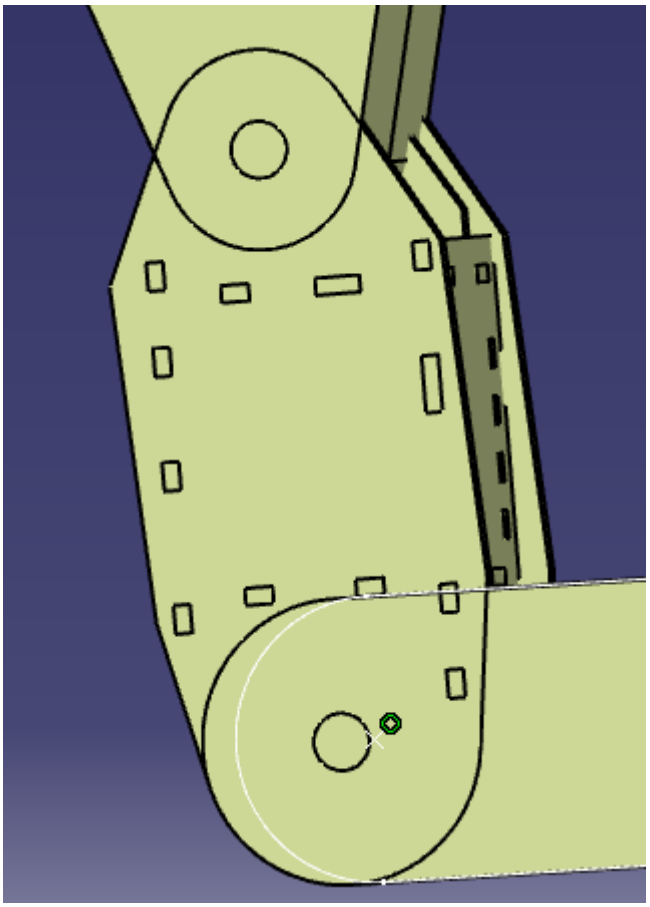

Figure 73, Joints 2 and 3 (Grindley, 2015)

The actuators used to drive joints 2 and 3 in figure 77 must have notable load baring capabilities, due to the them being located close to the centre of the manipulator meaning a substantial amount of torque may be generated at the points they pivot. Some linear actuators are well suited to this task; the actuators can be arranged in a push-pull configuration similar to a muscle in an arm. It is recommended that actuator 1 in table 2 be used in both cases this will give ample loading capability.

Joint 4

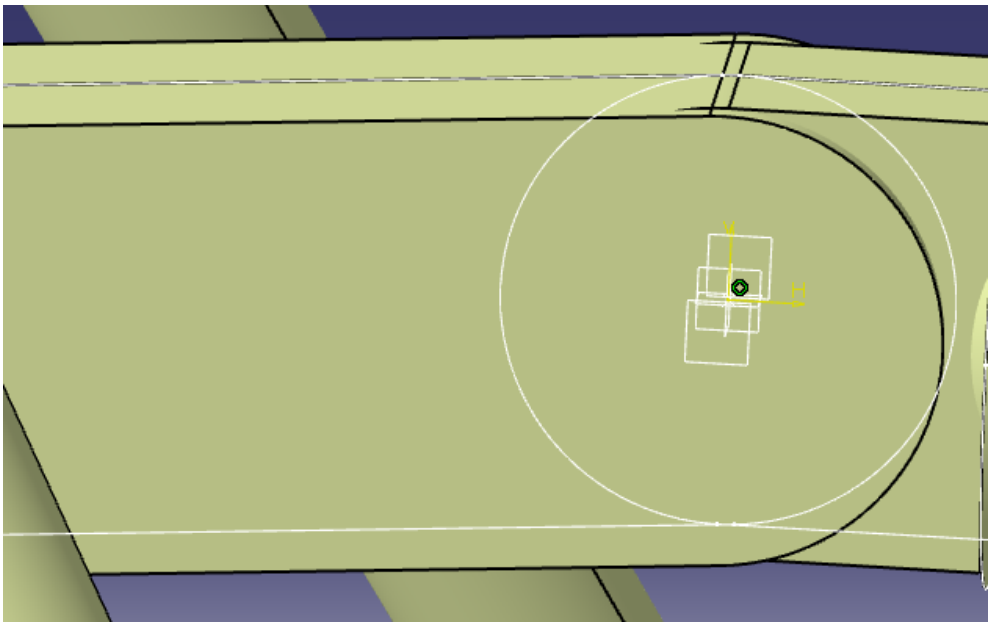

Figure 74, Joint 4 of the manipulator (Grindley, 2015)

Joint 4 shown in figure 78 is used to lower the front end of the manipulator to inspect underneath a device. Again similar to joints 2 & 3 a push pull arrangement similar to a muscle in an arm should be used, with the actuator mounted inside the structure below the centre. This arrangement would be best suited as the contraction of the linear actuator would lower the end of the manipulator without obstruction from the centre of the revolute joint. Actuator 1 from table 2 should be used to actuate this joint. Though its force produced is lower than the actuator 2 it is located at the end of the manipulator unlike joints 2 and 3, meaning that the torque required to produce movement should be lower. The position of the joint should be determined by an optical encoder described in chapter 5.

Joints 5

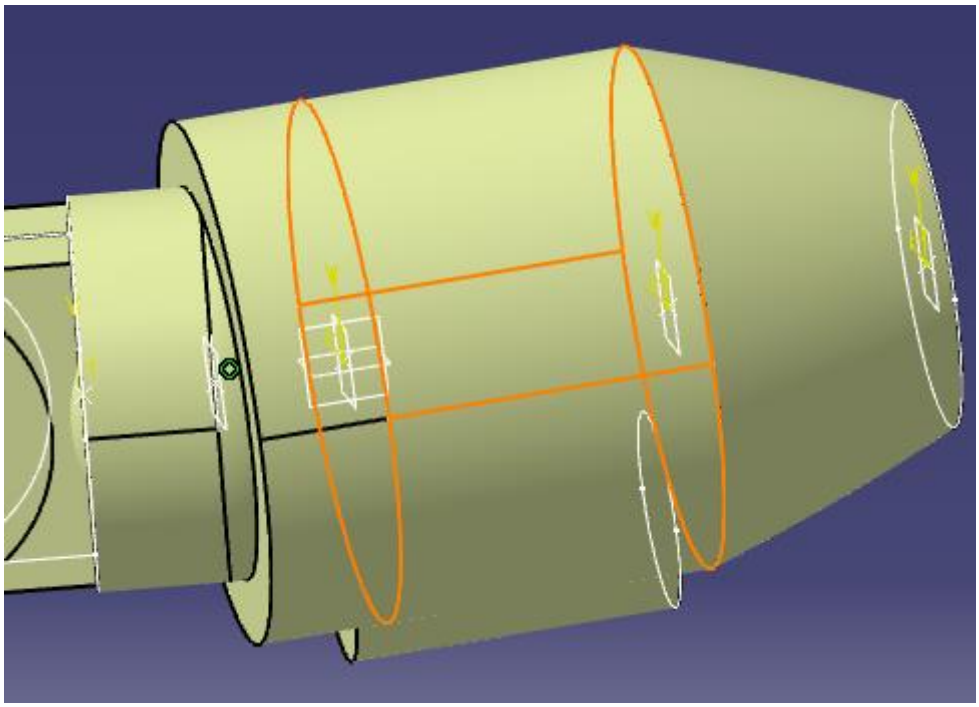

Figure 75, Joints 5 & 6 of the manipulator (Grindley, 2015)

Joint 5 in figure 79 is a spherical joint allowing for two degrees of freedom pitch and yaw. Compared with joints 2 and 3 the torque due to loading at the end of the manipulator is considerable less and as such actuators that and lower weight but produce less force may be better suited. Mounting them inside the end of the manipulator is recommended as the space in the link between joint 4 and 5 is very limited. This joint is spherical and encompasses 2 degrees of freedom, actuator 1 is recommended to actuate both of them. In terms of size and related limitations it does prompt the notion that redesigning the link to house the actuators may be recommended. This reinforces the notion that determining the loading of a manipulator and the actuators required prior to designing the structure is recommended for future iterations of the project.

Table of recommended actuators

Actuator number	Make, model and supplier	Rated output torque/ linear force	Rated input voltage and current	Mode of actuation	Velocity	Gear details	Weight
1	Progressive Automations, 2" Stroke 22lb Force Tubular Actuator	88.9N	12V, 9A	Linear, 50.8mm stoke	140mm/s	Not listed	1.4kg
2	Firgelli Automations, 2'' Stroke 150lb Force	606N	12V, 3A	Linear, 50.8mm stoke	10mm/s	20:1	Not listed

	Linear Actuator						
3	Cytron, Power Window Motor	2.9Nm	12V, 15A	Rotational, DC brushed	8.9rad/s	Not listed	Not listed
4	Rhino Motion Controls, High Torque DC Servo Motor 10RPM With Step/Dir Drive, Active-robotics.com	11.76Nm	12V, 7.5A	Rotational, Servo motor	1.05rad/s	1800:1 (calculated)	0.35kg
5	Servo city, 1 RPM Gear Motor, Active-robots.com	4.37Nm	12V, 95mA	Rotational, DC brushed	0.083rad/s	3000:1	0.141kg

Table 2, List of suitable electrical actuators

*All actuators shown are available from reliable suppliers

Chapter 5 - Electronics, FPGA sensor interfacing

A key facet of the design of the manipulator were the sensors and the control electronics; there were many options available both commercially and through open source distributions. The makeup of such a system should be dynamic and reconfigurable so as to not stifle any future development. With this in mind, a solution that was as reconfigurable as possible would be most desirable. As there would be a number of ways in which the proposed robotic solution could be upgraded and modified, a solution that would not require the control electronics to be replaced would be beneficial. FPGAs and System on chips were chosen early on as the most suitable option due to their highly customisable and reconfigurable nature. In this project an FPGA was used due to its availability for future development a System on Chip would likely be utilised as they would offer a single package solution. FPGA's were chosen despite the extra complexity that typically accompanies a simulated logic solution, noting also that control loops implemented on simulated hardware can't crash or suffer from run time errors unlike a solution directly implemented in ROS. As discussed in chapter 6, ROS was chosen as the best solution for complete system integration, simulation and control. ROS has an interfacing method known as ROS serial (Ferguson, 2015) which can be used by external systems such as the ZPU softcore to enable them to work as a node or service in ROS. This would allow ROS to directly read a sensor interface or command actuation through the FPGA using the ZPU softcore and wishbone interface.

During the initial stages of development, VHDL and C where chosen as the most ideal language for use when characterising the sensor interfaces. In the case of VHDL this was due to its natural systems based approach to design and also some prior knowledge of how to design logic with it. Another

advantage of using VHDL was the Matlab HDL coder, which can facilitate the design of complex systems in Matlab to be ported into VHDL with minimal development time.

Integration with higher level controllers was a key requirement, as an FPGA based solution was well suited for the interface between the sensors and the actuator, but it was not the best solution for control or the overall system integration due to the complex mathematical functions required. For example, inverse kinematics would require many fixed point mathematical operations and functions. That would need a larger resource allocation than is available on a low cost FPGA, in addition to limited support when working with human interface devices such as the Leap Motion.

A controller that supports a ROS compatible version of Linux such as Ubuntu ARM (Hendrix, 2014) was most suited for the higher level control, as ROS includes libraries for characterising inverse kinematic solutions, collision avoidance and route planning among others, this is discussed further in chapter 6.

To make it easier to interface the FPGA with a ROS system through ROS Serial, a microprocessor would be needed. It made sense to implement it as a soft core processor on the FPGA to ease integration and avoid any synchronisation and timing issues. Many soft cores were available from both open and closed source (OpenCores, 2015). Softcores such as the Xilinx MicroBlaze core and the AVR8 cores were easily available and widely used (Xilinx, 2014). The AVR8 is fully open source and works with the same instruction set at the Atmel AVR microprocessor product line. It is also fully customisable, but only has an 8bit address and data bus. Some of the components (for example the pulse counter described later) were 32bit, therefore it was decided not to use the AVR8 softcore, as it may add extra complexity when interfacing with the hardware. Another choice was the Xilinx MicroBlaze processor which is 32bit and hosts a range of peripherals and customisation options; however it is not open source. The MicroBlaze softcore (Xilinx, 2004) required proprietary software from Xilinx to be bought in order to allow for full customisation and utilisation. Xilinx also requires a royalty be paid for its implementation. It would have been the best option given that its IDE is well developed with comprehensive libraries and debugging facilities, but it was discounted due to the royalties and the cost of additional software which Xilinx does not provide for free.

The most flexibility came from the ZPU soft core processor. It uses the same Atmel AVR instruction set that Arduino is based on, it is also employs a 32bit address and data bus and a 32bit wishbone bus for custom peripherals. It is customisable though not to the same extent as the AVR8. The ZPU softcore can have as many GPIO's, UARTS, watchdog timers and Interrupts as required, assuming that the FPGA used has the resources for implementation. The processor is also tested up to and speeds of 100MHz.

The use of a softcore microprocessor was seen as an ideal way to interface the electronics with the FPGA hardware designs and ROS. In addition to this a ROS library is available as open source to operate within the Arduino environment though time constraints prevented it from being tested hence it has been omitted. As ROS has a set of libraries written in ISO C++ some of them within reason can be ported over and used on an Arduino. As the ZPU's architecture is based on the AVR's and most of the syntax is the same with a few hardware dependent exceptions such as the use of native ADC's available for use on an Arduino. This was seem as a major key advantage of using the ZPU softcore, the Adriano's peripheral libraries were also available to be used to on the ZPU making it an even more powerful solution. One key drawback was having to use of the Arduino environment to program it, being typically considered less professional in due to the lack of advanced debugging tools. The Eclipse

IDE (Eclipse, 2015) can be used instead, which allows for the same libraries to be used alongside full debugging support. Other libraries were also available and fully tested for sensor interacting, for example the BMP085 Barometric pressure sensor in addition to libraries for stepper motor control.

The family of FPGA's that proved most applicable were the Xilinx Spartan-3E family due to their low cost in addition to having ample resources for the purpose of this project, they were also available for use at no cost so this was a key justification in the decision to use this hardware. An abundance of examples and support is available for the Xilinx XC3S500E. Its specifications include 500,000 system gates, 10,476 logic cells for logic simulation and 116 IO pins which made it a cost effective solution both in terms of price and flexibility. In addition to this, many examples exist in which the ZPU softcore processor was implemented on it, thus facilitating its development.

To conclude, using an FPGA of SoC in conjunction with the ZPU soft core processor and the ROS embedded nodes originally designed for Arduino allows for fully customisable hardware in addition to simplified integration with ROS. At the onset of this project no examples or documentation relating to use of FPGA's and ROS existed online, arguably making this system the first of its kind.

Custom sensing solutions

A suit of sensors was needed for the project. In terms of the odometry, the manipulator requires a means of determining the positions of both its rotary and prismatic joints. In order to keep the sensor implementation simple, it was necessary to investigate a solution that could be re-used in a variety of ways that would ideally suit all of the odometry needs of the manipulator.

As one of the project objectives was to keep costs low, readily available and well tested components should be used where applicable. As the manipulators accuracy can only be as good as the accuracy off the odometry data being used for feedback the decision was made early on to find a means of sensing position that could provide the greatest level of precision at the least cost. A reusable technology that could be used to determine the position of a number of different rotational and prismatic joints was explored.

Contemporary measurement techniques for displacement fall into two main categories: absolute and incremental, where absolute techniques indicate the absolute position directly within a given error and incremental techniques give position relative to a known reference point and number of points passed since.

In the case of joint positions, the exact position must be known. Incremental techniques require the control system to locate the reference point or have prior knowledge of its location before any measurement can be considered valid. A typical example of an incremental sensor would be a roller ball mouse which uses a mono coded wheel with two photo interrupters, or a camera using optical flow to ascertain displacement with along a vector. In the application of a computer mouse a reference from the measuring surface is not needed. For a robotic joint with a finite arc or extension however, it is imperative to know its position at any instance, making incremental sensing techniques undesirable. Absolute encoders require a way in which positions can be individually identified. This requires the area that is being read (a strip or disk for example available in Appendix optical encoder section) to have a unique code identifying each known position; this makes their design and implementation potentially more complex. In addition to this reading, the coded area requires error correction and some level of redundancy to mitigate against any damaged parts of the coded area or

against ambiguous readings. Grey coding of the area and the checking of current readings against possible predicted readings was deemed a suitable solution, due to its simplicity and ease of implementation. This was chosen as grey coding is widely used in encoding methods particularly optical encoding. This is due to there being only one change between each element and its neighbouring element, hence making error correction far easier.

In a case where displacement sensors were used and the manipulator's control and sensor electronics were reset, the manipulator must move each joint until a reference is found in order to ascertain the current position of its joints. This is an unacceptable situation however in the case of this manipulator. In this case absolute position sensing techniques should be used to determine displacement of the revolute and prismatic joints.

Capacitance

Capacitive displacement sensing was also explored. Originally some research was done on capacitance digital Vernier callipers. Capacitive methods of displacement measurement can be used to measure very small displacements, though are typically limited to small measuring distances and require regular recalibration; the surface it is measuring must also be conductive. Depending on the way they are implemented a position reference will be required, which limits the usefulness of capacitive displacement sensors for this application. However their high accuracy could make them useful when measuring small displacements such as that of the drill moving forwards and backwards into the mine, and example of such a sensor would be the cylindrical sensors provided by (micro-epsilon.co.uk, 2015) in figure 80.

Figure 76, (micro-epsilon.co.uk, 2015)

IR reflectance sensors

A range of simple reflectance sensors such as the Fairchild QRE1113 were also considered. There are two ways in which these sensors can be made useful (Fairchild, 2015).

The first method of using this sensor that was explored this, was to measure the distance between the sensor and a reflective surface as it is intended to be used. As the sensor is Infra-red based, it can drift to some extent depending on the ambient temperature and degradation of the IR LED, though this drift may be mitigated by having reconfigurable bios resistors controlling it. It can also be susceptible to dirty environments and as such was not further researched.

The second method utilises an optical encoder and a slit disk such as the one found in the Appendix. The slit disk must be placed in front of each sensor to reduce the area it can read from. When several sensors are used together, the code forms an absolute optical encoder that can be read, though as discussed later, it was decided that a CCD reader would be better suited.

Other incremental displacement sensors are available, such as incremental linear sensors available from (Renishaw, 2015), though these can only be made to work if the manipulator design allows for the sensor's profile. For example with the (Renishaw, 2015), a tape with indentations accompanies the sensor. This must be mounted with the sensor in a way that ensures that the tape is always in tension. Such a design would require a spring that in turn would introduce a mass spring damper configuration into the odometry measurements. For this reason, this sensor was therefore discounted. Another though more complex optical solution was available; Linear CCD readers are cheap and have a high resolution suitable to optically read a coded surface. The Linear CCD reader chosen was the Toshiba TCD1304AP (Toshiba, 1997) this was because it has 3648 readable pixels within a line of 29.1mm which is ample for reading a coded area such as an optical encoder, so it was decided that it had appropriate resolution for the task. When compared to other Linear CCD's it was cheap and usually available for under £10 and only requires era simple sequence of signals to drive it. Another justification for its use was that it responds to light at a centre frequency of approximately 550nm (green/yellow). The datasheet suggests that its response due to ambient Infra-red may make it less susceptible to noise this is detailed as between 80% for near Infra-red and 20% for far Infra-red (Toshiba, 1997).

