Design and Implementation of a Six-Degree of Freedom Robotic Platform for Measuring the Forces of Flying Objects

Prepared By:

Richard Banda

Department of Electrical Engineering

University of Cape Town

Supervisor:

Professor J. Tapson

This thesis is submitted in partial fulfilment of the academic requirements for the Masters of Science Degree in Electrical Engineering, MSc (Eng) at the University of Cape Town

February 2011

DECLARATION

I, Richard Banda, hereby declare that this thesis is my own work. Where material generated from other sources is included, the parties and/or materials are indicated in the acknowledgements or are explicitly stated with references as appropriate.

This thesis has not been submitted nor will be submitted to any other Institution for any Degree or Examination.

Signature _____

Date _____

ACKNOWLEDGMENTS

I would like to thank Professor J. Tapson for putting forward this thesis topic and for being my supervisor for the duration of the project.

I would like to also thank Robyn Verrinder for being the co-supervisor of the project and for giving guidance and support throughout the duration of the project.

I would like to thank my sister Lydia Banda for continuously supporting and providing for me throughout the duration of the project.

I would like to also thank SDASM UCT for being my family at UCT and their efforts in providing encouragements to me in my days of despair.

I would like to also thank Glen Newins in the UCT Mechanical Engineering Department for his help in providing me with useful suggestions that helped in designing the platform and his help in the building of the platform.

I am very much grateful to my family for their constant support, encouragements and provisions throughout my studies at the University of Cape Town and I therefore would like to dedicate this project to them.

ABSTRACT

Bio-inspired robotics is a growing field at the University of Cape Town and also on a global scale. Although a large number of terrestrial robots have been built at the University, the university has recently initiated some projects for bio-inspired flying robots and six-degree of freedom platforms. Six degrees of freedom platforms are an important part in the development of robotic manipulators. These systems are widely used in various applications such as educational spheres, flight simulators and entertainment purposes. Six degree of freedom platforms are generally complicated and expensive.

The purpose of this project was to research, design and build a six-degree of freedom platform that measures the forces of flying animal robotics. The platform had to have an embedded force sensing mechanism and should be able to move in response to forces detected and measured. The platform had to measure the forces in six different orthogonal axes. The course of action and focus of the thesis was to research, design, build and control the robotic platform and it was an individual project.

A six-degree of freedom platform was designed and constructed. It was made from aluminium and teflon so as to reduce friction. It contained three rotational and three translational axes in structure. It had an in-built force sensing mechanism, which was a load-cell ring that used two strain-gauges, each in a Wheatstone bridge arrangement, with two standard resistors and a voltage-amplifying circuit built for the sensors. The platform also used servo motors as the actuators for the movements of the platform. A microcontroller or with analogue-to-digital converter and timer units was used. Software programs were written to control the platform.

Each rotational and each translational axis was thoroughly tested using known masses to provide the forces to be measured. In so doing, the force sensors were also calibrated for both tensile and compressive forces separately and individually. The addition of a multi-turn potentiometer in the Wheatstone bridge mechanism and voltage amplification from the sensors significantly reduced the zero error of the instrumentation of the platform.

Five of the six-degrees of freedom axes were implemented in the testing of the platform and they worked correctly. All the software modules and subsystems used for the project were also each tested separately and they worked successfully as their function could be confirmed and demonstrated. All the tests were written in separate files. While C programming language was used to write the control programs of the platform, Pro-Eng software was used to design the platform, Excel was used to tabulate the results and Matlab software was used to graph the results.

ABBREVIATIONS

DOF	Degrees of freedom
LCD	Liquid crystal display
LED	Light emitting diode

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE
INTRODUCTION

1.1 Background to Research Project

In recent years, there has been a great interest in the development of bio-inspired robotic structures like walking human robots, flying robotic birds, swimming robotic fish and other robots that mimic real biological behaviour of living creatures. **[1]** The University of Cape Town's Mechatronics, Control and Instrumentation research group has endeavoured to embark on building a robotic flying bat as part of a PhD project. In the testing of the project, the forces generated by the robotic bat are to be measured. When considering the flying movements and paths taken by a bat in flight, it will be noticed that bats have a complicated way of flying which cannot be modelled in 3D but 6D space. This is because, prey catching, obstacle avoidance, landing, for example, require different types of aerial manoeuvres. While in flight, bats can roll up to 180° and they have a rapid sequence of sideslips, alternating to the left and right and also upwards and downwards. This therefore increases their modelling space of flight. **[2]**

The University's Control Laboratory also has a helicopter that is used for practical teaching purposes in one of the major Electrical Engineering courses. The helicopter is used in a major project in the third year Control course where every student takes part in controlling the height position of the helicopter. This requires the students to balance out the forces acting on the helicopter in flight for them to stabilise and moderate its flight.

There is currently no platform available at the University of Cape Town which can accurately measure the forces generated by either of the above devices in 6D. Therefore, the Mechatronics, Instrumentation and Control research group developed a Masters project to solve the above problem. In solving the problem, a 6 DOF platform was to be designed and constructed.

Six degree of freedom (DOF) platforms have been made in the past. **[3]** These have had a widespread use and they have been used mainly in military affairs, aerospace, robo-cranes, vehicle testing, tourism and educational purposes. The most commonly used main 6 DOF platform is the Stewart platform created by Gough and Stewart from the UK in the period of

1954-1965, which is an example of a parallel manipulator discussed later in the document in section 2.5.4. Other 6 DOF platforms made are the Serial manipulators and Serial-Parallel manipulators also discussed later in the document.

Although a vast number of different kinds of 6 DOF platforms have been made, there has not been a 6 DOF platform that has been made that measures forces of flying objects in 6 different axes independently to one another, as far as we are aware.

1.2 Objectives of the Research Project

1.2.1 Purpose of the project

The major objective of the project is to **design a 6 Degrees of Freedom (6 DOF) robotic platform to measure the flying forces of any flying object that is able to hover and or glide in flight**, with focus on the flight of a robotic bat**.** The project primarily has to measure the forces of a flying robotic bat (for the PhD project) and secondarily hovering flights of man-made machines.

1.2.2 Specifications of the Project

The specifications of the project are as follows:

- The design should have 6 degrees of freedom
- The design must be able to move in response to force
- The design must measure forces in different axes independently to one another
- The design must be able to lock axes not needed
- The design must be low cost
- The design must be safe
- The design must be aesthetically pleasing
- The design should be simple to operate, with minimum human effort.

To achieve the above specifications the following procedures were to be done:

- Research on bat flight dynamics, the forces involved and the moments produced
- Research on man-made flying machines, focusing on the forces involved in flight
- Research on 6 DOF platforms focusing on the hardware designs
- Understand how forces are detected and measured in systems

- Design and build a 6 DOF robotic platform that measures the forces of flying objects
- Design and build force sensors and calibrate the sensors
- Write software which can control the robotic platform in C programming language
- Analyse the operation of the robotic platform
- Test the platform
- Analyse the results
- Draw conclusions from the results
- Make suitable recommendations based on the results.

1.3 Scope and Limitations of the Research Project

The project was to be carried out by an individual. The focus of this project was to build the 6 DOF robotic platform to measure forces that could successfully satisfy the need in the Electrical Engineering Department of the University of Cape Town described in the previous section.

The project was divided into three main areas of research:

i. Design, build and implement a 6 DOF platform to measure forces of flying objects in all axes
ii. Design, build and implement Instrumentation to measure the forces in all the robotic platform axes
iii. Develop software in C programming language to control the platform.

Though the project was for measuring the forces of a flying bat primarily, it was not tested on an actual flying robotic bat.

1.4 Thesis Development

The thesis starts with an introduction that discusses the purpose and the type of the project. It is then followed by the literature review that focuses on the forces of flying objects and different kinds of 6 degree of freedom platforms. Furthermore, a discussion of the hardware and software design and development for the project follows suit.

A detailed explanation of the methods used for the tests done on the project design follows the discussion section and a detailed description of the results follows that. A discussion of

the results and expectations is then outlined. Conclusions are then drawn based on the results of the experiments. Recommendations are then drawn and the improvements of future work are then made.

Table 1 below shows the work breakdown of the project.

Table 1: Work breakdown

YEAR	MONTH	WORK INTENDED	ACTUAL WORK DONE
2009	March	Literature Review	Literature Review
	June	1st design	1st design
	July		1st design evaluation
	August		2nd Design
	September	1st design testing	2nd design evaluation
	October	2nd Design (Final)	3rd design Final
2010	January	2nd Design (Final)	Instrumentation Design
	March	Final design testing	
	May	Thesis write up	
	July		Final Design Assembly
	August		Testing & Programming
	September	Submission	Thesis write up
	Feb		Submission

CHAPTER TWO

LITERATURE REVIEW

This section discusses the prior research studied in order to achieve the objectives of the research project. Firstly, it outlines the theory of bat flight, focusing on the axes involved in flying. It then discusses how human-made machines fly. In addition to that it outlines in detail how the forces are generated in bats and the moments involved in their flight paths. Lastly it discusses different six degree of freedom platforms that already exist.

2.1 Bat Flight

Figure 1 below shows a schematic of a bat in flight demonstrating all the six degrees of freedom, that is the three rotational degrees of freedom and the three translational degrees of freedom that must be controlled during flight.

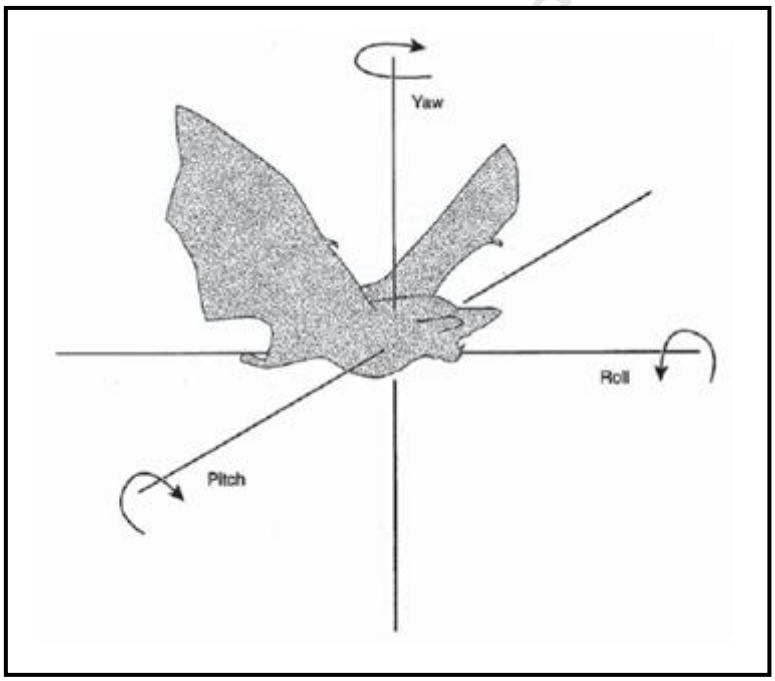


Figure 1: An illustration of a bat in flight showing the three rotational degrees of freedom (Adapted from Stockwell 2001)

2.1.1 Roll

Roll refers to the rotation about the median axis, which runs horizontally and longitudinally. That is, the rotation about the axis passing through the animal's centre of mass from the head to the tail in a horizontal plane. The control of roll moments can be achieved by the

differential twisting of the wings, so that angles of attack (that is the amount equal to the lift force which will be dealt with in the next section) become different on the two wings. A roll usually precedes a turn and can be used to change direction of flight in a bat flight. **[4]**

2.1.2 Pitch

Pitch is the rotation about the transverse axis, running horizontally and transversely through the centre of gravity in an animal flying horizontally. In other words it is the rotation about the axis that passes mediolaterally in a horizontal plane. It is closely related to speed control in the animal's flight and is best achieved by the tail movements. Downward movement of the tail produces upward lift, and a nose-down pitching moment results in faster gliding speed (that is along the forward direction). An upward movement of the tail in a bat produces a nose-up moment and slower speed. **[4]**

2.1.3 Yaw

Yaw is the rotation about the vertical axis. Yaw is controlled by the tail or by twisting and flexing of the wings to give different drag coefficients on the two sides. Generally bat flight is achieved by the vortex theory which is described in figure 2 below. The theory helps in the understanding of the controls that the animal uses in controlling the lift and forward kinematics of the flight. **[4]**

Figure 2: Showing the vortex rings produced by a bat in flight to enable the animal to hover and glide [5]

2.1.4 Vortex theory

According to the vortex theory, forces are created in a complicated way. The air current path is modelled by a chain of coaxial small-cored circular vortex rings stacked one upon another; each member of the chain is generated by a single wing-stroke. The air pressure gradients point around the wing to give rise to air currents that flow upward from the front edge of the wing towards the rear and around the rear edge of the wing back towards the front. The circular air current dissipates at the wingtips and the tail in the form of a vortex. In this the camber of a wing, in addition to increasing the velocity of air flow over it, also deflects the air downwards and induces rotational flow around the wing, and behind it. If Newton's third law is to be obeyed (which states that, action and reaction are equal and opposite), then the forces balance. The rotational movement of these vortices provide the force (aerodynamic force). This is because when the animal is in flight, it will be positioned at the upper, leading front of the vortices where the direction of air movement carriers the animal's body forward and upward. **[6]**

2.2 Flying Machines

Human made flying machines mostly depend on their wings for any form of flight. The shape of the aircraft wing is very important for lift, and is designed with great care. The wings are generally curved as shown in the diagram below and sometimes have curved upper surfaces and flatter lower surfaces. This is normally seen in modern day helicopter blade. The flight of these machines is based on a scientific principle called the Bernoulli's principle which is explained below.

2.2.1 Bernoulli's principle

The principle states that an increase in the velocity of a fluid (such as air currents) is accompanied by a decrease of pressure within that fluid. This was demonstrated by a Swiss scientist, Daniel Bernoulli (1700-1782). This explains in part why a wing lifts an airplane.

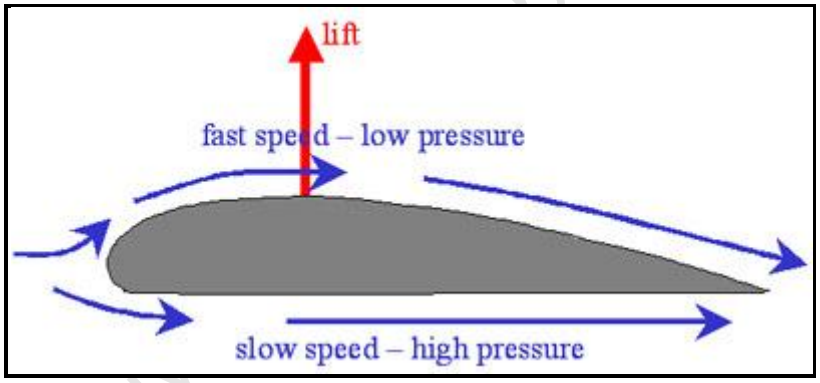


Figure 3: Showing the lift of a wing produced by the inverse proportionality of wind speed and pressure.
Adapted from Diracdelta Science and Engineering **[7]**

As the aircraft flies, the air that is moving more quickly above the curved wing starts to put less pressure on the wing while it adjusts to its new stream. Meanwhile, the air that is moving at a consistent speed below the wing maintains its rate of pressure. This pressure differential produces lift. The higher air pressure pushes the wing upward into the space where the air pressure is lower. The curve of a wing, therefore, affects how much lift force is generated when an aircraft moves through a stream of air. **[8]** A pilot can increase the lift for both cambered and symmetric wings by increasing the angle of attack of the wing with respect to the free-stream flow. That is the reason why an airplane rotates slightly at takeoff; the pilot is

increasing the angle of attack to generate more lift. If the angle of attack is doubled, the lift doubles, within the small appropriate range for flight. **[9]**

2.2.2 Airflow deflection

Another way of explaining how forces are generated in flying machines is airflow deflection. Airflow deflection makes use of Newton's Third Law of Motion. The aircraft wing deflects the air going over the upper surface downward as it leaves the trailing edge of the wing. According to Newton's Third Law (for every action there is an equal, but opposite reaction), if the wing deflects the air down, the resulting opposite reaction is an upward push. Deflection is an important source of lift. Planes with flat wings, rather than cambered, or curved wings must tilt their wings to get deflection.

In summary, the lift on an airplane is primarily generated by the wings. The amount of lift generated by the body, or fuselage, of the aircraft is very small. So, the lift force generated by the wings is the basic lift force on the airplane. Its direction is upward, opposite to the weight. If the lift is greater than the weight, the plane will fly. After the plane has climbed to the cruising altitude, it will level off (decrease the angle of attack to zero), and the lift force will be equal to or very slightly larger than the weight. If the lift force is smaller than the weight, the plane will lose altitude and return to earth. Thus man-made machines can be controlled for flight. **[9]**

2.3 Flying Forces

When an animal has to fly, it has to produce a force that will overcome its weight. This force is called the lift force. According to a study done by Norberg (1970), **[10]** a possible mechanism of the lift is generated in the 'tip-reversal' upstroke.

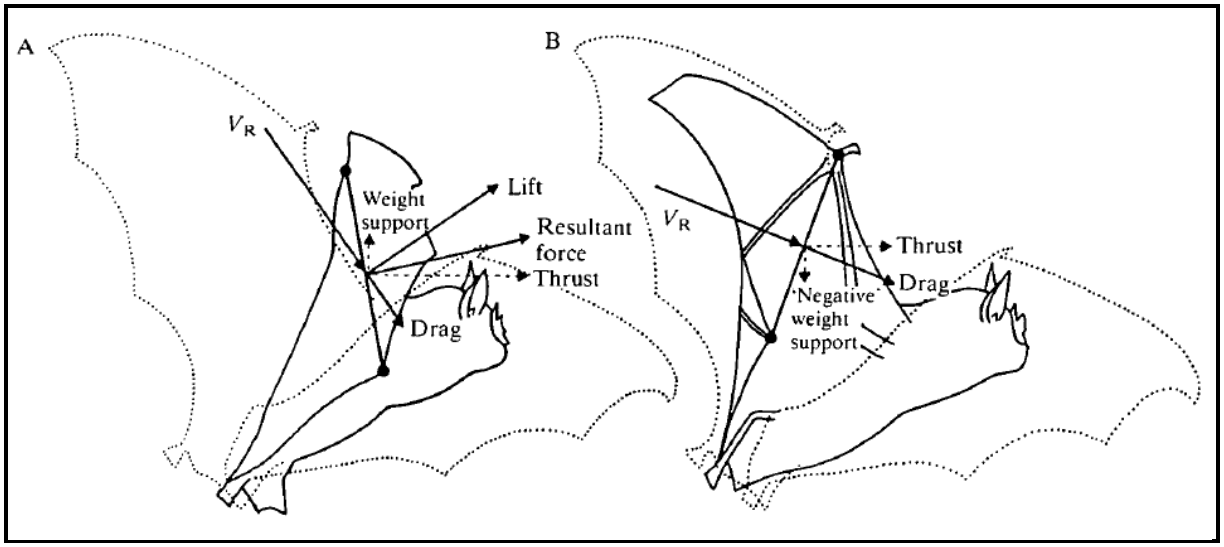


Figure 4: Showing the lift generation mechanism during the 'tip-reversal' up-stroke. (Taken from Norberg 1970) [10]

In part (A) in figure 4 above, the wing is moved backwards and upwards and the resultant airflow strikes the dorsal surfaces; and in this manner generates lift, thus providing weight support. The lift is directed forward and therefore acts as thrust. In part (B) during the flick, the relative airflow strikes the wing's dorsal surface at high incidence angles. Lift is not generated, but the resultant drag acts as thrust. V_R is the resultant velocity of the wing. **[10]**

Generally, when an animal is in vertical flight and during a wingbeat it experiences six forces:

- Weight, Mg (where M is body mass and g is the acceleration due to gravity);
- Induced drag, D_{ind},
- Parasite drag, D_{par};
- Profile drag, D_{pro};
- Inertial forces, F_b;
- Lift, L.

During the downstroke of an animal in flight, the wings move forward and downwards generating L, F_b, D_{ind} and D_{pro}, the consequent upward movement of the body generating D_{par}. Of these forces L, D_{ind}, and F_b will tend to accelerate the animal upwards, while Mg, D_{pro} and D_{par} will retard its upward movement. In the same way, F_b, D_{ind} and D_{pro} will tend to accelerate the animal backwards, although L should work against this. During the upstroke,

F_b, D_{ind} and D_{pro} will tend to accelerate the animal downwards and forwards. Lift is generated during the upstroke is directed forward and this is visually seen in the diagram below. **[10]**

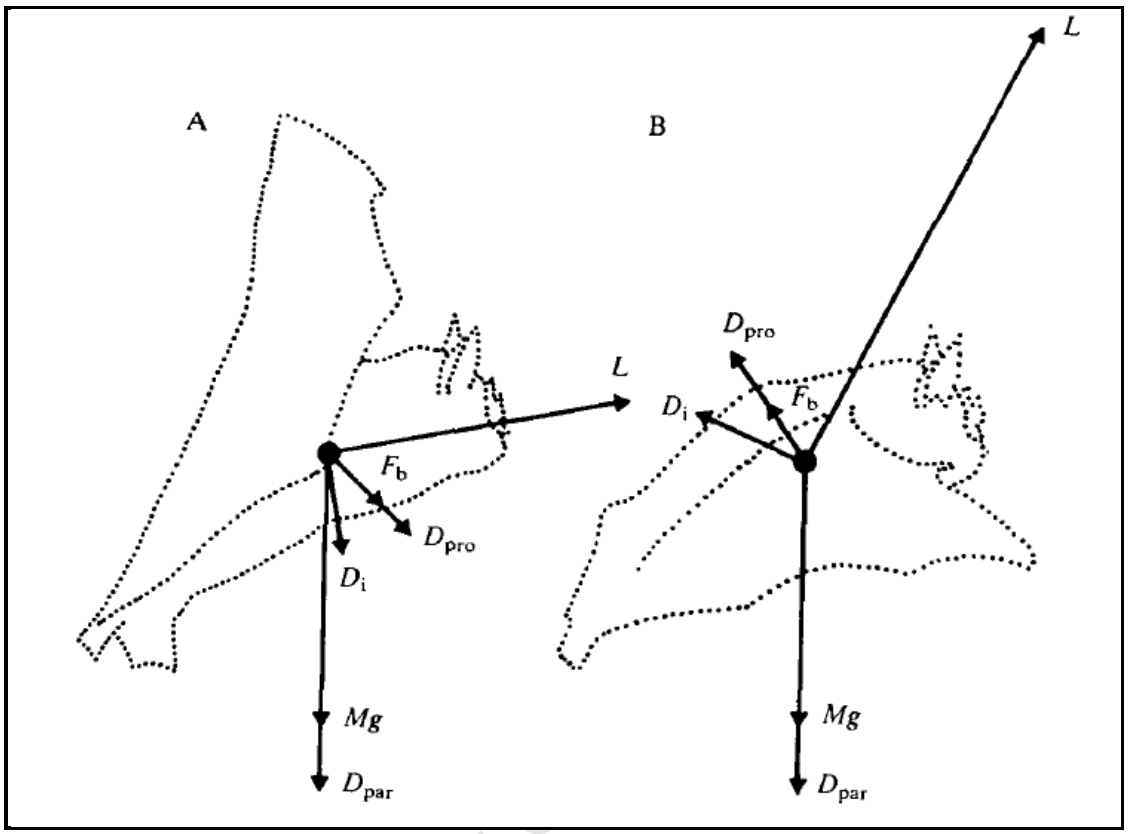


Figure 5: Showing the Mean forces acting on a bat flying vertically. (A) Upstroke; (B) Downstroke (Taken from Norberg (1970)) [10]

The instantaneous vertical, $F_{vert}(t)$ and horizontal, $F_{hor}(t)$, forces acting on the bat are, therefore:

$$F_{vert}(t) = L_{vert}(t) + F_{b,vert}(t) + D_{pro,vert}(t) + D_{ind,vert}(t) - D_{par,vert}(t) - Mg$$

And

$$F_{hor}(t) = L_{hor}(t) - F_{b.hor}(t) - D_{ind,hor}(t) - D_{pro,hor}(t) - D_{par,hor}(t)$$

During the downward stroke, and:

$$F_{vert}(t) = L_{vert}(t) - F_{b,vert}(t) - D_{ind,vert}(t) - D_{pro,vert}(t) - D_{par,vert}(t) - Mg$$

And

$$F_{hor}(t) = L_{hor}(t) - F_{b,hor}(t) - D_{ind,hor}(t) - D_{pro,hor}(t) - D_{par,hor}(t)$$

during the upstroke. Where $L_{vert}(t)$, $L_{hor}(t)$, $F_{b,vert}(t)$, $F_{b,hor}(t)$, $D_{ind,vert}(t)$, $D_{ind,hor}(t)$, $D_{Pro,vert}(t)$, $D_{pro,hor}(t)$, $D_{par,vert}(t)$ and $D_{par,hor}(t)$ are, respectively, the instantaneous vertical and horizontal components of L, F_b, D_{ind}, D_{pro} and D_{par}.

Aldridge (1987), also outlines how the instantaneous inertial force F_b can be calculated, as follows, The angular velocity of a wing is:

$$\omega = d\varphi/dt$$

Where φ is the positional angle of an animal in flight, this is shown in the figure below. The wing angular acceleration is, therefore given by:

$$d\omega = d^2\,\varphi/\,dt^2$$

And the acceleration of the wing's centre of mass is:

$$a_w\,(t) = rd\omega$$

where r is the distance between the centres of mass of the body and wing as in the figure 6 shown below. **[11]**

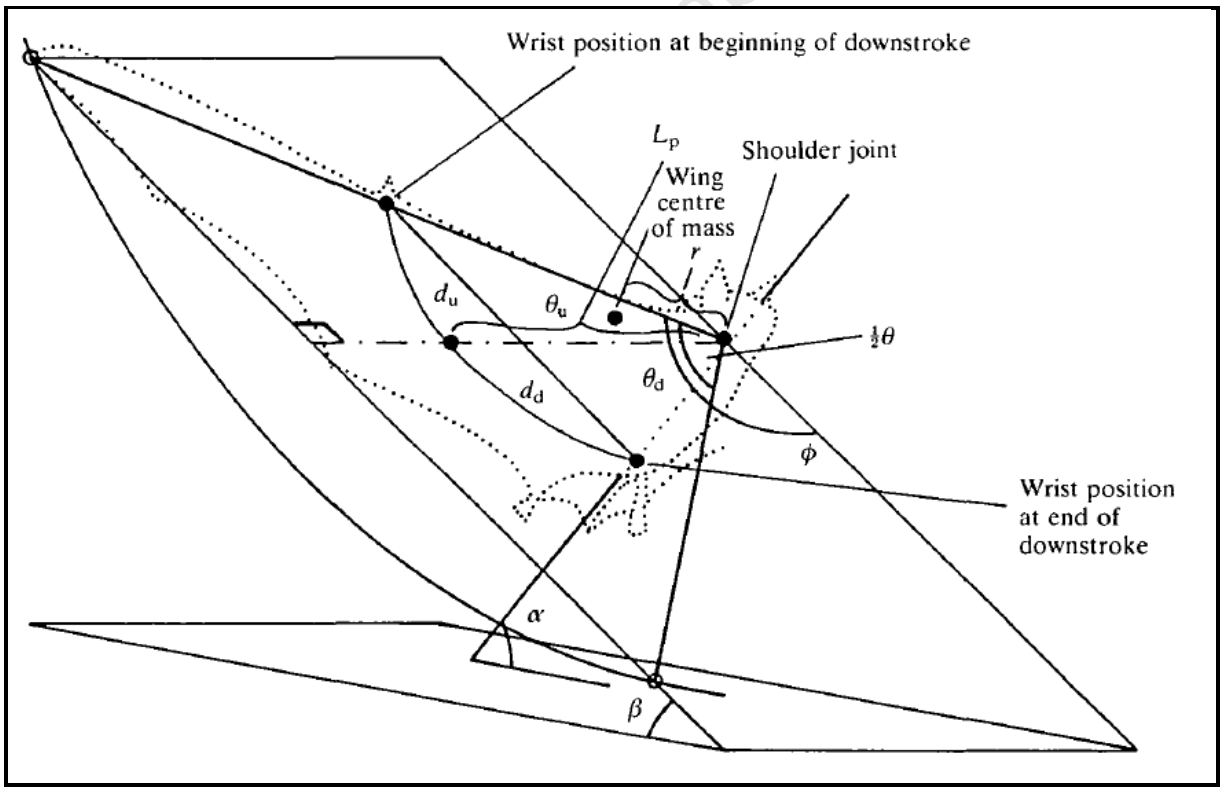


Figure 6: Diagram showing the various terms used in the text. (Taken from Aldridge (1987))

Note: β is the strokeplane angle, $\theta/2$ is the wing's angular excursion and θ_U and θ_d are the wing's positive and negative elevations, respectively. Φ is the positional angle and α is body angle. d_u and d_d are the distances, within the strokeplane, between the wrist and the longitudinal axis of the body at the beginning and end of the downstroke, respectively. L_p is the distance between a point midway between the two humeral joints and the wrist when the wing is fully extended and r is the distance between the centres of mass of the body and wing.

The total inertial force acting on the body due to the acceleration of one wing is:

$$F_w(t) = M_w a_w(t)$$

where M_w is the mass of the wing. In the absence of actual measurements of wing mass the vertical component of $F_w(t)$ is:

$$F_{w,vert}(t) = F_w(t) \sin\beta$$

The total vertical force, due to wing oscillation, experienced by the bat is therefore:

$$F_{b,vert}(t) = 2[F_w(t) \sin\beta]$$

and, finally, the vertical acceleration experienced by the animal due to wing acceleration is:

$$a_{b,vert}(t) = 2[F_w(t) \sin\beta]/M$$

Similarly, the instantaneous horizontal acceleration, $a_{b,hor}(t)$ experienced by the animal due to wing acceleration is:

$$a_{b,hor}(t) = 2[F_w(t) \cos\beta]/M$$

[12]

2.3.1 Wing Action in Flight

In a paper by Ellington (1984), the author states that when an animal is in flight, the acceleration and deceleration of the wings during a wingbeat will set the air surrounding the wings into motion. As a result, there is an apparent increase in wing mass; wing virtual mass, or added mass, m_v. **[13]** Theoretically, this mass is equal to the mass of air in an imaginary cylinder around the wing with the wing chord as its diameter. The virtual mass of a wing element at a distance l from the humeral joint can be calculated as:

$$m_v' = \rho\pi c^2/4$$

where c is wing chord length and p is air density. The total virtual mass of a wing is:

$$m_v = (\rho\pi/4)\int_{l=0}^{l=l_w} c^2 \,.\, dl$$

where l_w is wing length. Ellington (1984).

In calculating the inertial forces acting on the bat, it is assumed that the wings are either accelerating or decelerating throughout the wingbeat and thus includes m_v as a constant.
$D_{par}(t)$ is the sum of the instantaneous pressure and friction drags of the body and will tend to oppose the animal's vertical movement. $D_{par}(t)$ can be calculated as:

$$D_{par} = {}^1\!/_2\, \rho V^2 S_b C_{D,par}$$

where S_b is the projected area of the body perpendicular to the airflow and $C_{D,par}$ is the parasite drag coefficient.

$S_b C_{D,par}$ can be replaced by A_e a constant, which is the area of a flat plate with a parasite drag coefficient equal to 1, which gives the same drag as the body. **[14]**

So the parasite (body drag) can be written as:

$$D_{par} = {}^1\!/_2\, \rho V^2 S_b C_{D,par} = 1/2\rho V^2 A_e$$

D_{pro} is the sum of the instantaneous pressure and friction drags of the wings and can be calculated as:

$$D_{pro} = {}^1\!/_2\, \rho V^2 S C_{D,pro}$$

where V is the resultant velocity of the wing, S is the wing area of the animal and $C_{D,pro}$ is the profile drag coefficient.

2.3.2 Effects of Flying Speed on Forces

In a study on vertebrate flight, Norberg also states that when we consider an animal wing of span b moving horizontally with a speed *V*. **[15]** It is assumed that the wing affects the air in a vertical circle of diameter *b* around the aerofoil and that all the air passing this circle is given a downward *induced velocity w*. The mass of air passing through this circle of area $\pi b^2/4$

(called wing disk area S_d) is, in unit time, $\rho V\pi b^2/4$. The air is given momentum at a rate $\rho Vw\pi b^2/4$, which must equal the lift L, so that $w = 4L/\rho V\pi b^2 = L/\rho VS_d$. The rate at which the air is given kinetic energy is *(1/2) x (mass) x (velocity^2)* and is $\rho Vw^2\pi b^2/8 = 2L^2/\rho V\pi b^2$. Some of the work done against drag in propelling the wing must be used to give the same kinetic energy to the air. The associated drag is the induced drag, D_{ind}, and because the rate of working *(power) = (drag) x (velocity)* we obtain:

$$D_{ind} = \frac{2L^2}{\rho V^2 \pi b^2} = \frac{L^2}{2\rho V^2 S_d}$$

The average, effective, induced drag, D'_{ind}, is usually written as

$$D'_{ind} = \frac{2k(Mg)^2}{\rho V^2 \pi b^2}$$

Where k is the induced drag factor, Mg is the weight of the animal (where M is body mass and g acceleration owing to gravity), which equals the vertical lift in horizontal and hovering flight. Therefore, since the areas used for profile and parasite drag differ, the velocity of air meeting the wings and body in a flapping animal like a bat is different, because the resultant velocity for the wings includes flapping components. The resultant velocities of the wings and body have different directions and the drag components must be treated separately. **[15]** The wing profile drag can then also be written as:

$$D_{pro} = (\frac{1}{2})\rho V^2 S C_{D,pro}$$

Where S is the wing area of the animal and $C_{D,pro}$ is the profile drag coefficient and the parasite (body) drag as:

$$D_{par} = \left(\frac{1}{2}\right)\rho V^2 S_b C_{D,par} = (\frac{1}{2})\rho V^2 A_e$$

S_b is the frontal projected area of body and A_e is the area of a flat plate with the parasite drag coefficient $C_{D,par} = 1$, which gives the same drag as the body. A_e is conventionally named the equivalent flat plate area.

The total aerodynamic drag of a flying animal thus becomes:

$$D = D_{ind} + D_{pro} + D_{par}$$

$$D = \frac{2k(Mg)^2}{\rho V^2 \pi b^2} + \left(\frac{1}{2}\right)\rho V^2 (SC_{D,pro} + A_e).$$

2.4 Moments of Flight

In powered flight, flying bats do work with their flight muscles to move the wings in order to generate lift and thrust. The rate at which this work is done is the mechanical power required to fly. The mechanical power is the sum of the *aerodynamic power*, which is the power needed to obtain sufficient aerodynamic force, and the *inertial power*, which is the power needed to oscillate the wings.

The aerodynamic power consists of three components:

- *Induced power* P_{ind} - the rate of work required to generate a vortex wake whose reaction generates lift and thrust,
- *Wing profile power* P_{pro} - the work needed against form (pressure) and friction drag of the wings;
- *Parasite power* P_{par} – the work needed against form and friction drag of the body.

2.4.1 Inertial power, P_{iner}

In the study on vertebrate flight, **[15]** Norberg defines inertial power as the work needed to accelerate the wings at each stroke (that is to oscillate the wings). When we multiply the aerodynamic drag components (described above) by the speed V and adding inertial power P_{iner}, the total flight power then becomes:

$$P = 2kM\frac{g^2}{\rho V \pi b^2} + \left(\frac{1}{2}\right)\rho V^3 \left(SC_{D,pro} + A_e\right) + P_{iner}$$

The power components are associated with the muscles' mechanical efficiency η, the ratio of mechanical power output, P (which is the flight power or rate at which mechanical work is done), to metabolic power input, P_{ind} (the rate at which chemical, metabolic energy is

consumed). In addition to these components is the cost of inertial body functions, or resting metabolism, P_b, so the power input is: **[15]**

$$P_i = \frac{P_{ind} + P_{pro} + P_{par} + P_{iner}}{\eta} + P_b$$

According to a study by Pennycuick in 1975, the inertial power is also given by:

$$P_{iner} = I\omega/T$$

where I is the moment of inertia of the wings and ω the angular velocity of the wings. The moment of inertia depends on the distribution of the mass along the wing, and is:

$$I = \int_{r=0}^{r=I_w} m_w' r^2 \, dr$$

Where I_w is the wing length and m_w the mass of a wing-element at distance r from the fulcrum (shoulder joint) and he estimated the inertial power, to be given by:

$$P_{iner} = 16 J_w \pi^2 f^3 \varphi^2$$

where J_w is the moment of inertia of one wing about the shoulder joint, f is wingbeat frequency and φ is wingbeat amplitude. **[16]** The inertial power is lower in medium and fast flight than in hovering and slow flight, since both f and φ decrease with increased flight speed. **[17]**

2.4.2 Induced Power

The induced power is the main power in flight. In hovering flight P_{ind} is directly proportional to $Mg^{3/2}/b$, and in forward flight P_{ind} is directly proportional to Mg^2/b^2V and so to minimize induced power the weight should be low and the wingspan long.

