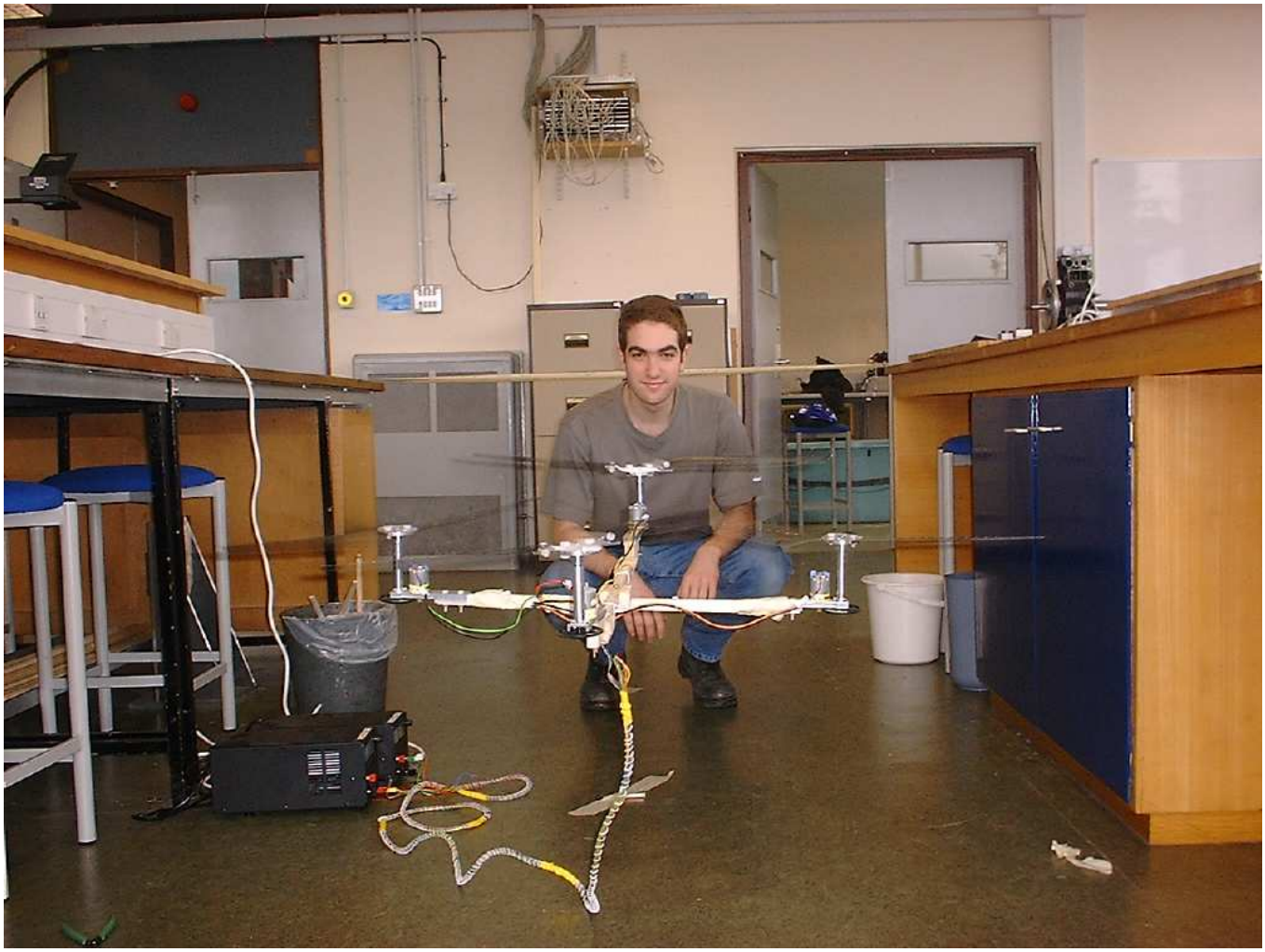
Helicopter Instrumentation

Edward Rosten
St Catherine's College

May, 2002



Abstract

This project involved the design and construction of a model four rotor helicopter with sufficient infrastructure so that, with further work, the helicopter could hover without interaction.

A sensor suite and actuation system were designed and constructed. A conventional 3 axis magnetic field sensor system was augmented by an optoelectronic sensor so that the new system could completely determine the helicopter's orientation. The whole sensor system was analysed to find the error rate and to demonstrate that it met the specification.

The dynamic and continuous behaviour of the motor/rotor systems were analysed. With regard to the continouous behaviour, a way of calculating the input required for a given thrust or torque was found from measurements of the system. In relation to the dynamic behaviour, it was demonstrated that it could be approximated by a first order system and the parameters for this system were found.

Contents

1 Introduction

The aim of this project is to design and build a four rotor helicopter including an instrumentation and actuation system suitable for a control system to let the helicopter hover autonomously.

A four rotor helicopter has several advantages over a conventional single or dual rotor helicopter. The main advantage is that all the rotor blades are fixed, which makes it mechanically much simpler. Secondly, since four rotors instead of one or two are being used, the rotors can be smaller which alleviates some of the problems associated with having high tip speeds on the rotors. The main problem with these helicopters is that they are much harder to fly since the act of making it move horizontally (by tilting it) also causes the helicopter to rotate.

Section 2 describes the physical construction of the helicopter. Section 3 discusses the requirements of the sensor system, describes its construction and analyses its performance. In section 4 the specifications and construction of the actuation system and the modifications made to it are described. Section 5 deals with the calibration of the motor/rotor system and includes the rotor time constants and plots of the calibration function.

2 Helicopter Model and Construction

A physical four rotor model helicopter is required for this project. Unfortunately, the only one that is commercially avaliable is not powerful enough to carry all the circuitry and instrumentation required for this project. It was therefore necessary to construct a helicopter. This was done using the main rotor blades, main motor and gearing system from the Eco-Piccolo helicopter (manufactured by Ikarus). This is a conventonal single rotor helicopter weighing 250g and capable of battery powered flight. More lifting capacity was generated by replacing the batteries with power from an external source.

A modification was necessary because if all the rotors rotate in the same direction, their torques are also in the same direction and these add up to produce a large net torque which twists the helicopter around very fast. To prevent this, two of the rotors must rotate in the opposite direction. In order to produce lift, these two rotors must have their blades angled in the opposite sense as well. No suitable rotors exist; the ones from the commercially avaliable four rotor helicopter are too small. The solution was to rotate the blades by 20°(about an axis running the length of the blades) such that the leading edge is horizontal when the rotors run in the opposite direction. This reduces the rotor efficiency since they are not designed to operate in this manner. As a result, using the same motors, they generate about 63% of the lift of the unmodified rotors (see section 5). Although this lift is less than would be expected, it is sufficient for the experiment.

To make the helicopter, several parts need to be manufactured. These are:

- Forward rotor head,
- Reverse rotor head,
- Helicopter frame,
- Motor/rotor assembly—this connects the motor, rotor and gears together and attaches them to the frame.

In order to reduce the damage due to crashing during testing, it was decided to make the helicopter frame as simple and cheap as possible, and weaker than all the other parts, reducing the chance of damage occurring to the more complex parts.

Engineering drawings for the major parts are given in Appendix A. The rotor/motor assembly consists of a motor adapter and a rotor mast to keep the rotors at suitable distance above the circuitry. The shaft connecting the main gear to the rotor head passes through one roller bearing (inside diameter 6mm, outside diameter 10mm, length 5mm) at each end of the rotor mast. These are held in place by the rotor head at the top and the main gear at the bottom. The rotor is held on to the rotor head by 5mm diameter nylon bolts. The main motor gear is made from a length of 8 tooth brass gear rod. A photograph of one of these assemblies is shown in Figure 1. The reversed rotors use an adapter, bolted to the rotor head, to twist the blades by 20°. The weight of the forward rotor assembly is 117g and the weight of the reverse rotor assembly is 137g.

The design of the frame is shown in Figure 2. The frame is constructed from balsa wood, resulting in a frame that is light, cheap and easy to fabricate. A photograph of the assembled helicopter is shown in Figure 3.

3 Helicopter Sensor Suite

In order to keep the helicopter stable, the pitch and roll must be measured so that they can be controlled. The helicopter has local axes (x, y, z). The x and y axes are aligned with the two spars of the helicopter. Pitch is the rotation about the y axis and roll is a rotation about the x axis. These are described more precisely in Section 3.4.

It has been shown [3] that some of these measurements can be obtained using the Earth's magnetic field. However, there is a problem with this. Consider the helicopter rotating in a uniform magnetic field. Any rotation of the helicopter can be transformed in to an equivalent rotation of the magnetic field. Now consider a rotation of the magnetic field about a vector aligned with the field. This will have no effect because the field is uniform. As a result, there is one angle that cannot be determined by taking measurements of the magnetic field alone. This angle must be measured using an unrelated system (see Section 3.2).

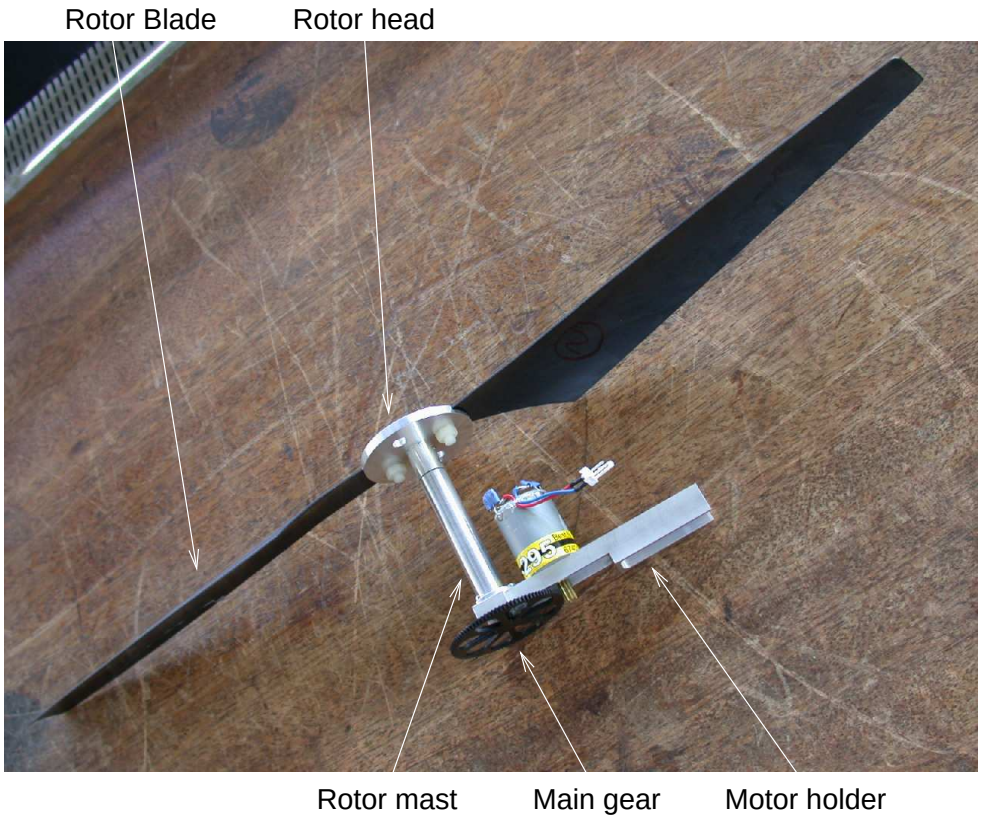


Figure 1: Motor/rotor assembly.

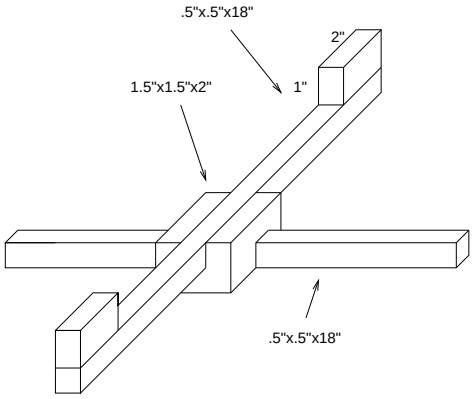


Figure 2: Helicopter frame.

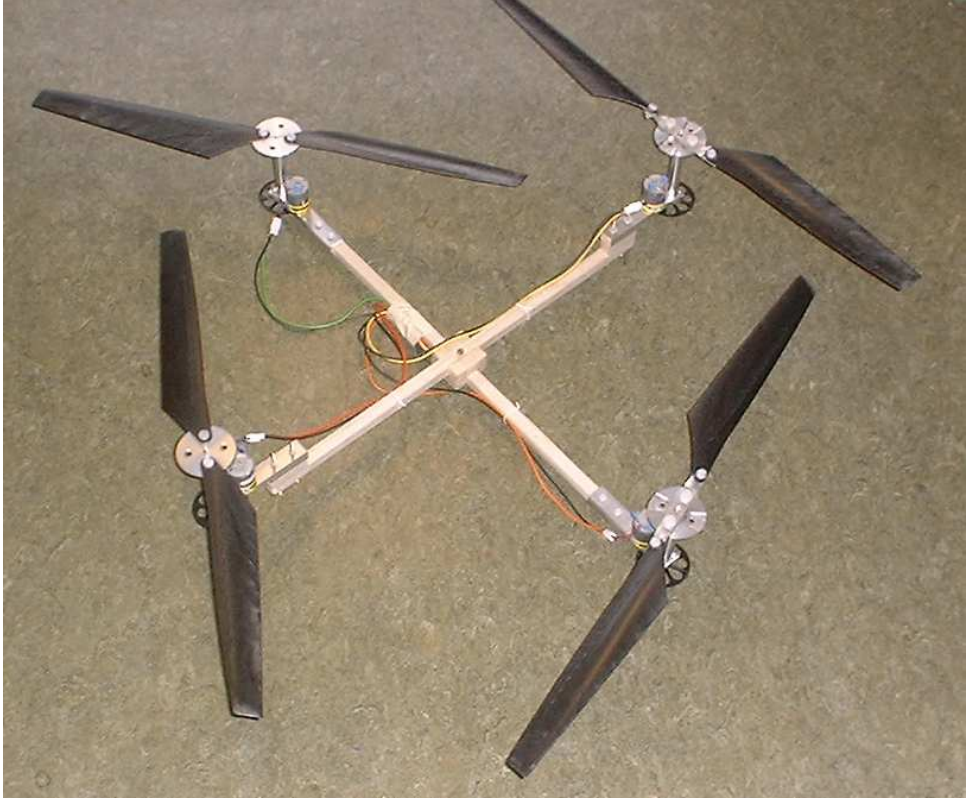


Figure 3: Helicopter.

3.1 Magnetic Field Sensors

3.1.1 Devices and Circuitry

The devices chosen are the Honeywell HMC 1002 magnetic field sensors. These consist of a Wheatstone bridge with magneto resistive elements in the bridge [1]. Magneto resistors have a resistance which varies with the strength of the magnetic field in the direction of the sensitive axis[1]. This means that the resistance, and hence bridge voltage, varies with the dot product of the magnetic field with a vector aligned along the sensor axis, up to an offset and a scaling factor. To get a useful reading, the sensors need to be calibrated to find the offset and scaling factor. This is discussed further in Section 3.1.3. The ADC (analogue to digital converter) chosen (see Section 3.3) responds in the range 0–5V. Assuming that the sensor has the maximum sensitivity stated in the data sheet [1], a gain of 250 is required. However, to avoid damaging the ADC, a gain of 200 was used. This is because some ADCs are very easily damaged by applying an input voltage which lies outside of its specified range of inputs. Magnetized objects (such as screwdrivers) can produce local magnetic fields that swamp the earth's field. It is therefore necessary to limit the gain to protect the ADC.

Since the sensor reading varies with the cosine of the angle to the earth's magnetic field, the variation in sensitivity (measured in volts/degree) is large.

[1]There is a small cross-axis effect, but the magnitude of this is about 0.3% of the magnitude of the main effect.

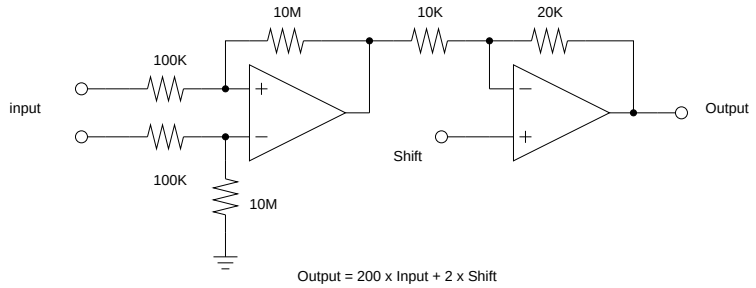


Figure 4: An amplifier and level shifter for the magnetic field sensors.

It has been determined that an accuracy of 1° is required. By considering the change in reading caused by a 1° change in angle as a proportion of the total measurable range, the required resolution of the ADC can be determined. To get the required precision of 1° with the sensor at the best position (perpendicular to the field), the ADC must be able to resolve down to one part in 136. An 8 bit ADC is suitable here since it can resolve down to one part in 256. To operate in the least sensitive region, however (aligned with the field), an accuracy of one part in 15600 is required. A 14 bit ADC is required to get this accuracy. This is a problem because building a circuit sophisticated enough to prevent the low order bits being swamped by noise is a non trivial task. Noise is likely to be a problem in this circuit because of the motors and microcontroller (see Section 3.3). There are two solutions. The most reliable is to mount a redundant set of sensors which all lie at 45 degrees to the existing sensors. The accuracy required to measure 1 degree with the sensors at 45° to the earth's magnetic field is one part in 191, so an 8 bit ADC will still suffice. A second set of sensors would give redundant information. This information can be mixed in proportion depending on the uncertainties of the readings to give a more accurate reading.

A simpler solution (and the one used) is to avoid operating the helicopter anywhere near to parallel to the field. It can be shown using elementary mechanics that if the helicopter maintains lift at an angle of 20°, (the helicopter's z axis sensor aligned with the field), it will accelerate sideways at about 3.5m/s^2. Therefore this is not a serious limitation.

There are several choices for amplifying the differential signal from the bridge. One option is to use the amplifier circuit shown in Figure 4. This consists of a basic differential amplifier with a gain of 100, followed by a differential amplifier with a gain of 2. The gain is performed in two stages because otherwise the input impedance for the first stage is too low. Most of the gain is obtained from the first stage to avoid unnecessary amplification of the noise from the first amplifier. This has several good points. It is inexpensive and also op-amps are available in high density configurations which is useful since the circuitry must be mounted on board the helicopter. The main disadvantage of this circuit is the relatively low input impedance which can load the bridge. This can be solved by using an instrumentation amplifier which can have a very high input impedance—up to $10^{15}\Omega$. The

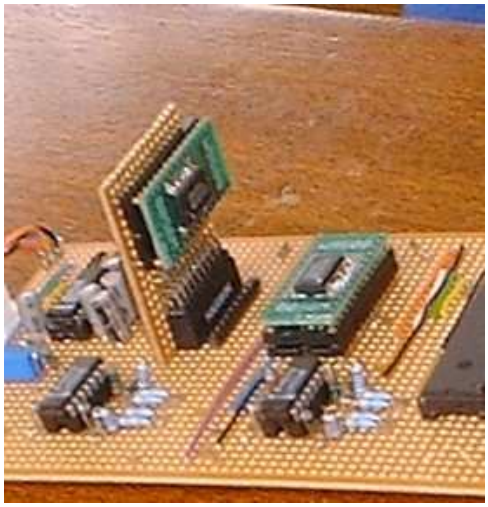


Figure 5: Magnetic field sensors

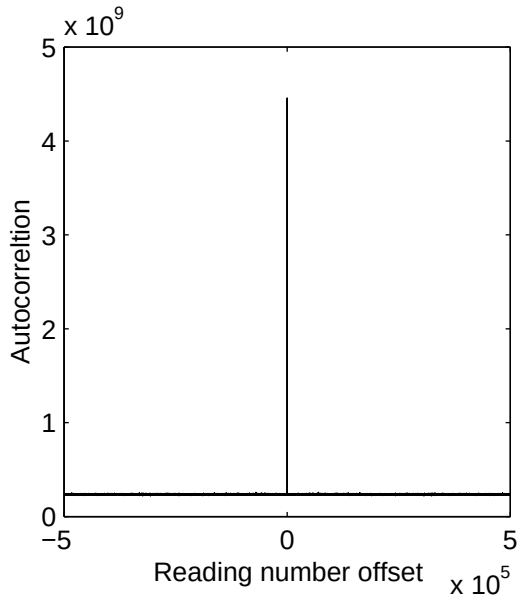
resistance of the bridge is very low [1]; each element only has a resistance of about 600Ω. The loading from the amplifier will only alter the output by about $\frac{1}{5}^{\text{th}}$ of the LSB (least significant bit) of the ADC, so it will not be detected. In the actual ADC used (a 10 bit device, see Section 3.3), the effect will be approximately $\frac{4}{5}^{\text{th}}$ of the LSB. However, as will be shown in Section 3.1.2, this will be swamped by noise in the circuit anyway.

There is the further issue that as the magnetic field cycles over the sensor, the magnetoresistors undergo hysteresis, leading to small errors. Further, if the sensors are placed near to a strong local field, they can undergo hysteresis. To counteract this, the sensors have some degaussing circuitry built in. They require a current pulse of at least 4A for a few microseconds to be passed through the degaussing strap. All the sensor circuitry including the circuit used to pulse the current is given in Appendix B.1 The current pulsing circuit is the same as the circuit given in the data sheet except with a small modification so that easily obtainable components can be used.