Figure 77, Fairchild QRE1113 reflectance sensor (Fairchild, 2015)

Simulated hardware on FPGA

Pulse duration counter

A pulse duration counter was designed for the drill, as detailed in chapter 1. Both the speed of the drill and any changes in its speed could be used to determine when the tool contacts the target surface and it's a safe feed rate. The maximum rate of rotation of the drill was to be approximately. 400,000 RPM (6666 Revolutions per second) unloaded. The air turbine itself has 8 blades making the approximate maximum number of pulses per second 53,328. The counter was designed to detect the number of clock periods taken for the each blade to pass the sensor from this; its speed can be easily derived. The Spartan-3 FPGA development board had a 28MHz clock. By virtue of a clock multiplier the 96MHz clock used by the ZPU softcore was produced. This was deemed suitably fast for the

counter to give adequate resolution for speed calculations, as shown in the calculation below. The signal characterising each event in which the blade passes the sensor encompasses one cycle going from logic 1 to 0 and back to logic 1. The counter can determine either the time period spent between cycles, depending on how it is configured. This is illustrated in the equation below.

$$\text{Number of clcok periods}_{\min} = \frac{\text{ClockSpeed}}{2 * \text{NumberOfBlades} * \frac{\text{RPM}_{\text{Max}}}{60}}$$

$$\frac{96 * 10^6}{2 * 8 * 6666} = 900 \text{ clock periods}$$

A theoretical minimum of 900 clock periods was deemed accurate enough to very quickly detect a change in speed, 900 is the minimum number of clock periods that should be observed at full speed making a small chance easy to detect. The maximum RPM of the drill is specified as 400,000 RPM under unloaded conditions, when loaded even slightly it is expected that it will decrease significantly and sharply upon contact with a surface. This effect may be used as a means of automatically judging the feed rate of the drill, through experimentation a safe speed that relates to the most efficient feed rate can be determined.

The high frequency update rate and the accuracy of the pulse duration counter used to measure the drill rotation speed means that the is redundancy in the data provided to the control loop that controls the feed rate of the drill, this sample rate should be ample to satisfy the nyquist criterion. The frequency will vary from its maximum of approximately 53,333Hz downward with respect to load, an initial estimate of 1kHz should be sufficient to ensure sufficiently fast control responses and also that the control loop will not run faster than the data rate of the sensor.

The ZPU softcore could be configured to carry out this task, however it would require interrupts and hardware counters in order to take into account any time incurred to service the interrupt routines while interfacing with ROS or other tasks. This reinforced the justification that developing some control loops and systems in simulated hardware on an FPGA may be the most optimal and elegant solution. That does not require a great level of understanding of the processing resources on a microprocessor or softcore processor. Note that the resolution is increased at lower rates of rotation.

Figure 82 below details the basic circuit proposed for pulse counting to determine the rate of rotation of the drill.

The LED control input determines whether the LED is on or off, note that due to the high frequency of the FPGA, it should not be pulsed. The photo diode bias is controlled by a 5kΩ digital potentiometer R2 (MCP4141-502E/P) to allow the light level that causes the comparator to trigger at to be changed through software on the ZUP via an SPI peripheral on the wishbone bus. In order to calibrate the sensor and account for any change in the ambient temperature. A 3.3V zener diode D1 is also included on the output of the circuit for additional over voltage protection. The output will be connected to a comparator U1A (MCP6541) an additional digital potentiometer R3 MCP4141-502E/P is included to allow the triggering voltage on the positive side of the comparator to be modulated. Figure 82 shows an example of the circuit, digital potentiometers R2 and R3 are not available in the Multisim libraries and have been illustrated by potentiometers note they are SPI controlled devices. The circuit is driven by 5v but the input to the FPGA is 3.3v, because of this the comparator is supplied with 3.3v by the

fixed voltage regulator U2 (LP2951ACN-3.3G). So that the zener diode does not become un-necessarily loaded when the comparator outputs logic high. The datasheet specifies that the maximum voltage produced by the photodiode is 1.2V leaving 3.8V to fall over the bios resistor, as the current is 1mA through it its resistance should be 3.8kΩ as such a 5kΩ digital potentiometer was chosen to allow this to be tuned if necessary.

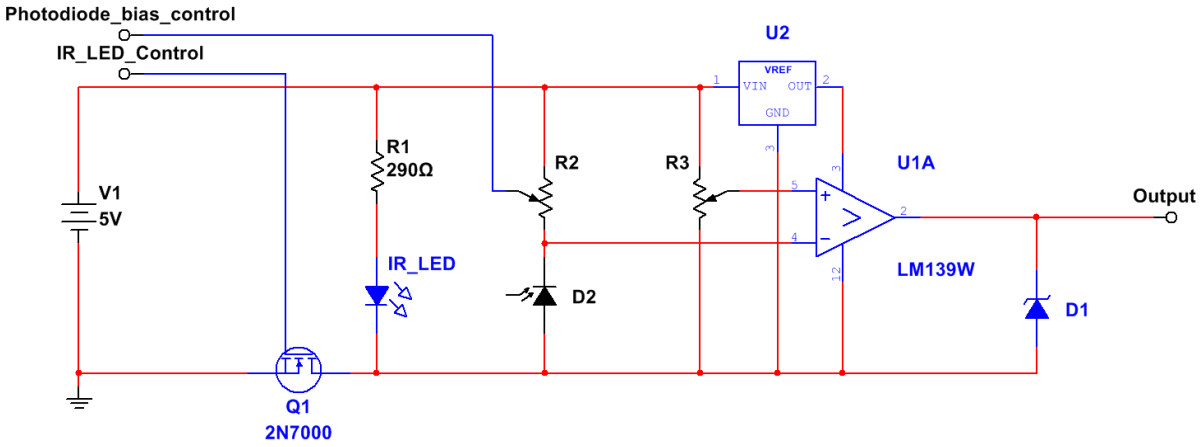

Figure 78, Basic photo interrupter circuit (Grindley, 2015)

Key components recommended for the circuit are

- OFL-3102 940NM IR LED (Fairchild semiconductor, 2015)
- OP906 photo diode (Multicomp, 2015)
- MCP4141-502E/P digital potentiometer 5kΩ (Microchip, 2015)
- MCP6541 comparator (Microchip, 2015)
- LP2951ACN-3.3G fixed 3.3v voltage regulator (On semiconductor, 2015)

As described above the system that determines the number of sample clock time periods between each cycle in which the Photo diode output is at logic 1, which is used to determine the rate of rotation of the drill. Initially as Matlab and Simulink were much more familiar development environments in which to work in, it was decided to initially use Simulink to design the counter that would count the number of time clocks periods between cycles. This entailed using an integrator with FI numbers; however the HDL coder in Matlab does not have any native compatibility with the wishbone interface which made its use as a tool to speed up the rate of development less useful. This meant the possibility of integrating the code generated from Simulink using the HDL coder was abandoned in favour of writing the whole system in VHDL, using a wishbone interface template

A simplified flow diagram indicating the way in which the counter works can be found in figure 83 below.

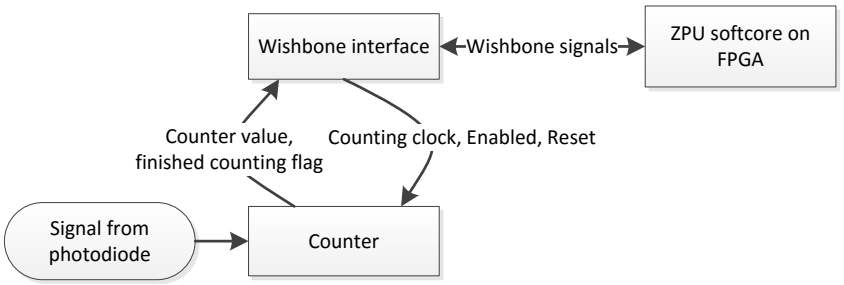

Figure 79, Illustration of wishbone counter opporation (Grindley, 2015)

The full code for the counter can be found in the HDL code section of the appendix. Note the counting clock, Enable and Reset signals are generated by the wishbone bus at the command on the ZUP softcore.

The counter was designed to be 32 bit; a larger number of bits were not needed as a 32 bit counter would overflow in 44.7 seconds if a clock frequency of 96MHz is used to increment it. The system comprises five synchronous processes.

An asynchronous acknowledge response for the wishbone interface and a counter overflow flag are also included.

```
-- Asynchronous acknowledge
  wb_ack_o <= '1' when wb_cyc_i='1' and wb_stb_i='1' else '0';
-- Asynchronous output overflow flag
  register2(0) <= CounterOverflow;
```

The first process increments the counter upon the counter clock's rising edge. It takes signal inputs as Counter Clock, Counter Overflow, Wishbone reset and also a flag to indicate if the counting process is complete. It is also responsible for indicating if the counter has overflowed and sets a flag as a signal to logic 1 should this happen. Firstly the process checks if the reset flag has been set by the wishbone interface or if the flag indicates that the counting process is complete, the reset signal comes from the wishbone interface and is triggered by the ZPU as well as Enable signal to enable the counting process. CounterProcessDone flag is set by the counter and instructs the process to copy the final counter value to a register in the wishbone interface to be read by the ZPU. Should this be the case, it resets the counter to 0. If not, it increments the counter, then checks if an overflow has occurred, setting the overflow flag if required.

```
--Counter process
   process(CounterClk, Counter, CounterOverflow, wb_rst_i, CounterProcessDone)
   begin
    if rising_edge(CounterClk) then
       if wb_rst_i='1' or CounterProcessDone = '1' then
         Counter <= (others => '0');
       else
         --Counter <= Counter + 1;
         Counter <= std_logic_vector( unsigned(Counter) + 1 );
         if Counter(31 downto 0) = "11111111111111111111111111111111" then --Detect Overflow
            Counter <= (others => '0');
            CounterOverflow <= '1';
         else
            --Counter <= Counter;
            CounterOverflow <= '0';
         end if;
       end if;
   end if;
   end process;
-- End counter process
```

The second process takes the signal from the photodiode and counter as inputs. If the signal input from the photo diode circuit in figure 82 is in a logic 1 state and the flag that indicates the counting process is complete in logic state 0, it sets a flag to indicate that the process has finished counting and copies the current counter value into a register used as a buffer. If it is not complete, it checks if the signal from the photodiode is in logic state 0 before setting the flag that indicates if the counter process is at 0. This prevents the counter value from being copied more than once upon one instant of the photo diode signal becoming logic 1.

```
--Input detect process
   process(CounterSignal, Counter, CounterProcessDone)
   begin
   if (CounterSignal'event) then
       if CounterSignal = '1' and CounterProcessDone = '0' then
         CounterProcessDone <= '1';
         FinalCounterValue <= Counter;
         --Counter <= (others => '0');
       else
         if CounterSignal = '0' then
            CounterProcessDone <= '0';
         end if;
         --register0 <= register0;
         --Counter <= Counter;
       end if;
   end if;
   end process;
--End Input detect process
```

The third process is in control of the IR LED it takes element 0 from Register 1 in the wishbone bus and outputs it directly to the LED signal port, which will be in turn connected to the LED drive control pin of the FPGA. It does this on the rising edge of the same clock the counter uses to increment. It should be noted that PWM or other pulsing techniques should not be used to control the LED as this would cause errors when determining the turbine's rotation period.

```
--Control the LED
  process(CounterClk)
  begin
      if rising_edge(CounterClk) then
         LEDcontrol_out <= register1(0);
      end if;
  end process;
--End LED control
```

An asynchronous multiplexer is used to address each wishbone register, the ZPU and counter process can address any of these registers. In this implementation, Register 0 is used to store the 32 bit counter value and element 0 of Register 2 is used to check if the counter has overflowed. This is to be read by the microcontroller. Element 0 of Register 1 is used for control of the IR LED; the remaining elements in Register 1 and register 2 are redundant. Note this is the asynchronous means by which the wishbone registers are read by the microcontroller.

```
-- Multiplex the data output (asynchronous)
process(register0,register1,register2, wb_adr_i)
begin
  -- Multiplex the read depending on the address. Use only the 2 lowest bits of addr
  case wb_adr_i(3 downto 2) is
    when "00" =>
       wb_dat_o <= register0;  -- Output register0
    when "01" =>
       wb_dat_o <= register1;  -- Output register1
    when "10" =>
       wb_dat_o(31 downto 0) <= (others => '0'); -- We put all upper 24 bits to zero
       wb_dat_o(7 downto 0) <= register2;        -- since register2 only has 8 bits
    when others =>
       wb_dat_o <= (others => 'X'); -- Return undefined for all other addresses
  end case;
end process;
```

The process below controls the interaction between the wishbone interface and the other processes that make up the counter system. If the wishbone interface issues a reset command Registers 0, 1 and 2 are cleared to '0' respectively. If the wishbone bus indicates that it wants to access one of the registers "wb_cyc_i='1'" and the bus is stable "wb_stb_i='1'" and that it wishes to write to a register "wb_we_i='1'" then it uses "wb_adr_i" to select the register and write to it. Otherwise if the counter process flag indicates that the counter has finished, it writes the counter value to Register 0, ready to be read by the microcontroller via the wishbone bus.

```
process(wb_clk_i)
begin
  if rising_edge(wb_clk_i) then  -- Synchronous to the rising edge of the clock
    if wb_rst_i='1' then
       -- Reset request, put register1 and register2 with zeroes,
       -- put register 3 with binary 10101010b
       register0 <= (others => '0');
       register1 <= (others => '0');
       register2 <= "00000000";
    else -- Not reset
    -- Check if someone is writing
       if wb_cyc_i='1' and wb_stb_i='1' and wb_we_i='1' then
         -- Yes, it's a write. See for which register based on address
          case wb_adr_i(3 downto 2) is
           when "00" =>
             register0 <= wb_dat_i;--wb_dat_i;  -- Set register0
           when "01" =>
             register1 <= wb_dat_i;  -- Set register1
           when "10" =>
             register2 <= wb_dat_i(7 downto 0); -- Only lower 8 bits for register2
           when others =>
             null; -- Nothing to do for other addresses
          end case;
        else
          if CounterProcessDone = '1' then ---Added
                register0 <= FinalCounterValue;
          end if;
       end if;

    end if;
  end if;
end process;
```

CCD Reader

A sensor that could be used for multiple purposes was desirable for example a single sensor type that could be used to determine all joint positions revolute and prismatic. Other sensors such as the IR reflectance sensors discussed above have limited use as they would require a small aperture to be placed in front of the sensor to allow it to read small features of an optical encoder. An example of such a small feature is illustrated in the optical encoder section of the appendix. A rotary encoder is of course most useful for measuring the angle of rotation, the example can also be made to be linear to measure the displacement of the prismatic joints opposed to circular to measure revolute joints.

It was decided that the Toshiba TCD1304AP (Toshiba, 1997) is an appropriate sensor to measure the absolute position of an optical encoder for revolute joints as well as linear displacement for prismatic joints. As discussed above it has 3648 readable pixels which is ample for the task, it is also cheap and requires timing signals that are relatively easy to produce it also has a maximum refresh rate of 270 lines per second. However it was difficult to interface with an FPGA because no such VHDL libraries exist (Toshiba, 1997). The advantages of having one cheap, readily available sensor that could be made to produce reliable absolute position data made it desirable and warranted the time spent developing a means of interfacing the sensor with the FPGA development board.

Two methods of interfacing were proposed. As interfacing with the sensor in terms of control signals was digital and required the use of state machines, state space within Simulink/Matlab was used initially in conjunction with the HDL coder to produce the VHDL code to interface with it. As no distinct examples of producing state machines and timing signals for use on FPGA's was available, this task was difficult. This aside, it nevertheless appeared easier than implementing it directly in VHDL.

The datasheet of the TCD1304AP (Toshiba, 1997) was first examined to determine the number of states needed for the state machine that produces the timing signals to drive the linear CCD reader. The sensor has two modes one in which the photo electric shutter activates upon reading every other pixel and another where the photo electric shutter activates once per line read out. If the shutter opens several times while the line is read out (first mode) then the time stamp for each pixel would be skewed relative to the next, so it was decided that having the same time reference for the whole line would be less likely to introduce errors into the readings. The timing diagram to achieve this can be found in figure 84, note the shutter marked SH is at logic 1 at the beginning and end of the sequence but at logic 0 during readout.

