# MEng Project: Flying Brain

Scott Alexander
s.alexander@student.reading.ac.uk
Cybernetics MEng
*School of Systems Engineering, The University of Reading, UK*

**Abstract**

This document explores the general concepts that are necessary to develop a skeleton, sensing system and aspects of a control model for a quadrotor unmanned aerial vehicle (UAV). The application of a recursive least squares algorithm for systems identification has been researched and applied to a nonlinear brushless RC motor to provide significant results that have allowed improvements within two different controllers. The construction of the skeleton has also been discussed after research into material properties and material reduction methods, to produce a lightweight but strong product that could accommodate a National Instruments sbRIO-9612 control board. For future developments, even though the board does provide the required functionality for a UAV, it is however too large, too heavy and requires too much power to be used practically for the purposes of a UAV. Finally the implementation of a cheap and lightweight sensing system incorporating an array of ultrasonics has been implemented showing that they can be used within a noisy environment as long as precautions are taken to reduce the disturbances that they may receive.

## 1. Contents

## 2. Introduction

In recent years quadrotor unmanned aerial vehicles (UAVs) have become popular within a variety of applications, such as military search and rescue and surveillance operations, primarily due to their vertical take-off and landing capabilities. The miniaturisation of components and reduction in costs have also made them more viable within more commercial markets, such as media and government services including the police and fire brigade. In fact on the 12th February 2010, the first arrest directly linked to the use of a quadrotor UAV took place, operated by the UK police force [1].

Another factor that makes quadrotor UAVs desirable is that they only consist of four moving parts as opposed to helicopters that require many more. Technically this makes them easier to manage, as non-moving parts tend not to fail due to general wear and tear. However, with the reduction of control-based parts, high-speed processors and modern inertial measurement units (IMUs) are vital to manage motors speeds to produce stability and adjust to account for disturbances [2].

However, as with all new technologies, quadrotor UAVs that are designed for commercial use come with a large price tag. The most advanced commercial retailer Dragonflyer Innovations has a series of models which start with the x4 [3] priced at aprroximatly $10,000 USD and move up to the new x8 for which expected prices are estimated to be approximately $40,000 USD. Cheaper alternatives include Parrots AR drone priced at £299 [4], which does show that stable control can be achieved on a cheaper budget, but does require user interaction for movement.

The factor that allows companies to push their prices higher is the type of control that is incorporated onto the UAV; models that are designed for the toy market only need to assist the stability of the product. However, models that are designed for commercial or military use are generally built to be completely self-controlled, which requires not only a stability control but also a pre-programmed routine that will perform a task while taking disturbances into account.

Disturbances can come in many forms, such as wind which will affect the stability control, but to a vehicle that requires no user intervention, physical objects such as walls can also be considered as a disturbance to the system. Objects therefore need to be detected for basic object avoidance. Two methods that currently exist on the market are ultrasonics [5] the cheaper of the two, and visual algorithms [6] that require more processing power.

Therefore this project aims to investigate control methods of quadrotor UAVs and construction process to create a stable hovering flying platform. Other areas to investigate include environment detection methods and lightweight material testing to produce a skeleton and object avoidance system suitable for flying within a budget of £1750.

## 3. Social, Health and Safety and Ethical Issues

Through the specification of the project a few issues became apparent in the realm of health and safety, ethics and social concerns. The majority existed within health and safety and were as follows:

- Propellers rotating at high speeds can cause considerable damage to a user; therefore protection must be taken into account within the design.

- Testing stages may produce unexpected flying movements; therefore the craft should be kept in a separate room and coupled to a test rig.
- The UAV should also contain a remote kill switch to turn it off instantly if a problem arises.
- Electrical safety precautions must be taken; therefore the craft will never be powered directly from the mains.
- The UAV will be kept away from fire hazards to prevent damage to the craft and to prevent any damage to combustible components such as its power source.
- All cables on the craft will be secured in a safe position away from the blades to prevent failure to components and electrical shock hazards.
- If a developer needs to perform construction work on the craft, power will be disconnected or dangerous components removed if power is necessary.
- The batteries will be drained slightly after each charge as to prevent surges.

Issues that become apparent in the legal, social and ethical domain were as follows:

- All professional software that needs to be used in the development of the control will have a licence purchased
- The UAV will be used within UK legislation and will not be used to inflict harm or violate an individual's privacy.
- Components that could inflict harm will be treated with the utmost respect and components that could violate an individual's privacy will only be used if no other component can perform in the same manner.
- The UAV will only be used indoors to avoid air traffic control issues.

## 4. Team Management

### *4.1 Detailed Specification*

Through initial research of the teams skill sets, suitable objectives were discussed that could be achieved over the course of twenty weeks. These objectives were as follows…

- Design and implement a simple control strategy (PID) that enables the UAV to achieve stable hover at 300mm for 1 minute.
- Design and implement an advanced control strategy (LQR) that enables the UAV to achieve stable hover at 300mm for 1 minute.
- Create a model of the UAV and apply it to a simulation.
- To build the UAV to be as light and strong as possible.
- To interface sensors for environment detection and use readings within a simulated environment.
- To develop a user friendly interface.

*4.2 Task allocation*

To meet the specifications each team member was allocated specific tasks based on their skills and desired area of interest as well as a team role that can be seen in Table 1.

**Table 1.** Task Allocation and Breakdown

| Member | Team Role | Control | Systems Identification | Electronics | Mechanical Design |
|---|---|---|---|---|---|
| Scott Alexander | Project Manager | Distance sensing integration | Motor testing | | Creation of UAV,<br>Material Analysis |
| Ian Roberts | Vice Manger and Health and Safety | | Distance Sensing | Daughter board development,<br>Device integration | |
| Ruth White | Secretary | PID Control,<br>LQR Control,<br>IMU integration | | | |
| Joe Warren | Treasurer | Daughter board integration | 3D modelling,<br>Distance modelling | | |
| Mark Ince | Marketing | Altitude Control,<br>Control integration,<br>User Interface development | | | |

Each member was also allocated a certain amount of time for each section using a Gannt chart so that progress could be monitored and milestones could be announced, so individual components of the project could be linked together without any delay.

## 5. Research

*5.1 Quadrotor UAV Design Principles*

The simple design used for most modern UAVs consists of four motors positioned at an equal distance from the centre of the craft in a cross formation. This allows the centre of gravity (COG) of the craft to be located directly in the centre of the cross formation, allowing each motor to have an equal force on the craft when running at the same speed.

The motors are paired such that two opposite motors move in a clockwise direction, while the other two motors move in an anti-clockwise direction (Figure 1). This is due to the gyroscopic forces of the blades created from torque, which if unbalanced will cause the UAV to turn on the spot, therefore if one motor increases in speed, its opposite counterpart must decrease in speed to maintain a gyroscopic force balance.

**Figure 1.** Simple design of a quadrotor UAV

However, even with a simplistic theory of flight for quadrotors, brushless motors, which are generally used, act in a non-linear fashion which makes matching their speeds difficult. Brushless motors also have a delay in their response to signals which has to be taken into account within the controller.

*5.2 Kinematics and Dynamics*

*5.2.1 Stable Hover*

For a UAV to hover in a stable position it must produce an upwards thrust (Ft) equal to the downwards forces acting upon it (m.g). All four motors must also produce an equal upwards force ($F1$ *to* $F4$) to prevent any rotation about any axis as well as to prevent turning due to gyroscopic forces. This can be described mathematically as shown in equations (1) and (2)

$$m.g = Ft(hover) \tag{1}$$

$$where \quad \frac{Ft(hover)}{4} = F1 = F2 = F3 = F4 \tag{2}$$

Further disturbances are likely to be present, such as wind or turbulence from the air being moved by the blades; however this can be accounted for within the UAV's control algorithms.

*5.2.2. Roll, Pitch and Yaw*

Roll, pitch and yaw are the names of the three movements that can occur along the three dimensional axis in which the UAV can move.

Roll occurs when the left motors speed is increases and the right motors speed is decreased, resulting in a movement on the sides of the craft but has no effect on the vertical

thrust. The same occurs if the motors' increasing and decreasing speeds are on the opposite sides but is considered a negative roll.

Pitch is a similar motion as roll, but occurs between the front and back end motors and as a result causes the UAV to move backwards or forwards with no effect on the vertical thrust. A forwards movement is considered positive and a backwards movement is considered negative.

Yaw is the movement that is controls the turning of the craft and works by controlling all four motors in pairs. It is explained in more detail in section 4.2.4. The planes in which the movements act on can be seen in Figure 2.

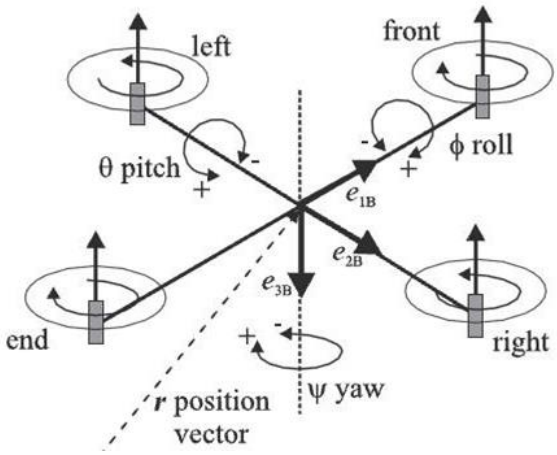


**Figure 2.** – Roll, Pitch and Yaw acting on a Quadrotor UAV [7]

*5.2.3 Forwards Movement*

To implement forwards movement on a quadrotor UAV requires knowing the roll, pitch and yaw accurately through the use of an accelerometer (present within an IMU). Forwards movement occurs when the craft is stable at an angle (desired pitch ($\theta$)). The larger the angle is to the horizontal axis of the UAV the faster the craft will move forwards. This can be seen in Figure 3.

Maintaining a constant forwards speed is difficult due to disturbances which will constantly affect the angle in which the craft is hovering at. However a control can take this into account and monitor the crafts speed such that excessive speed does not occur.

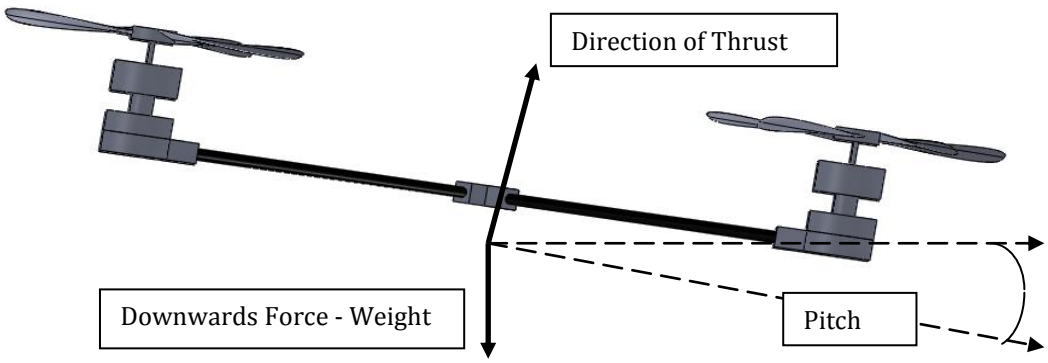


**Figure 3.** Forwards movement during tilt.

*5.2.4 Turning*

Turning occurs by taking advantage of the gyroscopic forces occurring due to torque in the motors. As the motors have counter rotating pairs, turning can occur by speeding up two opposite motors while reducing the speed of the other two motors such that the total force Ft is still the same. As the gyroscopic force is increased in one direction by two faster motors and decreased in the other by two slower motors, the resultant force causes the UAV to turn. The desired properties can be described mathematically as shown in (3), (4) and (5).

$$Ft(hover) = Ft(Turning) \tag{3}$$

$$where \quad F1 + F3 = x\,.\,(F2 + F4) \tag{4}$$

$$and \quad (F1 = F3)\,\&\,(F2 = F4) \tag{5}$$

$x$ is a scalar that should be chosen such that it produces small changes in the speed of the motors leaving the craft less susceptible to noise.

*5.2.5 Complex Dynamic Model*

The analysis of the UAV so far has been of a simple model but there are more factors to take into account, such as the inertia of the craft. However, a model by Bouabdallah [8] incorporates the additional parameters. The reason why Bouabdallahs method of dynamics was selected is because it assumes gyroscopic effects are negligible, as well as ground and blade flapping effects. The list of variables that the model uses can be seen in Table 2.

