UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

Design of a 2-axis, Continuous Rotation, Camera Control Platform

by

Gareth James Cawood

Mechatronic Project 488

Report presented in partial fulfilment of the requirements for the degree

BEng Mechatronics

at the University of Stellenbosch

Supervisor: Dr. H.A. Engelbrecht

October 2011

MECHANICAL PROJECT 488: EXECUTIVE SUMMARY

Title of Project
Development of a 2 Axis, continuous rotation, camera control platform
Objectives
The design and manufacture of a platform for a Basler A311fc camera which allows it full and continuous rotation in two planes. Position and speed should be controllable via software loaded on a computer that communicates with the platform
Which aspects of the project are new/unique?
The investigation of methods to transfer data and power to a camera while still allowing continuous rotation. The design of a platform that allows continuous rotation. The formulation a way to test the accuracy of the final product.
What are the findings?
That a design is possible which allows the continuous rotation of the camera in two axes as well as that such a product can be manufactured amd software can be developed to control it.
What value do the results have?
In tracking situations it means that a camera can continue to rotate where normal setups would reach the end of their range.
If more than one student was involved, what was my contribution?
N.A.
Which aspects of the project will carry on after completion?
The tracking of moving objects with the platform will be made possible. This will be done making use of image processing on the video feed and with the implementation of a control system.
What are the expected advantages of the continuation?
To extend the usability of the product.
What arrangements have been made to expedite continuation?
The software has been written in such a way that additions and modifications can be easily achieved and it can be easily integrated with other software. Software written will be cross platform compatible. A Mechatronics Project for 2012 has been put forth to continue this one.

Department of Mechanical and Mechatronic Engineering

Stellenbosch University

Declaration

I know that plagiarism is wrong.

Plagiarism is to use another's work (even if it is summarised, translated or rephrased) and pretend that it is one's own.

This assignment is my own work.

Each contribution to and quotation (e.g. "cut and paste") in this assignment from the work(s) of other people has been explicitly attributed, and has been cited and referenced. In addition to being explicitly attributed, all quotations are enclosed in inverted commas, and long quotations are additionally in indented paragraphs.

I have not allowed, and will not allow, anyone to use my work (in paper, graphics, electronic, verbal or any other format) with the intention of passing it off as his/her own work.

I know that a mark of zero may be awarded to assignments with plagiarism and also that no opportunity be given to submit an improved assignment. I know that students involved in plagiarism will be reported to the Registrar and/or the Central Disciplinary Committee.

Name: *Gareth James Cawood*

Student no: *15375137*

Signature:

Date: *21 October 2011*

Acknowledgements

I extend my thanks to my supervisor, Dr. Engelbrecht, for his continued support throughout the project, both financially and with his knowledge and enthusiasm.

Thanks also to the personnel at the Central Electronic Services for their help with various aspects of the project, specifically Wessel and Lincoln.

Contents

List of Figures

List of Tables

Nomenclature

Acronyms

bpp	-	bits per pixel
fps	-	frames per second
sps	-	steps per second
API	-	Application programming interface
ATX	-	Advanced Technology eXtended
CAD	-	Copmuter-Aided Design
CES	-	Central Electronic Services
DC	-	Direct Current
DV	-	Digital Video
GUI	-	Graphical User Interface
IC	-	Integrated Circuit
NFC	-	Near Field Communication
PCB	-	Printed Circuit Board
PLC	-	Programmable Logic Controller
PSU	-	Power Supply Unit
PWM	-	Pulse Width Modulation
RC	-	Resistor Capacitor
RPM	-	revolutions per minute
SSI	-	Synchronous Serial Interface
USB	-	Universal Serial Bus

Symbols

r	-	radius
t	-	time
B	-	Magnetic Flux Density
F	-	Force
I	-	Current
J	-	Moment of Inertia
K_e	-	Flux Constant
V	-	Voltage
α	-	angular acceleration
θ	-	general angle
ϕ	-	general angle
τ	-	torque
ω	-	angular velocity

note: to maintain consistency in the report, data transfer speeds are always given in bytes per second (Bps), and not bits per second (bps).

Chapter 1

Introduction

This report focuses on the design and manufacturing processes followed in the development of a two-axis turret platform capable of continuous rotation.

1.1 Motivation

The platform was specifically designed for use with the Basler A311fc camera. Several of these cameras were purchased by the MIH Media Lab at the University of Stellenbosch for use in the fields of augmented reality and tracking. The cameras were in good condition, but had no platform or mounting with which they could be controlled. A platform was desired that could track a moving object without being restricted to only one rotation. The project was proposed by Dr H.A. Engelbrecht.

1.2 Background

To allow the camera to 'look' in any direction, it is necessary for it to be able to rotate on two axes. For the purpose of this report the two axes were defined as the pan- (rotation in the horizontal plane) and tilt-axis (rotation in the vertical plane). These can be seen in fig 1.1.

The platform was divided into three layers. Each layer was separated from the others by one of the axes. The inner layer contains only the camera, separated from the middle layer by the tilt-axis. The middle layer contains Motor 1 and a sensor. The middle layer is then separated from the outer layer by the pan-axis. The outer layer consists of Motor 2, a sensor for the pan-axis, the microcontroller, computer and power supply for the system. This layout can be seen as a schematic in fig 1.2 and in practise in fig 1.3

The Basler 311fc camera is a 16 bit colour camera capable of high speed frame rate capture. During testing, data transfer between the camera and a computer reached a maximum transfer rate of 31.1 MBps. The camera software allows the user to independently set the resolution of the image captured, the shutter time per frame (which thus also sets the frames per second (fps) value) and the bits per pixel (16 or 8). At the maximum resolution of 658x496pixels, and 16 bit images, the maximum fps value that could be obtained was 50.4 fps. By lowering the resolution to 320x240, the maximum fps value could be increased

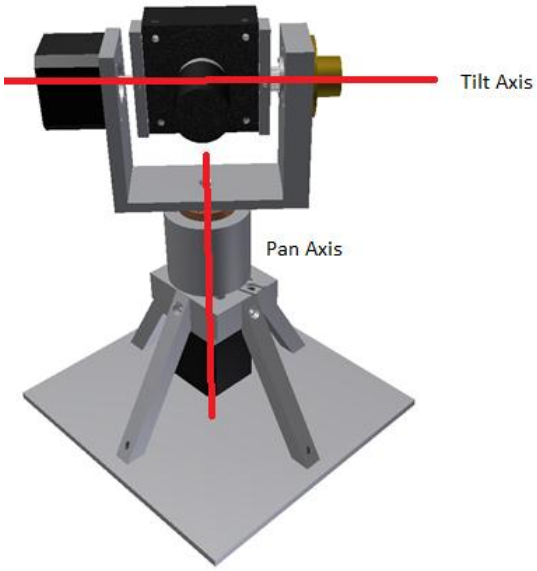


Figure 1.1: The two axes of the design

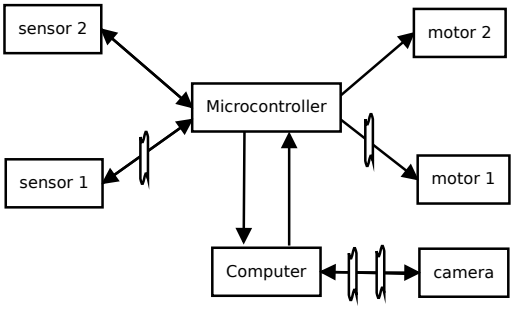


Figure 1.2: Schematic illustrating the layers of the design as well as the motors and slip-rings.

up to 132 fps. These factors combined influence the transfer rate of the camera. Data transfer is manged by a six pin FireWire port that also supplies power to the camera.

Physically the camera is fairly small, measuring 62x62 mm and 48 mm thick without a lens. A lens, that was acquired with the camera, is mounted on the front and allows the aperture and focus to be adjusted. A picture of the camera can be seen in fig 1.4.

FireWire is a data transfer protocol developed by the Apple Corporation. It follows the IEEE 1394 protocol. It was designed as a high speed networking type protocol to compete with USB. It is mainly used for video streams from webcams or transferring video footage from DV cameras. The Basler A311fc runs on FireWire 400 which is capable of speeds up to 49 MBps. A standard FireWire plug consists of four data pins. A six pin variation is used when power is required by the device, and typically up to 9 W is supplied (1394 Trade Association, 2010). Newer developments make use of a nine pin format for higher data transfer rates.

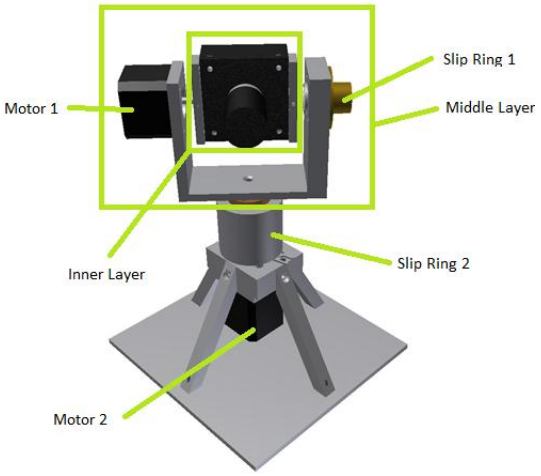


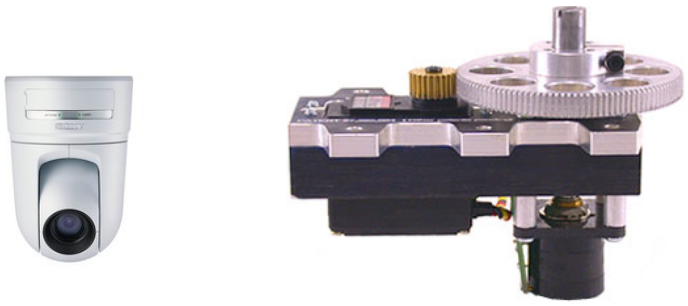
Figure 1.3: Model illustrating location of layers as well as the motors and slip-rings.

Figure 1.4: Image of the Basler A311fc camera (1394 Store.com, 2009)

To determine at what speed the platform would be required to rotate at, an experiment was performed. It was decided that the camera should be able to track a person running across its line of sight at a distance of 2 m. The camera's field of vision extends 40° in the horizontal plane, and 30° in the vertical plane. A subject was placed 2m away from the camera and instructed to run across its field of vision. The subject crossed the 40° in a time of 1 s. The distance covered was 74 cm. This equates to a rotational speed of 6.7 RPM. This was then used as the basis for design of the system.

1.3 Literature Synopsis

Although many commercial camera turret units are available, none could be found that were capable of continuous rotation. Several products are however able to rotate up to 720°. Furthermore the cameras used with the commercial units generally had a low frames per second (fps) value, varying between 15 and 30 fps. The Basler A311fc is capable of up to 73.5 fps at a resolution of 658x492 pixels (Basler, 2008). An example of a commercial product is the Sony Ipela Cam (see fig 1.5a), which is capable of 15 fps at a resolution of 640x480 and only one rotation (Sony, 2008). A platform which allows the mounting of a



(a) Sony Ipela Cam (Sony, 2008) (b) SPG425A-MR Multi Rotation Platform (Servocity, 2008)

Figure 1.5: Images of commercially available products

user-supplied camera is the SPG425A-MR Multi Rotation (fig 1.5b). This can rotate up to 1400° but only in one plane (Servocity, 2008). Prices for the Ipela Cam start at R15 000 and the SPG425A-MR at R8 000.

1.4 Objectives

Main Objective

The main objective of the project is to build a two-axis continuous rotation camera platform capable of a rotation speed of at least 6.7 RPM, with an accuracy of less than 1°, given the Basler A311fc.

This objective will be achieved by completing the following goals:

- The determination of a means of transferring data and power across both axes.
- The selection of a suitable drive method for rotation.
- The design of the physical platform.
- The determination of a means to monitor the platform.
- The development of the electronics required for control.
- The development of software with which to interface with the platform.

A user interface for the platform would have to be built and meet the following requirements:

- User must be able to see video stream.
- User must be able to set a rotational speed for both planes.
- User must be able to set a position for both planes.

1.5 Overview of Report

This document aims to put forth the steps that were taken in the design of the platform and discuss the success of the final design.

Chapter 2 outlines the major issues in the design of the platform, and looks at various ways to solve each problem. A general solution for each of the problems listed in 1.4 is given.

Chapter 3 takes the ideas from Chapter 2 and focuses on a specific solution for each one, taking the design further by setting specifications and making component decisions for the platform. The custom design of certain hardware aspects is also covered.

Chapter 4 describes the processes followed to obtain results and determine the accuracy of the system. First transfer rates are tested, followed by the sensors and then movement. A summary of electronic measurements is also given.

Chapter 5 gives a conclusion to the report with a summary of the key facts. It compares the objectives to what was achieved and gives recommendations for continued development.

A list of references is then given, followed by appendices that cover budget, timeline, source code, PCBs and other applicable data, including a techno-economic analysis of the project.

Chapter 2

General Design

2.1 Power & Data Transfer

The biggest restraint relating to this project is the need for continuous rotation. Conventional turret systems are limited to a single rotation for this specific reason. The wires that connect data and power lines to the camera and other circuitry will get twisted if the turret is to be rotated continuously. Continued twisting of wires causes them to tighten, and will eventually result in them breaking or cause other mechanical problems in the system. By limiting a turret to only one rotation it removes the risk of this happening.

When looking at continuous rotation, there are two main alternative technologies to investigate: these are wired and wireless methods. Wired methods make use of, for example, brushes and rings like in a motor to maintain a connection. Wireless methods make use of Wi-Fi, other antenna based protocols, or some form of induction based transfer method.

2.1.1 Wireless

There are several different wireless data-transfer technologies which are applicable. Networking protocols such as Bluetooth and Wi-Fi exist. There are also simpler protocols which make use of dedicated sender receiver units such as the Xbee range of products. Furthermore there are also Near Field Communication (NFC) devices.

The biggest drawbacks with any wireless system are the generally lower transfer rates and the higher latencies. During testing of the camera it was found to use, on average, 31.1 MBps. The camera is supposed to be capable of acquiring images of a resolution 640x480 pixels at a rate of 60 fps. This would equate to a transfer rate of:

$$\begin{aligned} \text{Transfer Rate} &= \text{Total Pixels} \times \text{bpp} \times \text{fps} \\ &= (640 \times 480) \times 16 \times 60 \\ &= 294912000 \text{ bps} \\ &= 35.15 \text{ MBps} \end{aligned} \tag{2.1}$$

It was unclear why the camera did not obtain the expected fps value during testing.