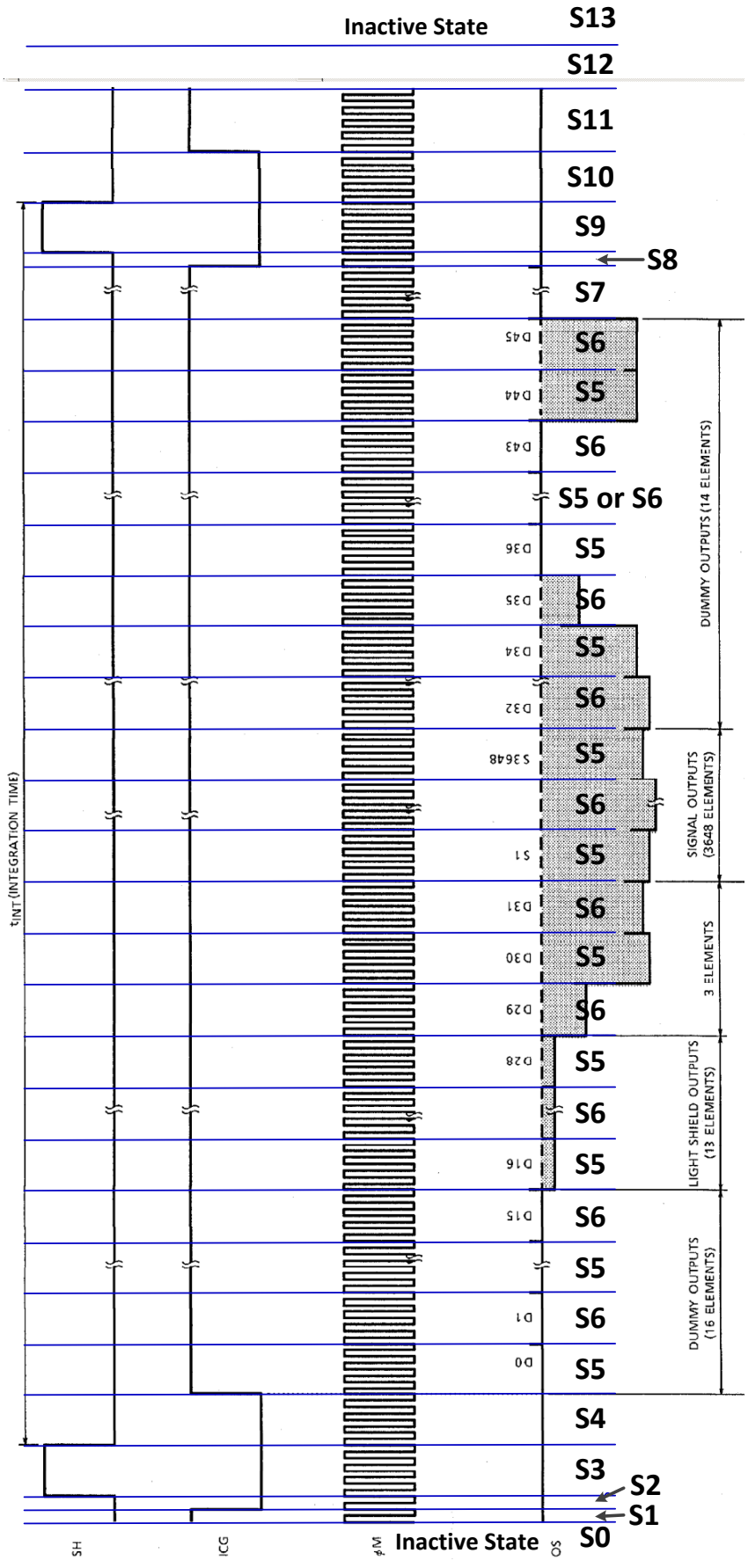

Figure 80 Timing diagram TCD1304AP (Toshiba, 1997)

The generation of the timing sequence was produced using a state machine. This utilises 14 different states including 0. Figure 84 shows how they were determined note that the automatic shutter is disabled in this case as it is the mode that is recommended for use in this project. The section marked S5 & S6 indicates many different transitions to read out the 3648 pixels and dummy pixels, this cannot be shown in the diagram as it would be too wide to display. The different states are briefly explained in the table 1 below. Note that states such as S0 and S13 appear to have no purpose however they are inactive states in which the control outputs are set to logic 0 and determine whether or not the CCD reader implementation is awaiting an enable signal logic 1 in the case of S0 or a reset signal logic 1 for S13 to proceed.

The outputs of the CCD controller to the input to the Linear CCD were ICG (Integration clear gate), SH (Shutter control) and $\emptyset\mathrm{M}$ being the clock input. The maximum clock speed of $\emptyset\mathrm{M}$ is 4MHz, with 4 clocks per pixel (Toshiba, 1997). As total of 3694 elements are to be read including dummy pixels, 11 clocks are needed at the start and 14 at the end sequence states S0 to S5 and S7 to S13 respectively as seen in table 1. Meaning the maximum refresh rate is approximately:

$$\text{CCD Refresh rate}_{\text{Max}} = \frac{4 * 10^6 \text{Hz}}{(14 + 11) + (3694 * 4)} = 270 \text{ lines per second}$$

As an initial assumption a control loop that operated at 50Hz would be suitable for the control of each joint in most circumstances, this indicates that a sampling frequency of 270Hz should be ample for the sensing needs of the joints in the manipulator.

State	Notes	Number of $\emptyset\mathrm{M}$ clocks until next state	Next state	SH	ICG
S0	Initial inactive state, set all control outputs to be logic 0	1	S1	0	0
S1	First initialisation state, set respective control outputs	1	S2	0	1
S2	Second initialisation state	1	S3	0	0
S3	Third initialisation state	4	S4	1	0
S4	Forth initialisation state	4	S5	0	0
S5	Reading pixel value (Increment counter to determine current pixel element)	4	S6	0	1
S6	Reading pixel value	4	S5 if pixel element is less than 3694, S7 if pixel element is equal to 3694	0	1

S7	First lead out state pixel values no longer valid	4	S8	0	1
S8	Second lead out state	1	S9	0	0
S9	Third lead out state	4	S10	0	1
S10	Forth lead out state	4	S11	0	0
S11	Fifth lead out state	4	S12	0	1
S12	End of sequence	1	S13	0	1
S13	End of sequence, set all control outputs to be logic 0	∞	None, S0 if wishbone reset or auto reset signal signals are logic 1	0	0

Table 1, States for reading Toshiba linear CCD

Matlab HDL implementation

When developing with the Statespace and the HDL coder in Matlab, it was not initially clear where to start. No examples of designing hardware interfaces with Matlab, or Statespace and the HDL coder were available, so everything was started anew. This was initially favoured because systems developed in Matlab could be simulated in Matlab first, it was anticipated that the rate of development could be increased by using Matlab even though little guidance was available. This was a key justification for using the HDL coder in Matlab to produce VHDL for the FPGA. Firstly the outputs and signals used within the design must be compatible with the STD_LOGIC_VECTOR type in VHDL; by default double precision floating point numbers are native in Matlab, which produce REAL number data types in VHDL. To avoid this FI (Fixed integer) numbers in Matlab were used; this brought associated problems that were discovered later. Typical math operators such as +, -, / and * that can be overloaded to work for different data types in Matlab, but cannot be used in the conventional sense when working with FI numbers. A different set of commands such as accumneg (accumulate negative -) and accumpos (accumulate positive +) specific to IF numbers was required instead.

After some experimentation with the settings within the HLD coder it was found that an extra clock for the system was needed, this unintentionally complicated the design because two clocks were effectively needed for the implementation. One required to drive the logic and ØM to control the flow between states. After a number of design iterations as it did not work right away, the following design was constructed. An overview of this can be seen in figure 85, note that due to its size and Simulink's means of displaying the diagram finer details cannot be seen.

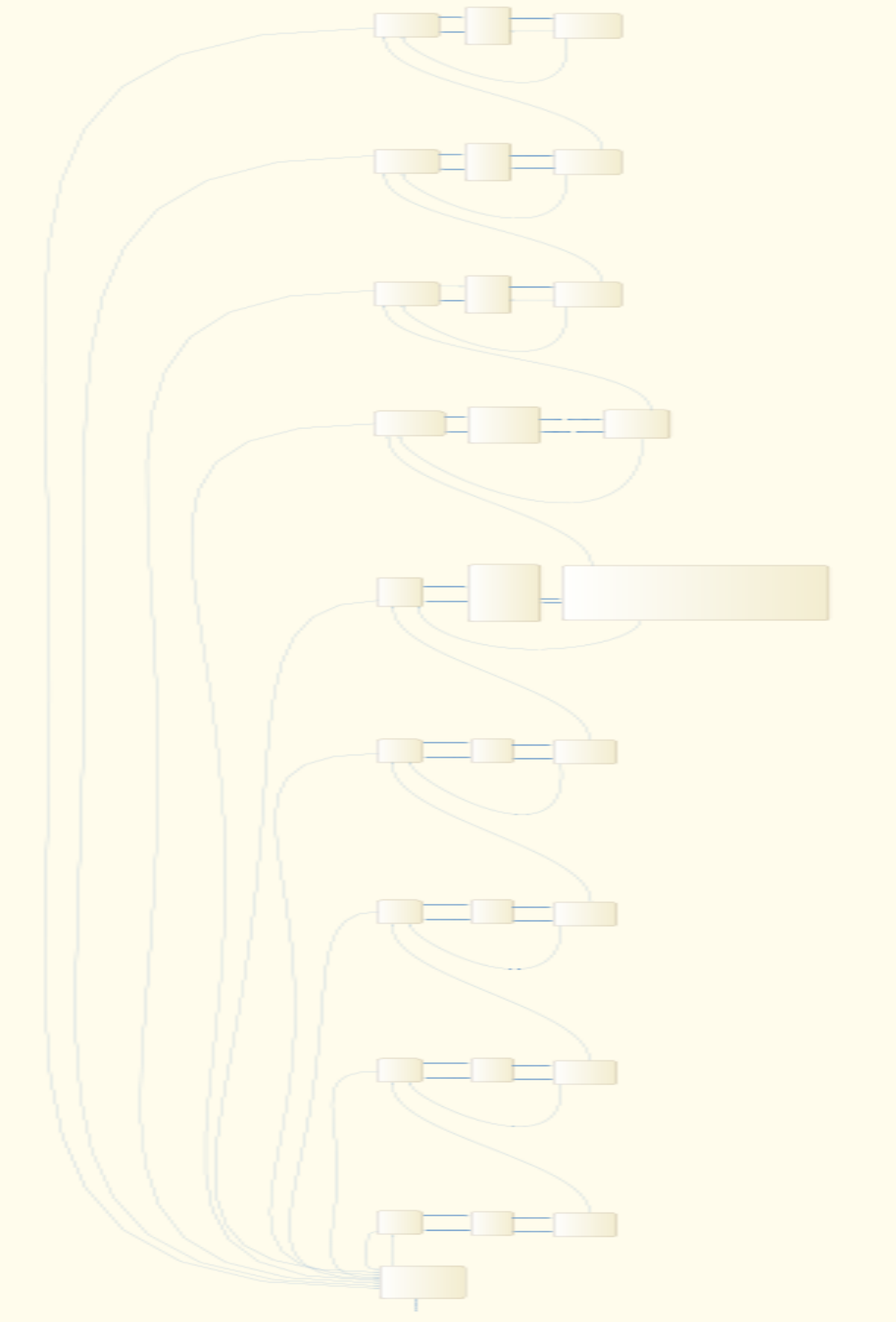

Figure 81, Overview of Statespace implementation (Grindley, 2014)

Figure 68 shows the initial state followed by the second. The initial state waits for the input "Enabled" to become Logic 1 and ØM (denoted as "TIC_TOC" in the design) to be logic 0 before starting the read sequence.

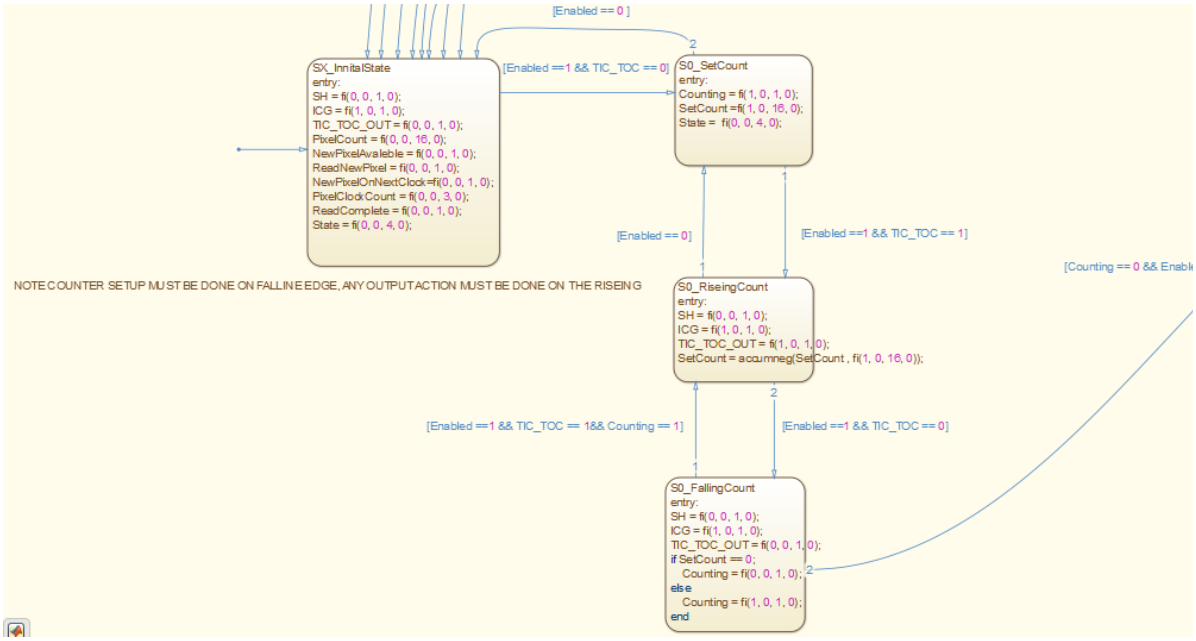

Figure 82, Close up of Statespace table (Grindley, 2014)

The top state is marked S0_SetCount the states below are marked S0_RiseingCount and S0_FallingCount respectively which hold the outputs SH and ICG at the same logic level until the correct number of ØM clocks has occurred, in the case of State 0 this is two when reading the pixel values this is set to 4 in accordance with the timing diagram this because each pixel required 4 ØMclocks to read it out and state 0 requires two ØM clocks to transit to the next state.

There are four initialisation states including 0 before the Pixel values are read, as there are 3694 elements to read state 4 loops 3694 times, before moving to the final states. This is illustrated in figures 87 and 88. Some outputs were also needed to indicate the current element being read or addressed and some control outputs to indicate when a pixel value was valid and should be read; others indicating when a new element was available to read. This was done to synchronise the data going into the RAM this also means that an ADC could also be used in conjunction with the CCD reader where the ADC could be set to start the conversion process when data valid is triggered. New pixels on next clock were added to assist with buffer control but later removed. These control outputs were denoted as 'PixelCount', 'NewPixelAvaleble', 'ReadNewPixel' and 'NewPixelOnNextClock'.

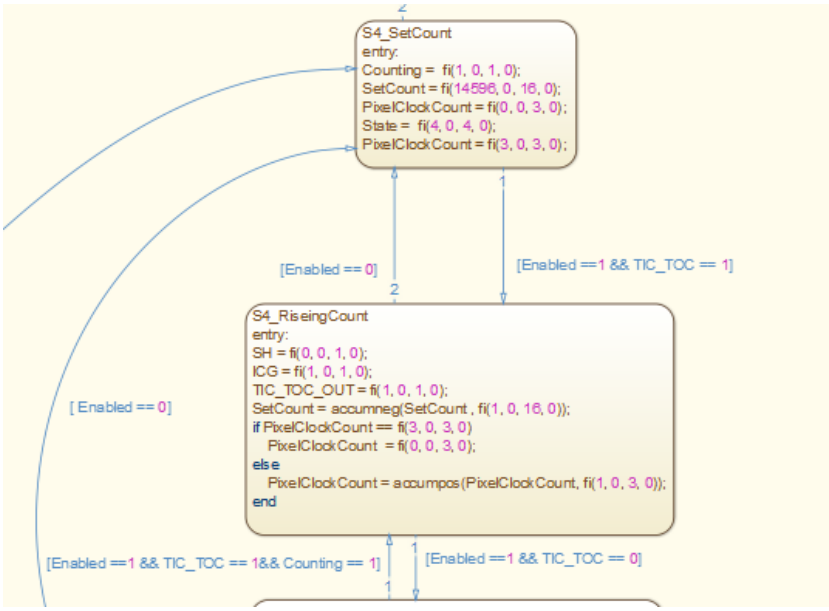

Figure 83, Zoomed view of Statespace blocks (Grindley, 2015)

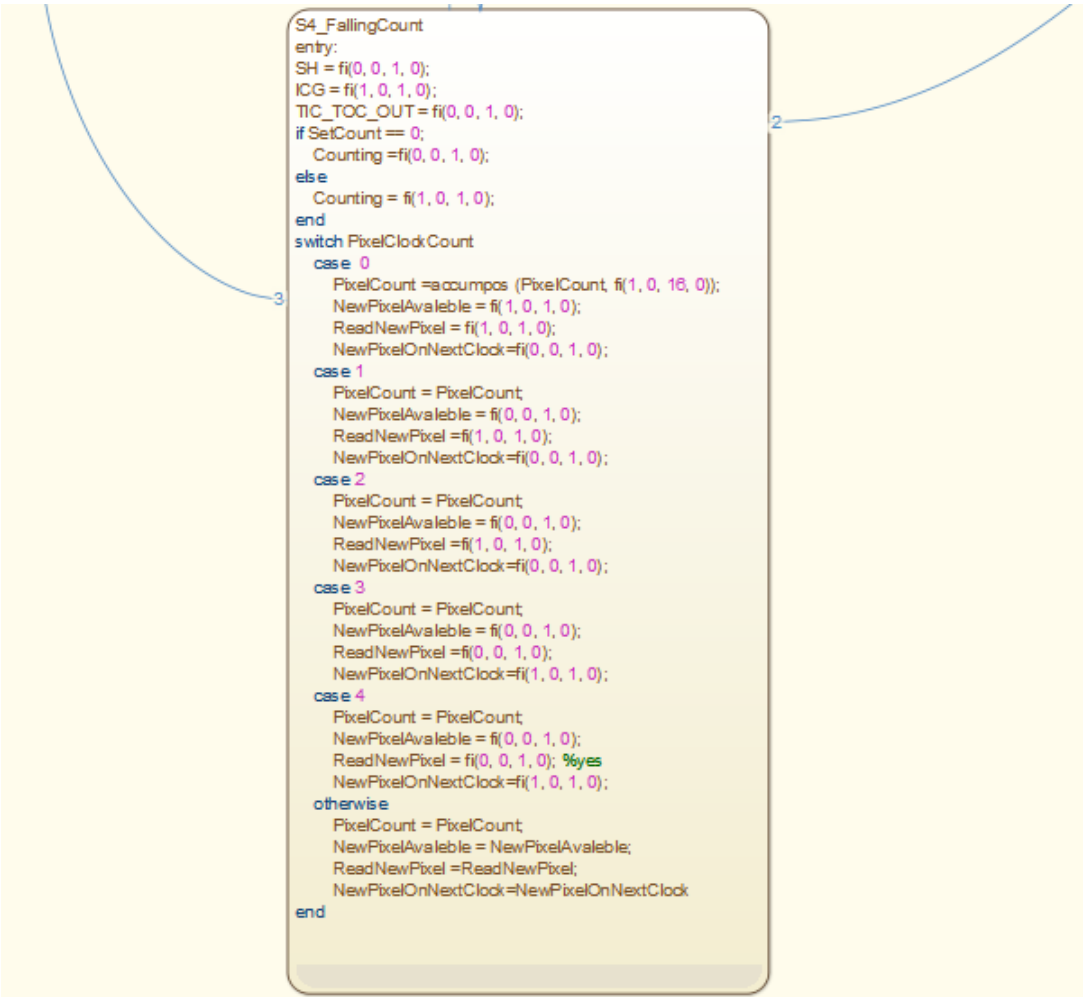

Figure 84, Zoomed view of Statespace blocks (Grindley, 2015)

This implementation was wrought with difficulty, when first simulated it was discovered that due to the way Statespace operates, the fundamental time step in the Simulink configuration must be an order of magnitude smaller than that of the ØM period and cannot be automatically selected meaning

that it must be configured manually in the settings in Simulink. It was found that a time step one forth the size of the ØM was suitable. This was discovered at great expense in terms of time. After the system was simulated and found to work repeatedly as expected, it was converted to VHDL. The design was then simulated with the ISIM part of the Xilinx ISE package, it was found to work in the same way as predicted in Matlab noting that the clock that drives the logic was an order of magnitude higher than ØM. The following stage was to implement the design on the FPGA itself, however this approach did not work despite a great deal of time spent trying to trace the problems back without a JTAG based debugger. It was decided to abandon the Statespace implementation in Matlab. It has been suggested that the reason it did not work was related to a synchronisation issue, a result of the complex VHDL implementation Matlab produced. It could also due to the version of VHDL the Matlab HDL coder uses, but without further investigation it remains unclear as to why it worked in simulation and not in practice.