**Table 2.** Quadrotor model variables.

| Unit | Description |
|---|---|
| $x, y$ | Position vector |
| $z$ | Altitude |
| $\phi$ | Roll |
| $\theta$ | Pitch |
| $\psi$ | Yaw |
| $U_1$ | Collective thrust |
| $U_{2,3,4}$ | Roll/Pitch/Yaw moment |
| $\Omega$ | Rotor disturbance |
| $\Omega_{F,S,A,P}$ | Rotor speed in Forward/Starboard/Aft/Port |
| $I_{x,y,z}$ | Inertia along roll/pitch/yaw axis |
| $J$ | Rotor inertia |
| $L$ | Distance between motor and vehicle center of mass |
| $m$ | Vehicle mass |
| $b$ | Thrust factor |
| $d$ | Drag factor |

The body of the craft is assumed to be rigid and therefore maintains a fixed frame $e_B$, which can be seen in Figure 2. The axis of the body is also linked in with an inertial frame; $e_1, e_2$ and $e_3$ by a position vector $r^T = (x, y, z)$ and three angles in Euler form $\Omega^T = (\phi, \theta, \psi)$. The vectors of the body's fixed frame are then transformed into the inertial frame by a rotation matrix R (7).

$$R = \begin{pmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \sin\psi\cos\theta & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{pmatrix} \quad (7)$$

The thrust of the motors can also be broken down into the thrust factor b of the device as well as the rotational speed of the motor $\omega_i$ as can be seen in (8) in which $i = 1, 2, 3, 4$ to represent the four motors on the craft.

$$F_i = b \cdot \omega_i^2 \quad (8)$$

Then the total thrust produced from the four rotors is:

$$T = \sum_{i=1}^{4} |F_i| = b \sum_{i=1}^{4} \omega_i^2 \quad (9)$$

Leading on from this, the acceleration of the craft can be found by differentiating the position vector r, producing equation (10).

$$\ddot{r} = \begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix} = g \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} - R\frac{T}{m}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad (10)$$

With the inertia matrix $I$ (11).

$$\ddot{r} = \begin{pmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{pmatrix} \quad (11)$$

As well as the vector M (12) which models the torque applied to the craft.

$$M = \begin{pmatrix} Lb(\omega_3^2 - \omega_4^2) \\ Lb(\omega_1^2 - \omega_2^2) \\ d(\omega_1^2 + \omega_2^2 - \omega_3^2 - \omega_4^2) \end{pmatrix} \quad (12)$$

And $M_g$ containing the gyroscopic torques of the system.

$$M_G = I_R \left( \dot{\Omega} \times \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right) \cdot (\omega_1 + \omega_2 - \omega_3 - \omega_4) \quad (13)$$

Can all be used along with the rotor inertia $I_R$ to produce a second set of differential equations as can be seen in (14).

$$I \cdot \ddot{\Omega} = -\left(\dot{\Omega} \times I \cdot \dot{\Omega}\right) - M_G + M \tag{14}$$

Using the two differential equations the inputs to the system can then be re-described as (15).

$$\begin{aligned} U_1 &= b(\Omega_P^2 + \Omega_S^2 + \Omega_A^2 + \Omega_P^2) \\ U_2 &= b(\Omega_P^2 - \Omega_S^2) \\ U_3 &= b(\Omega_P^2 - \Omega_A^2) \\ U_4 &= d(\Omega_S^2 + \Omega_P^2 - \Omega_A^2 - \Omega_P^2) \\ \Omega &= -\Omega_P + \Omega_S - \Omega_A + \Omega_P \end{aligned} \tag{15}$$

And following on, the complete system can be described as (16).

$$\begin{aligned} \ddot{x} &= (\cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi)\tfrac{1}{m} U_1 \\ \ddot{y} &= (\cos\phi \sin\theta \cos\psi - \sin\phi \cos\psi)\tfrac{1}{m} U_1 \\ \ddot{z} &= -g + (\cos\phi \cos\theta)\tfrac{1}{m} U_1 \\ \ddot{\phi} &= \dot{\theta}\dot{\psi} C_1 - C_4 \dot{\theta}\Omega + C_6 U_2 \\ \ddot{\theta} &= \dot{\phi}\dot{\psi} C_2 + C_5 \dot{\phi}\Omega + C_7 U_3 \\ \ddot{\psi} &= \dot{\theta}\dot{\phi} C_3 + C_8 U_4 \end{aligned} \tag{16}$$

In which the constants within the nonlinear equations are shown in (17).

$$\begin{aligned} &C_1 = \tfrac{I_y - I_z}{I_x},\ C_2 = \tfrac{I_z - I_x}{I_y},\ C_3 = \tfrac{I_x - I_y}{I_z},\ C_4 = \tfrac{J}{I_x}, \\ &C_5 = \tfrac{J}{I_y},\ C_6 = \tfrac{L}{I_x},\ C_7 = \tfrac{L}{I_y},\ C_8 = \tfrac{L}{I_z} \end{aligned} \tag{17}$$

### *5.3 Material analysis*

#### *5.3.1. Lightweight Materials*

Three materials show qualities that are needed to produce a lightweight yet strong skeleton and they are aluminium, plastic and carbon fibre.

Aluminium is a material that provides good levels of strength at a thickness of 1mm which is suitable for supporting components into place. It is also extremely light at this thinness but does become heavier when more strength is needed for dealing with forces, such as with the motors in the UAVs case. Aluminium is also durable so can withstand the force of multiple crashes during testing stages.

Plastic can provide the same amount of strength as aluminium at approximately half the weight, however it also very brittle. If plastic is to be used then it should be applied to non-vital areas that do not play a role in the structural support. However, since weight is a major factor within the design of the UAV there is not going to be many non-vital areas.

Finally carbon fibre is known for being able to withstand a large amount of stress and is generally used in kites due to this. Therefore, carbon fibre is suitable for areas of the UAV that will be subjected to large forces and may be subjected to deflection.

*5.3.2. Weight Reduction Techniques*

The primary way to reduce weight is to only use the amount of material that is necessary. However, even by reducing the amount that is being used, some areas will need to be thicker, to provide more support. However, even where the material is thicker sections can be removed from it to reduce its weight and maintain the same strength. Such an example of this is with iron bars, where the bar is cut to an I shape, removing the majority of the material. Another method in thin but long pieces of material is to cut multiple holes along its central line, maintaining strength along the entire piece of material, while cutting out nearly 40% of the material (Figure 4).

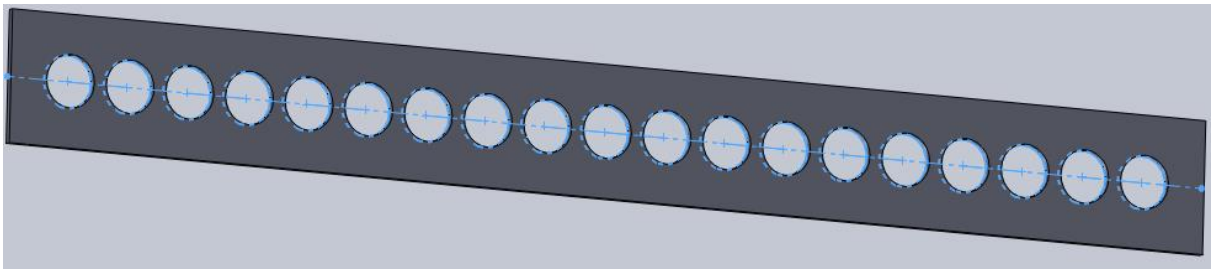

**Figure 4.** Weight reduction method in thin pieces of material

These methods also make manufacturing cheaper due to the reduced amount of material, which reduces the cost of the UAV if parts need to be made in an industrial environment.

*5.4. Systems Identification using RLS algorithms*

Due to brushless motors having a non-linear behaviour, as well as expected delays when increasing and decreasing speeds, due internal resistances and signal delays from the controller, a model needs to be acquired. This model needs to describe how the motors respond to the input signal of the system and produce an accurate value for what the motors speed actually is.

One method is to use a recursive least squares (RLS) algorithm [9] that works by using recorded data sets of the systems input and output signals and finds the coefficients of the systems transfer function through a series of iterations. There are four calculating parameters to an RLS [7], $\varepsilon$ is the modelling error (18), which is the difference between the measured output and predicted output. K is the weighting factor, (19) showing how much $\varepsilon$ will affect the parameters. P is a convergent matrix (20) and $\bar{\theta}$ is the model coefficient estimation (21).

$$\varepsilon(n) = y(n) - \phi^T(n)\bar{\theta}(n-1) \tag{18}$$

$$P(n) = P(n-1) - \frac{P(n-1)\phi(n)\phi^T(n)P(n-1)}{1+\phi^T(n)P(n-1)\phi(n)} \tag{19}$$

$$K(n) = P(n)\phi(n) \tag{20}$$

$$\hat{\theta}(n) = \hat{\theta}(n-1) + K(n)\varepsilon(n) \quad (21)$$

The number of coefficients to be used also needs to be provided to the algorithm in which the number provided can have a drastic effect on the system. One affect is that the stability of the calculated systems can change so it should also be analysed to make sure that they are suitable to implement.

If problems do occur within the learning processes a forgetting factor (λ) can be applied to the algorithm that essentially controls the number of values to use when calculating the next iteration. This helps as only the most recent data is used and only requires a small change to the algorithms code in parameter P (22).

$$P(n) = \frac{1}{\lambda}\left(P(n-1) - \frac{P(n-1)\phi(n)\phi^T(n)P(n-1)}{\lambda+\phi^T(n)P(n-1)\phi(n)}\right) \quad (22)$$

One problem that may occur using this algorithm is that it is designed for linear data, therefore knowing that the motors act in a non-linear fashion, linear sections of the data may need to be extracted from any collected data pending on initial testing results.

### *5.5 Ultrasonic Theory*

#### *5.5.1. General Theory of Ultrasonic Rangefinders*

Ultrasonic rangefinders operate by sending out a pulse of high-pitched sound around the 40-60 KHz range through a transmitter and then listen for the noise to rebound off an object back into its receiver. The time in which it takes to rebound can be recorded and therefore the distance can be calculated, as the speed of sound is practically constant at 343ms.

However, ultrasonics are prone to noise, rangefinders operating at 60KHz generally receive a lot of interference from indoor lighting, which operates at a similar frequency. Therefore they should not be used on indoor-based devices that will operate when lights are on.

In general, background noise can also cause interference with the device and one noisy component on the UAV is its motors. This could cause problems as if noise emitted from the motors is within the operating frequency, then the device will pick up miscellaneous readings. However, if the motors do produce noise at the operating frequency, the speed at which it is produced can be avoided.

The motors will also have an effect on the air density and movement around them. This in return produces a further problem for the ultrasonics as the speed and direction in which sound pulses are sent and received may be affected. Therefore the best way to avoid this problem is to have all ultrasonics a good distance away from the motors.

The benefits of using ultrasonic are that they are cheap and simple to implement, while they also provide a large detection range that is not only directly ahead, but slightly to the side of the device as well. The side detection has a range of approximately 60 degrees at the devices furthest detection point, increasing to 120 degrees range of detection and its closest detection point. The detection range of a typical ultrasonic range finder can be seen in Figure 5.

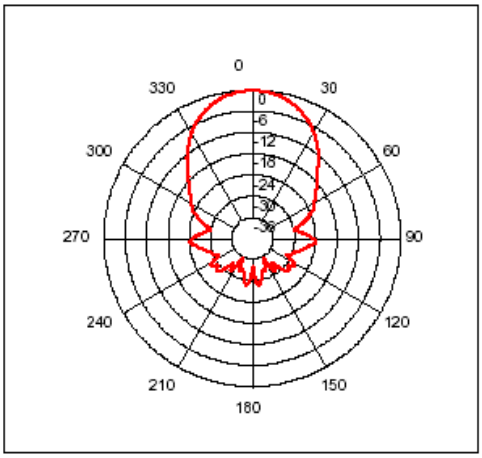


**Figure 5.** Typical detection range of a standard ultrasonic rangefinder [10]

*5.5.2. Time Division Multiplexing*

One serious problem that can occur when using multiple ultrasonic rangefinders close to each other is Time Division Multiplexing (TDM). TMD with ultrasonics occurs when a sound pulse from one ultrasonic device is received by a second ultrasonic device creating an incorrect reading [11]. This can occur when a signal rebounds off a wall towards the second receiver and is detected after a pulse is just emitted from that rangefinder showing that an object is closer than it really is (Figure 6).

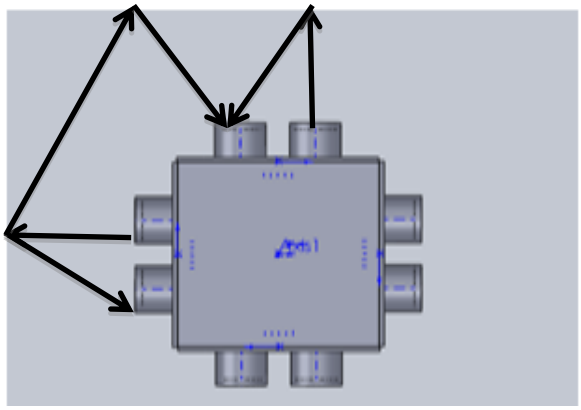

**Figure 6.** First signal being received by a second device

One way to prevent this problem is to trigger ultrasonic rangefinders at alternating times with a long enough delay between each trigger such that the original sound chirp diminishes. This can slow down the rate in which data can be acquired, however if it is not taken into consideration, the UAV will detect objects randomly, which may prevent it moving correctly or could cause a collision when avoiding an object that is not present.

*5.2. Health and Safety*

*5.2.1 Motor Guards*

The first thing to take into account with a quadrotor UAV is how to protect damage to its motors in case of a collision, as well as how to protect the user from any injuries. The

simplest way is to place a guard around the blades connected to the motors. However, there are a range of different designs on the market each having different pros and cons.

The first popular industrial design consists of each of the blades being surrounded with a ring. This protects each of the motors well and if damage does occur, it's likely that only one of the motors will be damaged, making the model easier and cheaper to fix. However, by putting a ring around each motor, weight is added to the craft (Figure 7).

**Figure 7.** Quadrotor design one [12]

Another popular design implements a single ring around the whole of the UAV at the blades height level (Figure 8). More damage can occur if the protection fails, however it is much lighter than the first design. Having a ring around the whole UAV also prevents objects getting caught in the UAVs blind spots, which can occur with design one.

**Figure 8.** Quadrotor design two [13]

The final design incorporates the best of both designs by implementing four thinner individual rings round each blade, as well as a square guard around the whole UAV preventing objects moving into the craft's blind spots (Figure 9).

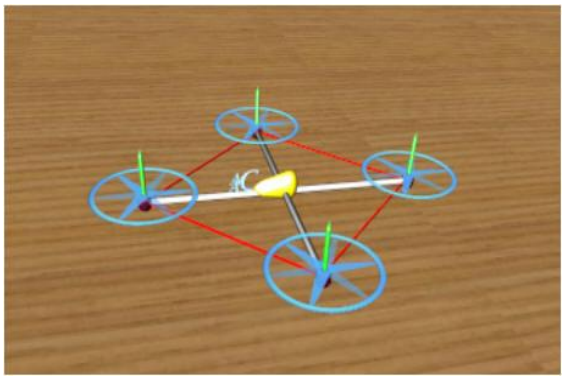

**Figure 9.** Quadrotor design three [14]

### *5.6.2 Power Cut Off Procedures*

If a collision occurs, the UAV needs to turn itself off quickly to minimise as much damage as possible. One way to do this is to incorporate a relay on to the craft that is connected to the power source linked to the motors. The relay has to be connected to the motors power source as linking it to the controller will not stop the motors instantly.