Standard Wi-Fi protocols only permit speeds up to 2.5 MBps (Mitchell, 2011). This is far slower than that desired for the video stream. Newer high speed Wi-Fi protocols (which are becoming more common and available) such as IEEE 802.11n allow for speeds in excess of 30 MBps (IEEE, 2010). An NFC solution could have worked well considering the small distances inherent in the design. Unfortunately the standard NFC protocol only deals with transfer rates up to 53 kBps (Nosowitz, 2011).

A new NFC product called TransferJet, in development by Sony, has also been investigated. Specifications for the protocol include transfer speeds of up to 44 MBps (Williams, 2010). It is however unclear how the constant rotation of the antennas relative to each other would influence the system. Further research of the protocol indicated that is is currently only used in Sony specific products, and individual components are not available for public use.

One problem with any wireless protocol would be the necessary alterations required to transfer data from the Firewire protocol into one that could be used by the specific wireless protocol. Furthermore, power would still be required for the camera and some of the other protocols.

Power can also be transferred via induction. This would however complicate matters. Motors draw fairly high currents and the magnetic fields created could interfere with those required for the data transfer.

2.1.2 Wired

Conventional turrets, not capable of continuous rotation, use ordinary wiring to transfer data and power. Those capable of two or three rotations make use of special brackets and harnesses which roll up around the shaft as it rotates.

However for this application it was decided to make use of slip rings. The main reasons for this being their simplicity and promised transfer speeds. Slip rings work on the same principle to what is found in many motors, with a spring loaded brush which maintains a connection with a ring. Generally the brushes are mounted on the outside and are fixed, while the rings are mounted on the shaft and rotate with the shaft. This means wires can be taken from the rings on the inside, and the brushes on the outside, but can remain fixed relative to their body. Fig 2.1 illustrates the concept. The figure is an example of a twelve circuit, through-axis slip ring. That is it has twelve tracks along which different signals can be applied and the axis is open on both ends.

Slip rings are manufactured on a commercial scale by several international companies, and on a design and manufacture basis by several local companies. Quotes were acquired from various companies and Moog were chosen. Moog are an international company with local offices whose large selection of slip rings and relatively competitive prices made it the top choice.

Moog (2010) indicated that their small slip rings could conduct signals up to 6.25 MBps and that special slip rings were capable of speeds up to 62.5 MBps. If small slip rings were to be used data transfer could be an issue, although the 6.25 MBps is still higher than those obtained via wireless methods.

Slip rings were available for order from various suppliers, with lead times of over 3 months expected. For this reason, a decision needed to be made fairly soon in the design process

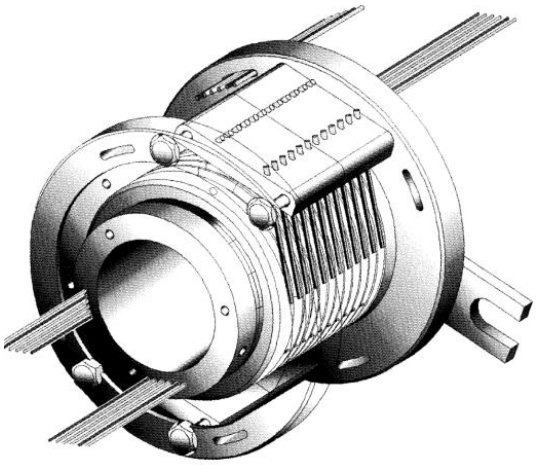


Figure 2.1: Illustration of a slip ring (Moog, 2010)

to ensure that they would arrive in time for implementation with the platform.

Another factor influencing the decision is the high price associated with slip rings. Quotes from different suppliers all returned with prices over R5000 per slip ring. Although this cost is very high, it was felt that working with slip rings would be the simplest solution and the one most likely to achieve the high transfer rates.

2.2 Drive Method

A motor is required to drive the camera in each axis. For tracking purposes the motors are only required to rotate at a relatively slow speed (determined to be 6.7 RPM). Two main options were researched, DC motors and Stepper Motors.

Servo motors are often used in this type of application due to their trait of being able to easily set the angle to which they must rotate. However they were excluded from the options as they are only capable of one rotation. Removing the one rotation limit on a servo motor is possible, but it removes its ability to rotate to a desired angle (Demers, 2008), voiding its benefits.

2.2.1 Torque Conversion

Power and speed ratings for motors vary immensely, and designs often make use of gears, belts or other transfer methods to adapt these ratings to fit in with the design. The downside of using these is that irregularities may occur. Gears have what is known as backlash; this is basically the small amount of free movement between gears when they are stationary. This affects both the control and accuracy of a system. Similarly, belts are known to slip and stretch during use, causing similar issues.

For this project it was desired that, as far as possible, no form of torque conversion should take place and that the respective shafts should be driven directly by their motors. This further reduces the complexity of the system and minimises the overall size of the platform.

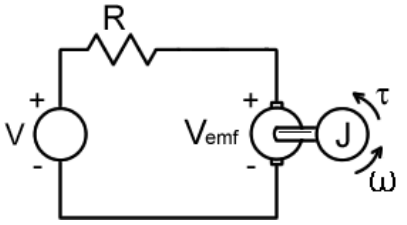


Figure 2.2: A simplified model for a DC Motor. (Petitt, 2003)

2.2.2 DC Motors

DC motors are fairly straightforward to control. DC motors work on the basic principal of the Lorentz force. This is the force that a current-carrying conductor has applied to it when in the presence of a magnetic field. The force is directly proportional to the current in the conductor, as can be seen from the following formula (Wildi, 2002):

$$F = B \cdot l \cdot I \tag{2.2}$$

Assuming the magnetic flux density (B) and length of the conductor (l) are constant. Knowing that

$$\tau = F \cdot r \tag{2.3}$$

We see that torque is directly related to the current flowing through the conductor. A simple model for a DC motor can be seen in fig 2.2. From the figure we see that

$$V = R \cdot I + V_{emf} \tag{2.4}$$

and Petitt (2003) tells us that

$$V_{emf} = K_e \cdot \omega \tag{2.5}$$

where K_e is a constant specific to the motor. From this we can see that V is directly proportional to ω. As such, by controlling the current and voltage across the circuit, one can control the torque and speed of the motor. This is fairly easy to setup, and there are many ICs on the market which allow voltage regulation by means of pulse width modulation (PWM). Directional control is achieved by reversing the polarity of the applied voltage.

One drawback of using DC motors would be the required control system. To know how far to move the motor, one has to know where the motor is at all times. For this a closed loop control system would be required, and as such some form of feedback would be necessary. This makes the accuracy of the system dependent on the sensors. Choosing a sensor with a higher accuracy would mean the accuracy of the system is increased.

Control would be continuous; even when it is desired for the system to be stationary, a small current would have to be present to generate a torque to counter that caused by gravity on the camera.

2.2.3 Stepper Motors

Stepper motors allow one to very precisely control the speed and position of the rotor. A stepper motor only has a set amount of positions at which it can be positioned, this is

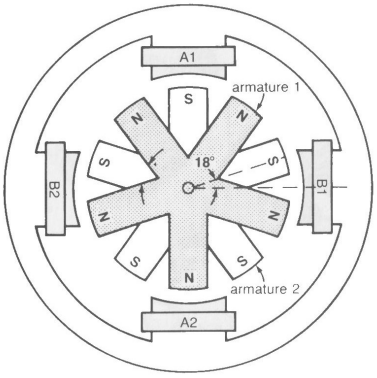


Figure 2.3: Picture of a Bi-polar Hybrid Stepper Motor (Wildi, 2002)

indicated by its step size, or steps per revolution rating. A stepper motor is controlled by pulsing the current through the coils in the motor. By doing this in the correct order one is able to progress the motor one step at a time. To alter the speed at which steps occur, one need only change the frequency of the pulsing. When not being pulsed, the motor will hold its position, even when an external torque is applied to it.

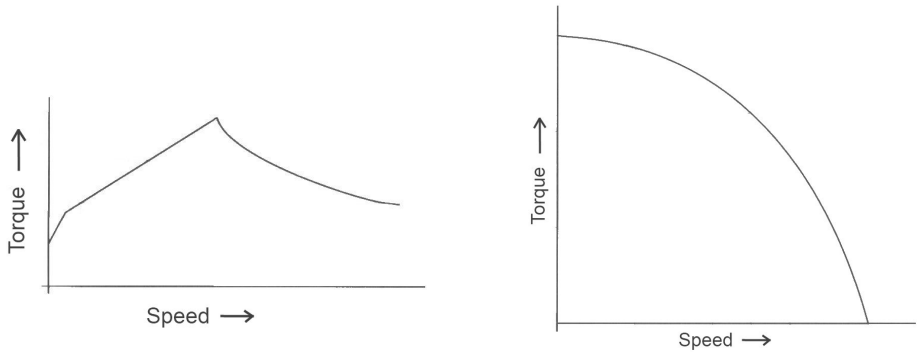
To explain how stepper motors work, one can look at fig 2.3. Figure 2.3 is a representation of a bi-polar hybrid stepper motor. It consists of a permanently magnetised armature and two sets of coils. In the current position coil A would be active, with a north pole generated at A1 and a south pole at A2. By turning off coil A, and turning on coil B, the armature will be attracted to the new poles. By choosing the direction of the current, the north and south poles can be reversed between B1 and B2, thus controlling the direction in which the motor turns. Continuous movement is achieved by successively switching on coils A and B.

The stepper motor in the figure has a step size of 18°; this can be halved by running the motor in half-step mode. In half step mode, the amount of steps possible by the motor is increased by exciting both coils at the same time. This causes the position of equilibrium of forces to be halfway between two normal step positions. By alternating between exciting one and both coils the whole time, half-step drive can be achieved.

One of the major benefits of using stepper motors is that great precision can be obtained without the need for feedback. Knowing the angle of each step, and monitoring the pulses given to the motor, one can calculate the change in angle without the need for a sensor. For example if a motor with a 1.8° step size is pulsed 4 times, we can know that it has progressed 7.8°. And as long as the motor does not slip, it can be assumed that the position is always known. The motor will generally only slip when the motor is accelerated faster than what the torque permits or if something obstructs the motor, preventing it from staying in step.

If one examines the general torque curve of a stepper motor (fig 2.4b), it can be seen that for the low speeds at which the motors will be run in this application, the highest torques are generated. If we compare this to a torque curve of a standard DC motor (fig 2.4a), we see that for low speeds the torque is at its lowest, gradually increasing as speed increases.

Stepper Motors were chosen mainly for their ease of control, their high torque at low speeds and the fact that no gears would be required.



(a) DC Motor (b) Stepper Motor

Figure 2.4: Speed vs Torque Graphs for typical motors (Wildi, 2002)

Stepper motors come in a variety of forms dependent on their internal structure. Uni- and bi-polar motors along with variable reluctance, permanent magnet and hybrid stepper motors. As far as performance is concerned the motors are fairly even when compared to their required power and physical size. Because a high degree of accuracy was desired in the project, hybrid motors were chosen. From research on a supplier's website, a step size of 1.8° was common for hybrid stepper motors whereas most of the other motors had a step size exceeding 7°.

At the stage of design where motors were chosen it was still unclear how the control of the motors would work relating to the slip ring requirements. For this reason the stepper motors with the lowest wire count were chosen. These happened to be bi-polar stepper motors, with four wires controlling two coils.

2.3 Mechanical Structure

To support the camera and other elements, a physical platform had to be designed. Although specific design can be very varied between products, the general design or form of appropriate structures is very limited. The most common design is in the form of a gimbal. According to Dictionary.com (2011), "A gimbal is a contraption, consisting of a ring or base on an axis, that permits an object mounted in it to tilt freely in any direction". They are often found on ships, or other areas where it is required for a device to remain horizontal. The rings rotate around the object, and gravity keeps it horizontal (see fig 2.5). By applying motors to the pivots, instead of letting them rotate freely, it is possible to control the direction that the object faces.

One of the shortfalls of a standard gimbal for use in this application would be that the rings limit the camera's field of vision. By modifying the design, it is possible to remove much of the excess support, but it still generally results in the camera being blocked by the platform when looking in one direction.

An alternative to this is to mount the camera off centre in the one axis, for example off to the right hand side of the gimbal in fig 2.6. By placing the camera on the outside of the platform, the field of vision is not limited at all. One drawback with this design is that the camera no longer rotates on the axis of rotation, but around it. This brings in interesting control concerns when tracking is applied. It also puts the centre of gravity off

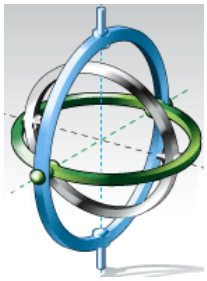


Figure 2.5: Illustration of a standard gimbal setup. (Strickland, 2008)

Figure 2.6: Illustration of a modified gimbal setup. (Amazon.com, 2011)

the axis around which it rotates, requiring higher torque motors and also resulting in a moment on the pan axis.

For the purposes of this project it was decided that it was acceptable for the cameras field of view to be blocked when looking in a certain direction. To keep the design simple a modified gimbal, 'Y' design was selected.

2.4 Monitoring of Platform

Although stepper motors are being used in this project, effectively reducing the need for system feedback, it was still decided to make use of sensors. Although they are not used in the control of the system, they were used during the setup and testing of the platform, and can also be used to check if slipping occurs during operation.

Although there are several different methods with which angular position can be measured, one of the most accurate, easy to use and common methods is to make use of hall effect sensors together with a magnet. Another common measurement system is to use specifically designed rheostats mounted on an axis, and to monitor the change in voltage as the resistance changes. These are fairly cheap and common enough, but the overriding factor here, again, is that continuous rotation is required. Rheostats are generally designed for one rotation only, and those modified for more lose resolution.

Hall effect sensors on the other hand have no physical connection between the rotating and fixed axis, thus they are free to be rotated as often as required. Hall effect sensors work by monitoring changes in a magnetic field, and varying their output accordingly. Thus by placing the hall effect sensor in line with a rotating axis, but mounted stationary,

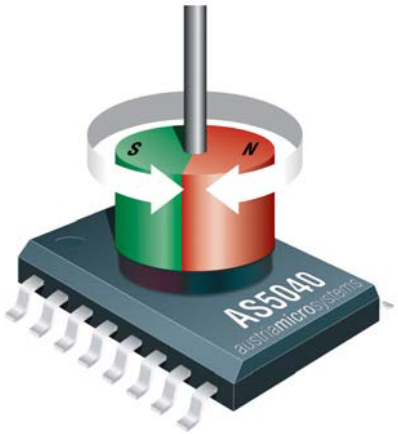


Figure 2.7: Depiction of a di-pole magnet above a hall effect sensor (Austria Micro Systems, 2009)

and placing a di-pole magnet on the rotating axis, the absolute angle can be determined. See fig 2.7 for an illustration.