Hard coded VHDL implementation

After the Matlab implementation was abandoned, it was decided to take the same approach, but to hard code the VHDL rather than rely on the Matlab HDL coder to implement it. Much of the same strategy was used. As discussed earlier, the wishbone interface used for the CCD reader was already available in the form of templates and examples provided so they were modified and re-used. Figure 89 shows a schematic of the CCD reader. Visible within it is the wishbone interface, the latch to control the CCD reader, the CCD reader waveform generator and a dual-port ram block from left to right respectively. The function of the wishbone interface is discussed in the section explain its function. A CCD control signal was used, which takes a data valid bit to indicate when the CCD's output should be updated in memory. This was used as a workaround for problems experienced with the wishbone interface, in that it was not latching outputs correctly. The CCD timing waveform generator in ISim which a commonly used simulator for Xilinx FPGA's produces a reset signal, an enable signal and auto-reset signal as inputs and outputs the pixel element addresses DataValid, FinishedDataRead, ICG, Mph, and SH as output signals all used to simulate the behaviour of the VHDL implementation. Note that ICG is the integration control signal; Mph is the clock pulse and SH controls the shutter for the CCD reader. Note that as Ø is not a valid character in VHDL so Mph was used as a substitute when referring to ØM.

As is expected, the enable and reset inputs to the waveform generator enable and reset the process. The auto reset signal automatically resets the process once complete. If it is not set, the waveform generator will only generate the timing waveform required to read one line once. When the auto reset is set to logic 1 the waveform will be generated indefinitely. There is another signal that allows the controller to switch between the electronic shutter enabled/disabled. Figure 92 shows the block diagram discribeing the CCD reader interface, note VHDL encompasses element 0 as one element so an 8 bit number will be illustrated as 0:7 opposed to 0:8.

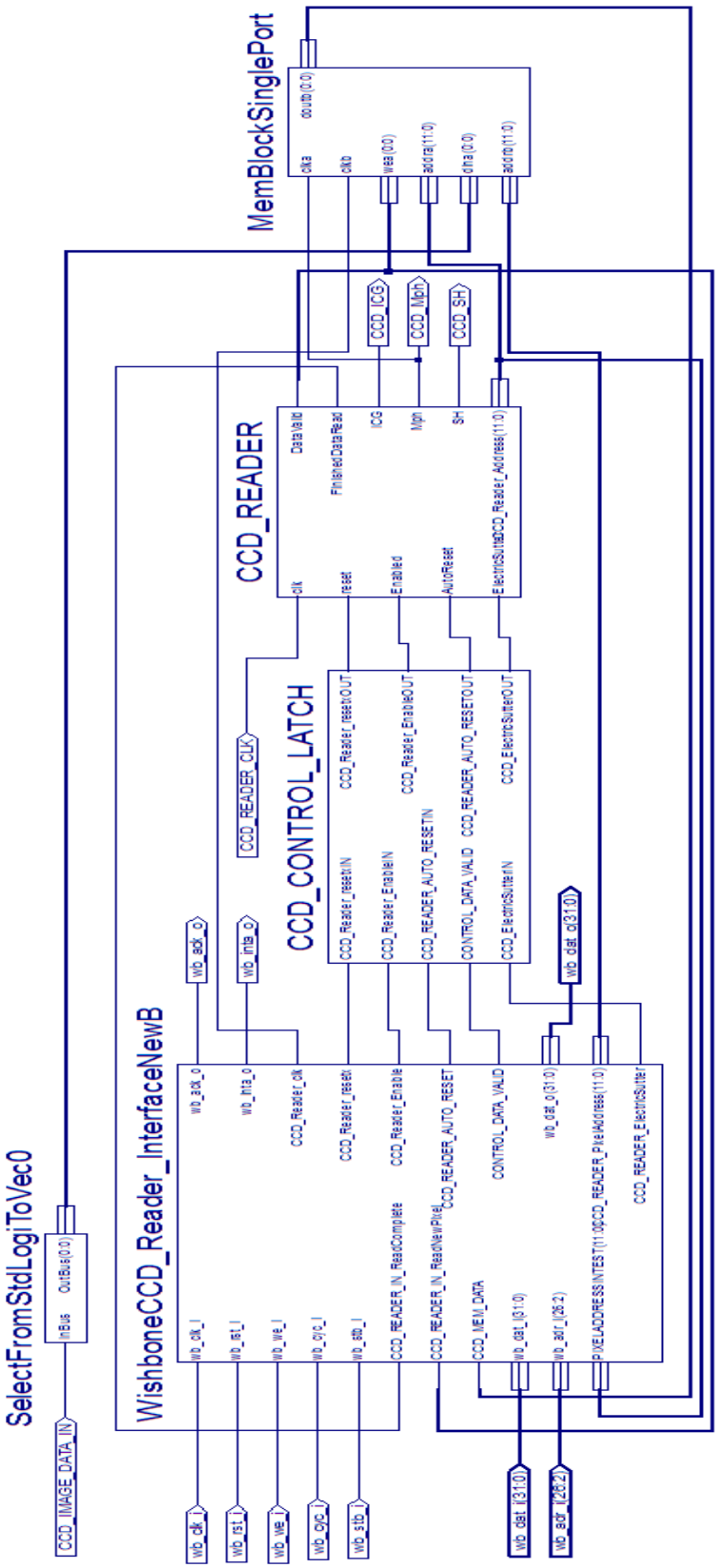

Figure 85, RTL of CCD reader (Grindley 2015)

When the first line has finished reading the FinishedDataRead signal is set to logic 1 and fed back to the wishbone interface to be used as an indicator as to when the line has finished reading, it is set to

logic 0 at any other instance. This flag is useful when the auto reset is set to logic 0 and the ZPU is in charge of when the CCD reader starts its reading process. The DataValid bit is used to control when port A of the dual port RAM is written to this is wea(0:0) in figure 89. The 12 bit signal that specifies the RAM address is set on the first clock of Mph and the DataValid signal set to logic 1 on the second clock of Mph, this ensures that the memory settles to a valid state, that no synchronisation issues could occur. This does not affect system performance and is illustrated in figure 90 below.

Figure 90 shows the first stage of the sequence. Note that the data valid signal in the second row is only valid on the second clock of the Mph of each pixel element and that the ICG, SH, and Mph outputs are consistent with that of the data sheet. Figure 91 shows the end of the sequence note that upon the last pixel element being read the Finished Data Read Flag is set, all timings are consistent with the timing diagram available in figure 84.

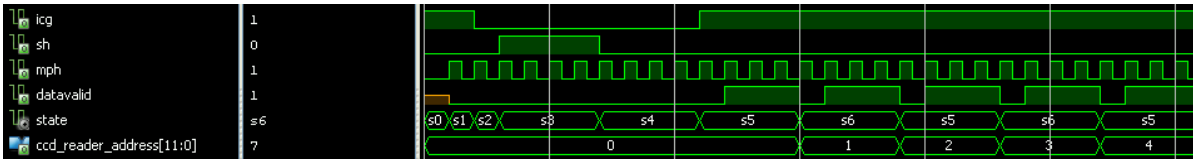

Figure 86, Beginning of the Linear CCD read sequence (Grindley, 2015)

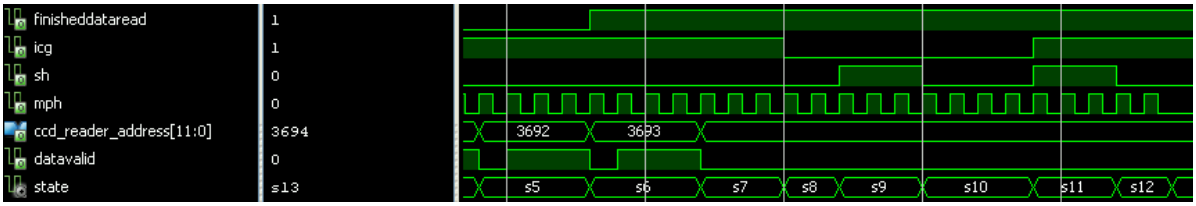

Figure 87, End of the Linear CCD read sequence (Grindley, 2015)

The use of a dual port RAM was inspired by research into GPU's and the way in which they use shared memory to allow different cores (processors) to share data. In the case of the CCD reader system, the wishbone interface interacts with the timing waveform generator and the dual port RAM as illustrated in figure 89. The system works by having the timing waveform generator interact with the Linear CCD reader and RAM; this means that the ZPU has no direct control of the CCD reader, only control of the peripheral via the wishbone interface an example of the code that does is listed below. This approach eliminates the need for extra software libraries, meaning that the microprocessor only needs to enable the core through the wishbone interface once and then leave it running. The timing waveform generator determines which pixel element is written to the RAM and when; the wishbone interface controls which pixel element is read from the RAM. This means the ZPU softcore only specifies the pixel elements address to read from the RAM, and then reads the value back through the wishbone interface, eliminating the need for any extra coding on the ZPU softcore. This frees up resources both in terms of code density and time taken to obtain a specific pixel value.

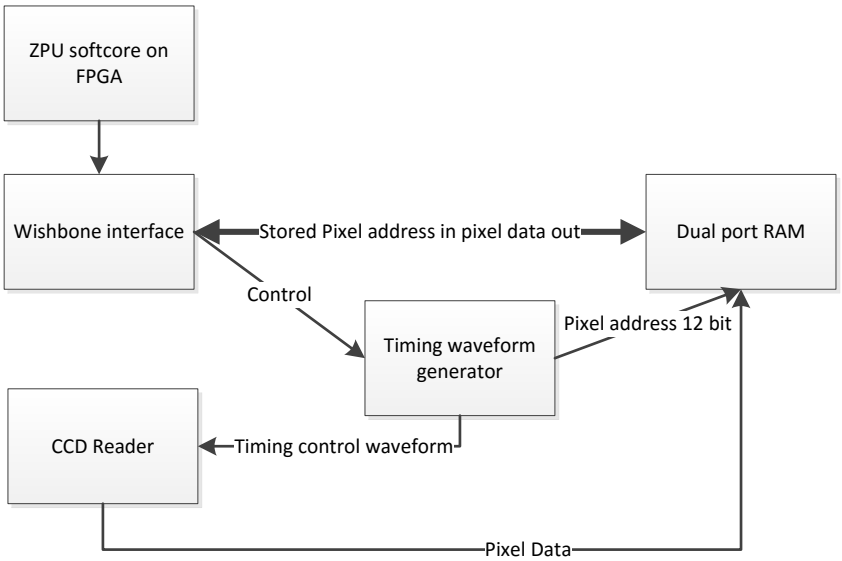

Figure 88, Flowchart of CCD reader interface (Grindley, 2015)

Wishbone interface and ZPU interaction

The wishbone interconnection interface was developed as a simple open source master slave interconnection architecture that is used as a means of establishing interconnection between IP cores and peripheral components on an FPGA, SoC or ASIC device. It allows developers to produce IP cores and peripherals that can interface between each other with ease and without the need to consider intellectual property issues or royalties.

The wishbone interface utilises a number of control signals to operate, the simple implementation used on the ZPU outputs signals to control the peripheral (Zylin, 2012). These control signals include Clock, Reset, Write enable, Signal stable, Bus cycle and an Acknowledge signal.
All control signals except the acknowledge signal are outputs from the master and inputs to the slave, with the acknowledge signal only being set to logic high when the Bus Cycle and Signal Stable signals are set to logic high.

The software interface in C does not have access to the control signals, the peripherals are accessed simply by writing to and reading from an address in a similar fashion to working with an array in memory. This is set up with two definition statements that can be found in the source code below in figure 93. The specifications of the wishbone bus are versatile allowing for a selection of between 8,16,32 or 64 bit data widths to suite the masters architecture and a choice between big and little endian data transfer. In the context of this project the ZPU softcore is the master that initiates data communication, the wishbone interface allows one data read or write per clock through its 32 bit data bus.

The wishbone interconnection architecture was chosen because it was included with the ZPU softcore and came with all the advantages that the ZPU softcore offers (Zylin, 2012). Such as compatibility with the Arduino code libraries and many additional peripherals in the form of libraries where already available such as VGA controllers, UARTS, SPI, PWM and I2C. They can be added by either dragging and dropping them into a schematic diagram or hard coding the peripheral into the design using VHDL. Using the wishbone specifications to develop an interface to work with the Toshiba TCD1304AP CCD reader was aided by templates that were used along with a set of working examples. The example that was used as a template utilised two 32bit registers and an 8 bit register as a control and configuration register. These are labelled as Register0, Register1 and Register2 in

the wishbone interface code in the appendix. Register2 is used as the control register, of the 8 bits only 4 are utilised and the rest left in reserve for additional features such as frame ID that are yet to be added. The configuration bits in Register2 are bit 0 “CCD Enabled”, bit 1 “CCD Reset”, bit 2 “CCD Auto Reset”, bit 3 “CCD Electric Shutter Enabled”.

The code used on the ZPU microprocessor initialisation routine shown in figure 93 was used to initialise the CCD reader peripheral explained below,

1. Write decimal 0 to register 0 to clear it.
2. Write decimal 0 to register 1 to clear it.
3. Write hexadecimal 0x01 to register 2 instructing the CCD reader interface to set enabled off, reset on, auto reset off, electric shutter off.
4. Write hexadecimal 0x0d to register 2 instructing the CCD reader interface to set enabled off, reset on, auto reset on, electric shutter on.
5. Write hexadecimal 0x0c to register 2 instructing the CCD reader interface to set enabled off, reset off, auto reset on, electric shutter on.
6. Write hexadecimal 0x0e to register 2 instructing the CCD reader interface to set enabled on, reset off, auto reset on, electric shutter on.

It is possible to reset and initialise the CCD reader with just two instructions but clearing everything first then enabling was considered to be best way of eliminating any possible errors from data being read from a previous session.

Register 0 was used to address elements in the controller’s memory to be read out which map directly to pixels on the CCD reader. This is done by writing the 12 bit address into the lower 12 bits of register 0, the other 20 bits are kept in reserve to be used in later implementations.
Register 1 is used as a write back register for the CCD reader interface, a write complete flag is written to element 0 of Register 1 it indicate that a new line has been read out this flag can be reset by the ZPU and is set every time a read line operation is completed. Element 1 of the register contains the binary value from the CCD reader at the location specified by the address in Register0 this should be updated immediately after Register 0 is written to or a new value in read into memory from the CCD reader. The rest of the elements in Register 0 are reserved for further implementations.

Note the wishbone template came in a bare bones format and the original comments made by the author of the wishbone template used to produce the wishbone peripherals mentioned in this project have been left in the code as required by the open source licence, all other code produced is original.

```
//This is what wishbone slot you are using
#define MYBASE IO_SLOT(9)
#define CCD_REG(x) REGISTER(MYBASE,x)

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);

  //Set Wishbone register 2 To start CCD_READER
  CCD_REG(0) = 0x00000000;
  CCD_REG(1) = 0x00000000;

  CCD_REG(2) = 0x01;       //enabled off, reset on, auto reset off, electric shutter off, 8-bit register control reg.
  delay(1);
  CCD_REG(2) = 0x0d;       //enabled off, reset on, auto reset on, electric shutter on, 8-bit register control reg.
  delay(2);
  CCD_REG(2) = 0x0c;       //enabled off, reset off, auto reset on, electric shutter on, 8-bit register control reg.
  delay(2);
  CCD_REG(2) = 0x0e;//0x03;       //enabled on, reset off, auto reset on, electric shutter on, 8-bit register control reg
  delay(2);
  delay(100);   //Allow memory to fill
  Serial.println("Setup Complete ");
}
```

Figure 89, Source code to initialise CCD reader peripheral (Grindley, 2015)

Chapter 6 - Kinematics and System Integration

Choice of integration system

There are several systems that can be used to integrate odometry data, control, user inputs, actuation and simulation all into one package. Three packages were found that may be suitable Microsoft Robot Developer Studio, V-rep, and Robot Operating System ROS

Microsoft Robotics Developer Studio

This was launched in 2006 and was amongst the first systems to integrate all the necessary functionality needed to design, enabling a developer to build and control a robot from scratch. Solidworks has a plugin called Collada that allows CAD designs to be imported into Robotics Developer Studio, allowing their kinematic model to be characterised (Microsoft, 2015).

The system boasts a 3D simulator with a physics engine to allow systems to be simulated before being built; this is effective but an equivalent is also available as part of V-rep and ROS so it does not stand out as a key feature in comparison.

Above all, Robotics Developer Studio provides a relatively easy to use interface in order to design, program and simulate robots. It is not however featured on ARM based development boards that typically run Linux distributions, such as Ubuntu ARM (Ubuntu, 2014). This however can be done with relative ease in ROS, giving ROS a key advantage. It is desirable to use an ARM based embedded system such as a CubieBoard or another embedded computer system for overall control and user integration. Microsoft Robotics Studio however makes this impractical and difficult to achieve.

Being a Microsoft based system; it is designed to run on operating systems designed by Microsoft. This dependency poses a problem for future development, as not only does the system need a Microsoft based operating system and runtimes to function but also requires developers to know their proprietary programing languages C# and (Visual programing language) VPL. Note that C# is also available for use in Linux. That said, its portability may be an issue, and alongside the inability to use an embedded computer running Linux, this puts Robotics Studio bottom of the list in comparison to ROS and V-Rep.

V-Rep

V-rep is a very user friendly system that incorporates a wide range of computer languages, including C/C++, Python, Lua, Java, Octave, Urbi and Matlab. This wide range of programing languages makes V-Rep very versatile, it is also accompanied by a comprehensive user manual (Coppelia Robotics, 2015).

The system allows for robot designs either to be created in the software itself or built in another 3D environment such as Solidworks and then imported, but this functionality was not investigated. The system includes a 3D simulator with four physics engines that can be easily invoked from the software. The physics engines included are Bullet, Newton, Vortex and ODE.

One key advantage of using V-rep is the range of plugins available, particularly the ROS plugin. This can be made to allow V-rep to pass simulated odometry data to ROS which can then be published to relevant subscribers in the ROS network. This process allows the control loops in ROS to respond and in turn publish their response to simulated actuators within V-rep. The ROS plugin enables the simulator and physics engines used in V-rep to effectively work with ROS. This would be a very useful feature but was not investigated further. It is however recommended for further research and will be noted in a later section.

The system is comprehensive and allows for CAD designs to be imported into it and simulated. Full system automation can be realised in V-rep such as advanced control methods and in some level of machine intelligence. A downside in comparison to ROS however is that V-rep does not have the same level of community support. It also requires an advanced operating system to run. Although this the same for ROS and Microsoft Robotics Studio, ROS however can run on an embedded ARM based system, and V-rep cannot work on a small low power embedded system. As such this disadvantage discounts its use as a means of unifying all of the odometry data and control for the system, though it remains an effective system to simulate control and physical responses.