The relay can then be connected to either a switch that the user can use in an emergency, or a touch sensor can be implemented on the craft such that if a collision occurs the UAV will cut itself out automatically. Both methods should ideally be used in case one of the methods fails.

## 6. Hardware

### *6.1. Control Board*

The control board to be used on the UAV is a National Instruments sbRIO-9612 [15] which has 32 analogue input pins as well as 110 digital input/output pins, which provide plenty of connections for attaching devices. However, the board does weigh 0.26kg and has an operating voltage between 19v to 24v, which is a large amount for a controller. The board is also an FPGA so it can be directly programmed using a software package called LABVIEW.

LabView uses a graphical program method such that individual modules can be created, interfaced and simulated before actually being programmed to the board. The software is also intuitive enough to produce a graphical user interface (GUI) as each module is produced.

### *6.2 Motors*

The motors that have been selected are brushless EMAX CF2812, primarily as their data sheet states that they should be able to produce enough force to lift approximately 0.6kg each. This is produced with 6x4 inch propellers, which reduces the size of the UAV as well as the weight, as each motor weighs 0.031kg. However, they are only stated as being 85% efficient, therefore tests will need to occur to determine their actual thrust.

If the motors do each lift the weight specified then the craft should come to a total weight of approximately 1.8kg as to provide a large amount of leeway in thrust for the controller to be able to operate correctly.

### *6.3 Power source*

Due to the power requirements of the National Instruments sbRIO board being between 19V to 24v, a large power source must be provided. This power source must be able to power the board as well as all four motors, while at the same time being as light as possible. To meet these needs two 11.1v Thunder RC 5000mAh three cell lithium-ion polymer batteries were selected. This model of battery was selected as they only weigh 0.36kg which saves a lot of weight compared to other batteries. However, using two will still add a considerable amount

of weight to the craft. If problems do occur then it is possible to tether the UAV's controller to a power source and use one battery to power the four motors for testing purposes.

*6.4 Inertial Measuring Unit*

The IMU is the fundamental device that will be measuring the pitch, yaw and roll gyroscopic outputs. The model that is being used is a Razor – ultra thin (SEN – 09431) [16] which incorporates a dual axis pitch and roll analogue gyroscope (LPR530AL), yaw rate gyroscope (LY530ALH) and a 3 axis accelerometer (ADXL335). Combining these components together gives the device 6 degrees of freedom which each have a 300mV/g sensitivity as well as a + or – 3g range. This will provide enough sensitivity to accurately measure the position of the UAV and is also fast enough to provide the data at suitable time frames to the controller.

*6.5 Ultrasonics*

To provide the UAV with enough sensors to detect objects within its environment a total of seven ultrasonic rangefinders shall be used. Four will be used around the body to detect side objects, one on top of the craft to make sure a collision with the ceiling does not occur and two on the bottom so that an average distance from the ground can be taken. An average is necessary as when the UAV is moving forwards the distance it will detect will be at an angle, therefore if one ultrasonic is positioned at the front of the craft (the lower section) and one on the rear (highest section), an average of the two will produce a value that is more accurate (Figure 9).

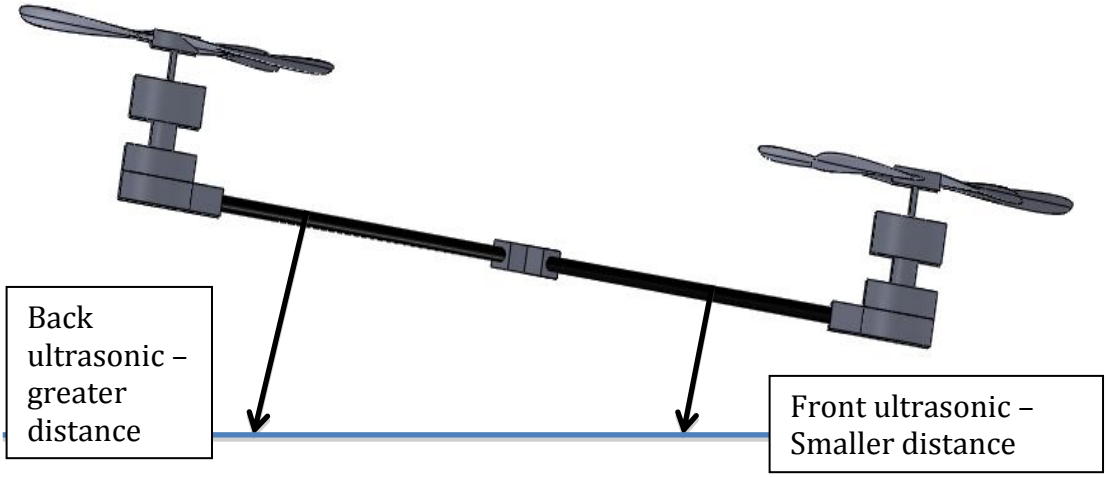


**Figure 9**. Demonstration of Ultraosnics working when craft is at an angle

It's not a perfect method and cannot work at height, due to trigonometric rules, or at large angles. However, ultrasonics have a limited range and the craft is not expected to tilt at a large angle which makes this method practical.

The device that has been selected to be used is a SRF05 rangefinder [10] which operates by receiving a short pulse for 10us, which then triggers the transmitter of the device to send out 8 short cycles of sonic bursts at 40 kHz (Figure 10). The receiver then listens for the response and returns a pulse whose length is proportional to the distance an object is away from the device. This pulse is between 100us to 25ms and will time out after 30ms providing a total distance range of four meters. However, to what accuracy it can detect up to four meters will need to be tested.

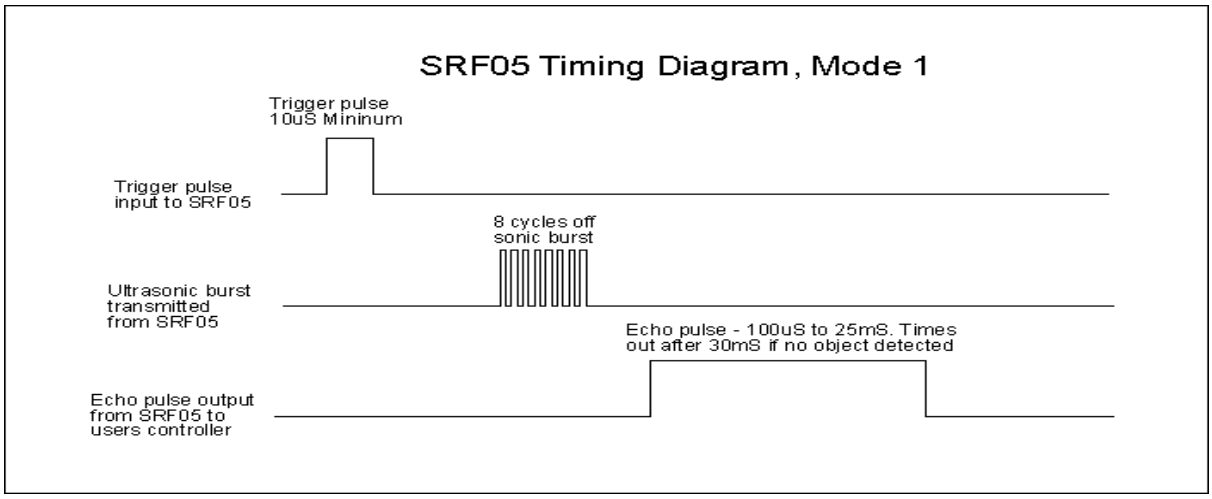


**Figure 10.** Ultrasonic operation signals [10].

*5.6. Connection Schematic*

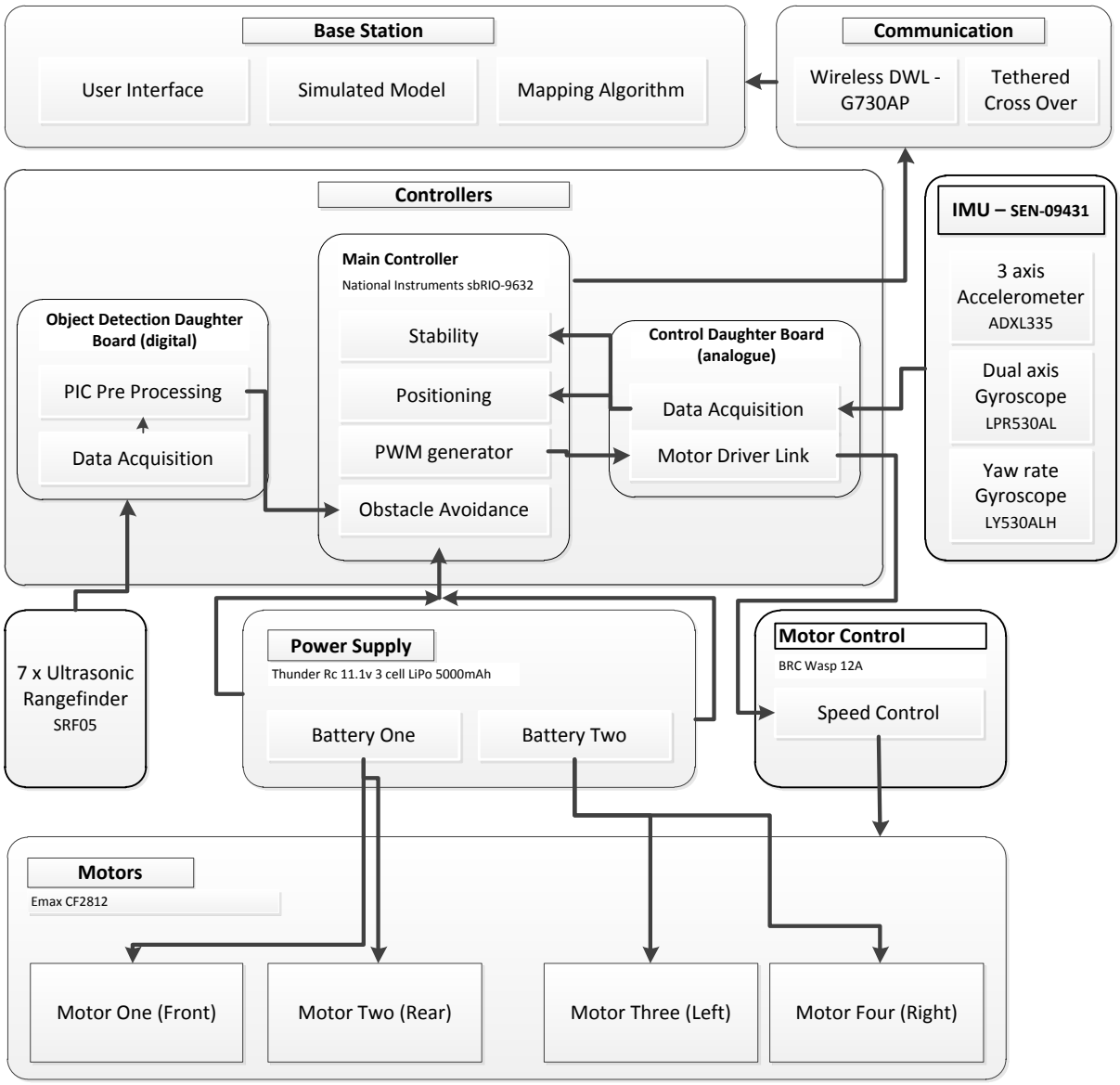


**Figure 11.** Topology view of the UAVs systems and components

## 7. Motor Analysis and Systems Identification

The first bit of information that was necessary to gain in development was how much force a motor could produce. To acquire this information, the motors were attached to a loadcell with its readings being feedback to a terminal.

### *7.1. Loadcell Test Set Up*

The Loadcell was connected to a Quansar 4 PCI board through a coaxial cable. The PWM signal being emitted from sbRIO board was also adapted to be attached to a coaxial cable and feed into a second port on the Quansar board, so both bits of data could be recorded simultaneously. The set up can be seen below (Figure 12).

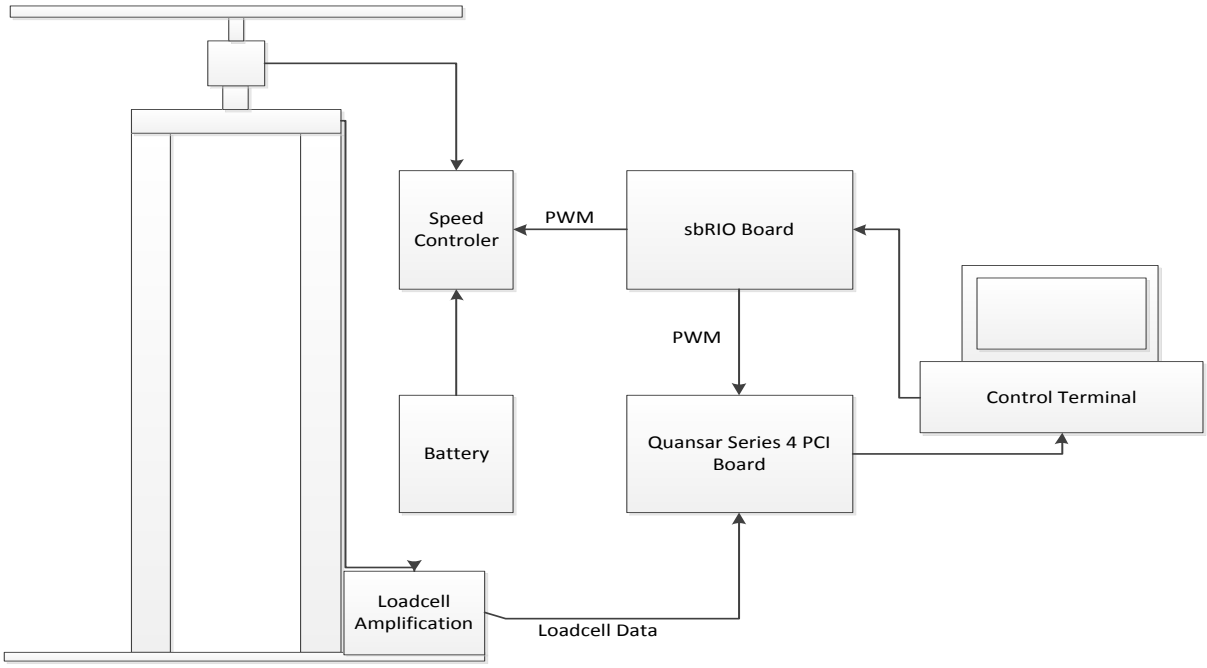


**Figure 12.** Loadcell Configuration

### *7.2. Weight Calibration*

Following the set up when values were being read graphically through Simulink, the returned voltages were compared when different weights were placed on top of the loadcell. The results can be seen in Table 3.