2.4.3 Profile Power

The profile power, increases with wing area (with length and width) and with increasing speed; P_{pro} is directly proportional to b^3S/T^3 in hovering flight **[18]** and P_{pro} is directly proportional to SV^3 in forward flight. Where T is wingstroke period and about proportional to b hence P_{pro} is directly proportional to S in hovering.

2.4.4 Effects of Animal Structure on Flight Dynamics

In a study by Rayner (1987), the ability to make tight manoeuvres in animal flight can be separated into two different components, the radius of the turn, which is a measure of manoeuvrability, and the roll acceleration, which is a measure of agility. **[19]**

Andersson and Norberg in their study in 1981 also stated that the roll acceleration is inversely proportional to the moment of inertia about the roll axis. **[20]** The moment of inertia about the roll axis originating from the body has been assumed to equal in magnitude to that originating from the wings.

These authors outlined that when estimating a wing's moment of inertia about the shoulder joint, using strip analysis, the moments of inertia about the long (chord-wise) axis of each individual strip should be taken into account (see figure 7 below). The moment of inertia about an axis parallel to a principal axis through the centre of gravity of a rectangular plate (wing strip) is obtained by means of a relationship called the ***parallel-axis theorem*** (or Steiner's proposition), which is given in most mechanics handbooks. The theorem states that the moment of inertia about the new axis is the moment of inertia about the original axis (through the centre of gravity) plus the mass of the object (strip) times the square of the distance between the two axes. If each strip is assumed to be a thin rectangle, the moment of inertia of one wing about the shoulder joint, J_w, is given by: **[21]**

$$J_w = \sum_{n=1}^{k} \left(m_n \ \ r_n^2 + \frac{m_n \, w_n^2}{12} \right)$$

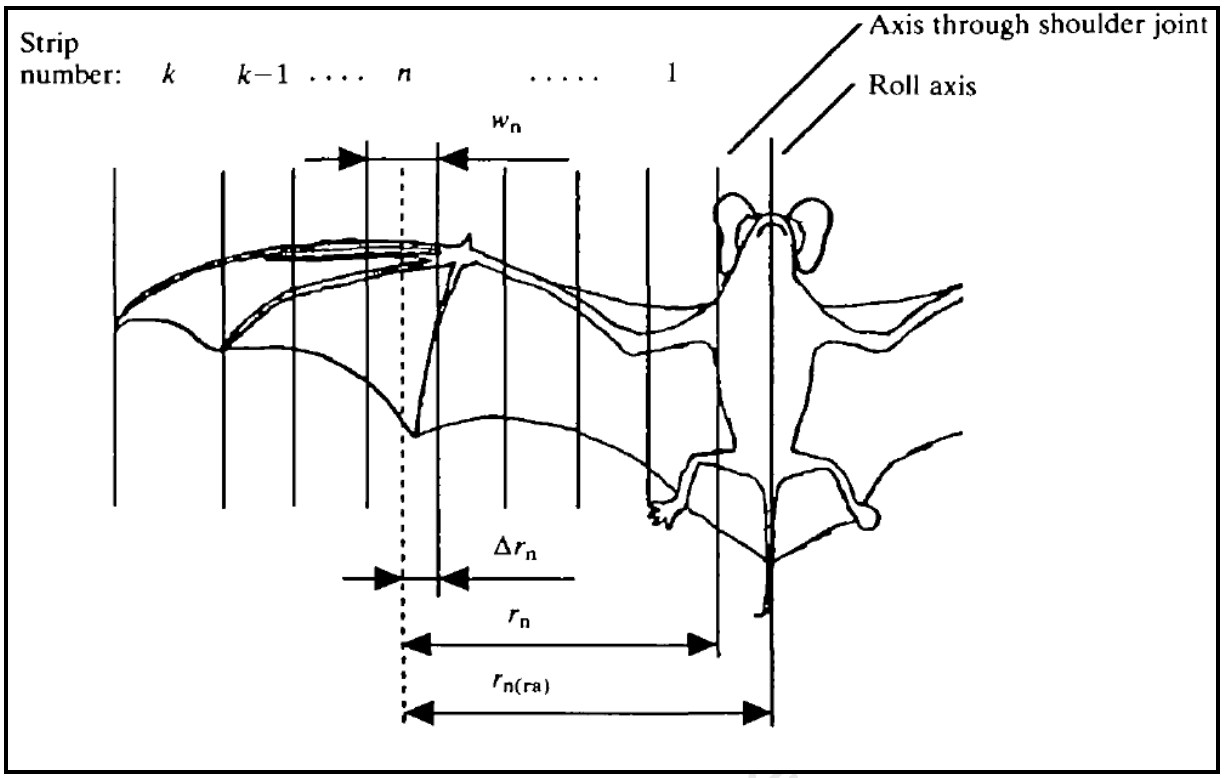


Figure 7: Showing the method for estimating the moment of inertia of a bat's wing about the shoulder joint and about the roll axis. (Adapted from Norberg and Tholleson)

for the *k* strips of the wing as shown above, where m_n is the mass, w_n is the width of the n^{th} strip and r_n is the distance from the shoulder joint to the centre of gravity of the nth strip, as indicated in the figure above. The moment of inertia about the centre of gravity for each strip (the second term of the equation above) will be negligibly small if the rectangle (strip) is sufficiently narrow. Ignoring this term, the wing's moment of inertia about the shoulder joint could then be expressed as: **[22]**

$$J_w = \sum_{n=1}^{k} m_n r_n^2$$

2.4.5 Effects of Animal Body on Flight Dynamics

In a paper on moments of inertia on bat wings and body, Tholleson and Norberg (1991) outline a procedure that was used to estimate the moment of inertia originating from the body (including the head). The head and body were considered to be ellipsoids mounted on one another with major axes coinciding as shown in the figure below. The head and body were also assumed to have equal density and uniform mass distribution, so the calculations he used

were based on volume distribution between head and trunk rather than on mass distribution. **[23]**

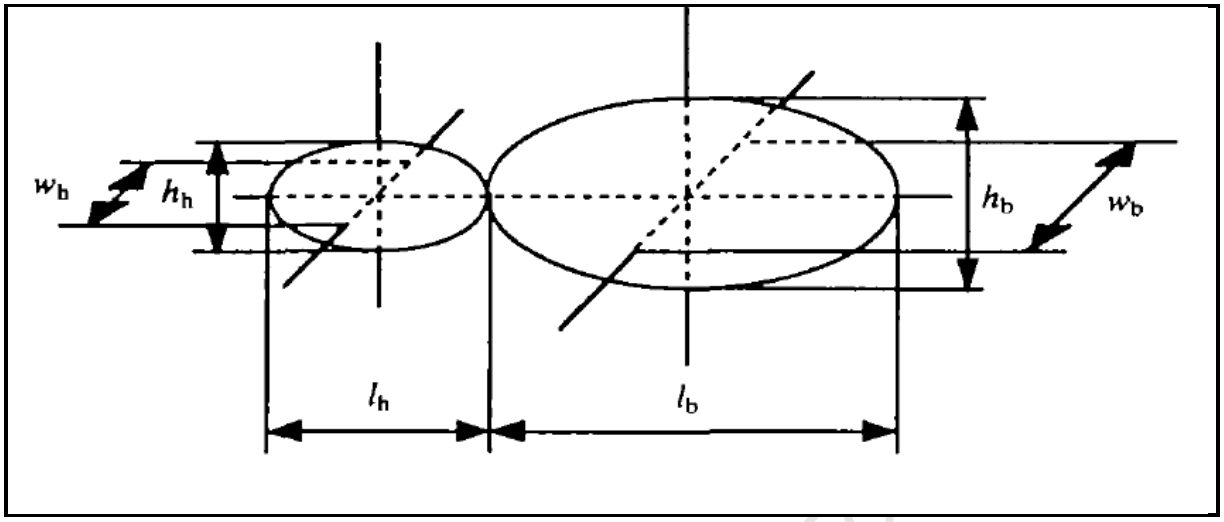


Figure 8: Showing the measurements used to estimate the combined moments of inertia of the body and the head about the roll axis. (Adapted from Norberg and Tholleson)

The volume V_e of an ellipsoid is:

$$V_e = \frac{4\pi}{3}\frac{l}{2}\frac{w}{2}\frac{h}{2} = \frac{\pi lwl}{6}$$

where l is the length, w is the width and h is the height of the ellipsoid, and the moment of inertia of the ellipsoid is:

$$J_e = \frac{m}{5}\left(\frac{h^2}{4} + \frac{w^2}{4}\right) = \frac{m}{20}(h^2 + w^2)$$

Alexander and Vernon, (1975). **[24]** The mass of the head was estimated as the combined mass of the head, m_h, and trunk, m_t, multiplied by the volume proportion that of the head which makes up of the combined volume of the head and trunk, $V_h/(V_h+V_t)$, as:

$$m_h = \frac{m_{t+h}\,V_h}{V_h + V_t}$$

Using the 3 equations above, the combined moment of inertia of the body and head about the roll axis then becomes:

$$J_{br} = \frac{m_{t+h}}{20}\left[\frac{l_h w_h h_h (w_h^2 + h_h^2) + l_t w_t h_t (w_t^2 + h_t^2)}{l_h w_h h_h + l_t w_t h_t}\right]$$

where indices *t* and *h* refer to trunk and head, respectively, and m_{t+h} being the mass of the trunk and head together.

Finally, the combined moment of inertia of wings and body plus head about the roll axis can be calculated as:

$$J_{roll} = 2J_{wr} + J_{br}$$

2.5 Six Degree of Freedom Platforms (6 DOF)

Six DOF robotic platforms have been made in the past and a fundamental issue in robotics is the design of these manipulators. There are a wide range of 6 DOF platforms that have been made and some of them are outlined briefly in this section. While there are an increasing variety of applications, some requirements are mostly the same across all platforms: **[25]**

- The mechanical system should be easy to manufacture at a low cost. As electronic components are becoming cheaper and cheaper, there is a general trend towards systems which are mechanically simple but computationally demanding.
- The manipulator should consume little energy.
- The end-effector should achieve high velocities and/or high accelerations. **[26]**

The project will focus on the following:

- Serial manipulators – These are robotic mechanical structures made up of a serial chain of rigid links, connected by (mostly revolute) joints where the motion of one link affects the others. **[27]**
- Parallel manipulators – These are closed-loop mechanisms in which the mobile platform (end-effector) is connected (linked) to the base by at least two serial kinematic chains. **[28]**
- Tendon based manipulator – These are mostly parallel manipulators that use cables (tendons) instead of rigid links to connect the end-effector to the base (immovable part of the system). **[29]**

2.5.1 Serial manipulators

These platforms used to be the most common type of robotic manipulators. They are not very energy efficient because each actuated link also has to carry all the subsequent links and their actuators. It is easy to achieve high accelerations and velocities because the end-effector moves generally faster than the actuated links. **[30]** Their main advantage is their large workspace with respect to their own volume and occupied floor space.

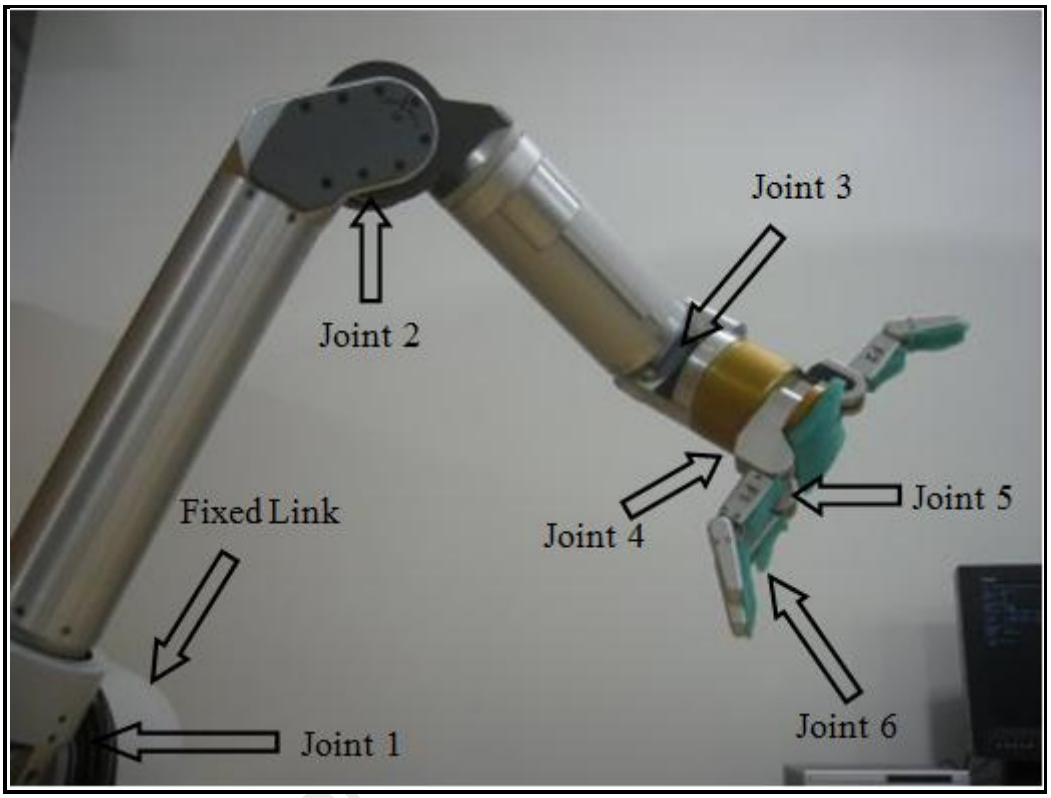


Figure 9: Showing a robotic arm with joints connected in a serial link to form a serial manipulator whereby one movement of a joint affects other joints [31]

Although these manipulators possess a lot of advantages, they also have disadvantages in that they have low stiffness inherent to an open kinematic structure. They also have errors accumulated and amplified from link to link in their performance. Another problem with these manipulators is that they have to carry and move the large weight of most of the actuators and thus they have a relatively low effective load that they can manipulate. **[32]**

2.5.2 Tendon-driven serial manipulators

These platforms are more energy efficient because the actuators are fixed and do not need to be moved. However, in three-dimensional applications it is very difficult to guide the tendons around joints. Their mechanical system is not that simple because the pulley carrying the load must be movable at least in one or two directions of translation. They are generally used in

practice mainly for planar systems like cranes. Figure 10 below shows the diagram of a Tendon-driven serial manipulator. **[33]**

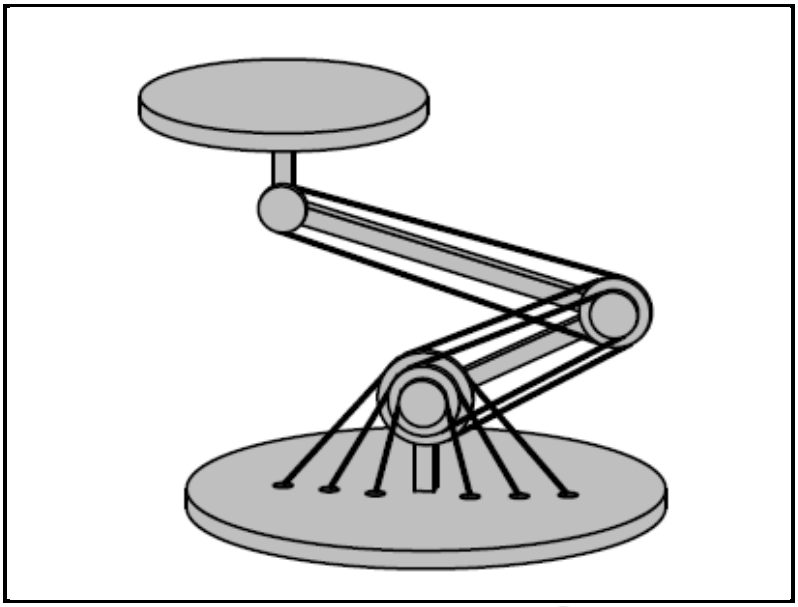


Figure 10: Showing a Tendon-driven serial system [Richard Verhoeven (2004)]

2.5.3 Tendon-based parallel robots

These platforms offer a new approach in the field of robotics. They are very energy efficient because the actuators are fixed and the payload is subdivided between actuators. Therefore, they are appropriate to handle heavy loads and can achieve high accelerations.

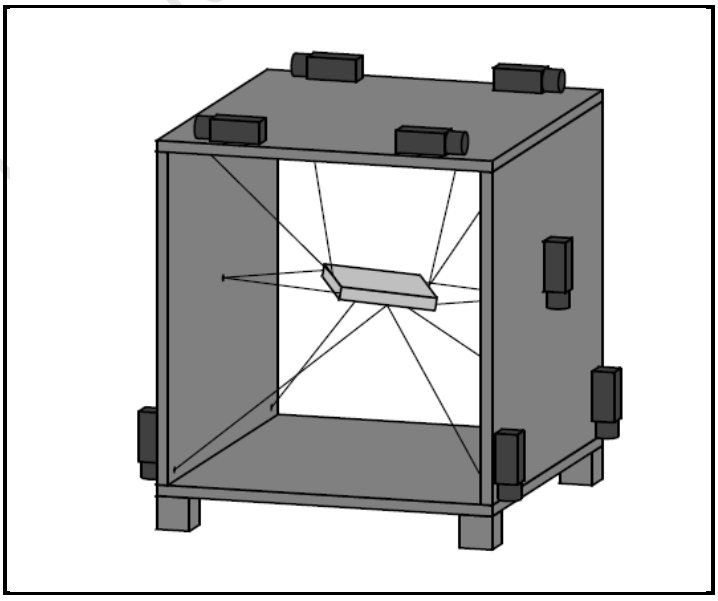


Figure 11: Showing a tendon-based parallel system. [Richard Verhoeven (2004)]

In analysing the system, the following characteristics are to be taken into much consideration:

- Forward kinematics – this is a computation of the position and orientation of the robot's end-effector as a function of its joint angles. **[34]**
- Reverse kinematics – this is a computation of the joint angles from the end-effector position and orientation (opposite of forward kinematics). **[34]**
- Singularity – this is a position in the robot's workspace where one or more joints no longer represent independent controlling variables. Commonly used to indicate a position where a particular mathematical formulation fails. **[35]**

The mechanical design of tendon based systems can be simple, but these systems are computationally demanding because:

- The workspace is restricted by the requirement to have positive but limited tension in all tendons. Their controllable workspace has a complicated shape and cannot be expressed in closed form. A related problem in these systems is to find an optimal distribution of tension between tendons, especially in highly redundant systems.
- For n end-effector degrees of freedom, at least $m = n + 1$ tendons are required; hence the systems must be kinematically redundant, and this can be used to reduce singularities. But there exist designs where additional legs do not change kinematics at all. This has to do with the problem of forward kinematics.
- Due to the nonconvex controllable workspace, even a simple point-to-point motion requires a complex algorithm to find a trajectory which lies inside the controllable workspace without consuming too much time and energy.

In a paper on the analysis of a tendon based parallel platform, Richard Verhoeven (2004) summarizes the workspace aspect of the platform. He says the major problem with a tendon-based Stewart platform is the rather small workspace compared to the serial manipulators. The technically interesting part of workspace is the set of those postures which satisfy the following conditions:

- the platform is controllable;
- tendon forces are positive;
- tendon forces lie between minimum and maximum tension;
- the end-effector is far from singularities;
- the structure is sufficiently stiff;

- tendons do not intersect with each other. **[36]**

2.5.3.1 Problems of Tendon-Based Parallel Robots

Although tendon-based parallel robots possess the above good qualities, they also have disadvantages that have been discovered and these are outlined below:

- To find acceptable force distributions in the tendons, but this can be solved by a nonlinear optimization algorithm.
- Trajectory planning of these systems faces a number of constraints, including workspace limitations, actuator and tendon limits, as well as smooth motion constraints.
- Force distribution on linear motion is difficult.
- Turning moments not easily measured.
- Workspace is limited by collisions between tendons, which appear only in systems with rotational DOFs. Collisions can be reduced if tendon attachment points are put together in groups.

In general, all tendon-driven robots, whether they are serial or parallel, share the property that tendons support tension but not compression. Thus, for *n* DOFs, at least *n + 1* tendons are required. In several publications by Tsai et al **[37]**, **[38]**, he examined these robots in the context of isotropic configurations of serial systems. The author explains that it is important to distinguish between actuator space (the tendon lengths/forces) and joint space (the angles/torques of the revolute joints). The matrix transforming the actuator forces in joint torques is called structure matrix. In his observation, it turns out that isotropic configurations are obtained if either *n +1* or *2n* tendons are used. In the *2n* case, each joint is then driven by two antagonistic actuators. This result is interesting because it shows us a fundamental difference between tendon-driven serial systems and tendon-based parallel ones: in the latter, joint space and actuator space coincide. In that sense, tendon-based parallel systems are directly driven.

2.5.4 Parallel robotic Manipulators

In these robotic systems, each actuated leg has to carry only a part of the payload; this is quite energy efficient and the robot can handle heavy loads. On the other hand, it also implies that the end-effector moves slower than the actuated joints. Control is a nontrivial issue because

the forward kinematics problem may have up to 40 solutions and in general it cannot be computed in real time. **[39]** A major problem in practice is the need for spherical joints with a very large angular range and a sufficient stiffness at the same time.

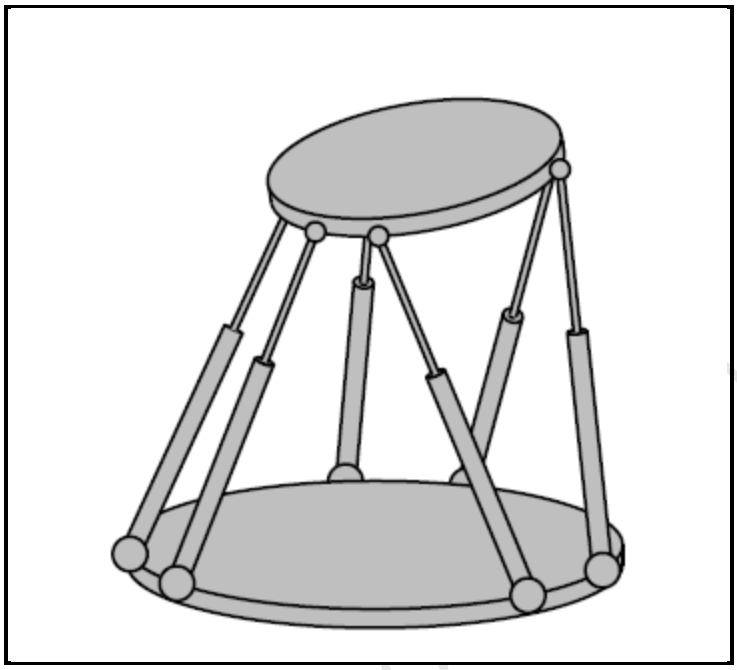


Figure 12: Showing a parallel system (Stewart Platform) (Adapted from Richard Verhoeven (2004))

2.5.4.1 Advantages of Parallel Manipulators

Six degrees of freedom: The Stewart platform consists of six struts which expand and contract between the movable platform, which carries a spindle, and the fixed platform. The coordinated motion of these six struts enables the spindle to move in any direction. In addition to the traditional motion in orthogonal axes, *X, Y,* and *Z,* the device is also able to move in the rotary complements of pitch, yaw, and roll *(q, a, j).* This advantage allows the spindle to reach unusual angles and geometrical features.

High precision and accuracy: The Stewart platform technique requires all six struts to alter their lengths if a change of the platform in only one axis is required. On the other hand, if only one strut alters its length, all six coordinates *(X, Y, Z, q, a, j)* will change. Unlike other multi-axis positioning devices, in which any change in one coordinate influences the position of the pivot point and the other coordinates, the Stewart Platform can compensate itself automatically. **[40]**

High Load/Weight Ratio: The Stewart platform offers high stiffness and rigidity of its components and all moving parts, such as bearings and joints, and drive screws. In addition, high stiffness, the Stewart platform's components will prevent any bending effect in the six legs, and that is what makes the Stewart platform very advantageous. The weight of a load in the platform is approximately equally distributed on the six parallel links. That means each link just suffers from only 1/6 of the total weight. Furthermore, under certain load, the struts on the Stewart platform act longitudinally and therefore exert either tension or compression on the struts in other words, no axial forces are applied. **[40]**

2.5.4.2 Disadvantages of Parallel Manipulators

Friction: Friction in the ball joints is a crucial problem for the Stewart Platform. The friction coefficient is about 0.8, and that is enough to exert some axial deflection on the struts that influences the accuracy and repeatability.

Length of Struts: The length of the struts affects the accuracy of the machine. When the length increases, the accuracy decreases dramatically (possibility of bending). This problem though, has been overcome by mapping each screw in the joints before installing in the machine in modern platforms. **[40]**

Dynamic thermal growth: This problem has appeared also in the serial linkage mechanism. That with the increase in the speed of the spindle, there is a dramatic increase in the dynamic thermal growth.

Calibration: The accuracy of the parallel-mechanism is not only dependent upon an accurate control of the length of its links but also upon knowledge of its geometrical characteristics. According to the fabrication tolerances many factors will play a role in the final accuracy of the machine. Up to 132 parameters must be specified to describe the geometrical characteristics of the mechanism which seems to be very difficult to adjust all parameter. Therefore, the calibration of the Stewart Platform is still an open problem. **[40]**

In summary, most 6 DOF platforms are simply different designs based on the above platforms. They could either be one or a combination of the above manipulators. In this project the design was based on a combination of two of the above manipulators. Those are the serial and parallel manipulators, thus producing a serial-parallel manipulator.

CHAPTER THREE
HARDWARE DESIGN

3.1 Platform Design

Specifications restated

In designing the actual mechanical structure of the platform, forces on six different axes were considered. The platform was designed in a way that it would have three translational degrees of freedom and three rotational degrees of freedom in a serial-parallel mechanism of a manipulator. In the scope of the design, if a force was to be applied along a specified axis, the platform had to respond by moving along the axis in response to the force along the axis. In other words, the movement of the platform had to equal the force applied along the axis in question. The axes of movement had to be independent of each other. That is, it should be possible to lock the unwanted axes and measure forces on the axes in question.

In designing the platform, the approach taken was to design the rotational and the translational degrees of freedom separately. Since the axes had to be independent to each other, a single axis had to be designed to work independently without affecting other axes. This was not possible because of the rotational movements of the platform. Simulating the designs was done using the Pro-Eng software [Pro Engineer Wildfire 3 Student Edition]. A wire model of the designs was made to provide a more real evaluation of the design at hand.

3.1.1 Rotational Design

3.1.1.1 Design one

The figure below shows how the mechanical structure was initially designed. It has two rotating square blocks (for the pitch (green) and roll (grey)) on its top and two supporting struts mounted on a circular rotating base (for the yaw (red)). The axes of the roll and pitch depend on the orientation of the base (yaw). It was possible to lock the axes and focus on a single axis provided the other axes are in a desired position. A wire model of the platform was designed and analysed.

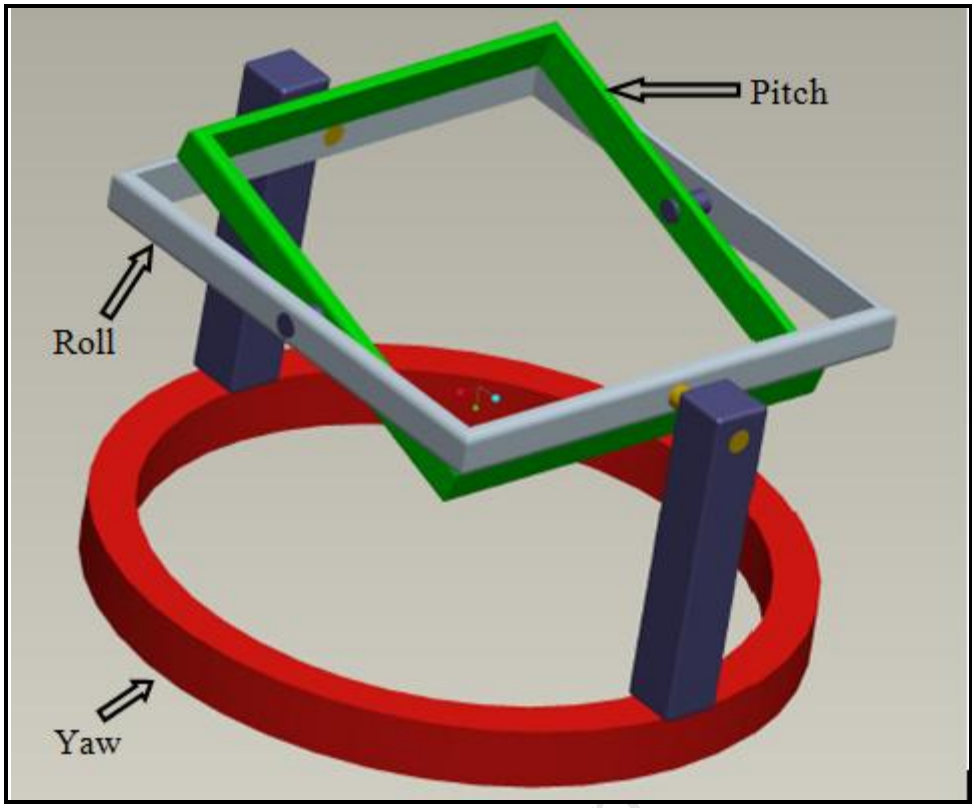


Figure 13: Showing the initial software design of the rotational part of the platform

Table 2: Table showing the properties of the initial design of the rotational system of the project

Axis	Part Colour	Length (mm)	Thickness (mm)	Height (mm)	Diameter (mm)
Pitch	Green	140	20	30	N/A
Roll	Grey	200	20	30	N/A
Yaw	Red	N/A	30	45	250
	Blue	170	30	30	N/A

The property outlined above were used to build the wire model shown below.

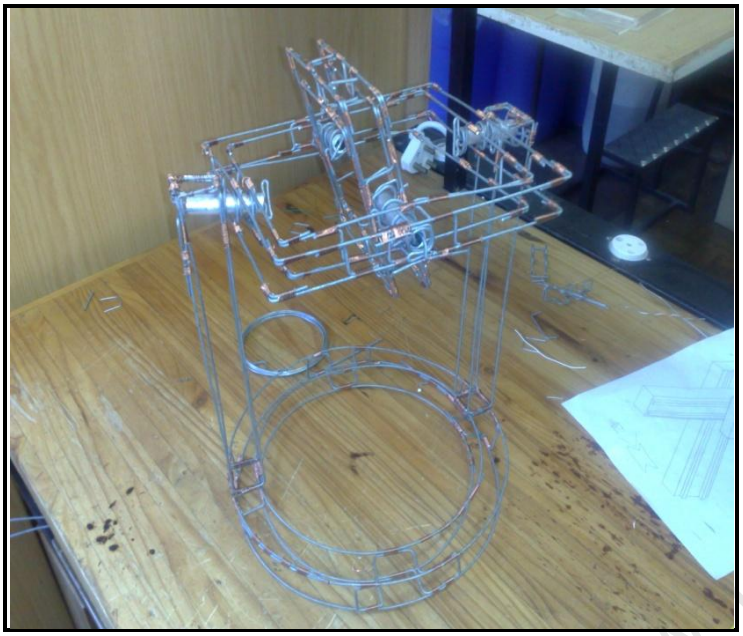


Figure 14: Showing the wire model of the initial hardware design

3.1.1.2 Design one analysis

Though all the rotational axes are present, and it was highly possible that they would function correctly, a lot of cons were present in this design.

- Powerful actuators would be needed to move the platform.
- Force sensor design and mounting would be very difficult.
- Positioning the translational part of the platform was difficult.
- Mounting the flying object will pose a huge amount of difficulty when connecting to each individual axes.
- Placing actuators and tendons would complicate the dynamics of the movements and forces on each axis.

3.1.1.3 Design two

After consultation with supervisors and lab-mates, a second design was generated. The second design took into consideration the placements of the translational axes and the attachment of the flying object. In the second design the rotational parts were placed at the bottom of the platform. Figure 15 below shows the design.

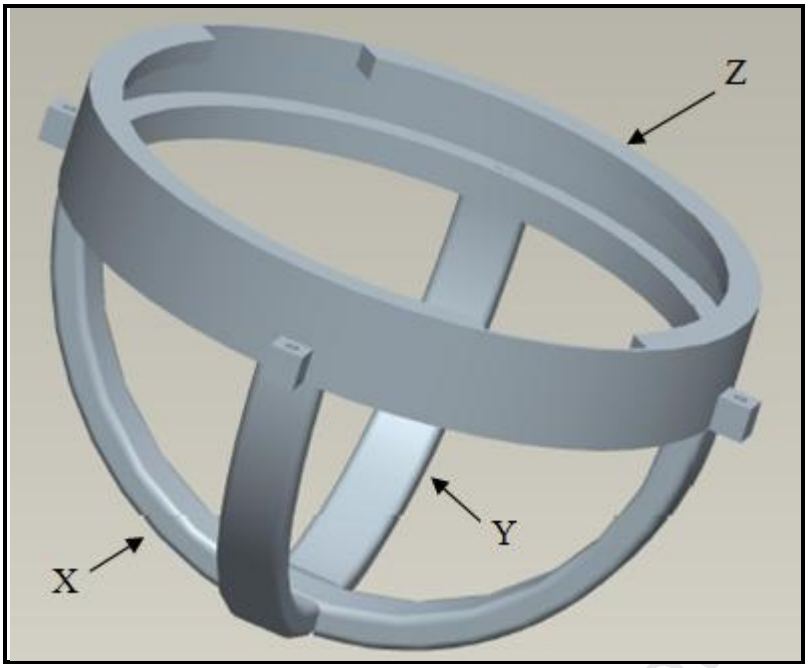


Figure 15: Showing the simulated rotational platform

Table 3: Showing the properties of the second design of the rotational system

Axis	Part	Thickness (mm)	Height (mm)	Diameter (mm)
Pitch	X	20	25	250
Roll	Y	20	25	250
Yaw	Z	20	45	250

The properties outlines above were used to build the wire model shown below.

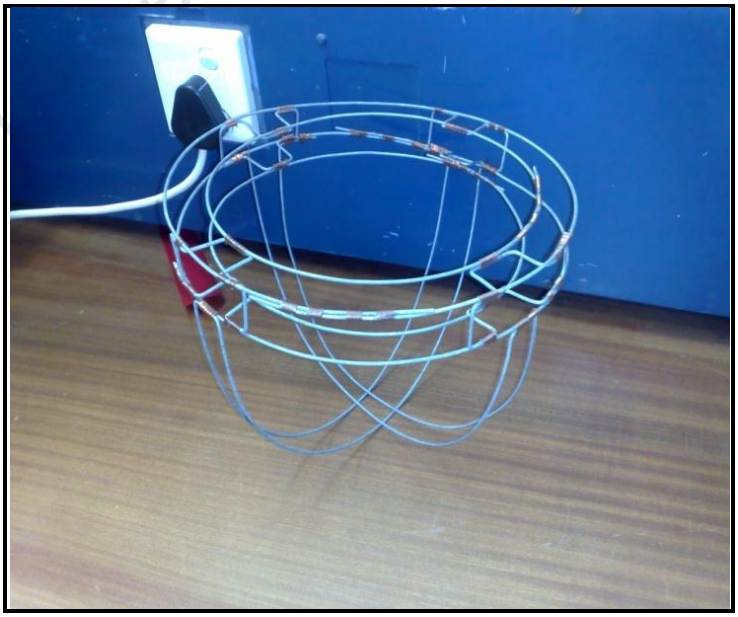


Figure 16: Showing the wire model of the rotational platform

3.1.1.4 Design two analysis

This design was much simpler than the first, and much more likely to achieve the objectives of the project. This design provided a good starting point for the mechanical structure for the project. A wire model of this design was made and evaluated. It was then concluded that the design had many cons that would hinder achieving certain aspects of the project requirements. These are listed below:

- An inevitable loss of smooth rotation
- Difficulty in platform positioning
- The size would make it difficult to control
- It would need a small resolution in tilting angle to achieve the goal.

3.1.1.5 Design three

The third design came to be after the evaluations of the second design. It was a modification of the second design to make sure the requirements of the project were satisfied. This design became the final design of the rotational part of the platform for the project. A lot of other options to modify the design were considered, for example the tendon placement and actuator mounting for ease of control of the platform. A dovetail joint was used to connect the tendon attachment handles to the base and is indicated below. Since the forces at hand would act vertically to the platform handles (green connectors), the dovetail joint restricts the upward (vertical) movements of the handles. This allows the platform to remain taut to the round base and maintains its axes all the time regardless of the platform orientation.