ROS (Robot operating system)

ROS

Robot operating system (ROS) is a thin client that works on top of an operating system; traditionally this was Ubuntu Linux though it can work on Mac OS and in some cases Windows. It works using a publisher subscriber based system using nodes and services (Willow Garage, 2015). As a very basic example, a node could be written to read odometry data from a sensor that publishes it to a subscriber. This could typically be another node or service. The subscriber in this case could be charged with logging the data or could be a PID controller, which in turn publishes a control response to a node that interfaces with an actuator.

ROS operation

ROS runs a service called ROS core, this is essentially a server that handles all of the nodes and services on the ROS network. It also handles connectivity between the nodes and services. ROS core must be started on a computer before the system will work. Its connectivity is IP based which makes its operation intuitive to those with IP networking experience and robust, but does introduce a processing overhead which should be considered when applying control loops in ROS (O'Kane, 2013). As with any IP network based system, there is the scope to operate nodes and services in geographically independent locations. This presents several advantages, as any ROS system that has

connectivity to the internet and sufficient bandwidth can then be controlled remotely, making it possible to remote control the manipulator without great difficulty, while inheriting the same level of security typical Linux system has. Note any remote operation would require an IP based network, this could be implemented locally with long range Wi-Fi or an internet connection.

It is suggested that remote connectivity could give other advantages such as allowing an expert on a specific mine to inspect a device or supervise the process of render safe procedures from anywhere in the world.

ROS systems are built into packages and launch files allowing several packages to be executed at once. This mitigates the need to check if specific dependences are present before using the system. In addition to this there are libraries available to integrate many interface devices such as the Leap Motion which is proposed as the main means of controlling the position, pose and operation of the manipulator.

ROS Redundancy

ROS also has simple native means of handling fault tolerance, for example when setting up a listener node to manage the message queue size. The message queue allows a means of managing messages and retransmitting them if they do not arrive. The code that initialises a listener node can be set to forget messages if the queue overflows but also receives and processes messages if the transmitting node goes offline temporarily. This provides a simple means of handling dropouts in the system, for example should a sensor become temporarily disconnected and the node that handles its transmission goes off-line. Then the listener can act on the missing sensor data when it comes back online assuming it does and the message queue hasn't overflowed. How effective this is depends on what the node or service is doing with the data some processes can tolerate such events such as data logging, other processes may struggle to cope such as control loops; assuming there is redundancy in the data, the latency is low and the sample rate is substantially high enough some control loops may still be able to function.

ROS Serial protocol

As discussed earlier, ROS has a protocol stack for interfacing with standard serial communications. This is especially useful when communicating with microcontrollers or a softcore in this case that do not possess the full ROS stack. In the case of this project, the FPGA based sensors interface with the ZPU softcore, which does not support an operating system such as Linux due to there being no memory manager and as such cannot incorporate the ROS stack in the traditional sense. By virtue of the ROS serial protocol, a developer is able to solve this by allowing a microcontroller to function as a node and publish or receive data to the ROS network (Ferguson, 2015).

The protocol stack breaks down into the following components shown in figure 94, this data was taken directly from the ROS serial reference guide (Ferguson, 2015).

Byte	1	2
Function	Sync Flag	Sync Flag / Protocol version

Byte	3	4
Function	Message Length (N) - Low Byte	Message Length (N) - High Byte

Byte	5	6
Function	Checksum over message length	Topic ID - Low Byte

Byte	7	N
Function	Topic ID - High Byte	Serialized Message Data

Byte	N + 8
Function	Checksum over Topic ID and Message Data

Figure 90 Structure of ROS Serial data frame

The protocol consists of 9 different components. During this project it was decided not to produce any code that used this stack as it already existed, though it was necessary to have some understanding of its function. Should the ROS serial protocol be implemented on a PIC18 for example the ROS serial protocol would have to be implemented in a custom library, using the standard embedded library but would need some appropriate modification. However a library is available for Arduino based microcontrollers. As constants such as sync flag do vary between different versions of ROS the ROS serial library is generated within ROS to ensure each instance of its use is compatible with the host system, in the case of this project it was ROS Groovy (Willow Garage, 2012).

Arduino's use the AVR8 architecture, and use the tool chain supplied by the AVR to compile code into binary executables. As discussed earlier, the ZPU soft-core emulates the AVR architecture allowing the same libraries to function on it while hosting a 32bit architecture rather than the typical 8bit one. The ROS serial library does operate on a ZPU with minimal reconfiguration (Ferguson, 2015).

Some time was spent attempting to make this work; however some compilation errors were experienced in the ZPU programing environment. This was caused by the stdint.h library being not defined, after some time spent debugging the problem a solution was found and the missing header file was written and included in the appendix at the end. This was related to the standard integer library and accounts for the different sizes of integers between different processor architectures.

ROS Industrial

There are many packages available on ROS; however, there seems to be some concern with the stability of some of the packages according to online forums. Consequently, selecting stable and functional packages was critical. TF for example is native to ROS and is considered stable; but additional packages, for example for path planning, were still needed. One all in one solution however that is considered stable and contains the required functionality is ROS Industrial (SwRI Robotics and Automation, 2015). Manufactures such as Bosh use it to control their automatic processes in their manufacturing plants and supply a suite of libraries that are comparable with their RTC robot. In the light of this, ROS industrial was considered suitable for use in this project. In terms of functionality it contains a suite of tools capable of importing robot designs assembled in Solidworks and also incorporates relevant odometry data in addition to path planning. The necessary inverse kinematics

and path planning is generated in ROS Industrial and is typically implemented through a package called MoveIt. MoveIt contains several tools allowing for trajectory planning and collision detection, which prevents the manipulator's links from colliding with themselves and other objects, assuming however that ROS is aware of them. ROS Industrial also needs to be aware of the mass of the links in order to simulate appropriate control responses; this can be characterised using URDF files that ROS uses to interoperate a robot's physical features. URDF files are characterised using XML. This process is complex and laborious but can be easily performed through the Solidworks URDF import tool in ROS.

Importing design into ROS

A tool exists for importing STL files into ROS. STL's are a CAD file format that is compatible with Solidworks. The tool allows the export of STL CAD file, then allows the user to specify joint positions that are can then be characterised into URDF files and imported into ROS. This process was tested and some difficulty was found in exporting CAD drawings from CATIA, as CATIA does not export STL files but rather IGS files. To solve this, 3DSmax was used as an intermediary allowing IGS CAD files to be converted into STL files, then imported into Solidworks (Brawner, 2014).

Overall ROS integration

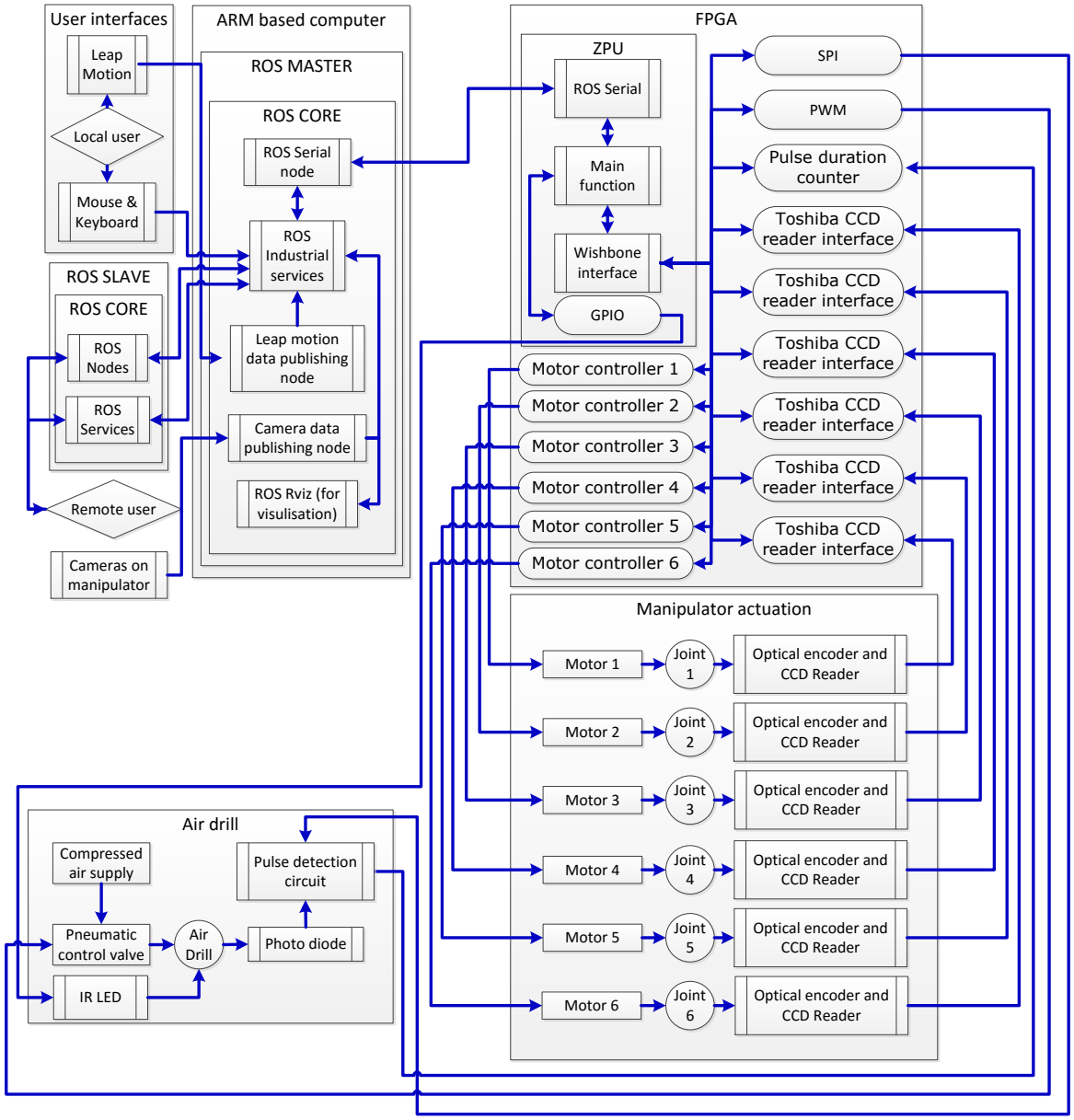

Figure 91, Diagram depicting ROS integration with currently proposed hardware (Grindley, 2015)

Figure 95 shows a flow diagram that depicts the overall system and how it should interact with the users and hardware. Note that the implementation is not complete as components such as the actuator that moves the drill head have not been characterised as there drive and sensors for odometry have not been specified yet. Figure 95 also includes a basic illustration of how the ROS network should interact with its nodes and services by publishing and subscribing to topics on the ROS network.

ROS Integration with Human interface devices

The control of the manipulator will utilise inverse kinematics libraries in ROS (Willow Garage, 2012), to allow the user to specify the position in space of the end effector and have a system calculate the appropriate joint positions, rather than the angle of each joint in a forward kinematic fashion. The use of inverse kinematics is more of a standard practice in most robotic applications and is easily

utilised in ROS's by path planning libraries such as MoveIt (SwRI Robotics and Automation, 2015). As this this is already well known, it was desired not to 'reinvent the wheel' and characterise all the kinematics equations relating to the system. One concern was the way in which an operator interacts with the manipulator. An intuitive system that allows the operator to interact with the manipulator would be ideal. Some robots in the past have been equipped with a controller similar to a typical RC hand set and in others, a games controller with a similar design to a PlayStation joy pad were used. With the manipulator being essentially a robotic arm with a set of tools at the end, a means of directing its movement in a similar way to how a person moves their hand would be ideal. Traditional controllers, as mentioned above cannot do this; some time therefore was spent looking at different ways to interact with the manipulator. Firstly results from experimentation with the Microsoft Kinect found that it was unable to estimate position at with short ranges of approximately 500mm depending on hand position and orientation. Furthermore, its accuracy was not deemed suitable and was found to be variable with respect to the distance from an object to a sensor. This however could be overcome by scaling any command to move from the Kinect sensor with a scaling factor of 0.2 for example. During the investigation moreover it was found that the Leap Motion (Leap Motion, 2015) sensor had much greater accuracy than the Kinect sensor and only operates when close to an object. It was configured to recognise human hands and not to produce a disparity map such as the Kinect sensor. This potentially makes its integration easier as far less signal processing would have been needed to identify a pose or gesture. The Leap motion also has a ROS package (Lier, 2015) which allows the system, to determine positions of any point on a palm, illustrated in figure 96. Note that figure 96 was produced using the Leap Motion visualisation tool and not ROS.

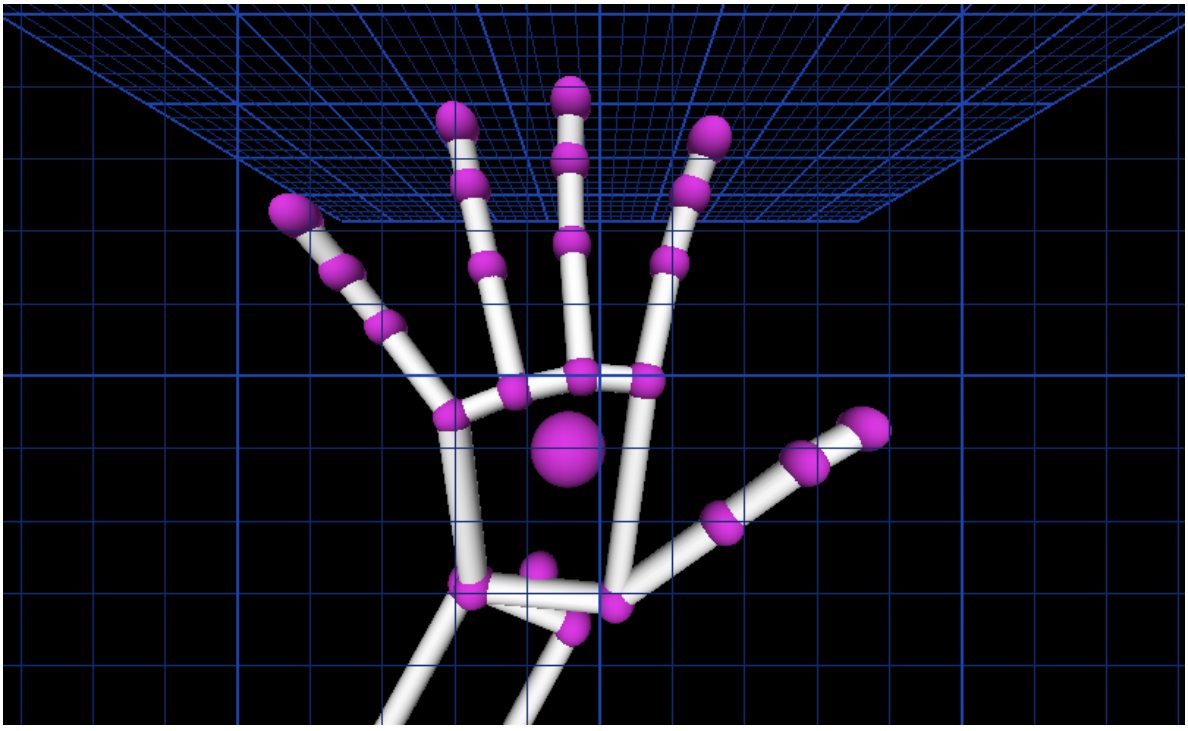

Figure 92, Illustration of Leap motion data (Grindley, 2015)

Kinematics and path planning

Although kinematics is managed by ROS, it is important to understand the way in which the control responses for the actuators are calculated.

Any joint position of a robotic manipulator can be characterised with a minimum of a cubic polynomial. This is illustrated in the equation below with each of its 4 coefficients a_0, a_1, a_2, a_3 corresponding to position, displacement, velocity and acceleration respectively (Craig, 1989). The same is true for the 3D positions in space in equation.

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

$$Position \begin{pmatrix} x \\ y \\ z \end{pmatrix} = p_0 + p_1 t + p_2 t^2 + p_3 t^3$$

Noting that systems operate at high speeds or have high mass it may be necessary to account for inertia with a 4th order polynomial. As stated in (Craig, 1989), when moving a joint from one position to another it is assumed that the end effector should always be under velocity and moreover that the velocity of the manipulator should be 0, before and after it moves. In order to move smoothly, the robot's joints must move in a way that ensures its position as a function with respect to time should be logarithmic, making its velocity parabolic and its acceleration linear (Craig, 1989). Path planning with a linear rate of acceleration should ensure that the end effector experiences the minimum amount of disturbance in terms of vibration.

By virtue of ROS the path planning element of the manipulators control can be automated within ROS MoveIt. As a result, limited time was spent on investigating path planning and kinematics as libraries were available in ROS to do it.

Chapter 7 - Conclusions and further work

Conclusion

Achievements and key contributions to the area

It should be noted that the overall contribution to the area is a set of recommendations and a proposed design of a robotic manipulator for use in land mine clearance. With reference to the objectives in chapter 1, the research done on assessing the needs of humanitarian demining organisations and making the distinction between those needs and that of military organisations early on in the project was advantageous. This allowed focus to be shifted onto the assessing those needs and identifying areas in which a robotic solution could be applied. One such issue identified in the study and described in chapter 2 is the potential scattering of metal components that make up the PROM-1 for example leading to false positives in a quality assurance phase of a demining operation. Furthermore the risk faced by a human deminer excavating, identifying and eliminating a mine is great enough to warrant an investigation into how that could be dealt with by robotic means. This lead to identifying excavation and the disabling of anti-personnel mines as an area to expand on, as is an area of research that had not seen any notable development. Though once this was identified the size of the scope of the project and the time left to do it meant only some aspects could be covered. With this in mind and the difficulties faced obtaining information on landmines, it was decided that the scope must be narrowed to dealing with three mines. After researching the challenges faced by deminers much of it anecdotal three criterions where decided on. They were, most common, most dangerous and likely to become unstable over time while being most likely to remain operational for

the longest time. The choices made were the Type72, PROM1 and M14 respectively. The reasoning and level of investigation that went into making these choices was appropriate as well as justified. A clear understanding as to how they operated and degraded was attained which is crucial when approaching a problem like this. As well as some of the economic factors that lead to their use, this was particularly relevant with respect to the Type72.