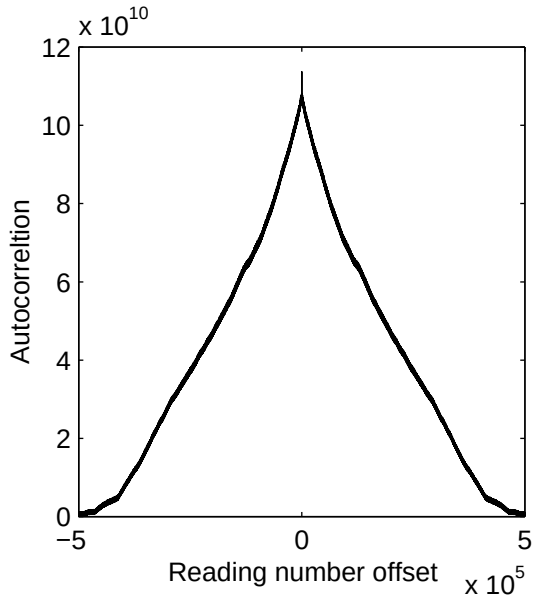
In order to get the required set of readings, three magnetic field sensors are necessary, mounted orthogonally to one another. A single HMC1002 contains two devices at right angles to each other. Using a second device mounted at right angles to the first, the required sensor orientations can be achieved. With this arrangement, two of the sensors point in the same direction and give redundant information. The physical layout of this is shown in Figure 5.

3.1.2 Noise Analysis

It is possible to perform a noise analysis using noise density data available for the sensors, resistors, amplifiers and the ADC, in an attempt to judge how accurate the circuit will be. However this will give a very generous result since it neglects one of the main sources of noise, interference. It is the author's experience that sources of interference can be considerable, especially when



Noise autocorrelation from the dummy sensor.



Noise autocorrelation from a magnetic field sensor.

Figure 6: Autocorrelation of the sensor noise.

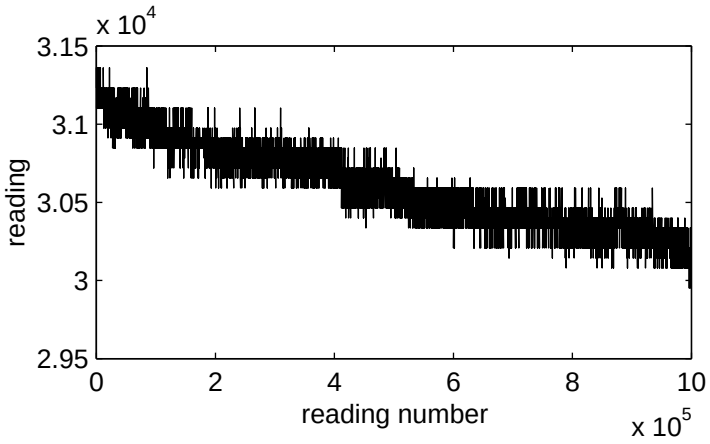


Figure 7: Plot of the sensor noise against time.

there are small signals and high frequency circuitry (the microcontroller is clocked at 10MHz) in close proximity. The best way, therefore, to get accurate information about the noise is to run the circuit for a considerable period of time to acquire a large set of data for analysis. One million readings were taken from the four sensors and a dummy channel. The dummy channel consists of a potentiometer connected between the rails used to supply the magnetic field sensors. The purpose of this is to provide a comparison so that it can be determined if noise in the amplifier circuitry has a significant effect.

The numbers received from the ADC are 10 bits long (varying from 0–1023); The noise from the sensors is approximately gaussian, but it is dis-

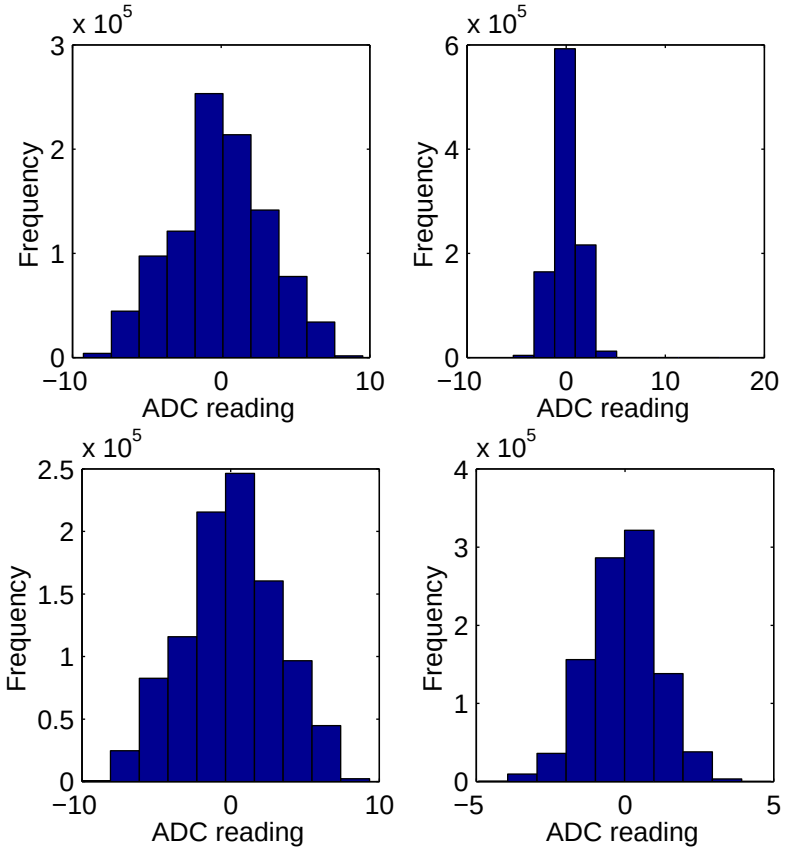


Figure 8: Histograms of the measured values from the sensors. All ADC readings have been shifted to have zero mean.

torted by quantization of the readings. The standard deviation from the sensors varies from 3.46 to 5.10. The standard deviation of the noise on the dummy sensor is much lower, being only 1.04. The main question is where the extra noise comes from. Figure 6 shows the autocorrelation of the noise for one of the sensors and for the dummy sensor. This shows that the noise on the dummy sensor is really white noise, but the noise on the other sensor is strongly correlated in time. Figure 7 shows the data from one of the sensors plotted against time. It shows a very clear trend. The data from the other sensors is very similar. It is unclear what causes this trend, but the most likely explanation is that the magnetization of the sensors is altered by their proximity to the monitor of the computer used to record the data. In order to calculate the noise standard deviation, the trend has to be removed. This was done by calculating the local mean of the data for each point and subtracting this mean. The mean is calculated from 1000 points around the point of interest. This has almost no effect on the standard deviation of the data for the dummy sensor, reducing it by 0.01%.

The corrected data has much lower standard deviations, 3.16, 1.27, 3.19 and 1.15 for sensors 1 to 4, The values are still above the standard deviation of the data from the dummy sensor. Histograms of the sensor noise are

shown in Figure 8. The noise is not completely gaussian, since the readings are quantized.

Two of the sensors have significantly higher noise than the other two. The two with low noise are aligned along the same axis. It is likely that the fields from the nearby monitor are nearly aligned with these sensors, and so induce little noise.

Placing an oscilloscope on the supply to the magnetic field sensors reveals that there is a 10MHz[2] oscillation in the voltage. This changes the input voltage into the bridge and results in a change in the differential voltage out, which is duly amplified and fed in to the ADC. This can be counteracted in several ways. In hardware, the voltage source to the bridge could be filtered with an LRC (inductor, resistor, capacitor) circuit. In software, the data could be sampled at a faster rate and filtered in software.

Since the noise is approximately Gaussian, the data can be converted into the probability that an error of 1 degree will occur in the measured angle when the sensors are at 45° to the magnetic field. The sensors outputs are measured by a 10 bit ADC (1024 values) and one part in 191 is required for 1° precision at 45° (see Section 3.1.1). This corresponds to 5.36 measured units. For an error of 1° to occur, with the best case estimate for the standard deviation, the data must be out by 4.74 standard deviations. By integrating the probability density function from 4.74 to ∞, it can be shown that the probability of an incorrect reading occuring is 1.05×10^{-6}. At a sampling rate of 100Hz, this corresponds to one incorrect reading every 2.6 hours. With the worst case standard deviation, the sensors will produce approximately 4 incorrect readings each second. This probability is relatively high. To reduce the probability of error to an acceptable level, the standard deviation of the data needs to be reduced. Taking the mean of n samples reduces the standard deviation by a factor of $\sqrt{n}$ [6]. Sampling at 600Hz and averaging reduces the error rate to 5 errors per hour which is an acceptable rate. This supports the assertion made in Section 3.1.1 that the ordinary amplifiers were suitable.

3.1.3 Calibration

As discussed in the Section 3.1.1, the sensor responses may vary by a factor of up to 1.6. Further, there is an arbitrary offset, partly due to the requirement that the sensor output has to be shifted to the 0–5V range required by the ADC and partly due to the inherent offset present in the sensors. Also, the sensitivity is altered slightly by a degaussing pulse. In order to calibrate the sensor, a set of readings taken with the sensor at a known angle to the magnetic field is needed.

The most straightforward way to do this is to start with the sensor aligned with magnetic north[3], with an elevation of 70°. Rotating the sensor by an angle α about an axis orthogonal to the magnetic field direction (for instance

[2]This is the clock frequency of the microcontroller. See Section 3.3.

[3]Magnetic north, as opposed to true north must be used since the deviation is significant with the high accuracies required.

a line running east-west) results in an angle α between the sensor axis and the magnetic field direction. Plotting the reading from the sensor against $\cos\alpha$ should (ideally) result in a straight line. By fitting a straight line to this data (for instance, using least squares), $\cos\alpha$ can be calculated from the sensor reading.

It must be noted that various devices (such as computer monitors) produce quite strong local magnetic fields. The readings must be taken a suitable distance from any of these devices, or they will be invalidated. Problems with interference from such stray magnetic fields prevented a calibration from being performed.

3.2 Measuring 3rd Degree of Freedom

Since the magnetic field sensors are being used for angular measurements, it follows that a fourth measurement is required in order to determine all the angles. This must be done by taking one more angular measurement using a system which does not depend on the Earth's magnetic field. There are two possibilities here: either measuring the rotation about the Z axis or measuring the angle between one of the other axes and its projection onto the ground plane using range finding. These angles approximate the pitch and roll if the pitch and roll are small. It is difficult to get range finding techniques to work accurately at a high speed [3]. For this reason, it is best to measure the rotation about the Z axis directly.

One way to do this is to have a fixed beacon placed away from the helicopter and use a sensor fixed to the helicopter to determine the angle to the beacon. This angle gives the rotation about the Z axis directly. There are two ways to determine the angle to the beacon. One is to have a sensor on board which continually tracks the beacon. Measuring the angular position of the tracking assembly gives the required angle.

Another method is to use a scanning receiver which continuously sweeps round. This receives a maximum signal when the detector points towards the source. This has several advantages over the previous method. Position measurement is potentially simpler. This will be discussed in Section 3.2.5. Also, it can measure the angle to more than one source. If more than one angle is measured, triangulation can be used to determine the position of the helicopter. Furthermore, since the device only has to spin around, rather than change direction rapidly, it is easier to build. This method was tried initially (see Section 3.2.6).

3.2.1 Implementation of a Scanning Sensor

The first choice to be made was what was to be emitted and received. Using induction would interfere with the magnetic field sensors. A high frequency radio link would require circuits operating at over 1GHz which are very difficult to build in practice. The other two choices were using an infra-red or optical link. An optical link was the initial choice because it is easier to

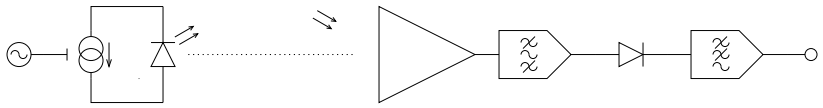


Figure 9: IR beacon and receiver block diagram.

debug, given that the emitted light can be seen. After a small amount of experimentation, the link was changed to an infra-red. Infra red LEDs are inexpensive and can emit much higher powers than inexpensive optical LEDs (0.1W compared to 0.01W), making the signals much easier to receive.

A system block diagram is shown in Figure 9. The output of the infra-red LEDs was modulated at 10kHz. This serves two purposes. Firstly, the received signal can be filtered to remove interference from ambient light and mains noise. Secondly, by having several sources at different frequencies, the angle to each of these can be measured using one sensor. This enables it to become a position measurement system as well.

The first part of the system is a frequency source connected to a voltage controlled current source, driving the LED. The controlled current source is required because the light output power of an LED is approximately proportional to current input, whereas it is exponentially related to voltage input.

3.2.2 Infra-red Link

There are many considerations to be taken into account in selecting the 'best' transceiver components for the infra-red link. From the perspective of power to cost ratio, mid range emitters (of the order of 160mW) provide the best solution. Another consideration is the angular distribution of the beam; a relatively wide beam angle is required to prevent the signal being lost if the sensor moves too far. The third consideration is whether the wavelength of the peak output for the emitter matches the wavelength of peak response for the receiver.

There are also many kinds of infra-red receiver:

1. Photodiodes,
2. Photo-transistors and photodarlingtons,
3. LDRs (light dependent resistors),
4. Monolithic sensor packages of different types:
    - Linear
    - TTL
    - Remote control.

A photodiode is a PN junction which produces a current output proportional to the intensity of the incident radiation. It is possible to use an LED as a photodiode, and this would result in a sensor perfectly matched to the

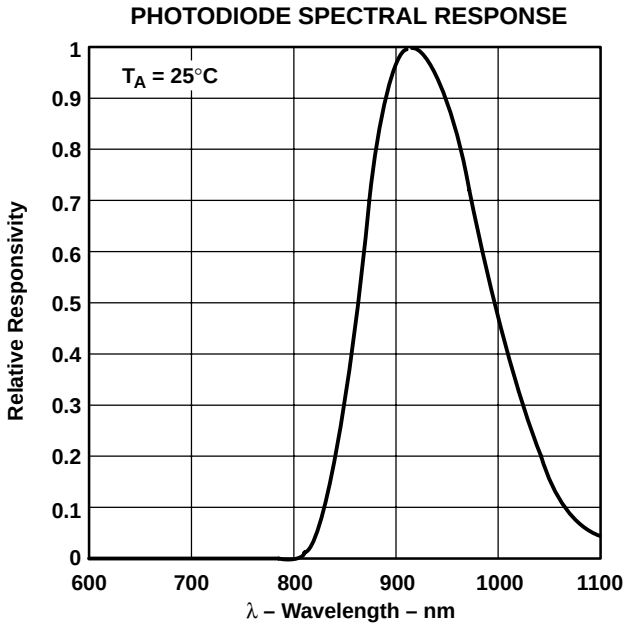


Figure 10: TSL261 spectral response. Reproduced with the kind permission of TAOS Inc.

emitter, but its performance would not be good since LEDs are not explicitly designed to act as photodiodes.

Photo transistors rely on radiation incident on the base allowing a current to pass through. These can be used in much the same way as photodiodes. Whilst they tend to have a higher gain they have a slower response. Photodarlington transistors have a still higher gain and slower response.

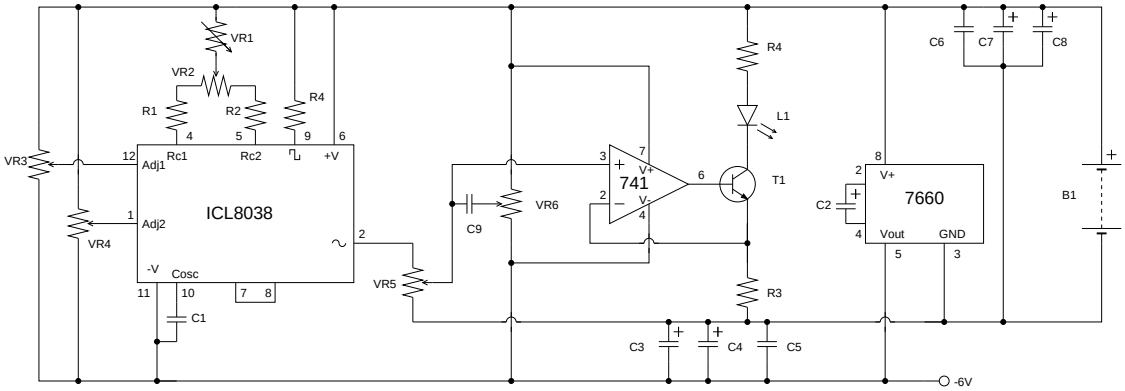
The response time of LDRs (light dependent resistors) is generally far too slow for them to be used in this application.

Monolithic sensors provide considerable advantages over discrete components. Firstly, they simplify the circuit since a large amount of the amplification is performed in the package. Secondly, since the construction is physically much smaller, mains interference induced in any loops in the circuit is greatly reduced. Mains interference is often a problem in circuits dealing with very small signals. The sensors come in several types. Using a sensor with a linear response provides the greatest flexibility and is used.

The pair of sensors used is described in Table 1. A feasibility study shows that they are suitable: the incident radiation on the sensor (assuming perfect alignment) is 60mW/Sr. At a distance of 2m, this becomes 119nW/cm^2. Multiplying this by the sensor responsivity [2] gives a voltage response of 2.75mV which is easily detectable. The detector also comes with a built in visible light filter which helps to reduce interference. Although the emitter and receivers do not have maximum responses at the same frequency, the receiver does have a very strong response at 910nm [2] (see Figure 10).

HIRL 5020 emitter	
Manufacturer	Rodan
Wavelength of peak output	940nm
Peak radiant intensity	60mW/Sr
TSL 261 amplified photodiode	
Manufacturer	TAOS (Texas Advanced Optical Solutions)
Wavelength of peak response	910nm
Responsivity	23mV/μW/Sr

Table 1: The emitter/detector pair used.



C_1	10nF	polypropylene	VR_1	2.2kΩ	frequency adjust
$C_{2,3,6}$	10μF	electrolytic	VR_2	2.2kΩ	mark/space adjust
$C_{4,7}$	470μF	electrolytic	$VR_{3,4}$	100kΩ	sine wave distortion
$C_{5,8,9}$	100nF	ceramic	VR_5	4.7kΩ	amplitude
$R_{1,\ 2}$	1kΩ		VR_6	100kΩ	level shift
$R_{3,4}$	20Ω		T_1	BFY51	
R_5	10kΩ		L_1	HIRL5020	
B_1	6V	PPJ96			

Figure 11: IR transmitter circuit.

3.2.3 Transmitter Circuit

The transmitter circuit is shown in Figure 11. The ICL8038 chip is set up to produce a sine wave, which is adjusted from about ±4.5V to 0–5V, using VR_5 and VR_6. This is fed into a voltage controlled current sink with a transconductance of 10mS. This drives a sinusoidal current of 0–50mA through the infra-red LED. The -6V supply is generated by the 7660 chip. There were a few difficulties with the circuit arising from undocumented effects. First, the oscillation frequency of the 8038 is not as independent of the supply voltage as the data sheet would have you believe. Secondly, the suggested 10μF smoothing capacitor on the output of the 7660 chip was not enough to prevent slight variations in the supply voltage under a small load. The result was a small oscillation in the supply voltage, at about one hundred cycles per second. The frequency was probably caused by beating between the ICL8038 (nominally running at 10kHz) and the 7660 (nominally running at 10kHz). Since the band pass filter in the receiver was very sharp (see Section 3.2.4), the result was a significant wobble in the amplitude of the received signal. This was solved by adding 470μF capacitors between the 0V rail and the supply rails.

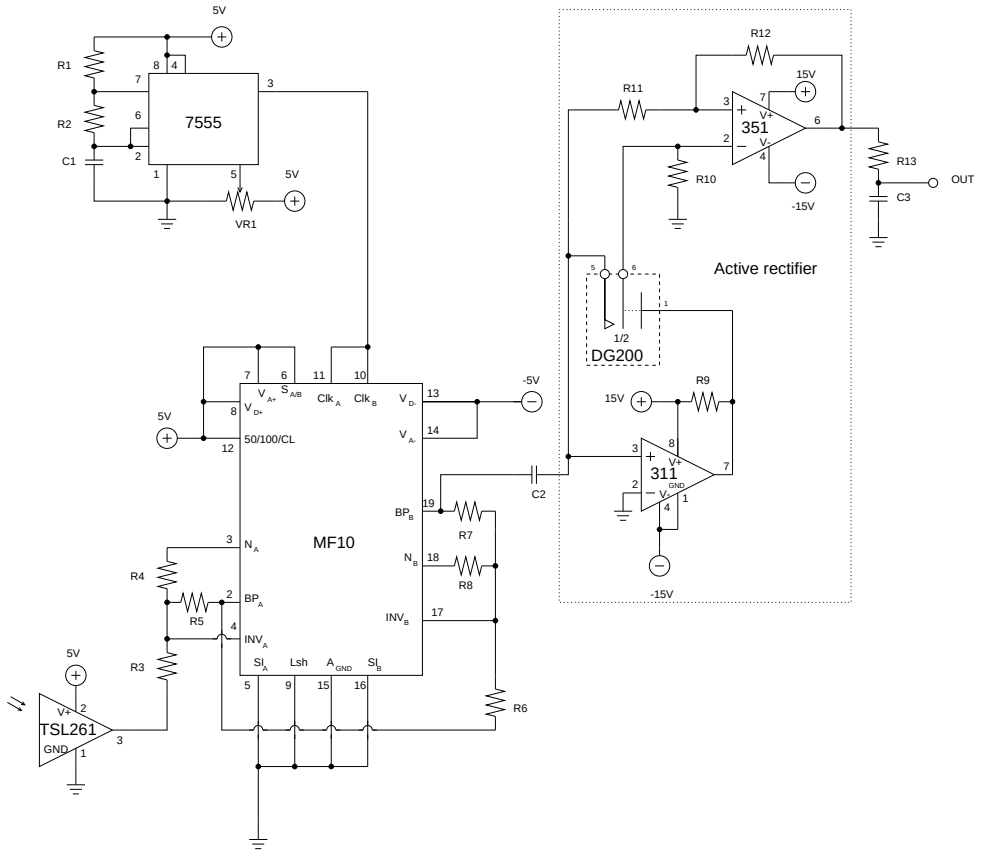
Furthermore, the sine wave oscillator can only produce a low distortion sine wave when its supply voltage is significantly higher than the minimum quoted in the data sheet. This results in wasted power transmission.