**Table 3.** Loadcell readings with different weights applied

| Weight (kg) | Loadcell Voltage (V) |
|---|---|
| 0 | 2.25 |
| 0.1 | 2.20 |
| 0.2 | 2.15 |
| 0.3 | 2.10 |
| 0.4 | 2.05 |

What can be seen from the results is that for every 100 grams there is a change of 0.05V, which means that acquired data from the motors can be converted into thrust.

### *7.3. Data Gathering and Analysis*

To see how the motors react under different conditions, different types of inputs where applied to the motors and its data was recorded. The inputs that were applied consisted of a step, sin, random 100ms, random 200ms, random 500ms and random 1000ms signal. The data was recorded every 0.001s so as to make the readings as accurate as possible and to produce smooth graphs.

Once the data was acquired, it then had to be coded to represent how much weight the motor was lifting and what PWM was being sent to the motor. The PWM had been sent through a second port of the sbRIO board pre coded into values ranging from 0.3V to 3.3V so that the quasar board could distinguish the signal easily. To turn the data into meaningful values the following transformations were performed.

$$\text{Loadcell signal} = -2.25 \times 200 + 150 \tag{23}$$

$$\text{PWM signal} = \times 10 \div 3.3 \tag{24}$$

The minus 2.25 in transformation in (18) is removing the weight of the motor connection block from the data as well as the weight of the motor. Looking at the data after these transformations showed that the maximum weight that each motor could lift was 0.46kg which was significantly less than expected, which meant weight reduction of the body was a big priority.

### *7.4. RLS Development*

The initial step to implementing an RLS algorithm was to apply formulas (13) to (16) within MATLAB where the initial coefficients of the system could be set and reviewed after the algorithm had finished working to make sure that the algorithm was working as expected. The code that was used can be found in Appendix A. Once the code had been implemented it was run with the coefficients of the system being a = [-1.5363 0.8607] and b = [0.0416 0.395] the graphical response can be seen in Figure 13.

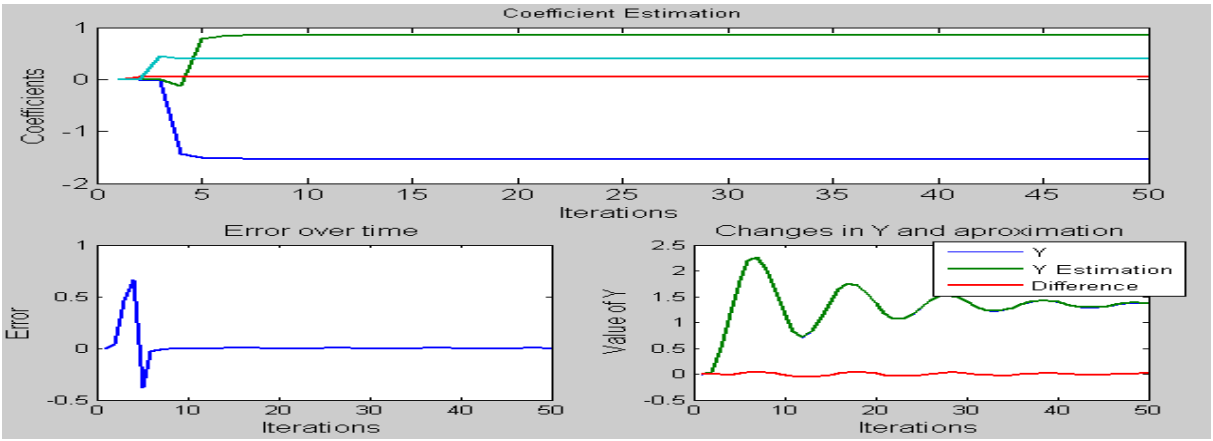


**Figure 13.** Results of initial RLS algorithm

The numerical results of A and B were A = [-1.5351 0.8596] and B = [0.0420 0.3948] once the algorithm had finished which is extremely close to the original coefficients of a and b. It can also be seen that the algorithm has found the system response after six iterations which shows that it's very fast however the system is working with a linear model.

The code was further adapted (Appendix B) to work with other datasets and to analyse the stability of the system produced as to make sure it is practical to apply to the motors. The results for a sin PWM input can be seen in Figure 14 calculating towards using two a and two b coefficients.

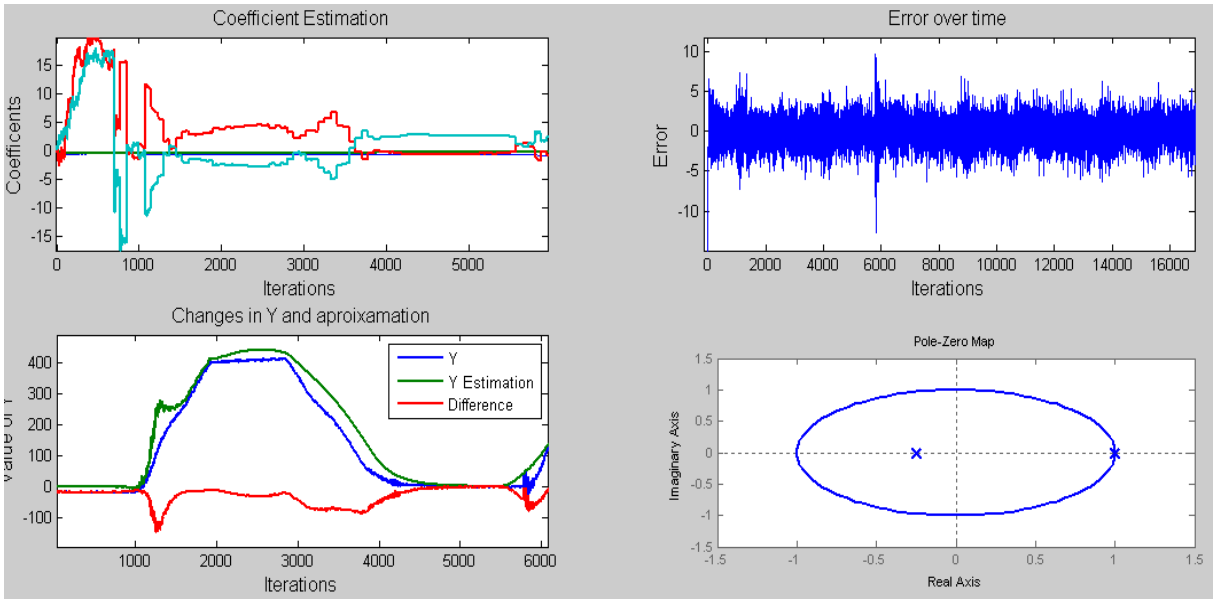


**Figure 14.** RLS results using motor results data to a sin input (2a and 2b)

What can be seen from these graphs is that the error over time is pretty much constant but the RLS has managed to estimate the response of the motors fairly well. One thing to take note of is that the stability of the system even with only two a and two b coefficients is critically stable. The system was further analysed with 3 a and 3 b coefficients (Figure 15).

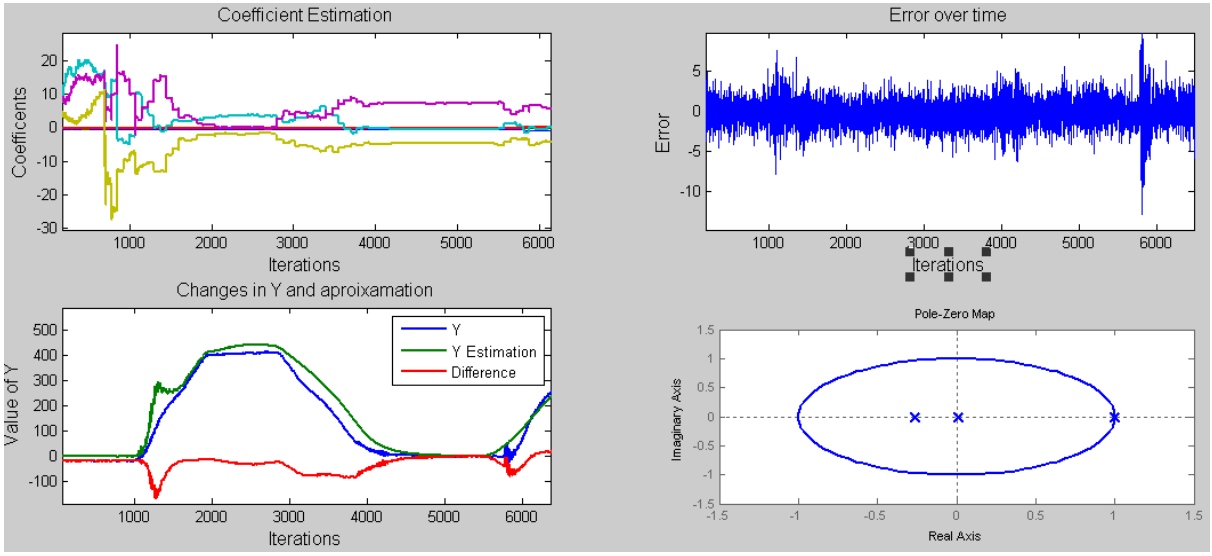


**Figure 15.** RLS results using motor results data to a sin input (3a and 3b)

The results shown in Figure 15 are fairly similar to Figure 14 which shows that the stability of the system is affecting the way in which the system can be modelled. Therefore, to keep things simple a 2a and 2b based coefficient model was used to simplify the model. Taking the calculated a and b coefficients the following transfer function for the system was produced (25).

$$R(z) = \frac{-2.963z + 1.093}{0.7461z + 0.2492} \quad . \tag{25}$$

Using the random data acquired produced an unusable result which means that to implement them a second method of analysis had to be incorporated.

*7.5. Further Systems Identification using MATLABs Ident Toolkit*

To incorporate the random signal data sets MATLABs systems identification toolkit ident was used. This provided a few benefits as opposed to the implemented code; firstly the toolkit returned a percentage match which assisted in visual matching. Secondly a second data set can be applied to the results for validation, allowing analysis of the results with other types of inputs. Therefore each data set was applied to an ARMAX RLS algorithm using a range of different numbers of a,b,c and d coefficients to see which systems produced the best matches. The results can be seen in Table 4.

**Table 4.** Complete results for each data sets percentage match

**Sin input data**

| Validation = sin | % | Validation = random 100ms | % | Validation = random 500ms | % | Validation = random 1000ms | % |
|---|---|---|---|---|---|---|---|
| amx8674 | 74.8 | amx6666 | 24.12 | amx8888 | 30.47 | amx8888 | 27.43 |
| mx9765 | 72.9 | amx9876 | 23.42 | amx6666 | 30.44 | amx7777 | 27.33 |
| amx11967 | 72.8 | amx9999 | 23.26 | amx10101010 | 30.33 | amx10964 | 27.22 |
| amx11867 | 71.8 | amx10101010 | 23.2 | amx7777 | 30.31 | amx10101010 | 26.76 |
| amx111067 | 71.4 | amx10964 | 23.07 | amx10964 | 30.22 | amx9999 | 26.68 |

**Random 100ms input data**

| Validation = sin | % | Validation = random 100ms | % | Validation = random 500ms | % | Validation = random 1000ms | % |
|---|---|---|---|---|---|---|---|
| 8888 | 21.5 | 111067 | 24.15 | 11967 | 30.56 | 8888 | 27.43 |
| 11967 | 21.5 | 6666 | 24.12 | 111067 | 30.52 | 9765 | 27.4 |
| 12867 | 21.4 | 9876 | 23.42 | 8888 | 30.47 | 11967 | 27.35 |
| 7777 | 21.4 | 10978 | 23.37 | 6666 | 30.44 | 7777 | 27.33 |
| 10974 | 21.2 | 8544 | 23.31 | 12867 | 30.42 | 10974 | 27.3 |

**Random 500ms input data**

| Validation = sin | % | Validation = random 100ms | % | Validation = random 500ms | % | Validation = random 1000ms | % |
|---|---|---|---|---|---|---|---|
| 9999 | 47 | 11867 | -18.73 | 11967 | 48.14 | 111067 | 12.12 |
| 9876 | 47 | 111067 | -18.76 | 11867 | 48.12 | 11967 | 12.12 |
| 10101010 | 47 | 11967 | -18.84 | 9765 | 48.12 | 12967 | 12.11 |
| 10964 | 46.9 | 10674 | -18.92 | 111067 | 48.11 | 9765 | 12.09 |
| 8664 | 46 | 9765 | -18.93 | 6666 | 48.11 | 11867 | 12.09 |

**Random 1000ms input data**

| Validation = sin | % | Validation = random 100ms | % | Validation = random 500ms | % | Validation = random 1000ms | % |
|---|---|---|---|---|---|---|---|
| 11067 | 26.2 | 12967 | 16.72 | 111067 | 27.61 | 111067 | 29.12 |
| 11967 | 26.1 | 8554 | 15.15 | 11967 | 27.11 | 11967 | 29.08 |
| 12867 | 25.9 | 8674 | 14.49 | 9999 | 26.44 | 9999 | 29.06 |
| 8888 | 25.8 | 9876 | 14.34 | 12867 | 26.36 | 12867 | 29.03 |
| 9999 | 25.8 | 10101010 | 13.46 | 12967 | 26.27 | 10101010 | 29.01 |

Following this the number of occurrence of each systems was tabulated by the number of times it appeared in the top 5, the number of times it was the second best system, and the number of times it produced the best system. The top systems can be seen in Table 5.