Free samples of the AS5040 were acquired from Austria Microsystems along with suitable magnets. Magnets were chosen by referencing a magnet selection guide (Austria Microsystems, 2009) and weighing up the factors. NdFeB (grade N35SH) magnets were chosen for from those available as they had the highest coercivity level. Coercivity is effectively the strength of the magnetic field and translates into better readings from hall effect sensors. The motors had holes at the back, which were open to the shaft. This was the ideal mounting point for the magnets. Testing of the ICs proved successful and they were selected for the system. The AS5040 is capable of outputting a 10 bit absolute value digitally via SSI or as an analogue value as a PWM signal. The 10 bit value is equivalent to a resolution of 0.35°, less than the resolution of the stepper motors, 0.45°.

Furthermore the AS5040 requires no calibration or temperature compensation and is also tolerant to misalignments in the axis and changes in the air-gap (Austria Micro Systems, 2009).

2.5 Control Electronics

2.5.1 Drive Control

Because stepper motors were chosen, some form of stepper motor controller was needed to simplify the control of the system. This control could be done from a microcontroller with the correct setup of transistors connected, but would require a fair amount of extra programming, with extended circuitry. Using a motor driver IC also allows for the reduction of wires needed to control the motor. To drive it directly would require four wires from the microcontroller to the motor, whereas using an IC, this can be minimised to only two wires from the microcontroller. The four wires of the motor are then connected to the IC, but the IC could be placed within the middle layer of the platform, thus only 2 wires would be required of the slip ring.

An IC solution was researched, and the ST L297 Stepper Motor Controller IC was selected.

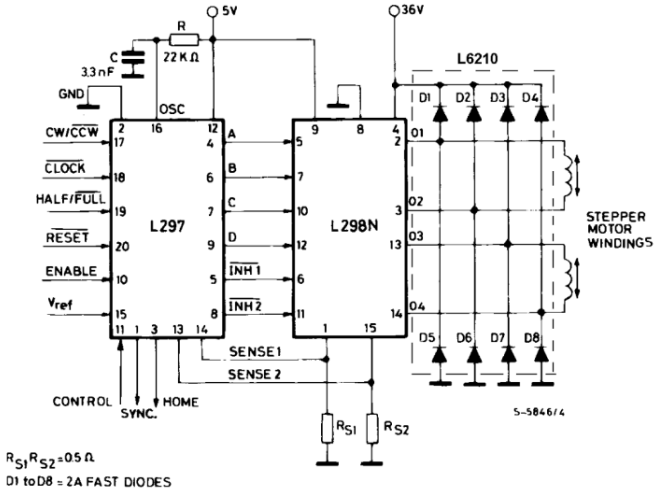


Figure 2.8: Suggested circuit with L297 & L298N working together (STMicroelectronics, 2001)

The L297 has a variety of inputs and outputs which allows customisation for almost any stepper motor. A reference voltage is given to indicate the required voltage drop over one of the stepper motor's coils. This is then controlled by using a PWM signal to drive the motor at a frequency set by an external RC circuit. Sensor resistors are placed in the circuit to set the maximum current per coil (STMicroelectronics, 2001).

The L297 does have one limiting factor, and that is its maximum constant current rating of 0.5 A per phase. The chosen motors required 1.68 A per phase. Fortunately this is a fairly common problem, and ST have a suggested circuit (fig 2.8) making use of the ST L298N Dual Full-Bridge Driver. This increases the maximum constant current to 2 A per phase.

Because the Basler A311fc runs off the FireWire protocol at 12 V, it was decided to design the system around this voltage. The AS5040 requires either 5/3.3 V and the L297 5 V. The L298N requires the same 5 V logic voltage as the L297, but also a higher voltage to run the motors from. 12 V was used for this.

The 5 V was easily achieved by making use of a standard 5 V voltage regulator.

The 12 V supplied by a standard FireWire port would not be sufficient to power the motors along with the camera. For this reason a separate 12 V power supply was required. For this purpose a standard ATX computer power supply unit (PSU) was acquired, capable of 15 A.

2.5.2 Microcontroller

A device was needed to bring the system together, to allow the control of the stepper motors, communicate with the sensors and interface with the user. The most common product to use here is a microcontroller. PLCs (Programmable Logic Controller) are also used to solve this problem, but mainly in the industrial sector on much larger applications. They are designed to be used in heavy duty areas, are generally physically larger than microcontrollers and are also considerably more expensive than microcontrollers.

A microcontroller was thus required to control the system, give the motor driver circuits the correct signals and read the inputs from the sensors. The microcontroller was also required to be the translator between the turret and the computer interface.

There are so many microcontrollers on the market these days that it generally comes down to personal preference, availability and specific requirements. The requirements for this project were fairly general and thus did not limit the choices.

An Arduino Uno was chosen based on some of the following factors:

- No external programmer is required to program it
- It has a built in Serial > USB (also used to program)
- Its ease of programming and the large amount of resources available
- Previous experience with the device
- There were no physical space restraints in the project

2.6 Interface Software

The user interface for the turret would take the form of custom software. The software would communicate with the microcontroller, sending it commands for the turret as well as receiving positional information from the sensors via the microcontroller. The software would use serial communication to communicate with the microcontroller.

It was desired that a GUI (Graphical User Interface) be given to the user. There are a host of different programming languages, all with the capability to communicate serially and develop GUIs. Again it comes done to ease of use and the developer's preference. Java was initially selected as it had a well developed package for dealing with serial communication and GUIs could be developed quickly and with ease.

After several attempts at setting up a serial connection between the software and the microcontroller failed, the topic was researched again and Python came out as a stronger candidate. Further development with Python quickly resulted in a connection with the microcontroller, and the rest of the software was designed around it.

Chapter 3

Specific Design

3.1 Power & Data Transfer

Moog (2010) lists the following factors to take into account during slip ring selection.

- System Interface Requirements - relating to physical design.
- Electrical Requirements - currents, voltages, bandwidth.
- Mechanical Requirements - rotation speed, life.
- Environment - operating temperatures, exposure to elements.

Because the platform itself had not yet been developed, interface requirements were not an issue, and the platform could be designed around the components chosen. The environment was also a minor issue as it would be used indoors at ambient temperatures.

For the 'Y' design of the turret, two slip rings were required. The first one would be used only for FireWire to connect the inner layer to the middle layer. Firewire makes use of six wires, so a six circuit slip ring was required. The Firewire protocol makes use of 12 V power at a maximum current of 0.75 A. Firewire devices are built to draw a maximum of 9W at the predetermined 12V (1394 Trade Association, 2010). From experimentation it was determined that rotation speeds would be less than 12 RPM where majority of the slip rings were capable of speeds up to 150 RPM. An SRA-73526-6 was chosen for this application as it best fit the requirements. It can be seen in figure 3.1a.

The second slip ring would be the connection between the middle layer and the outer layer. It would have to carry the Firewire signals, but also those to power Motor 1 and data from a sensor. The Moog slip-rings come in designs with multiples of six circuits. A six circuit slip ring would obviously not be enough so it was a decision between a twelve or eighteen circuit slip ring.

Research relating to sensors revealed that they could generally have their outputs minimised to one or two signals, and THAT drive control could be done with four or less circuits. It was determined that both the motor and the sensor could be powered from the 12 V line used for the Firewire. The decision of which specific twelve circuit slip ring would be used was left until a later stage when it was known what motors were going to be used. Once this was determined a Moog AC6438 was chosen, with a maximum current rating of 5 A per circuit, as this would be enough to drive the motor.

(a) SRA-73526-6 (b) AC6438

Figure 3.1: The two selected slip rings (Moog, 2010)

Table 3.1: Pin assignment for slip rings

SRA-73526-6		
1	black	FireWire GND
2	brown	FireWire TPB-
3	red	FireWire VCC
4	orange	FireWire TPA+
5	yellow	FireWire TPA-
6	green	FireWire TPB+
AC6438		
1	black	Ground
2	brown	+12 VDC
3	red	FireWire VCC
4	orange	Motor Clock
5	yellow	FireWire TPB+
6	green	Motor Direction
7	blue	FireWire TPB-
8	violet	Sensor Clock
9	grey	FireWire TPA+
10	white	FireWire TPA-
11	wht/blk	Sensor CSn
12	wht/brn	Sensor Data

A breakdown of what the wires in the slip rings were used for can be found in table 3.1.

3.2 Drive Method

Although the type of motor to use was already decided, the specific torque requirements of the motors still had to be determined. Several assumptions had to be made here, relating both to how the physical structure was going to look as well as how it was desired for the motor to react.

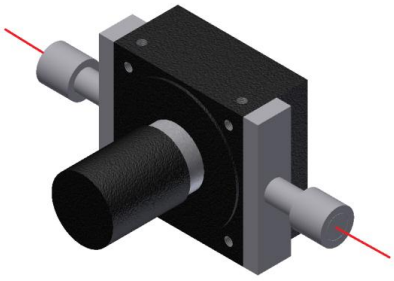


Figure 3.2: CAD model used to calculate the moment of inertia around the tilt-axis

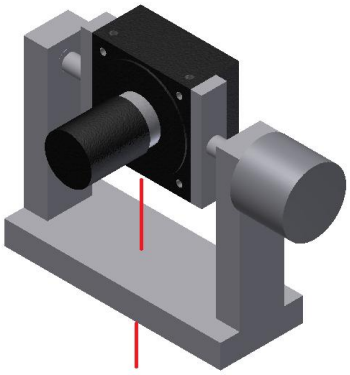


Figure 3.3: CAD model used to calculate the moment of inertia around the pan-axis

An initial CAD model was drawn up using the 'Y' design to determine the structure's moments of inertia. Autodesk Inventor 2011 was used to develop the models. It automatically calculates values such as centre of gravity and moment of inertia. The models used can be seen in figure 3.2 and figure 3.3. The values obtained for their moments of inertia were $1.322\text{x}10^{-4}$ kg·m^2 and $3.5\text{x}10^{-3}$ kg·m^2 respectively.

To calculate the required torque for each of these motors, the following formula from Kulakowski et al. (2008) was used:

$$\tau_I = I \cdot \alpha \tag{3.1}$$

To determine the acceleration (α), it was decided that the motor should be able to accelerate the camera from zero to the desired maximum speed within one step. From earlier calculations the maximum desired speed for tracking was determined to be 40°/s.

Assuming a motor with a 0.45° step size, or 800 steps per revolution, 40°/s equates to 88.9 sps. One step should thus take 0.011 s.

$$\begin{aligned} \alpha &= \omega \div t \\ &= 88.9 \text{ sps} \div 0.0112\ s \\ &= 7937.5 steps \\ &= 3572°/s^2 \\ &= 62 \text{ rad}/s^2 \end{aligned} \tag{3.2}$$

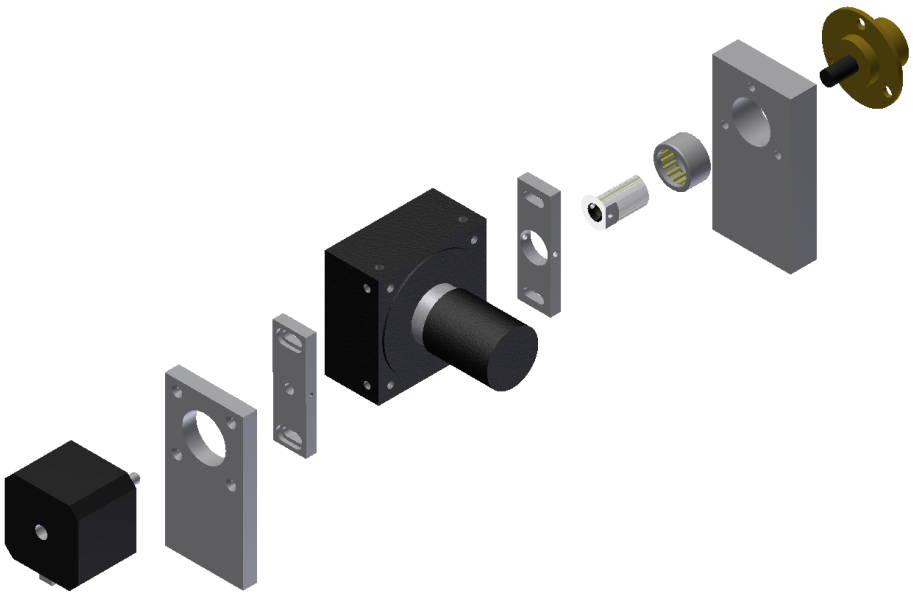


Figure 3.4: This image illustrates the support cylinder for slip ring 1

Using the CAD calculated moments of inertia, the required torque for Motor 1 is:

$$\begin{aligned} \tau_1 &= \alpha \times I \\ &= 62 \times 1.322 \times 10^{-4} \\ &= 8.2 \text{ N} \cdot \text{mm} \end{aligned} \tag{3.3}$$

and for Motor 2

$$\begin{aligned} \tau_2 &= \alpha \times I \\ &= 62 \times 3.5 \times 10^{-3} \\ &= 217 \text{ N} \cdot \text{mm} \end{aligned} \tag{3.4}$$

The motor with the lowest torque rating that met all the specifications and that was in stock with the suppliers, was a 220 N·mm motor. One was purchased for the tilt-axis. The cost of all the motors in that series of motors was fairly constant; the only major difference between the motors was their length. Because this was not an attributing factor to choice of the pan motor, a 440 N·mm motor was purchased giving it a good factor of safety.

The smallest step-size obtainable was 0.9°. When run in half-step mode this results in a 0.45° step size. Two motors manufactured by RS Components were acquired, with the mentioned specifications.

3.3 Mechanical Structure

With the specific Basler camera in mind, it would be fairly straightforward to design a 'Y' form gimbal. The camera is pivoted between two uprights, which allows it to do full rotations as far as tilt is concerned. The upright then rotates on the base to give it its panning ability.