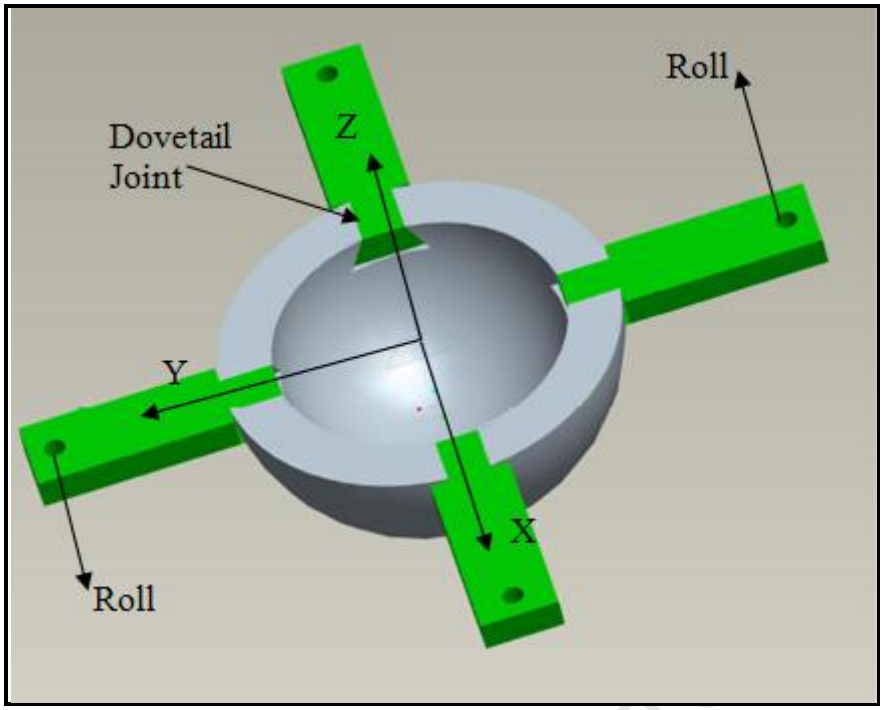


Figure 17: Showing the final design of the rotational part of the platform

3.1.1.6 Design three analysis

Based on software simulations, it was concluded that it was the best design for the rotational part of the project. Maintaining a smooth rotation and reducing the size of the platform would make it achieve the goals.

3.1.2 Translational Design

In designing the translational part of the system, the friction between the sliding plates was to be minimised as small forces generated by the flying object needed to be measured accurately. The surfaces had to be as close to being frictionless as possible. This property of the platform influenced the decision on the type of material to be used for the project. So to reduce friction between the sliding plates, it was decided that the sliding parts be made of different material. The reason is stated below.

When considering the factors that affect friction between dry surfaces, that is;

- For low surface pressures the friction is directly proportional to the pressure between the surfaces. As the pressure rises the friction factor rises slightly. At very high pressure the friction factor then quickly increases to seizing.
- For low surface pressures the coefficient of friction is independent of surface area.
- At low velocities the friction is independent of the relative surface velocity. At higher velocities the coefficient of friction decreases.

The above reasons also influenced the speed of the actuators in the control of the platform. The platform was perceived to have both the static and sliding friction. The two are outlined below.

3.1.2.1 Friction between plates

The static friction coefficient μ between two solid surfaces is defined as the ratio of the tangential force F required to produce sliding divided by the normal force between the surfaces N

$$\mu = F/N$$

For a horizontal surface the horizontal force F to move a solid resting on a flat surface

$$F = \mu \times mass\ of\ solid \times g$$

Where g is the acceleration due to gravity. If a body rests on an inclined plane the body is prevented from sliding down because of the frictional resistance. If the angle of the plane is increased there will be an angle at which the body begins to slide down the plane. This is the angle of repose and the tangent of this angle is the same as the coefficient of friction.

When a tangential force F surmounts the frictional force between two surfaces then the surfaces begin to slide over each other. If a body is resting on a flat surface, that body starts to move. The sliding frictional resistance is generally different to the static frictional resistance. The coefficient of sliding friction is expressed using the same formula as the static coefficient and is generally lower than the static coefficient of friction. **[41]** Figure 18 below shows the relationship between a net force and the static and sliding frictional forces.

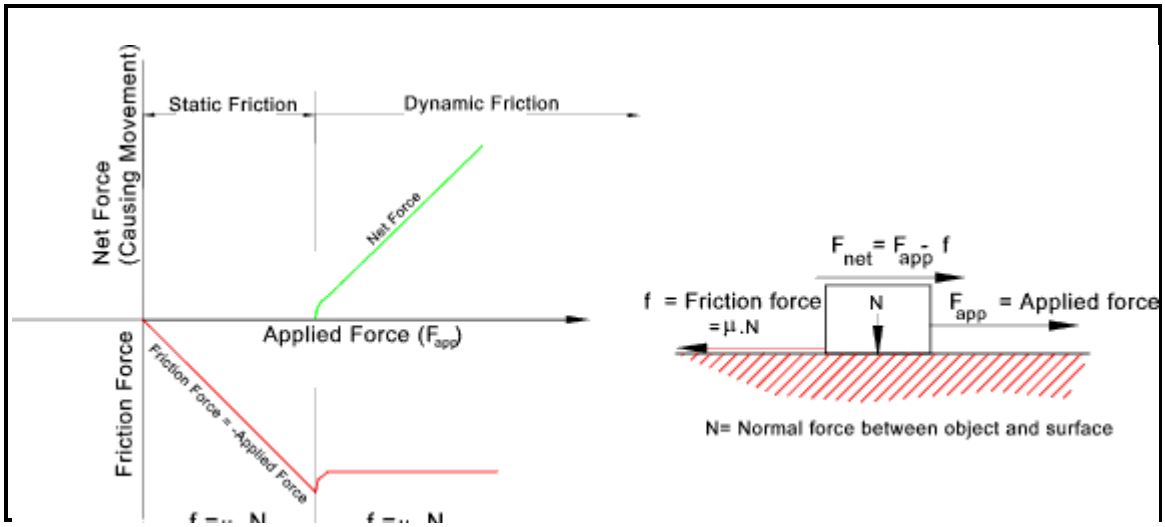


Figure 18: Figure showing the relationship of a net force on static and dynamic friction [41]

In figure 18 above, the static friction has to be overcome by the applied force before there can be any movement (sliding between surfaces). As the applied force increases, it eventually reaches a point where it is equal to the static frictional force. That point is called the limiting equilibrium point. A further increase in the applied force, the body starts to slide.

3.1.2.2 Orthogonal Plane Design

The three translational axes of this platform where designed to be orthogonal with respect to one another. The design followed the normal Cartesian 3D plane surfaces. The X plane was the base of the translational part of the platform and then the Y plane sliding over it and lastly the Z plane sliding orthogonally on the Y plane. The joint between the planes was to be able to make the sliding possible while maintaining the orthogonal angle between the planes of the two axes in question. The figure below shows the wire model of the X and Y planes.

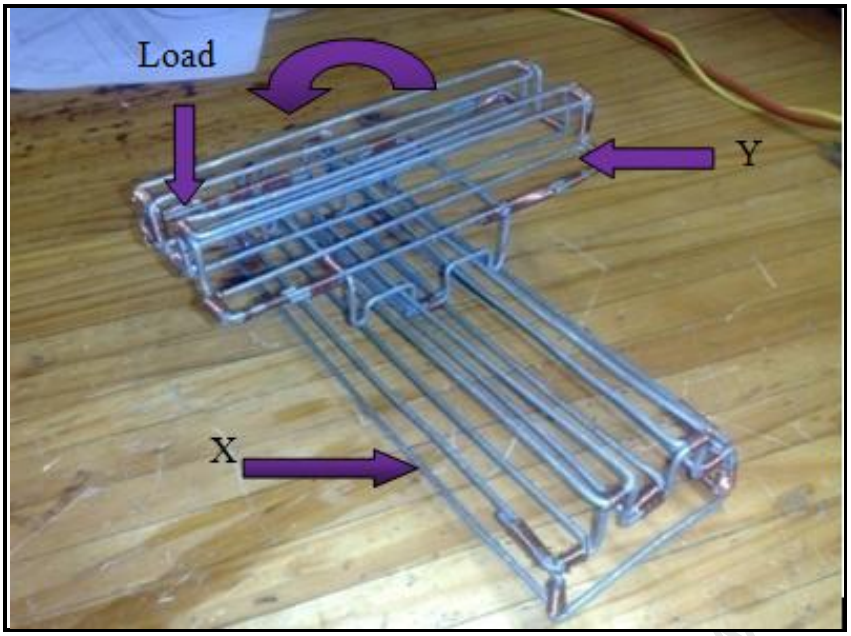


Figure 19: Showing the wire model of the initial translational design

3.1.2.3 Orthogonal Plane Design Analysis

It was discovered that with a weight loaded at the edge of the Y plane, it would topple off and not maintain stability. This problem led to a change of the joint between the planes. The modified joint that would have the above abilities and also prevent the plates from toppling regardless of platform orientation (tilt) is shown below.

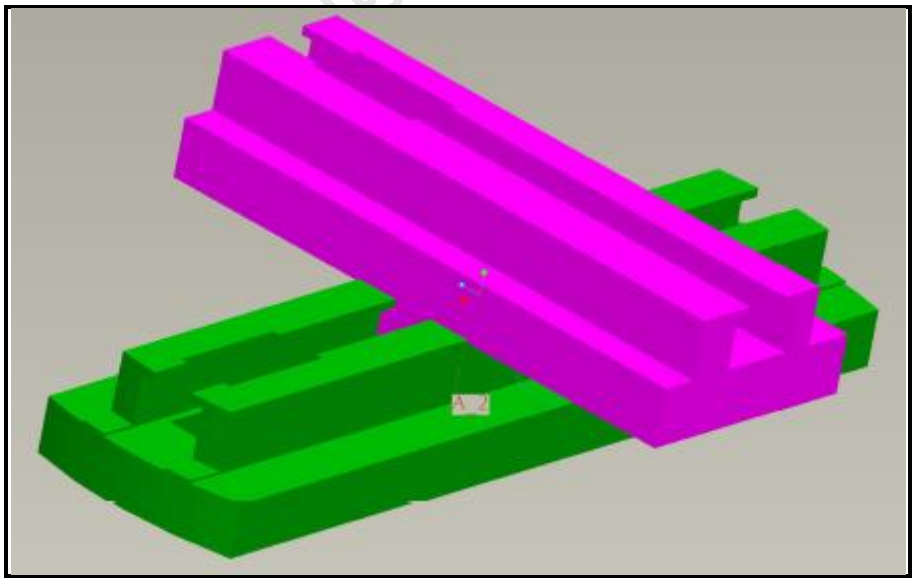


Figure 20: Showing the locking mechanism of two orthogonal sliding planes of the final design of the platform where the green block is the X-plate and the purple block is the Y-plate

3.1.3 The 3-Translational Axes Design

The translational part of the platform was designed to work independently with the rotational part of the platform. The 3-axes are shown below in a Pro-Eng design configuration. In the diagram below it is shown how it would connect to the rotational part to form a combined 6 DOF platform.

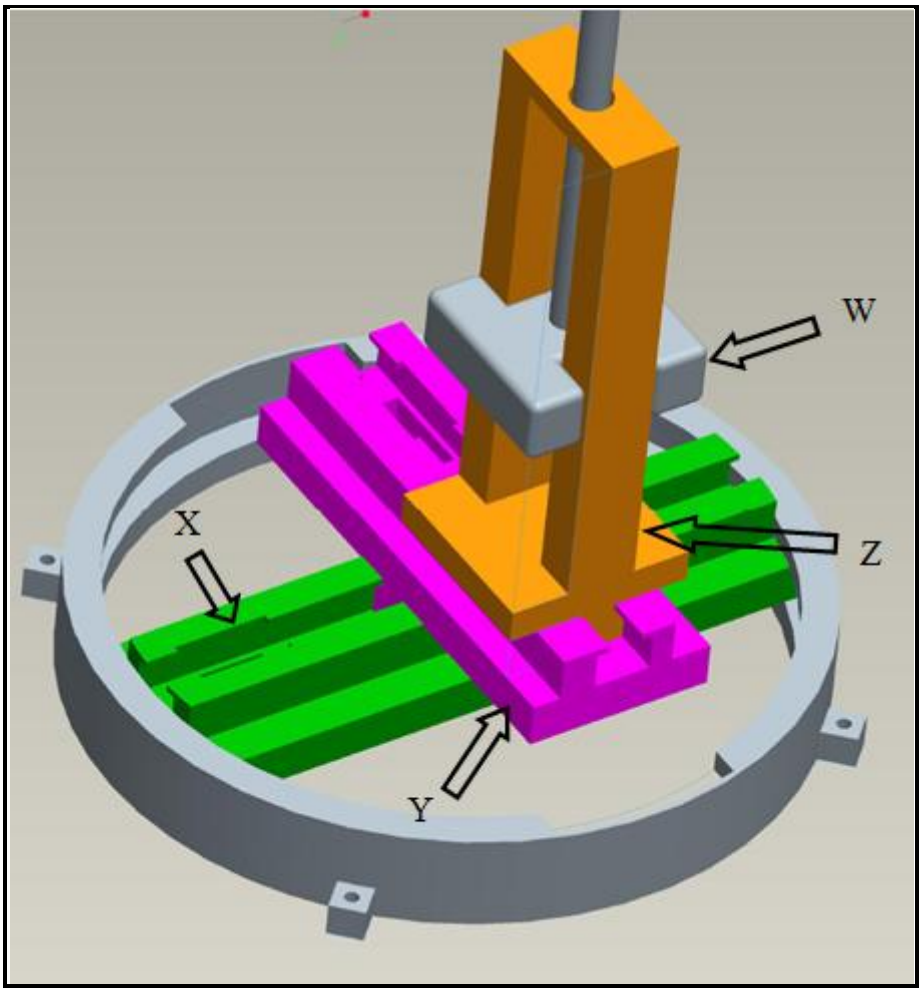


Figure 21: Showing the assembled locking mechanism of the 3 orthogonal translational axes of the design

The purple block is the Y-axis block that slides over the green (X) block. The actuators are mounted on the green block hence the X-axis controls the platform. The brown block is the Z-axis block that slides over the Y (purple) block and controlling the movement on the Y direction. The grey block W is the block that slides up and down and thus controls the up-down movements of the platform in the Z direction.

3.1.4 The 6 DOF Final Software Design

The final design of the project was also done in Pro-Eng. It is shown in the diagram below.

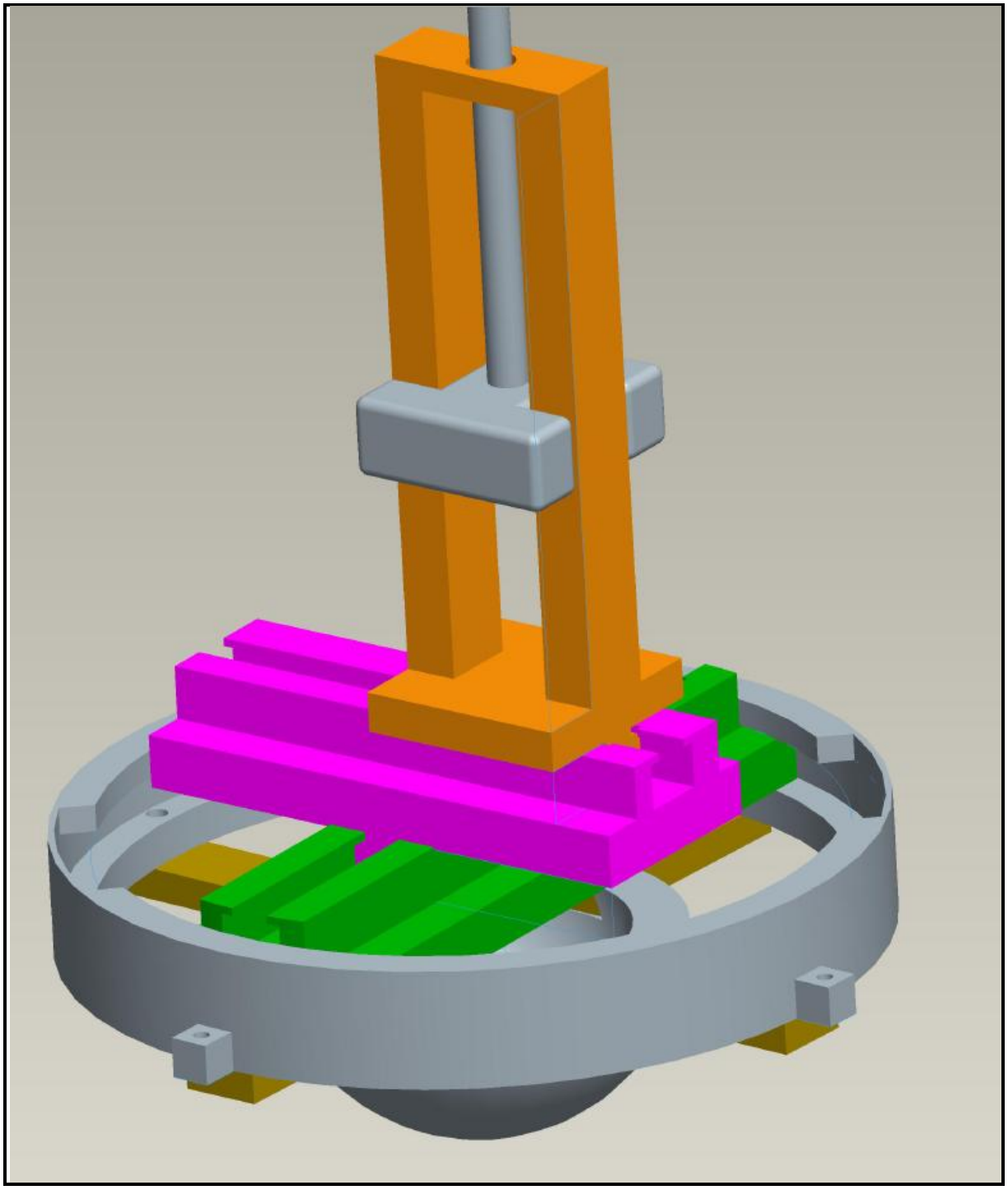


Figure 22: Showing the final intended design of the project platform

3.1.5 The 6 DOF Final Hardware Design

Although the actual design was intended to look exactly like the above software design, in the end it was modified as the department that was in charge of the prototyping machine could

not allow for the last bit of machining and so the up-down design of the Z-axis force measurement had to be changed and this is shown in the figure below. Though also not shown above, the contact surface between the translational planes was changed to Teflon-Aluminium surfaces because these two surfaces have a low Friction coefficient both statically and kinetically. As the forces to be measured are small, the contact surfaces had to be as close to being frictionless as possible.

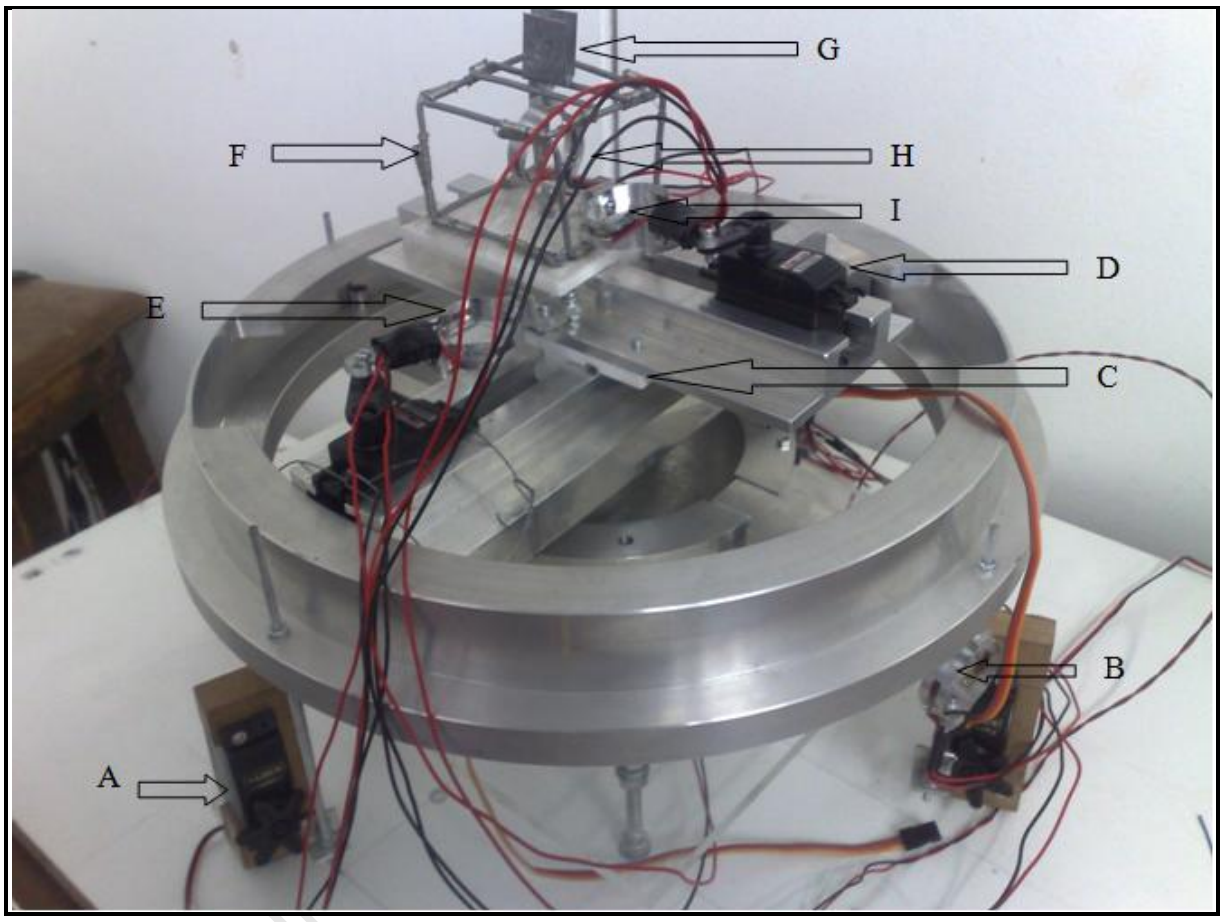


Figure 23: Showing the Final Actual design of the project. A and D are actuators (servo motors) responsible for rotational and translational motion of the system. B is a load cell that measures the rotational forces and E, I and H are load cells that measure forces on three different translational axes, that is the X, Y and Z respectively. C is the teflon block between aluminium surfaces to reduce friction and F is the Z axis block with a teflon base locked on the movable Y plate where the load cell that measures upward forces is mounted.

Table 4: Specifications of the Servos used

JR Servo Specifications	Size mm	Weight g	Torque (kg.cm)	Speed (Sec/60^{o})	Voltage (V)
Futuba S3010	40 x 20 x 38.1	41	6.5 @ 6V	0.16 @ 6V	6
Power HD 6001HB	41.9 x 20.6 x 39.6	43	6.7 @ 6V	0.14 @ 6V	6

Table 5: Showing the components used in the Design

Part	Size (mm)	Weight (Kg)	Quantity
Round Ring	R165 x 45 x 20	2.450	1
Power HD 6001HB servo	41.9 x 20.6 x 39.6	0.043	2
Futuba S3010 servo	40 x 20 x 38.1	0.041	4
Strain Gauge length	11 x 5	N/A	10
Load Cell	R15 x 8 x 0.5	0.011	5
Dow	R75 x R75	1.200	1
X-plate (Al)	R147.5 x 40	0.960	1
Y-plate (Al)	200 x 80 x 20	0.250	1
Y-plate (Teflon)	80 x 80 x 20	0.090	2
Z-Plate (Wire)	50 x 50 x 50	0.075	1
Dovetail rod	95 x 30 x 12	90.000	4

Table 2 above shows the specifications of the servo motors used and table 3 shows the material used for the project.

3.2 Platform Design Analysis

3.2.1 Hardware Dynamics Analysis

The hardware was analysed after it was fully built. This section outlines the mechanical analysis of the dynamic performance of the platform.

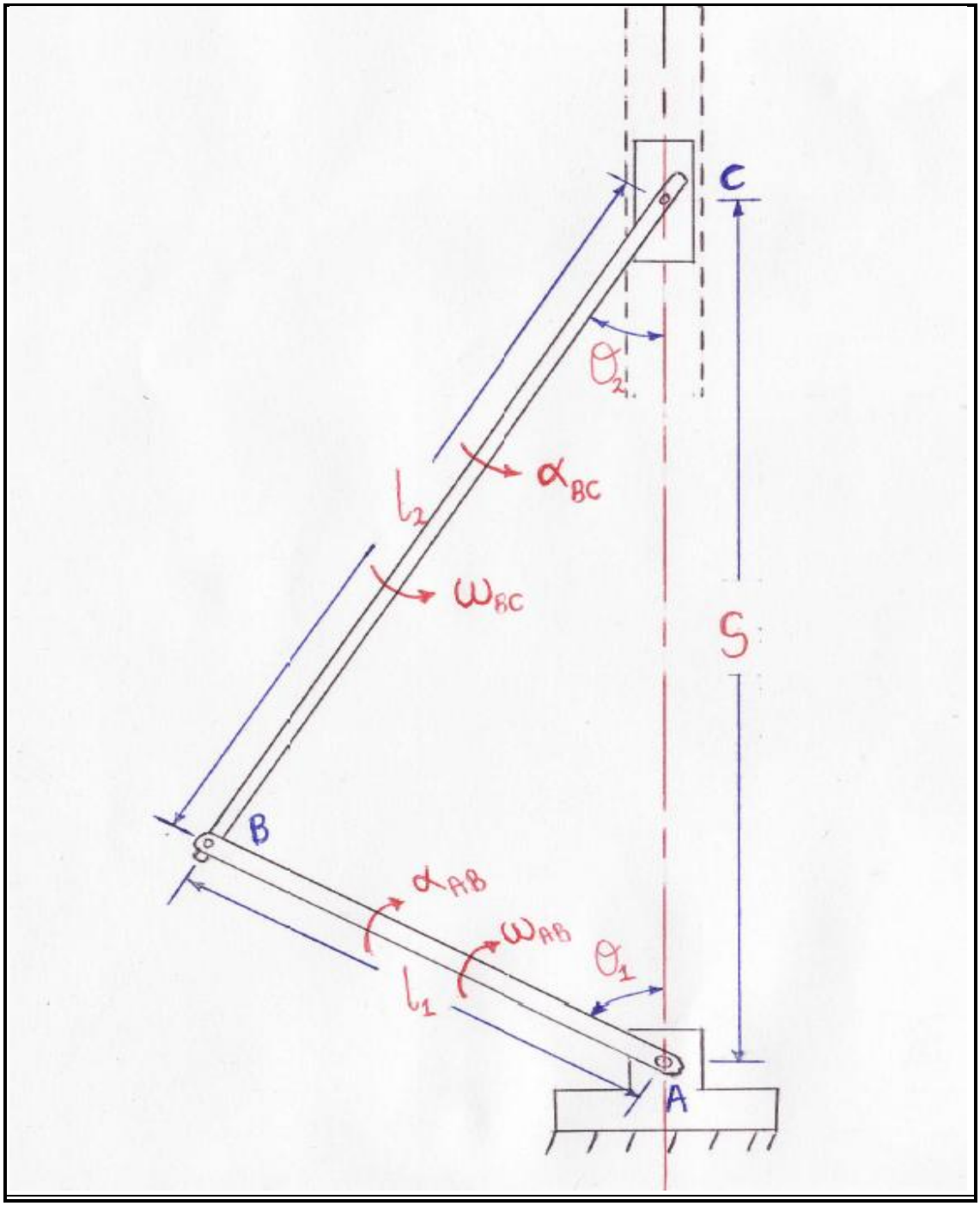


Figure 24: Showing the mechanical schematics of the actuator design where A is the servo motor, B is the servo horn and actuator connection and C is the sliding plane and S is the distance moved.

The figure above shows the free body diagram for analysing the translational part of the platform. Point A, represents the point at which the servo horn is attached on the servo motor (actuator). Bar AB represents the servo horn, and bar BC represents the bar supporting the force transducer. Block C, represents the movable (sliding) block on the plane. The force load cell and attachment beams where taken as a rigid beam in the analysis of the platform. The angle θ_1 represents the servo angle.

3.2.1.1 Distance Moved

In the design of the platform, the movement of the planes had to be restricted. This was based on the length of the servo horn and the amount of space available to mount the force transducers. The distances l_1 and l_2 where fixed distances and thus the length of distance s at any given time could be calculated using the cosine rule. The servo angle θ_1, was also known from software design in controlling the actuators. Using the cosine rule and the servo angle we get the relationship:

$$l_2^2 = l_1^2 + s^2 - 2 \cdot l_1 \cdot s \cdot cos\theta_1$$

Now in finding the actual distance moved, the reciprocating motion of the movable plane was considered with respect to the servo angle and changing the subject of the formulae we get:

$$l_2^2 = l_1^2 + s^2 - 2 \cdot l_1 \cdot s \cdot cos\theta_1$$
$$l_2^2 - l_1^2 = s^2 - 2 \cdot l_1 \cdot s \cdot cos\theta_1 + l_1^2[(cos^2\theta_1 + sin^2\theta_2) - 1]$$
$$l_2^2 - l_1^2 + l_1^2 - l_1^2 sin^2\theta_1 = s^2 - 2 \cdot l_1 \cdot s \cdot cos\theta_1 + l_1^2 cos^2\theta_1$$
$$l_2^2 - l_1^2 sin^2\theta_1 = (s - l_1 cos\theta_1)^2$$
$$s - l_1 \cdot cos\theta_1 = \sqrt{l_2^2 - l_1^2 sin^2\theta_1}$$
$$\therefore \quad s = l_1 \cdot cos\theta_1 + \sqrt{l_2^2 - l_1^2 sin^2\theta_1}$$

3.2.1.2 Force Angle

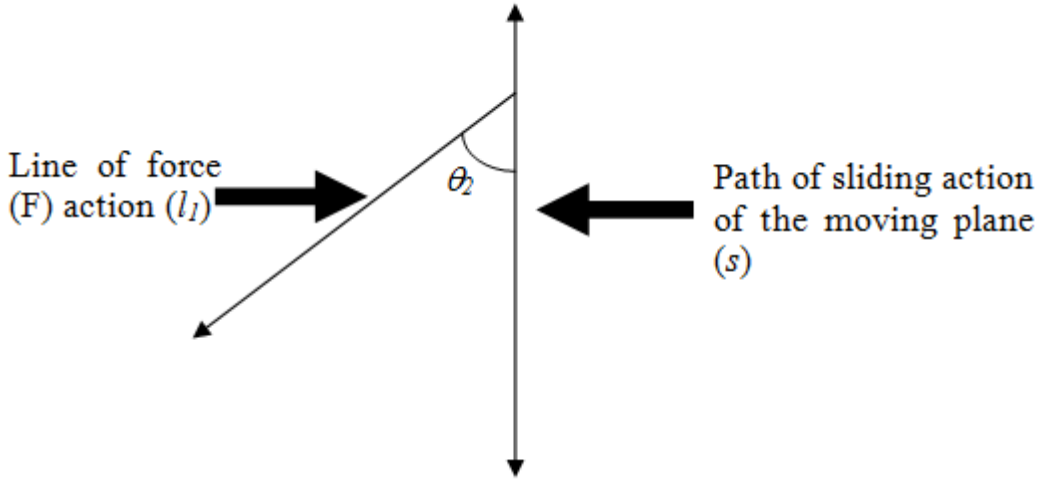
From the above diagram and using the sine rule, the angle θ_2 can be found to be:

$$\frac{sin\theta_1}{l_2} = \frac{sin\theta_2}{l_1}$$

$$sin\theta_2 = \frac{l_1 \sin\theta_1}{l_2}$$

$$\theta_2 = \sin^{-1}\left(\frac{l_1 \sin\theta_1}{l_2}\right)$$

3.2.1.3 Actual Force on Translational Axis



Therefore, from the above diagram, the actual force that causes the acceleration in the vertical axis is given by:

$$F_y = Fcos\theta_2$$

$$F_y = F\cos\left(\frac{l_1 \sin\theta_1}{l_2}\right)$$

Vector Analysis: The kinetic diagrams for both AB and BC are shown in the figure above. Here $\mathbf{a_c}$ is vertical since C moves along a straight-line path. Expressing each of the position vectors in Cartesian vector form:

$$r_B = -l_1 sin\theta_1 i + l_1 cos\theta_1 j$$

$$r_{C/B} = l_2 sin\theta_2 i + l_2 cos\theta_2 j$$

Crank Shaft AB (Rotation about a fixed axis):

$$a_B = \alpha_{AB} \times r_B - \omega_{AB}^2 r_B$$

$$a_B = \alpha_{AB} k \times (-l_1 sin\theta_1 i + l_1 cos\theta_1 j) - \omega_{AB}^2 (-l_1 sin\theta_1 i + l_1 cos\theta_1 j)$$

After expanding and collecting like terms together we get:

$$a_B = l_1\alpha_{AB}(-cos\theta_1 i + sin\theta_1 j) + l_1\omega_{AB}^2(sin\theta_1 i - cos\theta_1 j)$$

Connecting Rod BC (General plane motion): Using the result for a_B and noting that a_C is in the vertical direction, we have:

$$a_C = a_B + \alpha_{BC} \times r_{C/B} - \omega_{BC}^2 r_{C/B}$$

$$a_{Cj} = a_B + \alpha_{BC}k \times (l_2 sin\theta_2 i + l_2 cos\theta_2 j) - \omega_{BC}^2(l_2 sin\theta_2 i + l_2 cos\theta_2 j)$$

$$a_{Cj} = l_1\alpha_{AB}(-cos\theta_1 i + sin\theta_1 j) + l_1\omega_{AB}^2(sin\theta_1 i - cos\theta_1 j) + \alpha_{BC}k \times (l_2 sin\theta_2 i + l_2 cos\theta_2 j) - \omega_{BC}^2(l_2 sin\theta_2 i + l_2 cos\theta_2 j)$$

Since there is no *i* component but only the *j* component and after expanding all the equation and separating the *i* and *j* components we get:

$$0 = -l_2\alpha_{AB}cos\theta_1 + l_1\omega_{AB}^2 sin\theta_1 - l_2\alpha_{BC}cos\theta_2 - l_2\omega_{BC}^2 sin\theta_2$$

$$a_C = l_1\alpha_{AB}sin\theta_1 - l_1\omega_{AB}^2 cos\theta_1 + l_2\alpha_{BC}sin\theta_2 - l_2\omega_{BC}^2 cos\theta_2$$

3.3 Force Sensor Design

Two methods of force sensing were considered in building the platform. The methods are the current detection method on a servo motor and the strain gauge force detector.

3.3.1 Current Detection

In this method the servo motor (actuator) would also be a force transducer. It uses the characteristics of the servo motor. When a servo motor is given a signal, it will move to that position and maintains it until the signal given to it is changed. That means the servo motor will resist any force applied to its horns to maintain its position. In so doing, there is current rise relative to the amount of force applied to the servo motor. The current rise is assumed to have a linear bearing to the force applied. This method was tested in the lab by loading the servo motor with known weights and the response was a rise in current on the positive servo signal input. Thus through the amount of current detected, the force applied across the servo motor could then be calculated.

3.3.2 Strain Gauge

In general, unless some sort of feedback mechanism is used, a force measurement actually entails the measurement of a deflection with a suitable calibration between the force and the deflection it produces. In most dynamometers the force is applied to some sort of spring mechanism, and the deflection thus produced is measured. In this project, the strain gauge method was used. In designing a force dynamometer, two requirements that are always in opposition were considered. These are *sensitivity* and *rigidity*.

3.3.2.1 Choosing a Strain Gauge

There are many types of bonded wire resistance strain gauges that exist commercially. Such gauges cannot be used directly to measure deflection, but must be first cemented to some object which is to be strained. A strain gauge is applied to a structure by first cleaning the surface comprehensively and then cementing the gauge in place. After cementing, the units are dried. The strain gauge manufacturer supplies two important pieces of information with every gauge; the resistance R_g and the gauge factor GF. If a force is applied across the force transducer, the resistance changes by an amount ΔR_g and the length L changes by an amount ΔL. The dimensionless relationship between the two variables is called the gauge factor GF of the strain gauge and is given by:

$$GF = \left(\Delta R_g / R_g\right) \Big/ (\Delta L / L)$$

So the higher the gauge factor the greater the electrical output but in order to measure the change in resistance, it is usual to connect the strain gauges into a Wheatstone bridge network in such a way that the change in resistance of the strain gauges can be measured. If possible four gauges are used (passive or active) so that the system is independent of temperature changes. **[42]** In this project the strain gauge arrangement used is shown below.

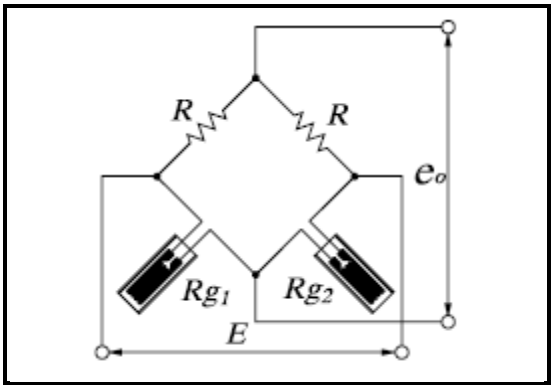


Figure 25: Showing the Wheatstone bridge arrangement, where R_g is the strain gauge, R is a standard resistor, E is supply voltage across the Wheatstone bridge and e_0 is the output voltage from the bridge.