3.2.4 Receiver Circuit

The receiver circuit is shown in Figure 12. The circuit uses the MF10 chip to create two 2^{nd} order band pass filters each with a Q (quality factor) of 20, making a 4^{th} order filter in total. The MF10 is a switched capacitor filter. The centre frequency of the filters is in this case one fiftieth of the input clock frequency. The circuit is clocked by a 7555 timer. The timing capacitor used is too small on its own to produce 500kHz [7] but the capacitance between circuit board tracks has a significant effect so the overall capacitance is larger.

The maximum AC output of the TSL261 is approximately 0.1V peak to peak (when the emitter is placed 30cm from the receiver). The maximum output range of the MF10 chip is about ±4V (when running off a 5V supply), so the gain was set to 40. The low pass filter on the output has a cutoff frequency of 100 Hz.

Rectification is not performed using a diode since the 0.2V drop (from a germanium PN junction) is far larger than some of the signals that will be received. An active rectifier is used instead. This part of the circuit inverts the incoming signal if it is less than 0V but leaves it unchanged otherwise. This is limited by the voltage offset of the LM311 comparator used, which is typically 2mV [9].



C_1	47pF	polypropylene	$R_{4,7}$	1.1kΩ
C_2	100nF	ceramic	$R_{5,8}$	22kΩ
C_3	10nF	ceramic	R_9	10kΩ
R_1	1kΩ		$R_{10,11,12}$	10kΩ
R_2	28kΩ		R_{13}	1kΩ
$R_{3,6}$	22kΩ			

Figure 12: Infra-red receiver circuit.

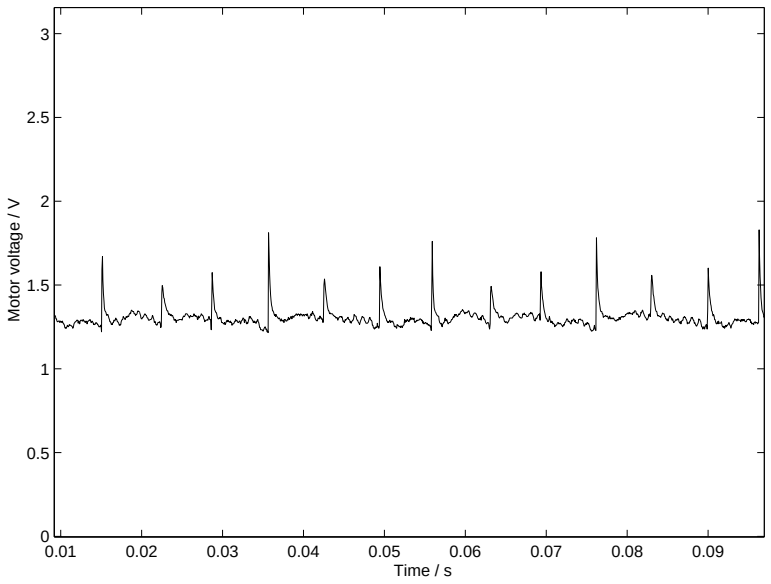


Figure 13: Plot of motor voltage with a 1Ω series resistor.

3.2.5 Sensor Angular Position Measurement

In order to determine the angle of the helicopter, the angle of the sensor must be found. There are several ways to do this. One way is to rotate the sensor in a deterministic manner using a stepper motor. Another way involves rotating the sensor with a normal motor and then measuring the position of the shaft that the sensor lies on (for instance by using a precision potentiometer or slotted disc).

A third option is also available. In a DC motor, the action of commutation causes brief interruptions to the current as the carbon brushes pass from one commutator segment to the next. A three pole motor—the most common kind of low power motor—produces six such interruptions during one revolution (see Figure 13). These can be counted to determine the position. By using an N:1 gear box, the motor position can be measured to an accuracy of $\left(\frac{360}{6N}\right)^{\circ}$.

This this method was used initially because the electrical and mechanical construction required is simpler than for the other method. The block diagram of the circuitry used to produce countable pulses from the motor is shown in Figure 14.

3.2.6 Construction, Performance and Analysis

The main issue in construction was to allow the signals to go from the continuously revolving sensor to the other circuitry. The solution was to use slip rings. A picture of the scanning sensor mounted on its gear box is shown in Figure 15.

Initially, the position sensing circuitry performed very well (as shown in Figure 16), with a voltage input of up to 2.4V. However, repeated running

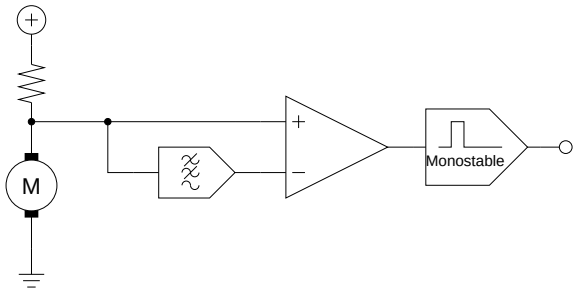


Figure 14: Circuit used to extract pulses from the motor in a countable form.

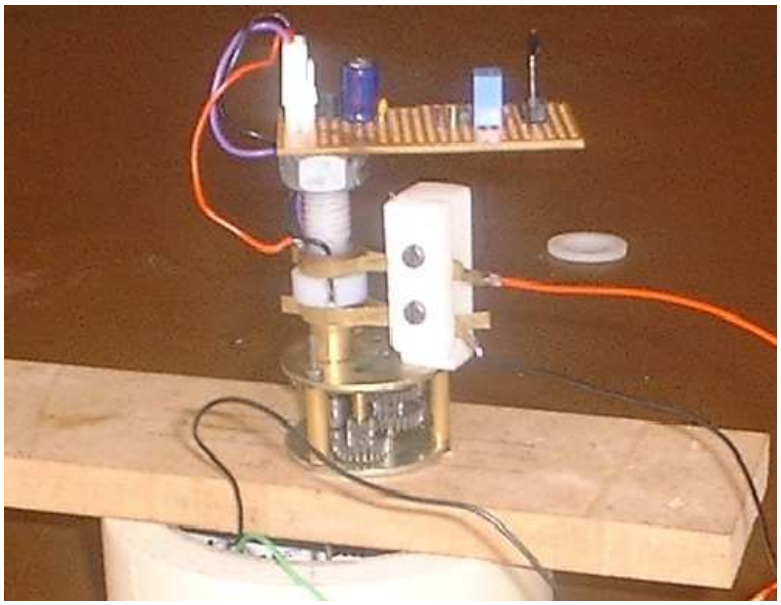


Figure 15: Scanning sensor.

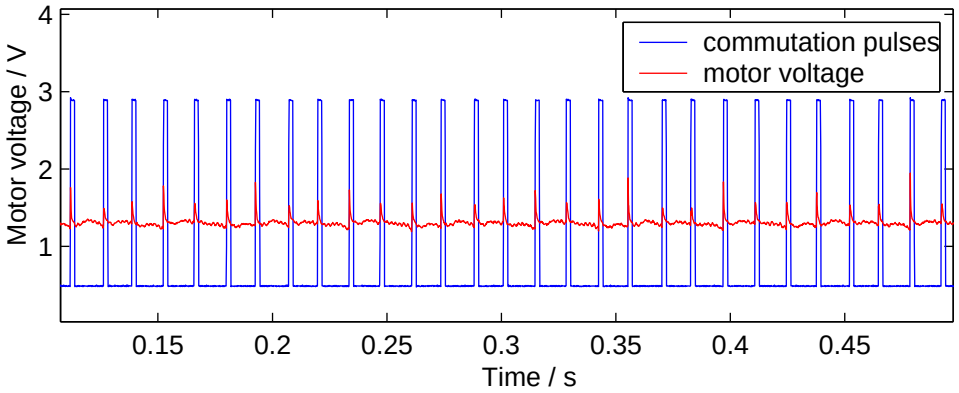


Figure 16: Plot of motor voltage and the pulses generated from it.

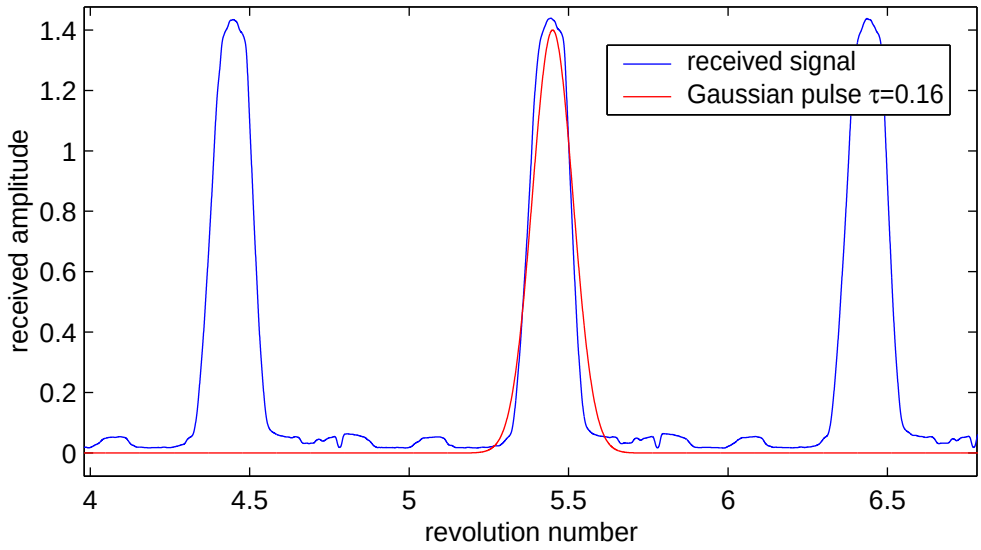


Figure 17: A plot of amplitude received as the sensor revolves. A Gaussian pulse is also plotted for comparison.

of the system for 10 minutes at a time, led to the position counter randomly losing about 60° every 20 or so revolutions. After running for a while, the performance of the circuit degraded badly. This was probably due to wear on the motor causing a much noisier signal. The missed pulses cluster in time.

The way to correct this would be to have an additional device to measure the angle in order to recalibrate the measurement done by counting the pulses. This would defeat the purpose of counting the pulses to find the position.

Figure 17 shows the amplitude of the received signal for several sweeps. As can be seen, although the peak is wide, it is very well defined. The data has been low pass filtered to remove noise. A Gaussian pulse is defined up to an arbitrary scaling factor by:

$$G(x) = e^{-\pi\left(\frac{x}{\tau}\right)^2}$$

As is also shown on the figure, the angular response can be approximated with a Gaussian pulse where $\tau \approx 0.16$. Taking the Fourier transform of the pulse shows that the amplitude of the angular frequency drops to 1% of its maximum value when the angular frequency reaches 7.57 cycles per revolution. The Nyquist sampling theorem states that the sampling rate must be double the highest frequency in order to reconstruct the original signal. This implies that by using only 16 samples, the original signal can be reconstructed. Therefore the centre of the peak can be found with only 16 samples. Figure 18 shows the estimated positions of the peaks.

The centre is estimated by fitting a quadratic curve to the highest three

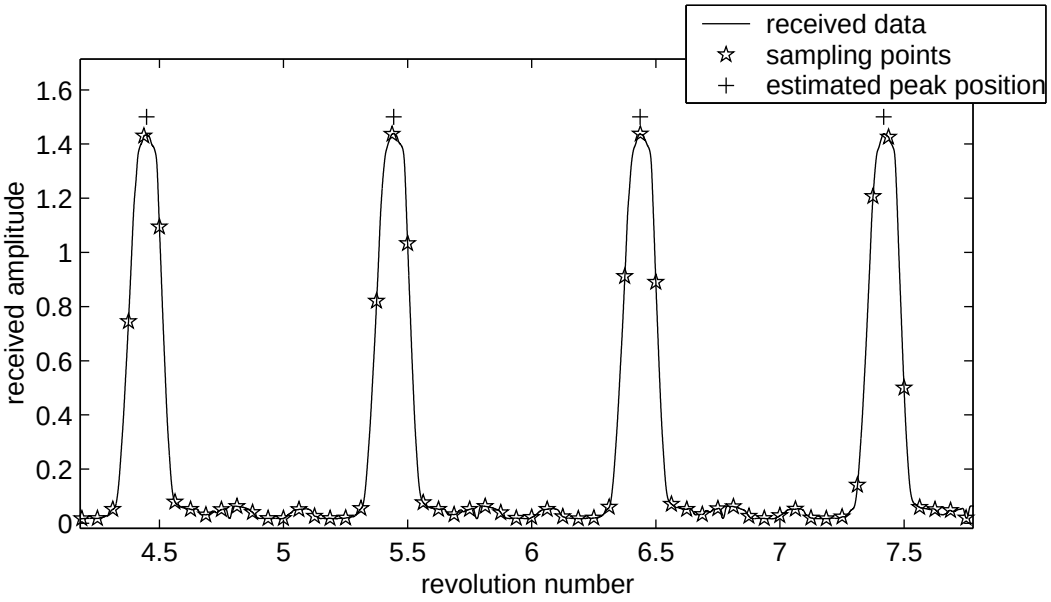


Figure 18: Plot of amplitude received as the sensor revoles with 16 sample points and the estimated peak position marked on.

points using the following formula:

$$x = -\frac{\frac{1}{2}(y_3 - y_1)}{y_3 - 2y_2 + y_1}$$

where the three points lie at $(-1, y_1)$, $(0, y_2)$ and $(1, y_3)$. The peak of the quadratic curve is used as the estimate of the peak of the data. Since the data is not in the shape of a Gaussian, and calculations for fitting quadratic curves are faster than those for Gaussians, a quadratic fit was used.

Sweeping the sensor around and taking sixteen samples does not solve any of the initial problems, but it does show that another method of sweeping the sensor is possible. Instead of one moving sensor, sixteen sensors can be placed in a circle. The sensor can now be 'swept' around by taking a reading from each of the sensors in turn. This has many advantages over the previous method. It removes the need to have an accurate measurement of the angle of the sensor, since this is known exactly. The sensor can be swept around very much faster since there is no longer any friction to battle against.

3.2.7 Construction and Performance of a Non Mechanical Sensor

The original sensor circuit was modified so that sixteen infra red sensors fed in to the filter via a sixteen to one analogue multiplexer. The full circuit is shown in Appendix B.1. The physical construction is shown in Figure 19. The cylinder that the infra-red sensors are attached to is 8cm in diameter. In the original circuit, the low pass filter on the output was fast enough. However to get a high sampling rate with the new system, a higher order filter is needed. There was insufficient time to build one of these, so the

Figure 19: 16 way infra red sensor.

sampling rate for the sensor was lowered. Readings were taken at the rate of 100Hz, but since there are sixteen sensors, the effective sampling rate is much lower.

To assess the accuracy of the sensor, 10,000 readings were taken with the emitter at a distance of one meter. In most cases, the distribution of readings was approximately Gaussian, with the standard deviation varying between 0.110 and 9.79. Figure 20 shows a histogram of the measured angles for sensor number 11, the sensor with the very high standard deviation in its readings. This sensor was defective and had a much lower response than the sensors around it. Replacing the sensor brought the standard deviation down to 0.461; that left the next highest at 1.85. The cause for the high deviation seems to be similar, but by this point, there were no more spare sensors. To improve matters, all of the sensors should be calibrated before, so that their outputs can be normalized, before attempting to calculate an angle from the readings. However, due to time constraints this was not possible. With this refinement, the maximum standard deviation will probably be around 0.5 degrees. This would result in there being a 2% chance of the reading not being within 1° of the correct value.

3.3 Implementation

The sensors on the helicopter need to have their values read, converted in to a digital signal and sent to a computer over a communications link. The most suitable way of doing this is with a microcontroller IC (integrated circuit), since this contains all of the circuitry needed to perform the required functions. These functions are an ADC, a UART (universal asynchronous receiver/transmitter)—used for communications to the com-

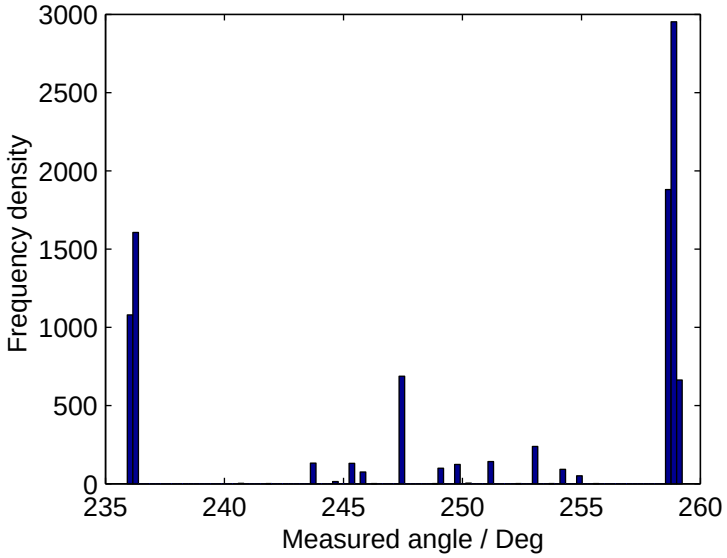


Figure 20: Histogram of angles measured for sensor number 11.

Word	Offset	Data	
0	0	255	Header
	1	255	
1	2	MSB	Potentiometer (dummy sensor)
	3	LSB	
2	4	MSB	Magnetic sensor 1
	5	LSB	
3	6	MSB	Magnetic sensor 2
	7	LSB	
4	8	MSB	Magnetic sensor 3
	9	LSB	
5	10	MSB	Magnetic sensor 4
	11	LSB	
6	12	4 bits	IR sensor number
	13	0	Padding
7	14	MSB	IR sensor
	15	LSB	

Table 2: Data packet.

puter's serial port—and digital outputs. The microcontroller chosen was the PIC16F877 (manufactured by Microchip Inc.).

The PIC contains a 10 bit ADC (exceeding the specification) with 8 inputs (enough for all the sensors). It also has enough output lines to drive the infra-red sensor multiplexer and the degaussing strap for the magnetic sensors.

The communication uses a wire piggybacking on the power cable to transmit serial data (using the RS232 protocol) to the computer. No handshaking is used; it is assumed that the computer is fast enough to always be able to receive the data. Since the system runs at a fixed sampling rate (100Hz), then if the computer is not ready to receive data, it would fail anyway. The data is sent at 56kb/s, 8 bits per byte in packets of 7 words consisting of 2 bytes each. The data packet specification is given in Table 2. It should be noted that the header sequence (one 16 bit word with all bits set to 1) can only occur in the header since the other words contain at most 10 bits of data, with the spare bits zero. All words are transmitted high byte first. The ADC data is 10 bits and one word from the ADC is packed in to the highest 10 bits of one 16 bit word. The lowest 6 bits of the word are set to zero.

The circuitry for the PIC includes a 10MHz crystal for the clock. This clock rate is fast enough to allow the PIC to perform all the required operations. This clock rate also allows the PIC to transmit serial data at 56kb/s with a very low timing error [10].

The communication programs for the PIC and computer are given in Appendix C and the full on board sensor circuit is given in Appendix B.1.

3.4 Angles from Measurements

The sensor suite provides four numbers: the dot product of the three magnetic sensor axes with the Earth's magnetic field and the helicopter's rotation about its Z-axis (ψ). What is needed is the pitch and roll (ϕ and θ) since it is these that must be controlled to zero in order to get the helicopter to hover. The three angles are θ, ϕ and ψ are defined in Figure 21.