**Table 5.** Occurrence of each system in the top five and totals of occurrences

| System | Occurrences | First | Second | Totals |
|---|---|---|---|---|
| 11967 | 10 | 2 | 5 | 17 |
| 111067 | 9 | 5 | 2 | 16 |
| 8888 | 6 | 4 | 0 | 10 |
| 9999 | 6 | 1 | 0 | 7 |
| 6666 | 5 | 1 | 2 | 8 |
| 11867 | 4 | 1 | 1 | 6 |
| 12967 | 3 | 1 | 0 | 4 |
| 8674 | 2 | 1 | 0 | 3 |
| 10101010 | 6 | 0 | 0 | 6 |
| 9765 | 5 | 0 | 2 | 7 |
| 12867 | 5 | 0 | 0 | 5 |

Therefore taking the best system with 11 a coefficients, 9 b coefficients, 6 c and 7 d coefficients produces the transfer function shown in (26).

*Transfer function:*

$$R(z) = \frac{-2.48\,z^8 + 4.797\,z^7 - 4.915\,z^6 + 4.252\,z^5 - 3.202\,z^4 + 4.108\,z^3 - 5.334\,z\text{^}2 + 5.745\,z - 2.97}{z^{11} - 2.908\,z^{10} + 4.618\,z^9 - 5.297\,z^8 + 4.303\,z^7 - 2.552\,z^6 + 0.8556\,z^5 - 0.1088\,z^4 + 0.07951\,z^3 - 0.1038\,z^2 + 0.209\,z - 0.09577} \quad (26)$$

*Sampling time:* 0.001

Both transfer functions can now be applied to both the PID and LQR controller to provide a better form of control over the motors' speeds.

## 8. Skeleton Design

### *8.1. Weight Restriction*

The primary problem to be aware of is the weight of the UAV. Through the systems identification it was found that each motor could lift 0.46kg which means that in total the craft could weight an absolute maximum of 1.84kg. However the key components on the craft already come to 1.35kg which can be seen in Table 6.

**Table 6.** Listing of key components

| Device | Description | Weight (g) |
|---|---|---|
| NI RIO board 9612 | Main Processor Provided (19v-24v) | 260.8 |
| 2xBatteries | Main power source providing 11v | 361 each |
| 7xSRF05 | Ultrasonic range finders | 30 each |
| 4 x EMax CF2812 | Motors | 41 each |

Therefore there is only 0.49kg left for the skeleton, but this cannot all be used as the motors cannot run on full power or they will not be able to produce any thrust for stabilisation control. In reality the skeleton had to be between 0.25kg to 0.3kg to provide this additional thrust.

### *8.2. Central Support Design*

The central support section is vital to the overall design of the UAV, therefore there are a few aspects that must be met:

- Firstly it must be large enough and strong enough to support the NI sbRIO board.

- There must be a good support structure to maintain the motors at right angles to each other.
- The method of connection to the motors must be rigid and strong enough to prevent deflection of the connection when a force of up to 5N is applied.
- It must also be lightweight.

The final design incorporates all of the above and uses hollow carbon fibre rods to support the motors, which should be strong enough by themselves; however two square supports have been included to reduce deflection in the rods (Figure 16).

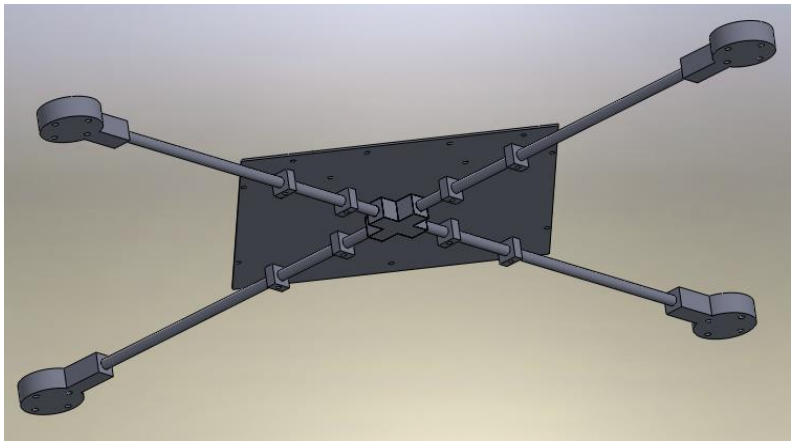

**Figure 16.** Central Support structure of the UAV

*8.3. Carbon Fibre Force Simulation*

To make sure that the carbon rods would be suitable to take the force of the motors, the design was subjected to a simulation express analysis in Solidworks to determine how much deflection would be present in the rods. The properties for the carbon fibre rod was discovered online [17] and applied to the simulation as well as 9.8N (1kg on the ends), which produced the results shown in Figure 17.

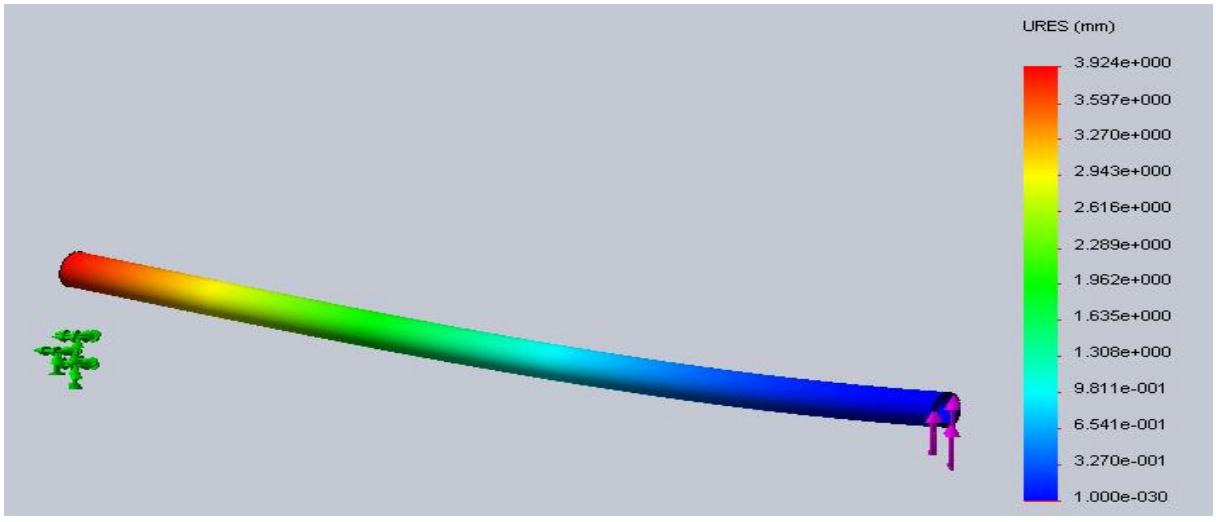


**Figure 17.** Displacement analysis of Carbon Fibre Rods

The results show that when a kg is applied to the ends of the rod a displacement of 3.924mm occurs. This is more than suitable as this is double the actual maximum thrust that

the motors can produce, therefore it is expected that at full thrust there would be a deflection of just 1.45mm. This should not have any major effects of the flight kinematics.

### *8.4. Outer Ring Adaption and Ultrasonic connection*

For health and safety reasons an outer ring is necessary to provide protection to the motors as well as the users of the UAV. However, connecting the ring to the UAV with the current simple design is tricky as the ring must be held securely to prevent any wobbling, which could be incorporated as noise within the controller. The ring is also on the outer most part of the craft, making it the best place to have some form of connection for the ultrasonics to attach to. Therefore, the ring should also be held at a distance far enough away from the motors to make noise from the motors undetectable to the ultrasonics.

The problem in producing this part is providing the strength necessary to hold the ring steady. Plastic could be used as it would provide the strength needed and is light weight but the brittleness of the material would not allow multiple collisions which may occur in the flight tests. Aluminium on the other hand adds additional weight as it becomes thicker to provide additional strength. However, some of this weight can be reduced by incorporating circular cut outs in non-vital areas. The design that was used can be seen in Figure 18.

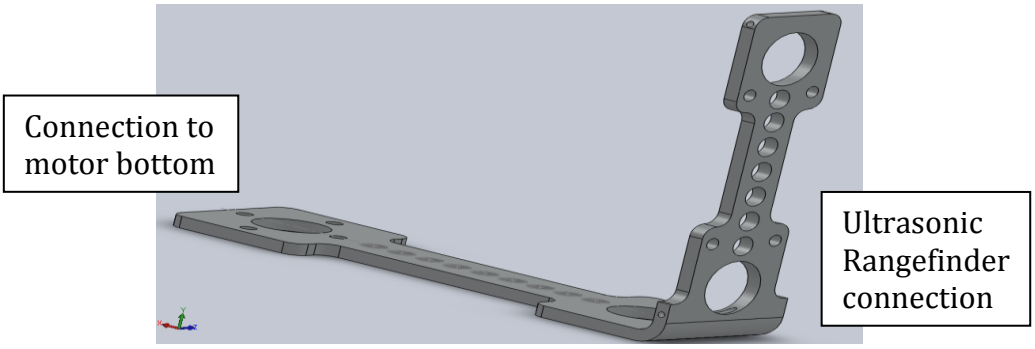


**Figure 18.** Outer Ring connector

Solidworks estimated this part at being only 0.022kg which again is suitable considering the amount of strength it provides to support the outer ring.

### *8.5. Final Design*

The final design of the UAV can be seen in Figure 19 which incorporates the previous aspects as well as a few extra features for landing and connection to a test rig.

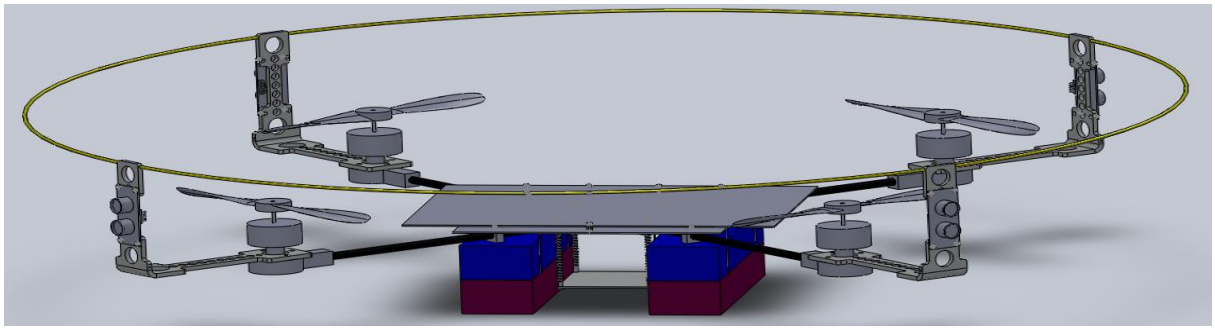

**Figure 19.** Final UAV design

Two foam pads have been incorporated underneath the batteries to cushion any impact during landing and take-off. An ultrasonic is also positioned inside each of the foam pads for the altitude controller, protecting them as well as reducing noise. There is also a square coupling pad underneath the craft, so that the craft can be connected to a test rig. The total weight of the skeleton was approximated by Solidworks to be 0.295kg which is at the upper limit of the weight restriction.

### *8.6. Test Rig*

A test rig has been designed to couple to the bottom of the UAV via a ball joint to allow as much freedom in movement as possible. The ball joint is further connected to a pole linked to a joint with 360 degrees of freedom allowing the UAV to move in any direction it wants around a circular circumference. The test rig can also be counter weighted at its opposite end if weight problems do occur, due to the little leeway there is for the controller, then extra weight can be added purely for testing purposes. The design of the test rig can be seen in Figure 20.

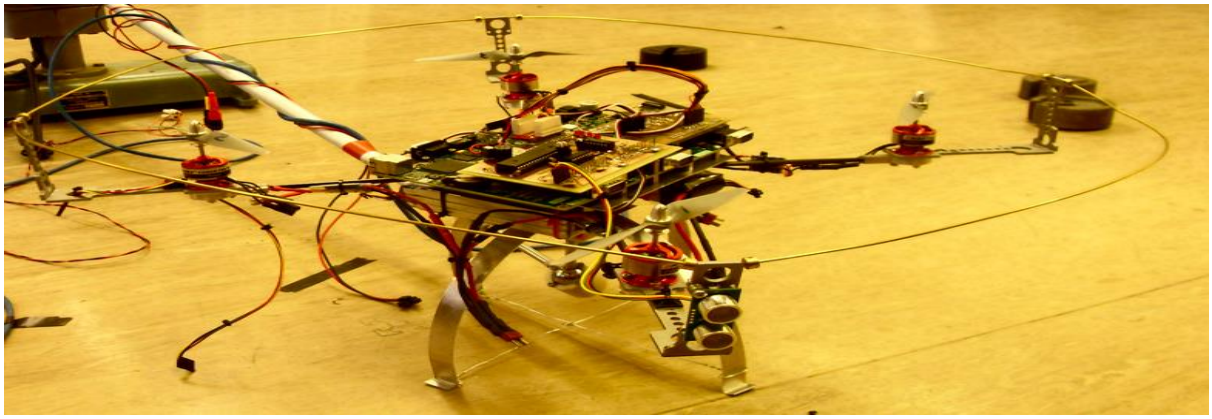

**Figure 20.** Solidworks Design for a Test Rig

### *8.7. Final Construction*

The final construction was made with very few problems and matched the design accurately except for the weight which was slightly heavier at 0.31kg. A problem did occur with the production of the outer ring connector as a hole was drilled towards the bend to provide room for a second safety ring however, during the bending of the part the hole split. Therefore, if a second safety ring is needed, the drilling of the second hole needs to be done after the bend so as not to damage the part.