The arrangement is a 2-active gauge system (for bending strain measurements). This system has Temperature compensation; the thermal effect of lead wires cancels and compressive/tensile strain cancels. The output is doubled. The output voltage e_0 is given by:

$$e_0 = \frac{E}{2} K_s . \varepsilon_0$$

Where K_s : Gauge factor, ε_0: Strain, E: Bridge voltage, R_g : Gauge resistance, and R: Fixed resistance.

The reason why the output is doubled is because the gauge resistors are undergoing different strains, one is under compression and the other is under tension. The one has positive strain and the other an equal (magnitude) and opposite strain. The structure of the load cell used in this project is shown in the figure below. Under the action of a tensile force on its axial dimension, the inside gauge experiences tension (stretches) and the outside gauge experiences compression (shrinks) and the opposite happens when it is a compressive force on its axial dimension.

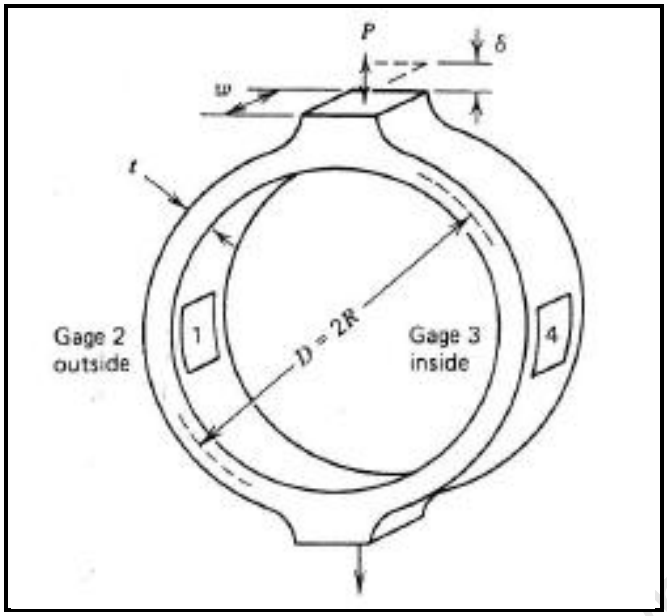


Figure 26: Showing the strain arrangement in a load cell, where gauge 1 and 3 are internal strain gauges and gauge 2 and 4 are external strain gauges. [43]

The internal and external strain gauges experience different strains in that as the internal gauges experience a tensile force, the other experiences compression as described above. *P* is the experienced force or load on the load cell which should act in a straight line as per load cell design. δ is the amount of stretch experienced by the load cell and t and w are the thickness and width of the load cell respectively. *D* is the diameter and *R* the radius of the load cell.

The relationship between the load and the stress is given by the following expression:

$$\sigma = 1.09\frac{PR}{wt^2}$$

σ is the Stress and the strain ε is given by:

$$\varepsilon = \frac{\sigma}{E}$$

Where *E* is the Young's Modulus

3.3.2.2 Voltage amplification

The strain gauge load cell outputs a very small voltage of the order of millivolts. This voltage is very small and thus voltage amplification was necessary to get a voltage that could be

worked with. The Freescale MC9S08GT16A microcontroller that was used in the project (discussed later in section 4.2) takes in a voltage between 0-3.3V on its analogue to digital inputs. Since the analogue input voltage is a continuous voltage the amplification had to maintain the continuity from the load cell. A 3-op amp in amp circuit was the first one to be considered.

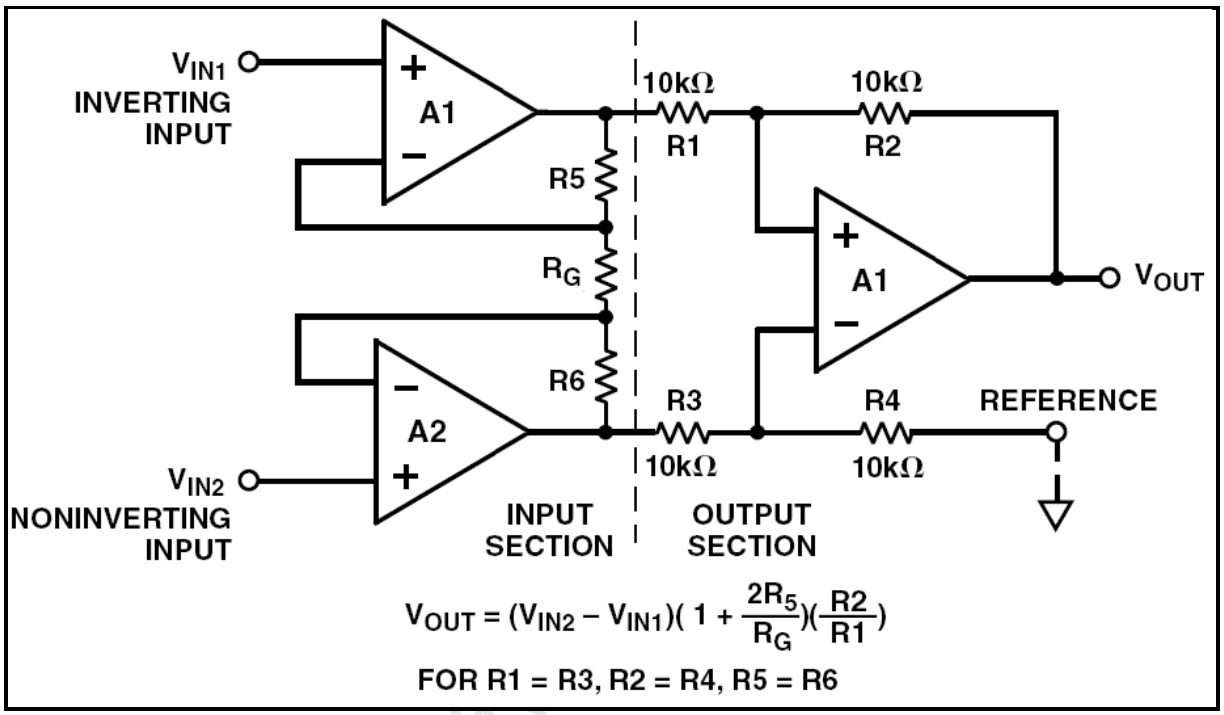


Figure 27: Showing a three-opamp instrumentation amplifier circuit
Adapted from the UCT EEE3017W Notes by Robyn Verrinder (2008)
Note: Where A1 and A2 are Operational amplifiers and R_G is the gain resistor.

The circuit partially worked but it was difficult to set the best accurate gain and proper calibration of the load cell. The output noise was also high. Since five load cells where used, this circuit would have been replicated 5 times and that was time consuming. After considering the circuit performance, monolithic chip instrumentation amplifiers were considered.

3.3.3 Circuit design

The AD620 integrated circuit (IC) was then used to better the performance. The AD620 is a low cost, high accuracy instrumentation amplifier that requires only one external resistor to set gains of 1 to 10 000. This IC possesses a high accuracy of 40 ppm maximum nonlinearity, low offset voltage of 50 μV max, and offset drift of 0.6 μV/°C max. This makes it ideal for use in precision data acquisition systems, such as weigh scales and transducer

interfaces. **[44]** This makes the IC a better device to use for measuring the forces from a load cell arrangement. In comparing the AD620 and the 3-in amp op amp the following result could be concluded:

3.3.3.1 Theory of operation

The AD620 is a monolithic instrumentation amplifier based on a modification of the classic three op amp approach. The absolute value trimming allows the user to program gain *accurately* (to 0.15% at G = 100) with only one resistor. The value of R_G also affects the transconductance of the preamp stage. When the value of R_G is reduced for larger gains the transconductance increases. This has three important advantages:

- Open-loop gain is boosted for increasing programmed gain, thus reducing gain related errors.
- The gain-bandwidth product increases with programmed gain, thus optimizing frequency response.
- The input voltage noise is reduced to a value determined mainly by the collector current and base resistance of the input devices.

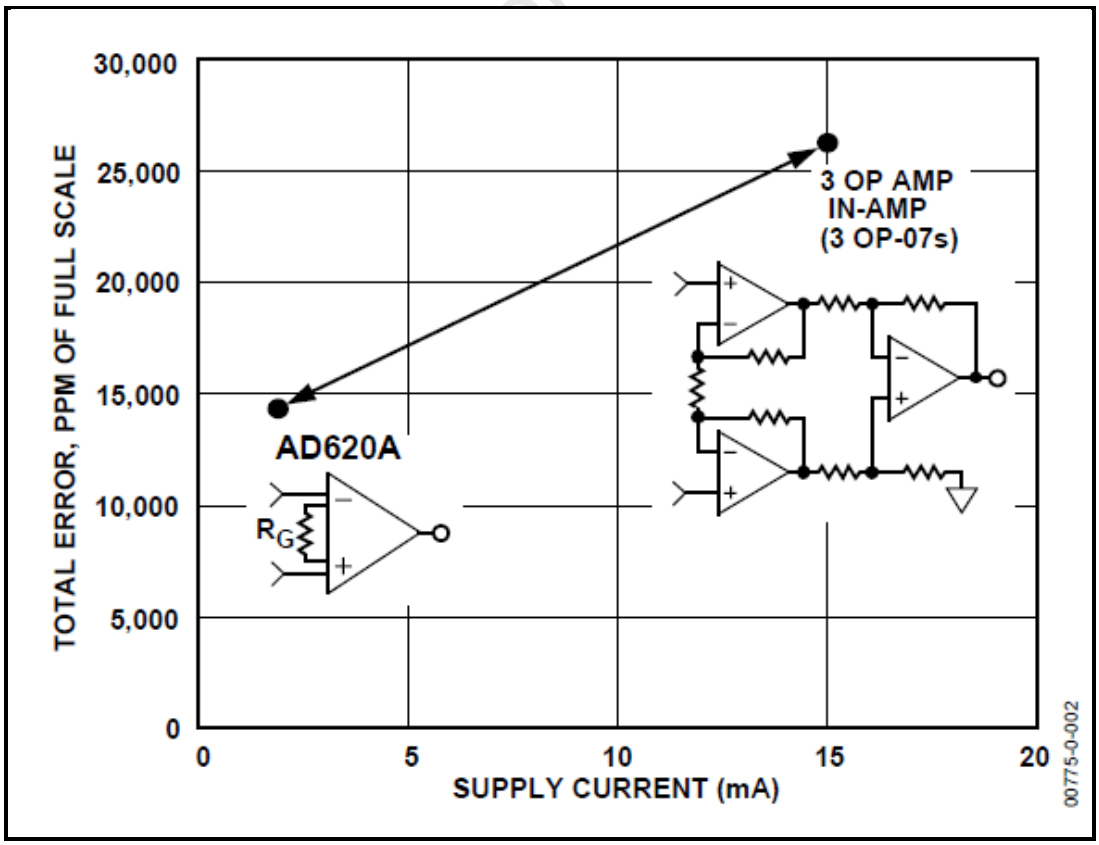


Figure 28: Showing 3 Op Amp In Amp designs Vs AD620
Adapted from the AD620 Datasheet

The two internal gain resistors of the AD620 are trimmed to an absolute value of 24.7 kΩ, allowing the gain to be programmed accurately with a single external resistor. **[44]**

$$G = \frac{49.4k\Omega}{R_G} + 1$$

$$R_G = \frac{49.4k\Omega}{G - 1}$$

From the above equations, it was decided that the total gain necessary for an optimal output be done only once on the value of the gain resistor R_G. In addition to the AD620 IC in the whole amplifying circuit, a unit gain subtractor was added to the AD620 output to remove any common-mode signal, thus yielding a single-ended output referred to the Ref pin potential. A decoupling capacitor was also added to make sure a constant voltage is supplied to the load cell and the circuit. The final circuit for the project is shown in the diagram in figure 29 below.

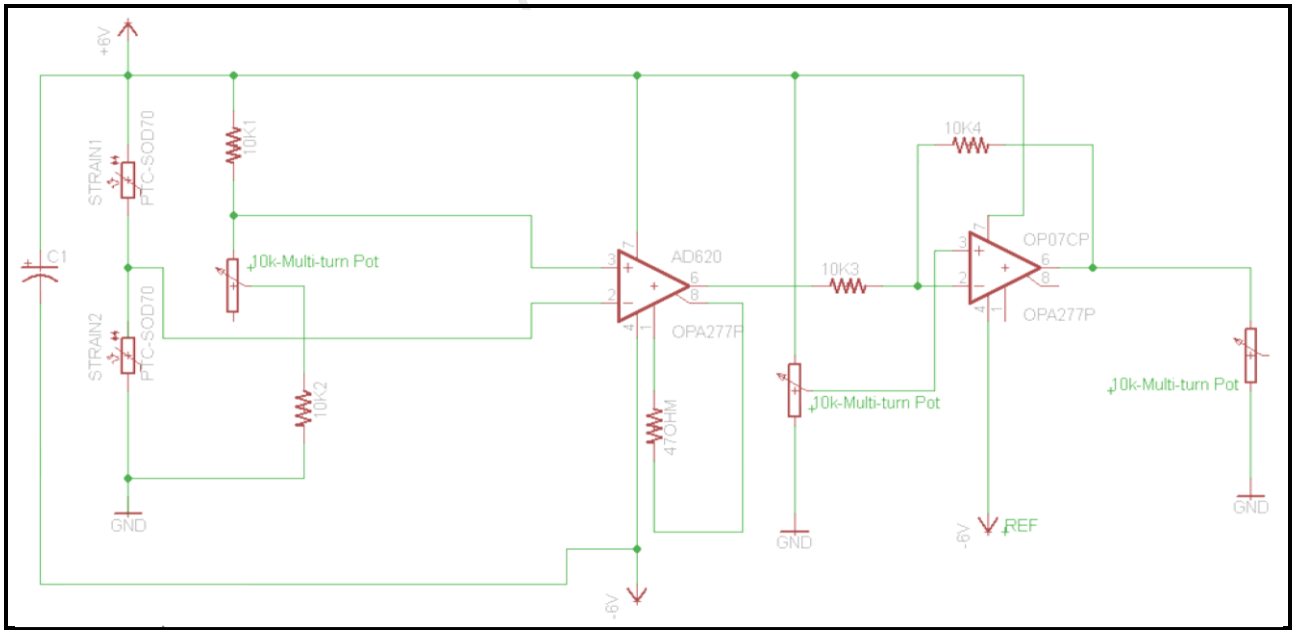


Figure 29: Showing an Eagle CAD schematic of the final circuit used in the project

CHAPTER FOUR
SOFTWARE DESIGN

4.1 Actuators

4.1.1 Servo motor

Servo motors fall into a category of closed loop systems that have characteristics that are desirable for controlling assembly position. These were incorporated in the design of the platform so that they are positioned to function efficiently. The servo motors generally possess the following features:

Constant Velocity

They have the ability to maintain speed prescribed by the velocity command input. In this project that is done in software programs for controlling the platform.

Infinite Acceleration

The motor must be able to provide infinite acceleration and this is demonstrated by the abrupt starting and stopping of the motor when it responds to commands. In practise, the actual servo motors have a critically damped response. The infinite acceleration is often traded off for damping necessary to prevent it from oscillating.

Linear response

The motor responds proportionally to changes in the velocity-input commands.

Hold Position

The micro-controller is set to drive the servo motor in such a way that the motor can hold its position regardless of inertia from a previous movement or any external force that may push on the table in its axis. Theoretically, servo motors will oscillate (critically damped) whenever the velocity command is zero volts. These oscillations are actually the forward and reverse bridge drivers alternating to hold position. In this project the microcontroller controls the motor position by varying the duty cycle (PWM, to be discussed in the next section) output of the servo drive transistors (transistors operate as switches).

Overload protection

The current through the servo is sensed using the low resistance (0.1 Ω resistor). The servo has a current to voltage circuit on each side of the bridge, so when excessive current is detected (that is a high voltage drop), the servo will trip the solenoid breaker by energizing it. **[45]**

4.2 Microcontroller

The microcontroller used in the project is the MC9S08GT16A processor which is a member of the HCS08 Family of 8-bit microcontroller units (MCUs). This family of microcontrollers uses an enhanced HCS08 core and are readily available at low cost at the University of Cape Town. The main applications that this processor is used for are the LCD, Memory, Timers and the analogue to digital converters (ADCs). These modules are discussed briefly in the figure below. **[46]**

4.2.1 Liquid Crystal Display (LCD)

These display modules have the ability not to just display numbers but also letters, words and most ASCII symbols. This ability makes them more versatile than the familiar 7-segment light emitting diode (LED) display. In this project, an LCD was used to display the magnitude of force acting on a specific axis. This was done through software programming. This module shares the digital I/O pins with the port B module in the GT16A microcontroller.

4.2.2 Memory

Memory devices come in many different forms. There are volatile (lose data when not powered) and non-volatile (store their data in power absence). The GT16A processor has an on-chip memory that consists of RAM, FLASH program memory for non-volatile data storage, plus I/O and control/status registers. The FLASH memory is designed primarily for program storage. In-circuit programming allows the operating program to be loaded into the FLASH memory after final assembly of the application product.

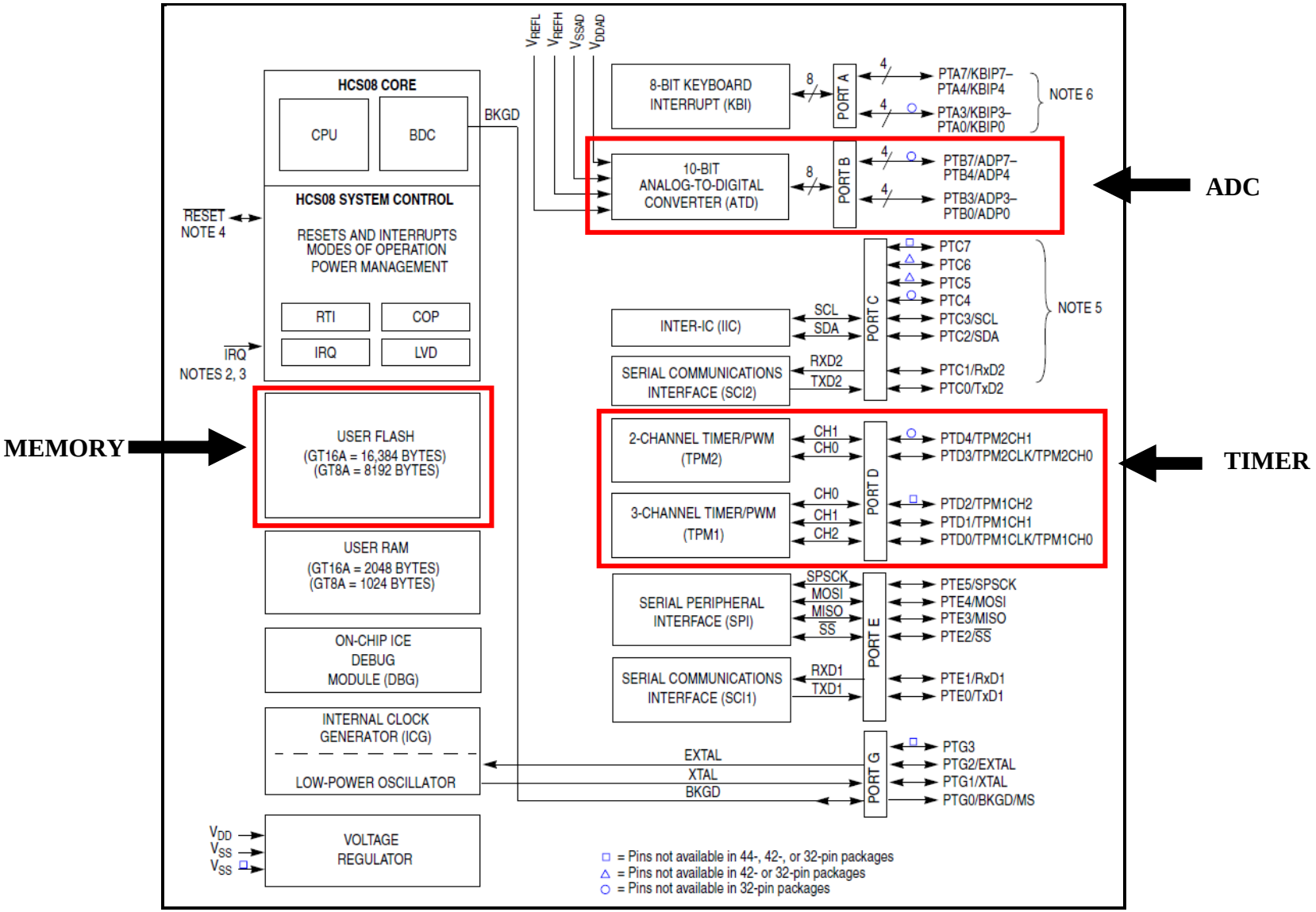


Figure 30: Showing the MC9S08GT16A Block Diagram [46]

A MC9S08GT16A micro-controller was used for programming the control algorithms of the 6 DOF platform for this project. The GT16A used in this project comes in a 44 pin package

4.2.3 Timers

The GT16A contains two independent timers/Pulse Width Modulation (PWM) or TPM modules which individually support the conventional input capture, output capture, or buffered edge aligned PWM. A control bit in each TPM configures all channels to operate as centre-aligned Pulse Width Modulation (CPWM) functions. The timing functions in each timer are based on a 16-bit counter with prescaler and modulo features to control the frequency and range (that is the period between overflows). The timing system of the GT16A is ideally suited for a wide range of control applications, and the centre-aligned PWM capability extends the field of applications to motor control. This makes the GT16A very applicable in this project. The timer system of the GT16A includes a 3-channel TPM1 and a separate 5-channel TPM2. The diagram below shows the detailed block diagram for the timer system in a GT16A microcontroller. **[46]**

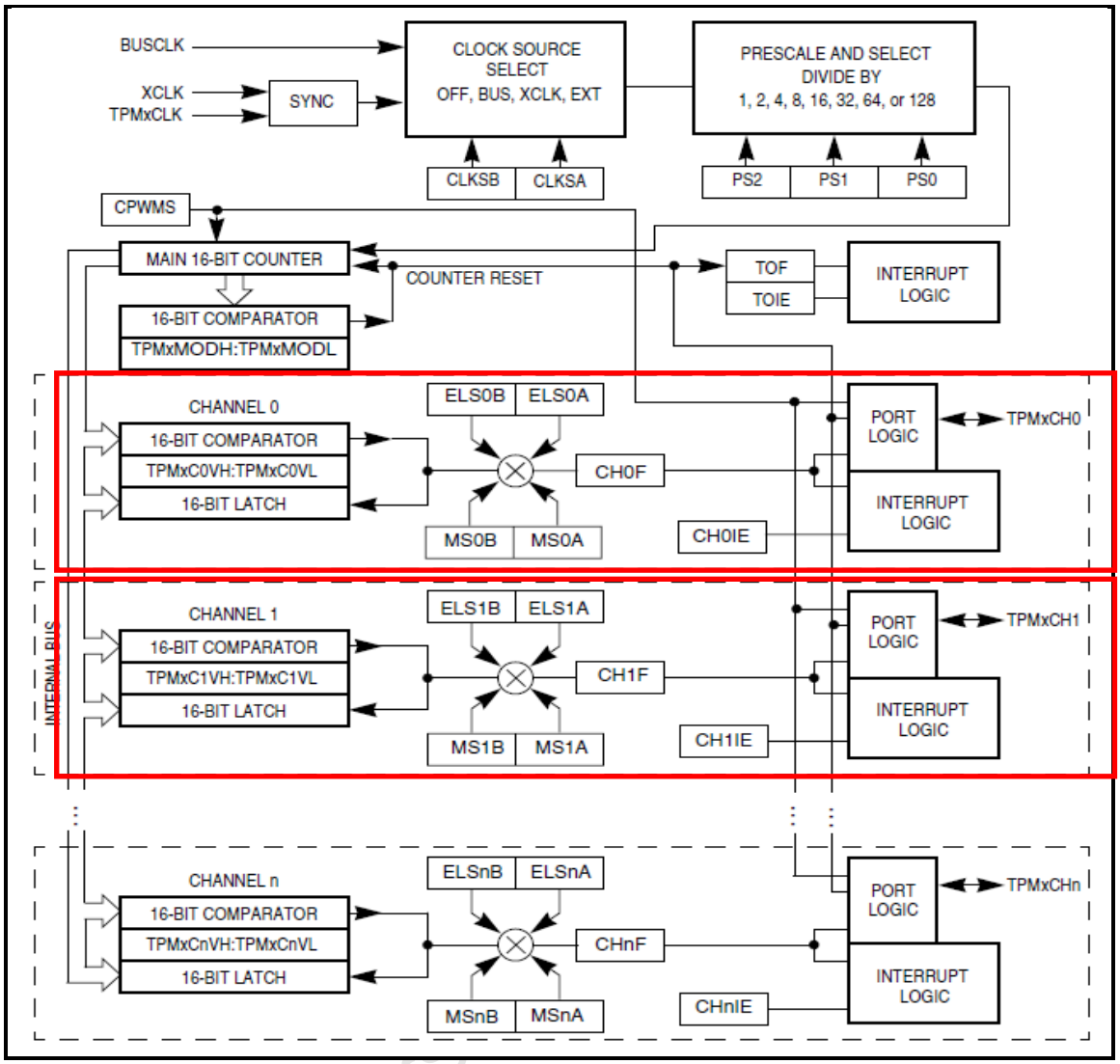


Figure 31: Showing the TPM Block diagram
The blocks show each Timer channel's entire block diagram **[46]**

The central component of the TPM is the 16-bit counter that can operate as a free-running counter, modulo counter or a general up/down counter when the TPM is configured for CPWM. The TPM counter provides the timing reference for the input capture, output capture, and edge aligned PWM functions. The timer counter modulo registers TPMxMOD controls the modulo value of the counter. Software can read the counter value at anytime without affecting the counting sequence. All TPM channels are programmable independently to one another as input capture, output capture or buffered edge aligned PWM channels. This is desirable for the project because six different servo motors are used as platform actuators. The servo will have to move in response to different commands from the software and force sensors on different axes. **[46]**

4.2.4 Analogue to Digital Convertor (ATD)

The GT16A contains one 8-channel analogue-to-digital (ATD) module. The eight ATD channels share port B. Each channel can independently and individually be configured for general-purpose I/O or for ATD functionality. The ATD module is an analogue-to-digital converter with a successive approximation register (SAR) design with sample and hold.

The GT16A also possesses a mode control unit that communicates with the sample and hold (S/H) machine to collect the samples and perform conversions. The operation happens the following way: The mode control unit signals the Sample and Hold (S/H) machine to begin collecting a sample and for the SAR machine (described above) to begin receiving a sample. At the end of each sample period, the S/H machine signals the SAR machine to begin the analogue-to-digital conversion process. The conversion process gets terminated when the SAR machine signals the end of conversion to the mode control unit. The mode control unit is the one that organizes the conversions; it also specifies the input sample channel, and moves the digital output data from the SAR register to the result register. **[46]**

Figure 32 below shows the block diagram of the ADT in the GT16A

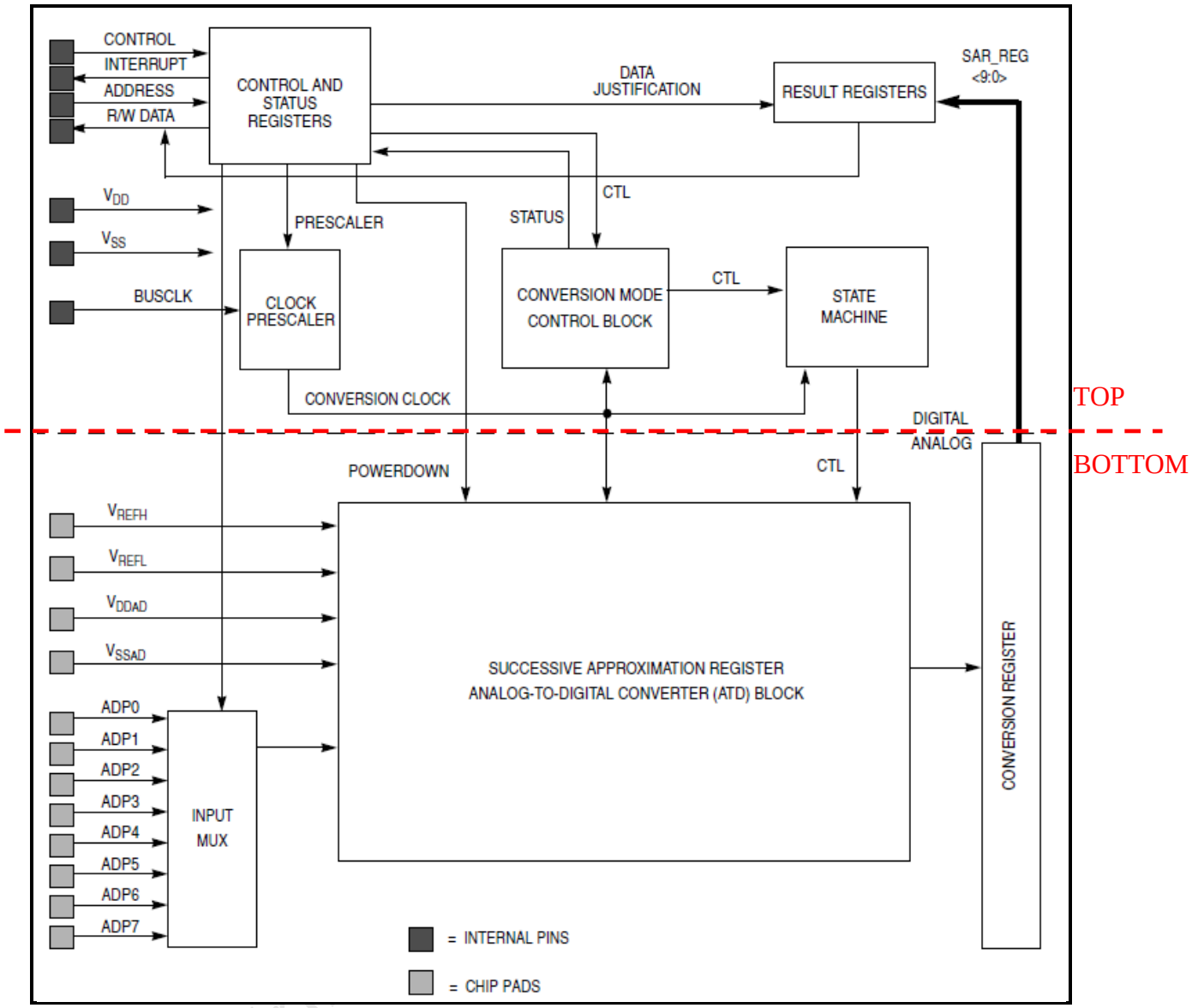


Figure 32: Showing the Block Diagram of the ATD in a GT16A [46]
The top part deals with digital signals and the bottom part with analogue signals.

4.2.4.1 Sampling and Storing

The S/H machine accepts analogue signals and stores them as capacitor charge on a storage node which is located in the SAR machine. At each time, only one sample can be held so the S/H machine and the SAR machine cannot run simultaneously even though they are independent machines. The figure below shows the placement of the various resistors and capacitors.

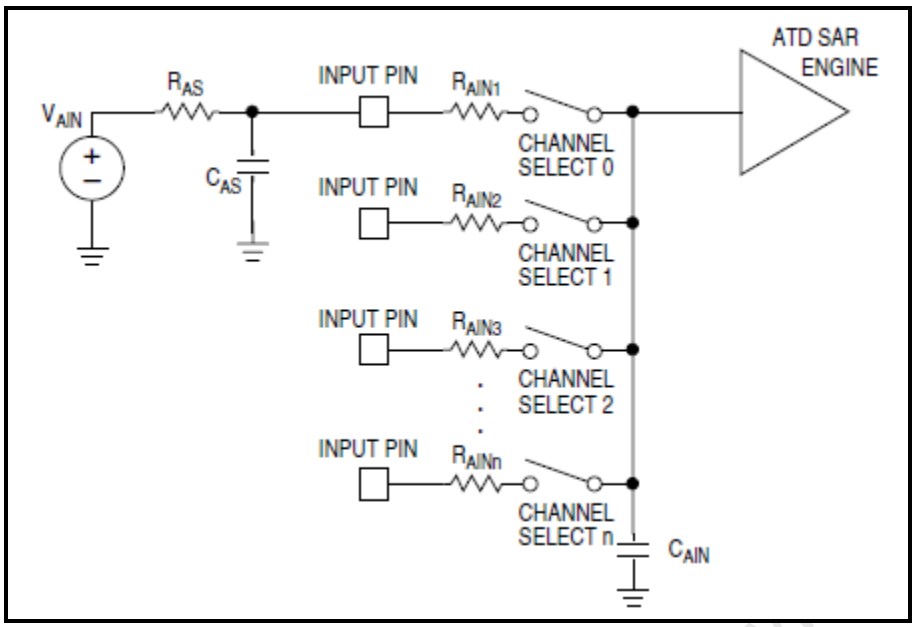


Figure 33: Resistor and Capacitor Placement [46]

4.3 Programming

The software was designed to be responsible for the entire control algorithm needed to make the platform perform to its requirements. This includes interfacing all the hardware modules, performing calculations, analysing the stored and collected data (from electronic hardware) and communicating between devices and actuators.

The major software to control the platform was written in C using the Freescale Code Warrior (C compiler Version 6.1). Matlab (version 7.6.0 (R2008a)) was also used to plot graphs of the relationship of different variables pertinent to the project. All functions and modules where written and tested separately. The outline of the testing is described later in this section. Furthermore, Eagle Cad (Layout Editor Version 5.9.0) was used to design the circuit schematic and circuit board schematic.

The software was designed to receive data from the sensors through the ADC, analyse it and calculate the equivalent force relative to the signal from the sensors and also to move the platform to a new position depending on the amount of force the sensors measured. This outline is shown below.

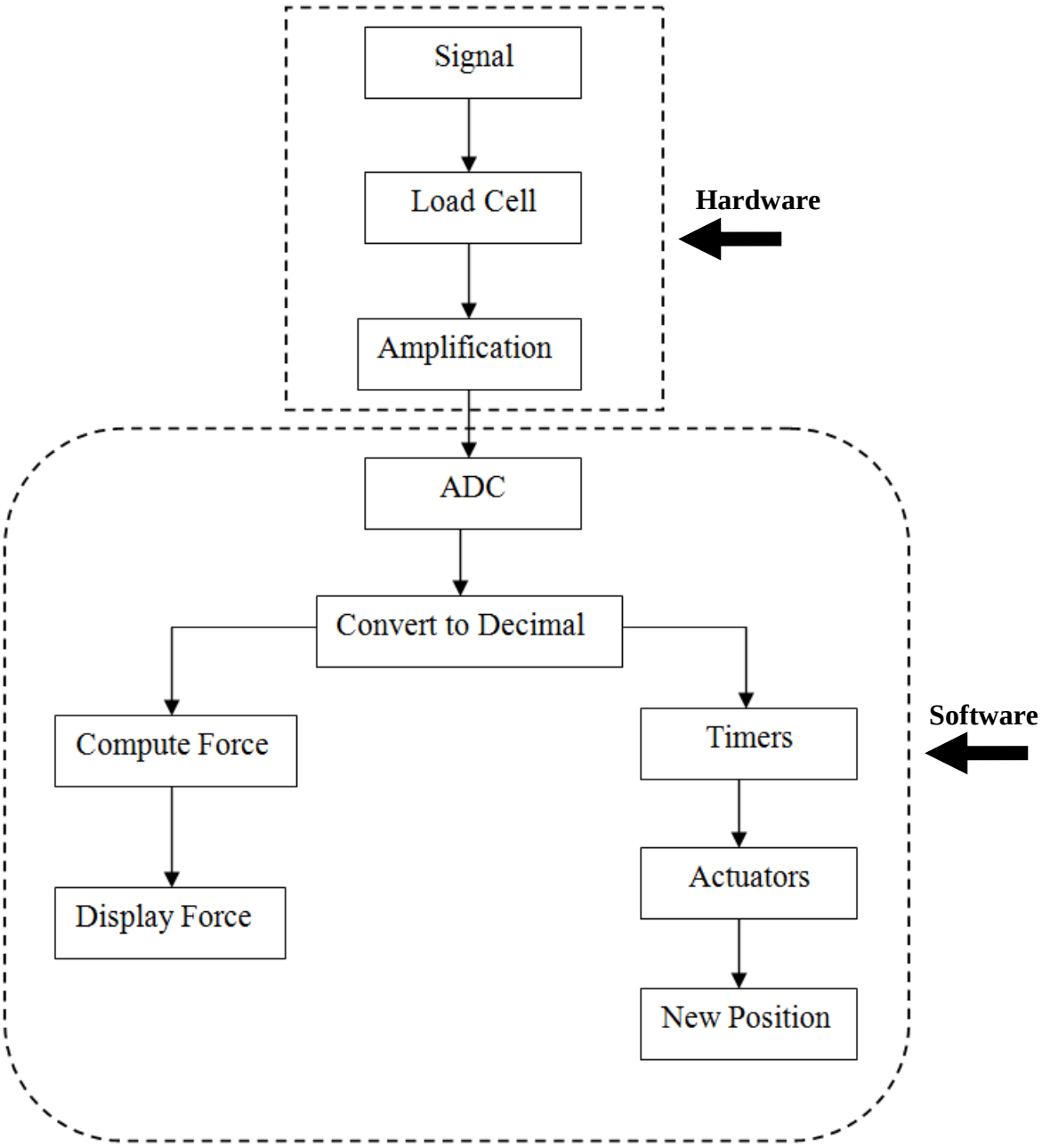


Figure 34: Showing the hardware and software system design integration

4.3.1 Force Computation

When computing the force, the platform was at its static position and thus the timers were in charge of the dynamics of the platform. Only after the platform had moved to a new position was the new force measured again. The general system diagram of the Main Function for computing the forces and moments is shown below. The main software program consisted of different parts; the set up functions, the ADC conversion function, the force computation function, the BCD and ASCII conversion function and lastly the display function.