To get the helicopter from the position where it is aligned with the global axes to the final position, a rotation is required. This can be expressed in terms of rotations about the helicopter's local X, Y and Z axes by angles of θ, ϕ and ψ. These three rotation matrices are $\mathbf{R}_x$, $\mathbf{R}_y$ and $\mathbf{R}_z$. The rotation $\mathbf{R}_x$ is given by:

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix}$$

The next rotation is along the line $\mathbf{R}_x \times [\ 0\ 1\ 0\ ]^\mathsf{T}$, i.e. along the new Y axis for the helicopter. The rotation matrix for a rotation by an angle α along a

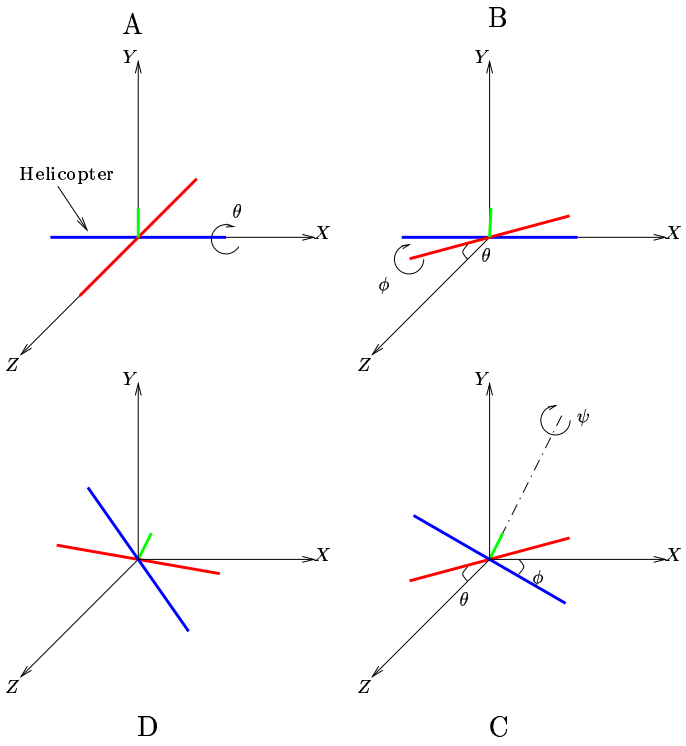


Figure 21: Definition of θ, ϕ and ψ.

line $\mathbf{m} = [\ m_1\ m_2\ m_3\ ]^\mathsf{T}$ is given by [5]:

$$\mathbf{P}(\mathbf{m}, \alpha) = \mathbf{I}\cos\alpha + (1 - \cos\alpha)\,\mathbf{m}\,\mathbf{m}^\mathsf{T} + \sin\alpha \begin{bmatrix} 0 & -m_3 & m_2 \\ m_3 & 0 & -m_1 \\ -m_2 & m_1 & 0 \end{bmatrix}$$

therefore

$$\mathbf{R}_y = \mathbf{P}\left(\mathbf{R}_x \times [\ 0\ 1\ 0\ ]^\mathsf{T}, \phi\right).$$

The final rotation is along the line given by $\mathbf{R}_y\mathbf{R}_x \times [\ 0\ 0\ 1\ ]^\mathsf{T}$. This gives an overall rotation, $\mathbf{R}$:

$$\mathbf{R} = \mathbf{R}_z\mathbf{R}_y\mathbf{R}_x$$

where

$$\mathbf{R}_z = \mathbf{P}\left(\mathbf{R}_y\mathbf{R}_x \times [\ 0\ 0\ 1\ ]^\mathsf{T}, \psi\right)$$

The sensors are aligned with the helicopter axes so, in the local axes, the matrix of sensor axes $\mathbf{S}_l$ is given by:

$$\mathbf{S}_l = \mathbf{I}$$

Therefore, in global axes, the matrix of sensor vectors is given by:

$$\mathbf{S} = [\ \mathbf{s}_1\ \mathbf{s}_2\ \mathbf{s}_3\ ] = \mathbf{R}.$$

A sensor reading, d_i, is given by the dot product of the sensor axis:

$$d_i = \mathbf{s}_i^\mathsf{T} \cdot \hat{\mathbf{h}}$$

where $\hat{\mathbf{h}}$ is the global magnetic field. $\hat{\mathbf{h}}$ is in the X, Z plane at an angle of ζ to the X axis ($\zeta \approx 70°$). The vector of sensor readings, $\mathbf{d}$, is given by:

$$\begin{aligned}
\mathbf{d} &= \mathbf{S}^{\mathsf{T}}\hat{\mathbf{h}} \\
&= \mathbf{R}^{\mathsf{T}}\hat{\mathbf{h}} \\
&= \begin{bmatrix} C_\psi C_\phi C_\zeta + S_\zeta S_\phi S_\theta C_\phi + S_\zeta C_\theta S_\phi \\ -S_\psi C_\phi C_\zeta + S_\zeta C_\theta C_\psi - S_\zeta S_\psi S_\phi S_\theta \\ S_\phi C_\zeta - S_\theta C_\phi S_\zeta \end{bmatrix}
\end{aligned}$$

where:

$$\begin{aligned}
C_\alpha &= \cos\alpha \\
S_\alpha &= \sin\alpha
\end{aligned}$$

This can be equated to the actual normalised sensor readings $\mathbf{r}$, so that:

$$\mathbf{d} - \mathbf{r} = [\ 0\ 0\ 0\ ]^{\mathsf{T}}$$

which can be expressed in the functional relationship:

$$\mathbf{F}(\mathbf{\Theta}) = 0$$

where:

$$\mathbf{\Theta} = \begin{bmatrix} \theta \\ \phi \end{bmatrix}.$$

This is a nonlinear equation in two unknowns (ζ and ψ are known) and can be solved by the Newton-Raphson method. In the case where the sensors are ideal, the equations will have a redundancy (the reason for this is given in section 3). In the real world there will be a slight inconsistency. The Newton-Raphson method for multidimensional equations is [4]:

$$\mathbf{\Theta}_{\text{new}} = \mathbf{\Theta}_{\text{old}} + \delta\mathbf{\Theta}$$

where:

$$\begin{aligned}
\mathbf{J}\delta\mathbf{\Theta} &= -\mathbf{F}, \\
J_{ij} &\equiv \tfrac{\partial F_i}{\partial \Theta_j}.
\end{aligned}$$

Since $\mathbf{J}$ is not square, it can not be inverted. This problem can be solved in two ways. The redundancy can be removed or the pseudo-inverse of $\mathbf{J}$ can be taken to solve for the least squares approximation of $\delta\mathbf{\Theta}$:

$$\delta\mathbf{\Theta} = -\left(\mathbf{J}^{\mathsf{T}}\mathbf{J}\right)^{-1}\mathbf{J}^{\mathsf{T}}\mathbf{F}$$

The Newton-Raphson method suffers from the problem that it can wander very far from the solution if the initial guess is bad. However, in this case it is suitable since the initial position of the helicopter can be known almost exactly and every subsequent position will be quite close to the previous position, meaning that the first guess is always good. Another problem is that the equation may become singular, but there are established numerical procedures for dealing with this [4].

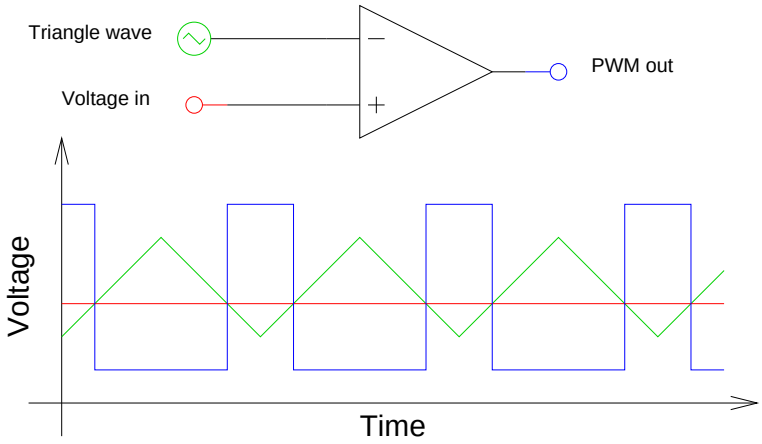


Figure 22: Using a triangle wave to produce PWM.

4 Power Controller

The power controller must be able to control the power through four independent motors simultaneously at a voltage of up to 12V and with currents of up to 5A per motor. Further, the device needs to be interfaced with digital hardware easily. In the first instance, it is located on the ground and is driven from an IBM compatible computer. In future, it is envisaged that the power controller will be on board the helicopter and it was designed with this in mind.

4.1 Pulse Width Modulation

The standard way to achieve power control in a device such as a motor is to use PWM (pulse width modulation). This involves feeding a square wave of fixed frequency and voltage into the motor, where the on time and off time of the wave are varied. This controls the average voltage and hence the power. PWM is a very efficient method of doing this, since the power components (such as power transistors) are either fully on or fully off. When they are fully off, they have a very high resistance and therefore dissipate almost no power. When fully on, they can have resistances as low as 0.01Ω and therefore dissipate very little power. Other methods involve controlling the voltage by dropping excess voltage across a dummy load and hence dissipating the excess power as heat. This is not only inefficient but problematic. It is a non trivial problem to dissipate up to 240W without allowing any of the power components to overheat.

PWM can be achieved in a number of ways using hardware and software. A hardware implementation is more suitable than a software implementation since failsafe mechanisms (such as current limiting) can be built into hardware easily. As it turns out, this is a very necessary feature and is discussed further in Section 4.4. The easiest way to perform PWM is by comparing a triangle wave to a steady voltage. This is shown in Figure 22.

Generation of an accurate triangle wave is most easily performed using an oscillator IC such as the 8030. One problem with the circuit is that when a slowly varying voltage is fed into a comparator, the output can burst into oscillation around the point where it switches. This occurred with one of the comparators in the circuit. It is caused by the high frequencies in the sharp output transition coupling back to the input. The LM339 data sheet [8] (that of the comparator used) suggests causing hysteresis on the output (by using a small amount of positive feedback) to avoid this. This gets around the problem by requiring a quite large change to cause another output transition. This change is more than that caused by the coupling effect.

However, even a small amount of hysteresis would affect the output too much to be used for PWM. The solution used was to short the output to signal ground with a 1nF capacitor. This softens the transition by shorting out very high frequencies and therefore prevents them coupling back to the input. However, since the transition is softened, the power switching components switch less quickly and, as a result, dissipate more power. Since the transition time is very much less than the period, the effect is not significant except at low power. At low power (i.e. narrow pulses), the output actually fails to reach the maximum value. However, this does not matter because the motors will not be used at very low power as they will not generate enough lift.

4.2 Analogue Demultiplexing

The PWM circuit drives four motors and therefore requires four analogue inputs. There are no easily available I/O devices for the PC which have four analogue output channels (this also applies to the PIC microcontroller). The solution is to multiplex the analogue signals on to a single wire (so that a one channel analogue output can be used) and use digital signals to decide which channel the analogue signal corresponds to. The block diagram is shown in Figure 23. The clock clocks the sample-and-hold IC selected by the channel select inputs. This remembers the voltage for the selected channel and stores it until it is clocked again. The clock signal needs to last for at least $10\mu s$ in order for the voltage from the sample and hold IC (the LF398) to be equal to the voltage in. The logic used is inverse logic, i.e. the channel select input corresponding to the channel to be used must be low and then the clock has to pulse low.

4.3 Power Switching

There are several options available for power switching components. These are:

Thyristors These provide a very low impedance, but they only switch off when the current is interrupted. That means that the PWM clock would have to be synchronized with the motors which is

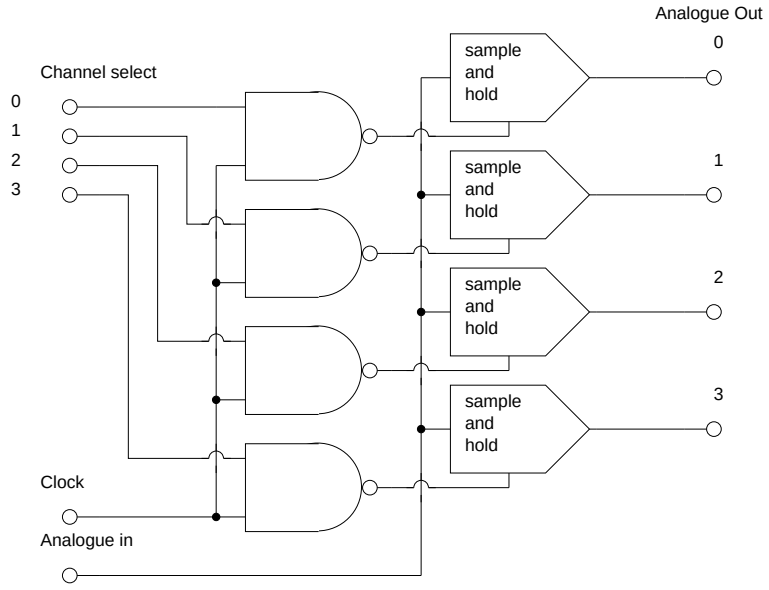


Figure 23: Analogue demultiplexer for the PWM circuit.

not possible, since all of the motors can be going at different speeds. GTO (gate turn off) thyristors are available, which solve these problems, but these are expensive and hard to source.

Solid state relay These provide a very small impedance and are suitable for very high powers, but they are expensive.

BJT (bipolar junction transistor) These are very cheap, but tend to have higher forward voltage drops (and therefore higher power dissipation) than any of the other options listed here. Since high power BJTs tend to have a low gain, they need to be put in a Darlington pair configuration so that they can be driven from small signal sources.

MOSFET These are cheap and tend to have lower resistances than BJTs. They also have an insulating gate, so they can be driven by low power devices such as the comparators used to get the PWM wave form.

MOSFETs of type PHP3055 are used. These have an on resistance of 0.015Ω and can deal with a gate-drain voltage of 20V. Their production has been discontinued since they were obtained for the project.

The rapidly changing voltage applied to the motor causes a large EMF (electro-motive force) to be generated because of the relatively high inductances present in the motor. A reverse biased diode placed across the motor short circuits these out to prevent damage to the power switching components. It is the author's experience that omitting these is a very reliable way to break the MOSFETS.

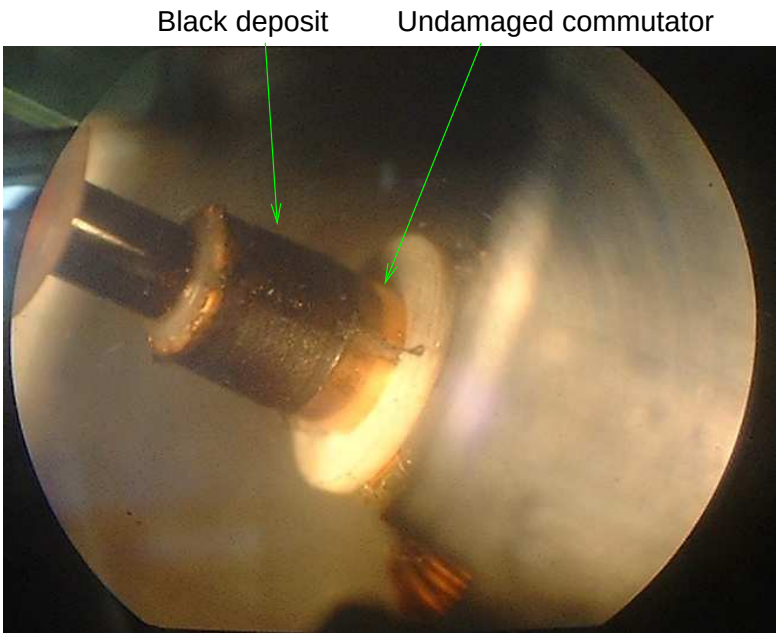


Figure 24: A motor with a burned out commutator.

4.4 Current Limitation

The motors perform well up to a limit of about 6A. Exceeding this limit even by a fairly small amount causes the motors to burn out very rapidly. The destruction is not caused by over heating, but by arcing on the commutator. Figure 24 shows a magnified picture of a commutator of one of the destroyed motors. The area used by the brushes is a blackened. The thick black deposit (likely to be CuO) insulates the commutator and prevents enough current from going through the rotor windings. Since this is the only problem, the motors can be restored by scraping off the black deposit. The performance is as good as the performance of new motors.

The solution to the problem of burning out is to employ a current limiting device in the circuit. Merely turning the voltage down is insufficient, since high currents will flow when the motor is required to produce a large amount of torque. The circuit was modified by adding a current sensing resistor and a feedback loop to reduce the request voltage into the PWM circuit.

The current sensing element is a 0.1Ω resistor constructed from an open coil of nichrome resistance wire (see Figure 25). This solution is not without its problems. At 5A, the resistors dissipate 2.5W and get very hot. The result of this can be seem as blackening on the circuit board underneath one of the resistors. Since the resistors are bare wire, they can experience such temperatures without sustaining any damage, but not without having their resistance altered significantly. The circuit reached equilibrium with a 3A limit. Simply raising the current limit to 5A is not a suitable solution since this would allow current surges to occur before it reached equilibrium. A fan was used to cool the resistors and, because of their open design, they did not

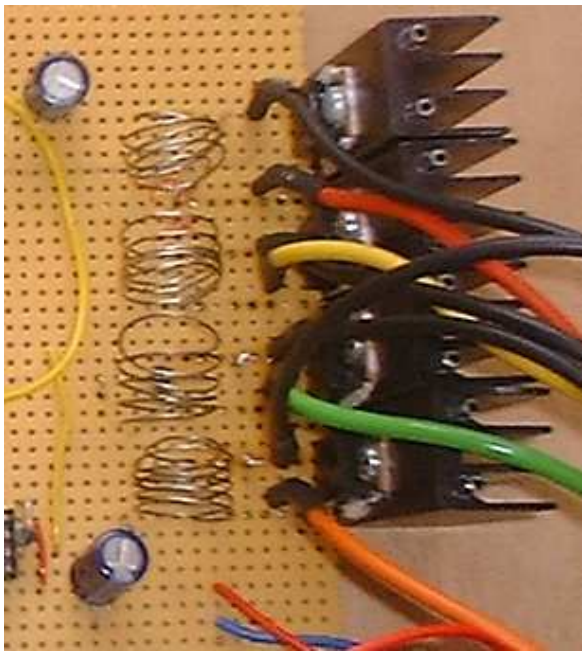


Figure 25: Current sensing resistors.

get warm enough to cause a significant change in the current limit. This does not pose a problem for placing the board on the helicopter, since it already has four much larger 'fans' built in.

If the circuit were to be rebuilt there are more suitable solutions for sensing current. There are two methods. One is to use a Hall effect sensor to detect the current flowing through a loop of wire, by sensing the magnetic field generated. These tend to be large and heavy. The second method relies on the rapidly changing current produced by the PWM circuit. Since a rapidly varying current is passed through the motor, a transformer can be used to sense the current by placing it in series with the motor, and measuring the voltage across it. The transformer can be made small since the power transformed and mutual inductance needed are low. A transformer based system needs to be calibrated with the motors as a load (as opposed to a resistance), since the voltage output depends on the rate of change of current, which is affected by the inductance of the motor.

4.5 Computer Interfacing

The most convenient way of interfacing with the computer is via the parallel port. The parallel port is much simpler to use from both a hardware and software perspective than the serial port. Also, since the PIC has a parallel port, the same hardware can be adapted to work from a PIC. The circuit used is shown in Figure 26. Data transmission is not performed using a standard parallel port protocol, but instead uses one designed for simplicity. The parallel port data lines (D_0–D_7) carry both the data for the DAC (digital

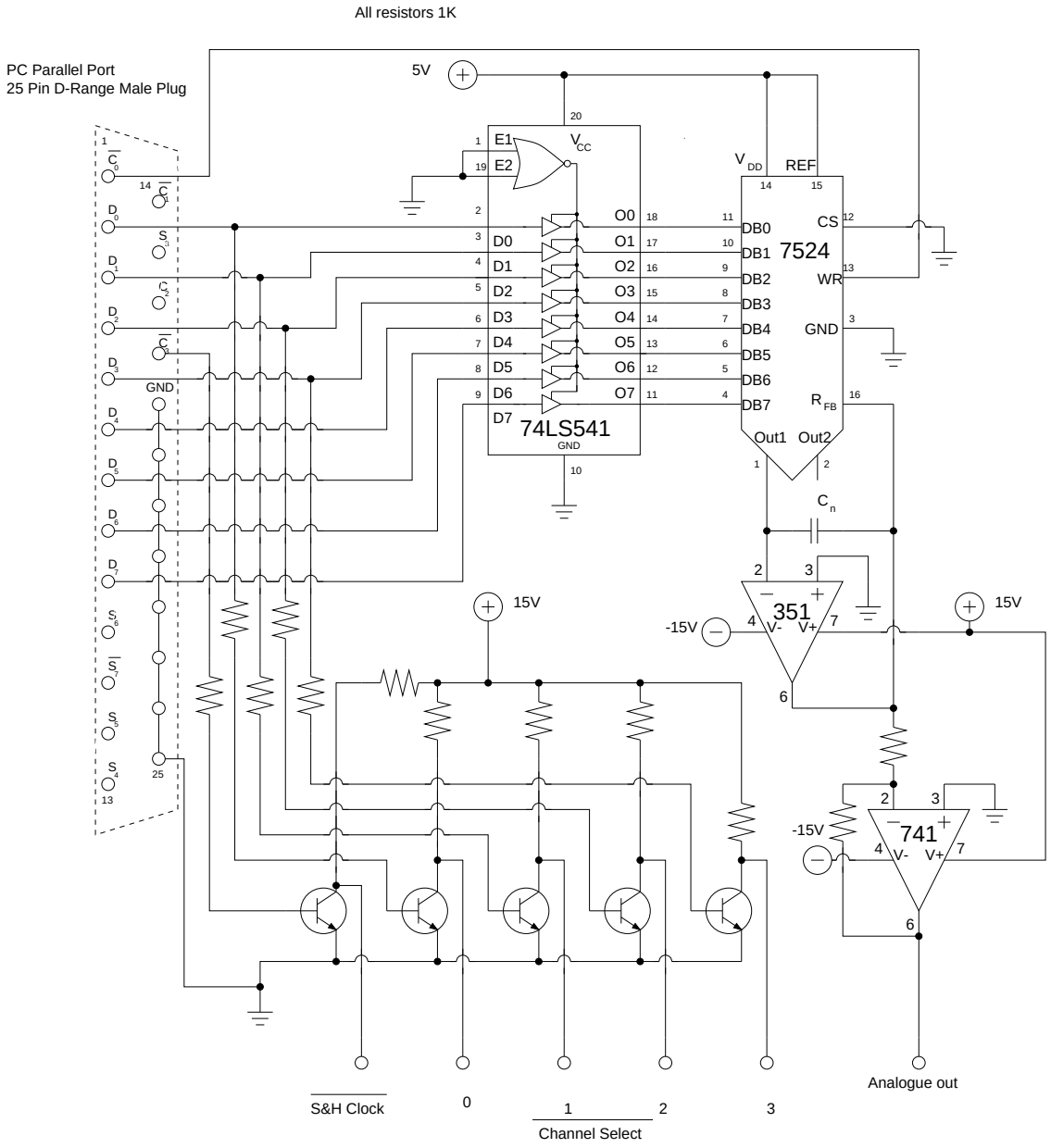


Figure 26: Parallel port interface for the PWM circuit.