A second problem occurred when attaching the test rig to the UAV, the foam pads being used for the landing gear were not big enough to provide room for the ball joint underneath the coupling pad. Therefore four legs were attached to the coupling pad during the testing process to overcome this problem. The final construction attached to the test rig can be seen in Figure 21.

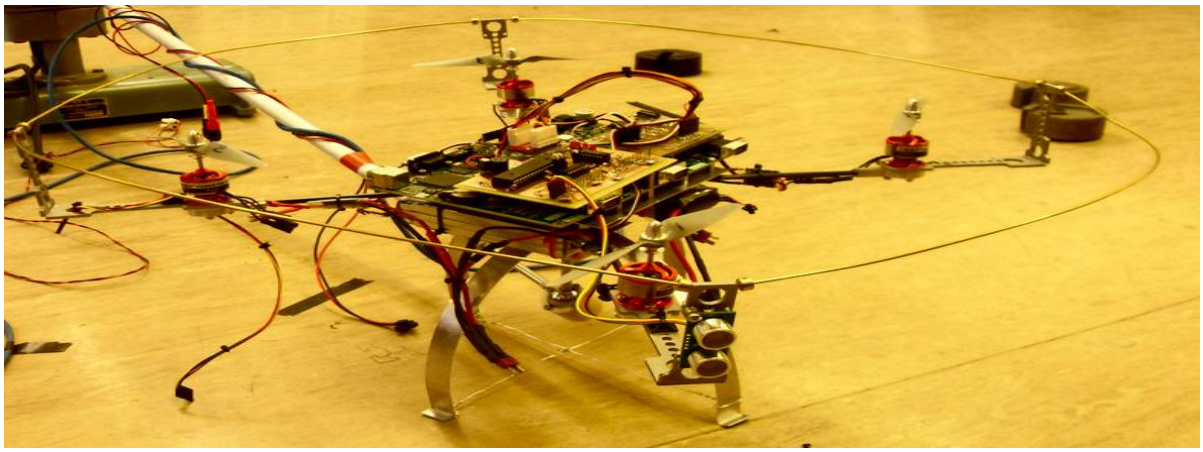

**Figure 21.** Constructed UAV as of 27/03/2011

## 9. Implementation of the Ultrasonic Object Detection System

### *9.1. Ultrasonic Device Positioning*

The ultrasonics have been positioned within the UAV's design to prevent noise from the motors affecting them, as well as to increase the distance between each device to prevent TDM. However, to eliminate the problem completely opposite ultrasonics can be triggered to eliminate TDM completely such that ultrasonics 1, 3, 5 and 7 are triggered first, followed by a delay before the rest of the ultrasonics are triggered (Figure 22).

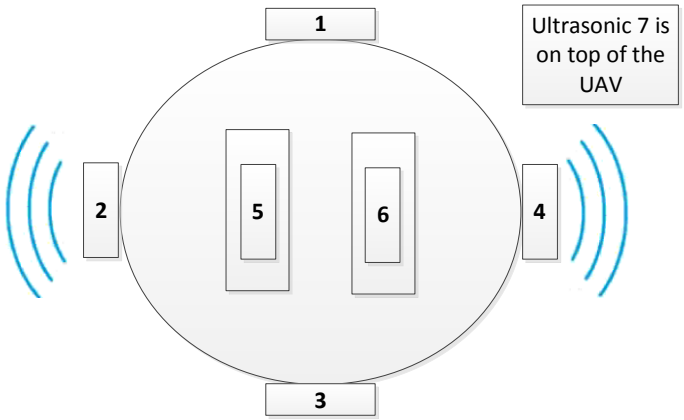


**Figure 22.** Ultrasonic positions from the bottom of the craft

### *9.2. Ultrasonic Operation Schematic*

To keep the design as cheap and simple as possible, a PIC was selected to be used with a shift register, and a 4x dual input or gate to trigger and receive signals from the seven ultrasonic range finders. Therefore by using a shift register, the PIC only needs to provide five digital pulse signals and by incorporating the response of the ultrasonics into the 4x dual input or gate means that the PIC only needs to have 4 input capture pins (Figure 23).

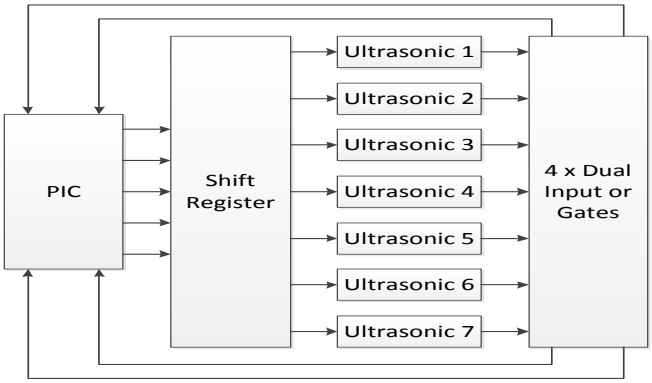


**Figure 23.** Ultrasonic Array Schematic

*9.3. PIC Implementation*

A dsPIC30F4011 [18] was selected to be used due to its properties of running at 7.37MHZ, having four input capture modules and has the ability to implement UART communications The PIC was then programmed by breaking each task down, the tasks were as follows.

- Implementation of the shift register such that it could trigger four ultrasonics followed by the last three ultrasonics after a short delay.
- Implementation of four input capture modules and four timers such that a pulse could be recorded and timed from each of the outputs of the 4x dual input or gates.
- Implementation of a UART serial communication line to return the seven times of the ultrasonics responses as well as a packet header and footer to identify each packet on the receiving end.

*9.3.1. Shift Register Implementation*

The first aspect that was implemented was the shift register trigger. The device operates by setting an output pin to the shift register to a 0 or 1, and then switching a clock line to feed the values into its serial section of flip flops. Once set, the parallel section is then switched, sending a pulse out to the ultrasonics as needed. For the purposes of the array, four ones are initially input into the shift register followed by three zeros and then reset and repeated triggering four and then three ultrasonics respectively. This method eliminates chirps being read by another ultrasonic, as long as opposite ultrasonics are connected correctly to the ports.

*9.3.2. Input Capture Implementation*

The second aspect that was implemented was the input capture modules. The method that has been used is for the input capture pin to detect a falling edge, such that when the ultrasonics signal is triggered, a 16 bit timer will start and then stop when a falling edge is detected. For this to occur within a certain range, a timer pre scale must be set to return a specific clock value. Since the PIC can handle 16 bit values, a range between 100 to 30000 is suitable for the ultrasonics data, allowing room for error margins. To accomplish this, a timer prescale of 8 was calculated and implemented within the code that would produce values within this range when the processor is operating at its maximum speed.

*9.3.3 UART serial communication Implementation*

To implement the UART communications, the packet size was determined by the size of the timers which were 16bit. Therefore for a packet to be meaningful, it was designed to start with the value of 1 and finish with a value of 2. If any errors occurred with the ultrasonics then a value of 40000 would be returned. All of these values are outside the bounds of the data range so that the sbRIO board could detect them easily. Therefore, as each part of the packet is 16bits long then in total each packet consists of 144 bits, which meant that for the ultrasonics data to be relevant to the control, it had to be sent fast. To ensure that the data was sent fast, a Baud Rate (pulses per second) of 38400 was selected, which would effectively transmit a single pulse every 0.2μs. The design of the packet can be seen in Figure 24.

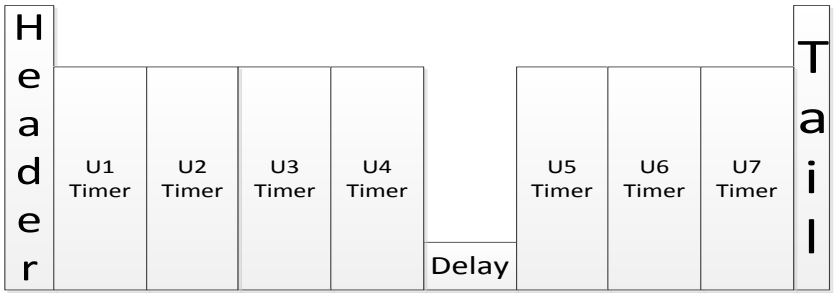


**Figure 24.** Packet design to be sent on the serial line

Only the transmission line of the UART was implemented as the sbRIO board would not need to communicate with the PIC, only interpret the information that was sent to it.

The full implemented code can be seen in Appendix C.

*9.4. Problems that Occurred*

Only one problem occurred using this method of operation, which was that the shift register operates as expected, but had compatibility issues with the ultrasonics, causing the response of the device to produce many falling edges over a short period of time. This as a result meant that the PIC could not get an accurate value for the distance and could not be changed to deal with a random varying number of falling edges.

Therefore to overcome this problem, the shift register was removed from the circuitry and was replaced with seven digital lines from the PIC, going directly to each ultrasonic rangefinder. The reason why the shift register was used was to save pins on the PIC such that a smaller version could be implemented at a later date, making the design cheaper and lighter. The amended code to implement the new method can be found in Appendix D.

*9.5. Results of Sensing System*

*9.5.1. Serial Communications*

Following the changes to PICs code and hardware configuration, the serial data line was tested using an open source program called RealTerm. The results of having three ultrasonics connected to the board can be seen in figure 25.


**Figure 25.** Results of serial communications line

What can be assessed from figure 25 is that the serial communications is working correctly as can be seen from the layout of the information. The first part of the packet is a 1, followed by the seven timer values for the distances detected, finishing with a 2 to close the packet. It can also be seen that on the lines where the ultrasonics are connected, the values are between the range of 100 to 30000, which is what was wanted, as well as a value of 40000 for where an error has occurred (no ultrasonics connected).

### *9.5.2. Ultrasonics Distance Sensing Analysis*

As the serial communication line was working, the ultrasonics could be tested to see how accurately they were by using the values being returned to the sbRIO board. Table 7 shows the results in the form of timer values and the distance using a formula produced by a systems identification method, for which more information can be found in [19].

**Table 7.** Ultrasonic readings at ranging distances

| Object Distance / cm | Reading 1 | Reading 2 | Reading 3 | Average Reading | Distance using Calculations / cm |
|---|---|---|---|---|---|
| 1 | 791 | 767 | 787 | 781.6666667 | 0.96 |
| 5 | 991 | 993 | 995 | 993 | 4.95 |
| 10 | 1277 | 1283 | 1276 | 1278.666667 | 10.34 |
| 20 | 1765 | 1757 | 1759 | 1760.333333 | 19.43 |
| 25 | 2046 | 2048 | 2051 | 2048.333333 | 24.87 |
| 30 | 2318 | 2324 | 2319 | 2320.333333 | 30.01 |
| 35 | 2588 | 2583 | 2573 | 2581.333333 | 34.93 |
| 40 | 2861 | 2855 | 2854 | 2856.666667 | 40.13 |
| 45 | 3120 | 3114 | 3118 | 3117.333333 | 45.05 |
| 50 | 3389 | 3379 | 3386 | 3384.666667 | 50.10 |
| 55 | 3647 | 3650 | 3649 | 3648.666667 | 55.08 |
| 60 | 3916 | 3923 | 3919 | 3919.333333 | 60.19 |
| 65 | 4158 | 4157 | 4161 | 4158.666667 | 64.71 |
| 70 | 4425 | 4437 | 4417 | 4426.333333 | 69.76 |
| 75 | 4697 | 4689 | 4693 | 4693 | 74.80 |
| 80 | 4956 | 4948 | 4959 | 4954.333333 | 79.73 |
| 85 | 5237 | 5246 | 5242 | 5241.666667 | 85.16 |
| 90 | 5514 | 5498 | 5497 | 5503 | 90.09 |
| 95 | 5764 | 5763 | 5771 | 5766 | 95.05 |
| 100 | 6042 | 6043 | 6057 | 6047.333333 | 100.37 |

What can be seen from this data is that the ultrasonics are extremely accurate within a metre range, which should be more than suitable for object avoidance upon the UAV.

## 10. Overall Results

As of the 24/03/2011, the UAV has been tested attached to the test rig in which collisions have occurred. Damage to the craft has been minimal, with only some slight bending in the safety ring but significant damage occurred with the coupling pad on the test rig, which had to be replaced with a thicker piece of aluminium. Therefore, in terms of the design of the craft, it has been successful and has produced a skeleton that is suitable not only for UAV purposes, but also for testing purposes.

The controller has not yet been successful in maintaining the craft at a stable hover. However, there is evidence that it is working, as when a tilt occurs, the craft does move in the opposite direction to the tilt but overshoots, causing oscillations. Therefore, the control is close to achieving stability but needs to be fine turned further. The outer ring also does not wobble due the supports, preventing further noise on the system when oscillating.

As far as the transfer function produced from the systems identification, the motors work well with both formulas produced, providing a better response in conjunction with the controller. However, the response could be improved by incorporating a nonlinear based system identification algorithm to provide results closer to 90%, as opposed to averaging approximately 55%, as the ARMAX model does across the range of data sets.

The ultrasonics have shown themselves to be able to operate well with motors present and the method of implementing them has been successful. However LABVIEW does not like using serial through an FPGA, therefore the ultrasonics data has not been able to be passed back to the sbRIO board as of yet. Further work needs to be done to either adapt LABVIEW to be able to manage a serial signal, or find an alternative method of passing the distances back to LABVIEW.