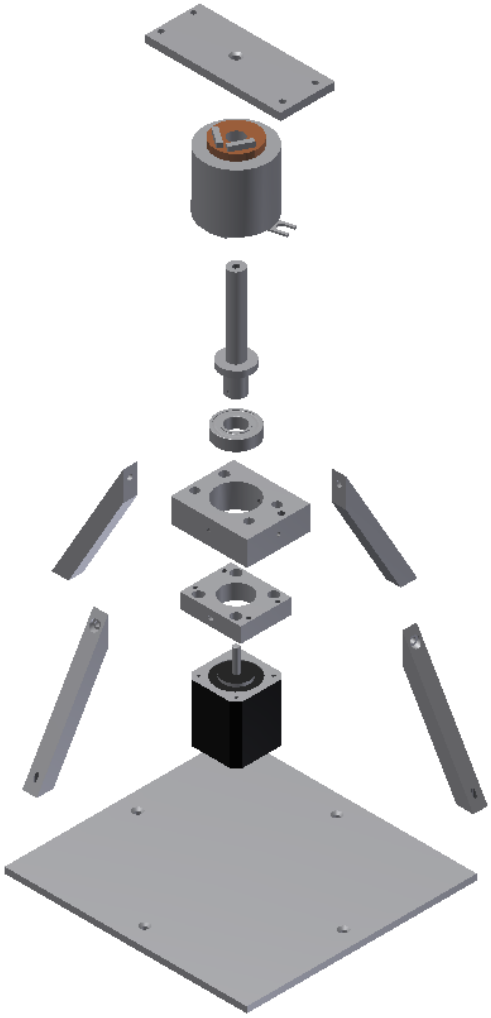


Figure 3.5: This figure illustrates the assembly of the pan-shaft

This design does prevent the camera from seeing in one direction (directly below the camera), but this was deemed acceptable for the project. At this stage of design, the only reference images for the slip rings was a rough 2D diagram of each slip ring from their data sheets. Although not all dimensions were given, fair assumptions as to the design were made and CAD models for the slip rings were developed. Further CAD models were developed for the two motors and one for the camera.

The instructions for the installation of the slip rings indicated that they should only be fixed on one side and that the other side should be free to rotate. This is so as to prevent premature failure due to an induced strain if one side is not centered correctly (Moog, 2010). The slip ring at risk here was that for the tilt, as the initial plan was to support the one side of the camera with the motor and the other with the slip ring. Instead a shaft was designed to fit over the slip ring. It is supported by a needle roller bearing, and allows the slip ring to rotate freely within it. This is illustrated in fig 3.4.

The pan slip ring is of a through-hole design. This means it fits around the rotating shaft. This shaft supports the full weight of the middle and inner layers, and rests on a ball bearing. This is able to support both axial and radial forces. This is illustrated in fig 3.5

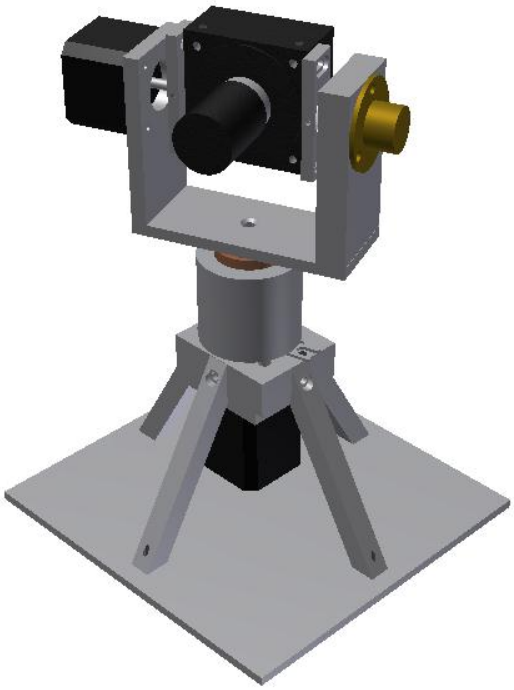


Figure 3.6: Image of fully assembled platform

The structure is held in place by four legs which are attached to a base plate. During the design, manufacturability and easy construction were kept in mind at all times. Furthermore care was taken to counter sink holes where possible, and minimise the visual impact of bolts to give an overall neater appearance.

An assembled image of the platform can be seen in fig 3.6.

Aluminium was chosen to be used for all manufactured parts. The main advantages being its lightness and ease of machining. It is also relatively low cost and doesn't corrode under normal conditions.

> **note:** It is assumed that the forces present in the system are small enough to be ignored with regards to issues of strain and fatigue, when compared to the volume and structural properties of the materials used.

Manufacture of the platform was done by personnel at CES (Central Electronics Services). Third angle projected orthographic drawings were given to them for each manufacturable part. The slip rings, motors and bearings were also given to them to ensure fitting. Minor modifications were made to the design during manufacturing, but the overall structure and parts total was not altered.

3.4 Monitoring of Platform

The pin-out of the AS5040 can be seen in fig 3.7. The majority of the pins were left floating for the setup used, as recommended by the data-sheet for pins not in use. The

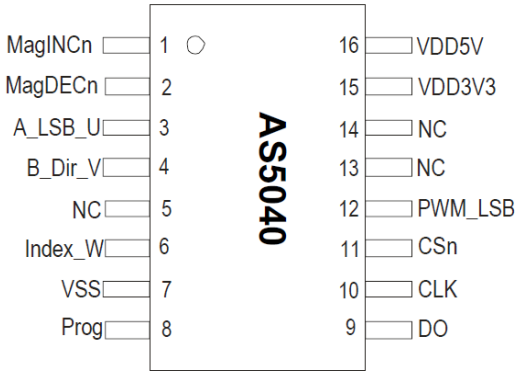


Figure 3.7: Pin-out for AS5040 Hall Effect Sensor

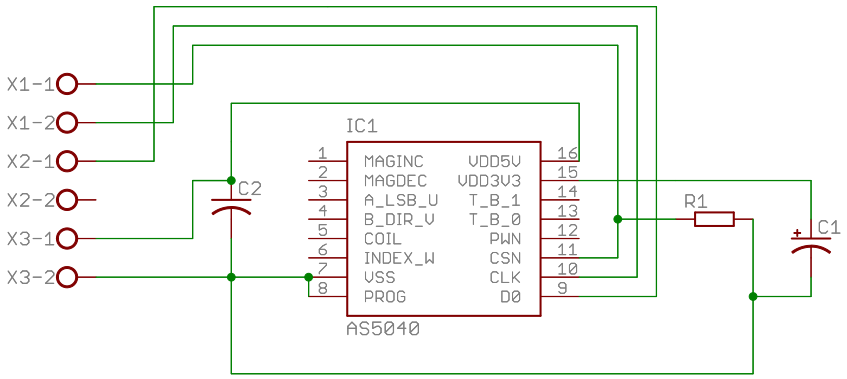


Figure 3.8: Circuit Diagram for AS5040 Hall Effect Sensor

AS5040's SSI connection is used to receive data in this instance, so only the applicable pins were set. They are as follows:

- Prog - Programming was not used for this application, so connected to VSS (GND)
- D0 - Data output line goes to microcontroller
- CLK - Clock input from microcontroller
- CSn - Chip Select, input from microcontroller, also has a 4.7 kΩ resistor between the pin and ground. The pin has an internal pull-up resistor, but must be pulled low externally ($R_{ext} \leq 5$ kΩ)
- VDD3V3 - When in 5 V operation, this pin must be buffered by a capacitor between 2.2–10 μF.
- VDD - 5 V supply voltage given and a 100 nF capacitor was placed between this pin and GND

The full circuit diagram for the AS5040 can be seen in fig 3.8

As mentioned previously, the motors had a hole at the back which exposed the shaft. Mounting the magnets on the shaft was simple enough, and meant there was space to play with to mount the sensors themselves.

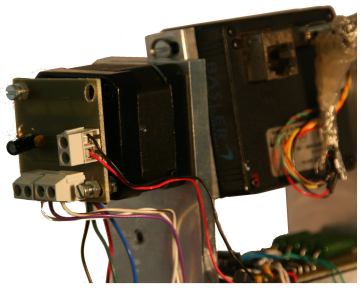


Figure 3.9: Hall Effect Sensor PCBs mounted to back of motors

To ensure the centering of the AS5040 chip around this axis, the PCB was designed to be bolted to the back of the motor (fig 3.9), thus if the design centered the chip on the PCB, by placing holes in the correct place, the chip would automatically be centered around the axis.

The design of the PCB can be seen in Appendix D.2

3.5 Control Electronics

The basic design for the motor controller circuit was given in fig 2.8, as provided by the L297's data sheet.

The decision was taken that only the control signals required to drive the motors should be sent across the slip ring, so as opposed to having just the motor in the middle layer, there would also be the motor driver circuit. The circuit design would then entail minimising the inputs required to successfully drive the motor. For the purposes of this project, the only two things that needed to be controlled were the direction and the clock for step control.

Other pins which were fixed were given the following values:

- Half/!Full - Set high to enable half step mode
- Reset - Set high as circuit need never be reset
- Enable - Set high so as to be always enabled
- V_{ref} - A reference voltage had to be given equal to that required by the motor. Using the 5 V signal, a simple voltage divider was made up using two resistors. The desired V_{ref} was 2.8 V. A 22 kΩ and a 27 kΩ resistor were used.

$$\begin{aligned} V_{ref} &= R_1/(R_1 + R_2) \times 5V \\ &= 27/(22 + 27) \times 5 \\ &= 2.76 \text{ V} \end{aligned} \tag{3.5}$$

- Control - Set high, sets PWM control on motor
- Sync - Output left floating, not used

- Home - Output left floating, not used
- Osc - By adapting this input the frequency of the PWM signal is altered. A formula for the frequency is given by

$$f = 1/(0.69 \cdot R \cdot C) \tag{3.6}$$

  The default values for this, given in the datasheet, are R = 22 kΩ and C = 3.3 nF. This equates to $f = 19.96$ kHz. This was deemed adequate for the purpose.

- Sense - Two sense resistors were required to control the current in the system. The voltage drop over the resistors would be the same as V_{ref} of 2.8V. The desired current was 1.68 A (RS Components, 2006).

$$\begin{aligned} V &= I \cdot R \\ 2.8 &= 1.68 \cdot R \\ R &= 1.66\ \Omega \end{aligned} \tag{3.7}$$

  The resistors used here would have to be high power resistors. The power it would have to dissipate would be equal to:

$$\begin{aligned} P &= V \cdot I \\ &= 2.8 \cdot 1.68 \\ &= 4.7\ \text{W} \end{aligned} \tag{3.8}$$

  To obtain the odd resistance value and minimise the power per resistor, two 3.3 Ω resistors were put in parallel for each of the sense inputs in the circuit, thus giving 1.65 Ω, close to the desired resistance.

The rest of the inputs and outputs on the L297 connected directly with those on the L298N. All the information required for this circuit was acquired from the L297's data sheet (STMicroelectronics, 2001).

The only additional circuitry for the L298N were four pairs of diodes to dissipate remaining currents in the inductance loads of the motor. An IC containing the eight diodes is available in the form of the L6210, but no local supplier was able to acquire one. Instead eight individual fast-switching diodes were used.

From this setup the schematic shown in fig 3.10 is visible. Also visible in the figure is an L78M00 positive voltage regulator including two capacitors as suggested in its data-sheet (STMicroelectronics, 2000).

Because this board was required to be placed in the middle layer of the design, there were restrictions placed on its physical size. The easiest place to put the board was on the flat part of the 'Y' (see fig 3.11). The design just had to ensure that it wasn't too high to impede the rotating of the camera. The board was designed to be slightly smaller than the 88x45 mm available. The L298N handles very high power, and as such a heat sink is required for its continued successful operation. For this reason it was placed on the edge of the PCB, with its back facing outwards so as to be able to make use of the aluminium structure of the turret as a heat sink.

To keep costs low, the same design was used for the motor driver board placed in the outer layer even though the space constraints were no longer applicable. Although the motors provide differing torques the same circuitry could be used for both.

The design of the PCB can be seen in Appendix D.1.

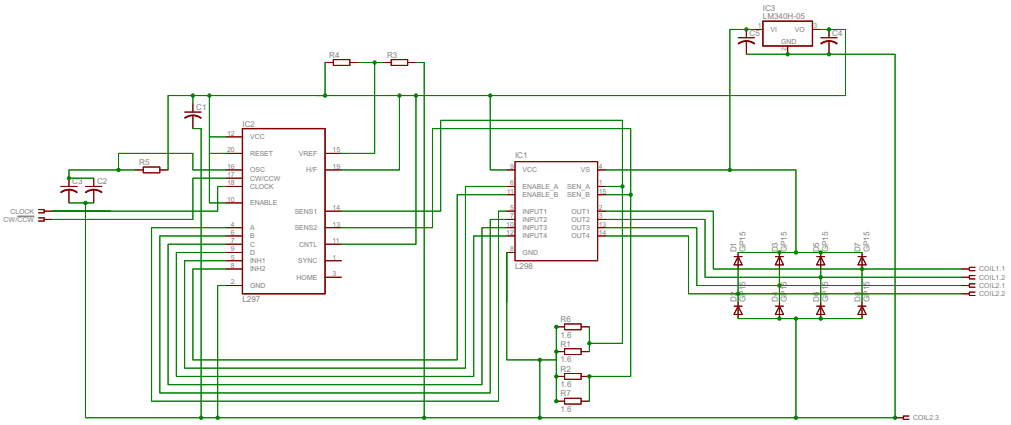


Figure 3.10: Circuit Diagram for Motor Driver Circuit

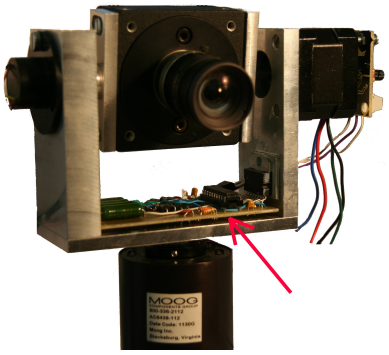


Figure 3.11: Placement of Motor Driver Circuit in middle layer

3.6 Microcontroller

The Arduino requires no extra circuitry and gets power from the same USB connection that it uses for serial communication.

A row of pin headers were acquired that could be soldered onto, and then be plugged into the Arduino. Three sets of headers were made; their correlation to the pins on the Arduino can be seen in table 3.2

A flowchart explaining the main process of the Arduino can be seen in fig 3.12. The source code can be seen in its entirety in Appendix E.1.