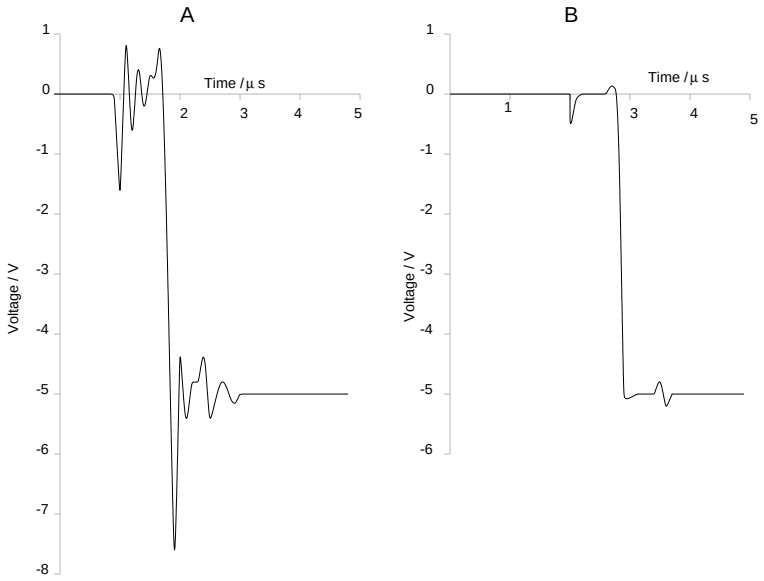


Figure 27: Sketch of the output of the DAC after a large change in value.

to analogue converter) and the channel number. A clock pulse[4] on $\overline{C_0}$ causes the number present on the data lines to be converted to an analogue signal and saved until the next pulse. A negative pulse on $\overline{C_1}$ ($\ldots 0V, 5V, 0V \ldots$) commits the analogue value to the selected PWM channels. The first PWM channel is selected by D_0 being low, etc.

The TTL level outputs (0–5V) from the parallel port need to be coupled to the CMOS level inputs (0–15V) required by the logic chips on the PWM board, otherwise the PWM circuit will not respond reliably to the signals. This is done by the transistor inverters.

The board was tested to see if it was fast enough for the task. Figure 27a shows that the DAC (and amplifier) have a large oscillation and a quite long settling time. This is fixed by the capacitor C_n in the circuit diagram. The resulting plot of voltage against time is shown in Figure 27b. The faster settling allows the parallel port to put data into the DAC at a maximum rate of about 500KB/s. Although this data rate is above the specified maximum for the standard parallel port, it is well within the specification of the ECP (extended capability port), which most PCs have. To put this in perspective, 500KB/s is fast enough to produce a low resolution raster scan display, as shown in Figure 28. This is easily fast enough to drive the PWM circuitry since the sample and hold ICs require the analogue value to be stable for at least $10\mu s$.

[4]This means telling the computer to output the sequence $\ldots 0, 1, 0 \ldots$ on that wire. Since the $\overline{C_0}$ control line uses inverse logic, the output goes $\ldots 5V, 0V, 5V \ldots$

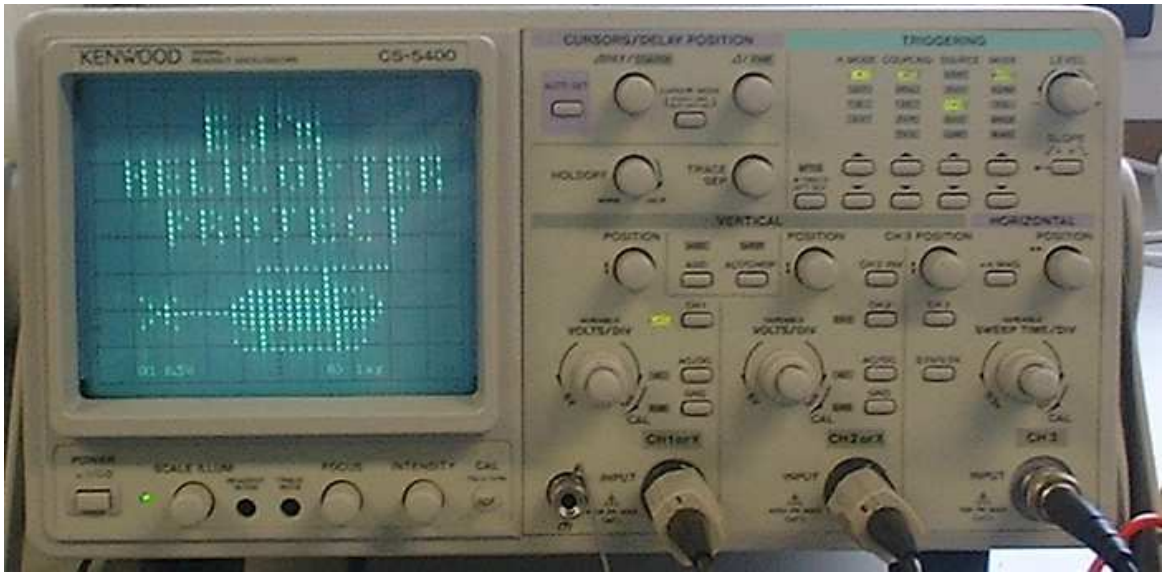


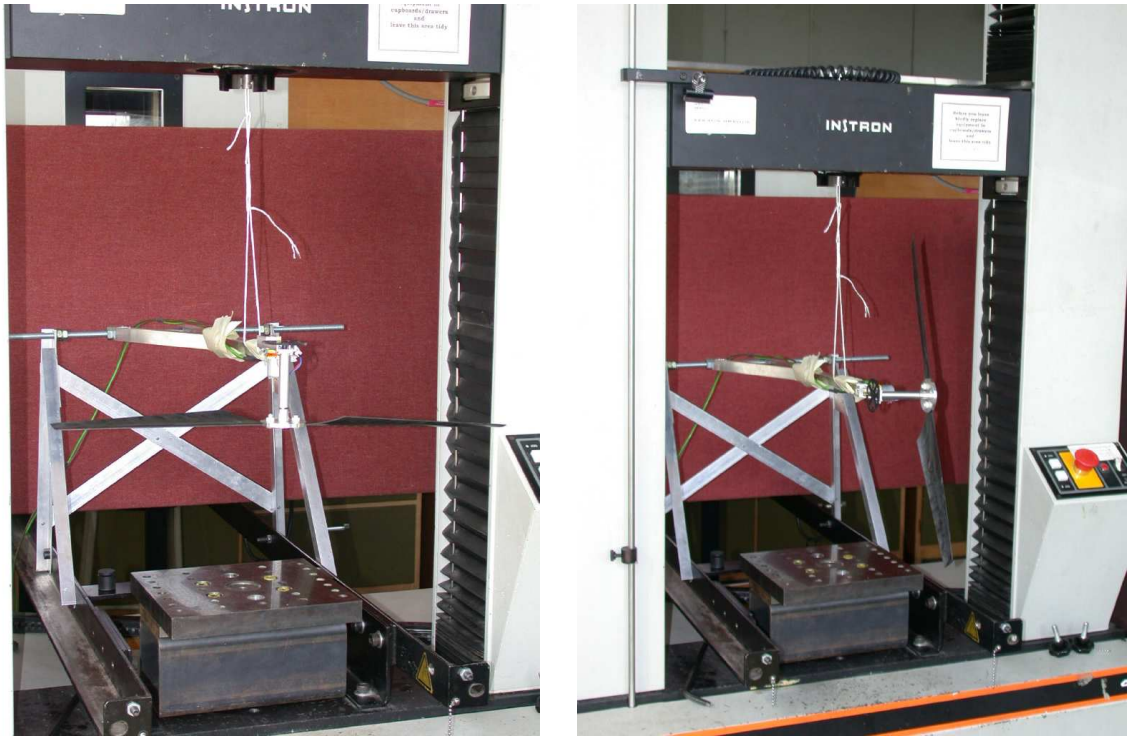
Figure 28: A low resolution raster scan display on an oscilloscope.

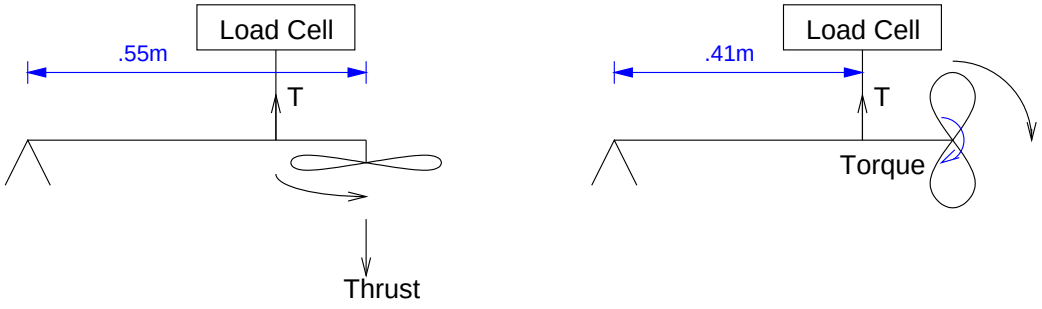
5 Rotor Calibration

The motor/rotor system is not linear. To make the system easier to control, the rotor thrust and torque as a function of input must be found. This makes it possible to derive another function which produces the correct output to get thrust as a linear function of input. The system has several nonlinear parts. The motor speed is a nonlinear function of the input voltage which is the parameter set directly. As well as varying from motor to motor, it will vary from rotor to rotor. An imbalance in the mass of the rotors and rotor head will cause the system to vibrate which dissipates power that would otherwise turn the rotors. There is little point in just measuring the motor speed though, since the thrust produced is a nonlinear function of the rotor speed. As well as producing thrust, the motors produce torque. This needs to be controlled, since an imbalance in torques produced by the motors will cause the whole helicopter to rotate, twisting up the umbilical cable. Since one set of rotors is angled backwards so as to produce thrust when rotated the opposite way (see Section 2), they will be less efficient since they have not been designed to operate in this way. As a result, they will produce less lift for a given power input.

The force produced was measured by attaching the rotor to a bar—pivoted at one end—and measuring the force downwards produced by means of a load cell connected to the bar. Attaching the rotor so it produces force downwards allows the lift to be measured. Attaching it sideways allows the torque to be measured. Pictures and diagrams are shown in Figure 29.

The PWM circuit was stepped from 0 (minimum) to 255 (maximum) in steps of 5, each step occurring every three seconds. At the end of the run, the speed was dropped to zero for six seconds, raised to 255 in a single step and held for 6 seconds, then reduced to zero in single step (see Figure 30). The





Thrust = 0.745 x T

Torque = .41 x T

Figure 29: Rotor calibration rig.

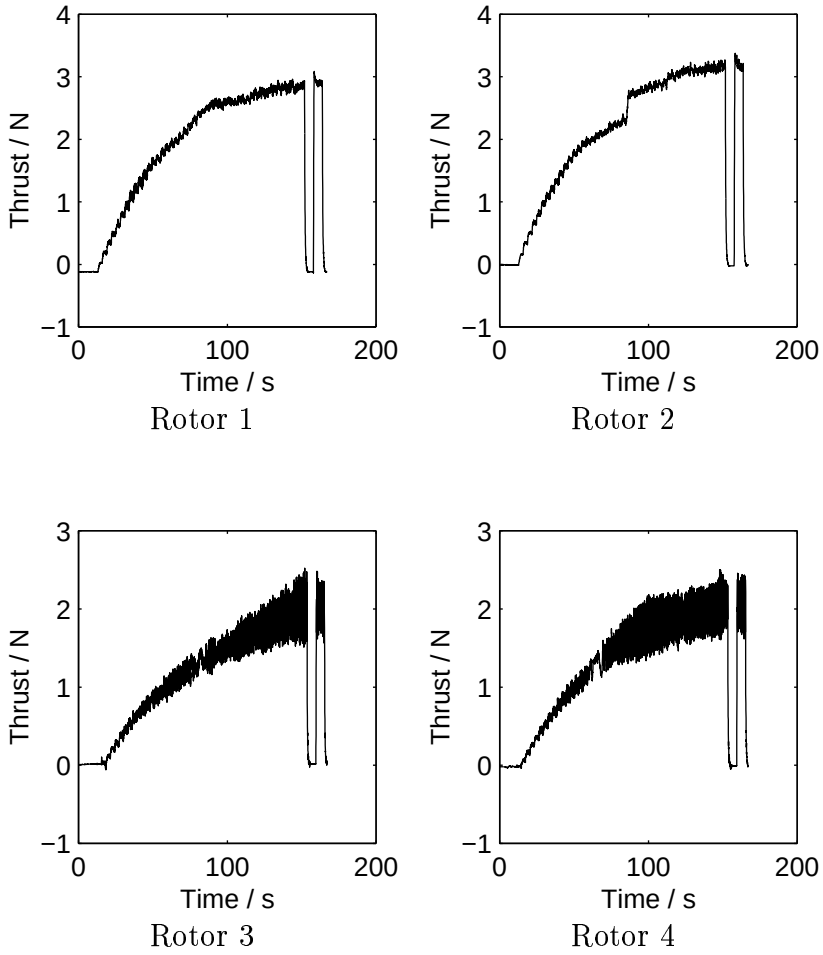


Figure 30: Thrust data for all of the rotors.

second part of the run is used to estimate the time constant of the system. Load cell measurements were taken continuously throughout the experiments at the rate of 20 per second.

5.1 Rotor thrust calibration

A graph of the rotor thrusts against time is shown in Figure 30. The noise is caused by a slight imbalance in the rotors, which makes the assembly wobble. The wobble on the reversed rotor blades (rotors 3 and 4) is much more severe since the rotor heads are less precisely machined than the rotor heads for the normal rotors. This leads to an imbalance. The noise from the wobble is periodic and high frequency. At times, the aliasing of this noise produces glitches large enough to affect the data after processing (see Figure 31). This is discussed further in Section 5.3 where more severe problems with aliasing are dealt with. As can be seen in Figure 30 the data is noisy (especially for rotors 2 and 3) and as a result must be filtered to remove the noise before it is useful. There are 52 points of interest (the output set to $0, 5, \ldots, 255$) and the thrust at these points must be extracted after filtering. These are

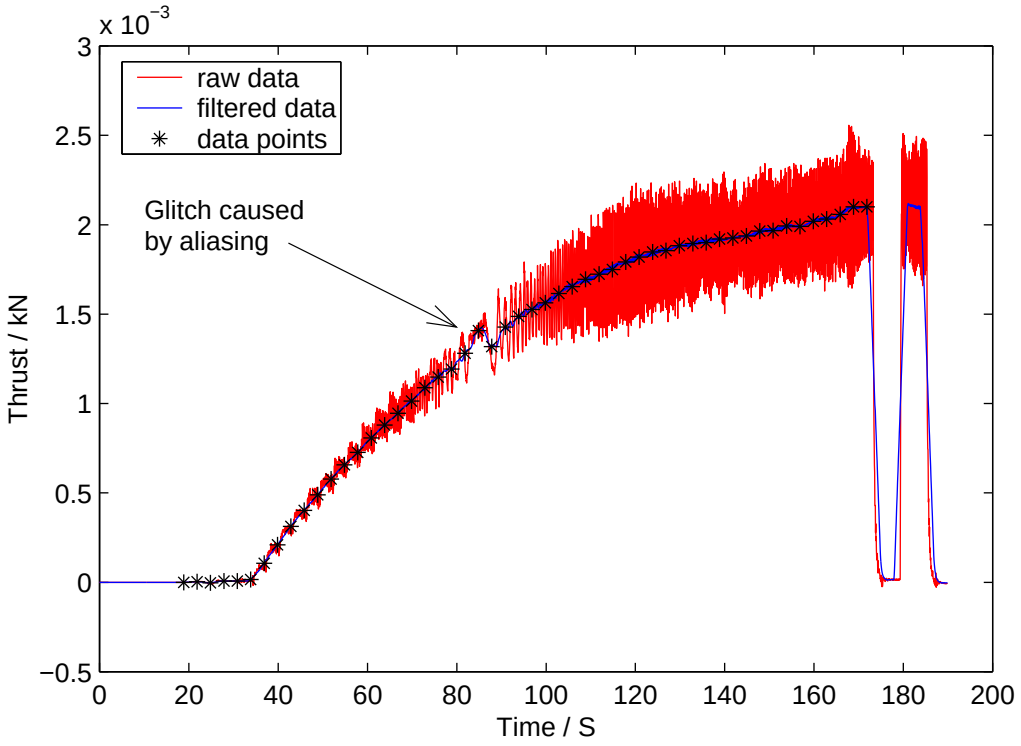


Figure 31: Thrust for rotor 4 with processing.

taken to be 1.5 seconds after a change is sent to the PWM circuit. Figure 31 shows the filtered data and extracted points. The points are extracted by working out when the test stops by looking at the gradient of the data. A slightly different heuristic was used for each rotor test since the data is too complicated to allow this kind of test to be implemented trivially. By taking filtered data at the points of interest, a graph of the rotor thrust against input can be plotted. This is shown in Figure 32.

As can be seen, the thrust produced by rotors 3 and 4 is significantly lower than for rotors 1 and 2. Rotors 3 and 4 are the ones with reversed blades. They were run at a lower voltage (9V as opposed to 12V) as otherwise they went from being still to drawing the maximum current permitted by the current limiter in a very short range of inputs. Reducing the input voltage therefore increases the precision.

To get calibrated data sets, the function of thrust against demand needs to be inverted. This is done by taking thrusts from 0 to 4.1N, every 0.05N and finding the input value corresponding to this thrust value. If there is no exact match, then linear interpolation is used to find the required input. The full tables of this data are given in appendix D. Plots of these are shown in Figure 33. The data is given in appendix D.1.

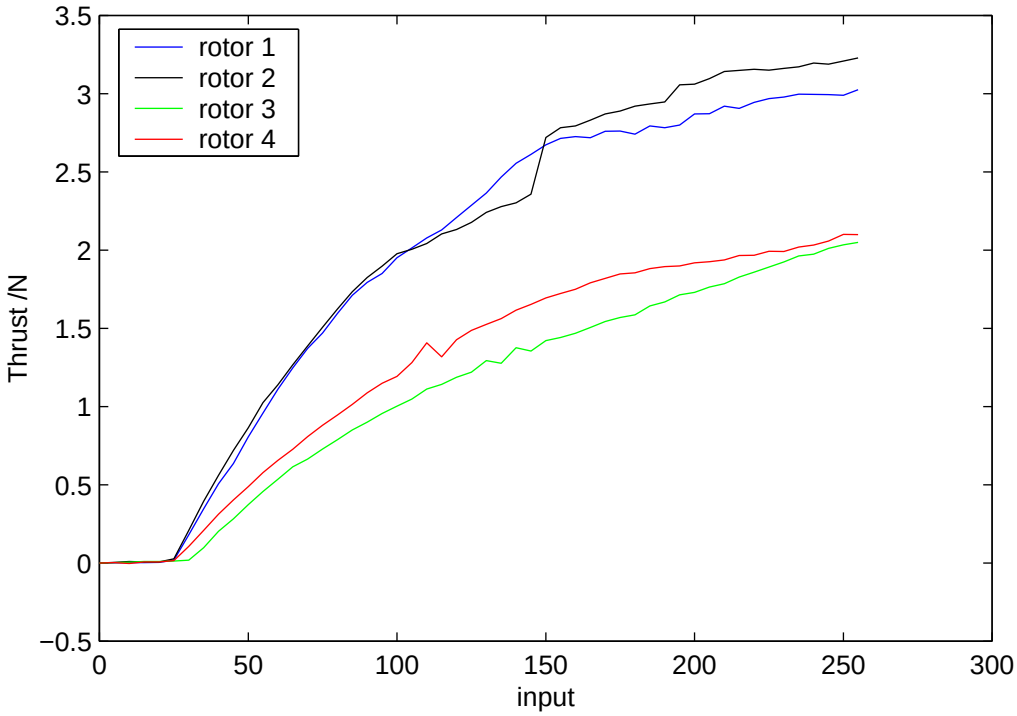


Figure 32: Rotor thrust against input.