## 11. Conclusion and Future work

The results show that the design and construction process of the UAV have produced a successful UAV skeleton that is light, strong and durable, however, the weight limitations were exceeded slightly to do so. The main problem was the size, weight and power consumption of the NI sbRIO board, which meant the design had to be bigger and heavier to provide space for the board, as well as heavier due to more power sources being needed. Therefore, if this project was to be repeated, a smaller control board would be recommended as it would reduce the power needs, size and weight. Infact the motors could produce more thrust with bigger propellers, which cannot currently occur otherwise they would overlap the board producing noise.

What could be used instead of sbRIO board is a series of PIC's interfaced with C#, which would reduce the weight and power requirements of the UAV drastically, while still providing a user interface.

The ultrasonics have also shown themselves to be practical on board the craft with the noise reduction methods implemented, however, communicating their responses back to the sbRIO board has proven difficult due to restrictions. However, the analogue ports of the sbRIO board have been used to interface other components of the UAV successfully, therefore for future work the signal returned to the PIC could be output to the analogue section of the sbRIO board in which the voltage is dependent on the distance.

Systems identification of the motors with varying delays and input types has shown to be useful such that a transfer function has been produced to be used within the controller. The best response is still when the inputs provided to the motor are within the linear operating range of the motor, therefore for future work within this area, a nonlinear systems identification method should be implemented within a LABVIEW environment, as opposed to MATLAB in which the linear model was produced.

The test rig has also been a good asset to the project allowing testing to occur in a safe manner, however it has been noticed during testing that the UAV is not compensating for the extra inertia that the rig produces during flight, meaning that if the UAV does reach a point where its stable, movement will become difficult as stopping will be impossible if the craft is programmed to act in an un-tethered way. However this is the same for any testing rig as weight will be attached to the craft, therefore the only factor that can change to reduce this problem in the future is to reduce the weight of the test rig.

It would also be a good idea to adapt the coupling pole to move backwards and forwards to add a third dimension of movement to the rig such that excessive noise in the restricted plane can be adapted safely instead of producing an uncontrollable movement.

Therefore, as a whole, the individual modules created for this project have been proven to be successful. However, problems have become apparent during the assembling stages of the project, which still need some work to produce an overall working successful system. This does not however mean that the project has failed, as it has provided an insight to the systems involved with a quadrotor UAV for all team members, as well as the time management that is necessary for a project of this scale.

## 12. Acknowledgements

The author acknowledges the work of the following individuals: Dr Victor Becerra, Stephen Gould, Dr Sillas Hadjiloucas, Tim Holt, Mark Ince, Balazs Janko, Ian Roberts, Joe Warren, and Ruth White.

## 13. References

[1] **Hull, Liz.** Drone makes first UK 'arrest' as police catch car thief hiding under bushes. *Daily Mail Online.* [Online] 12th Feb 2010. [Cited: 03 04 2011.] http://www.dailymail.co.uk/news/article-1250177/Police-make-arrest-using-unmanned-drone.html.

[2] **Amir, M.Y and Abbas, V.** Modeling of Quadrotor Helicopter Dynamics. *IEEEXplore Digital Library.* [Online] 11th Apr 2008. [Cited: 08 04 2011.] http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4505621.

[3] **Dragonfly Innovations.** DragonFlyer X4. *Dragonfly Innovations INC.* [Online] 2010. [Cited: 14 02 2011.] http://www.draganfly.com/uav-helicopter/draganflyer-x4/.

[4] **Parrot.** AR Drone How does it work? *Parrot AR Drone.* [Online] 2010. [Cited: 19 02 2011.] http://ardrone.parrot.com/parrot-ar-drone/uk/how-does-it-work.

[5] **Vigneshram.B, Patwardhan, Prof Amit and Henry, Prof Rabinder.** Theoretical Analysis of Quadrotors. *Scribd.* [Online] 14 03 2007. [Cited: 08 04 2011.] http://www.scribd.com/doc/25490458/Theoretical-Analysis-of-Quad-Rotors-1.

[6] **Minh, Ly dat and Ha, Cheolkeun.** Modeling and control of a quadrotor MAV using vision based measurement. *IEEE Explore Digital Library.* [Online] 15 Oct 2010. [Cited: 07 04 2011.] http://ieeexplore.ieee.org/search/srchabstract.jsp?tp=&arnumber=5668079&queryText%3Dquadrotor+cameras%26openedRefinements%3D*%26searchField%3DSearch+All

[7] **Wu, Yiting.** Development and Implementation of a Control System for a quadrotor UAV. [Online] Mar 2009. [Cited: 26 03 2011.] http://www.hs-weingarten.de/~mobilelab/docs/Master_Thesis_Yiting_Wu.pdf.

[8] **S. Bouabdallah, R. Siegwart.** Nonliner Control of Quadrotor. *AuthorStream.* [Online] 06 Aug 2010. [Cited: 09 04 2010.] http://www.authorstream.com/Presentation/aSGuest58990-460904-nonlinear-control-of-quadrotor/.

[9] **Hong, Dr. Xia.** RLS Sytems Identification. [Online] Aug 2010. [Cited: 06 4 2011.] http://www.personal.reading.ac.uk/~sis01xh/teaching/Sysid/Sysid4.pdf.

[10] **Robot-Electronics.** SRF05 - Ultrasonic Ranger. *Robot-Ulectronics.* [Online] [Cited: 02 04 2011.] http://www.robot-electronics.co.uk/htm/srf05tech.htm.

[11] **Wirnitzer, Bernhard.** Interference Cancelation in Ultrasonic Sensor Arrays by Stochastic Coding and Adaptive Filtering. *CiteSeerX.* [Online] 1998. [Cited: 15 03 2011.] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.131.5301&rep=rep1&type=pdf.

[12] **Cyber Technology.** Cyber Quad. *Cyber Technology Advanced UAV Solutions.* [Online] [Cited: 02 04 2011.] http://www.cybertechuav.com.au/-Overview,85-.html.

[13] **Acroname Robotics.** NIST Disaster City Event. *Acroname Robotics.* [Online] 28 06 2007. [Cited: 02 04 2011.] http://www.acroname.com/travel/NIST_6_2007/quadrotor.jpg.

[14] **Srikanth, Manohar B.** Adaptive Control of Quadrotor. [Online] [Cited: 02 04 2011.] http://www.mit.edu/~srimano/research/.

[15] **National Instruments.** User Guide NI sbRIO-961x/963x/964x and NI sbRIO-9612XT/9632XT/9642XT. *National Instruments.* [Online] Jun 2010. [Cited: 14 03 2011.] www.ni.com/pdf/manuals/375052c.pdf.

[16] **SparkFun.** IMU 6DOF Razor - Ultra-Thin IMU. *Sparkfun Electronics.* [Online] [Cited: 16 03 2011.] http://www.sparkfun.com/products/9431.

[17] **Carbon Fibre Tubes LTD.** Carbon Fibre vs Aluminium and steel. *Carbon Fibre Tubes LTD.* [Online] [Cited: 6 04 2011.] http://www.carbonfibertubes.co.uk/technology.html.

[18] **Microchip.** dsPIC30F4011/4012 Data Sheet, High Performance Digital Signal Controllers. *Microchip.* [Online] 2005. [Cited: 08 04 2011.] ww1.microchip.com/downloads/en/devicedoc/70135c.pdf.

## Appendix A. RLS Base Code

```
%Implements a simple RLS algorithm
%Scott Alexander
%Cybernetics MEng
%Reading Univeristy

%Environment set up to clear variables and workspace
clc                     %clears workspace
clear all               %clears all variables


%system set up - set coefficients and allow user to input system needs

%[b,a] =dord2

%set coefficients for testing
a=[1 -1.5363 0.8607];
b=[0 0.0416 0.395];

%gets user to input what time of input they would like
System.Input = input('would you like (1)step, (2)sin or (3)random input to
the system? >');
if System.Input == 1;
    u=ones(1,50);
end
if System.Input == 2;
    u=5*sin(1:2:70);
end
if System.Input == 3;
    u = 5*randn(1,200);
end

%Gets user to select if they want noise or not
System.Noise = input('Would you like to induce noise on this system? (1) No
(2) Yes >');
if System.Noise == 1;
    y=filter(b,a,u);
end

if System.Noise == 2;
    y=filter(b,a,u)+0.1*randn(size(u));
end

%allows the user to input a forgetting factor
Lander = input('Input a forgetting factor >');

%sets all the parameters
Na = numel(a)-1;
Nb = numel(b)-1;
LN = 1000;

theta_nminus1  =zeros(Na+Nb,1); % Initialise the estimate of theta to zero
P_nminus1=LN.*eye(Na+Nb);     % Initialise P where LN is a large number
Theta=[];                   %initialises a history vector
Epsilon=[];                 %initialises a history vector
Yestimate=[];               %initialises a history vector
Ydifference=[];             %initialises a history vector
Z=[];                       %initialises a history vector
```

```
% Step through data and for each new point generate a new estimate
for n=1:length(y) % Change 10 to length(y) once you have the code working

    % set py to the previous Na y values
      py=zeros(1,Na);
      for i=n-1:-1:n-Na
         if i>0 py(n-i)=y(i);   end
      end
      % set pu to the previous Nb u values
      pu=zeros(1,Nb);
      for i=n-1:-1:n-Nb
          if i>0 pu(n-i)=u(i);   end
      end

      % Construct varphi from py' and pu'
      phi= [-(py)';pu'];

      % Use varphi(n), y(n) theta(n-1) and P(n-1) to iterate the next
estimate
      epsilon= y(n)-(phi'*theta_nminus1);  %calculate epsilon
      %calculate P including a forgetting factor
      P= (1/Lander)*(P_nminus1-
((P_nminus1*phi*phi'*P_nminus1)/(Lander+phi'*P_nminus1*phi)));
      K= P*phi;   %calculate P
      theta= theta_nminus1 + K*epsilon;  %calculate theta

      %Yestimate
      %set the estimated coeffecients using theta
      A = [1 theta(1) theta(2)];
      B = [0 theta(3) theta(4)];
      %calculate an estimate of y
      yestimate = filter(B,A,u);

      % get ready for the new iteration
      theta_nminus1=theta;     %theta_nminus one is set to current theta
      P_nminus1=P;              %P_nminus one is set to current P
      Theta=[Theta; theta'];        %Data is added to history vector
      Epsilon=[Epsilon;epsilon];    %Data is added to history vector
      Yestimate = [Yestimate; yestimate(n)];    %Data is added to history
vector
      ydifference = y(n)'-yestimate(n); %calculate the difference between y
and the estimation
      Ydifference = [Ydifference;ydifference];  %Data is added to history
vector

      %loop number check for plotting
      Z=[Z;n];

end % and so it ends

% If you have recorded parameter evolution you can plot with
%plot coefficient estimation
subplot(2,2,[1 2]);
plot(Theta);
title('Coefficient Estimation')
xlabel('Iterations')
ylabel('Coefficients')
```

```
%plot the error over time of the estimation
subplot(2,2,3);
plot(Epsilon);
title('Error over time')
xlabel('Iterations')
ylabel('Error')

%plot y against its estimate and show the difference between them
subplot(2,2,4)
plot(Z,y-0.01,Z,Yestimate,Z,Ydifference*10)
title('Changes in Y and aproixamation')
xlabel('Iterations')
ylabel('Value of Y')
legend('Y','Y Estimation','Difference')
a
b
A
B
```