The main loop of the Arduino runs in the following manner:

- The serial buffer is read for data. If it contains data in the correct format a second byte is read. These are decoded and a function is called to either set a new speed for the platform, or send the platform to new positions.
- Next the sensors are read. A function is called which processes the SSI data. Four sets of data are then sent to the computer in 2 byte sets. The four sets are from the two sensors, and the two step counts that are kept.
- The current conditions of the platform are compared to the desired ones and certain

Table 3.2: Arduino Pin Outs

Pan

2	-	Motor Clock
3	-	Motor CWCCW
4	-	Sensor Select
5	-	Sensor Clock
6	-	Sensor Data

Tilt

8	-	Motor Clock
9	-	Motor CWCCW
10	-	Sensor Select
11	-	Sensor Clock
12	-	Sensor Data

Power

GND	-	System Ground
5V	-	Tilt Sensor +5V

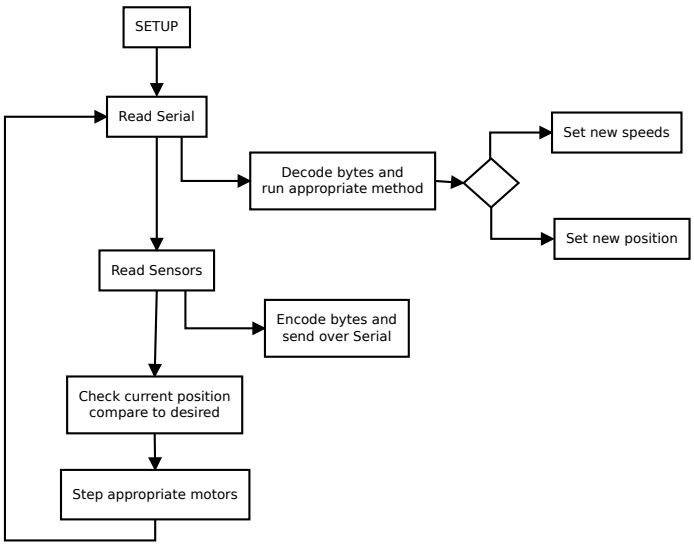


Figure 3.12: Flowchart reflecting the main process run on the Arduino

Table 3.3: Sensor Byte encoding

Byte 1 - 1YYZ ZXXX Byte 2 - 0XXX XXXX Where YY indicates which sensor (01 for sensor 1 and 10 for sensor2), ZZ indicates an error (00 for no error, 01 for offset, 10 for overflow or linearity and 11 for range) and XXXXXXXXXX is the 10 bit sensor reading.

variables are set

- If the motors need to move on a step, this is done next. It is based on the time since the last step, and every half this time, the pin is toggled between high and low. The motors step whenever the pin goes low.
- Then the serial buffer is checked again.

The format for encoding the position can be seen in table 3.3

The code to read the SSI input from the rotary encoders is based loosely on code obtained at RepRap (2010) and was made available under a GNU licence.

3.7 Interface Software

For debugging purposes the angles from the rotary encoders were also displayed in the software, along with where the motors should be (from counting steps) and the angles that the motors are going to (from user input).

Python is compatible with various operating systems, has APIs for handling serial communication and GUI, and it also supports the OpenCV library (for video processing).

A flowchart explaining the main process run by the software can be seen in fig 3.13. The source code can be seen in its entirety in Appendix E.2.

A screenshot from the software can be seen in fig 3.14

The software operates in the following method:

- When the program starts a GUI is brought up.
- The user is required to create the serial connection via the menu option.
- At this point the serial connection is created and a new thread is started which continuously waits for data to be received on the serial port.
- If it receives data in the correct form, it decodes it, updates the GUI and stores the information in a CSV file. The information relates to positional data from the Arduino.
- At the same time the GUI gives the user the ability to view positional data and control the platform.
- When input data is confirmed the data is encoded (using the format described in table 3.4) and then sent via the serial connection to the Arduino.

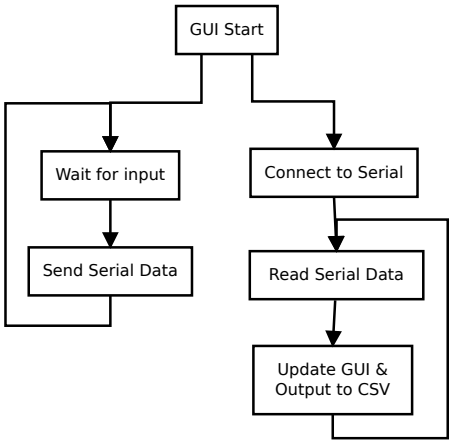


Figure 3.13: Flowchart reflecting the main process of the software

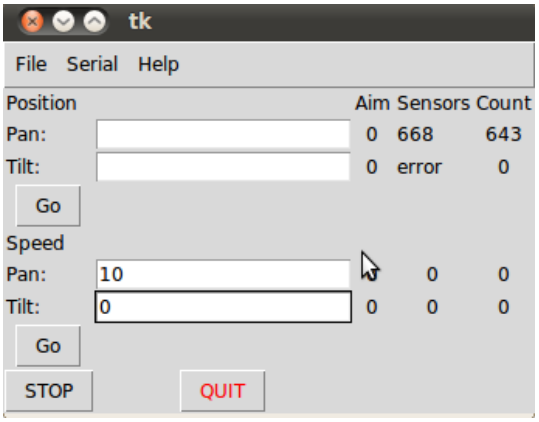


Figure 3.14: A screenshot taken from the user interface software

Table 3.4: Command Byte Encoding

Byte 1 - 1YXX XXXX Byte 2 - 0XXX XWWW Byte 3 - 0WWW WWWW Where Y is a 1 to indicate angle, and 0 to indicate speed. XXXXXXXXXX is a 10 bit value for pan and WWWWWWWWWW for tilt. Although direction of a speed cannot be specified directly, this can be achieved by first setting a position in the direction in which you wish to travel and then setting the speed

Fabio (2010) was referenced when the setting up of serial communication was done from Python.

Chapter 4

Testing & Results

4.1 Data Transfer

To monitor data transfer rates Basler's Pylon Viewer software was used. From the software several aspects of the camera can be altered and monitored. A screenshot of the software, with camera in use, can be seen in fig 4.1. Along the left hand side one can set the bits per pixel, exposure time and resolution of the image. These all have an affect on the data transfer rate.

The live capture rate and bandwidth usage were available on the bottom right, below the image. Although it was desired to log this information for comparison, no solution could be found to enable this. While testing, these values were monitored for changes.

Making use of a standard shielded twisted-pair 2 m long Firewire cable, a maximum data

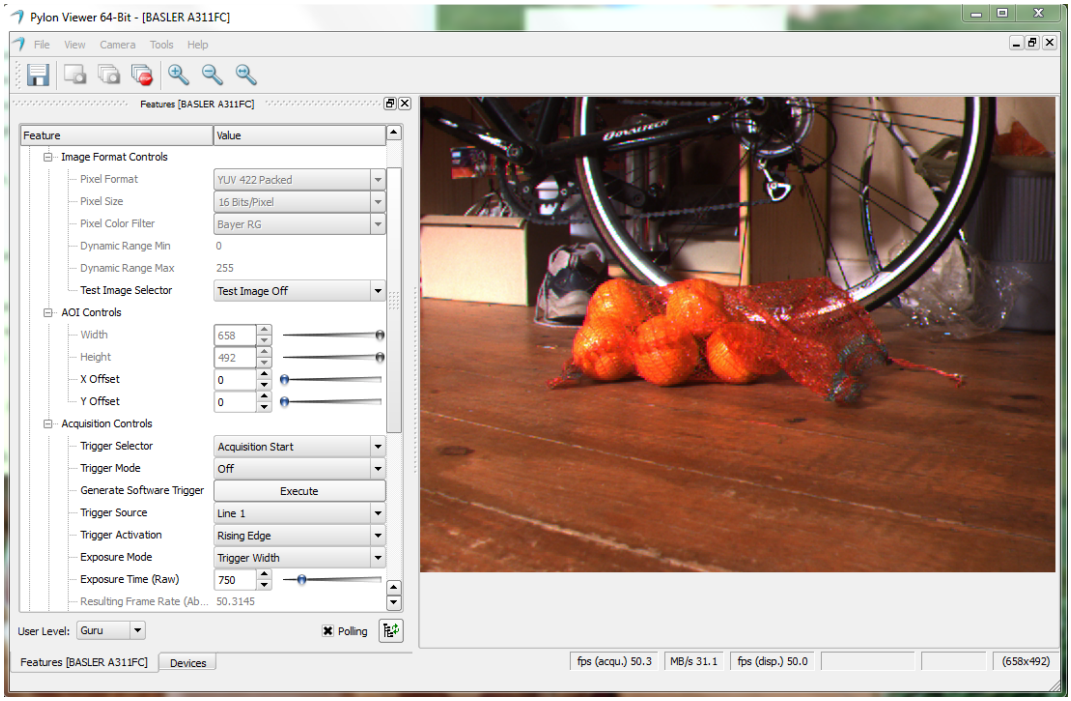


Figure 4.1: Screenshot from Basler's Pylon Viewer Software

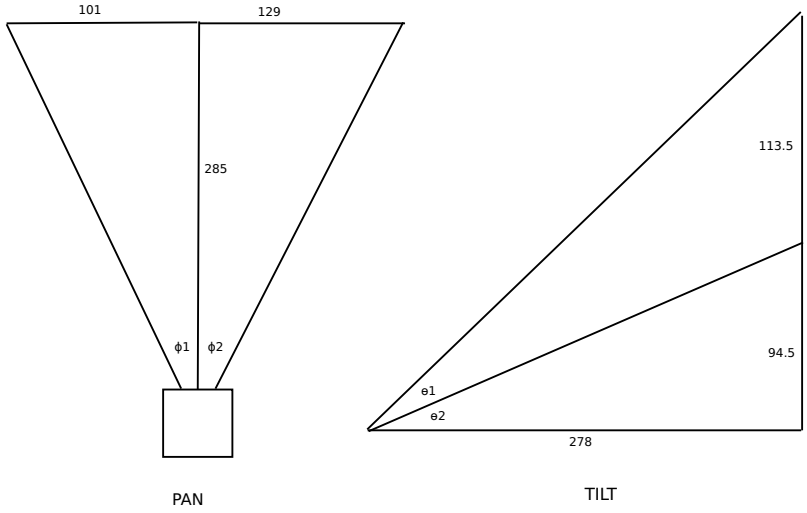


Figure 4.2: Image demonstrating setup for sensor testing

transfer rate of 31.1 MBps was obtained. It is unsure what the limiting factor was here, as Firewire is theoretically capable of speeds upwards of 49 MBps. The cameras instructions and datasheets gave no indication of the actual expected transfer rates.

This cable was modified for use with the platform. The cable was cut in half, with one half being used to connect the camera to the slip rings, and the other half to connect the other side of the slip rings to the computer. To minimise the impact from cutting the wire, the data wires from the slip rings were also twisted in their pairs. Once assembled, tests of the data transfer were run.

The platform was tested both when driven by the motors and when merely pushed with the hand. By hand, rotational speeds of over 200 RPM were obtained, and when driven by the motors, of over 40 RPM. This was in both planes. Tests were done at a resolution of 658x492 at 16 bpp and 50.3 fps. Throughout the testing the bandwidth remained at 31.1 MBps and fps remained constant, with the occasional drop to 49 fps.

4.2 Sensor Accuracy

To determine the accuracy of the the sensors, the platform was placed at a specific distance from a wall, marks were made on the wall and the camera was moved to these marks. A cross-hair was overlayed on the video feed to determine the point at which the camera was facing. Notes were taken of the change in the angle given by the sensors and these values were compared to those calculated with trigonometric functions, based on the geometry in fig 4.2. Table 4.1 shows the outcome of the test.

The results from the pan sensor are fairly accurate, with the difference between the measured and sensors values being 0.2° and 0.6° respectively. The results from the tilt sensor are slightly better with a 0.2° and 0.3° difference in the angles. These differences can be put down to the sensors 0.35° resolution and claimed accuracy of ±0.5° (Austria Micro Systems, 2009).

Table 4.1: Table reflecting sensor accuracy

Angle	Measured	Sensor
ϕ_1	19.5°	19.3°
ϕ_2	24.4°	25°
θ_1	18.8°	18.6°
θ_2	18.0°	18.3°

4.3 Movement Results

Because the system was run in an open loop format, one never knows if the system is in the correct position compared to what it should be. The software was designed so present both the sensor readings of the state of the platform as well as where the system thinks it is from counting steps.

Tests were performed on both the pan and tilt functions of the platform. For both of these, tests were performed for the speed and positional characteristics.

During testing a problem developed with the motor driver board for the tilt shaft which resulted in erratic movements and prevented control of that axis. For testing purposes the driver board used for panning was also used for tilting. This did mean that the system could not be tested for both pan and tilt movement at the same time. This should, however, not be an issue, as the two are controlled independently and the fact that each one works on its own, means it can be assumed that they would work together as well. The problem was determined to be caused by a malfunctioning IC that was damaged due to a short. New components could not be acquired in time.

4.3.1 Pan Tests

Figures 4.3 and 4.4 show data obtained during testing of pan movement. Both graphs consist of three plots. Firstly the desired position or speed for the platform, secondly the position that the microcontroller believes it's at, and thirdly the sensor data.

Fig 4.3 is a set of results when sending commands to the platform to go to specific positions. The blue line is the desired position, and the graph shows how both the clock and sensor values chase after the desired angle. The graph does look peculiar in one aspect, and that is the vertical lines of the sensor and clock which indicate that the platform went through the 360° position.

The system is designed to take the shortest distance to the position requested, so sets its direction automatically. The motor goes at a set speed (40°/s) until it reaches what it believes to be the correct position. For this reason the clock and desired values line up exactly. By looking at the raw data it can be seen that for the majority of the time when the system is stable, the sensor value is within 0.6° of the clock value. This indicates a good level of accuracy in the system. One concerning feature is when t = 97 s, and there is a 3° difference between the sensor and clock values. This could be as a result of the sensor being slightly misaligned, which would lead it to be accurate for majority of positions, but out at certain points.