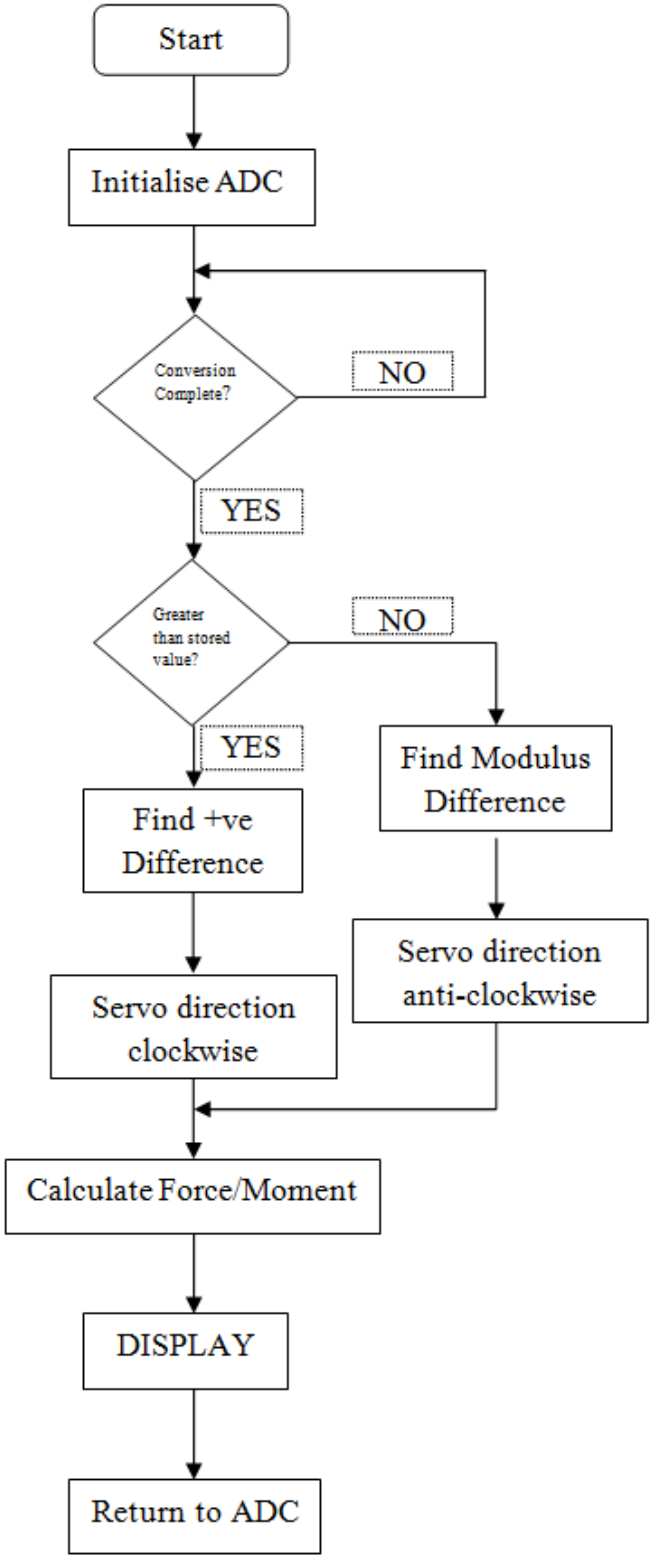


Figure 35: System Diagram of the Main Force Computation Function

4.3.2 Position Computation

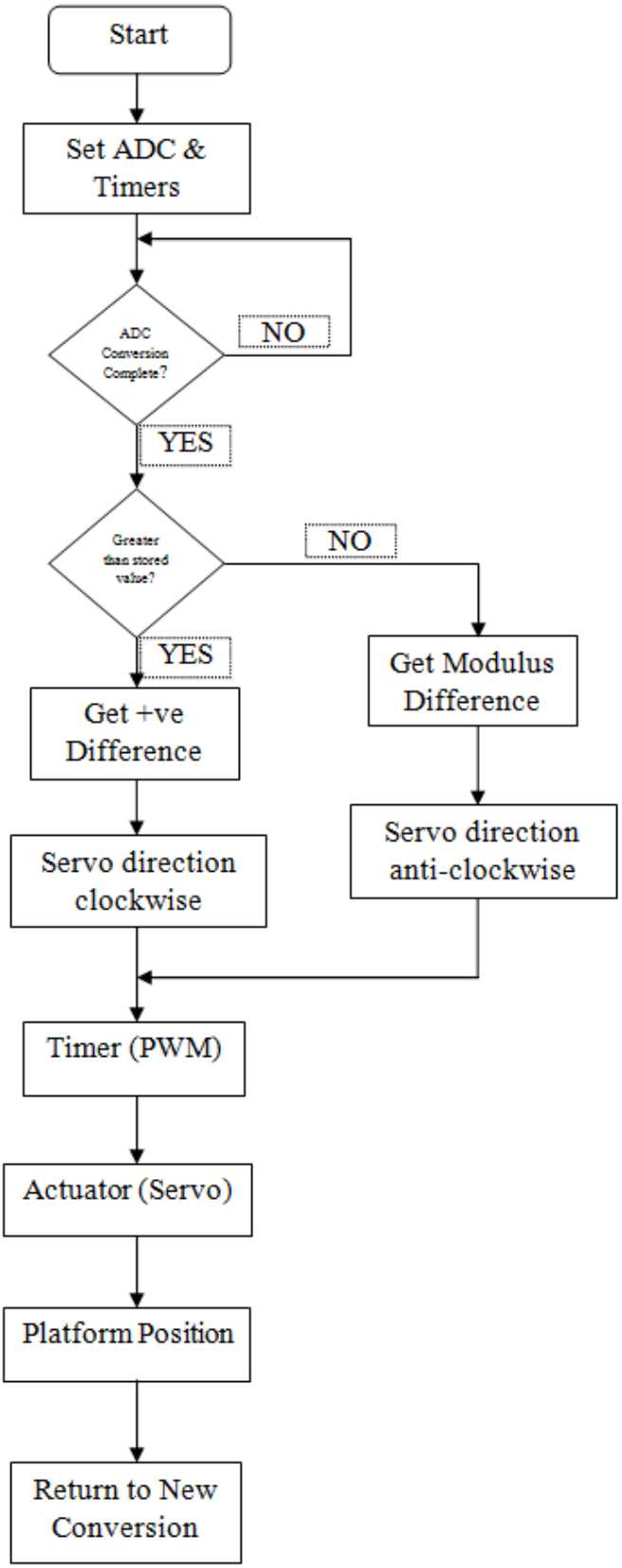


Figure 36: System Diagram of the Main Actuator control Function

4.3.3 Include files

The main software used in the control of the platform included five header files that were used to define common parameters used throughout the program. These included the MC9S08GT16A, hidef, string, LCD and Math which are explained in the appendix.

4.3.4 Actuator Control

Servo motors need a PWM input to them to provide instructions to the way they should operate by sending a pulse width modulation or PWM signal to the PWM input pin. PWM is a modulation technique whereby the width of a pulse over the period of a signal, known as the duty cycle, is varied to represent the amplitude of an analogue signal, in our case from the ADC. The PWM signal in this project was part of the software design. The PWM signal is controlled by the values that are put in the Modulo Registers and Prescaler. The servo compares that signal to the actual position of the servo and adjusts the servo accordingly. The internal circuitry of the servo expects a constant 50Hz PWM signal (a 50 Hz signal is one that repeats every 20 ms). The duration of the high signal determines the position that the servo attempts to maintain. It should be noted that the servo must continually receive this signal in order to maintain its position. Though it needs a full 20ms cycle, only the first 10th of the period carries all the instructions and scales as follows:

- 1.0ms = full left = 0^0
- 1.5ms = middle = 90^0
- 2.0ms = full right = 180^0

Anything between the above times sets the angle in which it must move.

4.3.5 PWM

The values therefore of the Modulo Registers and Prescaler are determined in the following manner.

$$P_S = \frac{df_B}{2^{16}}$$

Where P_S is the prescaler, d is the intended delay; f_B is the GT16A bus frequency (18.874368MHz). The result should be rounded up to the nearest power of two. If the result is bigger than 128, then it means the time delay is too long and thus will need multiple timer

periods. The rounded number (P_{SR}) becomes the prescaler value. The Modulo Register value (M_V) thus becomes:

$$M_V = \frac{df_B}{P_{SR}}$$

[47]

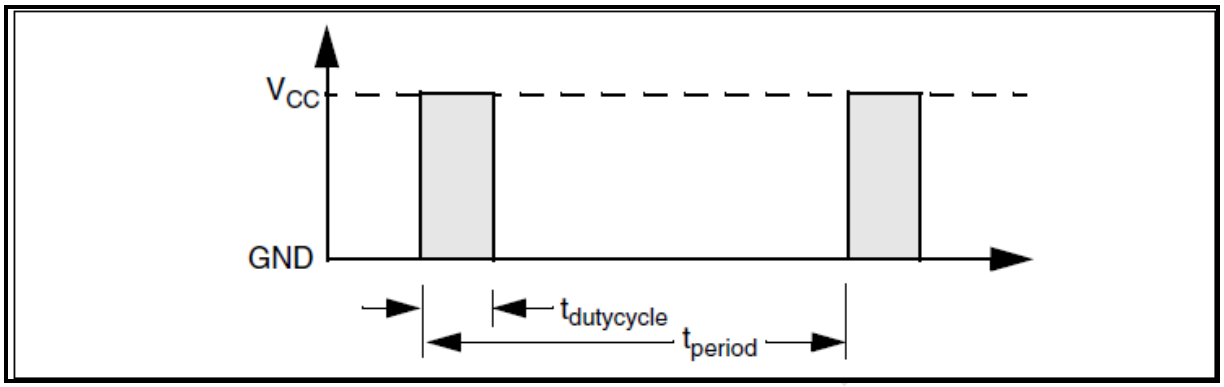


Figure 37: Showing a PWM signal

Duty Cycle is expressed as the percentage of time the signal is high for the given period.

duty cycle = 100 x ($t_{\text{dutycycle}}$) / (t_{period})

For example, a 10% duty cycle applied to a 3Vcc signal approximates an analogue voltage of 0.3V once this signal is averaged by an integrator/filter circuit.

CHAPTER FIVE

TESTING

5.1 Software Testing

5.1.1 Liquid Crystal Display (LCD)

The LCD was first tested using the demo code provided on the GT16A kit to make sure that every part of it works. It then was tested by running a program written by my supervisor and already known to work because it had been used in lecture lessons.

5.1.2 Analogue to Digital Converter (ADC)

The ADC was tested by connecting it to the Light Emitting Diodes (Port A) and as it was incremented the number would be depicted by the LED. There were 8 LEDs and with Port A7 being the most significant bit port and A0 being the least significant bit. Since an 8-bit ADC mode was used, the highest number that would be obtained was 2^8 and that is 256 meaning it is from 0 – 255. When the ADC input is 0V, no LED will be on and if it is above 2.7V they will all be on. The numbers were also displayed on the LCD, so as to make the testing easy.

5.1.3 Timers

The timers were also tested manually using an oscilloscope (Agilent Technologies, DSO3062A digital storage oscilloscope). The PWM was measured and it had to have a length of 20ms for it to be able to control a servo motor. The duty cycle was monitored by connecting the potentiometer output to the timers. As the potentiometer was varied the duty cycle had to also change.

5.2 Hardware Testing

5.2.1 Sensor Calibration

The calibration of the force sensor for the platform was performed gravimetrically and using a well calibrated force balance instrument. The correlation between the forces and sensor voltage output was obtained by hanging a series of weights on a point directly in line with the end of the sensor. This was to make sure that a force in line with the axis is measured. Each sensor on the different axes (rotational and translational) was calibrated independently, as follows:

5.2.2. Translational Plates

The sensor was oriented horizontally and a force balance was mounted to provide a constant force to act carefully on the force sensor. The sensor was positioned with its base parallel with the horizontal plane to ensure that only the deflection corresponding to the force in that axis direction was measured. For each trial, a baseline was measured and subtracted.

The force balance was then set to a different force, and the sensor/force system was allowed to come to rest. The difference between baseline voltage and the new voltage with the weight present was then calculated, tabulated in a Microsoft excel table and saved. This was done four times for each force on the force balance, and for each axis. The forces used range from 0 – 3 N. During experiments, the force sensor position was set exactly at the position in which it was when the experiment started. This was done by powering the servo motor (actuator) and setting it to a fixed position for the duration of the entire experiment. Furthermore, the force balance was kept parallel to the axis in question for every force change that was made. This ensured that the sensor output voltage was calibrated to provide an accurate measurement of forces in that axis. This was done for both the compressive and tensile forces independently on the sensor and the results were separated in the table.

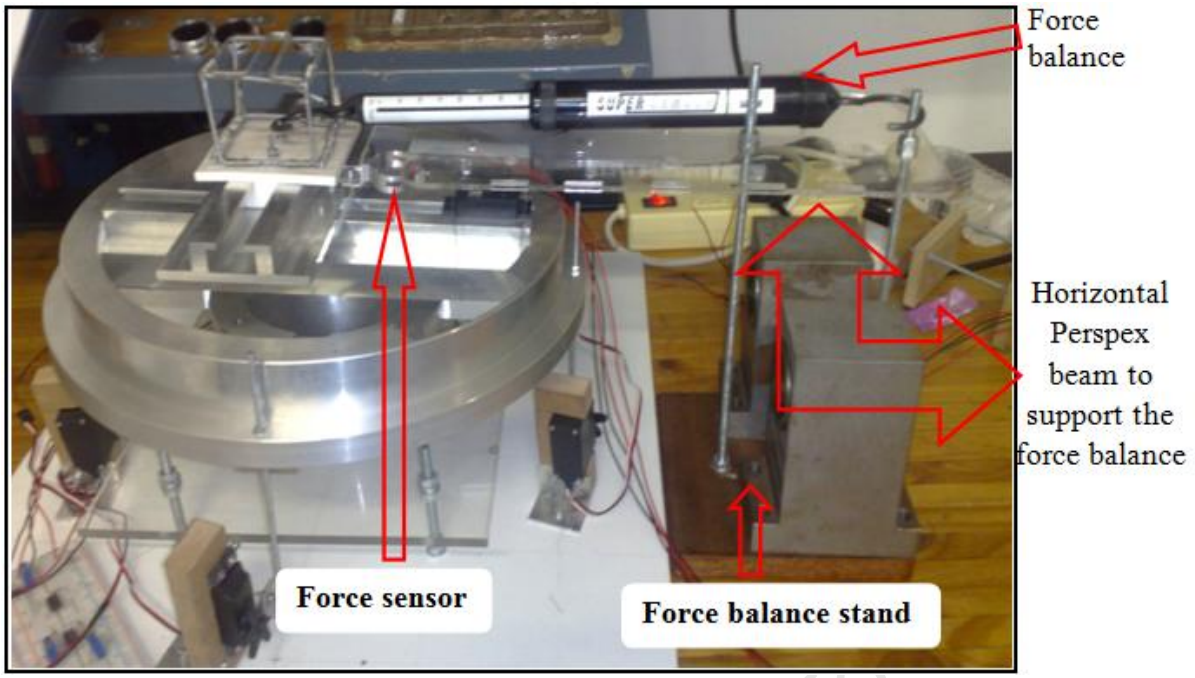


Figure 38: Showing the testing method on the translational plates whereby a horizontal perspex beam is used to support the force balance which is attached to a sliding plate, and in parallel with a force sensor, to provide a measurable force for testing and calibrating the sensors.

5.2.3 Rotational Plates

A very light plastic tube was mounted vertically and directly in-line with the force sensor oriented vertically to use gravity as a constant force to act on carefully weighed masses. The vertical position of the sensor parallel was to ensure that only that intended axis measured the deflection corresponding to the weight. For each trial, a baseline was measured and subtracted. A weight was then added in the plastic tube, and the sensor/weight system was allowed to come to rest. The difference between baseline voltage and the new voltage with the weight present was then calculated, tabulated in a table and saved, as was done for the translational experiments. This was done four times for each weight, and for each axis. On each axis, both the compressive and tensile forces were applied on the sensor and the results tabulated separately. The weights used were between 15 and 25 small ball weights of 12g each. This however, gives out a range of force from 0-3N. During experiments, the position of the sensor was at exactly the location where the weights had been hung. This ensured that the sensor output voltage was calibrated to provide an accurate measurement of the forces. This is shown in the figure below:

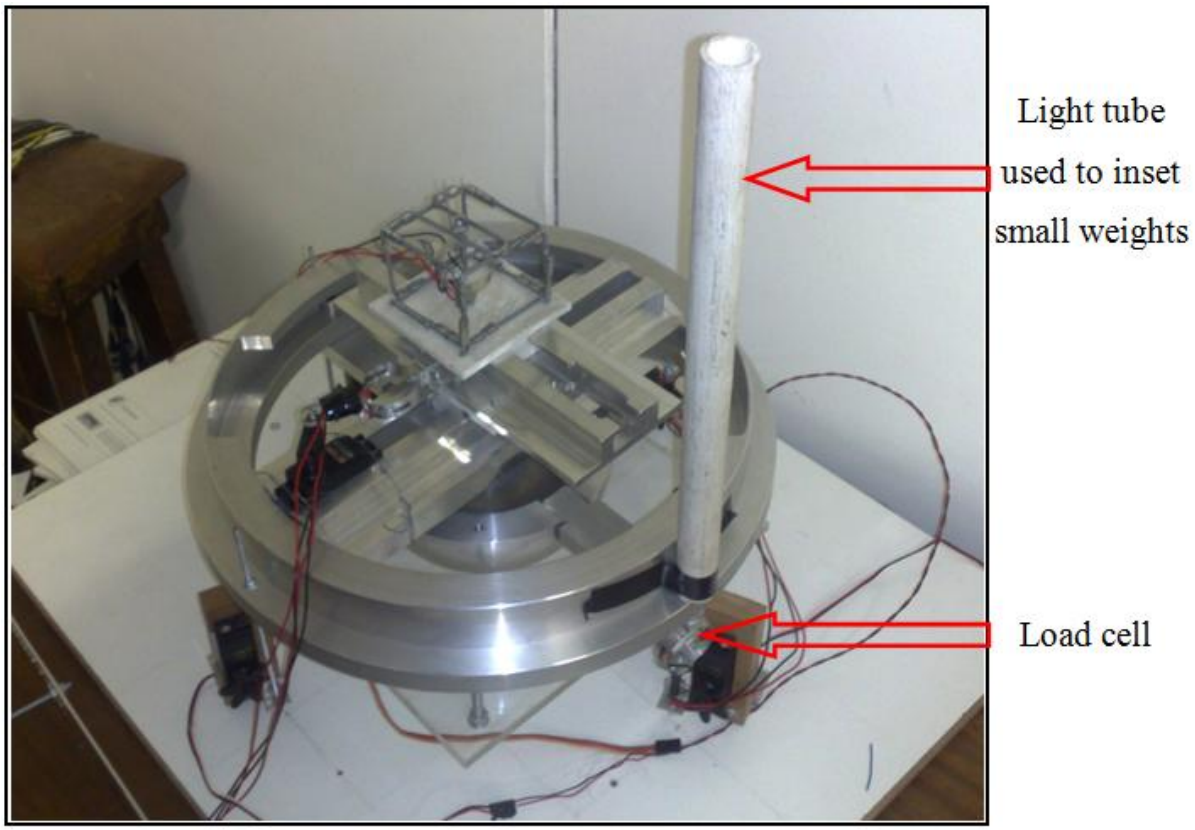


Figure 39: Showing the testing method on the rotational plates where a light tube is mounted vertically in line with a load cell and small weights of known mass inserted in the tube to make sure all the force is directly affecting the load cell for ease of testing and calibrating.

5.2.4 Data tabulation

A table of results of each experiment was created and the parameters of the experiment were recorded carefully. The mass at each increment was recorded separately until about between 15 and 25 different masses were added and measured. Four different voltage equivalent to the masses where recorded. In the same table, the moments created by the force on the rotational axis were recorded accordingly. In addition, the cumulative voltages equivalent to the cumulative mass was recorded and an expected ADC-value was recorded. The sensitivity of each sensor response to an added mass was recorded in Volts/Newton.

CHAPTER SIX

RESULTS

6.1 Software

The software parts were all tested separately as described in the testing section above.

6.1.1 Liquid Crystal Display

The LCD worked correctly.

6.1.2 Analogue to Digital Converter

The ADC worked correctly.

6.1.3 Timers

The results from measuring the output from the timers on a Cathode Ray Oscilloscope are shown below.

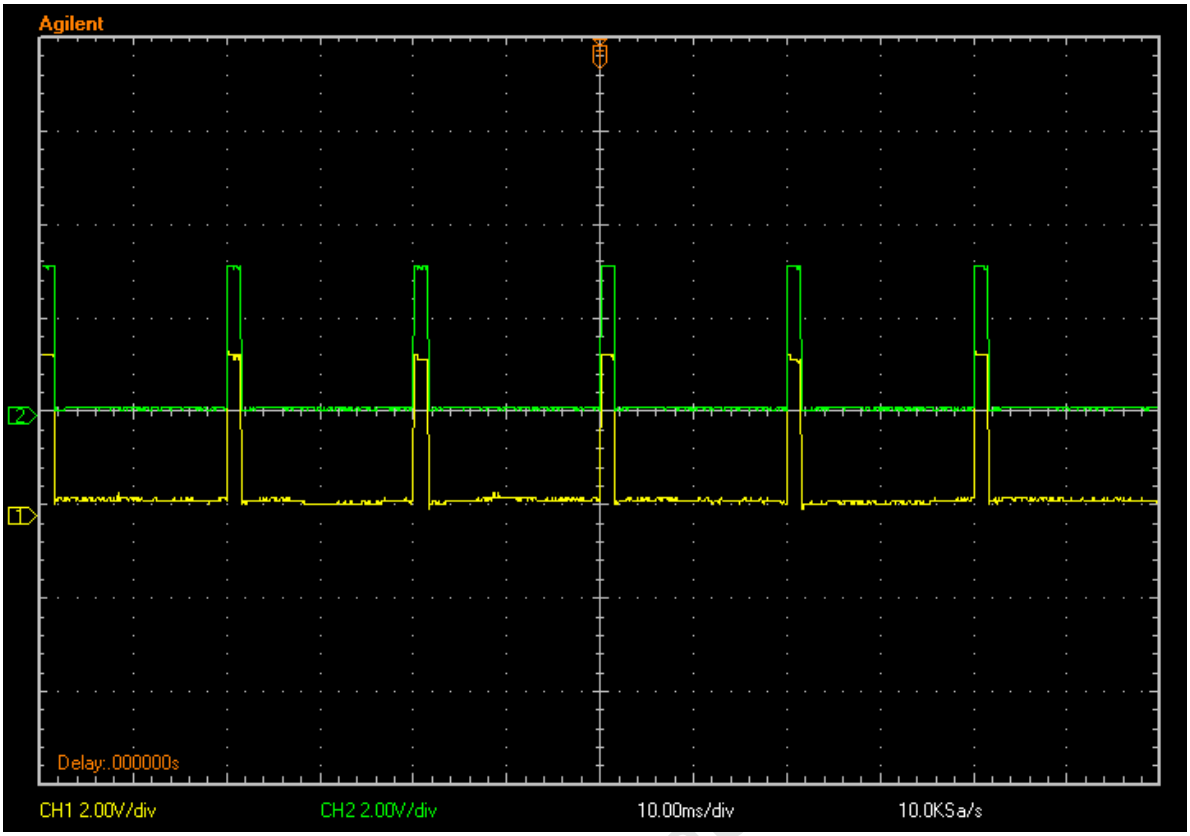


Figure 40: Showing the PWM output from an Oscilloscope, where 1 and 2 represent timer channels 1 and 2 measurements from the port D pins used to control the actuators.

In the figure above, two synchronous wave forms are shown. These were measured from two different timer channels from the GT16A. They both had a maximum voltage of 3V and a 10% duty cycle that will make the servo turn 180°.

6.2 Hardware

All the hardware testing and naming was done based on the orientation of the robotic flying bat given in figure 1 in the literature review.

6.2.1 Pitch- Forward Rotation

This is when the system rotates clockwise around the X axis. In the system, this happens when the force is exerted on the load cell directly in line with the Y plate, resulting in the rotation with the x-axis as the center of rotation as shown in figure 41 below.

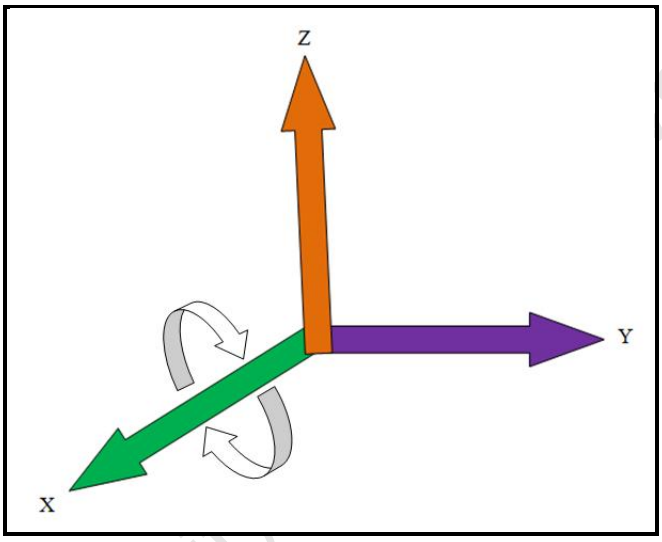


Figure 41: Showing the Pitch forward rotation that normally happens when the flying object is twisted in the in the nose-down sense.

The relationship between the forces and moments versus the voltage produced by the force transducers is shown below. In the graphs below the residual is also shown. Residuals are differences between the one-step-predicted output from the model and the measured output from the validation data set. Thus, residuals represent the portion of the validation data not explained by the model.

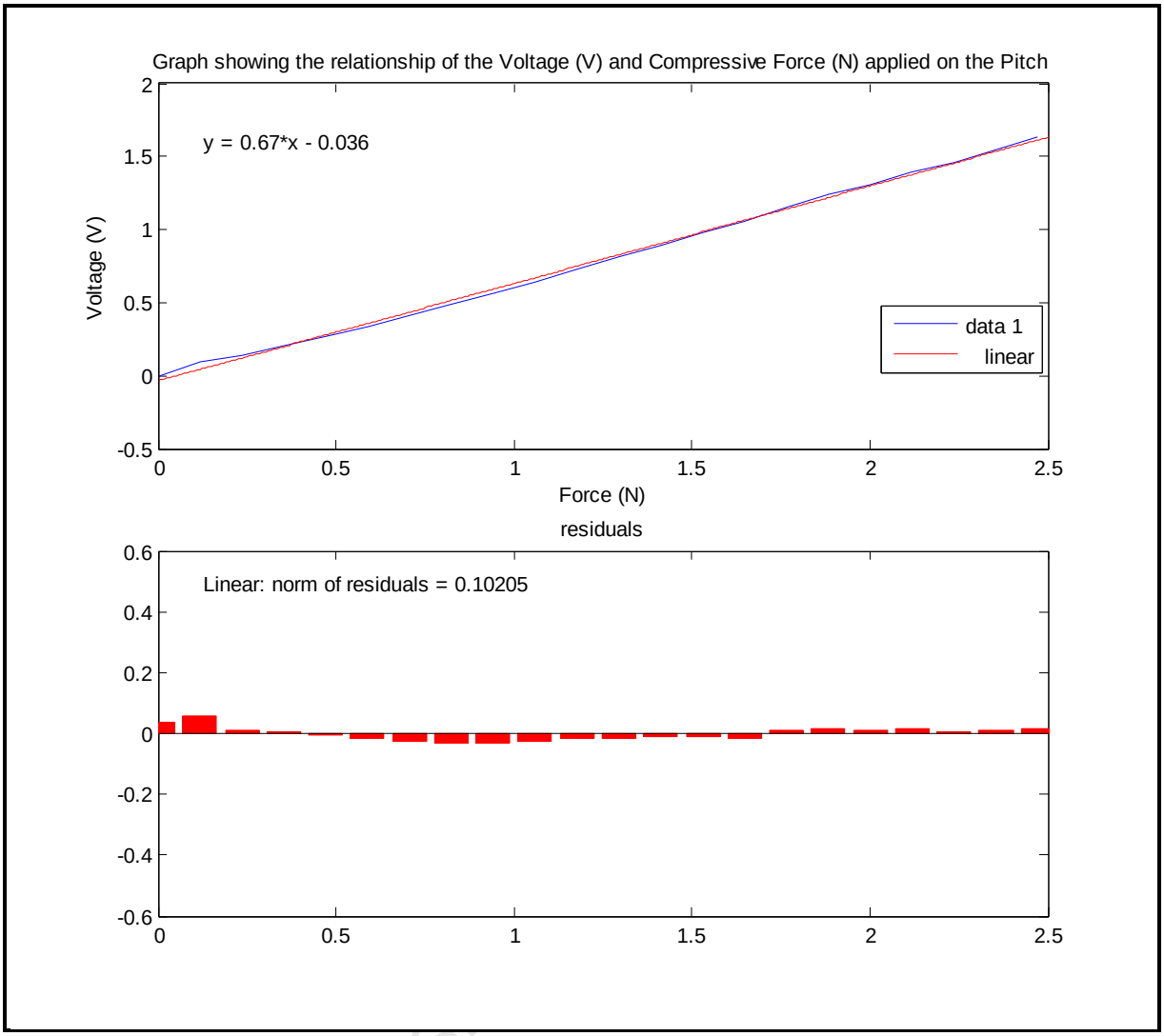


Figure 42: Showing the relationship between the force and the voltage output and the residuals of the linear best fit line

The relationship shown in figure 42 above can be best approximated by a best fit line which shows it to be linear (or proportional). Table 6 below describes the characteristics of the relationship, where σy and σx are the standard deviations of y (voltage) and x (force) respectively.

Table 6: Showing the relationship between Voltage and Force for the forward pitch movement

Relationship [V/N]	σy [V]	σx [N]	Residual
$y = 0.66701x - 0.035844$	0.5104	0.7644	0.10205

The moment relationship is also a linear graph with an algebraic expression given by:

$$y = 4.4468x - 0.035857 \text{ [V/Nm]}$$

The figure below shows the moment relationship of the force applied and the voltage.

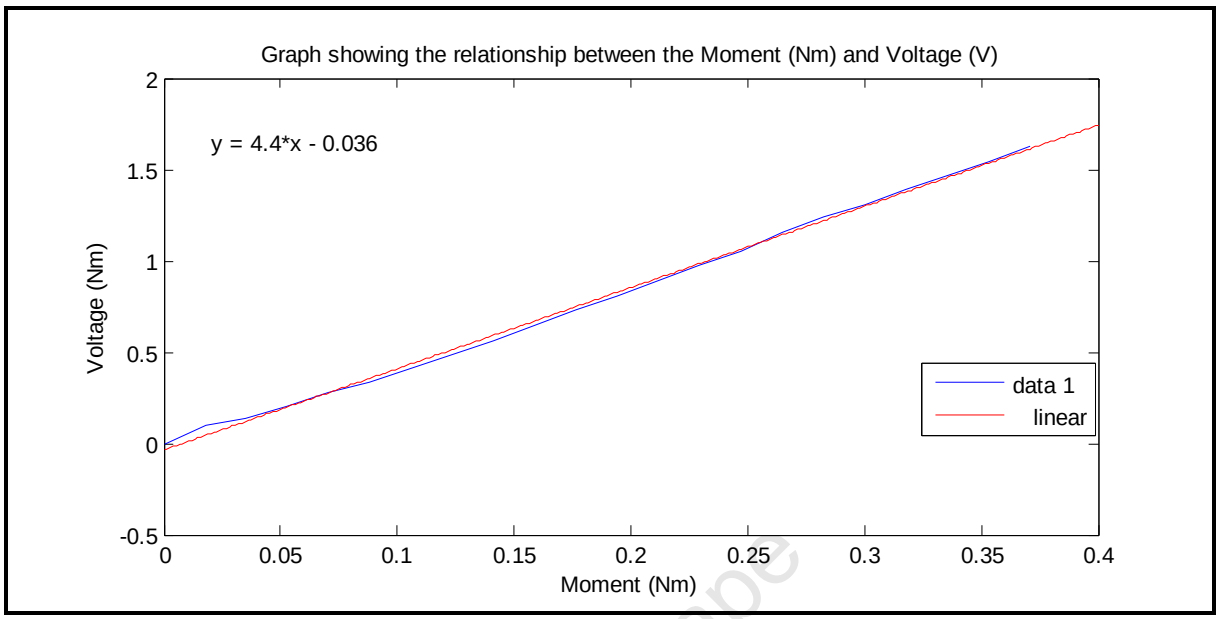


Figure 43: Showing the Moment and Voltage output relationship

6.2.2 Pitch- Backward Rotation

This is when the system rotates anti-clockwise around the X axis as shown figure 44 below. This is the same center of rotation but opposite in direction to the one described above.

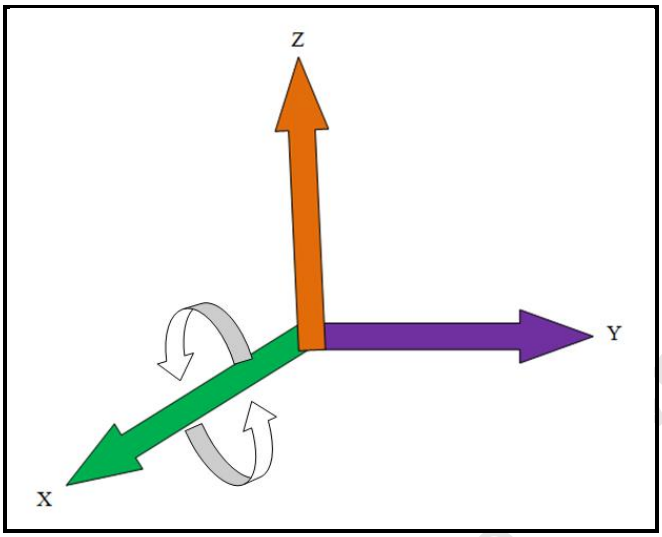


Figure 44: Showing the Pitch backward rotation that normally happens when the flying object is twisted in the in the nose-up sense.

The relationship between the forces and moments versus the voltage obtained from the amplification circuit is shown in figure 45 below. The relationship can be best approximated by a best fit line which is a quadratic function. This (quadratic function) is a function given by a polynomial with the variable raised to highest degree (power) of 2 as seen in the equation below. Table 7 below describes the characteristics of the relationship, where σy and σx are the standard deviations of y (voltage) and x (force) respectively.