Research into robotic manipulation and conventional means of removal was undertaken, though as the only information or examples of demining equipment consisted of heavy mechanical means and tracked or wheeled robots such as the Wheelbarrow or Dragon Runner robots. This made achieving the objective of developing a robotic means of excavation and disabling a landmine more difficult. There was no specific work that could be built on or complemented; meaning the techniques and means of removal proposed had to be devised from scratch. There were various concerns that also needed to be addressed, such as the potential issues with resonance effects of the components that make up the mines. The proposed use of an air turbine to the drill was a well justified choice, irrespective of the additional issues involved in its use such as the need for an air pump, spare capacity, airlines and control valves. In order to make the whole device work a means of sensing the state of the manipulator and a way of unifying the overall control of the design was needed, including an intuitive means of interfacing with the user. A large contingent of the overall time spent on this project was spent on this. In terms of obtaining odometry data the Toshiba linear CCD was proposed for use on the system. The interfaces developed were a potentially cheap and a potentially accurate all round means of obtaining odometry data. Its reusability, mainly for determining joint positions made the investment in terms of time in designing a VHDL interface for it well justified. With regard to the user interface the choice of the Leap Motion as a human interface controller was also a novel choice, it was even selected before it was available to the public. The Leap Motion would provide a means of allowing the user to control the manipulator in a similar way as they would if they were doing the demining themselves. This would be a step up from the simplistic means of controlling many modern conventional clearance robots with modified game console controllers for example. Another advantage of the Leap Motion controller is its interface to Robot Operating System (ROS), which should not be understated this provides a robust means of interfacing with the Leap Motion. Furthermore Robot Operating System was chosen as a unified means of controlling the proposed manipulator, this was done because of its open source publisher/subscriber based approach to data distribution and node and service based modular design. It can work on embedded Linux based systems and has a huge set of libraries that could be invoked to give the developer the most flexibility possible, which was not the case with the completion such as Microsoft Robot Developer Studio. Another key decision was to make use of simulated hardware to give developers greater flexibility. The use of an FPGA with a simulated softcore processor on it also allows it to be interfaced with Robot Operating System easily using the ROS Serial library. This allows any peripherals designed on the FPGA to be interfaced with the wishbone interconnection bus connected to the simulated softcore processor and hence Robot Operating System. When considering the scope of future development and what has been recommended and produced in terms of the groundwork done on the sensors, electronics and overall control strategy for the manipulator, a great level of flexibility is available potential developers.

In summery the project was a success and achieved the objectives laid out, though no physical working model was produced the groundwork for a more in depth study and further development has been produced.

Weak points

As a large amount of time was spent on trying to understand the problem a limited amount of time was left for development. Had the problem been understood to the same level to as it is now at the beginning, it's likely that a demonstration of concept model still would not have been produced. Hence a key deficiency in the very first ideas that drove its original development, as producing a demonstration of concept model would have been the best way to demonstrate its principles and would have driven further development. As this was not achieved the concept cannot be proved at this stage.

The incremental nature of the development meant to some degree design strategies were researched but not implemented, one such example of this was the use of ROS. When it was first discovered as a means of unifying system control, a lot of time went into trying to determine if it was suitable for the task and could do everything the projects full implementation required. Such as communication with embedded devices and characterisation of the structures kinematics through the MoveIt package available in ROS Industrial.

This research was done first working through tutorials and online documentation as at the time it was first researched no books available. Though this approach was useful, project relevant practical work such as designing nodes and services to work on a ROS network was not undertaken due to time constraints. Given that ROS is so heavily featured in this implantation, it is a weak point in the design strategy as so much of the overall functionality such a kinematics, control and user interface relies so heavily on its use.

The underlining principles that this project hinges on are often assumptions that cannot be proven without field trials or extensive experimentation. Such as the use of epoxy to jam the firing mechanism of a landmine or that an air drill will be able to drill through a device such as the PROM1 as the materials used to make it and there properties are not know. If such assumptions turn out to be incorrect then a completely different strategy may be needed.

As this project was a learning exercise in many respects, a large contingent of the original CAD models and VHDL code was discarded as they did not work as anticipated or were not as they were intended to be. This learning process was costly in terms of time and far better attempts at design were made later on in the project.

Unresolved issues

The Toshiba linear CCD interface was not tested in full, only the driving circuit; VHDL interface and C code to initialise the simulated hardware interface were designed but were not used together in practice. Though the VHDL interface should work as a timing simulation was undertaken it still needs to be put into practice, as well its use with an optical encoder.

The development of the vibrating brush was not properly realised, nor the tool used to inject the epoxy into the hole. Some conceptual ideas exist though time constraints again prevented them from being produced.

Although a great deal of time was spent on the air drill, a readymade solution or a custom design that allowed a cartridge containing a turbine to be placed into a holder rather than a turbine by itself would be a better solution. For example the original drill holder adopted this approach; more time needs to

be spent on this. A rapid prototyped part would likely suffer from fatigue after moderate use so a milled aluminium holder would likely be better. This is due to air leakage which is a problem for the design proposed. One other possibility that was not full investigated was the use of a readymade cosmetic air drill, though it operates at lower speed it is a readily available mature product that may be a more suitable option.

Future work

As stated above, should the project continue, the first priority would be to consult with relevant individuals who either work or have experience of working in in the humanitarian demining industry. Further insight into the processes involved and the problems faced by deminers in the field is an essential first step in any further work.

Any further work should also be meticulously project managed. The initial approach to the design was incremental and done from a technical viewpoint and though there was a level of project management, in practice it was to a degree informal. The future success of the project should it continue, hinges on effective project management, the full use of formal requirements and more effective time and project management.

Further research should be carried out on antihandling devices and techniques to deal with them, this was not heavily researched in the time available. Although there was some investigation during the course of the project it did not achieve the level of focus that it perhaps should have.

Any future work as detailed above should begin by specifying the drive and actuation available and applicable to the manipulator. The mechanics and design of the structure should follow, it must be remembered be based on components that are readily available and are from a reliable source. Concentration should be shifted onto the end effector of the manipulator that uses the proposed tools. As such, the design should be split up into two different implementations one that is typical of the basic design in figure 28 and another that can be connected to an existing robot.

More research is needed on the makeup and the frequency response of the target mines and indeed other mines that are in use. This can be done through testing, but would require disarmed devices which are particularly difficult to obtain.

Reference

Mikulic, D. (2012) *Design of Demining Machines* United States : Springer

Romero, A. and Fernandez, E. (2013) *Learning ROS for Robotics Programming*. Birmingham, United Kingdom: Packt Publishing

Gonzalez, R. and Woods, R. (2007) *Digital Image Processing.* Upper Saddle River, United States: Pearson Education (US)

Kronos Group (2015) *Collada ROS Plugin* [online] available from <https://www.khronos.org/collada/> [01 May 2015]

Arima Lasers (2005) *AlGaInP Visible Laser Diode* [online] available from <http://www.farnell.com/datasheets/1961388.pdf> [08 June 2015]

Chemringeod (2015) *Water Disruptor*. [online] available from <http://www.chemringeod.com/products/defeat/> [08 June 2015]

King, C. (1996) *Jane's Mines and Mine Clearance*. Coulsdon, United Kingdom: Jane's Information Group

Coppelia Robotics (2014) *V-Rep*. Coppelia Robotics. [online] available from <http://www.coppeliarobotics.com/helpFiles/index.html> [01 May 2014]

Wolf, A. (2005) *Grippers in Motion*. Berlin, Germany: Springer-Verlag Berlin and Heidelberg GmbH & Co. KG

AZO Robotics (2015) *Wheelbarrow Mark 8 Plus II Remote Eod Vehicles from Remotec UK Ltd.* [online] available from <http://www.azorobotics.com/equipment-details.aspx?EquipID=308> [08 May 2015]

Easy Composites (2015) *Carbon Fiber Price*. [online] available from <http://www.easycomposites.co.uk/Products/Prepreg-Carbon-Fibre-Sheet.aspx> [01 June 2015]

De Wolf, F. (2008) *Report on the Influence of Magnet(-Tools) on the Type 72B Antipersonnel Mine*. [online] available from <http://www.gichd.org/fileadmin/pdf/LIMA/MagClutTNO2008_AnnexB.pdf> [01 June 2014]

Øyvind, H. (2012) *ZPU - The World's Smallest 32 Bit CPU with GCC Toolchain*. [online] available from <http://opencores.org/project,zpu> [01 May 2014]

Human Rights Watch (1993) *Land Mines in Angola / an Africa Watch Report*. New York, United States: Human Rights Watch

Lenarcic, J. (2006) *Advances in Robot Kinematics, Mechanisms and Motion*. New York, NY, United States, Springer-Verlag New York Inc.

O'Kane, J. (2013) *A Gentle Introduction to ROS*. Columbia, United States: CreateSpace Independent Publishing Platform

Craig, J. (1989) *Introduction to Robotics Mechanics and Control. 2nd edn*. Harlow, United Kingdom: Pearson Education Limited

Monin, L. and Gallimore, A. (2002) *The Devil's Gardens : A History of Landmines*. London, United Kingdom : Vintage Publishing

Metalmania UK (2015) *Price of Aluminium Bilit*. [online] available from <https://www.metalmaniauk.com/product/aluminium-plate-70mm-x-145mm-x-10mm-2104/> [1 May 2014]

Microchip (2011) *MCP6541/1R/1U/2/3/4*. Microchip. [online] available from <ww1.microchip.com/downloads/en/DeviceDoc/21696f.pdf> [1 June 2015]

micro-epsilon.co.uk (2015) *Cylindrical Capacitive Displacement Sensor*. [online] available from <http://www.micro-epsilon.co.uk/displacement-position-sensors/capacitive-sensor/capacitive-displacement-sensors/index.html> [1 June 2015]

SMC USA (2015) *SMC Linear Pneumatic Actuators.* [online] available from <http://www.smcusa.com/top-navigation/cad-models.aspx/53526> [1 June 2015]

Microsoft (2014) *Robotics Developer Studio. Microsoft*. [online] available from <https://msdn.microsoft.com/en-us/library/bb648760.aspx> [3 February 2014]

Dunlop, N. (1994) *War of the Mines : Cambodia, Landmines and the Impoverishment of a Nation*. London, United Kingdom : Pluto Press

Strack, P., Steiner, C. (2004) *Capacitive Displacement Sensor*. US 6700391 B2. [online] available from <https://www.google.com/patents/US6700391> [1 May 2014]

QinetiQ (2015) *Dragon Runner 20*. [online] available from <https://www.qinetiq-na.com/wp-content/uploads/data-sheet_dr-20.pdf> [1 May 2015]

QinetiQ (2015) *Dragon Runner, Unmanned Robotic Systems*. [online] available from <https://www.qinetiq-na.com/products/unmanned-systems/dragon-runner/ http://www.qinetiq.com/services-products/survivability/ugv/documents/brochure-dragon-runner.pdf> [1 May 2015]

McGrath, R. (2000) *Landmines and Unexploded Ordnance: A Resource Book*. London, United Kingdom : Pluto Press

Remotec (2015) *Remotec Wheelbarrow Mk9 Datasheet*. [online] available from <http://www.northropgrummaninternational.com/wp-content/uploads/2013/09/Remotec-Wheelbarrow-Mk9.pdf> [1 February 2015]

Toshiba (1997) *Toshiba TCD1304AP CCD*. [online] available from <http://www.spectronicdevices.com/pdf/Toshiba/TCD1304AP.pdf> [3 November 2014]

Ubuntu (2014) *Ubuntu Server for ARM*. [online] available from <http://www.ubuntu.com/download/server/arm> [3 November 2014]

United States Department of State and Office of Weapons Removal and Abatement (2009) *Scoping Study of the Effects of Aging on Landmines*. [online] available from <https://www.jmu.edu/_images/cisr/_pdfs/aging1.pdf> [3 November 2014]

Pedroni, V. (2004) *Circuit Design with VHDL*. [online] available from <https://is.muni.cz/el/1433/podzim2008/PV200/um/_eBook__MIT_Press_-_Circuit_Design_with_VHDL__2005_.pdf > Cambridge, Mass., United States : MIT Press Ltd [3 November 2014]

Ferguson, M. (2015) *ROS Serial Wiki* [online] available from <wiki.ros.org/rosserial> [13 June 2015]

Willow Garage (2012) *ROS Groovy*. [online] available from <http://wiki.ros.org/groovy> [3 November 2014]

Military Periscope (2011) *PROM-1 Anti-Personnel Mine* [online] available from <https://www.militaryperiscope.com/mdb-smpl/weapons/landmine/antipers/w0004529.shtml> [8 November 2014]

Eclipse IDE (2015) *Eclipse Integrated Development Environment* [online] available from <http://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/marsr> [13 June 2015]

Xilinx (2015) *MicroBlaze Processor Reference Guide*. [online] available from <http://www.xilinx.com/support/documentation/sw_manuals/mb_ref_guide.pdf> [13 December 2014]

Zylin consulting (2012) *Zylin AS*. [online] available from <http://opensource.zylin.com/zpudocs.html> [13 December 2014]

Renishaw (2015) *Renishaw Optical Encoders* [online] available from <http://www.renishaw.com/en/optical-encoders--6433> [13 June 2015]

Microchip (2015) *MCP6541 Push-Pull Output Sub-Microamp Comparators* [online] available from <http://www.farnell.com/datasheets/1641674.pdf> [13 June 2015]

Microchip (2015) *7/8-Bit Single/Dual SPI Digital POT with Non-Volatile Memory* [online] available <http://www.farnell.com/datasheets/630445.pdf> [7 October 2015]

On semiconductor (2015) *100 mA, Low Power Low Dropout Voltage Regulator* [online] available from <http://www.farnell.com/datasheets/1829347.pdf> [13 June 2015]

Multicomp (2015) *OFL-3102 Infrared Emitter* [online] available from <http://www.farnell.com/datasheets/1697424.pdf> [13 June 2015]

SwRI Robotics and Automation (2015) *ROS Industrial* [online] available from http://rosindustrial.org/> [13 June 2015]

Fairchild semiconductor (2015) *QRE1113 Miniature Reflective Object Sensor* [online] available from <https://www.fairchildsemi.com/datasheets/QR/QRE1113.pdf> [13 June 2015]

Brawner, S. (2015) *SolidWorks to URDF Exporter* [online] available from <http://wiki.ros.org/sw_urdf_exporter> [13 June 2015]

Lier, F. (2015) *Leap motion ROS package* [online] available from <http://wiki.ros.org/leap_motion> [13 June 2015]

Leap Motion (2015) *Leap Motion Human Interface Device* <https://www.leapmotion.com> [13 June 2015]

Craig, J. (1989) *Introduction to Robotics: Mechanics and Control 2nd Revised edition* Harlow, United Kingdom : Pearson Education Limited

Appendix

HDL Code

Wishbone counter

```
----------------------------------------------------------------------------------
-- Company: Gadget Factory
-- Engineer: Alvaro Lopes
--
-- Create Date:    13:56:50 12/10/2013
-- Design Name:
-- Module Name:    TEMPLATE_zpuino_wb_Wishbone - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
-- This is an example template to use for your own Wishbone Peripherals.
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments: All rights reserved ©
--
----------------------------------------------------------------------------------
-- This example uses asynchronous outputs.

----------------------------------------------------------------------------------
-- Company: Coventry University
-- Engineer: Josef Grindley
--
-- Create Date:            10:12:35 15/8/2014
-- Design Name:            Wishbone Counter Interface
-- Module Name:    TEMPLATE_zpuino_wb_Wishbone - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
-- This code takes the wishbone interface supplied by Gadget Factory and adds a counter allowing
for pulse duration counting
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
-- This example uses asynchronous outputs.

library ieee;
```

```
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;

entity WishboneCounter is
  port (
        CounterClk                      :       in      std_logic;
        CounterSignal           :       in      std_logic;
        LEDcontrol_out          :       out std_logic;

    wb_clk_i                    :       in std_logic;
        wb_rst_i                        :       in std_logic;
    wb_dat_o                    :       out std_logic_vector(31 downto 0);
    wb_dat_i                    :       in std_logic_vector(31 downto 0);
    wb_adr_i                    :       in std_logic_vector(26 downto 2);
    wb_we_i                             :  in std_logic;
    wb_cyc_i                    :       in std_logic;
    wb_stb_i                    :       in std_logic;
    wb_ack_o                    :       out std_logic;
    wb_inta_o                   :       out std_logic
         );
end entity WishboneCounter;



architecture rtl of WishboneCounter is
        signal register0                :       std_logic_vector(31 downto 0); --
Register 0 (32 bits)
        signal register1                :       std_logic_vector(31 downto 0); --
Register 1 (32 bits)
        signal register2                :       std_logic_vector(7 downto 0);  --
Register 2 (8 bits)
        signal Counter                                  :               std_logic_vector(31
downto 0) := "00000000000000000000000000000000";
        signal FinalCounterValue        :       std_logic_vector(31 downto 0);
        signal CounterOverflow          :       std_logic := '0';
        signal CounterProcessDone       :       std_logic := '0';

begin
 -- Asynchronous acknowledge
        wb_ack_o <= '1' when wb_cyc_i='1' and wb_stb_i='1' else '0';
 -- Asynchronous output overflow flag
        register2(0) <= CounterOverflow;

--Counter process
        process(CounterClk, Counter, CounterOverflow, wb_rst_i, CounterProcessDone)
        begin
   if rising_edge(CounterClk) then
                if wb_rst_i='1' or CounterProcessDone = '1' then
                        Counter <= (others => '0');
                else
```

```
                        --Counter <= Counter + 1;
                        Counter <= std_logic_vector( unsigned(Counter) + 1 );
                        if Counter(31 downto 0) = "11111111111111111111111111111111" then
        --Detect Overflow
                                Counter <= (others => '0');
                                CounterOverflow <= '1';
                        else
                                --Counter <= Counter;
                                CounterOverflow <= '0';
                        end if;
                end if;
        end if;
        end process;
-- End counter process

--Input detect process
        process(CounterSignal, Counter, CounterProcessDone)
        begin
        if (CounterSignal'event) then
                if CounterSignal = '1' and CounterProcessDone = '0' then
                        CounterProcessDone <= '1';
                        FinalCounterValue <= Counter;
                        --Counter <= (others => '0');
                else
                        if CounterSignal = '0' then
                                CounterProcessDone <= '0';
                        end if;
                        --register0 <= register0;
                        --Counter <= Counter;
                end if;
        end if;
        end process;
--End Input detect process

--Control the LED
 process(CounterClk)
 begin
                if rising_edge(CounterClk) then
                        LEDcontrol_out <= register1(0);          --THE FIRST BIT OF REGISTER1
CONTROLS THE SENSOR LED 1 ON, 0FF
                end if;
 end process;
--End LED control

 -- Multiplex the data output (asynchronous)
 process(register0,register1,register2, wb_adr_i)
 begin
  -- Multiplex the read depending on the address. Use only the 2 lowest bits of addr
  case wb_adr_i(3 downto 2) is
                when "00" =>
                        wb_dat_o <= register0;  -- Output register0
```

```
                when "01" =>
                        wb_dat_o <= register1;  -- Output register1
                when "10" =>
                        wb_dat_o(31 downto 0) <= (others => '0'); -- We put all upper 24 bits to
zero
                        wb_dat_o(7 downto 0) <= register2;       -- since register2 only has 8 bits
                when others =>
                        wb_dat_o <= (others => 'X'); -- Return undefined for all other addresses
   end case;
  end process;

  process(wb_clk_i)
  begin
   if rising_edge(wb_clk_i) then  -- Synchronous to the rising edge of the clock
                if wb_rst_i='1' then
                        -- Reset request, put register1 and register2 with zeroes,
                        -- put register 3 with binary 10101010b
                        register0 <= (others => '0');
                        register1 <= (others => '0');
                        register2 <= "00000000";
                else -- Not reset
                -- Check if someone is writing
                        if wb_cyc_i='1' and wb_stb_i='1' and wb_we_i='1' then
                          -- Yes, it's a write. See for which register based on address
                                case wb_adr_i(3 downto 2) is
                                        when "00" =>
                                          register0 <= wb_dat_i;--wb_dat_i;  -- Set register0
                                        when "01" =>
                                          register1 <= wb_dat_i;  -- Set register1
                                        when "10" =>
                                          register2 <= wb_dat_i(7 downto 0); -- Only lower 8 bits
for register2
                                        when others =>
                                          null; -- Nothing to do for other addresses
                                 end case;
                                else
                                        if CounterProcessDone = '1' then ---Added
                                                        register0 <= FinalCounterValue;
                                        end if;
                        end if;

                end if;
   end if;
  end process;
end rtl;
```