What the graph does present quite clearly is the settling time, per se. Because of the way the system was designed, speed is constant. There are no curves present which could

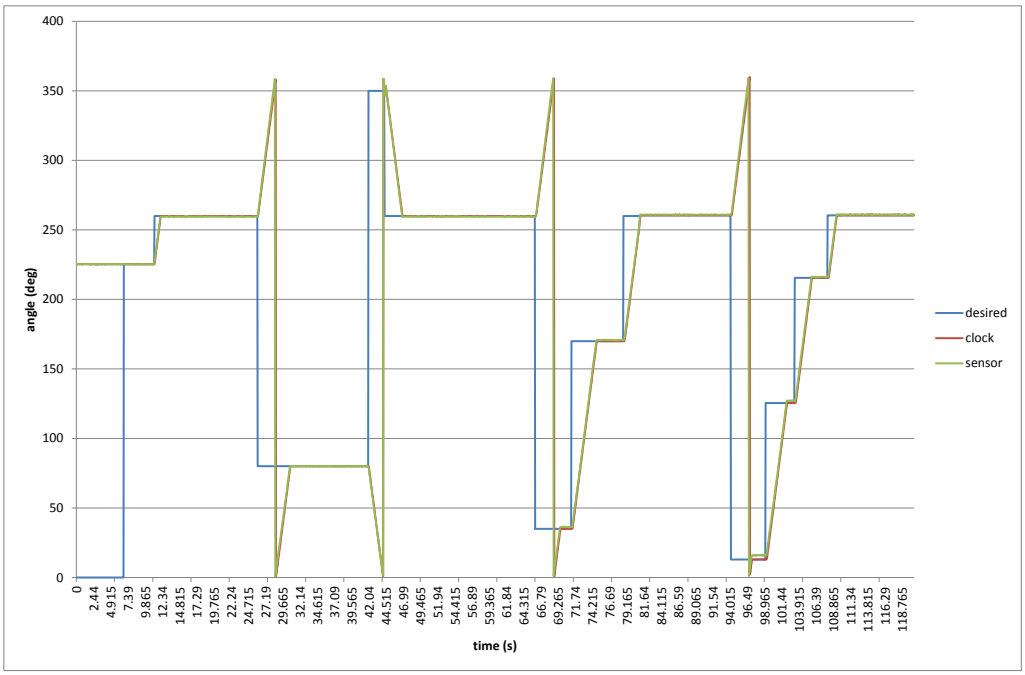


Figure 4.3: Graph indicating changing position readings for Pan shaft

indicate the slowing down or speeding up of the motors as the distance between there current and desired positions change. As a result, the settling time of the system is based purely on the distance from the desired position and the constant speed at which it travels. This can be formulated into the following:

$$t_{set} = \frac{\theta_{desired} - \theta_{current}}{\omega} \tag{4.1}$$

where $\omega = 40°/s$.

Another common factor in control systems, overshoot, is also left nullified due to the control method. Because the system was designed to go from standstill to 40°/s and back to zero within individual steps, the system does not overshoot its desired position.

When looking at fig 4.4 similar observations are made. Firstly there is no settling time. The motor starts travelling at the desired speed almost immediately. The system was told to go to the following speeds progressively, returning to 0 each time, 10, 20, 40, 60, 100, 150 and 200°/s. These can be clearly seen in the graph.

One thing to notice is the apparent oscillation of the clock and sensor readings. This has to do with the limited resolution of both the sensor and the clock readings, as well as the discrete nature of the recorded values. Data was sent from the microcontroller to the computer once every 50 ms. To get the data for the speed graphs, the positional data was differentiated by taking the difference between two successive positional readings and dividing it by the time difference. As a result, depending on the exact moment that data was sampled, you get large changes in position, followed by small ones.

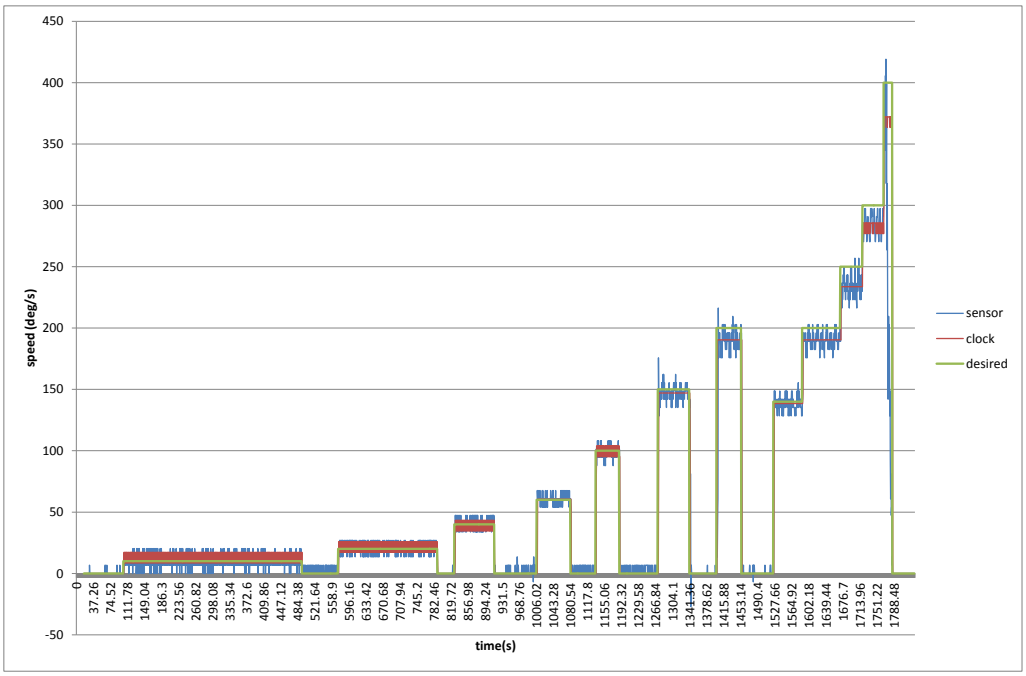


Figure 4.4: Graph indicating changing speed readings for Pan shaft

Table 4.2: Speed results for Pan Movement in °/s

desired	sensor	clock	error
10	10.4	10.4	4.0%
20	20.7	20.6	3.5%
40	40.8	41.0	2.5%
60	60.7	60.6	1.2%
100	99.8	99.9	0.2%

Table 4.2 gives a summary of the graph, when the average speed readings are calculated over the time that it was rotating at a specific speed. These results are fairly accurate, with a maximum deviation of 4%.

When the speed was set to 150°/s, the motor slipped at first (this was noted visually, but can also be seen by the spike in the sensor speed), and then recovered. This implies that the torque that the motor was generating was not enough to accelerate the platform to the desired step within one step. This means that the positional accuracy of the system is compromised, as the system thinks it is somewhere different to where it actually is. The last set of results were obtained by ramping the system. It achieved 140°/s in one step, and slipping only occurred when 400° was attempted. It should be noted that the ability of the system to achieve these speeds was not successful, with the actual speed being 5% less than desired. It was however not expected or required for the system to achieve such high speeds.

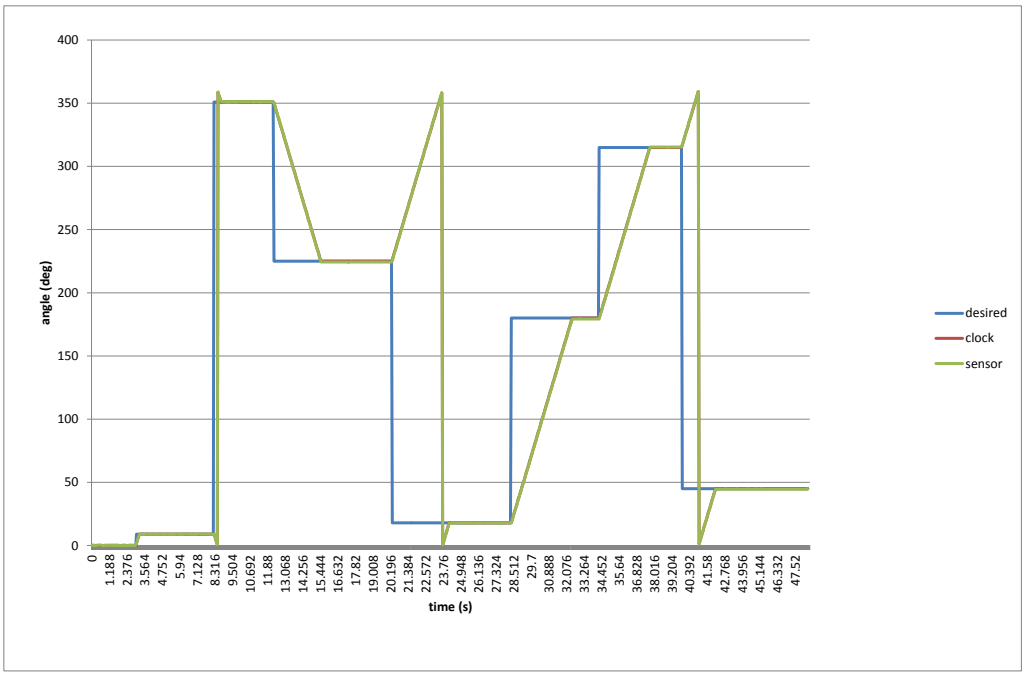


Figure 4.5: Graph indicating changing position readings for Pan shaft

Table 4.3: Speed results for Tilt Movement in °/s

desired	sensor	clock	error
10	10.4	10.0	4.0%
30	28.9	29.2	3.7%
40	39.5	39.4	1.5%
60	57.1	58.3	4.8%
100	95.5	95.6	4.5%
200	183.3	183.3	8.4%
400	358.2	358.1	10.5%

4.3.2 Tilt Tests

As with the panning, the effectiveness of the tilt function of the platform was tested by first commanding it to move to different positions, and then to run at different speeds. Fig 4.5 represents the data obtained when testing the positional accuracy of the platform in the tilt plane. The results are practically the same as those obtained for the pan tests.

As the desired angle changes, so the readings from the clock and sensor follow after it until the correct value is achieved. The same constant speed is maintained when moving to a new position and as such the same formula as in equation 4.1 is applicable for the settling time. The overshoot is zero. When viewing the raw data it can be seen that the difference between the clock and sensor value is never more than 0.6°. The same as for the pan movement.

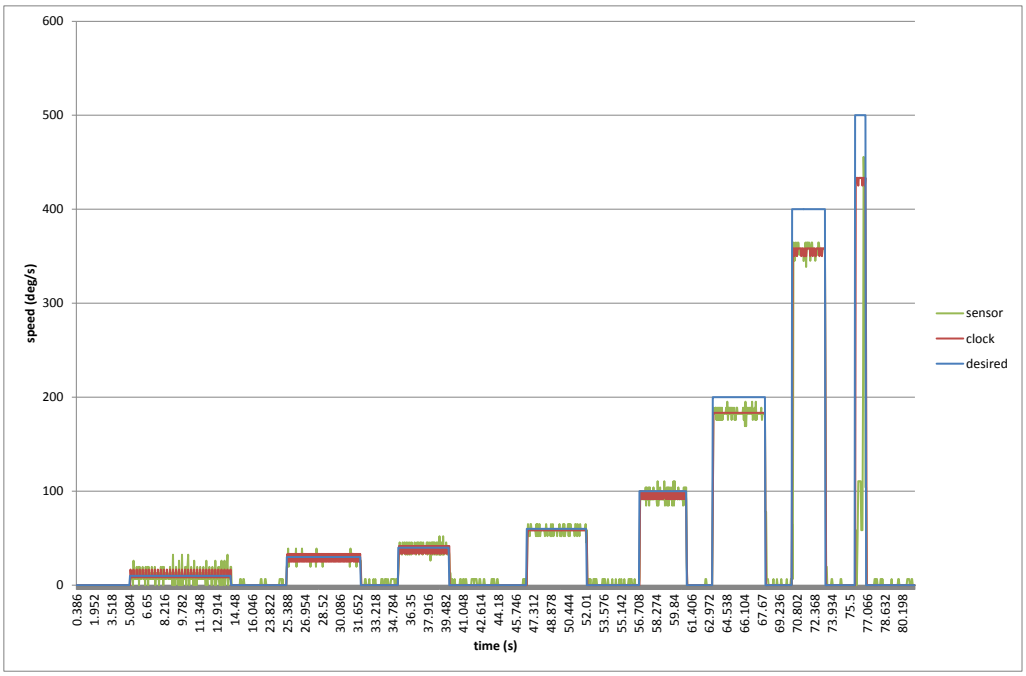


Figure 4.6: Graph indicating changing speed readings for Pan shaft

Next, the data relating to moving at a fixed speed is analysed. As with the pan movement, the sensor and clock readings look fairly eratic. Once again the motor speed was slowly increased, returning to zero between each step. A compilation of the average speed readings over each section can be seen in table 4.3. From this we can see that up to 100°/s the achieved speed is fairly accurate.

To test the extremes of the turret, commands were sent for it to move at 200, 400 and 500°/s. From visual inspection of the platform it did not appear to slip until it attempted 500°/s. This is supported by the sensor data. However the achieved speed is noticeably less than that which was desired. This is evidenced by the fact that both the clock and sensor have similar readings. This leads one to declare that there must be an error in either the software or the methods used to acquire the data.

4.4 Electronic Characteristics

The motors were designed to operate at 1.68 A per phase. When run in half step mode, the motors would switch between using both phases, and just one phase each step, meaning the circuit would switch between using 1.68 A and 3.36 A. Measuring the current drawn while the motors were stationary either indicated a 1.61 A or a 3.18 A current, depending on where the motor stopped. While rotating, each motor board circuit drew 2.29 A.

Each sensor board drew 22.1 mA regardless of whether the motor was turning or whether information was being read from the IC.

Chapter 5

Conclusion

This document put forth the steps followed in the development of a two-axis continuous rotation camera platform. The platform was successfully designed to accommodate the Basler A311fc, and permitted continuous rotation while maintaining a steady FireWire connection with a computer without compromising the data transfer.

A suitable drive method was selected that theoretically enabled 0.5° accuracy. Hall effect sensors were selected to monitor the platform for testing purposes. It was determined that as long as the motors are not extended beyond their torque capabilities, there is no need for the sensors.

A software interface was developed that permitted the user to set a rotational speed for each axis or allowed the user to move the platform to a specific position. Furthermore the software displayed live data relating to the position of the platform. Although the software itself did not display the video image, software is available which displays the video feed alongside the control software.

A microcontroller was used, and software developed for it that connected the various electronic aspects of the project together, and also managed the control of the platform.

The end product met all the requirements for the project (rotational speeds of up to 120 RPM were achieved where only 6.7 RPM was required and the positional accuracy of less than 1° was also achieved), although several improvements can still be made. Although this was expected for the tilt motor, the pan motor was only designed with a safety factor of 2 in mind. This over performance could be assigned to a miscalculation in the moment of inertia of the platform, including an error in the original model.

At certain speeds, while visually monitoring the platform, the platform appears to shudder as it rotates. Although the video feed is not affected too badly, it is not ideal and also causes excess noise. The structural rigidity of the system can be investigated to improve this matter. Part of the problem is due to the discrete nature of the stepper motors, in that the control steps it, then it waits till it receives the next step command. This is simply the nature of stepper motors, and although it was considered during the design, it was not expected to have a major effect on the system due to the small step size. The software developed is fairly rough and can be expanded to better handle exceptions, display the speed of the platform, incorporate the video stream and correct other small issues.