Table 7: Showing the relationship between Voltage and Force for the backward pitch movement

Relationship [V/N]	σy [V]	σx [N]	Residual
$y = -0.11371x^2 + 0.58561x + 0.04575$	0.2392	0.7304	0.10106

The moment relationship is also a quadratic graph with an algebraic expression given by:

$$y = -5.0544x^2 + 3.9044x + 0.045725 \text{ [V/Nm]}$$

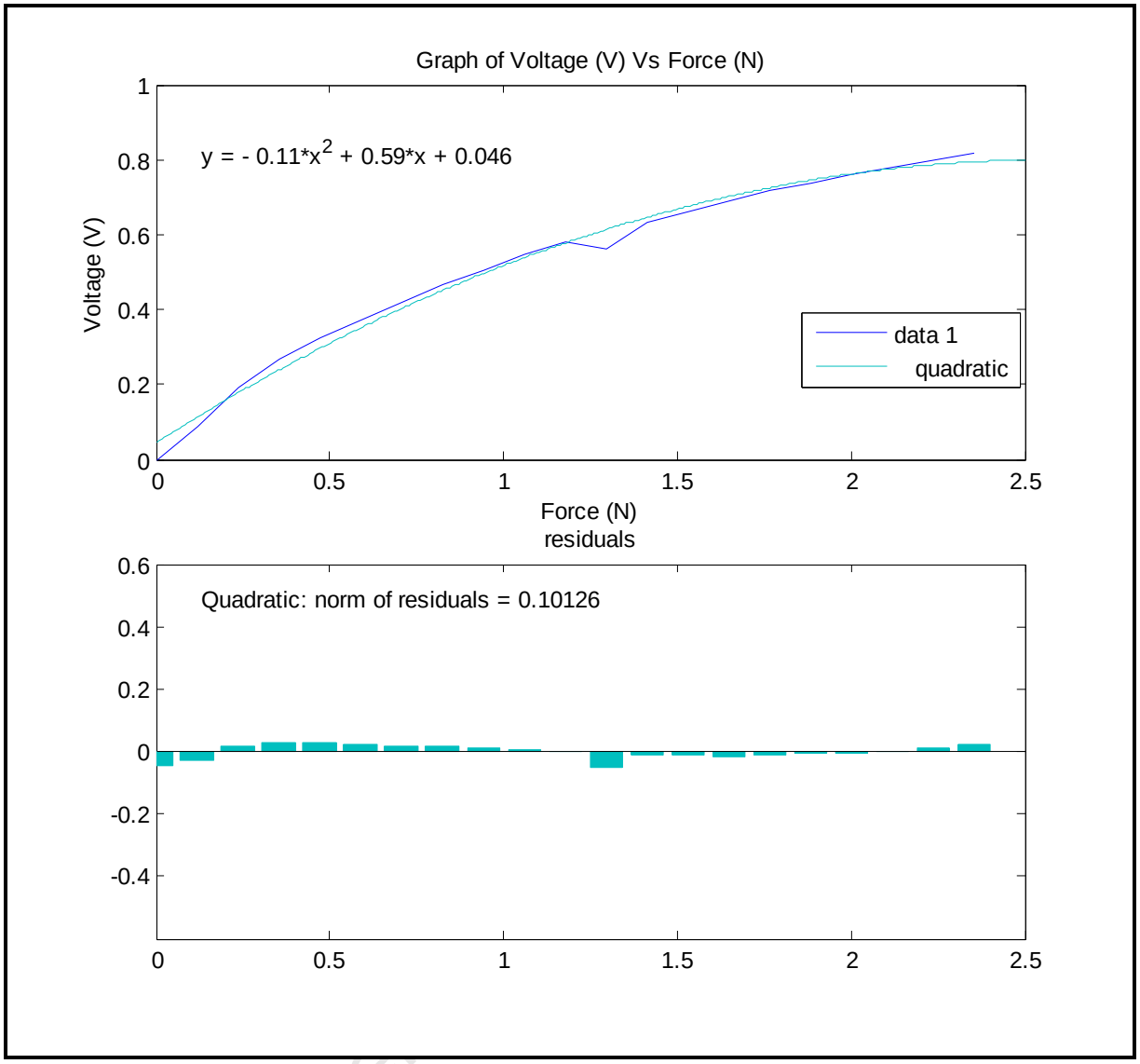


Figure 45: Showing the relationship between the force and the Voltage output and the residuals of the quadratic best fit line for the backward pitch movement

The moment relationship is shown in the appendix

6.2.3 Roll- Forward Rotation

This is when the system rotates anti-clockwise around the Y axis as shown in figure 46 below.

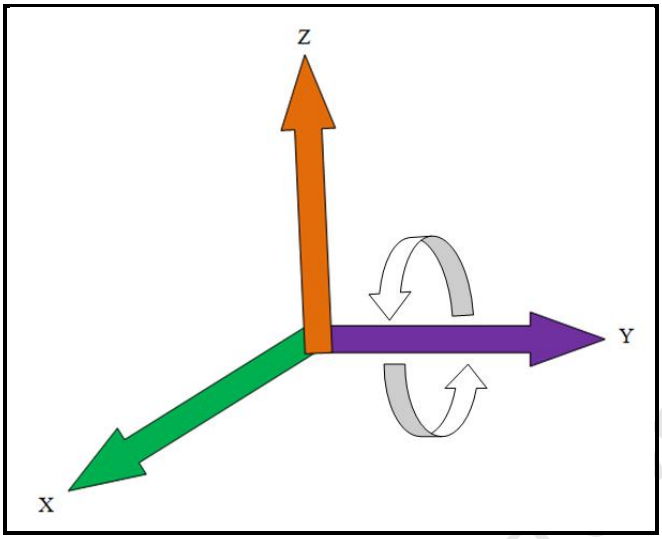


Figure 46: Showing the Raw forward rotation that normally happens when the flying object is twisted sideways with the right wing lower than the left wing and normally precedes a turn to the right.

The relationship between the forces and moments versus the voltage output from the force transducers is shown below.

Table 8: Showing the relationship between Voltage and Force for the forward roll movement

Relationship [V/N]	σy [V]	σx [N]	Residual
$y = 0.60405x - 0.053394$	0.4832	0.7984	0.13851

The moment relationship is also a linear graph with an algebraic expression given by:

$$y = 0.2474x + 0.013935 \text{ [V/Nm]}$$

The actual figures of the relationships shown above are found in the appendix.

6.2.4 X-axis Force (Tensile)

This is when the load cell mounted on the X plate to measure the forces in that axis is under tensional force (stretched), as shown in the figure below.

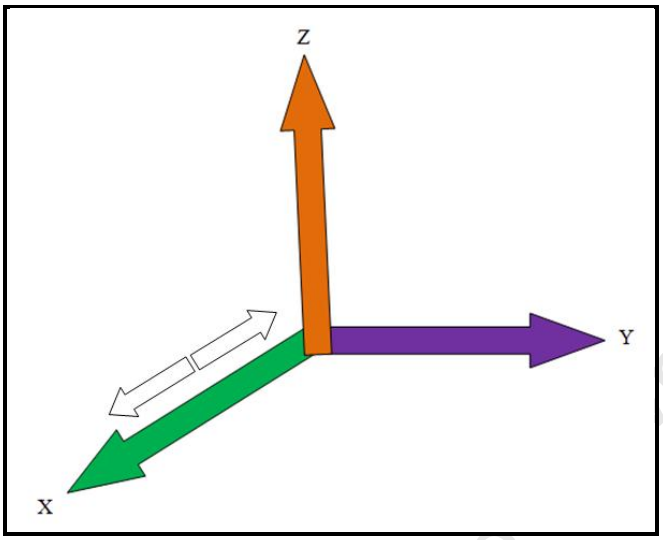


Figure 47: Showing the tensile forces acting on the load cell mounted on the X plate and this happens when the flying object moves sideways to the left linearly.

After testing the translational axis as described in the testing section the following results were the outcome. The relationship between the tensile forces versus the voltage output from the force transducers is shown in figure 48 below. The relationship can be best approximated by a best fit line which is a cubic function. This (cubic function) is a function given by a polynomial with the variable raised to highest degree (power) of 3 as seen in the equation below. Table 9 below describes the characteristics of the relationship, where σy and σx are the standard deviations of y (voltage) and x (force) respectively.

Table 9: Shows the relationship between the voltage and the tensile forces acting on the X plane

Relationship [V/N]	σy [V]	σx [N]	Residual
$y = 0.031687x^3 - 0.13078x^2 + 0.13634x - 0.0098829$	0.4441	1.427	0.11454

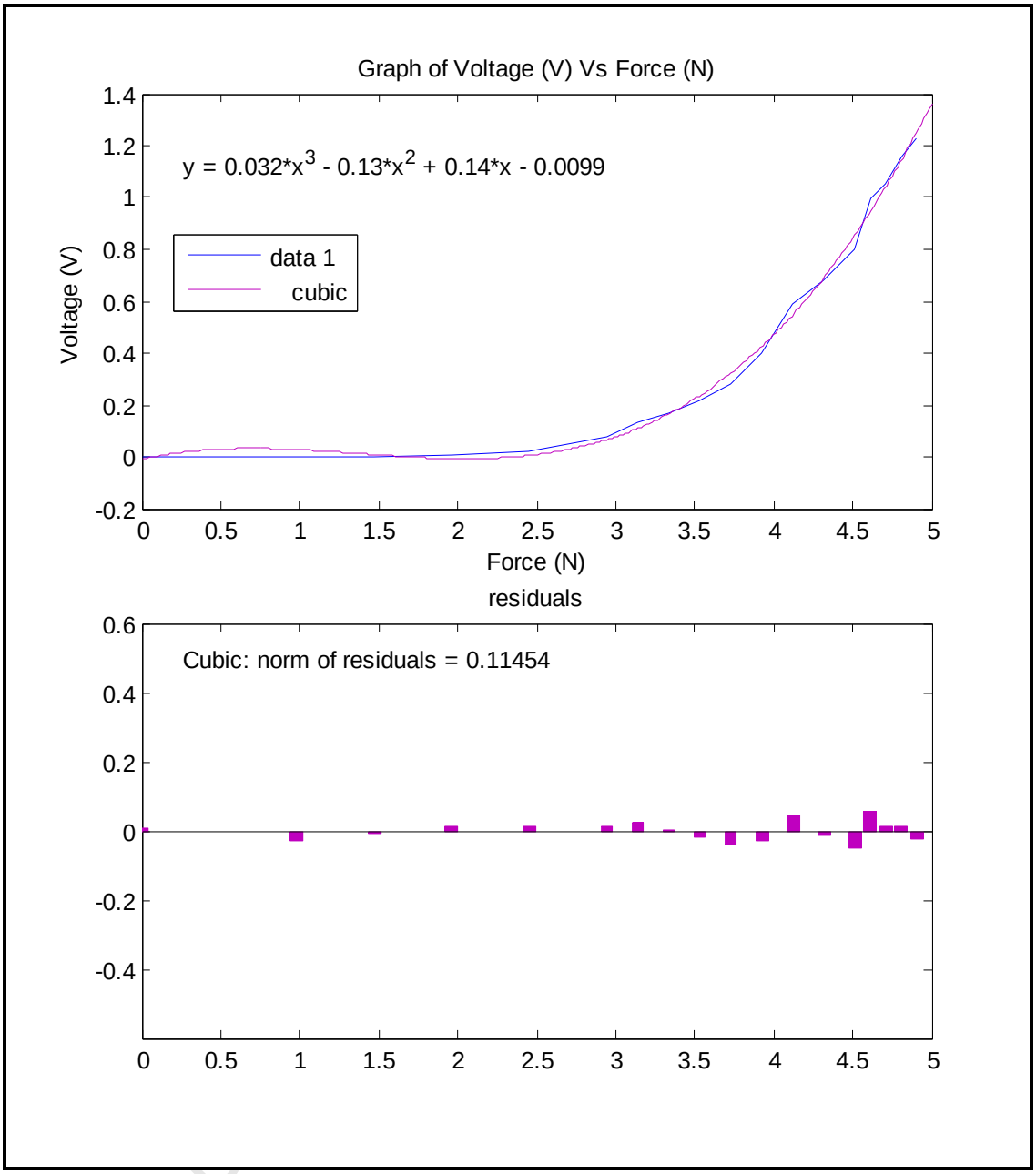


Figure 48: Showing the relationship between the force and the Voltage output

6.2.5 X-axis Force (Compressive)

This is when the load cell mounted on the X plate to measure the forces in that axis is under compression force, as shown in the figure below.

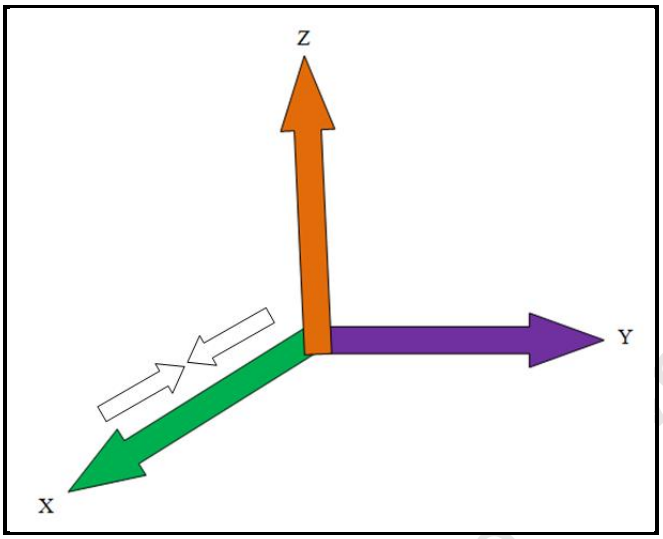


Figure 49: Showing the compression forces acting on the load cell mounted on the X plate and this happens when the flying object moves sideways to the right linearly.

The relationship between the Compressive forces versus the voltage output from the force transducers is shown below. The relationship can be best approximated by a best fit line which shows it to be a quadratic function. Table 10 below describes the characteristics of the relationship, where σy and σx are the standard deviations of y (voltage) and x (force) respectively.

Table 10: Shows the relationship between the voltage and the Compressive forces acting on the X plane

Relationship [V/N]	σy [V]	σx [N]	Residual
$y = 0.091809x^2 + 0.049085x - 0.078959$	0.3741	0.9069	0.11422

The actual graphic relationship described above is shown in the appendix.

6.2.6 Y-axis Force (Tensile)

This is when the load cell mounted on the Y plate to measure the forces in that axis is under tensional force (stretched), as shown in the figure below.

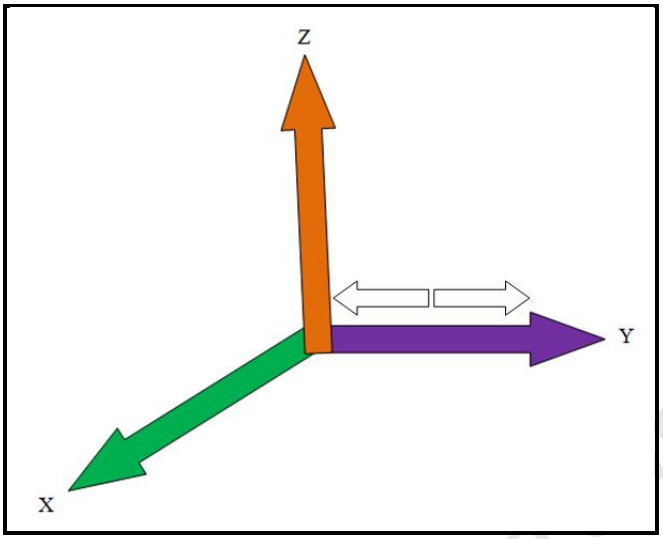


Figure 50: Showing the tensile forces acting on the load cell mounted on the Y plate and happens when the flying object is moving forward in a linearly.

The relationship between the tensile forces versus the voltage output from the force transducers is shown in figure 51 below. The relationship can be best approximated by a best fit line which shows it to be 4th degree polynomial. This is a function or polynomial with the variable raised to highest degree (power) of 4 as seen in the equation below.

Table 11: Shows the relationship between the voltage and the tensile forces acting on the Y plane

Relationship [V/N]	σy [V]	σx [N]	Residual
$y = -0.41076x^4 + 3.1353x^3 - 8.1996x^2 + 9.2334x - 3.7508$	0.4945	0.5746	0.12891

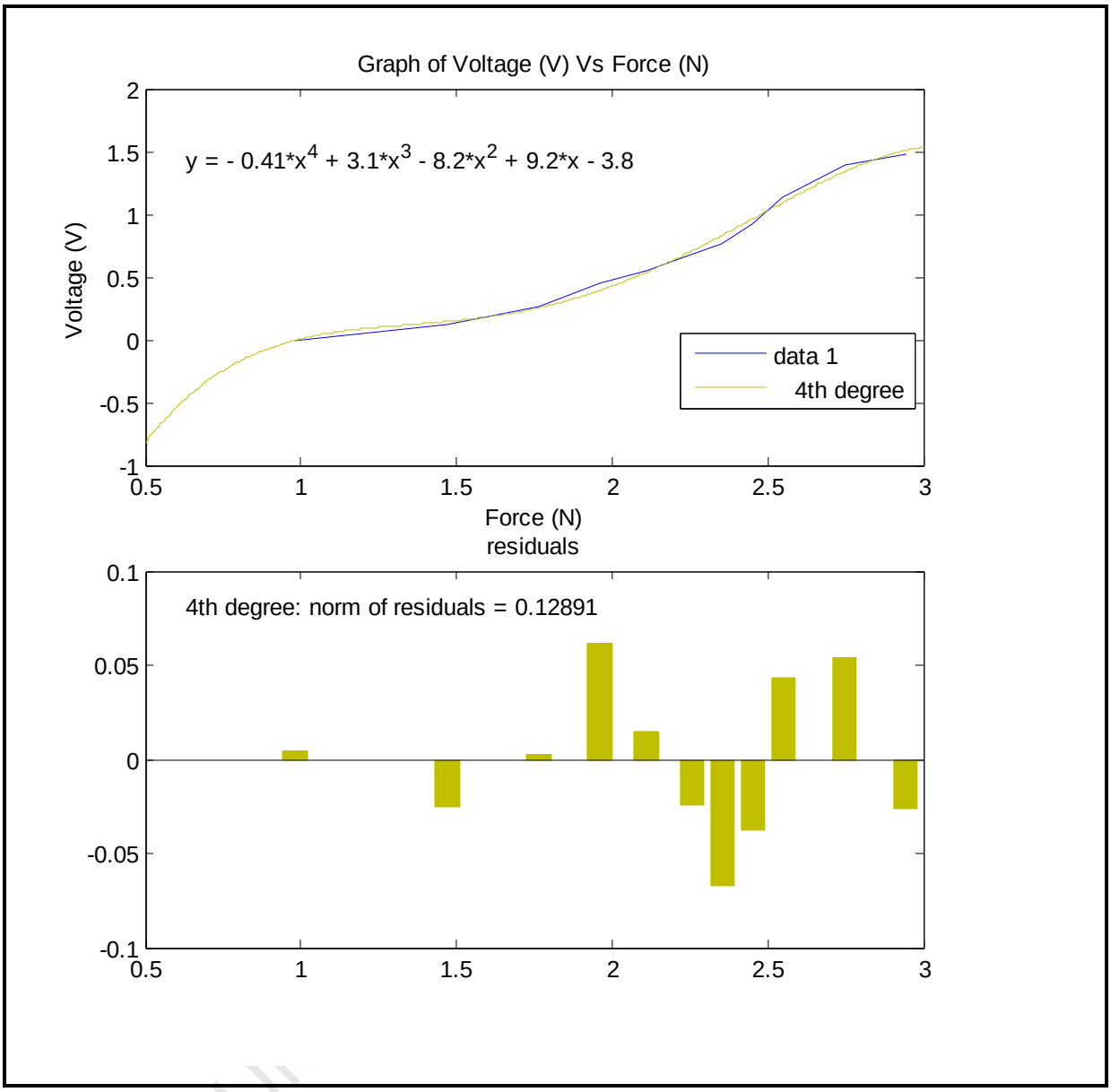


Figure 51: Showing the relationship between the force and the Voltage output

6.2.7 Y-axis Force (Compressive)

This is when the load cell mounted on the X plate to measure the forces in that axis is under compression force, as shown in the figure below.

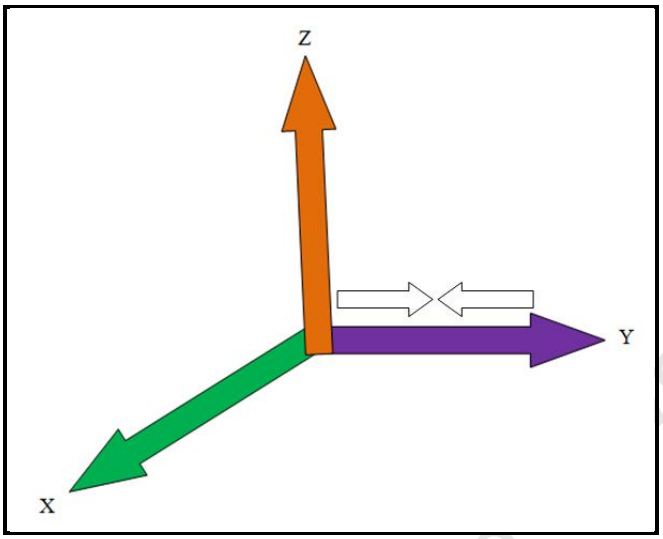


Figure 52: Showing the compression forces acting on the load cell mounted on the Y plate and happens when the flying object is moving backwards in a linear manner.

The relationship between the compressive forces versus the voltage output from the force transducers is shown below. The relationship can be best approximated by a best fit line which shows it to be a cubic function.

Table 12: Shows the relationship between the voltage and the Compressive forces acting on the Y plane

Relationship [V/N]	σy [V]	σx [N]	Residual
$y = -0.06066x^3 + 0.3358x^2 + 0.09069x - 0.023001$	0.5470	0.9089	0.095013

The actual graphical representation of the above relationship is shown in the appendix.

6.2.8 Z -axis Force (Tensile)

This is when the load cell mounted on the Z plate to measure the forces in that axis is under tensional force (stretched), as shown in the figure below.

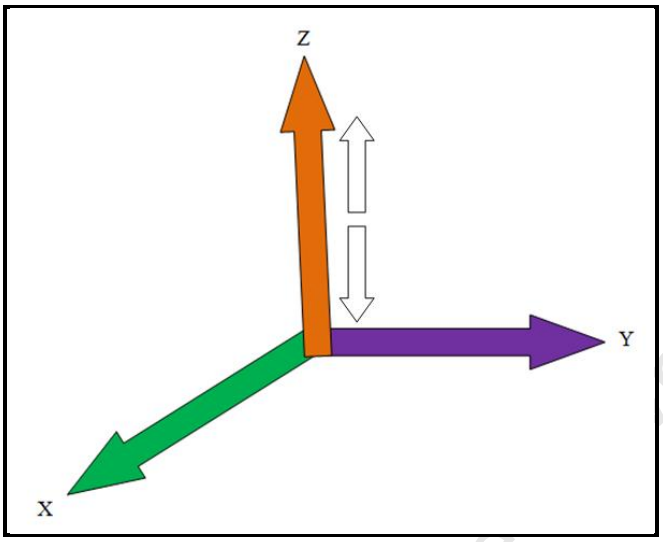


Figure 53: Showing the tensile forces acting on the load cell mounted on the Z plate and happens when the flying object is moving upwards in a linear manner.

The relationship between the tensile forces versus the voltage output from the force transducers is shown below. The relationship can be best approximated by a best fit line which is a cubic function.

Table 13: Shows the relationship between the voltage and the tensile forces acting on the Z plane

Relationship [V/N]	σy [V]	σx [N]	Residual
$y = -0.014087x^3 + 0.14675x^2 - 0.097381x - 0.0037956$	0.4491	1.3680	0.11764

The actual graphical representation of the above relationship is shown in the appendix.

6.2.9 Distance moved

After the dynamics analysis, the relationship of the servo angle and the distance moved in each axis was expressed as:

$$s = l_1 \cdot cos\theta_1 + \sqrt{l_2^2 - l_1^2 sin^2\theta_1}$$

Where s is the distance moved and l_1 is the length of the servo horn, l_2 is the length of the connecting rod (or the force transducer), θ_1 is the servo angle. Graphically the relationship is shown below.

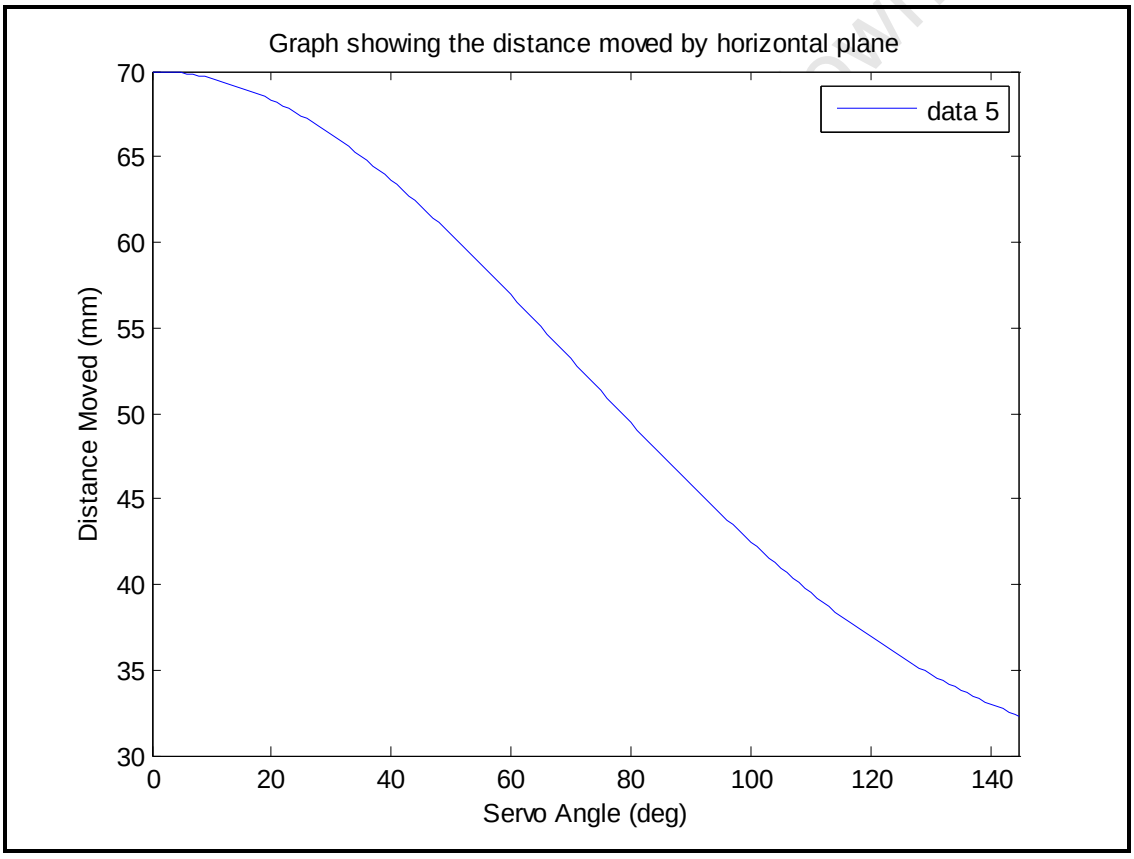


Figure 54: Showing the distance moved relative to actuator movement

The furthest distance moved by the translational plane is 70mm in the entire swing of the servo movement. Figure 54 above shows a predicted output relationship and not the actual measured output.

Relationship between the Servo angle and the Force angle (θ_2)

Algebraically, it is shown as

$$\theta_2 = \sin^{-1}\left(\frac{l_1 \sin\theta_1}{l_2}\right)$$

And graphically shown to be:

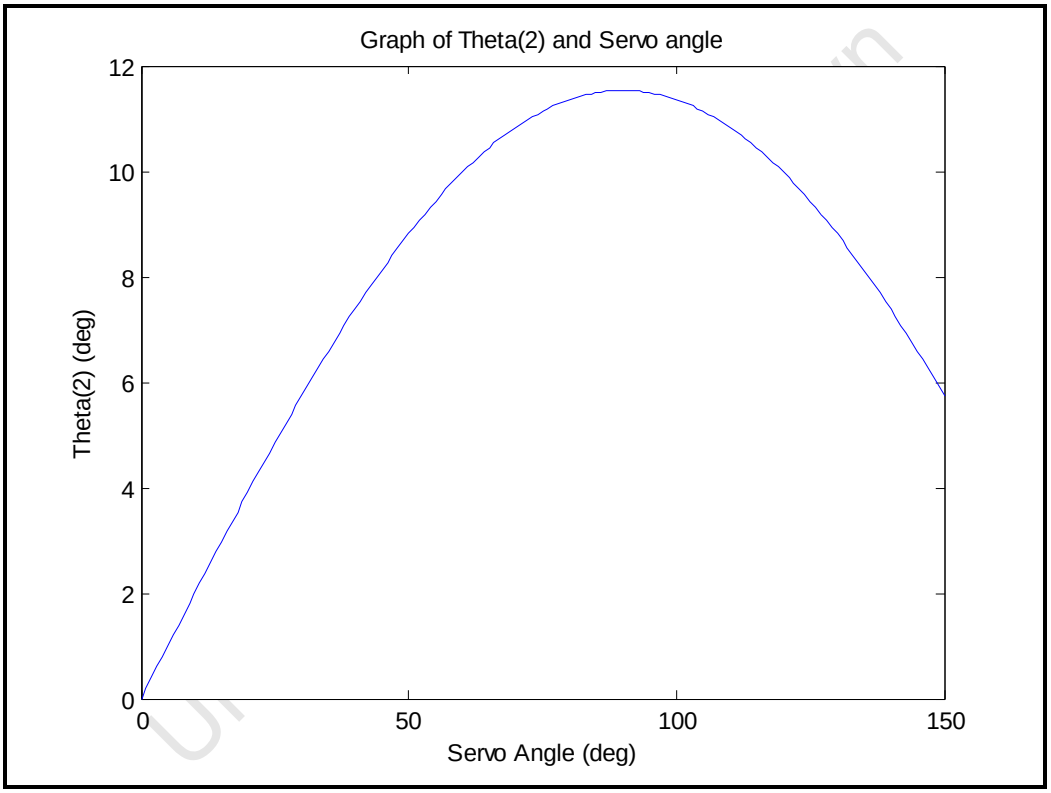


Figure 55: Showing the force angle relative to servo angle

The maximum angle of θ_2 is 11.5° for the entire movement of the servo angle with a resolution set to a maximum of 150°.

Actual Force

The actual force is algebraically shown as:

$$F_y = F\cos\left(\frac{l_1 \sin\theta_1}{l_2}\right)$$

Where F is the measured force on the force transducers, θ_1 is servo angle and l_1 and l_2 are constant length. At any given servo angle, the unit force F causing acceleration is graphically shown to be:

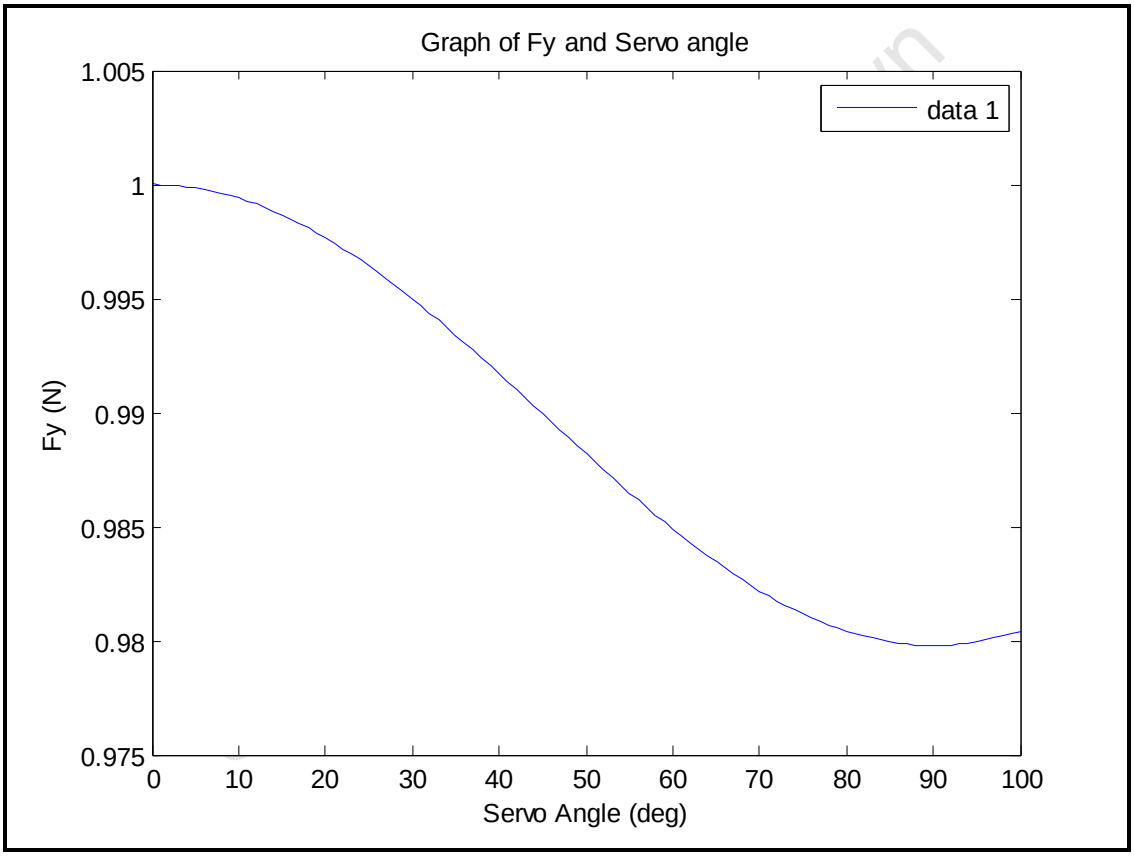


Figure 56: Showing the actual force relative to servo position

CHAPTER SEVEN
DISCUSSION

7.1 Hardware Design

7.1.1 Load Cell Ring

In general the flying forces of a bat are very small from 0.1 to 10N. In designing the strain ring, the aim was to get 300-800με which was a workable amount. The thickness of the strain gauge was 5mm and thus the ring width had to be bigger than 5mm for it to provide enough mounting space. A width of 8mm was chosen. The ring was also meant to be as small as possible and a radius of 12.5mm was chosen. Since the load-cell was meant to be as sensitive as possible, a thickness of 0.5mm was chosen for it. The values chosen above were a result of many calculations and thus gave the following results:

$$\sigma = 1.09\frac{PR}{wt^2}$$

Where *P* is load, *R* is radius, w is width and t is the thickness

$$\sigma = \frac{1.09 \times 5 \times 15 \times 10^{-3}}{8 \times 10^{-3} \times (0.5 \times 10^{-3})^2}$$

And this giving out 40875MPa and since the aluminium Young's modulus is 70GPa the micro-strain was:

$$\varepsilon = \frac{\sigma}{E}$$

Where σ is the stress and *E* is Young's modulus

$$\varepsilon = \frac{40.875 \times 10^6}{70 \times 10^9}$$
$$\varepsilon = 583.928\mu\varepsilon$$

The result was suitable for the project. The load cell output before amplification was as high as 11mV on simple hand compression and elongation of the ring. The load cell worked correctly.

7.1.2 Amplification Circuit

After amplification the voltage ranged to as high as 5.5V at the circuit output and it was then divided to give a maximum of 3V for the microcontroller, using the potentiometer at the end of the op-amp output of the circuit. The circuit introduced a very small amount of noise, and it was considered negligible and thus ignored as it did not affect the circuit performance. The circuit became highly sensitive after inserting and adjusting the potentiometer between the two standard resistors at the Wheatstone bridge of the circuit. The potentiometer was there to zero the output of the load cell when no force was exerted on the rings, and hence to zero the circuit output. The circuit worked correctly.

7.1.3 Platform

The platform was designed and simulated in Pro-Eng CAD. The CAD software gave good visual evidence of how the platform would look like when completed, but it did not show the real performance of the platform; for example how much friction was to be expected between sliding plates.

The Raw, Pitch, X, Y and Z plates of the platform all maintained their axis of alignment in the design and thus measured the forces axially only. The forces measured in the axis thus were appropriate forces.

When considering the Z plate produced in the final design, it was different to the software design. It was changed because a lot of time would have been necessary to wait for the prototyping machine to make the Z plate exactly the way it was designed in the software. In future work, it will be recommended that the Z plate be constructed as it was in the software, but nevertheless the supplementary design of the Z plate also worked well.

The Yaw design of the platform to make 6 DOF was not implemented in the design. This was because the Yaw rotational forces of the bat from theory are very small and the platform was made from aluminium. Though aluminium is a very light metal, due to the design setup of the

platform, it was realised that a large amount of force would have been needed to move the platform a small amount along the Yaw axis. The X-plate sliding in the grove of the platform round ring was the part designed for the Yaw movement.

To reduce the friction in the sliding planes, teflon was used because it has a very low co-efficient of friction with aluminium. This is shown in table 4 below.

Table 14: Showing the coefficients of friction

Material 1	Material 2	Coefficient Of Friction			
		Dry		Greasy	
		Static	***Sliding***	***Static***	***Sliding***
Aluminium	Aluminium	1.05-1.35	1.40	0.30	0.50
Aluminium	Teflon	0.24	0.19	0.08	0.10

The design of the platform gave 6 degrees of freedom, but the constructed system had 5 working degrees of freedom.

7.1.4 Actuators

The HB6001 and Futuba servo motors used in the project worked as expected.

7.2 Results

Generally, as seen from the results, the voltage outputs were proportional to the force. This form of pattern could be modelled by some mathematical algebraic expressions that were recorded in the results section. The mathematical expressions were also used to ease the programming of the movement of the axes in question. Although the relationships resemble a proportional structure, they were not all best fitted to be linear, but ranged from linear to 4th order polynomial. This could have been because of the non linear deformation of the load cell when there was a force was applied.