## Appendix B. RLS Amended Code for Data Set Integration

```
%Applies ARMAX to real data
%Tests with different numbers of coefficents
%Looks to see if systems are stable
%Scott Alexander
%Cybernetics MEng
%Reading Univeristy

%Environment set up to clear variables and workspace
clc                     %clears workspace
clear all               %clears all variables
                                % put raw data into variable ry
load InputinPWM.mat
load OutputInThrust.mat
%System set up - User can set up system parameters noise, forgetting
factor, num of coefficents
Lander = input('Input a forgetting factor >');              %sets
forgetting factor
%Print message to user on options
disp('System (1) - 2a,2b and 2c coefficients');
disp('System (2) - 3a,2b and 2c coefficients');
disp('System (3) - 3a,3b and 3c coefficients');
disp('System (4) - 4a,3b and 3c coefficients');
System.Co = input('Which system would you like? 1-4 >');    %gets user
input of system

if System.Co == 1               %sets number of coeffiects and creates y
vector for system 1
    Na=2;
    Nb=2;
    Nc=2;
    y=OutputInThrust;
end
if System.Co == 2               %sets number of coeffiects and creates y
vector for system 2
    Na = 3;
    Nb = 2;
    Nc = 2;
    y=OutputInThrust;
end
if System.Co==3                 %sets number of coeffiects and creates y
vector for system 3
    Na = 3;
    Nb = 3;
    Nc =3;
    y=OutputInThrust;
end
if System.Co == 4               %sets number of coeffiects and creates y
vector for system 4
    Na = 4;
    Nb = 3;
    Nc=3;
    y=OutputInThrust;
end

u = InputinPWM;           %creates a vector of inputs

%Allows the user to produce noise on the system
```

```
System.Noise=input('would you like to induce noise on the system? (1) no
(2) yes >');
if System.Noise == 2            %If the user does want noise
    for i=1:length(y)           %produce a loop
        store = y(i);
        store = store+0.1*randn(size(u(i)));  %and add random noise to all
y vector positions
        y(i)=store;              %store new y value
    end
end

LN = 1000;
theta_nminus1  =zeros(Na+Nb,1); % Initialise the estimate of theta to zero
P_nminus1=LN.*eye(Na+Nb);       % Initialise P where LN is a large number
Theta=[];                       % Initialise Theta History Vector
Epsilon=[];                     % Initialise Epsilon History Vector
Yestimate=[];                   % Initialise Yestimate History Vector
Ydifference=[];                 % Initialise Ydifference History Vector
Z=[];                           % Initialise iteration counting History
Vector

% Step through data and for each new point generate a new estimate
for n=1:length(y) % Change 10 to length(y) once you have the code working

    % set py to the previous Na y values
      py=zeros(1,Na);
      for i=n-1:-1:n-Na
         if i>0 py(n-i)=y(i);   end
      end
      % set pu to the previous Nb u values
      pu=zeros(1,Nb);
      for i=n-1:-1:n-Nb
          if i>0 pu(n-i)=u(i);   end
      end

      % Construct varphi from py' and pu'
      phi= [-(py)';pu'];
      epsilon= y(n)-(phi'*theta_nminus1);      %calculates epsilon
      %calculates P using a forgetting factor
      P= (1/Lander)*(P_nminus1-
((P_nminus1*phi*phi'*P_nminus1)/(Lander+phi'*P_nminus1*phi)));
      K= P*phi;          %calculates K
      theta= theta_nminus1 + K*epsilon;     %calulates theta

      %Set up estimated coefficients using theta depending on system
selected
      if System.Co == 1
        A = [1 theta(1) theta(2)];
        B = [0 theta(3) theta(4)];
      end

      if System.Co == 2
        A = [1 theta(1) theta(2) theta(3)];
        B = [0 theta(4) theta(5)];
      end

      if System.Co == 3
        A = [1 theta(1) theta(2) theta(3)];
        B = [0 theta(4) theta(5) theta(6)];
      end
```

```
      if System.Co == 4
        A = [1 theta(1) theta(2) theta(3) theta(4)];
        B = [0 theta(5) theta(6) theta(7)];
      end

      %Creates a yestimate vector for estimates of y
      yestimate = filter(B,A,u);

      % get ready for the new iteration
      theta_nminus1=theta;          %sets Theta_nminus1 to current
iteration value
      P_nminus1=P;                  %sets P_nminus1 to current iteration
value
      Theta=[Theta; theta'];            %adds to history vector
      Epsilon=[Epsilon;epsilon];        %adds to history vector
      Yestimate = [Yestimate; yestimate(n)];    %adds to history vector
      ydifference = y(n)'-yestimate(n); %calulates differnce between y and
estimation of y
      Ydifference = [Ydifference;ydifference];  %adds to history vector

      %loop number check for plotting
      Z=[Z;n];      %adds to history vector

      % complex plane set up
      if n == length(y); %only happens after last calculation
          if System.Co == 1;
              a1=theta(1);
              a2=theta(2);
              den = [1 a1 a2];
          end
          if System.Co ==2;
              a1=theta(1);
              a2=theta(2);
              a3=theta(3);
              den = [1 a1 a2 a3];
          end
          if System.Co ==3;
              a1=theta(1);
              a2=theta(2);
              a3=theta(3);
              den = [1 a1 a2 a3];
          end
          if System.Co ==4;
              a1=theta(1);
              a2=theta(2);
              a3=theta(3);
              a4=theta(4);
              den = [1 a1 a2 a3 a4];
          end
      end
end % and so it ends

%plot Coefficent estimation
subplot(2,2,1);
plot(Theta);
title('Coefficient Estimation');
xlabel('Iterations');
ylabel('Coefficents');
```

```
%plot error over time
subplot(2,2,2);
plot(Epsilon);
title('Error over time');
xlabel('Iterations');
ylabel('Error');

%plot Y against the estimation of y with the differnce as well
subplot(2,2,3);
plot(Z,y,Z,Yestimate,Z,Ydifference) ;
title('Changes in Y and aproixamation');
xlabel('Iterations');
ylabel('Value of Y');
legend('Y','Y Estimation','Difference');

%plots to see if system is stable
subplot(2,2,4);
pzmap(1,den);
hold on;
axis([-1.5 1.5 -1.5 1.5]);
i=1:101;
plot(cos(i*pi*2/100),sin(i*pi*2/100));
```

## Appendix C. PIC code Ultrasonic sensing system using a shift register

```
#include "p30f4011.h"

#define TRUE 1
#define FALSE 0

#define TR_SER LATEbits.LATE4
#define TR_SRCK LATEbits.LATE1
#define TR_SRCLR LATEbits.LATE0
#define TR_RCK LATEbits.LATE2
#define TR_RCLR LATEbits.LATE3

unsigned char FirstHalf = TRUE;
unsigned long int TimeOut = 0;
#define TIMEOUTVALUE 20000                  //20000 worth of counts, if more, timeout
occurs

//#define UART_ASCII_MODE TRUE
#define FIFO_SIZE 32                                //always Power of 2
unsigned char txFIFO[FIFO_SIZE] = {0};
unsigned int txFIFOhead, txFIFOtail;

void delay (){
       int i,j;
       for (i=0; i < 15; i++){
              for (j= 0; j< 100; j++){

              }
       }
}

void uartProcessFIFO()
{
       if (U1STAbits.TRMT)
              if (txFIFOhead != txFIFOtail)
              {
                     U1TXREG = txFIFO[txFIFOtail]; txFIFOtail = (txFIFOtail+1) &
(FIFO_SIZE);
              }
}

void uartSendInt16(unsigned int value)
{
#ifdef UART_ASCII_MODE

#else
              txFIFO[txFIFOhead] = (value & 0xFF); txFIFOhead = (txFIFOhead+1) &
(FIFO_SIZE);
              txFIFO[txFIFOhead] = (value >> 8);          txFIFOhead = (txFIFOhead+1) &
(FIFO_SIZE);
#endif
}

void uartSendBOL()
{
#ifdef UART_ASCII_MODE

#else
       uartSendInt16(1);     //header bytes
#endif
}

void uartSendEOL()
{
```

```
#ifdef UART_ASCII_MODE

#else
	uartSendInt16(2);	//footer bytes
#endif
}

void KickUS()
{
	if (FirstHalf)
	{
		TR_SRCLR=0; TR_SRCLR=1;		//Reset all
	}

	TR_SER=FirstHalf;					//Clock out 4 lots of 0 (upper
four will be ones!)
	TR_SRCK=1; TR_SRCK=0; TR_SRCK=1; TR_SRCK=0;
	TR_SRCK=1; TR_SRCK=0; TR_SRCK=1; TR_SRCK=0;

	TR_RCK=1; TR_RCK=0;			//Shift them out
	//Delay here for 10uS	//This kicks off all four USs
	TR_RCLR=1; TR_RCLR=0;			//Clear outputs

	TMR3 = 0;	//Reset TIMER3 Counter here
}

void Idler()		//This is used to send stuff on UART if anything in FIFO
{
	uartProcessFIFO();	//take care of UART
					//do whatever else we may need to
}

int main(void){

	TRISB = 0x30; // set port b to 0011 0000 allowing inputs to RB4 and RB5
	TRISD = 0x03; // set port d to 0000 0011 allowing inputs to RD0 and RD1
	TRISE = 0x00; // set RE0 to RE4 to outputs
	TRISF = 0x10; // set seriel UTX to output and URX to input

	IC1CON = 0x02; // set IC1CON to falling edge
	IC2CON = 0x02; // set IC1CON to falling edge
	IC7CON = 0x02; // set IC1CON to falling edge
	IC8CON = 0x02; // set IC1CON to falling edge

	T3CON = 0x8000;


	U1BRG = 11;			//Set Baud rate to 38400 @ 8MHz clock
	U1STA = 0x0400;			//TXEN
	U1MODE = 0x8000;	//UARTEN

	FirstHalf = 0;

	while(1){


		FirstHalf ^= 1;			// XOR 1
		KickUS();

		//Wait for all US to fall or TIMEOUT
		if (FirstHalf) while ( !(_IC1IF && _IC2IF && _IC7IF && _IC8IF) && (TimeOut++
< TIMEOUTVALUE) ) { Idler(); };
		if (!FirstHalf) while ( !(_IC1IF && _IC2IF && _IC7IF) && (TimeOut++ <
TIMEOUTVALUE) ) { Idler(); };

		//test if we are here because of timeout or valid readings
```

```
                        //Read out all 16 bits COMPARES and send to UART FIFO
                        if (FirstHalf) uartSendBOL();//StartPacket

                        if (_IC1IF) uartSendInt16(IC1BUF); else uartSendInt16(40000);
                        if (_IC2IF) uartSendInt16(IC2BUF); else uartSendInt16(40000);
                        if (_IC7IF) uartSendInt16(IC7BUF); else uartSendInt16(40000);

                        if (FirstHalf) if (_IC8IF) uartSendInt16(IC8BUF); else
uartSendInt16(40000);
                        if (!FirstHalf) uartSendEOL();      //FinishPacket

                        _IC1IF=0; _IC2IF=0; _IC7IF=0; _IC8IF=0; TimeOut =0;

        }

}
```

## Appendix D. PIC code Ultrasonic sensing system without a shift register

```
#include "p30f4011.h"

_FOSC(CSW_FSCM_OFF & FRC_PLL4);
_FWDT(WDT_OFF);

#define TRUE 1
#define FALSE 0
#define LED1 LATEbits.LATE0
#define LED2 LATEbits.LATE1
#define LED3 LATEbits.LATE2
#define LED4 LAT
+Ebits.LATE3


unsigned char FirstHalf = TRUE;
unsigned long int TimeOut = 0;
#define TIMEOUTVALUE 20000                    //20000 worth of counts, if more, timeout occurs

//#define UART_ASCII_MODE TRUE
#define FIFO_SIZE 64                                    //always Power of 2
#define FIFO_MASK FIFO_SIZE-1
char txFIFO[FIFO_SIZE] = {0};
unsigned char txFIFOhead, txFIFOtail;

void delay (){
        int i,j;
        for (i=0; i < 15; i++){
                for (j= 0; j< 800; j++){

                }
        }
}

void delay2 (){
        int i,j;
        for (i=0; i < 20; i++){
                for (j= 0; j< 1; j++){

                }
        }
}



void uartProcessFIFO()
{
        if (U2STAbits.TRMT)
                {
                if (txFIFOhead != txFIFOtail)
                {
                        U2TXREG = txFIFO[txFIFOtail]; txFIFOtail = (txFIFOtail+1) & (FIFO_MASK);
                        LED4 = 1;
                }
        }
}

void uartSendInt16(unsigned int value)
{
#ifdef UART_ASCII_MODE

#else
                txFIFO[txFIFOhead] = (value & 0xFF);            txFIFOhead = (txFIFOhead+1) & (FIFO_MASK);
                txFIFO[txFIFOhead] = (value >> 8);              txFIFOhead = (txFIFOhead+1) & (FIFO_MASK);
#endif
}
```

```
void uartSendBOL()
{
#ifdef UART_ASCII_MODE

#else
        uartSendInt16(1);   //header bytes
#endif
}

void uartSendEOL()
{
#ifdef UART_ASCII_MODE

#else
        uartSendInt16(2);   //footer bytes
#endif
}

void KickUS()
{
        if (FirstHalf)
        {
                LATB|= 0xb5;
                LATC|= 0x4000;
                delay();
                LATB&=~0xb5;
                LATC&=~ 0x4000;
        }
        else {
                LATB|= 0x3a;
                LATC|= 0x2000;
                delay();
                LATB&=~ 0x3a;
                LATC&=~ 0x2000;
        }



        TMR3 = 0;           //Reset TIMER3 Counter here
        _IC1IF=0; _IC2IF=0; _IC7IF=0; _IC8IF=0;
}

void Idler()                    //This is used to send stuff on UART if anything in FIFO
{
        uartProcessFIFO(); //take care of UART
                                                        //do whatever else we may need to
}

int main(void){

        ADPCFG = 0xFFFF;

        PWMCON1=0xff00;

        TRISB = 0x030; // set port b to 0011 0000 allowing inputs to RB4 and RB5
        TRISC = 0x0000;  // set port b to 0000 0000
        TRISD = 0x03; // set port d to 0000 0011 allowing inputs to RD0 and RD1
        TRISE = 0x00;
        TRISF = 0x10; // set serial UTX to output and URX to input

        IC1CON = 0x02; // set IC1CON to falling edge
        IC2CON = 0x02; // set IC1CON to falling edge
        IC7CON = 0x02; // set IC1CON to falling edge
        IC8CON = 0x02; // set IC1CON to falling edge

        T3CON = 0x8010;             //Timer for US
```

```
	PR2 = 1439;				// div by 1439 of 28789 yields 20Hz
	T2CON = 0x8030;		//Run, 1:256 prescale yields 28789Hz


	U2BRG = 11;				//Set Baud rate to 38400 @ 8MHz clock
	U2STA = 0x8400;		//TXEN
	U2MODE = 0x8000;		//UARTEN
	U2MODEbits.UARTEN = 1;
	U2STAbits.UTXEN = 1;

	FirstHalf = 0;
//	TR_SRCLR=1;				//idle in HIGH as it's active low
//	TR_RCLR=1;				//idle in HIGH as it's active low

	txFIFOhead = 0; txFIFOtail=0;

	while(1){
		LATB &=~0x000F; // Turn off al LEDs

		FirstHalf ^= 1;			// XOR 1
		KickUS();

		//Wait for all US to fall or TIMEOUT
		 if (FirstHalf) while ( !(_IC1IF && _IC2IF && _IC7IF && _IC8IF) && (TimeOut++ <
TIMEOUTVALUE) ) { Idler(); };
		if (!FirstHalf) while ( !(_IC1IF && _IC2IF && _IC7IF)			&& (TimeOut++ <
TIMEOUTVALUE) ) { Idler(); };

		//test if we are here because of timeout or valid readings


			//Read out all 16 bits COMPARES and send to UART FIFO
			if (FirstHalf) uartSendBOL();//StartPacket

			 if (_IC7IF) uartSendInt16(IC7BUF+3); else uartSendInt16(40000);
			 if (_IC8IF) uartSendInt16(IC8BUF+3); else uartSendInt16(40000);
			 if (_IC2IF) uartSendInt16(IC2BUF+3); else uartSendInt16(40000);

			if (FirstHalf){
			 if (_IC1IF) uartSendInt16(IC1BUF+3); else uartSendInt16(40000);}

			if (!FirstHalf) uartSendEOL();		//FinishPacket

			_IC1IF=0; _IC2IF=0; _IC7IF=0; _IC8IF=0;
			TimeOut=0;

	//slow down
		while(!_T2IF){ Idler(); };

		_T2IF=0;

	}

}
```