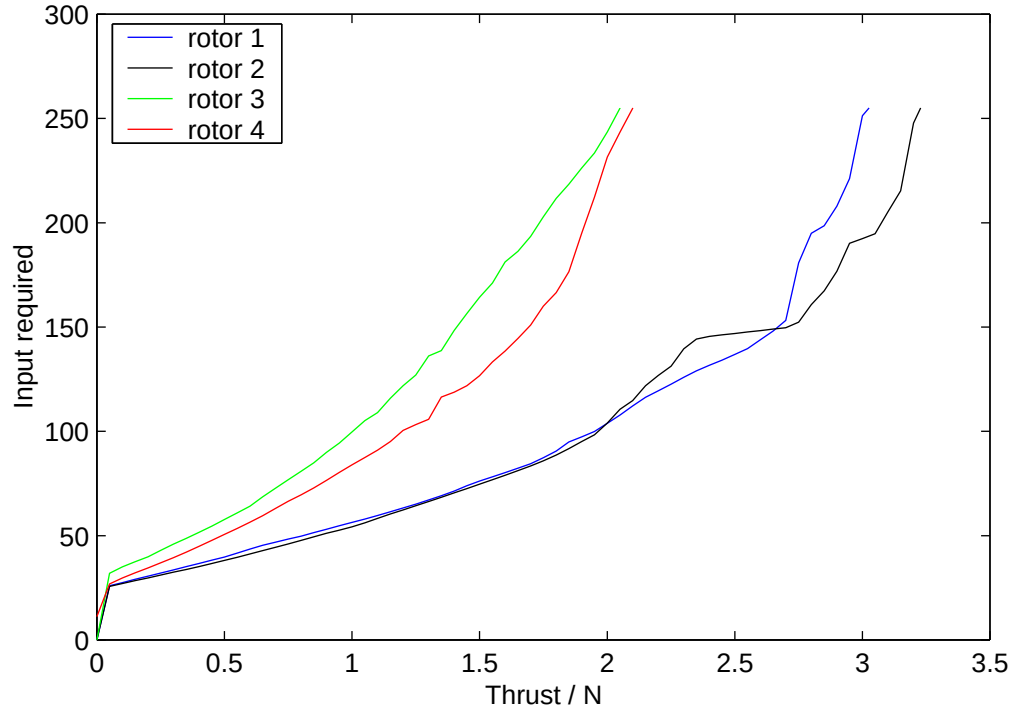


Figure 33: Required demand against request thrust.

5.2 Rotor Time Constant Measurement

The dynamic behaviour of the motor/rotor system can be approximated as a linear 1st order system. This assertion will be demonstrated later. The system has been given a step up to respond to (as shown in Figure 31). As a result, the system should respond thus:

$$\xi = \alpha\left(1 - e^{-\tau_1 t + \gamma_1}\right)$$

where τ_1 is the time constant of the system and γ_1 is the offset in time where the step occurs. ξ is the thrust. α is a scaling factor put in since the thrust does not vary from zero to one. α can be found easily since:

$$\lim_{t\to\infty} \xi = \alpha.$$

The step was held for six seconds since this is very much longer than the time constant of the system. By taking logarithms, the other unknowns become the coefficients of a 1st order polynomial:

$$T = -\tau_1 t + \gamma_1$$

where:

$$T = \ln\left(1 - \frac{\xi}{\alpha}\right).$$

Similarly, with a first order approximation, the system will respond to a step from maximum to zero thus:

$$\xi = \beta e^{-\tau_2 t + \gamma_2}$$

Again, this can be made in to a 1st order polynomial:

$$T = -\tau_2 t + k$$

where

$$\begin{aligned} T &= \ln(\xi), \\ k &= \ln(\beta) + \gamma_2. \end{aligned}$$

The system was fed with a square pulse (a step up followed by a step down) of sufficient width to ensure that the first step had settled before the second step occurred. By taking a selection of points, the coefficients can be calculated to a good degree of accuracy using a least squares approximation. The starting position and all subsequent points of interest on the pulse response can be determined using similar heuristics to those used in finding the end of the thrust calibration curve. The time constants calculated using this method are given in Table 3.

In some ways the results are as expected, in others not. Rotors 3 and 4 have a higher τ_2 than rotors one and two. This is expected since they lose energy through drag more quickly and lose far more energy through vibration. This makes them stop rapidly. There is also much more variation between τ_1

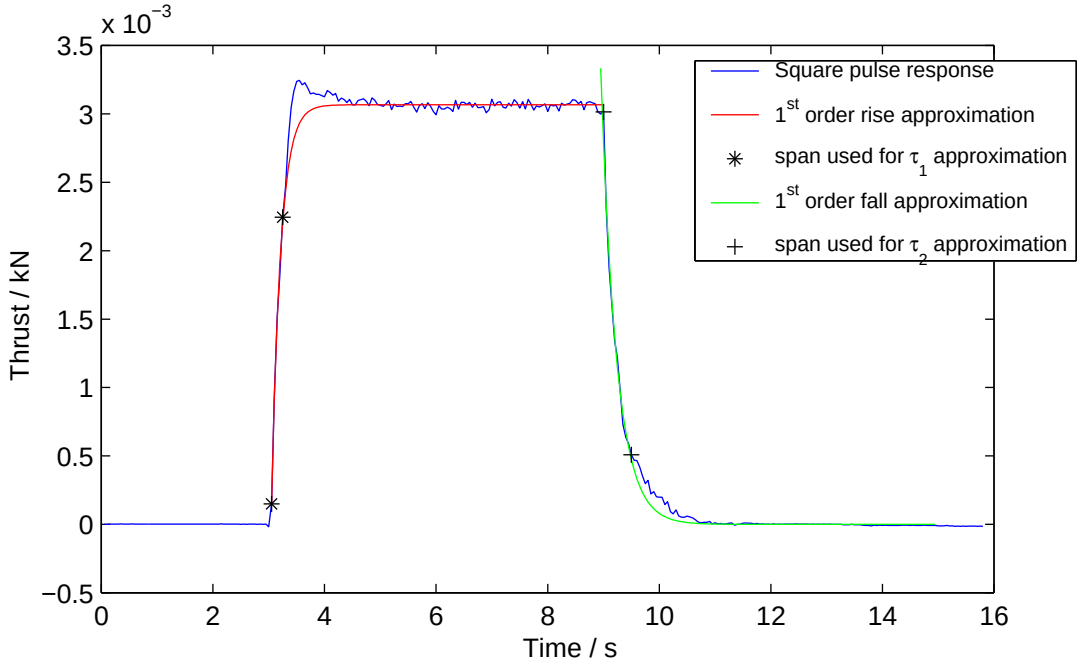
Rotor	τ_1 / s^{-1}	τ_2 / s^{-1}
1	6.14	5.70
2	5.73	5.60
3	6.79	6.46
4	11.94	6.28

Table 3: Rotor time constants.

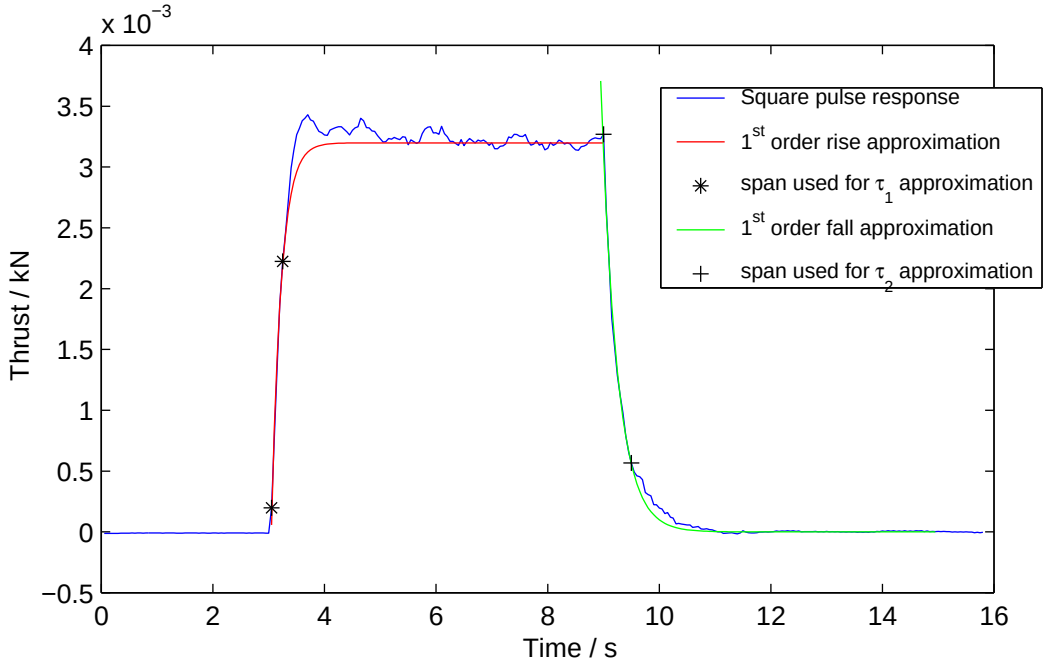
than τ_2. This is also expected since τ_2 does not depend much on the motor properties (the motors offer little resistance when they are open circuit), but τ_1 depends very strongly on the motor properties. As can be seen in Section 5.1, the motor properties vary significantly. The first time constant for rotor 4 is surprisingly high. Graphs of the pulse response superimposed with the first order approximations are shown in Figure 34. As can be seen in the graphs, the responses for the step upwards all fit the early part of the step very well. This suggests that the high τ_1 is not an artifact of measurement, but is in fact due to the motor. These graphs also show how good the approximation to a first order linear system is. The first order approximation works reasonable well for the step up, towards the beginning of the step. Rotors 1 and 2 break away from the linear response earlier than rotors 3 and 4. The first order approximation for the step down is rather better. The overshoot on the step up is not second order behaviour since the initial gradient is not consistent with second order behaviour.

Rotors 3 and 4 seem to be faster than predicted, getting to the final value quicker than the first order approximation suggests. Although rotors 1 and 2 reach the final value quickly, they both overshoot and take a relatively long time to settle down.

The first order linear approximation seems to fit the result from the step down more closely than for the step up. There is, however, a consistent underestimate for the initial gradient in all cases. This makes the system slightly faster than the approximation predicts. As can be seen, there is a small error, where the thrust measured before the step starts is slightly higher than after the step response finishes. This has been taken in to account by shifting the data relevant to the step down so that it finishes at zero, before performing the calculations.

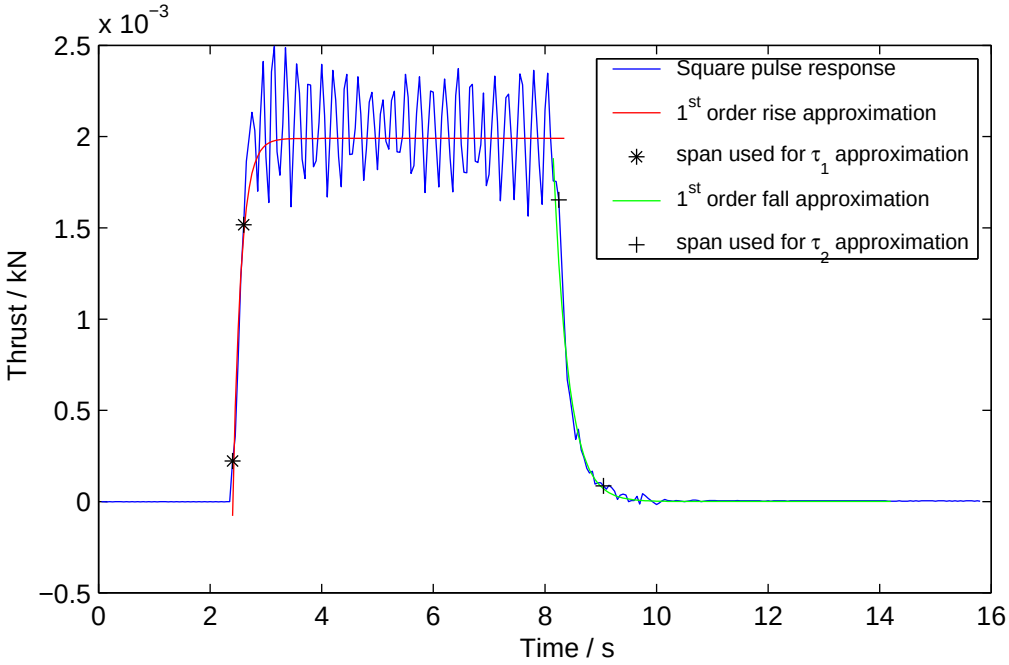


Rotor 1

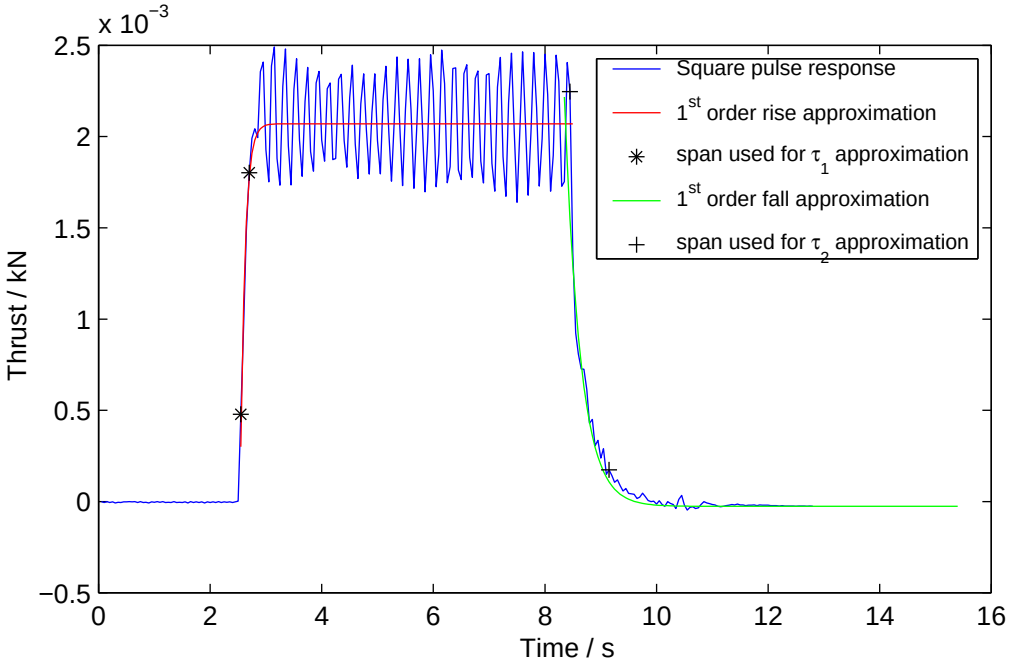


Rotor 2

Figure 34: First order approximation to rotor responses.



Rotor 3



Rotor 4

Figure 34: First order approximation to rotor responses.

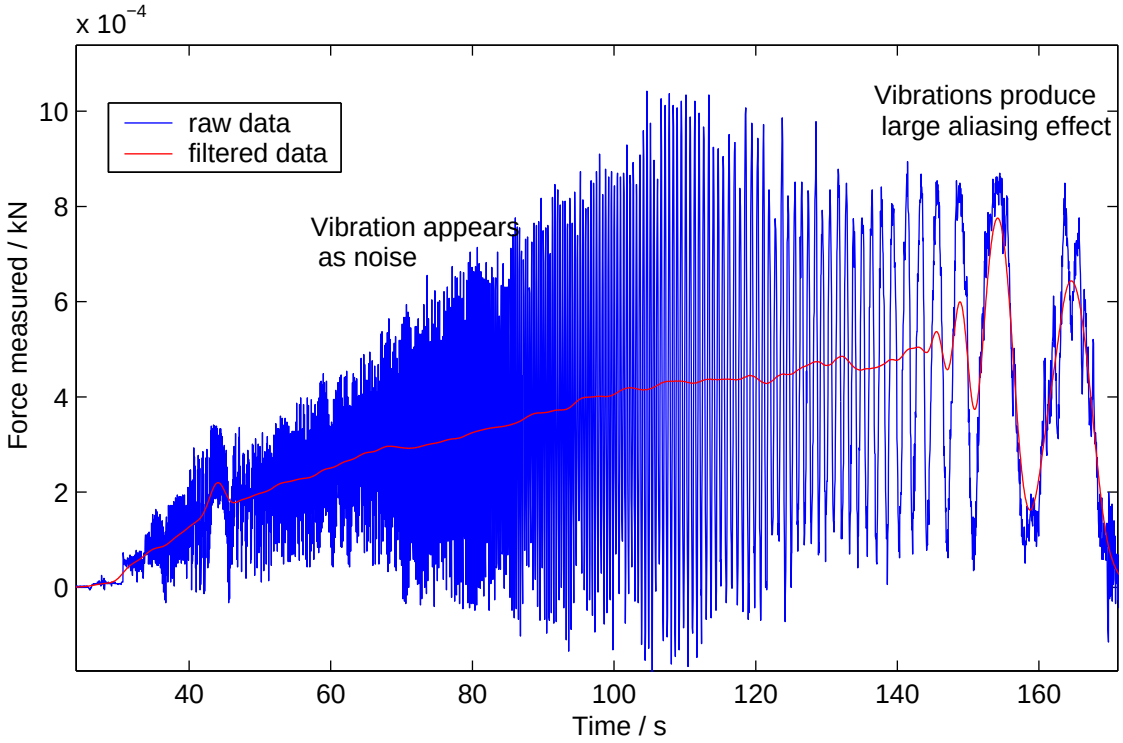


Figure 35: Vibrations produce aliasing.

5.3 Rotor Torque Calibration

Determining the rotor torque is a slightly more difficult problem than determining the lift. First, the force produced by the torque is smaller. Secondly, the force vector being measured is in the plane of the vibrations, so the vibrations have a much larger effect on the final reading; the magnitude of the noise caused by the vibrations is over double the magnitude of the force being measured. This means that any detrimental effect the vibrations have on the signal will be far greater. The worst effect by far is aliasing, since this can produces low frequency components in to the signal which cannot easily be filtered out. The worst case occured with rotor 1 (see Figure 35). Towards the end of the test, aliasing of the noise from the vibrations causes large low frequency components to be introduced which swamp the signal. There is no easy way around this since the system used is not designed for sampling at the rates required to avoid the effects of aliasing. This problem most seriously presents itself for rotor 1. To attempt to get sensible values from the data, two procedures were applied to the data. Firstly, it was filtered much more heavily than for the other rotors, then a rule was applied to each data point in turn to ensure that the data increased monotonically. While the resultant data is not a completely accurate representation of the real case, it is more accurate than the heavily aliased data. Graphs of torque produced against input are shown in Figure 36. These functions have been inverted and graphed in Figure 37 and tabulated in appendix D.2.

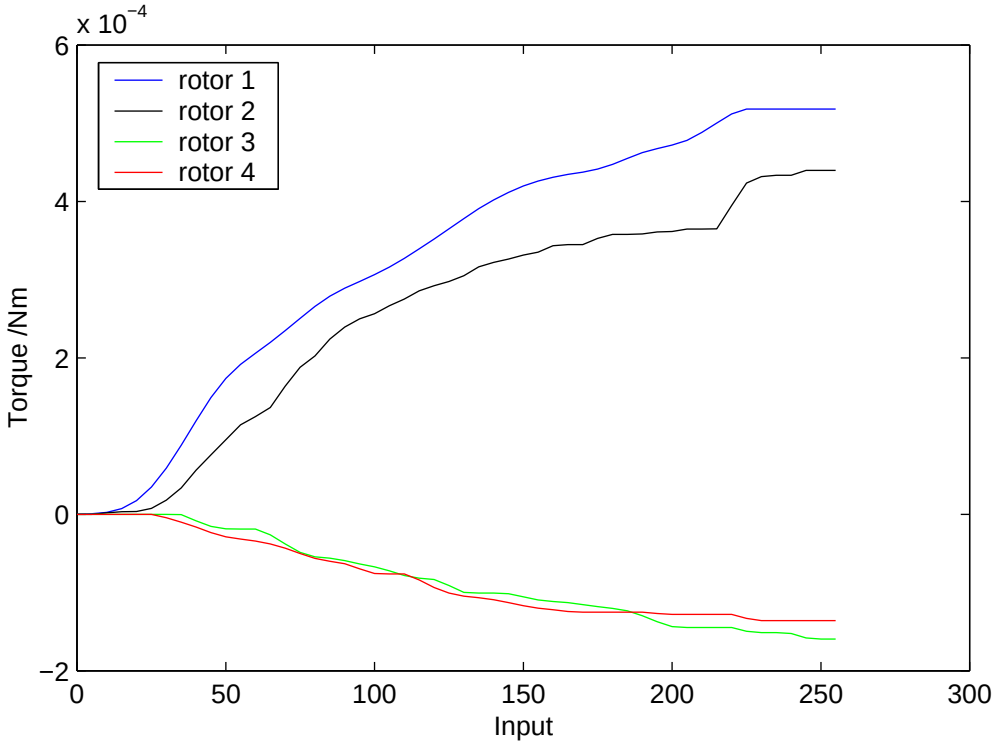


Figure 36: Torque against input.