Some graphs have an almost flat response over a significantly wide range of forces from 0N to about 2.5N; this may have been caused by the not so accurate tuning of the circuit. The ADC may have been receiving a negative voltage (though it would display 0V because it was programmed to measure positive forces) and thus a larger force would be needed to overcame the negative output from the circuit to obtain a positive force response that will be displayed

on the LCD. The circuit was however not tampered with during the whole testing procedure described in the testing section.

The relationship between all forces and voltages was expected to be linear as depicted by the strain gauge datasheets, but it was not so in the practical testing of the project. This may have been caused by the not perfectly symmetric attachment of the strain gauges on the rings, and therefore resulting in the strain gauges not getting equal levels of strain and hence the non linear relationships obtained. Furthermore, the thickness of the rings was also not uniform. In addition to that, the non-uniform deformity of the rings when they were stretched and compressed, could also have accounted for the non linearity of the force and voltage relationship.

Although the relationships may not all have been linear, they are at least monotonic. It was decided to best fit quadratic, cubic and 4th order polynomials in the project so as to ease the programming of the platform. For future work, it will be recommended that the non linear relationships be modelled with piecewise linear sections. This method was planned to be used in the project but it was not implemented. The expectation was that they would deliver the same results and thus only one method was implemented.

7.3 Forces

The figure below shows orthogonal forces of a bat in flight oriented at an angle θ.

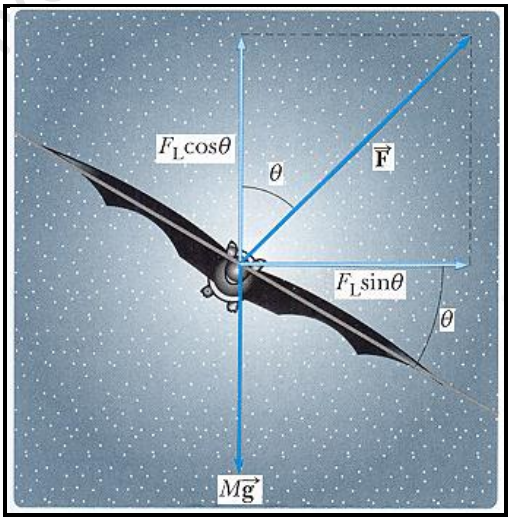


Figure 57: Showing lift forces of a bat in flight resolved into normal Cartesian axes [48]

All the forces that will be measured on the load cell would be a result of the resolved forces of all the forces that are in charge of flight in flying objects in the respective X, Y, Z, Roll and Pitch axes. This includes the forward, backward, left and right side slips, nose-down and nose-up movements of the flying objects. All these forces are outlined in detail in the literature review.

7.4 Software

7.4.1 Design

The whole design of the hardware was done in Pro-Eng CAD. The platform was first drawn by individual parts and then it was assembled together in Pro-Eng. This CAD software was used because it provided the accurate visual of what the platform would look like in real life and thus could be evaluated before it was built. Though Pro-Eng provided the above advantage, it did not provide the reality of the performance of the platform. For example, the amounts of friction to be overcome and how powerful the actuators were to be to control the entire platform were not modelled.

7.4.2 ADC

The ADC was tested using the LEDs and LCD of the processor as described in the testing section. This was because the testing was done by manually adjusting the potentiometer on the processor connected to the ADC input. In adjusting the potentiometer, the LEDs had to light up in the order of the adjustment and the result (number) shown by the LED was meant to be displayed on the LCD.

The ADC worked correctly.

7.4.3 Timers

The timer was tested manually using an oscilloscope and the results shown in the graph in the results section. The timer PWM was also connected to a servo motor to check if it was working correctly. During the experiments, the timers were reset manually because the forces measured had settle out on the screen and be recorded before the platform could change its position.

The timer worked correctly.

7.4.4 LCD

The LCD was tested using the demo code as described in the testing section so that all the sections/parts of the LCD could be tested and any defects be noticed because the demo code had an outlined testing procedure.

The LCD worked correctly

7.4.5 Micro-processor

The GT16A processor only has two timers and this resulted in the use of three GT16A processors for the project. This however did not hinder any progress in acquiring them as they are readily available in the Electrical Engineering Electronic component stock. For future work, a processor with more timers can be used in controlling the servos motors.

The processor worked correctly

7.4.6 Programmes

All the control mechanisms of the platform were done through programming the platform in C language. These include programming the ADC, Timers, LCD and servo motors. The programmes were written in Freescale Code warrior and loaded into the GT16A micro-processor.

All the programmes written worked correctly.

7.5 Assumptions Made

Some assumptions were made during the project to acquire the results described above and these are:

- There friction between sliding plates was assumed to be insignificant
- The axis of rotation for the platform was assumed to be the same and smooth
- During the pitch and roll movements, the platform was assumed to have been moving directly in-line with the actuators, as in the case of the sliding plates as shown in the design analysis of the horizontal plate movements
- The servo motors used were all ideal
- The actuator attachments were rigid

- All the forces caused by the rolling and sliding effects of the platform itself were negligible and the forces measured were fully a result of the forces from the flying object
- Actuators' weights were negligible.

Before the work of this project was started, 6 DOF platforms were used in a large number of other applications as mentioned previously. The 6 DOF platforms can also be used to simulate the flight paths of flying animals or man-made machines with complicated flight paths. The design can also be used to find or detect a position in space, though within a localised area within its workable space.

It can also be said that from the results, that the forces of flying objects can be measured with acceptable accuracy using a 6 DOF platform with embedded force transducers. Thus for future developments of flying objects, the platform can be used to measure forces in question. With more mathematical model computations applied to the forces measured, more complicated forces can be measured by the platform, for example getting the magnitude of a single force that may result in all the three rotational movements.

CHAPTER EIGHT

CONCLUSIONS

The following conclusions are drawn based on the results and platform analysis.

8.1 Requirements Achieved

A 6 DOF robotic platform was designed and constructed. This platform contained six on-board servo motor actuators that were responsible for the platform movements. In addition to that, the servo motors were also in charge of locking the platform axes to a fixed position if they were not in use. The platform also contained 5 load-cell strain rings mounted on five different axes. These rings measured the forces at respective axes independently to one another. The signal from the load-cells was amplified to a workable voltage through a circuit and fed into a microcontroller. The microcontroller was then used to calculate the forces and program the direction and amount of movement the servo had to move. The circuits, micro-processor and servo motors were powered by the lab power-supply.

8.2 Hardware

Although the system design was structurally designed to have 6 degrees of freedom, it had 5 working degrees of freedom. The forces were measured appropriately in all the axes as the load cells, actuators and circuits performed desirably. While the system design was aesthetically pleasing, the joints between the actuators and transducers would occasionally become loose during testing, but would be tightened. The material used also led to the system being moderately heavy. The circuit board was not machined and thus resulted in all the circuits used in the project being built on bread-board.

8.3 Software

All the C programmes coded in the GT16A for programming the actuators (servo motors), timers, analogue to digital convertors and LEDs performed correctly. All these programmes could be loaded on the GT16A memory and thus the possibility of using the platform outdoors provided it had a suitable on board power supply. In addition to that, Pro-Eng CAD provided good visuals of the system but not the real performance of platform. Furthermore, Eagle CAD software also provided a good circuit and board design schematics.

8.4 General

Generally, the final actual system design was environmentally and operationally safe to use and very user friendly. It was also durable and affordable because it used standard parts and readily available material.

CHAPTER NINE

RECOMMENDATIONS

The following recommendations are drawn based on the above conclusions.

9.1 Current Study

Platform

Further work should be done on the design so as to implement the sixth degree of freedom and be able to measure forces in all the six axes.

Transducer

A force transducer should be designed so as to be able to better measure forces in a rotational movement, for example if the 6th degree of freedom had to be implemented.

A simpler force transducer should be designed to measure linear forces because the load-cell rings need a lot of precision in manufacture.

Sliding

The use of linear bearings between sliding plates so that friction may be reduced as much as possible is recommended.

Material

The use of much lighter material in the design of the platform, for example plastic material since the forces involved are small, so that the platform may be light and very portable is recommended.

Power supply

The use of an on board power supply such as a battery with recharge function so that the platform may be tested and used outdoors is recommended.

Size

Reduce the size of the platform so that the control mechanism is made much easier.

Actuators

Position the actuators on the platform at a place where their weight will have no effect in straining the load-cell.

Joints

The joints between actuators and the force transducers and the platform should be made rigid so that they do not loosen during testing and thus will result in the correct strain being exerted on the load-cells.

Circuit Board

Manufacture the circuit board and mount it at the base of the platform in the integrated design so that all the hardware is on the platform.

9.2 Further Studies

Servo control

A feedback mechanism to control the speed of response of the motors should be considered and implemented as flying objects do not all fly at the same speed.

Automatic response

The robotic system must be made to respond automatically to the forces when the flying object is attached.

LIST OF REFERENCES

[1] *"Self-Organization, Embodiment, and Biologically Inspired Robotics,"* Rolf Pfeifer, Science 318, 1088 (2007); DOI: 10.1126/science.1145803.

[2] U. M. Norberg, *"Some Advanced Flight Manoeuvres of Bats"*, Department of Zoology, University of Gdteborg, Fack, 5-400 33 Gdteborg 33, Sweden,1976, pp 489-495.

[3] *"In Motion Simulation Products."* http://www.inmotionsimulation.com/products.html **[21-11-2010].**

[4] U. M. Norberg, *"Vertebrate Flight"*, Zoophysiology , Vol. 27, S. D. Bradshaw, S. Ishii, Ed. Springer-Verlag Berlin Heidelberg: 1989, pp 73-75.

[5] Science Daily: *"Bat flight generates complex aerodynamic tracks"*. http://www.sciencedaily.com/releases/2007/05/070510160911.htm **[21-11-2010].**

[6] C. P. Ellington, *"Vortices and hovering flight"*, In: Nachtigall W. Ed. Instationare Effekte an schwingenden Tierflugeln, Steiner, Wiesbaden, pp 64-101.

[7] Diracdelta Science and Engineering, *"Lift."* http://www.diracdelta.co.uk/science/source/l/i/lift/source.html **[01-02-2011].**

[8] *"Forces of Flight – Lift*." http://www.fi.edu/flight/own2/lift.html **[21-11-2010].**

[9] *"Principles of flight – Instructor."* http://wings.avkids.com/Book/Flight/instructor/forces-01.html **[21-11-2010].**

[10] U. M. Norberg, *"Hovering flight of Plecotus auritus Linneaus"*, Bijr. Dierk, 1970, pp 45, 62-66.

[11] H. D. J. N. Aldridge, *"Body accelerations during the wingbeat in six bat species:, the function of the upstroke in thrust generation."* J. Exp. Biol. 130, 1987, pp 275-293.

[12] J. M. V. Rayner, H. D. J. N. Aldridge, *"Three-dimensional reconstruction of animal flight paths and the turning flight of microchiropteran bat,"* J. Exp. Biol. 118, 1985, pp 247-265.

[13] C. P. Ellington, *"The aerodynamics of hovering insect flight I-VI"*. Phil. Trans. R. Soc. Ser. B 305, 1984, pp 1-181.

[14] C. J. Pennycuick, *"Mechanics of flight," In Avian Biology,* Vol.. 5, D. S. Farner, J. R. King and K. C. Parkes, Ed. London: Academic Press, 1975, pp. 1-73.

[15] U. M. Norberg, *"Vertebrate Flight", Zoophysiology* , Vol. 27, S. D. Bradshaw, S. Ishii, Ed. Springer-Verlag Berlin Heidelberg: 1989, pp 19-35.

[16] U. M. Norberg, *"Vertebrate Flight", Mechanics, Physiology, Morphology, Ecology and Evolution*, Vol. 27, W. Burggren, H. C. Heller, Berlin: Springer Verlag, 1990, pp 25 – 30.

[17] C. J. Pennycuick, *"Mechanics of flight," In Avian Biology*, Vol.. 5, D. S. Farner and J. R. King Ed, London: Academic Press, 1975, pp 1-5.

[18] J. M. V. Rayner, *"A new approach to animal flight mechanics,"* J Exp Biol 80, 1979, pp 17-54.

[19] J. M. V. Rayner, *"Flapping flight aerodynamics and the evolution of flight in bats,"* Biona Report 5, 1986, pp 27-74.

[20] M. Andersson and R. A. Norberg, *"Evolution of reversed sex dimorphism and role partitioning among predatory birds, with a size scaling of flight performance,"* Biol. J. Linn. Soc. 15, 1981, pp 105-130.

[21] R. Dudley and C. P. Ellington, *"Mechanics Of Forward Flight In Bumblebees, Kinematics and Morphology,"* Department of Zoology, University of Cambridge, J. exp. Biol. 148, 1990, pp 19-32.

[22] S. J. Kirkpatrick, *"The moment of inertia of bird wings,"* J. Exp. Biol. 151, 1990, pp 489-494.

[23] R. Dudley and C. P. Ellington, *"Mechanics Of Forward Flight In Bumblebees, kinematics and morphology,"* Department of Zoology, University of Cambridge, J. exp. Biol. 148, 1990, pp 36-52.

[24] R. M. C. N. Alexander, and A.Vernon, *"The mechanics of hopping by kangaroos,"* Macropodidae, ZooL, Lond. Yll, 1975, pp 265-303.

[25] M. Hiller, S. Fang, S. Mielczarek, R. Verhoeven and D. Franitza, *"Design, analysis and realization of tendon-based parallel manipulators,"* Mechanism and Machine Theory 40, 2005, pp 429–445, Available online on www.sciencedirect.com

[26] E. H. L. Aarts, and P. J. M.Van Laarhoven, *"Statistical cooling: A general approach to combinatorial optimization problems,"* Philips Journal of Research, 1985, pp 193–226.

[27] B Shirinzadeh, S. K. Saha, G. Alici, *"Dynamic Model Simplification of Serial Manipulators"*, On Robotics & Automation, ISRA 2006, San Miguel Regla Hotel, Hgo, Mexico, Aug. 25-28, 2006, pp 1-6.

[28] *"Modular Robotics & Robot Locomotion"*, Parallel Robot Design, http://155.69.254.10/users/risc/www/para-intro.html **[01-02-2011]**

[29] S. Fang, D. Franitza, M. Torlo, F. Bekes, and M. Hiller, *"Motion Control of a Tendon-Based Parallel Manipulator Using Optimal Tension Distribution,"* IEEE/ASME Transactions On Mechatronics, Vol.. 9, No. 3, September 2004.

[30] Manfred Hiller, Shiqing Fang, Sonja Mielczarek, Richard Verhoeven, Daniel Franitza, *"Design, analysis and realization of tendon-based parallel manipulators,"* Mechanism and Machine Theory 40, 2005, pp 429–445, Available online on www.sciencedirect.com

[31] *"Barrett Technology Whole Arm Manipulator (WAM),"* http://www.barrett.com/robot/index.htm **[01-02-2011].**

[32] D. L. Pieper, *"The Kinetics of manipulators under computer control,"* Thesis, Stanford University, Department of Mechanical Engineering, 1968

[33] R. Verhoeven, *"Analysis of the Workspace of Tendon-based Stewart Platforms"*, Dr Thesis, Universität Duisburg-Essen, 29 July 2004.

[34] E. Red, "Manufacturing and Mechanical Engineering," Robotics overview, http://research.et.byu.edu/eaal/html/RoboticsReview/body_robotics_review.html **[01-02-2011].**

[35] "Learn about robots", Robotics glossary, http://www.learnaboutrobots.com/glossary.htm **[01-02-2011].**

[36] M. Hiller, S. Fang, S. Mielczarek, R. Verhoeven and D. Franitza, *"Design, analysis and realization of tendon-based parallel manipulators,"* Mechanism and Machine Theory 40 ,2005, pp 429–445, Available online on www.sciencedirect.com

[37] Ou, Yeong-Jeong and Tsai, Lung-Wen, *"Advances in Robot Kinematics and Computational Geometry,"* Design of a three-dof tendondriven manipulator with the characteristics of equal maximum tensions, In Lenarˇciˇc, Jadran and Ravani, Bahram, Ed. Dordrecht, Netherlands: . Kluwer Academic Publishers, 1994, pp 369–378.

[38] Ou, Yeong-Jeong and Tsai, Lung-Wen. *"Robotics: Kinematics, Dynamics and Control,"* Theory of isotropic transmission for tendon-driven manipulators, USA, Minneapolis: ASME, 1994, pp 53–61.

[39] Dietmaier, Peter. *"Parallel Manipulators: Stewart Paltform, The Stewart-Gough platform of general geometry can have 40 real postures,"* In Lenarcic and Husty Ed, pp 7–16.

[40] *"Nanotechnology and Advanced Motion Controlled Positioning System"* http://www.alioindustries.com/hexapod_news.html **[21-11-2010]**.

[41] *"Roymech: Coefficients of friction"* http://www.roymech.co.uk/Useful_Tables/Tribology/co_of_frict.htm **[01/11/2010].**

[42] D. A. Milner and J.D. Brindlay, *"The two-dimensional force dynamometer in cutting process of peripheral milling,"* Microtechnic, 1982, pp 89–92.

[43] J. W. Dally, W. F. Riley and K. G. McConnell, *"Instrumentation for Engineering Measurements,"* New York: John Wiley & Sons, pp. 249 – 252.

[44] Analogue Devices Incorporated. (2004). *"Low Cost Low Power Instrumentation Amplifier,"* AD620 Data sheet, pp 1-13.

[45] Unidec Technology Paper. (March 6 2002). *"Servo Systems"*.

[46] HCS08Microcontrollers. (2006). *MC9S08GT16A/GT8A devices data sheet*, Rev 1, July, pp 156-230.

[47] Samuel Ginsberg, University of Cape Town, Department of Electrical Engineering, EEE3017W, GT16 notes, 2008.

[48] *"Science Questions."* http://www.cramster.com/answers-mar-10/physics/physics-maximum-lift-force-bat-proportional-squar_796672.aspx **[14-12-2010].**

APPENDIX A

Circuit Schematics

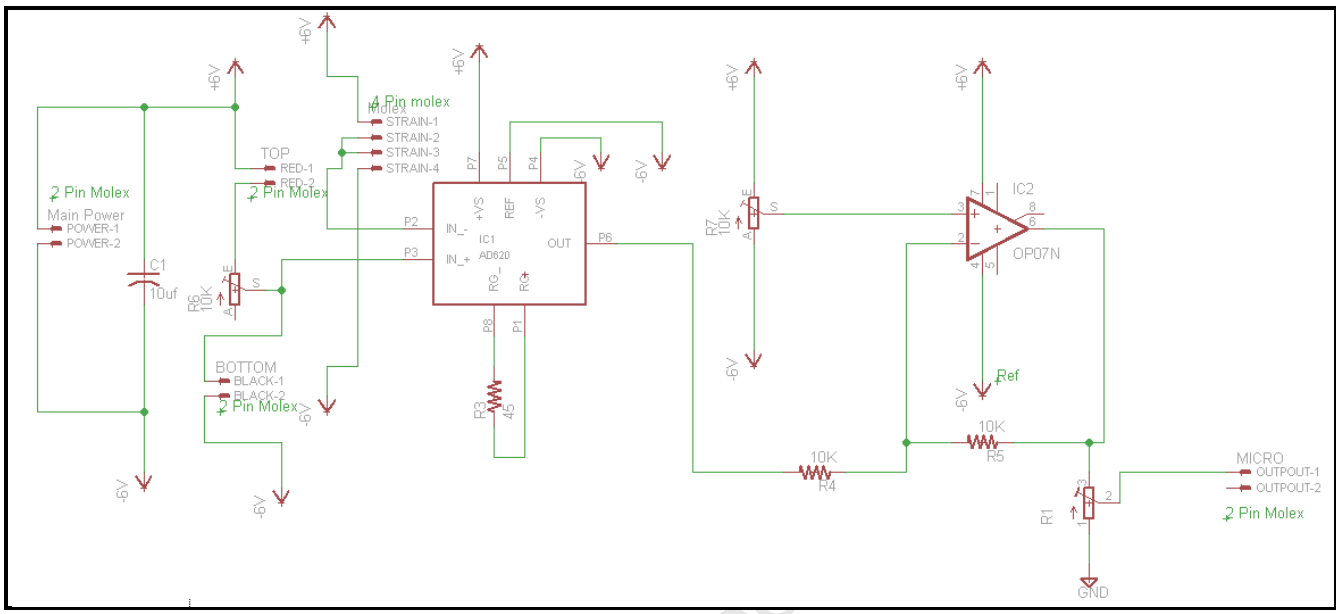


Figure 58: Showing an Eagle CAD schematic of the final circuit to be printed on circuit board

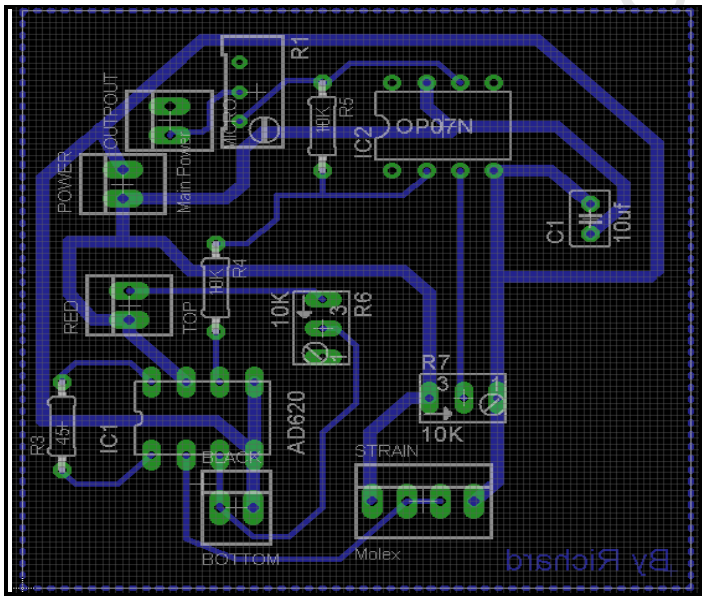


Figure 59: Showing the Eagle CAD final circuit board arrangement design

APPENDIX B

Microcontroller

Flash Memory

It has the following features:

- Size is 16384 bytes (32 pages of 512 bytes each)
- Single power supply program and erase down to 1.8 V
- Command interface for fast program and erase operation
- Auto power-down for low-frequency read accesses

Microcontroller Registers

The registers are divided into three groups and these are:

- Direct-page registers (0x0000 through 0x007F): These are located in the first 128 locations in the memory map, so they are accessible with efficient direct addressing mode instructions.

- High-page registers (0x1800 through 0x182B): these are used much less often, so they are located above 0x1800 in the memory map. This therefore leaves more room in the direct page for more frequently used registers and variables.

- Non-volatile registers (0xFFB0 through 0xFFBF): these consist of a block of 16 locations in FLASH memory at 0xFFB0–0xFFBF.

Microcontroller Timers

Gt16A timers have the following features:

- It has a total of eight channels that may each be configured for buffered, centre-aligned pulse width modulation (CPWM).

- Has a clock source to prescaler for each TPM that is independently selected as bus clock, fixed system clock, or an external pin

- It has a 16-bit free running (CPWM) count operation and a 16-bit modulus register to control the range.

Timer x Status and Control Register (TPMxSC)

The TPMxSC contains controls that relate to all channels within the timer module. These include the overflow status flag and control bits that are used to configure the TMP configuration, interrupt enable, clock source, and the prescale divisor.

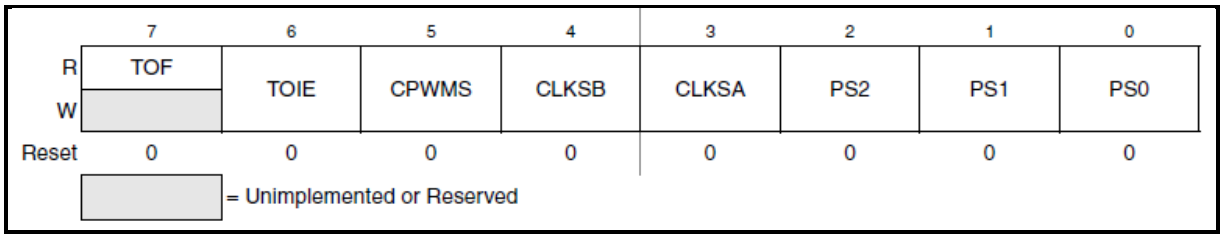


Figure 60: Timer x Status and Control Register (TPMxSC).
Adapted from the MC9S08GT16A datasheet

TPMxSC Register Field Descriptions

7 *TOF: Timer Overflow Flag* — The flag is set when the TPM counter changes to 0x0000 after reaching the modulo value programmed in the TPM counter modulo registers. When the TPM is configured for CPWM, TOF is set after the counter has reached the value in the modulo register, at the transition to the next lower count value.

6 *TOIE: Timer Overflow Interrupt Enable* — This is a read/write bit that enables TPM overflow interrupts. If TOIE is set(interrupt enabled), an interrupt is generated when TOF equals 1. Reset clears TOIE. If not set then the interrupts are inhibited and this is done by software polling.

5 *CPWMS: Center-Aligned PWM Select* — This is a read/write bit that selects CPWM operating mode. The Processor Reset clears this bit so the TPM operates in up-counting mode for input capture, output compare, and edge-aligned PWM functions. When the CPWMS is set, it reconfigures the TPM to operate in up-/down-counting mode for CPWM functions. Reset clears CPWMS.

4:3 *CLKS[B:A]: Clock Source Select*—this is a 2-bit field that is used to disable the TPM system or select one of three clock sources to drive the counter prescaler. The external source and the XCLK are synchronized to the bus clock by an on-chip synchronization circuit.

2:0 *PS[2:0]: Prescale Divisor Select* — This is a 3-bit field that selects one of eight divisors for the TPM clock input . This prescaler is located after any clock source synchronization or clock source selection, so it affects whatever clock source is selected to drive the TPM system.

When using the GT16A it should be noted that, the maximum frequency that is allowed as an external clock is one-fourth of the bus frequency.

Analogue to Digital Convertor (ATD)

It possesses the following desirable features for the project:

- Has a 8-/10 bit resolution
- 14.0 μsec, 10-bit single conversion time at a conversion frequency of 2 MHz which is fast enough for this project
- Has a conversion complete flag or conversion complete interrupt generation which will enable the ease in programming.
- Analogue input multiplexer for up to eight analogue input channels
- Single or continuous conversion mode

Functional Description

The ATD uses a successive approximation register (SAR) design. The ATD contains all the necessary elements to perform a single analogue-to-digital conversion. When programming the processor, a write to the ATDSC register initiates a new conversion and a write to the ATDC register interrupts the conversion in progress but it will not initiate a new conversion. A write to the ATDPE register aborts the existing conversion but will not initiate a new conversion. If a conversion is already running when a write to the ATDSC register is made, it will be aborted and a new one will be started.

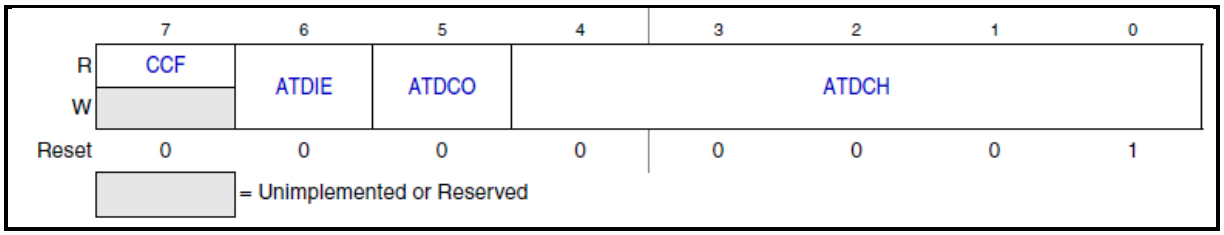


Figure 61: ATD Status and Control Register (ATDSC)

Where:

7 CCF: Conversion Complete Flag—The CCF is a read-only bit which is set each time a conversion is complete. The CCF bit is cleared whenever the ATDSC register is written.

When it is 0, then the conversion in progress is not complete and a 1 means the conversion is complete.

6 ATDIE: ATD Interrupt Enabled—when the bit is set, an interrupt is generated upon completion of an ATD conversion. The interrupt will remain pending as long as the conversion complete flag CCF is set. When the bit is not set (0), it means the interrupts are disabled and set (1) means interrupts enabled

5 ATDCO: ATD Continuous Conversion—when this bit is set, the ATD will convert samples continuously and update the result registers at the end of each conversion. When this bit is cleared, only one conversion is completed between writes to the ATDSC register. When the bit is not set (0) it means the ADC operates on Single conversion mode and if the bit is set (1) it operates on continuous conversion mode.

4:0 ATDCH: Analog Input Channel Select—the bits select the analogue input channel in which a signal is sampled and converted to digital codes.

Include Files

i. MC9S08GT16A.h is the standard Microprocessor header which contains the definitions for all the macros and function registers to setup the system specifications of the GT16A Micro-processor.

ii. hidef.h contains the fundamental declarations that enable the setup of Interrupts in the entire software.

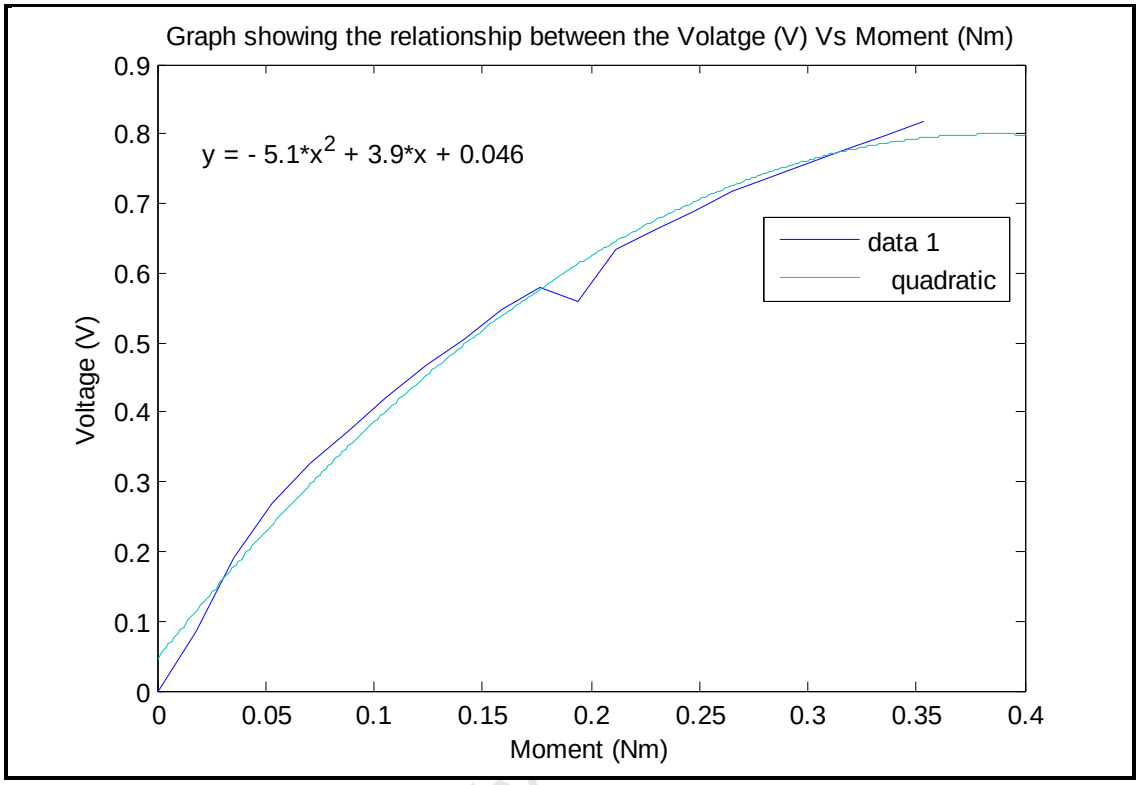
iii. string.h contains the standard C strings library

iv. LCD.h contains the fundamental declarations that enable the setup of the port lines of the LCD to be used by the GT16A processor.