One expected continuation from this project is the development of software and a model for tracking moving objects. New software will most likely have to be developed for this,

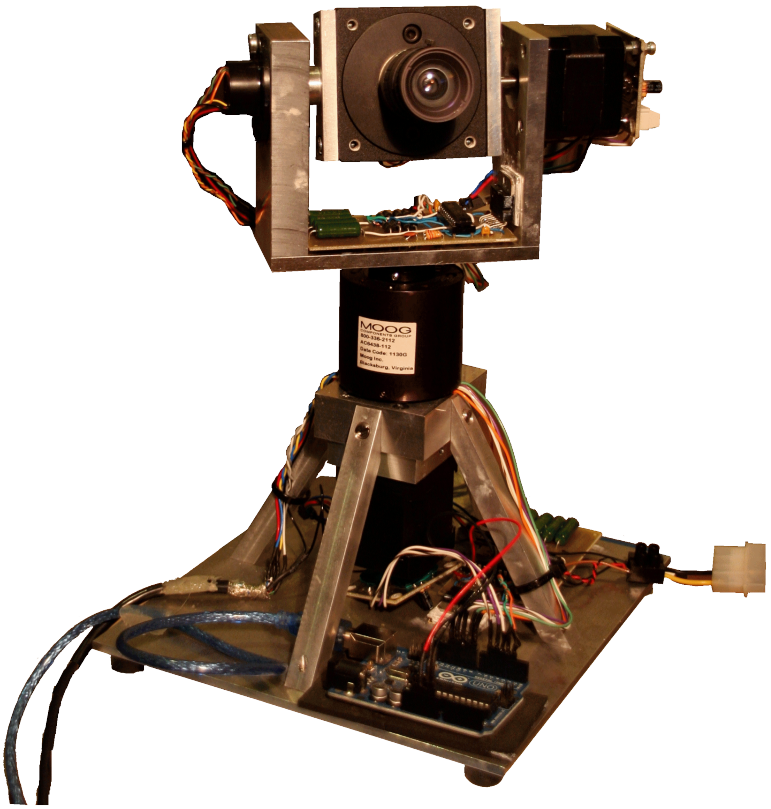


Figure 5.1: Photo of the assembled platform

and alterations to the control software on the microcontroller may also be required. The assemebled platform can be seen in fig 5.1

List of References

1394 Store.com. Basler a311fc. Online, 2009. URL http://www.1394store.com/eshop/assets/product_images/ba102.jpg.

1394 Trade Association. *Firewire Design Guide*, 2010. URL http://www.1394ta.org/developers/DesignGuide/TAFWDesignGuide.pdf.

Amazon.com. Induro ghb2 gh-series aluminium gimbal head. Online, 2011. URL http://www.amazon.com/Induro-GHB2-Aluminum-Gimbal-Tripods/dp/B002W08N4G?tag=food1d1-20.

Austria Micro Systems. *10bit 360° Programmable Magnetic Rotary Encoder*, 2009.

Austria Microsystems. *Magnet Selection Guide*, 2009.

Vision Technologies Basler. Basler a310f user's manual. Technical report, Basler Vision Technologies, December 2008. URL http://www.baslerweb.com.

J Demers. *How to modify a servo motor for continuous rotation*, 2008. URL http://www.ranchbots.com/club/papers/Modifying%2520Hobby%2520Servo%2520Motors%2520for%2520Continuous%2520Rotation.pdf.

Dictionary.com. Define gimbals. Online, 2011. URL http://dictionary.reference.com/browse/gimbals.

Fabio. Serial rs232 connections in python. Online, 2010. URL http://www.varesano.net/blog/fabio/serial%20rs232%20connections%20python.

IEEE. Ieee standard - 802.11n-209. Online, 2010. URL http://standards.ieee.org/findstds/standard/802.11n-2009.html.

Bohdan T Kulakowski, John F Gardner, and J. Lowen Shearer;. *Dynamic Modelign and Control of Engineering Systems*. Cambridge University Press, 2008.

Bradley Mitchell. How fast is 902.11g wi-fi networking. Online, 2011. URL http://compnetworking.about.com/od/wirelessfaqs/f/howfastis80211g.htm.

Components Group Moog. Motion technology catalog. Print, 2010.

Dan Nosowitz. Everythign you need to know about near field communication. Online, January 2011. URL http://www.popsci.com/gadgets/article/2011-02/near-field-communication-helping-your-smartphone-replace-your-wallet-2010/.

Josh Petitt. Embedded systems. Online, 2003. URL http://robotics.ee.uwa.edu.au/courses/embedded/tutorials/tutorials/tutorial_6/Tutorial_6.htm.

Research Foundation RepRap. Magnetic rotary encoder 1.0. Online, 2010. URL `http://reprap.org/wiki/Magnetic_Rotary_Encoder_1.0#Digital_SSI_Output`.

RS Components. *RS Data Sheet 42mm 1.8' High Torque Stepper*, 2006.

Servocity. Spg425a-mr multi-rotation. Online, 2008. URL `http://www.servocity.com/html/spg425a-mr_multi-rotation.html`.

Sony. Sony product detail page sncrz25n. Online, 2008. URL `http://pro.sony.com/bbsc/ssr/productSNCRZ25N/`.

STMicroelectronics. *L78M00 Series Positive Voltage Regulators*, 2000.

STMicroelectronics. *L297 Stepper Motor Controllers*, 2001.

Johan Strickland. What is a gimbal – and what does it have to do with nasa? Online, 2008. URL `http://science.howstuffworks.com/gimbal1.htm`.

Theodore Wildi. *Electrical Machines, Drives, and Power Systems*. Prentice Hall, New Jersey, 5th edition edition, 2002.

Martyn Williams. Sony to launch first transferjet devices this week. Online, Jan 2010. URL `http://www.pcworld.com/businesscenter/article/187110/sony_to_launch_first_transferjet_devices_this_week.html`.

Appendix A

Budget

Below can be seen a breakdown of costs for the project, both the budgeted (A.1) and actual costs (A.2) as well as a specific breakdown of the component costs (A.3).

Table A.1: Forecast Budget for development

Activity	**Engineering Time**		**Running Cost**	**Facility Use**	**Capital Costs**	**CES**			**Total**
						Labour		**Material**	
	hr	*R*	*R*	*R*	*R*	*hr*	*R*	*R*	*R*
Literature Study	40	14000	300						14300
Data and Drive Identification	40	14000	500		200				14700
Platform Design	40	14000	100	200					14300
Develop Model	50	17500		500					18000
Design Review	5	1750							1750
Acquire Components	10	3500	200		2000				5700
Manufacture	30	10500	200	0	500	20	5000	200	16400
Develop Software	200	70000	1000	2000					73000
Final Report	60	21000							21000
Total	475	166250	2300	2700	2700	20	5000	200	179200

Table A.2: Cost Breakdown of Project

Activity	**Engineering Time**		**Running Cost**	**Facility Use**	**Capital Costs**	**CES**			**Total**
						Labour		**Material**	
	hr	*R*	*R*	*R*	*R*	*hr*	*R*	*R*	*R*
Literature Study	40	14000	300						14300
Data and Drive Identification	40	14000	500		200				14700
Platform Design	70	24500	100						24600
Design Review	20	7000							7000
Acquire Components	20	7000	200		16341				23541
Manufacture	30	10500	200		50	40	6000	200	16950
PCB Development	30	10500			1010				11510
Develop Software	110	38500	1000	2000					41500
Final Report	90	31500							31500
Total	450	157500	2300	2000	17601	40	6000	200	185601

Table A.3: Component Costs

Components			
	Slip Rings	R	14 579.00
	Motors	R	886.85
	PCBs	R	1 010.00
	Arduino	R	240.94
	Miscellaneous	R	634.32
Platform			
	Labour	R	6 000.00
	Material	R	200.00
	Miscellaneous	R	40.00
Total		R	23 591.11

Appendix B

Gantt Chart

Fig B.1 gives an indication on the planned progression of the project and compares it to the actual progress.

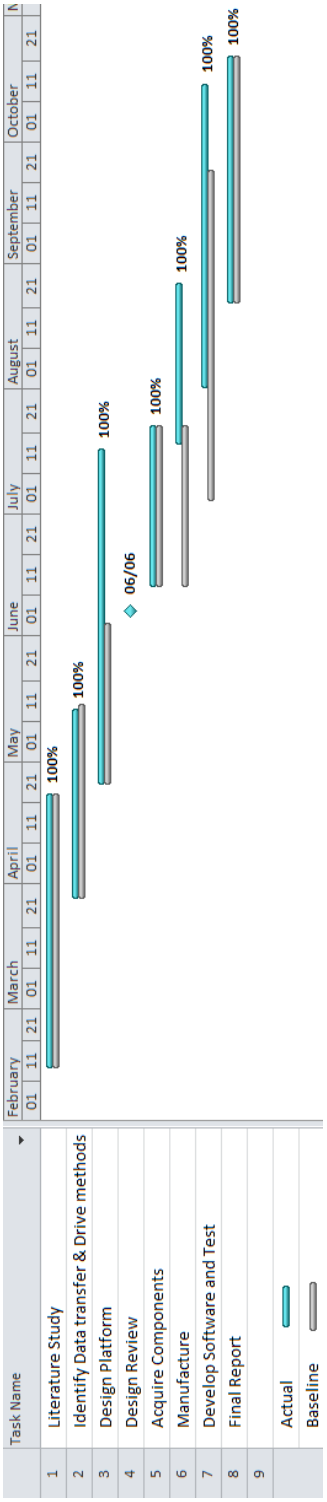


Figure B.1: Gantt Chart displaying project progress

Appendix C

Techno-Economic analysis

The key aspect of this project is its uniqueness. Although there are products which achieve some of the objectives set out in this project, none of them achieve them all. It must further be pointed out that this was a developmental project and the amount of labour results in a product that is far more costly than commercial products.

The total cost of the project came to R185 601 (see table A.2), of this R23 591 fell under capital and manufacturing costs (see table A.3). The budgeted capital and manufacturing costs were far less than this, mainly due to the costs incurred in acquiring slip rings.

Although less hours were billed than originally planned, the total cost of the project still works out R6 000 more expensive than budgeted (see table A.1). Once again this is due to the unforseen cost of the slip rings.

From a timescale perspective, the major factor was the time taken to design the platform itself. The start of this was delayed slightly while waiting for information relating to the slip rings. During the June/July holiday only minimal work was done on the project, which resulted in the platform only getting manufacture during August. This resulted in a delay of the development of the software and subsequently testing.

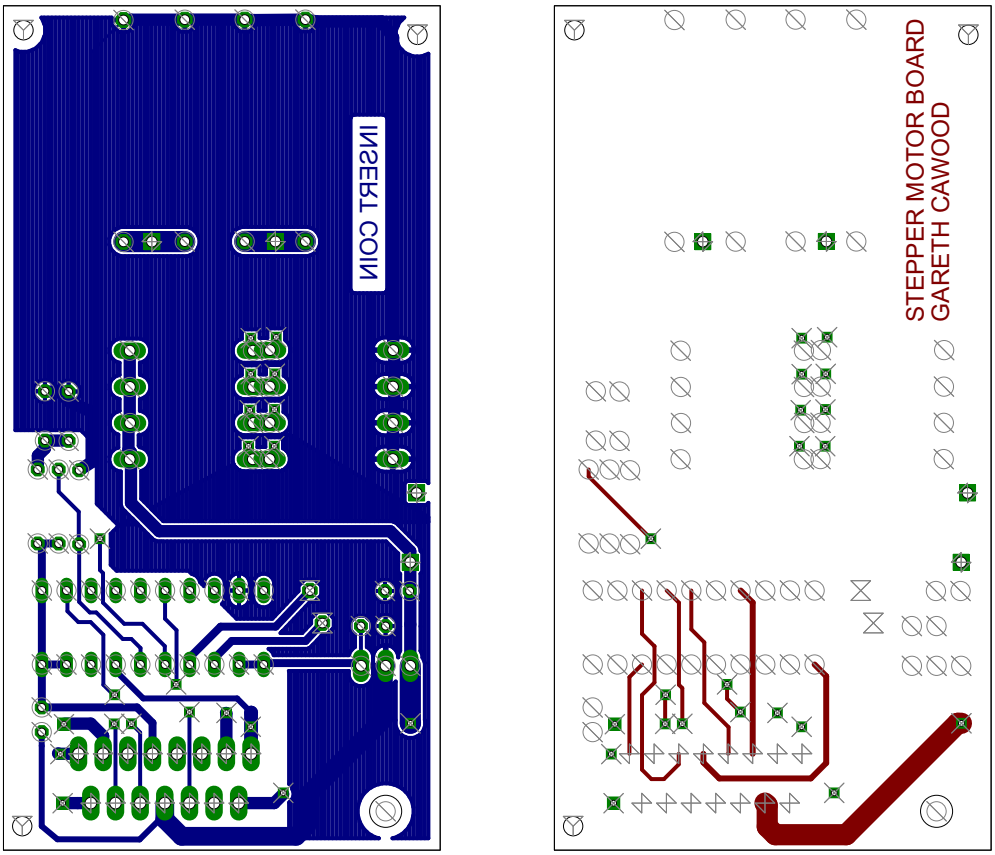
Now that all the development has been done, it would be fairly easy to reproduce the product. The cost would be over R20 000, but when comparing this to the R15 000 that some products cost, this could be seen as acceptable, especially with the advantage of continuous rotation.

Appendix D

Printed Circuit Boards

D.1 Motor Driver

The circuit board for the motor driver was designed as a double sided board in EagleCAD.



(a) Bottom (b) Top

Figure D.1: Design of the motor driver's PCB

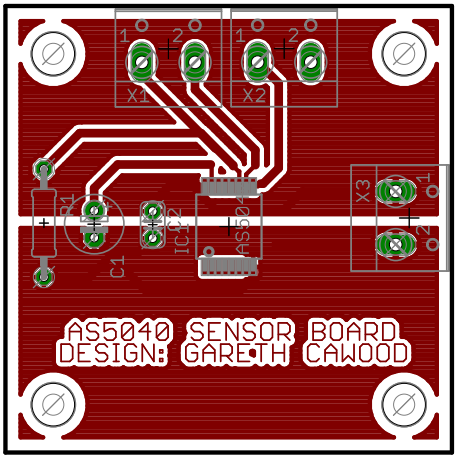


Figure D.2: Design for the AS5040's PCB

D.2 Sensor Board

The circuit board for the AS5040 was designed as a single sided board in EagleCAD.