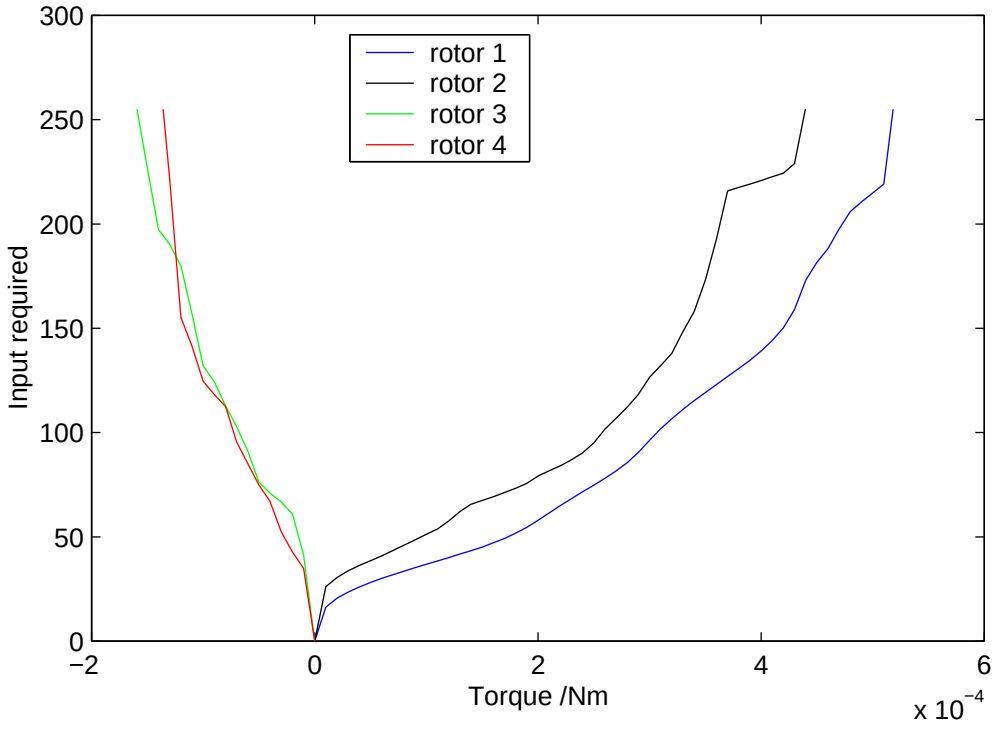


Figure 37: Input required against torque.

6 Conclusion

The infrastructure of a helicopter with the potential to be free flying has been built. This includes:

- A sensor suite capable of measuring the two rotations of interest in the helicopter to a high accuracy. The suite contains an optical sensor for measuring the yaw to accuracy of 1°. The suite contains magnetic field sensors capable of resolving the other required angles to a high accuracy, such that with the required accuracy of 1° only four erroneous readings will occur per hour.

- A communication link capable of transferring all of the sensor data to a computer with an RS-232 port using a packet based protocol which can recover if part of the data stream becomes corrupted.

- A power controller board capable of interfacing with any standard logic level source (such as a PC parallel port or microcontroller). The power controller includes separate current limitation for each channel, which prevents the motors becoming burned out under a high load.

- A physical helicopter able to lift the sensor and communication circuitry. This includes a cheap, easily fabricated frame which reduces the financial and time costs of breakages during testing.

- Characterization of the physical helicopter, involving functions relating the input required for a given lift/torque and time constants for the rotors. This characterization removes some of the nonlinearity of the system which makes controlling the helicopter easier.

There are a number of interesting things which could be done to take the project further. A control program needs to be written which takes in the readings, calculates the pitch and roll and feeds these in to a PID controller to control the motor inputs. At a later stage, the control subsystem can be moved on board the helicopter by using a more sophisticated microcontroller.

To improve the performance, various things can be done. The low pass filter on the yaw sensor could be improved to allow a higher sampling rate to be used. Adding a second IR transmitter, and a second filter to the yaw sensor would allow two angles to be measured. Using triangulation, the position of the helicopter could be measured. This would enable the helicopter to be able to hover without drifting.

Finally, it would be useful to add height sensors to allow the helicopter to take off and land in a controlled manner. With these additions, the helicopter could be positioned at any point in the 3D space covered by its sensors.

7 Acknowledgements

I would first like to thank my supervisor Prof B. Kouvaritakis for his invaluable help and enthusiasm. I would also like to thank John Hastings and John Barton for advice about manufacturing the helicopter, Roger Stone for help with the Instron testing machine, Mike Stockford for helping me in the electronics lab and Anthony Cooper for advice on model helicopters. I would also like to thank everyone else who gave me useful suggestions.

References

[1] *1- and 2-Axis Magnetic Sensors (900248 Rev. B).* Honeywell International Inc. (`www.honeywell.com`)

[2] *TSL260, TSL261, TSL262 Light to Voltage Optical Sensors.* TAOC Inc. (`www.taosinc.com`)

[3] *Helicopter instrumentation and control.* Marko Bacic (May, 2001)

[4] *Numerical recipies in C (second edition)* William H. Press, Saul A. Teukolsky, William T. Vetterling Brian P. Flannery, Cambridge University press (1999)

[5] *OpenGL® Programming Guide: The Official Guide to Learning OpenGL, Version 1.2* Mason Woo, Jackie Neider, Tom Davis, Dave Shreiner, OpenGL Architecture Review Board, Addison-Wesley Publishing Company (1994)

[6] *Advanced Engineering Mathematics, 7$^{\text{th}}$ edition* Erwin Kreyzig, Wiley (1993)

[7] *ICM7555, ICM7556 General Purpose Timers (2876.4)* Intersil (`www.intersil.com`)

[8] *LM139/LM239/LM339/LM2901/LM3302 Low Power Low Offset Voltage Quad Comparator* National Semiconductor Inc. (`www.national.com`)

[9] *LM111/LM211/LM311 Voltage Comparator* National Semiconductor Inc. (`www.national.com`)

[10] *PIC16F87X Data Sheet* Microchip Technology Inc. (`www.microchip.com`)

[11] *Interfacing the Standard Parallel Port* Craig Peacock, `www.beyondlogic.org/spp/parallel.htm`

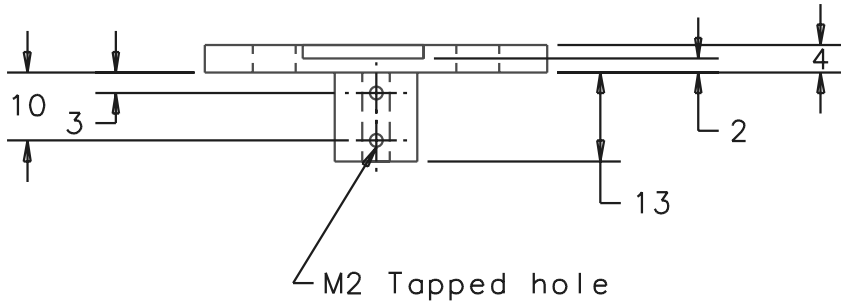
[12] *Interfacing the Serial Port Parts 1–2* Craig Peacock, `www.beyondlogic.org/serial/serial.htm`

[13] *Interfacing the Serial Port Parts 3–4* Craig Peacock, `www.beyondlogic.org/serial/serial1.htm`

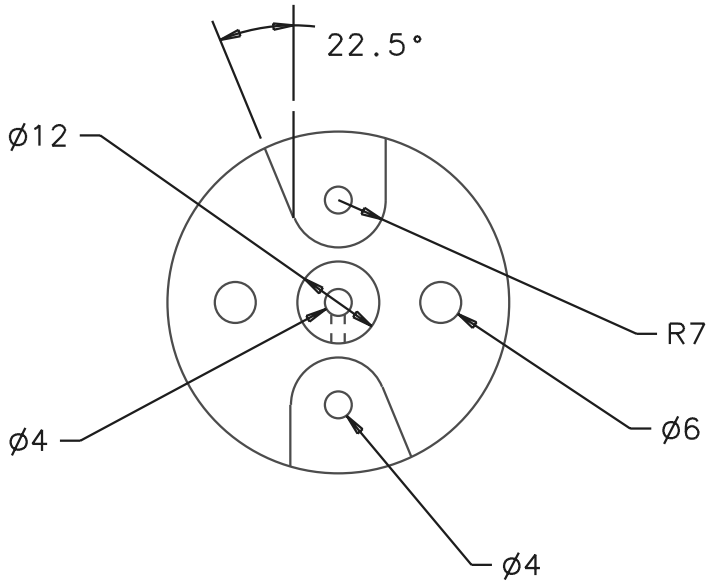
A Engineering drawings for helicopter parts

Name: E. Rosten
Project: 02/4YP/BK/110
Part: Rotor head

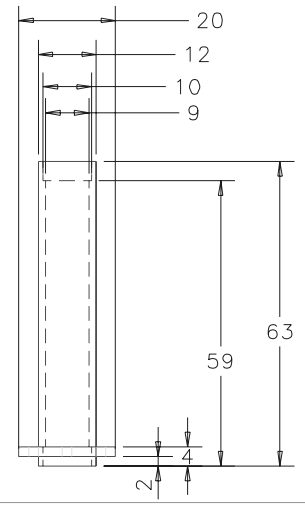
Material: Dural
Number: 5 off
Dimension: mm



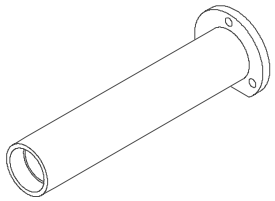
3-TOP



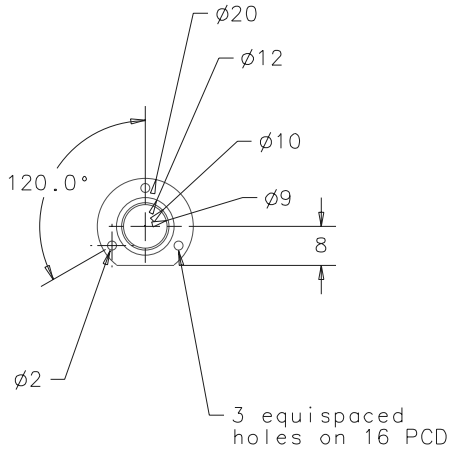
2-FRONT



4-TOP



5-ISOMETRIC

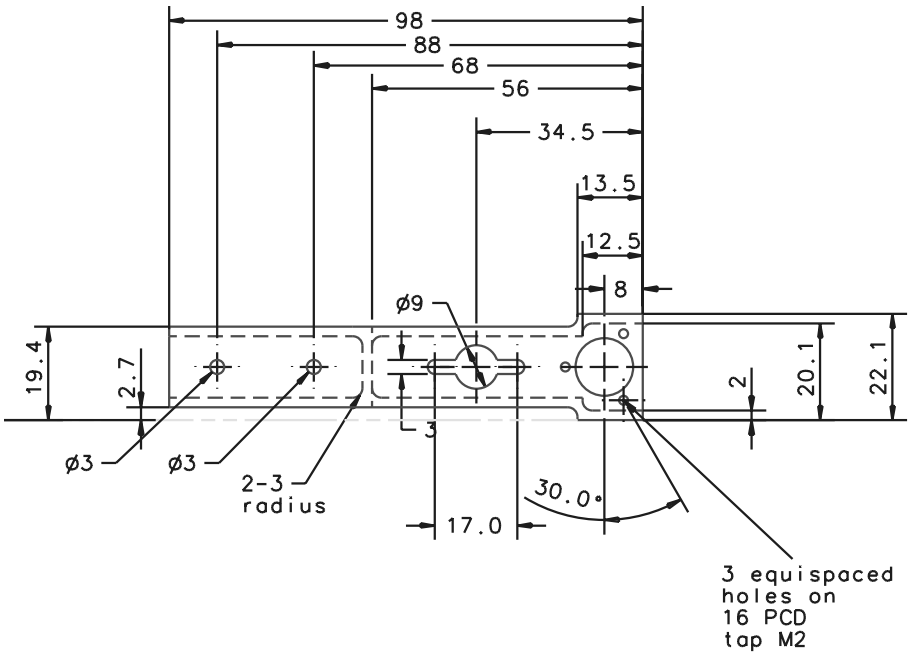


2-FRONT

Name: E. Rosten
Project: 02/4YP/BK/110
Part: Rotor Mast

Material: Aluminium
Number: 5 off
Dimensions: mm

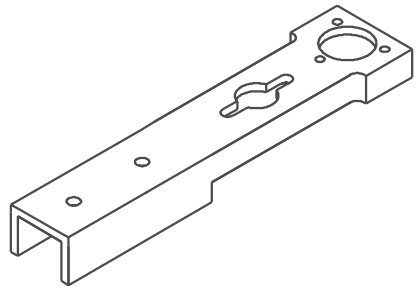
3-RIGHT



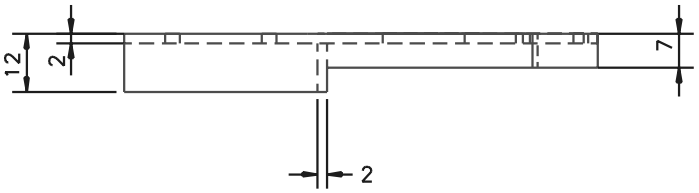
1-TOP

Name: E. Rosten
Project: 02/4YP/BK/110
Part: Main rotor assembly

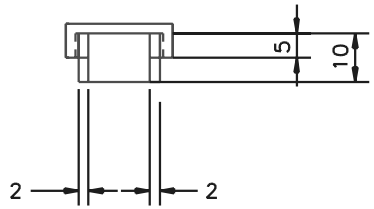
Material: Aluminium
Number: 8 off
Dimensions: mm
Radii: 2 - 3



5-ISOMETRIC



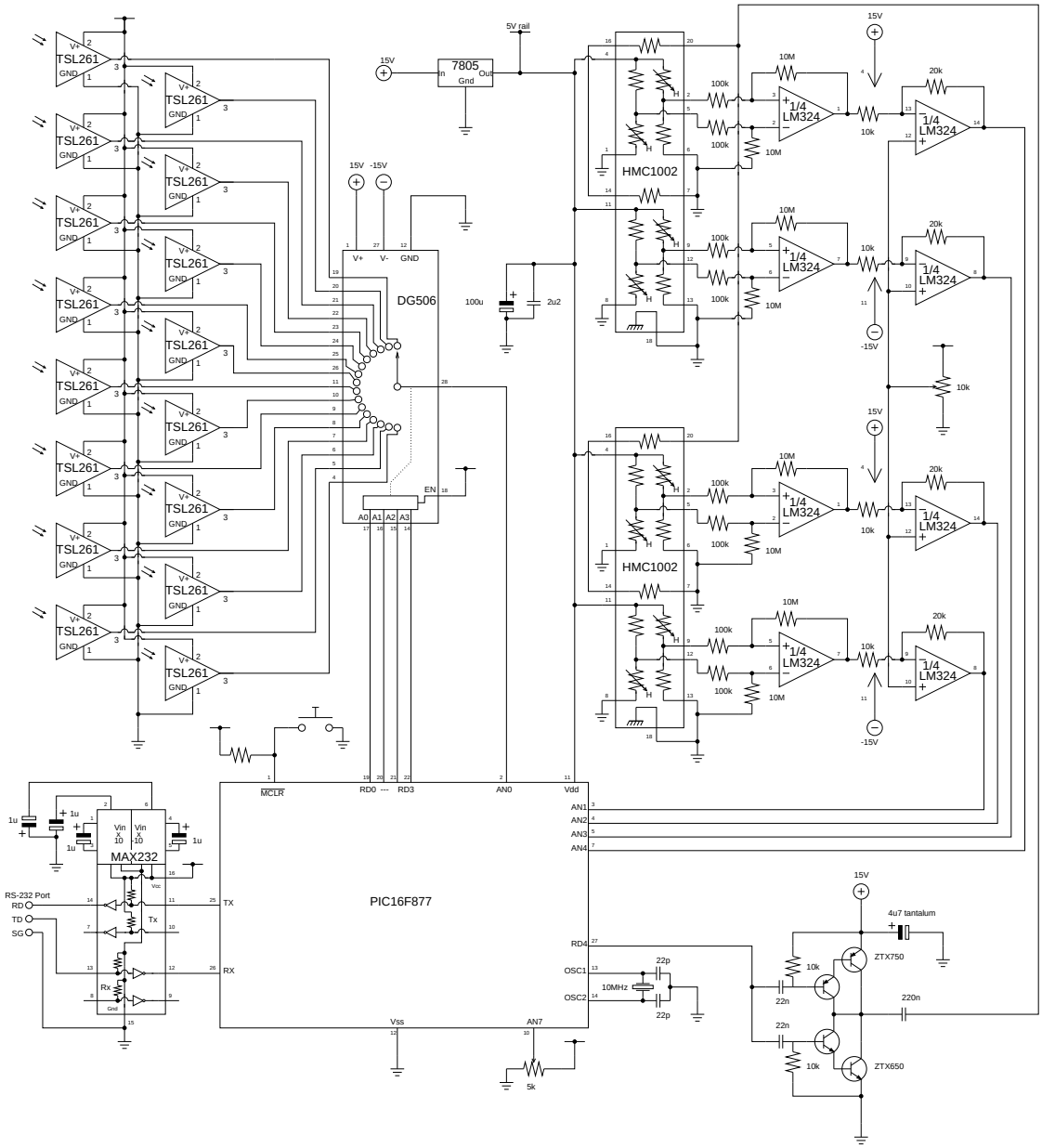
2-FRONT



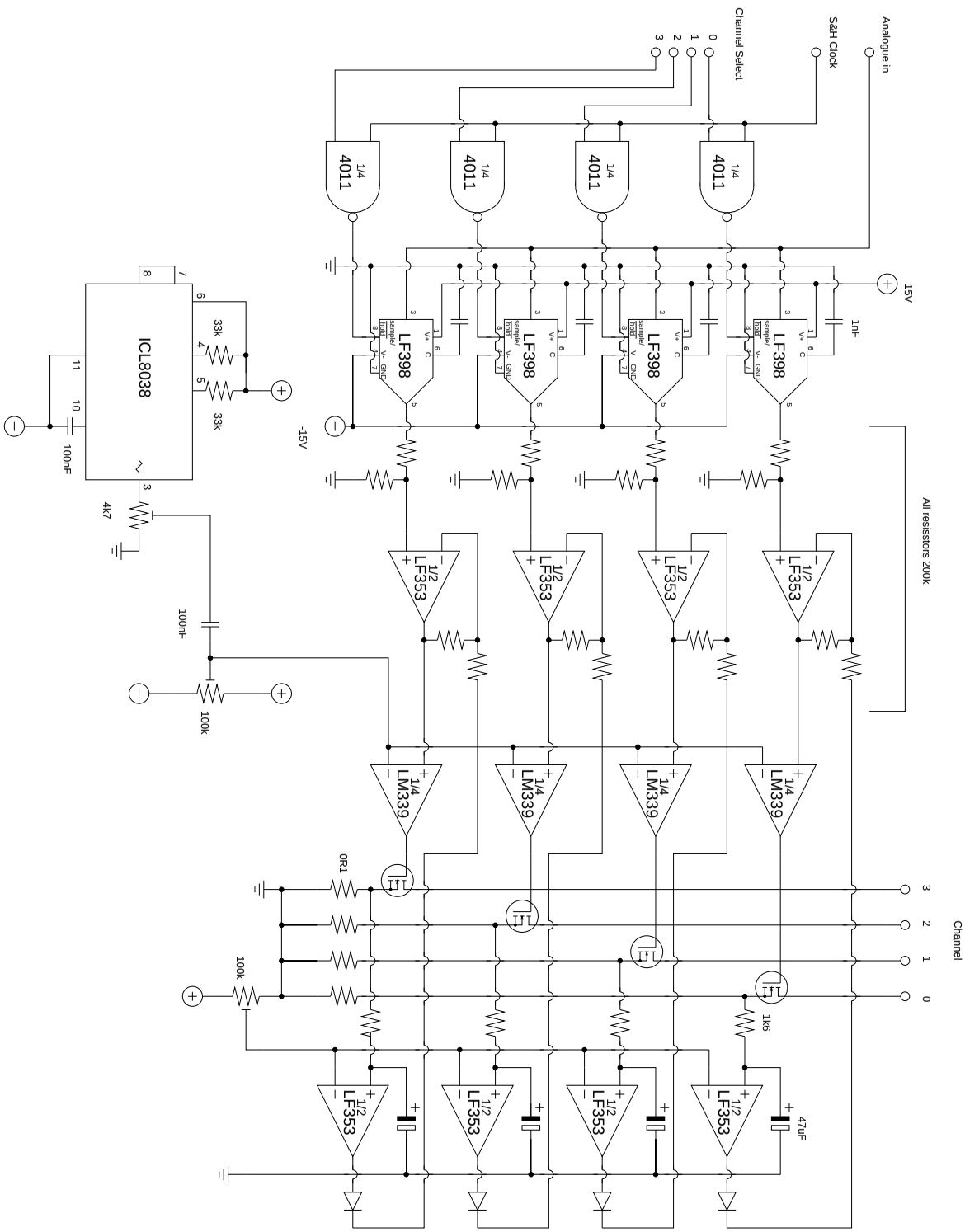
3-RIGHT

B Circuitry

B.1 Full helicopter sensor circuit



B.2 Full PWM circuitry



MOS-FETs	PHP3055	Not all IC power supplies shown.
Diodes	1N4001	Motors connected from +V to channel 0–3.