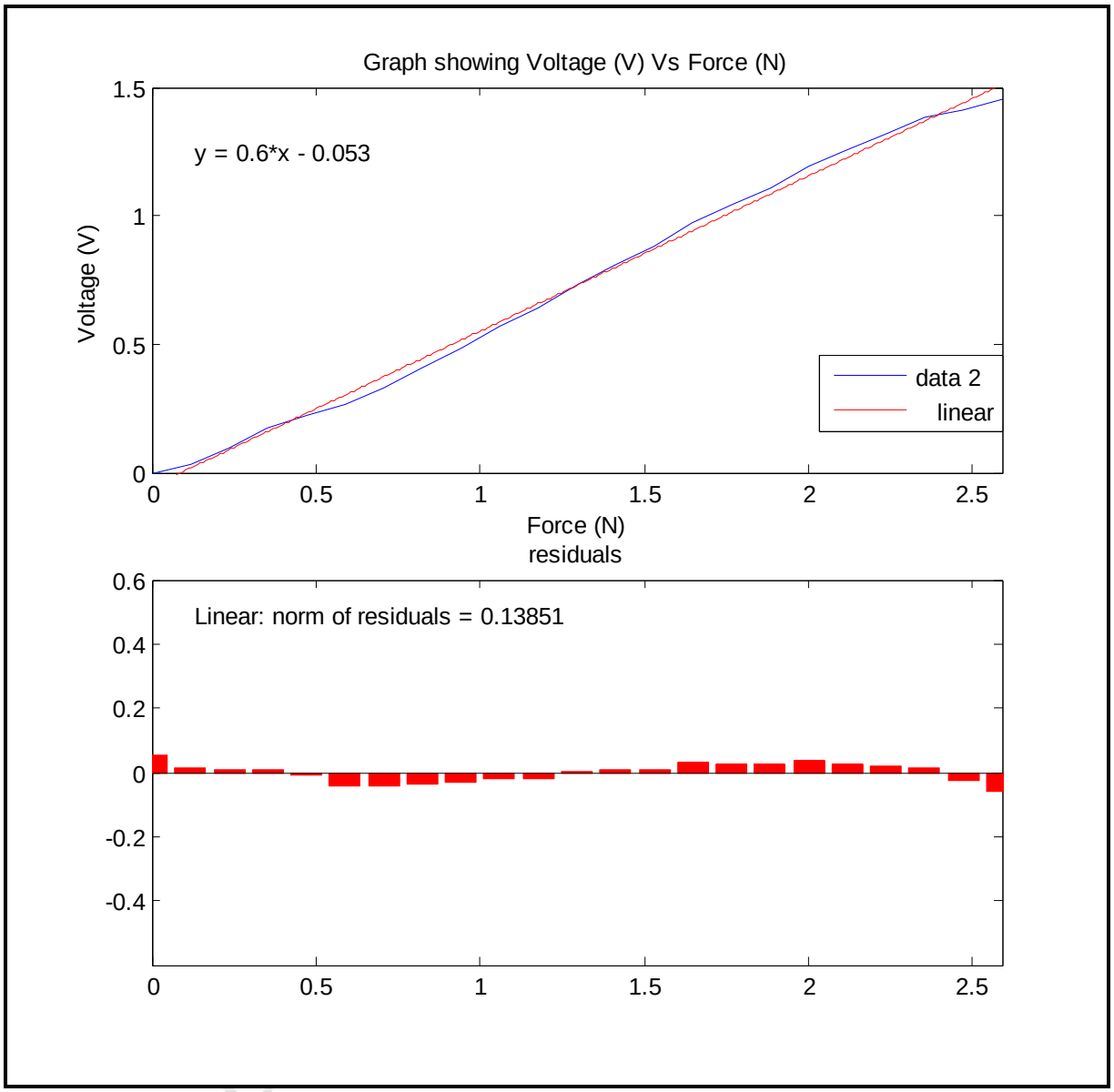
v. math.h contains the standard C math library

APPENDIX C

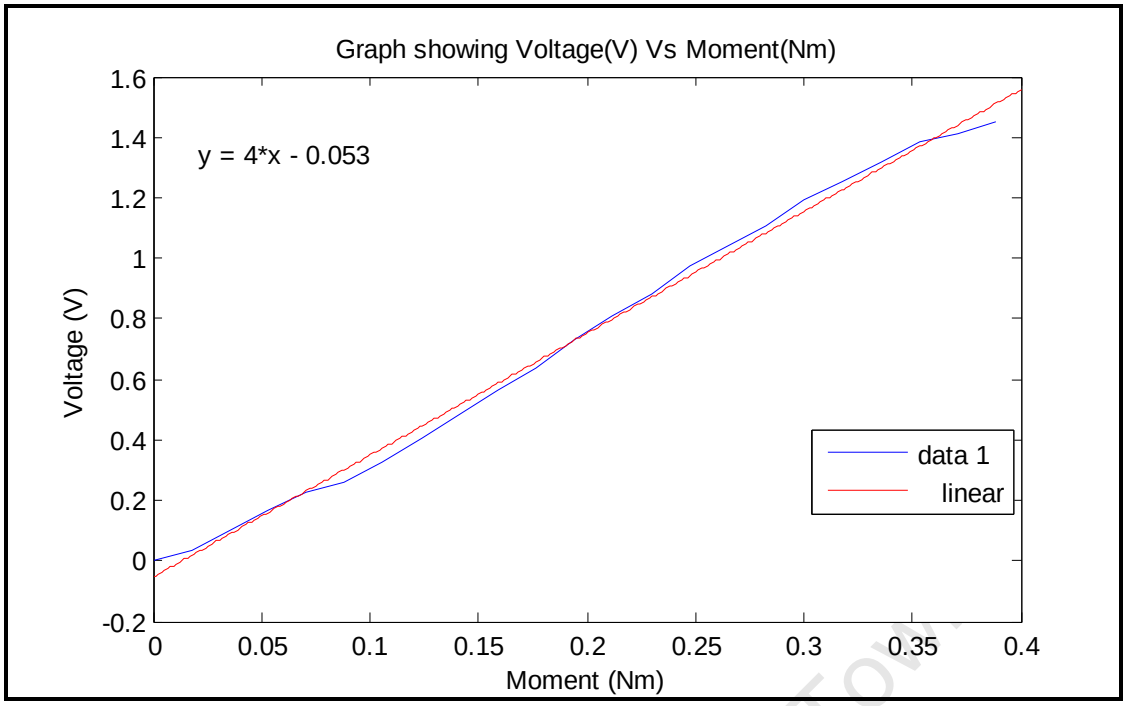
Graphs showing the relationship of voltages, forces and moments



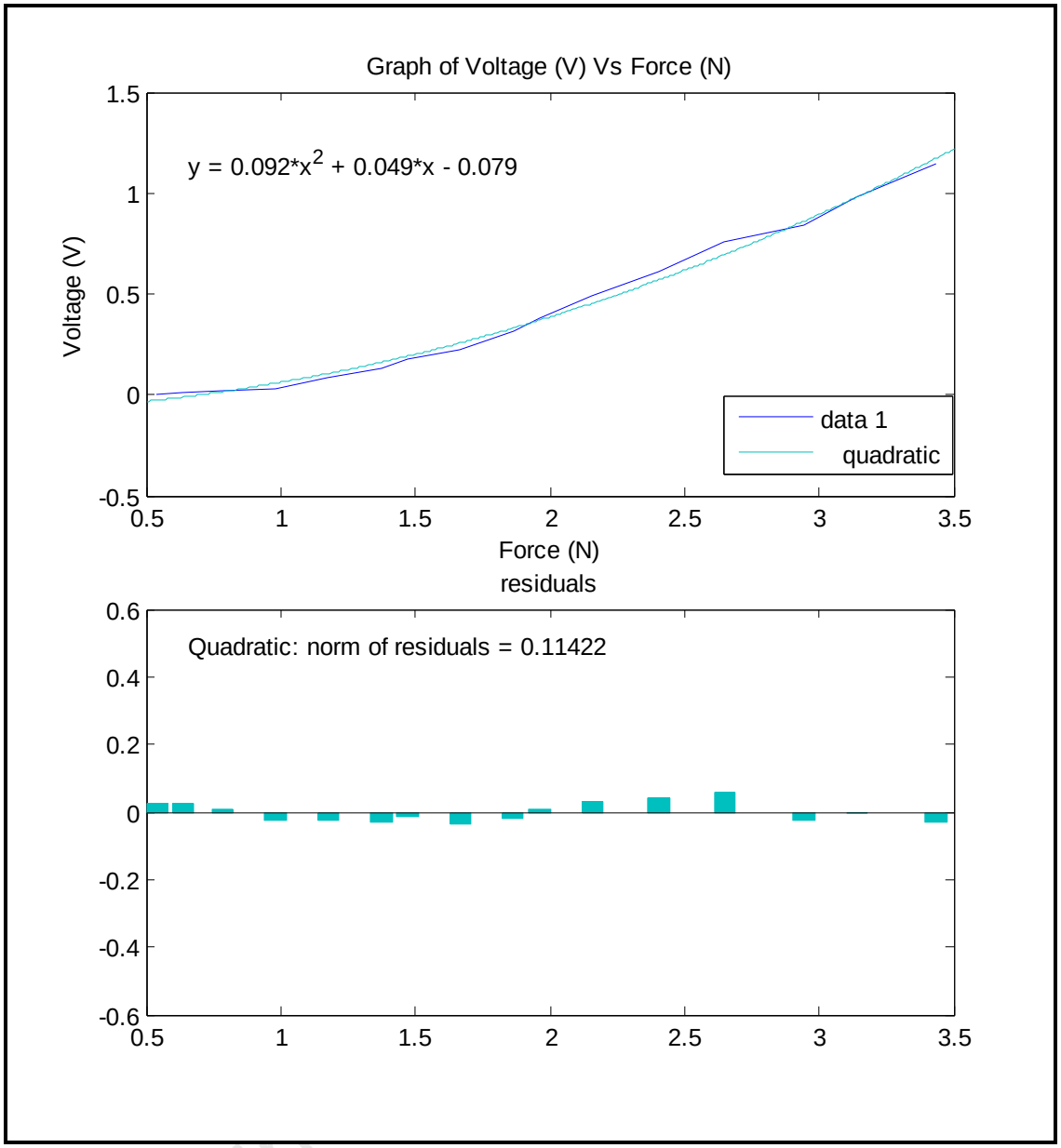
Showing the Moment and Voltage output relationship for the backward pitch movement



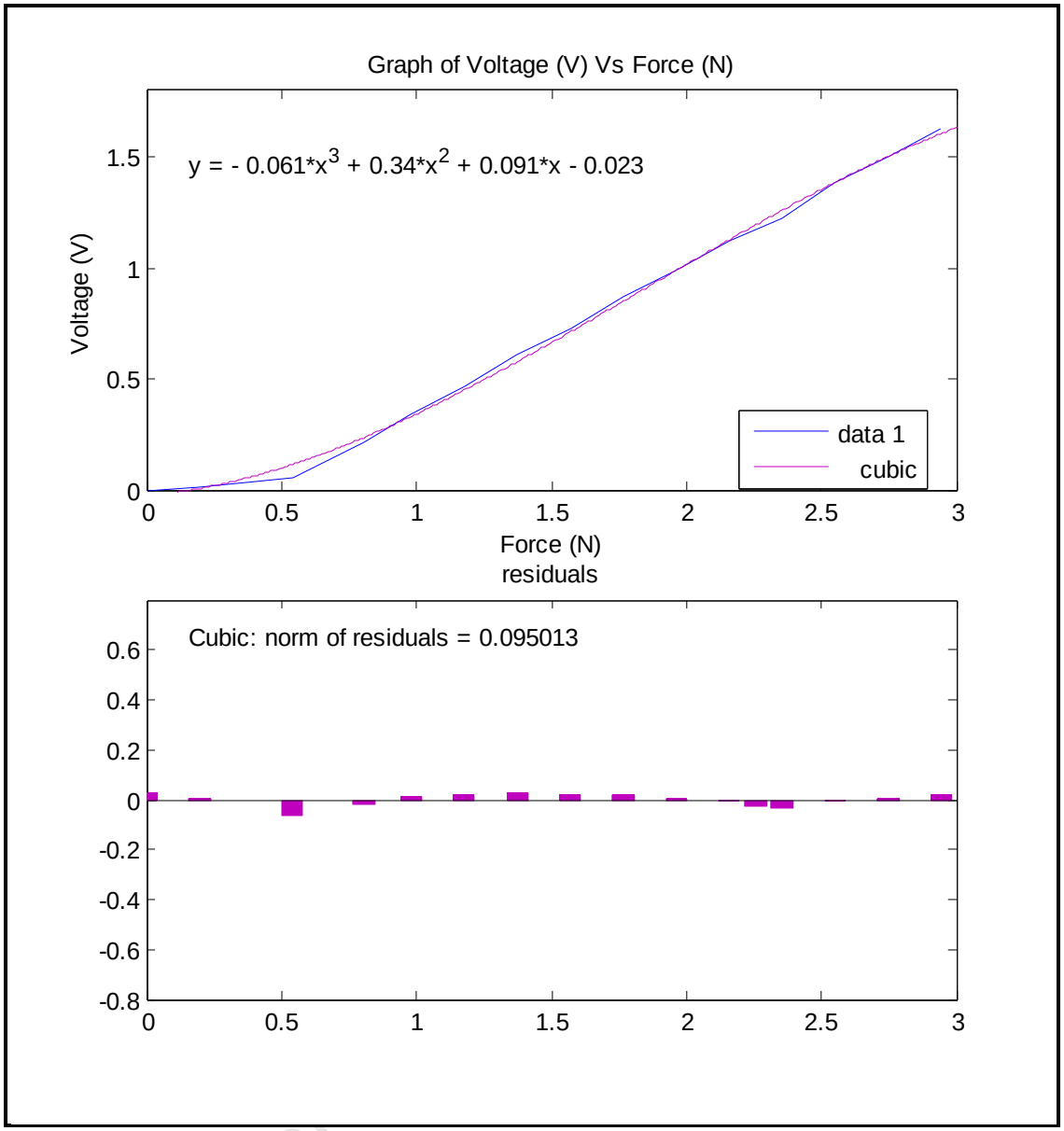
Showing the relationship between the Voltage Output (V) and the Compressive Force (N) on the Roll axis



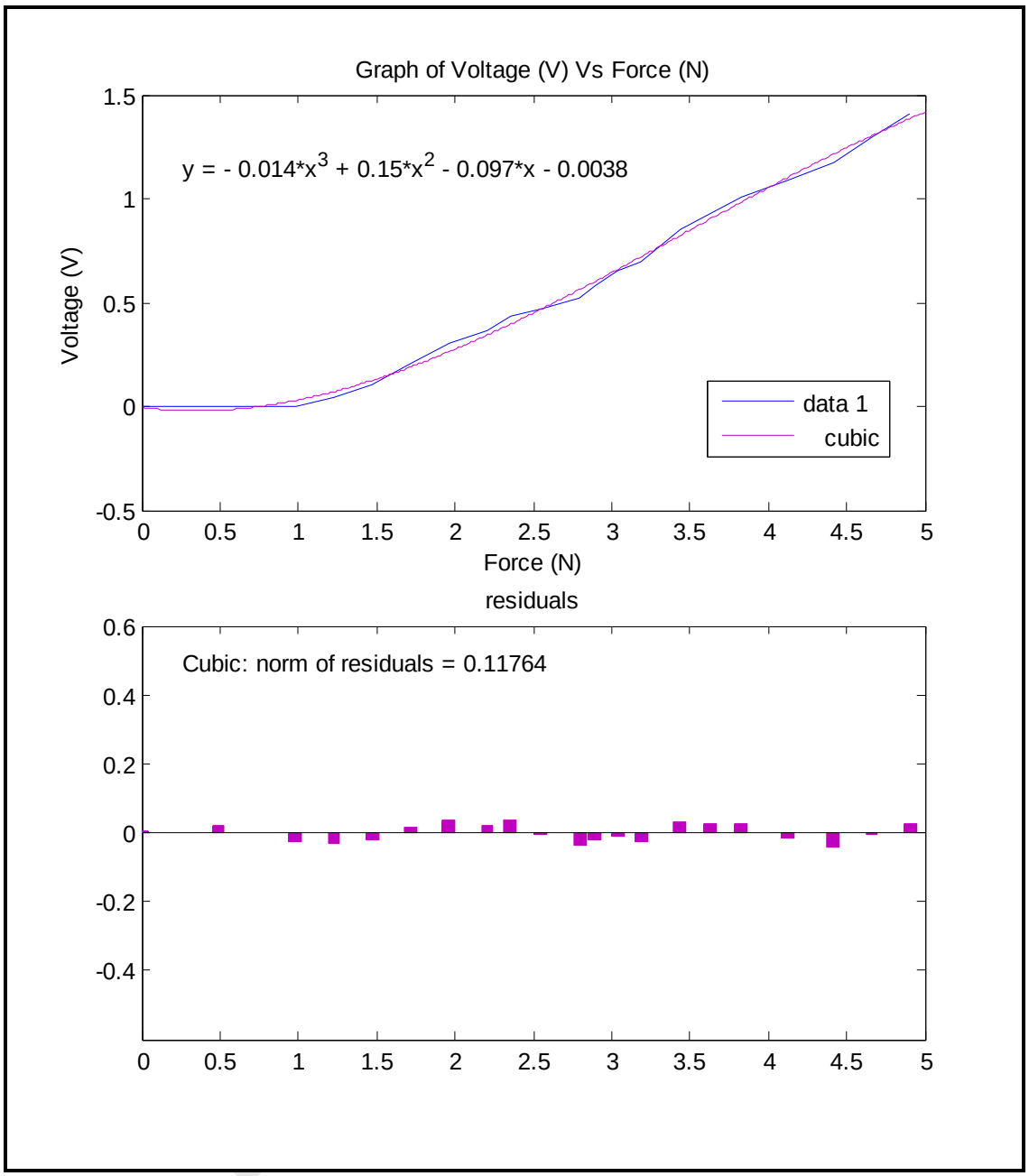
Showing the Moment and Voltage output relationship the forward Roll movement



Showing the relationship between the force and the Voltage output the Compressive forces acting on the X plane



Showing the relationship between the Force (N) and the Voltage (V) for the compressive forces acting on the Y plane



Showing the relationship between the force and the Voltage output for the tensile forces acting on the Z plane

APPENDIX D

C-code

Programme calculates the forces on the X and Y axes only

```
//#########################################################################
//=  This program will sample pin ADC0 and ADC1. The ADC is set       =
//=  to continuous 8-bit mode. The value will be converted to a       =
//=  force and displayed on the LCD. The timers are used to control   =
//=  the actuator by providing a duty cycle.                          =
//#########################################################################

//#########################################################################
//    The Include Files
//#########################################################################

#include <hidef.h> /* for EnableInterrupts macro */
#include <MC9S08GT16A.h> /* include peripheral declarations */
#include <string.h>
#include "LCD.h"
#include <math.h>

//#########################################################################
//   Declaration of Functions used
//#########################################################################
void lcd_init (void);
void lcd_string (unsigned char *instring);
void lcd_command (unsigned char command);
void Init_ADC(void);
void   Convert_toBCD(unsigned int number, unsigned char* result_ptr);
char*  Convert_toASCII(unsigned char* result_ptr);



//#########################################################################
```

```
//     Global VAriables - set in the durect page in the Ram
//#############################################################################
unsigned char ATD_channel;
unsigned char conv_result[4];
unsigned char conv1[5];
unsigned char conv2[5];
unsigned int  result [2];
unsigned int  thot [4];
unsigned char *ptr1;
void servox (void );
void duty_x (void );
void duty_y (void );
void servoy (void );
void Force(void);
unsigned char* thou;
unsigned char *thou1;
unsigned char *thou2;
unsigned int  in1;
unsigned int  in2;
unsigned int  volt;
unsigned int  valA;
unsigned int  valB;
unsigned int  i;
unsigned int  Diff1;
unsigned int  Diff2;
unsigned int  force;
unsigned int  forcex;
unsigned int  forcey;
unsigned int  fo;
unsigned int  serx;
unsigned int  sery;
unsigned int  se;
```

```
//#############################################################################
//     Main Function
//#############################################################################
void main(void) {

    SOPT=0x53;       //disable watchdog
    asm ("CLI");     //needed for the serial monitor
    IRQSC=0x00;              //disable interupts

 //Bus speed set to 18.874368MHz giving T = 53ns
    ICGC1 = 0x38;       //%00111000       // sets the micro to use the extrenal occilator Pg103 in Data sheet
    ICGC2 = 0x70;    //%01110000;    // sets the micro to use the extrenal occilator

    PTADD=0xFF;      // Set port A direction register for output
    PTAD=0xFF;       // Load bitpattern on LED's. LED's are active low.

//Set up the IO port directions for the LCD
    PTBDD=0xFC;
    ptr1 = conv_result;
    thou1 = conv1;
    thou2 = conv2;
    ATD_channel = 0;

   //Initialising the ADC
    ATDC  = 0xA4;    // Set ADC to be in continous mode
    ATDPE = 0x03;     //Set the PTB0/AD1P0 and PTB1/AD1P1 pin as an ADC inputs, rest of the port is IO
    ATDSC = 0x20;

//#############################################################################
//Sampling and Storing the initial values of the ADC
//#############################################################################
for (i=0;i<2;i++){
```

```
    ATDSC = 0x20;
  if (i==0){
     in1=ATDRH * 10;
     ATDSC = 0x21;
     while (!ATDSC_CCF);
     }
  if (i==1){
    in2=ATDRH * 10;
    ATDSC = 0x20;

  }

}

//#############################################################################
//     Main Sampling and Storing Function
//#############################################################################
for(;;)
  {
     while (!ATDSC_CCF);

     {

         if (ATD_channel == 0)
         {
            result[ATD_channel] = ATDRH;     //Sample ADC channel 0 and store value
            serx = ATDRH * 10;
            PTAD  =~ result[ATD_channel];    // Display converted ADC value on LEDs

            servox();

            ATD_channel = 1;               //Change ADC channel to channel 1
            ATDSC = 0x21;
         }
```

```
        else if (ATD_channel == 1)
        {
                                    //Sample channel 1 and store value
           result[ATD_channel] = ATDRH;
           PTAD  =~ result[ATD_channel];    // Display converted ADC value on LEDs
           sery = ATDRH * 10;

           servoy();

           ATD_channel = 0;
           ATDSC = 0x20;
        }
        else
        {
           ATD_channel = 0;                 // If any error occurs, make channel 0 the default
channel and continue
        }
      }
  }
}


//#########################################################################
//   =Function that calculates the X-Axis force and displays it on the LCD       =
//   =If Channel 0 ADC value(serx) > stored value(in1), then force is tensile    =
//   =If Channel 0 ADC value(serx) < stored value(in1), then force is compressive =
//   = X tensile equation is force = 0.031687x^3-0.13078x^2+ 0.13634x-0.0098829  =
//   = X Compressive equation is force = 0.091809x^2+ 0.049085x- 0.078959        =
//   = where x is the the difference between measured value at the time          =
//   =   - initial value                                                         =
//#########################################################################

void servox (void) {
```

```
    if (serx > in1){
       Diff1 = serx - in1;
       forcex = (31*Diff1*Diff1*Diff1)-(130*Diff1*Diff1)+(136*Diff1)-10;
       valA = valA + Diff1;
    }
    else if (in1 > serx){
       Diff1 = in1 - serx;
       forcex = (91*Diff1*Diff1)+(49*Diff1)-78;
       valA = valA - Diff1;
    }
    else{
       valA = valA;
       forcex = 0;

    }


//#############################################################################
    //This section Displays the force value on the LCD
    //The actual force is the displayed value divided by 1000
//#############################################################################
    duty_x();
    lcd_init();
    lcd_command (0x01);  //clear screen
    Convert_toBCD(forcex, thou1);
    lcd_string("ForceX :");
    lcd_string(Convert_toASCII(thou1));
    lcd_string("N");
}

//#############################################################################
//   =Function that calculates the duty-cycle needed to control the      =
//   =actuators and controls the resolution to +- 40 degree from their   =
//   =initial position                                                   =
```

```
//#############################################################################
void duty_x (void ){

    if (valA < 1100){

       valA = 1100;
       }
    else if (valA > 2300){

       valA = 2300;
       }
    else {

       valA = valA;
       }

    TPM1MOD = 23584;       // Set Timer 1 Modulo value
    TPM1C0V = valA;        // For a Duty Cycle in Channel 0 of TPM1
    TPM1C0SC = 0x28;

    TPM1C1V = valA;        // Channel value Register
    TPM1C1SC  =0x28;       // Timer x Channel n Status and Control Register (TPMxCnSC)
    TPM1SC    =0x0C;

}

//#############################################################################
//   =Function that calculates the Y-Axis force and displays it on the LCD       =
//   =If Channel 1 ADC value(sery) > stored value(in2), then force is tensile    =
//   =If Channel 1 ADC value(sery) < stored value(in2), then force is compressive =
//   =Y tensile force = -0.41076x^4+ 3.1353x^3  -8.1996x^2+9.2334x- 3.7508       =
//   =Y Compressive force = -0.06066x^3+ 0.3358x^2+ 0.09069x-0.023001            =
//   = where x is the the difference between measured value at the time          =
//   =   - initial value                                                         =
```

```
//#############################################################################
void servoy (void ){
          if (sery > in2){
        Diff2 = sery - in2;
        forcey          =          (-410*Diff2*Diff2*Diff2*Diff2)+(3155*Diff2*Diff2*Diff2)-
(8200*Diff2*Diff2)+(9233*Diff2)-3750;
        valB = valB + ((125 *Diff2)/255);
    }
    else if (in2 > sery){
        Diff2 = in2 - sery;
        forcey = (-60*Diff2*Diff2*Diff2)+(336*Diff2*Diff2)+(90*Diff2)-23;
        valB = valB - ((125*Diff2)/255);
     }
     else{
        valB = valB;
        forcey = 0;

     }


//#############################################################################
    //This section Displays the Force value on the LCD
    //The actual force is the displayed value divided by 1000
//#############################################################################
    duty_y();
    lcd_init();
    lcd_command (0xC0);  //clear screen
    Convert_toBCD(forcey, thou2);
    lcd_string("ForceY :");
    lcd_string(Convert_toASCII(thou2));
    lcd_string("N");

 }
```

```
//#############################################################################
//    =Function that calculates the duty-cycle needed to control the      =
//    =actuators and controls the resolution to +- 40 degree form their   =
//    =initial position                                                   =
//#############################################################################
void duty_y (void ){

     if (valB < 900){

         valB = 900;
         }
    else if (valB > 2450){

         valB = 2450;
    }
    else {

         valB = valB;
           }

     TPM2MOD = 23584;        // Set Timer 2 Modulo value
     TPM2C0V = valB;         // For a Duty Cycle in Channel 0 of TPM2
     TPM2C0SC = 0x28;

     TPM2C1V = valB;         // Channel value Register
     TPM2C1SC  =0x28;        // Timer 2 Channel n Status and Control Register (TPMxCnSC)
     TPM2SC    =0x0C;

}

//#############################################################################
//    =Function that converts a decimal number to BCD for display on LCD =
//#############################################################################
void  Convert_toBCD(unsigned int number, unsigned char* result_ptr)
```

```
{
    unsigned int remainder = 0;
    unsigned int remainder1 = 0;

    result_ptr[0]  = (number/1000)%10;
    result_ptr[1]  = (number/100)%10;
    result_ptr[2]  = (number/10)%10;
    result_ptr[3]  = number%10;

}
//###############################################################################
//    =Function that converts a BCD number to a string for display on LCD=
//###############################################################################
char*  Convert_toASCII(unsigned char* result_ptr)
{
    unsigned int i;

    for(i=0;i<4;i++)
    {
      result_ptr[i] = result_ptr[i]+48;
    }

    return result_ptr;
}
//###############################################################################
//                END OF PROGRAM
//###############################################################################
```

Programme calculates the forces on the rotational Pitch and Roll axes

```
//#########################################################################
//=   This program will sample pin ADC0 and ADC1. The ADC is set        =
//=   to continuous 8-bit mode. The value will be converted to a        =
//=   force and displayed on the LCD. The timers are used to control    =
//=   the actuator by providing a duty cycle.                           =
//#########################################################################

//#########################################################################
//     The Include Files
//#########################################################################
#include <hidef.h> /* for EnableInterrupts macro */
#include <MC9S08GT16A.h> /* include peripheral declarations */
#include <string.h>
#include <LCD.h>
#include <math.h>
//#########################################################################
//    Declaration of Functions used
//#########################################################################
void lcd_init (void);
void lcd_string (unsigned char *instring);
void lcd_command (unsigned char command);
void Init_ADC(void);
void paw(void);
void roll(void);
void delay5 (void);
void plat_init(void);
void Force(void);
void duty_change2(void );
void duty_change1(void );

void   Convert_toBCD(unsigned int number, unsigned char* result_ptr);
char*  Convert_toASCII(unsigned char* result_ptr);
```

```
//#############################################################################
//     Global VAriables - set in the direct page in the Ram
//#############################################################################

unsigned char ATD_channel;
unsigned char conv_result[4];
unsigned int  result [2];
unsigned int  rest;
unsigned int  valA;
unsigned int  A;
unsigned int  i;
unsigned int  B;
unsigned int  volt;
unsigned int  volt1;
unsigned int  valB;
unsigned int init_val;
unsigned int init_val1;
unsigned int Diff;
unsigned int Diff1;
unsigned int dum;
unsigned int force;
unsigned int valF;
unsigned int val_dummy;
unsigned int val;
unsigned char *ptr1;
unsigned int t;
unsigned char *thou;
unsigned char *thou1;
unsigned char *thou2;
//#############################################################################
//     Main Function
//#############################################################################
void main(void)
```

```
{
   SOPT=0x53;    //disable watchdog
   asm ("CLI");  //needed for the serial monitor
   //IRQSC=0x00;                 //disable interupts

 //Bus speed set to 18.874368MHz giving T = 53ns
 ICGC1 = 0x38; //%00111000          // sets the micro to use the extrenal occilator
ICGC2 = 0x70; //%01110000;    // sets the micro to use the extrenal occilator

   PTADD=0xFF;
   PTAD=0xFF;
 //Set up the IO port directions for the LCD
   PTBDD=0xFC;

   ptr1 = conv_result;
   ATD_channel = 0;

   //Init_ADC();
     ATDC  = 0xA4;
     ATDPE = 0x03; //Set the PTB0/AD1P0 pin as an ADC input, rest of the port is IO
     ATDSC = 0x20;
//#############################################################################
//Sampling and Storing the initial values of the ADC
//#############################################################################
for (i=0;i<2;i++){
  if (i==0){
        init_val=ATDRH * 10;
     ATDSC = 0x21;
     while (!ATDSC_CCF);
    }
  if (i==1){
     init_val1=ATDRH * 10;
     ATDSC = 0x20;
    }
```

```
 }
 //plat_init();
//#############################################################################
//     Main Sampling and Storing Function
//#############################################################################
for(;;)
 {
      while (!ATDSC_CCF);
       {
          if (ATD_channel == 0)
         {
            result[ATD_channel] = ATDRH;
            volt = ATDRH * 10;
            PTAD  =~ result[ATD_channel];    // Display ADC value on LEDs

            paw();
            lcd_command (0x01);
            Convert_toBCD(result[ATD_channel], ptr1);
            lcd_string("Result: ");
            lcd_string(Convert_toASCII(ptr1));

            ATD_channel = 1;
            ATDSC = 0x21;
         }
         else if (ATD_channel == 1)
         {
            result[ATD_channel] = ATDRH;
            PTAD  =~ result[ATD_channel];    // Display ADC value on LEDs
            volt1 = ATDRH * 10;
             roll();
            lcd_command (0xC0);
            Convert_toBCD(result[ATD_channel], ptr1);
            lcd_string("Result: ");
            lcd_string(Convert_toASCII(ptr1));
```

```
            ATD_channel = 0;
            ATDSC = 0x20;
        }
        else
        {
            ATD_channel = 0;
        }

      }
//#####################################################################
  }
}
void Force(void)
{
   unsigned int remainder1 = 0;
   int th;
   int h;
   int te;
   int u;

   th = (force/1000);
   h = (force/100);
   remainder1 = (force%100);
   te = (remainder1/10);
   u = force%10;

   thou[0]= h;
   thou[1]= te;
   thou[2]= u;
   thou[3]= '\0';


//#############################################################################
   //This section loads the force value on the LCD
```

```
   //The actual force value is the displayed value divide by 1000
//#############################################################################
   lcd_init();
   lcd_command (0x01);  //clear screen
   lcd_string("Force(N):");
   lcd_string(Convert_toASCII(thou));

}
//#############################################################################
//    =The function calculates the force on the Pitch axis                    =
//    =Pitch forward force = 0.66701x- 0.035844                               =
//    =Pitch backward force = -0.11371x^2+ 0.58561x+ 0.04575                  =
//    =where x is the difference between the measured value atthat instance and =
//    =the initial value recorded at the start as the zero value              =
//#############################################################################
void paw(void)
{
      unsigned int remainder2 = 0;
      int hun;
     int ten;
     int un;
     int thu;
     int r = 0;

 while (r<30){
     if (volt > init_val){
        Diff = volt - init_val;
        force = (666*Diff)-35;
        valA = valA + Diff;
        //init_val = valA;
     }

      else if (init_val >volt){
        Diff = init_val - volt;
```

```
        force = (-113*Diff*Diff)+(586*Diff)+45;
        valA = valA -Diff;
        //init_val = valA;
      }
// Otherwise dont change the value
      else {
          valA = volt;
          force = 0;
          //duty_change1();
      }
      r++;
 }
      Force();
      duty_change1();
}
//interrupt 8
//#########################################################################
//    =The function calculates the force on the Roll axis                =
//    =Roll forward force = 0.60405x -0.053394                           =
//    =where x is the difference between the measured value atthat instance and =
//    =the initial value recorded at the start as the zero value         =
//#########################################################################

void roll(void)
{
  int p = 0;
   while (p<30){
     if (volt1 > init_val1){
        Diff1 = volt1 - init_val1;
        force = (604*Diff1)-35;
        valB = valB + Diff1;
        init_val1 = valB;
      }
```

```
    /*  else if (init_val1 > volt1){
        Diff = init_val1 - volt1;
        force = (((3*Diff*Diff) + (670*Diff) +45000)/1000000);
        //valB = valB - Diff1;
        init_val1 = valB;
      }*/
// Otherwise dont change the value
      else {
          valB = volt1;
          force = 0;
      }
      p++;
 }
      duty_change2();



}
//interrupt 8
//#########################################################################
//duty change1 and 2 set the resolution of the actuator movement to
// +- 40 degrees from their initial position
//#########################################################################
void duty_change1(void )
{
     if (valA < 600){
          valA = 600;
        }
    else if (valA > 1800){
          valA = 1800;
    }
    else {
         valA = valA;
          }
```

```
    TPM1MOD = 23584;       // Set Timer 1 Modulo value
    TPM1C0V = valA;        // Channel 0 of TPM1
    TPM1C0SC = 0x28;

    TPM1C1V = valA;        // Channel value Register
    TPM1C1SC  =0x28;       // Timer x Channel n Status and Control Register (TPMxCnSC)
    TPM1SC    =0x0C;
}

void duty_change2(void)
{
    if (valB < 600){
       valB = 600;
       }
   else if (valB > 1800){
       valB = 1800;
       }
   else {
           valB = valB;
           }
    TPM2MOD = 23584;       // Set Timer 1 Modulo value
    TPM2C0V = valB;        // Channel 0 of TPM1
    TPM2C0SC = 0x28;

    TPM2C1V = valB;        // Channel value Register
    TPM2C1SC  =0x28;       // Timer x Channel n Status and Control Register (TPMxCnSC)
    TPM2SC    =0x0C;
}
// 0.5 second delay
void delay5 (void)
//a delay used by the LCD functions.
{unsigned int count1;
unsigned int count2;
 for (count1=0;count1<500;count1++) {
```

```
     for(count2=0;count2<1200;count2++);
 }
}
void  Convert_toBCD(unsigned int number, unsigned char* result_ptr)
{
    unsigned int remainder = 0;
     result_ptr[0]  = (number/100);
    remainder      = number%100;
    result_ptr[1]  = (remainder/10);
    result_ptr[2]  = (remainder%10);
    result_ptr[3]  = '\0';

}
char*  Convert_toASCII(unsigned char* result_ptr)
{
    unsigned int i;

    for(i=0;i<3;i++)
    {
      result_ptr[i] = result_ptr[i]+48;
    }
        return result_ptr;
}

void Init_ADC(void)
{
        ATDC  = 0xA4;
        ATDPE = 0x01;
        ATDSC = 0x20;
}

//#############################################################################
 //plat_init was used to reset the timers (actuator) to their initial position
 //when intially debugging the programme
```

```
//#############################################################################
void plat_init(void)
{
    TPM1MOD = 23584;      // Set Timer 1 Modulo value
    TPM1C0V = 1200;       //  Channel 0 of TPM1
    TPM1C0SC = 0x28;

    TPM1C1V = 1200;       //;Channel value Register
    TPM1C1SC  =0x28;      //;Timer x Channel n Status and Control Register (TPMxCnSC)
    TPM1SC    =0x0C;

    TPM2MOD = 23584;      // Set Timer 1 Modulo value
    TPM2C0V = 1200;       // Channel 0 of TPM2
    TPM2C0SC = 0x28;

    TPM2C1V = 1200;       //;Channel value Register
    TPM2C1SC  =0x28;      //;Timer x Channel n Status and Control Register (TPMxCnSC)
    TPM2SC    =0x0C;


}

//#############################################################################
 //                          END OF PROGRAM
//#############################################################################
```

Program calculates the up-down forces on the Z axis

```
//#########################################################################
//=   This program will sample pin ADC0 and ADC1. The ADC is set       =
//=   to continuous 8-bit mode. The value will be converted to a       =
//=   force and displayed on the LCD. The timers are used to control    =
//=   the actuator by providing a duty cycle.                           =
//#########################################################################

//#########################################################################
//     The Include Files
//#########################################################################
#include <hidef.h> /* for EnableInterrupts macro */
#include <MC9S08GT16A.h> /* include peripheral declarations */
#include <string.h>
#include "LCD.h"
#include <math.h>

//#########################################################################
//    Declaration of Functions used
//#########################################################################
void lcd_init (void);
void lcd_string (unsigned char *instring);
void lcd_command (unsigned char command);
void Init_ADC(void);
void   Convert_toBCD(unsigned int number, unsigned char* result_ptr);
char*  Convert_toASCII(unsigned char* result_ptr);
void UP_DOWN(void);
void Force(void);
//#########################################################################
//     Global VAriables - set in the durect page in the Ram
//#########################################################################
unsigned int  weight;
unsigned int  z;
```

```
unsigned int  i;
unsigned int  Diff;
unsigned int  force;
unsigned int  new_val;
unsigned int init_val;
unsigned char ATD_channel;
unsigned char conv_result[4];
unsigned int  result [2];
unsigned char *ptr1;
unsigned char *thou;
unsigned char *thou1;
unsigned char *thou2;

//#############################################################################
//     Main Function
//#############################################################################
void main(void) {

       SOPT=0x53;              //disable watchdog
       asm ("CLI");            //needed for the serial monitor
    //IRQSC=0x00;               //disable interupts

 //Bus speed set to 18.874368MHz giving T = 53ns
       ICGC1 = 0x38;           //%00111000   sets the micro to use the extrenal occilator
       ICGC2 = 0x70;           //%01110000;  sets the micro to use the extrenal occilator

       PTADD=0xFF;
       PTAD=0xFF;
 //Set up the IO port directions for the LCD
       PTBDD=0xFC;

       thou = conv_result;
//Init_ADC();
       ATDC  = 0xA4;
```

```
      ATDPE = 0x03; //Set the PTB0/AD1P0 pin as an ADC input, rest of the port is IO
      ATDSC = 0x20;
//#########################################################################
//Sampling and Storing the initial value of the ADC
//#########################################################################
for (i=0;i<2;i++){
  ATDSC = 0x20;
  weight = ATDRH;        //get the initial weight value from the initial ADC value
  while (!ATDSC_CCF);
 }
 //#########################################################################
//    Main Sampling and Storing Function
//#########################################################################
for(;;) {
        z = ATDRH;
        PTAD  =~ ATDRH;    // Display ADC value on LEDs
        UP_DOWN();

      }
}
//#########################################################################
//   =Function that calculates the Z-Axis force and displays it on the LCD       =
//   =Only the upward force is considered and calculated in this program         =
//   = y=  -0.014087x^3+ 0.14675x^2  -0.097381x-0.0037956  where y = force and x  =
//   = is the voltage from ADC
//#########################################################################
void UP_DOWN(void)
{
     if(z>weight){
     Diff = z - weight;
     force = -(14*Diff*Diff*Diff) + (146*Diff*Diff)-(97*Diff) - 3;
     }
     else{
     force = 0;
```

```
    }


//#############################################################################
    //This section Displays the value on the LCD
    //The actual force is the value displayed divided by 1000
 //#############################################################################
     lcd_init();
     lcd_command (0x01);  //clear screen
     Convert_toBCD(force, thou);
     lcd_string("Up_Force :");
     lcd_string(Convert_toASCII(thou));
     lcd_string("N");


}

//#############################################################################
//    =Function that converts a decimal number to BCD for display on LCD
//#############################################################################
void  Convert_toBCD(unsigned int number, unsigned char* result_ptr)
{
   //result_ptr[0]  = (number/1000)%10;          //(out/1000)%10;
   result_ptr[0]  = (number/100)%10;
   result_ptr[1]  = (number/10)%10;
   result_ptr[2]  = number%10;

}
//#############################################################################
//    =Function that converts a BCD number to a string for display on LCD
//#############################################################################
char*  Convert_toASCII(unsigned char* result_ptr)
{
   unsigned int i;
```

```
    for(i=0;i<3;i++)
    {
      result_ptr[i] = result_ptr[i]+48;
    }
  return result_ptr;
}
//#############################################################################
//                  END OF PROGRAM
//#############################################################################
```