Appendix E

Source Code

E.1 Microcontroller

```
// NOTE: This program makes use of the micros() function to get the system time,
    the value overflows after approximately 70min, and as such the system should be
     reset once an hour if possible, if not possible code should be modified to
    make use of the millis() function which only overflows after 50 days.

long timenow;
double time1 = 0; //time between steps for motor 1
double time2 = 0; //time between steps for motor 2

int angle1 = 0; // value from serial factor of 800
int angle2 = 0; // value from serial factor of 800
int pos1 = 0; //stepper motor 1 counts, factor of 800
int pos2 = 0; //stepper motor 2 counts, factor of 800

byte inByte1 = 0; //first byte read on serial
byte inByte2 = 0; //second byte read on serial
byte inByte3 = 0; //third byte read on serial
byte move1 = 0; //set to true if one of the motors shoudl be moving
byte move2 = 0; //set to true if one of the motors shoudl be moving

int goto1 = -1; //position motor should go to, factor of 800
int goto2 = -1; //position motor should go to, factor of 800
int goat1 = 0; // speed that motor must travel at
int goat2 = 0; // speed motor must travle at
int angle1now = 0; // value from sensor, factor of 1024
int angle2now = 0; // value from sensor, factor of 1024

long timeprev1 = 0;
long timeprev2 = 0;
long prev = 0;

void setup()
{
  Serial.begin(19200);

  //PAN
  pinMode(2, OUTPUT); //motor clock
  pinMode(3, OUTPUT); //motor CW/CCW
  pinMode(4, OUTPUT); //sensor SELECT
  pinMode(5, OUTPUT); //sensor CLOCK
  pinMode(6, INPUT); //sensor DATA

  //TILT
  pinMode(8, OUTPUT); //motor clock
  pinMode(9, OUTPUT); //motor CW/CCW
  pinMode(10, OUTPUT);//sensor SELECT
```

```
  pinMode(11, OUTPUT);//sensor CLOCK
  pinMode(12, INPUT);//sensor DATA

  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);
  digitalWrite(4, HIGH);
  digitalWrite(5, HIGH);
  digitalWrite(8, HIGH);
  digitalWrite(9, HIGH);
  digitalWrite(10, HIGH);
  digitalWrite(11, HIGH);
}

void loop()
{
  //read new commands
  if (Serial.available() > 0){
    inByte1 = Serial.read();
    if (inByte1 > 127){ //ie if the first bit is a 1
       // digitalWrite(13, HIGH);
      while (Serial.available() <= 0){
         }
         inByte2 = Serial.read();

         while (Serial.available() <= 0){
         }
      inByte3 = Serial.read();

      byte fhb1 = (inByte1 << 2);
      fhb1 = fhb1 >> 2;
      byte shb1 = (inByte2 >> 3);
      byte fhb2 = (inByte2 << 5);
      fhb2 = fhb2 >> 5;
      byte shb2 = inByte3;

         angle1 = (fhb1<<4) + shb1;
         angle2 = (fhb2<<7) + shb2;

         if ((inByte1 & B01000000) == B01000000){
            position(angle1, angle2);
         }
         else{
            speed(angle1, angle2);
         }
    }
  }

  //read angle & send angle from sensor
  if ((millis() - prev) > 50){
    angle1now = readPosition(5, 6, 4);
    Serial.print ((angle1now >> 7)|B11100000, BYTE);
    byte tempang = angle1now;
    Serial.print (tempang & B01111111, BYTE);

    angle2now = readPosition(11, 12, 10);
    Serial.print ((angle2now >> 7)|B10100000, BYTE);
    tempang = angle2now;
    Serial.print (tempang & B01111111, BYTE);

    Serial.print ((pos1 >> 7)|B11000000, BYTE);
    tempang = pos1;
    Serial.print (tempang & B01111111, BYTE);

    Serial.print ((pos2 >> 7)|B10000000, BYTE);
    tempang = pos2;
    Serial.print (tempang & B01111111, BYTE);
    prev = millis();
  }

 //check if position/speed correct
```

```
   if (goto1 >= 0){
      if (pos1 == angle1){
         move1 = 0;
      }
   }
  else{
      if (goat1 == 0){
         move1 = 0;
      }
   }

   if (goto2 >= 0){
      if (pos2 == angle2){
         move2 = 0;
      }
   }
  else{
      if (goat2 == 0){
         move2 = 0;
      }
   }

  //step motors
  timenow = micros();
  if (((timenow - timeprev1 - (time1)) > 0) && move1){
    if(digitalRead(8)){
      digitalWrite(8, LOW);

      if (digitalRead(9)){
        pos1--;
        if (pos1 == -1)
          pos1 = 799;
      }
      else{
        pos1++;
        if (pos1 == 800)
          pos1 = 0;
      }
      timeprev1 = timenow;
    }
    else
      digitalWrite(8, HIGH);
  }

  if (((timenow - timeprev2 - (time2)) > 0) && move2){
    if (digitalRead(2)){
      digitalWrite(2, LOW);

      if (digitalRead(3)){
         pos2--;
         if (pos2 == -1)
           pos2 = 799;
      }
      else{
        pos2++;
         if (pos2 == 801)
           pos2 = 1;
      }
      timeprev2 = timenow;
    }
    else
      digitalWrite(2, HIGH);
  }
}

int readPosition(byte clock, byte data, byte select)
{
  unsigned int position = 0;

  //shift in our data
```

```
  digitalWrite(select, LOW);
  delayMicroseconds(1);
  byte d1 = shiftIn(data, clock);
  byte d2 = shiftIn(data, clock);
  digitalWrite(select, HIGH);

  //get our position variable
  position = d1;
  position = position << 8;
   position |= d2;
  position = position >> 6;

  //check the offset compensation flag: 1 == started up
  if (!(d2 & B00100000))
    position = 2048;//000 01 000 0 0000000

  //check the cordic overflow flag: 1 = error
  if (d2 & B00010000)
    position = 4096;//000 10 000 0 0000000

  //check the linearity alarm: 1 = error
  if (d2 & B00001000)
    position = 4096;//000 10 000 0 0000000

  //check the magnet range: 11 = error
  if ((d2 & B00000110) == B00000110)
    position = 6144;//000 11 000 0 0000000

  return position;
}

//read in a byte of data from the digital input of the board.
byte shiftIn(byte data_pin, byte clock_pin)
{
  byte data = 0;

  for (int i=7; i>=0; i--){
    digitalWrite(clock_pin, LOW);
    delayMicroseconds(1);
    digitalWrite(clock_pin, HIGH);
    delayMicroseconds(1);

    byte bit = digitalRead(data_pin);
    data |= (bit << i);
  }
  return data;
}

void position (int ang1, int ang2)
{
   time1 = 11250;//equates to 40deg/s
   time2 = 11250;
   goto1 = ang1;
   goto2 = ang2;

  if (pos1 < ang1){
    if ((ang1 - pos1) < abs(ang1 - 800 - pos1)){
      digitalWrite(9, HIGH);
    }
    else{
      digitalWrite(9, LOW);
    }
  }
  else{
    if ((pos1-ang1) < abs(pos1 - ang1 - 800)){
      digitalWrite(9, LOW);
    }
    else{
      digitalWrite(9, HIGH);
    }
```

```
  }
  if (pos2 < ang2){
    if ((ang2 - pos2) < abs(ang2 - 800 - pos2)){
      digitalWrite(3, LOW);
    }
    else{
      digitalWrite(3, HIGH);
    }
  }
  else{
    if ((pos2-ang2) < abs(pos2 - ang2 - 800)){
      digitalWrite(3, HIGH);
    }
    else{
      digitalWrite(3, LOW);
    }
  }
  goat1 = 0;
   goat2 = 0;
  move1 = 1;
  move2 = 1;
}

void speed (int spd1, int spd2)//accept speed in deg/s
{
   goat1 = 1;
   goat2 = 1;
   goto1 = -1;
   goto2 = -1;

  if (spd1 > 0){
    time1 = round(1000000/(spd1/0.45));
    move1 = 1;
  }
  else{
    move1 = 0;
  }
  if (spd2 > 0){
    time2 = round(1000000/(spd2/0.45));
    move2 = 1;
  }
  else{
    move2 = 0;
  }
}
```

E.2 User Interface

```
import threading
from Tkinter import *
from datetime import datetime
import serial
import sys

def quit_cmd():
   print "QUIT NAOW"
   f.close()

def serialConnect(): #create serial connection
   global ser
   ser = serial.Serial(
      port='/dev/ttyACM0',
      baudrate=19200,
      parity=serial.PARITY_NONE,
      stopbits=serial.STOPBITS_ONE,
      bytesize=serial.EIGHTBITS
   )
   ser.open()
   ser.isOpen()
```

```
    print "Serial On"
    dt1 = datetime.now()
    dt2 = datetime.now()
    while ((dt2.second - dt1.second) < 1):
        dt2 = datetime.now()

    t1 = SerialUpdate()
    t1.start() #start new thread

#encodes commands to be sent. Bitwise defining turned out to be rather
#tricky in python, thus the long function
def string2bytes (angle, anglei, pos):
    ang =  bin(int(angle))
    lengt = len(bin(int(angle)))

    byt1 = "0"
    byt2 = "0"
    byt3 = "0"
    byt4 = "0"

    byt1 = '000000';

    if (int(angle) == 0):
        byt2 = '0000'

    if (int(angle) >0):
        byt2 =  '000' + str(ang[lengt-1:])

    if (int(angle) >1):
        byt2 =  '00' + str(ang[lengt-2:])

    if (int(angle) >3):
        byt2 =  '0' + str(ang[lengt-3:])

    if (int(angle) >7):
        byt2 =  str(ang[lengt-4:])

    if (int(angle) >15):
        byt1 =  '00000' + str(ang[lengt-5:lengt-4])

    if (int(angle) >31):
        byt1 = '0000' + str(ang[lengt-6:lengt-4])

    if ((int(angle) >63)):
        byt1 = '000' + str(ang[lengt-7:lengt-4])

    if ((int(angle) >127)):
        byt1 = '00' + str(ang[lengt-8:lengt-4])

    if ((int(angle) > 255)):
        byt1 = '0' + str(ang[lengt-9:lengt-4])

    if (int(angle) > 511):
        byt1 = str(ang[lengt-10:lengt-4])

    angle = anglei
    ang =  bin(int(angle))
    lengt = len(bin(int(angle)))

    byt3 = '000';
    if (int(angle) == 0):
        byt4 = '0000000'

    if (int(angle) >0):
        byt4 =  '000000' + str(ang[lengt-1:])

    if (int(angle) >1):
        byt4 =  '00000' + str(ang[lengt-2:])

    if (int(angle) >3):
```

```
        byt4 =  '0000' + str(ang[lengt-3:])

    if (int(angle) >7):
        byt4 =  '000' + str(ang[lengt-4:])

    if (int(angle) >15):
        byt4 = '00' + str(ang[lengt-5:])

    if (int(angle) >31):
        byt4 = '0' + str(ang[lengt-6:])

    if ((int(angle) >63)):
        byt4 = str(ang[lengt-7:])

    if ((int(angle) >127)):
        byt3 = '00' + str(ang[lengt-8:lengt-7])

    if ((int(angle) > 255)):
        byt3 = '0' + str(ang[lengt-9:lengt-7])

    if (int(angle) > 511):
        byt3 = str(ang[lengt-10:lengt-7])

    if (pos == 1):
        byt1 = '1'+'1'+byt1
    else:
        byt1 = '1'+'0'+byt1

    byt2 = '0' + byt2+byt3
    byt3 =  '0' + byt4

    return chr(int(byt1,2)), chr(int(byt2,2)), chr(int(byt3,2))

#send a speed command to the arduino
def serial_send_spd():
    b1, b2, b3 = string2bytes (tilt_speed_ach.get(), pan_speed_ach.get(), 0)

    ser.write(b1)
    ser.write(b2)
    ser.write(b3)

#send a position command to the arduino
def serial_send_pos():
    b1, b2, b3 = string2bytes (tilt_ach.get(), pan_ach.get(), 1)

    ser.write(b1)
    ser.write(b2)
    ser.write(b3)

class SerialUpdate(threading.Thread):
    def run(self):
        global f
        j = 0
        f = open('log1.csv', 'w')
        out = 2
        out2 = 2
        print "yes this is on"

        while True:
            out = long(ord(ser.read(1)))
            dt3 = datetime.now()
            if out != '':
                if out > 127:
                    #check if sensor 1 or 2
                    out2 = long(ord(ser.read(1)))
                    f.write(str(long(dt3.microsecond/1000))+',')
                    f.write(str(out) + ',')
                    f.write(str(out2) + ',')

                    if (out - 128) > 63:
```

```
                    #sensor 1
                    if (out - 128 - 64) > 31:
                        #update sensor 1 value
                        f.write('sensor 1,')
                        if (out - 128-64-32) > 7:
                            sens1.set("error")
                        else:
                            sens1.set(str(out2 + ((out-128-64-31)*(2**7)-128)))
                        f.write(sens1.get() + ',')
                    else:
                        #update counter 1 value
                        f.write('counter 1,')
                        count1.set(str(out2 + (out-128-64)*(2**7)))
                        f.write(count1.get() + ',')
                else:
                    #sensor 2
                    if (out - 128) > 31:
                        f.write('sensor 2,')
                        if (out - 128-32) > 7:
                            sens2.set("error")
                        else:
                            sens2.set(str(out2 + (out-128-31)*(2**7)))
                        f.write(sens2.get() + ',')
                    else:
                        f.write('counter 2,')
                        #update counter 2 value
                        count2.set(str(out2 + (out-128)*(2**7)))
                        f.write(count2.get() + ',')
                j+=1
                if (j < 4):
                    f.write(',')
                else:
                    f.write(pan_ach.get()+ ','+ tilt_ach.get() + ',' +pan_speed_ach.
                        get() + ',' + tilt_speed_ach.get()+'\n');
                    j=0

class GUI(threading.Thread):
    def run(self):
        global root
        root = Tk()

        def go_pos():
            print "Go to place"
            pan_ach.set(pan_new.get())
            pan_new.delete(0,END)
            tilt_ach.set(tilt_new.get())
            tilt_new.delete(0,END)
            serial_send_pos()

        def go_speed():
            print "Go @ speed"
            pan_speed_ach.set(pan_new_speed.get())
            pan_new_speed.delete(0,END)
            tilt_speed_ach.set(tilt_new_speed.get())
            tilt_new_speed.delete(0,END)
            serial_send_spd()

        def stop_pos():
            print "STOP"
            pan_speed_ach.set(0)
            tilt_speed_ach.set(0)
            serial_send_spd()

        def serialDConnect():
            ser.close()
            print "Serial Off"

        def serialRead():
            while ser.inWaiting() > 0:
                print ser.read(1)
```

```
#The setup for the GUI is extremely long and was thus removed from the report. The
    setup is fairly straightforward, it just results in a large amount of code


t2 = GUI()
t2.start()
```