CCD Reader

CCD Reader Wishbone interface

```
----------------------------------------------------------------------------------
-- Company: Gadget Factory
-- Engineer: Alvaro Lopes
--
-- Create Date:    13:56:50 12/10/2013
-- Design Name:
-- Module Name:    TEMPLATE_zpuino_wb_Wishbone - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
-- This is an example template to use for your own Wishbone Peripherals.
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
-- This example uses asynchronous outputs.


----------------------------------------------------------------------------------
--
--
--  Modified by Josef Grindley 2015, Coventry University
--  All rights reserved ©
--
----------------------------------------------------------------------------------


library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;

entity WishboneCCD_Reader_InterfaceNewB is
  port (
--          wishbone_in                                     : in std_logic_vector(61 downto 0);
--          wishbone_out                                    : out std_logic_vector(33 downto 0);

    wb_clk_i: in std_logic;
            wb_rst_i: in std_logic;
    wb_dat_o: out std_logic_vector(31 downto 0);
    wb_dat_i: in std_logic_vector(31 downto 0);
    wb_adr_i: in std_logic_vector(26 downto 2);
    wb_we_i:  in std_logic;
```

```
    wb_cyc_i: in std_logic;
    wb_stb_i: in std_logic;
    wb_ack_o: out std_logic;
    wb_inta_o:out std_logic;

            CCD_Reader_clk                          : out     std_logic;
            CCD_Reader_resetx                       : out     std_logic;
            --CCD_Reader_resetDEL                   : out     std_logic;
            --CCD_Reader_clk_enable                 : out     std_logic;
            --CCD_Reader_OS                         : out     std_logic;
--          CCD_Reader_CCD_Clock                    : out     std_logic;
            CCD_Reader_Enable                       : out     std_logic;
            CCD_READER_PixelAddress      : out std_logic_vector(11 downto 0);
            CCD_READER_AUTO_RESET                   : out std_logic;
            CCD_READER_ElectricSutter    : out std_logic;
            CONTROL_DATA_VALID                      : out std_logic;

--          CCD_READER_IN_ceout                                     : in std_logic;
           PIXELADDRESSINTEST                                       : in std_logic_vector(11
downto 0);
            --CCD_READER_IN_NewPixelAvaleble        : in std_logic;
            --CCD_READER_IN_NewPixelOnNextClock     : in std_logic;
            CCD_READER_IN_ReadComplete                      : in std_logic;
--          CCD_READER_IN_NewEncoderValue         : in std_logic;
            CCD_READER_IN_ReadNewPixel                      : in std_logic;
            CCD_MEM_DATA                                                    : in
std_logic                       --Note should be 8 bit for RGB reader
            --CCD_READER_IN_EncoderReadOut        : in std_logic_vector(15 downto 0);
            --CCD_READER_PixelAddress             : in std_logic_vector(15 downto 0)
            --CounterClk                                             : in      std_logic;
            --CounterSignalPin                                       : in      std_logic;
            --LEDcontrol_out                                         : out std_logic
            );
end entity WishboneCCD_Reader_InterfaceNewB;



architecture rtl of WishboneCCD_Reader_InterfaceNewB is

--Define your registers here
  signal register0: std_logic_vector(31 downto 0); -- Register 0 (32 bits)
  signal register1: std_logic_vector(31 downto 0); -- Register 1 (32 bits)
  signal register2: std_logic_vector(7 downto 0);  -- Register 2 (8 bits)

--Wishbone signals - Don't touch.
--  signal  wb_clk_i:    std_logic;               -- Wishbone clock
--  signal  wb_rst_i:    std_logic;               -- Wishbone reset (synchronous)
--  signal  wb_dat_i:    std_logic_vector(31 downto 0); -- Wishbone data input  (32 bits)
--  signal  wb_adr_i:    std_logic_vector(26 downto 2); -- Wishbone address input  (32 bits)
--  signal  wb_we_i:     std_logic;                -- Wishbone write enable signal
--  signal  wb_cyc_i:    std_logic;                -- Wishbone cycle signal
```

```
-- signal  wb_stb_i:    std_logic;                -- Wishbone strobe signal
--
-- signal  wb_dat_o:    std_logic_vector(31 downto 0); -- Wishbone data output (32 bits)
-- signal  wb_ack_o:    std_logic;                -- Wishbone acknowledge out signal
-- signal  wb_inta_o:   std_logic;
--
--ADDED FOR COUNTER
--          Notes for counter
--
--          signal Counter:                         std_logic_vector(31 downto 0);
--          signal FinalCounterValue:       std_logic_vector(31 downto 0);
--          signal CounterClk:                      std_logic;
--          signal CounterOverflow:                 std_logic;
--          signal CounterSignalPin:std_logic;
--          signal CounterProcessDone:      std_logic;
--END OF COUNTER

begin
-- Unpack the wishbone array into signals so the modules code is not confusing.
--  wb_clk_i <= wishbone_in(61);
--  wb_rst_i <= wishbone_in(60);
--  wb_dat_i <= wishbone_in(59 downto 28);
--  wb_adr_i <= wishbone_in(27 downto 3);
--  wb_we_i <= wishbone_in(2);
--  wb_cyc_i <= wishbone_in(1);
--  wb_stb_i <= wishbone_in(0);
--
--  wishbone_out(33 downto 2) <= wb_dat_o;
--  wishbone_out(1) <= wb_ack_o;
--  wishbone_out(0) <= wb_inta_o;
-- End unpacking Wishbone signals


 -- Asynchronous acknowledge

 wb_ack_o <= '1' when wb_cyc_i='1' and wb_stb_i='1' else '0';

          CCD_Reader_clk <= wb_clk_i;
 -- Multiplex the data output (asynchronous)

 process(register0,register1,register2, wb_adr_i)
 begin

  -- Multiplex the read depending on the address. Use only the 2 lowest bits of addr

  case wb_adr_i(3 downto 2) is
   when "00" =>
    wb_dat_o <= register0;  -- Output register0
   when "01" =>
    wb_dat_o <= register1;  -- Output register1
   when "10" =>
```

```
      wb_dat_o(31 downto 0) <= (others => '0'); -- We put all upper 24 bits to zero
      wb_dat_o(7 downto 0) <= register2;       -- since register2 only has 8 bits
    when others =>
      wb_dat_o <= (others => 'X'); -- Return undefined for all other addresses
  end case;

 end process;

-- process(wb_clk_i)
-- begin

--        if rising_edge(wb_clk_i) then  -- Synchronous to the rising edge of the clock

--                if wb_rst_i='1' then
--                        CCD_Reader_clk                                  <=
        '0';
--                        CCD_Reader_resetx                       <=              '0';
--                        CCD_Reader_clk_enable           <=      '0';
        --Taken out due to driveing multiple signals
--                        CCD_Reader_Enable                       <=      '0';
                --else
                --      CCD_Reader_clk                                  <=
        wb_clk_i;
                        --CCD_Reader_resetx                       <=              wb_rst_i;
                        --CCD_Reader_clk_enable           <=      wb_dat_i(0);            --
NOTE THESE MUST BE SET APPROPREATLEY
                        --CCD_Reader_Enable                       <=      wb_dat_i(1);
                --      CCD_Reader_resetx                       <=
        wb_dat_i(2);
                        --CCD_Reader_resetDEL                     <=      wb_dat_i(2);
--                end if;

--        end if;
--  end process;

 process(wb_clk_i)
 begin

  if rising_edge(wb_clk_i) then  -- Synchronous to the rising edge of the clock

    if wb_rst_i='1' then
      -- Reset request, put register1 and register2 with zeroes,
      -- put register 3 with binary 10101010b

      register0 <= (others => '0');
      register1 <= (others => '0');
      register2 <= "10101010";
                        --ADDING RESET FOR COUNTER
                        --Counter         <= (others => '0');

                        --Added from above
```

```
                        --CCD_Reader_clk                                <=
        '0';
                        CCD_Reader_resetx                       <=              '0';
--                      CCD_Reader_clk_enable           <=      '0';
        --Taken out due to driveing multiple signals
                        CCD_Reader_Enable                       <=      '0';
                        CCD_READER_AUTO_RESET           <=              '0';
                        --End of Added from above
    else -- Not reset

      -- Check if someone is writing
      if wb_cyc_i='1' and wb_stb_i='1' and wb_we_i='1' then
        -- Yes, it's a write. See for which register based on address
        case wb_adr_i(3 downto 2) is
          when "00" =>
            register0 <= wb_dat_i;--wb_dat_i;  -- Set register0
          when "01" =>
            register1 <= wb_dat_i;  -- Set register1
          when "10" =>
            register2 <= wb_dat_i(7 downto 0); -- Only lower 8 bits for register2
          when others =>
            null; -- Nothing to do for other addresses
        end case;
                        ---Added
                        --REGISTER 0 USED FOR CPU >>> WB
                        --Bit 2 : CCD_READER_PixelAddress : out std_logic_vector(11 downto 0)
                        --REGISTER 1 USED FOR wb >>> CPU
                        --Bit 0 : CCD_READER_IN_NewPixelAvaleble;
                        --Bit 1 : CCD_READER_IN_NewPixelOnNextClock;
                        --Bit 2 : CCD_READER_IN_ReadComplete;
                        --Bit 3 : CCD_READER_IN_ReadNewPixel;
                        --Bit 4 : CCD_MEM_DATA;
                        --REGISTER 2 USED FOR Config
                        --Bit 0 : CCD_Reader_Enable
                        --Bit 1 : CCD_Reader_clk_enable

                                CCD_READER_PixelAddress <= register0(11 downto 0);

                                register1(0) <= CCD_READER_IN_ReadComplete;
                                register1(1) <= CCD_MEM_DATA;

                                register1(31 downto 20) <= PIXELADDRESSINTEST;

                                CCD_Reader_Enable               <= register2(0);
--                              CCD_Reader_clk_enable           <= register2(1);
                                CCD_Reader_resetx               <= register2(1);
                                CCD_READER_AUTO_RESET           <= register2(2);
                                CCD_READER_ElectricSutter       <= register2(3);
                                CONTROL_DATA_VALID              <= register2(4);        --
Data valid bit for external latch
      end if;
```

```
    end if;

   end if;

  end process;

end rtl;
```

CCD control latch

```
----------------------------------------------------------------------------------
--
--
--  Created by Josef Grindley 2015, Coventry University
--  All rights reserved ©
--
----------------------------------------------------------------------------------


library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;

entity CCD_CONTROL_LATCH is
 port (
        CCD_Reader_resetxIN             : in    std_logic;
        --CCD_Reader_clk_enableIN       : in    std_logic;
        CCD_Reader_EnableIN             : in    std_logic;
        CCD_READER_AUTO_RESETIN         : in std_logic;
        CCD_ElectricSutterIN            : in std_logic;


        CONTROL_DATA_VALID                          : in std_logic;

        CCD_Reader_resetxOUT                        : out   std_logic;
        --CCD_Reader_clk_enableOUT      : out   std_logic;
        CCD_Reader_EnableOUT                        : out   std_logic;
        CCD_READER_AUTO_RESETOUT                    : out std_logic;
        CCD_ElectricSutterOUT           : out std_logic
        );
end entity CCD_CONTROL_LATCH;



architecture rtl of CCD_CONTROL_LATCH is
 signal register0: std_logic_vector(4 downto 0);
begin

CCD_Reader_resetxOUT                    <= register0(0);
```

```
--CCD_Reader_clk_enableOUT   <= register0(1);
CCD_Reader_EnableOUT                        <= register0(1);
CCD_READER_AUTO_RESETOUT<= register0(2);
CCD_ElectricSutterOUT             <= register0(3);


 process(CONTROL_DATA_VALID)
 begin

 --   if rising_edge(CONTROL_DATA_VALID) then  -- Synchronous to the rising edge of the clock

     if CONTROL_DATA_VALID='1' then
                                    register0(0) <= CCD_Reader_resetxIN;
--                                  register0(1) <= CCD_Reader_clk_enableIN;
                                    register0(1) <= CCD_Reader_EnableIN;
                                    register0(2) <= CCD_READER_AUTO_RESETIN;
                                    register0(3) <= CCD_ElectricSutterIN;
     else -- Not reset
                                    register0 <= register0;
     end if;

--    end if;

 end process;

end rtl;
```

CCD reader

```
----------------------------------------------------------------------------------
-- Company: Coventry University
-- Engineer: Josef Grindley
--
-- Create Date:    21:07:00 08/20/2015
-- Design Name:            CCD_READER
-- Module Name:    CCD_READER - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments: All rights reserved ©
--
----------------------------------------------------------------------------------


library ieee;
```

```
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;


ENTITY CCD_READER IS
  PORT(
    clk                                 :       IN      STD_LOGIC;
    --input   :    IN       STD_LOGIC;
    reset                               :       IN      STD_LOGIC;
                Enabled                                 :       IN
        STD_LOGIC;
                AutoReset                               :       IN
        STD_LOGIC;
                ElectricSutter                  :       IN              STD_LOGIC;
    --output      :       OUT     STD_LOGIC_VECTOR(1 downto 0);
                CCD_Reader_Address      :       OUT     unsigned(11 downto 0);--
STD_LOGIC_VECTOR(13 downto 0);
                DataValid                               :       OUT     STD_LOGIC;
                FinishedDataRead                :       OUT     STD_LOGIC;
                ICG                                             :       OUT
        STD_LOGIC;
                Mph                                             :       OUT
        STD_LOGIC;
                SH                                                      :       OUT
        STD_LOGIC);
END CCD_READER;

ARCHITECTURE Behavioral OF CCD_READER IS
  TYPE STATE_TYPE IS (s0, s1, s2, s3, s4, s5, s6, s7, s8, s9, s10, s11, s12, s13 );
  SIGNAL state                          :       STATE_TYPE;
        SIGNAL ProcessCount             :       unsigned(13 downto 0) :=
"00000000000000";
        SIGNAL NextStateCount       :       unsigned(13 downto 0) := "00000000000000";
        SIGNAL CCDAddress                       :       unsigned(11 downto 0) :=
"000000000000";
        SIGNAL Mph_enabled              :       STD_LOGIC;
        SIGNAL ICG_buff                         :       STD_LOGIC;
        SIGNAL Mph_buff                         :       STD_LOGIC;
        SIGNAL SH_buff                          :       STD_LOGIC;
        SIGNAL ElectricSutterFlag       :       STD_LOGIC := '0';
BEGIN
  PROCESS (clk, reset)
  BEGIN
    IF reset = '1' THEN
      state <= s0;
    ELSIF (clk'EVENT AND clk = '1') THEN
                if Enabled = '1' then
                        --ProcessCount <= ProcessCount - 1;
                                CASE state IS
                                        WHEN s0=>
```

```
                                        DataValid <= '0';
                                                IF ProcessCount = 0 then
                                                        state <= s1;
                                                        ProcessCount <= NextStateCount;
                                                ELSE
                                                        ProcessCount <= ProcessCount - 1;
                                                END IF;
                                        WHEN s1=>
                                                IF ProcessCount = 0 then
                                                        state <= s2;
                                                        ProcessCount <= NextStateCount;
                                                ELSE
                                                        ProcessCount <= ProcessCount - 1;
                                                END IF;
                                        WHEN s2=>
                                                IF ProcessCount = 0 then
                                                        state <= s3;
                                                        ProcessCount <= NextStateCount;
                                                ELSE
                                                        ProcessCount <= ProcessCount - 1;
                                                END IF;
                                        WHEN s3=>
                                                IF ProcessCount = 0 then
                                                        state <= s4;
                                                        ProcessCount <= NextStateCount;
                                                ELSE
                                                        ProcessCount <= ProcessCount - 1;
                                                END IF;
                                        WHEN s4=>
                                                IF ProcessCount = 0 then
                                                        state <= s5;
                                                        ProcessCount <= NextStateCount;
                                                ELSE
                                                        ProcessCount <= ProcessCount - 1;
                                                END IF;
                                        WHEN s5=>
                                                IF ProcessCount = 0 then
                                                        ProcessCount <= NextStateCount;
                                                        CCDAddress      <= CCDAddress + 1;
                                                        state <= s6;
                                                        --ElectricSutterFlag <= not
ElectricSutterFlag;
                                                ELSE
                                                        ProcessCount <= ProcessCount - 1;
                                                END IF;

                                                if ProcessCount > "00000000000000" then--
"00000000000010" then--ProcessCount /= NextStateCount and ProcessCount >
"00000000000000" then
                                                        DataValid <= '1';
                                                else
```

```
                                        DataValid <= '0';
                                end if;

                        WHEN s6=>
                                IF ProcessCount = 0 then
                                        IF CCDAddress < "111001101101" then   --
Max pixel address is "00111001101101" out of 3694
                                                state <= s5;
                                        ELSE
                                                state <= s7;
                                        END IF;
                                        CCDAddress      <= CCDAddress + 1;
                                        ProcessCount <= NextStateCount;
                                        DataValid <= '0';
                                ELSE
                                        ProcessCount <= ProcessCount - 1;
                                END IF;

                                if ProcessCount > "00000000000000" then--=
"00000000000010" then--ProcessCount /= NextStateCount and ProcessCount >
"00000000000000" then
                                        DataValid <= '1';   -- CHANGED TO ENSURE
ONLY ONE WRITE PER ELEMENT CYCLE
                                        --DataValid <= '0';
                                else
                                        DataValid <= '0';
                                end if;

                        WHEN s7=>
                                DataValid <= '0';
                                IF ProcessCount = 0 then
                                        state <= s8;
                                        ProcessCount <= NextStateCount;
                                ELSE
                                        ProcessCount <= ProcessCount - 1;
                                END IF;
                        WHEN s8=>
                                IF ProcessCount = 0 then
                                        state <= s9;
                                        ProcessCount <= NextStateCount;
                                ELSE
                                        ProcessCount <= ProcessCount - 1;
                                END IF;
                        WHEN s9=>
                                IF ProcessCount = 0 then
                                        state <= s10;
                                        ProcessCount <= NextStateCount;
                                ELSE
                                        ProcessCount <= ProcessCount - 1;
                                END IF;
                        WHEN s10=>
```

```
                                        IF ProcessCount = 0 then
                                                state <= s11;
                                                ProcessCount <= NextStateCount;
                                        ELSE
                                                ProcessCount <= ProcessCount - 1;
                                        END IF;
                                WHEN s11=>
                                        IF ProcessCount = 0 then
                                                state <= s12;
                                                ProcessCount <= NextStateCount;
                                        ELSE
                                                ProcessCount <= ProcessCount - 1;
                                        END IF;
                                WHEN s12=>
                                        IF ProcessCount = 0 then
                                                state <= s13;
                                                ProcessCount <= NextStateCount;
                                        ELSE
                                                ProcessCount <= ProcessCount - 1;
                                        END IF;
                                WHEN s13=>
                                        IF ProcessCount = 0 then
                                                if AutoReset = '1' then
                                                        state <= s0;
                                                        ProcessCount <= NextStateCount;
                                                        CCDAddress <= "000000000000";-
-(others <= '0');
                                                        ElectricSutterFlag<=    '0';
                                                end if;
                                        ELSE
                                                ProcessCount <= ProcessCount - 1;
                                        END IF;
                                END CASE;
                        END IF;
        END IF;
  END PROCESS;

  PROCESS (state, ProcessCount)
  BEGIN
    CASE state IS
      WHEN s0 =>
                                NextStateCount <= "00000000000000";  --NextStateCount <=
"00000000000001";
        ICG_buff                        <= '1';
                                Mph_enabled                     <= '0';
                                SH_buff                         <= '0';
      WHEN s1 =>
                                NextStateCount <= "00000000000000";
        ICG_buff                        <= '1';
                                Mph_enabled                     <= '1';
                                SH_buff                         <= '0';
```

```
        WHEN s2 =>
                                NextStateCount <= "00000000000011";
          ICG_buff              <= '0';
                                Mph_enabled             <= '1';
                                SH_buff                 <= '0';
        WHEN s3 =>
                                NextStateCount <= "00000000000011";
          ICG_buff              <= '0';
                                Mph_enabled             <= '1';
                                SH_buff                 <= '1';
        WHEN s4 =>
                                NextStateCount <= "00000000000011";
          ICG_buff              <= '0';
                                Mph_enabled             <= '1';
                                SH_buff                 <= '0';
        WHEN s5 =>
                                NextStateCount <= "00000000000011";
                                --NextStateCount <= "00000000000001"; -- for s6
            CHANGED TO ENSURE 4 CLOCKS PER ELEMENT
                                --SH_buffuterState := SH_buffuterState + 1
                                --      if SH_buffuterState = 4 then
                                --              SH_buff = SH_buffuterState(0);
                                if ElectricSutter = '1' then
                                        SH_buff                 <= '1'; --
ElectricSutterFlag;                     --'1';
                                else
                                        SH_buff                 <= '0';
                                end if;
                                ICG_buff                        <= '1';
                                Mph_enabled             <= '1';
        WHEN s6 =>
                                IF CCDAddress < "111001101101" then   --Max pixel address is
"00111001101101" out of 3694
                                        NextStateCount <= "00000000000011"; -- for s5
                                        --NextStateCount <= "00000000000001"; -- for s5
            CHANGED TO ENSURE 4 CLOCKS PER ELEMENT

                                ELSE
                                        NextStateCount <= "00000000000010"; -- for s7
                                END IF;

                                --if ElectricSutter = '1' then
                                --      SH_buff                 <= ElectricSutterFlag;
                    --'1';
                                --else
                                --      SH_buff                 <= '0';
                                --end if;

                                --NextStateCount <= "00000000000001";
          ICG_buff              <= '1';
                                Mph_enabled             <= '1';
```

```
                                SH_buff                 <= '0';
        WHEN s7 =>
                                NextStateCount <= "00000000000001";
          ICG_buff              <= '1';
                                Mph_enabled             <= '1';
                                SH_buff                 <= '0';
        WHEN s8 =>
                                NextStateCount <= "00000000000010";
          ICG_buff              <= '0';
                                Mph_enabled             <= '1';
                                SH_buff                 <= '0';
        WHEN s9 =>
                                NextStateCount <= "00000000000011";
          ICG_buff              <= '0';
                                Mph_enabled             <= '1';
                                SH_buff                 <= '1';
        WHEN s10 =>
                                NextStateCount <= "00000000000010";
          ICG_buff              <= '0';
                                Mph_enabled             <= '1';
                                SH_buff                 <= '0';
        WHEN s11 =>
                                NextStateCount <= "00000000000001";
          ICG_buff              <= '1';
                                Mph_enabled             <= '1';
                                SH_buff                 <= '1';
        WHEN s12 =>
                                NextStateCount <= "00000000000001";
          ICG_buff              <= '1';
                                Mph_enabled             <= '1';
                                SH_buff                 <= '0';
        WHEN s13 =>
                                NextStateCount <= "00000000000001";
          ICG_buff              <= '1';
                                Mph_enabled             <= '0';
                                SH_buff                 <= '0';
      END CASE;
    END PROCESS;

    FinishedDataRead <= '1' when CCDAddress >= "111001101101" else '0';   -- 3693 is the last pixel
    element
    Mph <= Mph_enabled and Clk;
    ICG <= ICG_buff;
    SH <= SH_buff;
    CCD_Reader_Address <= CCDAddress;
    END Behavioral;
```

Vector logic converter

```
    ----------------------------------------------------------------------------
    --
```

```
--
--  Created by Josef Grindley 2015, Coventry University
--  All rights reserved ©
--
---------------------------------------------------------------------------------


library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;

entity SelectFromStdLogiToVec0 is
  port (
                    OutBus : out  std_logic_vector(0 downto 0);
                    InBus              : in std_logic
            );
end entity SelectFromStdLogiToVec0;



architecture rtl of SelectFromStdLogiToVec0 is

begin
                                OutBus(0) <= InBus;
end rtl;
```

Optical encoder

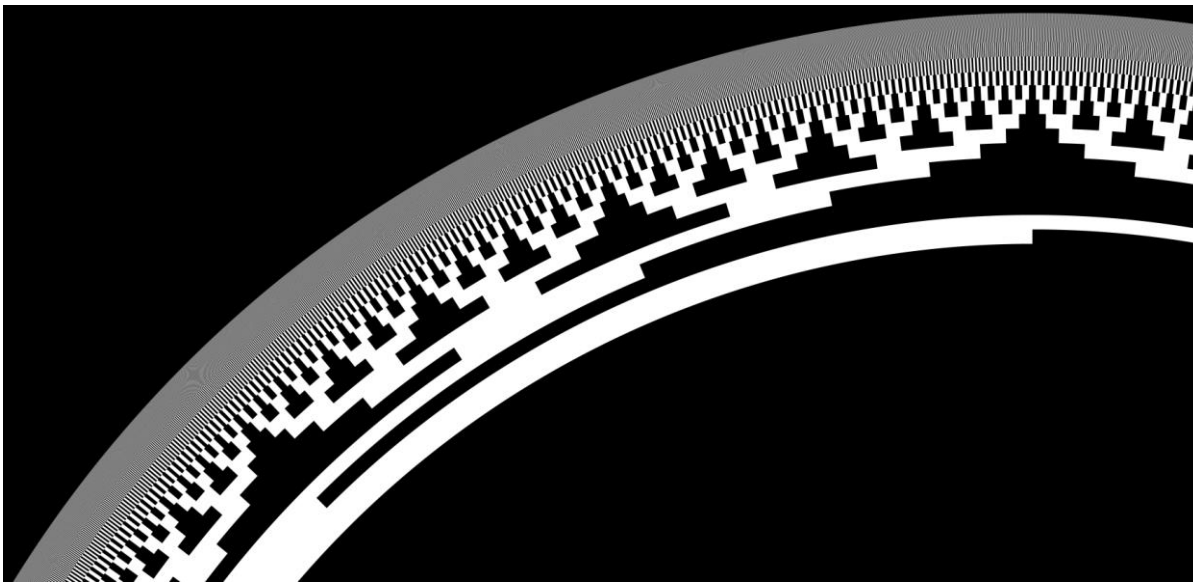

Figure 93

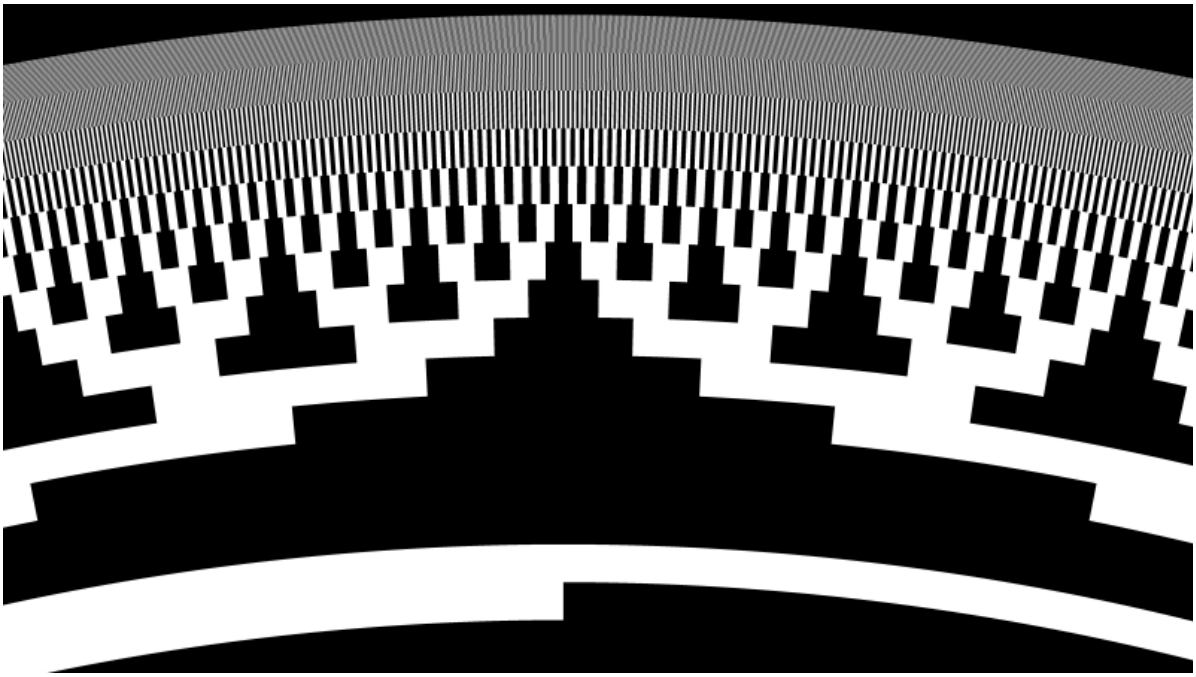

Figure 94

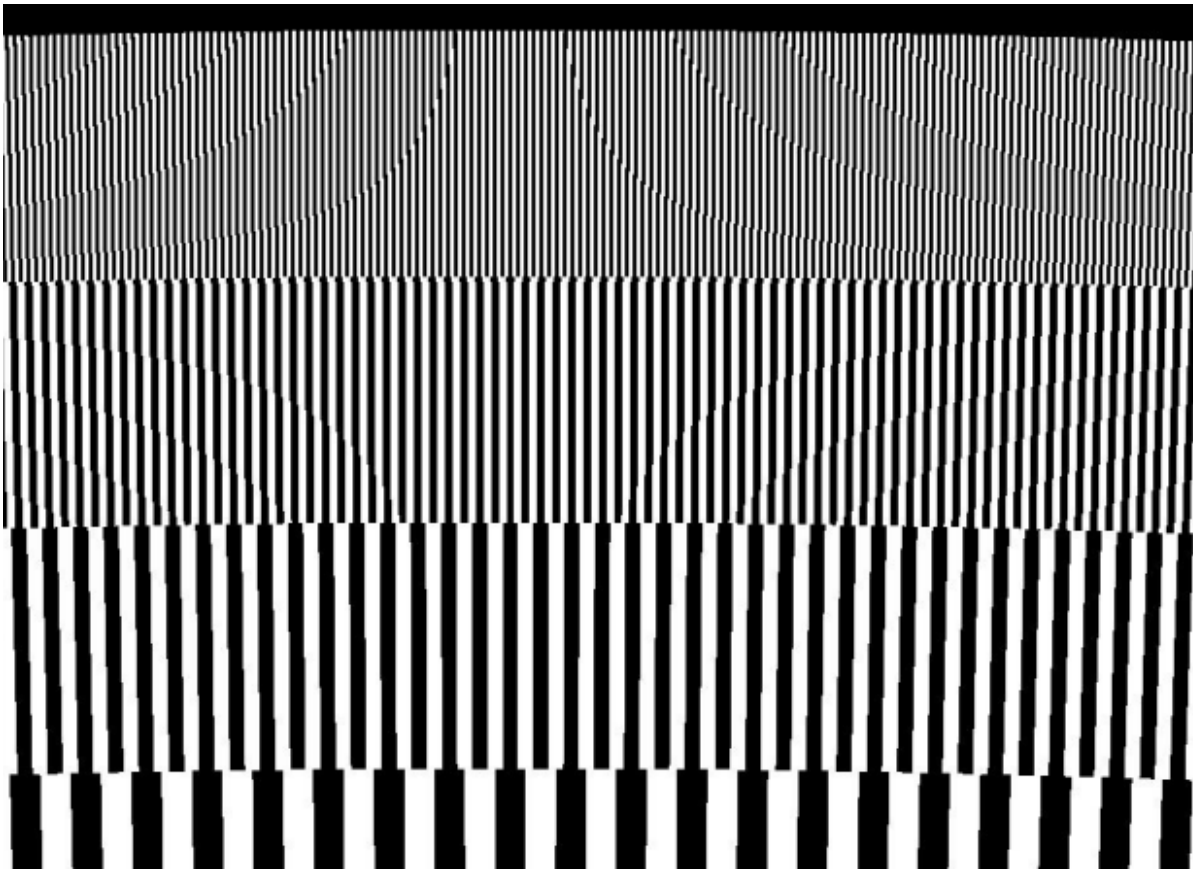

Figure 95

Slit Disk

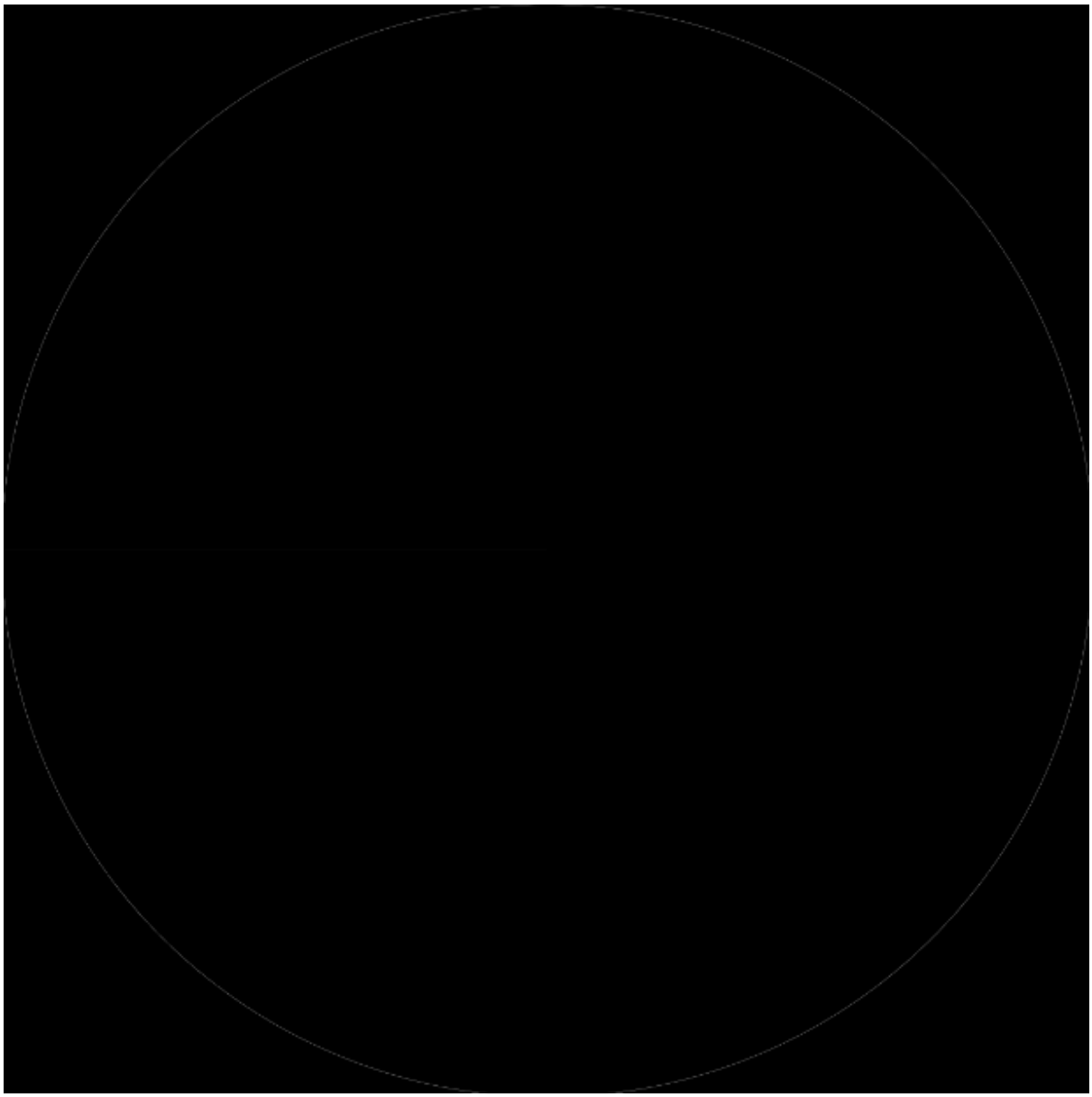

Figure 96

Standard integer library (Stdint.h)

```
/* -*- c -*- */

/*

 * Copyright 2007,2009 Free Software Foundation, Inc.

 *

 * This program is free software: you can redistribute it and/or modify

 * it under the terms of the GNU General Public License as published by

 * the Free Software Foundation, either version 3 of the License, or
```

```
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>.
 */
// This code was modified by Josef Grindley, 2015
#ifndef INCLUDED_STDINT_H
#define INCLUDED_STDINT_H

typedef signed char       int8_t;
typedef unsigned char     uint8_t;
typedef short             int16_t;
typedef unsigned short    uint16_t;
typedef int               int32_t;
typedef unsigned int      uint32_t;
typedef long long int               int64_t;
typedef unsigned long long int    uint64_t;

//typedef int               intptr_t;
//typedef unsigned int      uintptr_t;
#endif /* INCLUDED_STDINT_H */
```