C Code

C.1 PIC code for the sensor circuit

```
;*******************************************************************************
;
; This program is designed to read data from the theta_z sensor. The sensor
; consists of 16 IR sensors connected to a multiplexer. The multiplexer is
; controlled by PORTD<0:3>. The multiplexed output is connected through a
; filter and rectifier in to the analogue port, AN0.
;
; After reading the 2 bytes from the ADC, the PIC sends FF, FF, ADCH, ADCL down
; the serial port, for collection by different program.
;
;*******************************************************************************
    list    p=16f877
    #include p16f877.inc

delay_counter   EQU 0x20
delay_counter1  EQU 0x24
tmp     EQU 0x21
tmp1        EQU 0x22
int_data    EQU 0x23
tmp_status  EQU 0x70
ir_sensor   EQU 0x71
tmp3        EQU 0x72
tmp4        EQU 0x73

    goto    start           ;Reset vector
    nop
    nop
    nop
    nop                     ;Interrupt vector
    nop

start

    bcf     STATUS, RP0
    bcf     STATUS, RP1

;****************************** Serial Port Initialisation ********************
    bsf     STATUS, RP0     ;Select Bank 1

    movlw   0x0A            ;Set baud rate to 57600
    movwf   SPBRG

    bcf TXSTA, TX9          ;8 bit transmission
    bsf TXSTA, TXEN         ;Enable transmission
    bcf TXSTA, SYNC         ;Asynchronus transmission
```

```
    bsf TXSTA, BRGH         ;High speed baud rate

    bcf     PIE1, TXIE      ;Disable interrupts

    bcf TRISC, 6            ;Enable the RX pin for output

    bcf     STATUS, RP0     ;Go to bank 0
    bsf RCSTA, SPEN         ;Enable the serial port

;*********************************************************************************

;*******************************ADC initialisation*****************************
    banksel ADCON1          ;Bank 1
    movlw   0x00
    movwf   ADCON1          ;Bits <0:3>0000: All 8 channels as ADC
                            ;inputs,  Vdd, Vss as voltage refs.
                            ;Bits <7>0 Left justify data.

    banksel ADCON0
    bsf ADCON0, ADCS1       ;ADCON<7:6>10, use ADC clock as fosc/32
    bcf ADCON0, ADCS0       ;This is required for 10MHz clock.
    bsf ADCON0, ADON        ;Enable ADC module


    banksel PIE1
    bcf PIE1, ADIE          ;Disable A/D interrupts.
;*********************************************************************************

;*****************************Port D initialisation ***************************

    banksel TRISD
    clrf    TRISD
    banksel PORTD
    movlw   0
    movwf   PORTD
;*********************************************************************************

;*******************************Misc**********************************************
    movlw   0
    movwf   ir_sensor
;*********************************************************************************

main:

    banksel PORTD           ;Select the IR sensor, before the pause
    movfw   ir_sensor       ;so the 1st order filter has time to
    movwf   PORTD           ;Settle.
```

```
    call    pause
    movlw   0xff            ;Send 0xff, 0xff
    call    send_w          ;The 2 start bytes for the transmission
    call    send_w          ;protocol.


    movlw   7               ;Send the POT (for debug purposes)
    call    send_adc_w

    movlw   1               ;Send all the magnetic field sensors
    call    send_adc_w
    movlw   2
    call    send_adc_w
    movlw   3
    call    send_adc_w
    movlw   4
    call    send_adc_w

    movlw   0               ;Send the IR sensor number padded out
    call    send_w          ;to 16 bits.
    movfw   ir_sensor
    call    send_w
                            ;Transmitting 14 bytes takes about 0.0025 S
                            ;We need a delay of 0.01 S for the LPF to
                            ;Settle. Pause for 0.0075 S
    movlw   0
    call    send_adc_w


    incf    ir_sensor, 1
    btfss   ir_sensor, 4
    goto    main

    movlw   0
    movwf   ir_sensor

goto main


;*****************************************************************************
;
;Functions.
;

;*****************************************************************************
;
; send_w
;
```

```
; Transmits the contents of W down the serial port. The functino returns after
; the byte has been sent. The function leaves the PIC in bank 1.

send_w:                     ;This sends W down the serial port.
    banksel TXREG
    movwf   TXREG
    banksel TXSTA

poll_serial:
    btfss   TXSTA, TRMT     ;Bit test, skip if set
    goto    poll_serial

    return

;*****************************************************************************
;
; send_adc_w
;
; This reads a value from ADC #w
; timing is done by polling the ADC. The function returns when the bytes have
; been read and sent down the serial port

send_adc_w:

    banksel ADCON0

    andlw   0x07            ;Mask off W
    movwf   tmp3            ;Shift to the CHS2:0 position
    rlf tmp3, 1
    rlf tmp3, 1
    rlf tmp3, 1

    movfw   ADCON0
    andlw   b'11000111'     ;Remove the old ADC number
    iorwf   tmp3, 0         ;Put in the new ADC number
    movwf   ADCON0          ;Commit data to the ADC controller


        ;Pause for a bit to allow Ch to charge. Each instruction takes
        ;4/Fosc secs to execute, except "true" conditional tests and
        ;gotos which take 8/Fosc. Fosc=10e6. Delay 20us to be safe.

    movlw   d'12'           ; 0.4 uS
    movwf   tmp3            ; 0.4 uS

adc_pause_loop:
    decf    tmp3            ; 0.4 uS
    btfss   STATUS, Z       ; 0.4 uS (normally), 0.8 on last loop
```

```
    goto    adc_pause_loop  ; 0.8 uS (except on last loop)

        ;Time = 0.8 + n*(0.4+0.4+0.8) - 0.8 + 0.4
        ;     = 0.4 + n*(1.6) uS
        ;     = 20 uS
        ; n = 12.25, use n=12

    bsf ADCON0, GO          ;Start conversion


poll_ADC:

    btfsc   ADCON0, GO      ;Bit test f, skip (if) set
    goto    poll_ADC


    banksel ADRESH
    movfw   ADRESH
    call    send_w
    movfw   ADRESL          ;Now we're in bank 1
    call    send_w

    return


;*********************************************************************************

pause                       ;.8 to get here

    bcf     STATUS, RP0     ;.4 uS

    movlw   d'11'           ;.4 uS
    movwf   delay_counter1  ;.4 uS

loop1
        ;*************************************************************


    clrf    delay_counter   ;.4
loop

    nop                     ;.4 uS
    nop                     ;... .8uS

    decf    delay_counter   ;.4 uS
    btfss   STATUS, Z       ;.4 uS  or .8uS (last time)
    goto    loop            ;.8 uS  or 0    (last time)
```

```
                ;Loop takes 2.4 uS * 256 = 614.4
                ;Correcting for the last iteration and
                ;pre step  T=614.4
    ;*************************************************************

decf    delay_counter1  ;.4 uS
btfss   STATUS, Z       ;.4 uS or .8 (last step)
goto    loop1           ;.8 uS or 0  (last step)

return                  ;.8 uS

    ;Total time is 2.0 + N*(614.4 + .4 + .4 + .8) - .4 + .8
    ; = 2.2 + N * 616.0 uS
    ;We need a 0.0075 S delay, or 7500 uS
    ;need N=12.17. Use N=12

end
```

C.2 Receiver code for the PC

C.2.1 reader.c

```
/* Basic helicopter data reader.

This program reads successive packets from COM1 and prints out the packet. As
an added bonus, it calculates the angle measured from the theta_z sensor and
prints that as well.

Press <Esc> to quit.

E. Rosten 2002

*/

#include <stdio.h>
#include "com.h"
#include "packet.h"


void main(void)
{

    int             c, maxpos;
    unsigned int    ch, max;
    void interrupt (*old_isr)();
    double          real_pos, qi_a, qi_b, y1, y2, y3;

    long            n=0;

    unsigned far int*   packet;

    unsigned int    ir_sensors[16] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

    printf("This is %s\n", __FILE__);
    printf("Packet size is %i words.\n", WORDS_PER_PACKET);


    sleep(1);
    enable_com1();

    packet = &c;

    while(packet)
        {
        if((packet = get_packet()) == 0) continue;
        clrscr();
```

```
        for(c=0; c< WORDS_PER_PACKET; c++)
            printf("packet[%i] = %u\n", c, packet[c]);

        ir_sensors[packet[5]] = packet[6];

        max = 0;

        for(c=0; c< 16; c++)
            {
            if(ir_sensors[c] > max)
                {
                max = ir_sensors[c];
                maxpos = c;
                }
            printf("%-5i%i\n", c, ir_sensors[c]);
            }

        printf("**** %i *%i* %i   ", (16+maxpos-1)%16, maxpos, (maxpos+1)%16);

        //Quadratic interpolation

        y1 = ir_sensors[(16+maxpos-1)%16];
        y2 = ir_sensors[maxpos];
        y3 = ir_sensors[(maxpos+1)%16];

        qi_a = (y3 - 2*y2 + y1) / 2;
        qi_b = (y3 - y1) / 2;

        real_pos = -qi_b / (2 * qi_a) + maxpos;

        if(real_pos < 0) real_pos = 16 + real_pos;

        printf("%f degrees  ", real_pos * 360 / 16);
        printf("%i\n", ++n);
        }
    disable_com1();
}
```

C.2.2 packet.h

```
#ifndef INC_PACKET_H

#define INC_PACKET_H

#define WORDS_PER_PACKET 7

unsigned far int* get_packet();

#endif
```

C.2.3 packet.c

```
/* Com port reader packet input interface

This defines the packet primitive get_packtet(). And, yes, it is really
primitive. Read the packet protocol to see why.

E. Rosten 2002
*/

#include "com.h"
#include "packet.h"

/* Protocol definition     */

/* The protocol is a packet based protocol. The packet consists of the header
   bytes 0xff, 0xff followed by a number of 10 bit integers packed in to 16
   bit words with the bottom 6 bits set to zero.

   The words are transmitted in big endion format.

   Becasue the integers are only 10 bits, the sequence 0xff,0xff can only be
   found in the header.

   The number of 16 bit words in the packet is defined by WORDS_PER_PACKET
*/

unsigned int input_packet[WORDS_PER_PACKET];

unsigned far int * get_packet()
{
    int             ffs = 0;  //A count of the consecutive 0xff received
    int             word_num;
    int    inp;

    while(ffs != 2)         //Wait for the starting signal
        {
        inp = get_byte();

        if(inp == -1) return 0;

        if(inp == 0xff)
            ffs++;
        else
            ffs = 0;
        }

//The header has been received, so read the packet data
    for(word_num = 0; word_num < WORDS_PER_PACKET; word_num++)
```

```
        {
        //Get high byte
        inp = get_byte();

        if(inp ==  -1) return 0;
        input_packet[word_num] = inp << 8;

        //Get Low byte
        inp = get_byte();
        if(inp ==  -1) return 0;
        input_packet[word_num] |= inp;
        }
    return input_packet;
}
```

C.2.4 com_defs.h

```
/* Com port reader defines
E. Rosten 2002
*/

/* PIC (programmable interrupt controllers */
#define PIC1    0x20    //IRQs 0-7
#define PIC1_IRQ_MASK   (PIC1+1)
#define PIC2    0xA0    //IRQs 8-15
#define PIC2_IRQ_MASK   (PIC2+1)
#define PIC_END_INTERRUPT 0x20

/*Com port addresses */
#define COM1    0x3F8
#define COM2    0x2F8
#define COM3    0x3E8
#define COM4    0x2E8

/*Com port interrupt vectors */
#define COM1_INT    0x0C
#define COM2_INT    0x0B
#define COM3_INT    0x0C
#define COM4_INT    0x0B

/*Com port IRQ masks*/
#define COM1_IRQ_MASK   0x10
#define COM2_IRQ_MASK   0x08
#define COM3_IRQ_MASK   0x10
#define COM4_IRQ_MASK   0x08

/* Choose COM port       */
#define COM_PORT        COM1
#define COM_INT_VEC     COM1_INT
#define COM_IRQ_MASK    COM1_IRQ_MASK

/*Define registers and bits */
#define COM_DIVISOR_LATCH_LOW_BYTE      COM_PORT
#define COM_RX_HOLD_BUFFER              COM_PORT
#define COM_DIVISOR_LATCH_HIGH_BYTE     (COM_PORT+1)

/*Interrupt enable register */
#define COM_INTERRUPT_ENABLE_REGISTER   (COM_PORT+1)
#define COM_IER                         COM_INTERRUPT_ENABLE_REGISTER

#define COM_ENABLE_RX_INT       0x01        //-------1
#define COM_ENABLE_TX_CLEAR_INT 0x02        //------1-

/*FIFO control register */
```

```
#define COM_FIFO_CONTROL_REGISTER       (COM_PORT+2)
#define COM_FCR                         COM_FIFO_CONTROL_REGISTER

#define COM_1_BYTE              0x00        //00------
#define COM_4_BYTES             0x40        //01------
#define COM_8_BYTES             0x80        //10------
#define COM_14_BYTES            0xC0        //11------
#define COM_64_BYTE_FIFO        0x20        //--1-----
#define COM_DMA                 0x08        //----1---
#define COM_CLEAR_TX_FIFO       0x04        //-----1--
#define COM_CLEAR_RX_FIFO       0x02        //------1-
#define COM_ENABLE_FIFO         0x01        //-------1

/*Line control ergister */
#define COM_LINE_CONTROL_REGISTER       (COM_PORT+3)
#define COM_LCR                         COM_LINE_CONTROL_REGISTER

/*The divisor latch byte registers are shadowed. The DLAB (divisor latch
access bit) must be set in order to gain access to these two registers*/

#define COM_DLAB                0x80        //1-------
#define COM_NO_PARITY           0           //--000---
#define COM_ODD_PARITY          0x08        //--001---
#define COM_EVEN_PARITY         0x18        //--011---
#define COM_HIGH_PARITY         0x28        //--101---
#define COM_LOW_PARITY          0x38        //--111---
#define COM_1_STOP_BIT          0x00        //-----0--
#define COM_2_STOP_BIT          0x04        //-----1--
#define COM_5_DATA_BITS         0x00        //------00
#define COM_6_DATA_BITS         0x01        //------01
#define COM_7_DATA_BITS         0x02        //------10
#define COM_8_DATA_BITS         0x03        //------11

/*Modem control register */
#define COM_MODEM_CONTROL_REGISTER      (COM_PORT+4)
#define COM_MCR                         COM_MODEM_CONTROL_REGISTER

#define COM_FORCE_DATA_TERMINAL_READY   1   //-------1
#define COM_FORCE_DTR           0x01        //-------1
#define COM_FORCE_REQUEST_TO_SEND       2   //------1-
#define COM_FORCE_RTS           0x02        //------1-
#define COM_AUX_OUT_1           0x04        //-----1--
#define COM_AUX_OUT_2           0x08        //----1---

/*Line status register */
#define COM_LINE_STATUS_REGISTER        (COM_PORT+5)
#define COM_LSR                         COM_LINE_STATUS_REGISTER
```

```
#define COM_ERROR_IN_RX_FIFO    0x80        //1-------
#define COM_EMPTY_RX_DATA_REG   0x40        //-1------
#define COM_EMPTY_TX_DATA_REG   0x20        //--1-----
#define COM_BREAK_INTERRUPT     0x10        //---1----
#define COM_FRAMING_ERROR       0x08        //----1---
#define COM_PARITY_ERROR        0x04        //-----1--
#define COM_OVERRUN_ERROR       0x02        //------1-
#define COM_DATA_READY          0x01        //-------1
```

C.2.5 com.h

```
#ifndef INC_COM_H
    #define INC_COM_H

    int     get_byte(void);
    void    enable_com1(void);
    void    disable_com1(void);

    extern  unsigned long int wait_counter;
#endif
```

C.2.6 com.c

```
/* Com port reader.
This file defines 3 external functions:

void enable_com1(void)  enables a basic 56K com port on COM1
void disable_com1(void) diasables it

int  get_byte()         returns a byte from the com port. It stalls until a
                        byte becomes avaliable. A return of -1 indicates that
                        escape ahs been pressed.

E. Rosten 2002
*/

#include "com.h"
#include "com_defs.h"

#define RING_BUF_SIZE 1024

int buf_start = 0;
int buf_end = 0;
unsigned char buffer[RING_BUF_SIZE+1];

unsigned long wait_counter = 0;

void interrupt com_port_int();
void interrupt (*old_isr)();

void enable_com1(void)
{
    outportb(COM_IER, 0);           //Temporarily disable interrupts until the
                                    //Serial port is set up

    old_isr = getvect(COM_INT_VEC); //Remember the old interrupt service
                                    //routine

    setvect(COM_INT_VEC, com_port_int);


    /********            Start COM port settings                    *********/


    //Set the baud rate first.

    outportb(COM_LINE_CONTROL_REGISTER , COM_DLAB);

    outportb(COM_DIVISOR_LATCH_LOW_BYTE , 0x02); /*Set the baud rate clock
                                    divider. The clock runs at 115,200 KBaud,
```

```
                                    so /2 gives 57,600. The default is 2400
                                    set by the BIOS. The clock divider is a 2
                                    byte integer, little endian:*/
    outportb(COM_DIVISOR_LATCH_HIGH_BYTE, 0);

    //Set up the other serial port properties:
    outportb(COM_LCR, COM_NO_PARITY | COM_1_STOP_BIT | COM_8_DATA_BITS);

    //Set up the FIFO
    outportb(COM_FCR, COM_ENABLE_FIFO | COM_CLEAR_TX_FIFO | COM_CLEAR_RX_FIFO|
                      COM_14_BYTES);

    //Set up the output lines. Som machines need AUX 2
    outportb(COM_MCR, COM_FORCE_DTR | COM_FORCE_RTS | COM_AUX_OUT_2);

    //Set up the IRQ mask in the progrmmable interrupt controller
    outportb(PIC1_IRQ_MASK,(inportb(PIC1_IRQ_MASK) & ~COM_IRQ_MASK));


    //Set up the UART to interrupt on received data
    outportb(COM_IER , COM_ENABLE_RX_INT);
}

void disable_com1()
{

    //Disable interrupts from the UART
    outportb(COM_IER , 0);

    //Disable interrupts from the PIC
    outportb(PIC1_IRQ_MASK,(inportb(PIC1_IRQ_MASK) | COM_IRQ_MASK));

    //Restore the old ISR
    setvect(COM_INT_VEC, old_isr);
}

void interrupt com_port_int()  /* Interrupt Service Routine (ISR) for PORT1 */
{
    unsigned int c;

    while(inportb(COM_LINE_STATUS_REGISTER) & COM_DATA_READY)
        {
        c = inportb(COM_RX_HOLD_BUFFER);

        buffer[buf_end++] = c;
        buf_end %= RING_BUF_SIZE;
        }
```

```
    //Tell PIC1 that the interrupt has been serviced
    outportb(PIC1, PIC_END_INTERRUPT);
}

int get_byte()
{
    int ret;

    while(buf_start == buf_end && !(kbhit() && getch()==27))wait_counter++;

    if(buf_start==buf_end) return -1;

    ret = buffer[buf_start++];

    buf_start %= RING_BUF_SIZE;

    return ret;
}
```

D Rotor calibration function

D.1 Thrust

Rotor1		Rotor2		Rotor3		Rotor4	
Thrust	Input	Thrust	Input	Thrust	Input	Thrust	Input
0.00	0.00	0.00	0.00	0.00	0.00	0.00	11.21
0.05	25.97	0.05	25.64	0.05	32.00	0.05	26.89
0.10	27.51	0.10	27.02	0.10	35.09	0.10	29.67
0.15	29.06	0.15	28.40	0.15	37.46	0.15	32.13
0.20	30.59	0.20	29.78	0.20	39.84	0.20	34.55
0.25	32.10	0.25	31.12	0.25	42.92	0.25	36.98
0.30	33.60	0.30	32.46	0.30	45.93	0.30	39.41
0.35	35.11	0.35	33.79	0.35	48.69	0.35	42.08
0.40	36.66	0.40	35.15	0.40	51.58	0.40	44.84
0.45	38.21	0.45	36.65	0.45	54.59	0.45	47.74
0.50	39.76	0.50	38.16	0.50	57.74	0.50	50.63
0.55	41.65	0.55	39.67	0.55	60.92	0.55	53.45
0.60	43.60	0.60	41.23	0.60	64.09	0.60	56.42
0.65	45.41	0.65	42.80	0.65	68.57	0.65	59.57
0.70	46.87	0.70	44.37	0.70	72.79	0.70	63.09
0.75	48.33	0.75	46.04	0.75	76.78	0.75	66.44
0.80	49.79	0.80	47.76	0.80	80.89	0.80	69.52
0.85	51.41	0.85	49.48	0.85	84.92	0.85	72.91
0.90	53.06	0.90	51.10	0.90	89.95	0.90	76.50
0.95	54.71	0.95	52.66	0.95	94.42	0.95	80.29
1.00	56.35	1.00	54.22	1.00	99.76	1.00	83.96
1.05	58.00	1.05	56.10	1.05	105.10	1.05	87.42
1.10	59.65	1.10	58.28	1.10	109.02	1.10	90.97
1.15	61.44	1.15	60.42	1.15	115.87	1.15	95.17
1.20	63.26	1.20	62.37	1.20	121.84	1.20	100.41
1.25	65.10	1.25	64.32	1.25	126.98	1.25	103.26
1.30	67.10	1.30	66.35	1.30	136.17	1.30	105.76
1.35	69.11	1.35	68.43	1.35	138.66	1.35	116.46
1.40	71.42	1.40	70.51	1.40	148.36	1.40	118.75
1.45	73.98	1.45	72.59	1.45	156.54	1.45	121.88
1.50	76.21	1.50	74.67	1.50	164.28	1.50	126.63
1.55	78.21	1.55	76.82	1.55	171.12	1.55	133.25
1.60	80.23	1.60	78.97	1.60	181.18	1.60	138.49
1.65	82.38	1.65	81.18	1.65	186.35	1.65	144.53
1.70	84.53	1.70	83.45	1.70	193.43	1.70	150.92
1.75	87.37	1.75	85.87	1.75	202.88	1.75	160.07
1.80	90.57	1.80	88.62	1.80	211.64	1.80	166.53
1.85	94.87	1.85	91.72	1.85	218.49	1.85	176.55
1.90	97.41	1.90	95.18	1.90	226.29	1.90	195.10
1.95	99.89	1.95	98.31	1.95	233.37	1.95	212.24
2.00	103.84	2.00	103.98	2.00	243.54	2.00	231.55
2.05	107.78	2.05	110.59	2.05	254.98	2.05	243.41
2.10	112.15	2.10	114.69	2.05	255.00	2.10	255.00
2.15	116.33	2.15	121.92	2.05	255.00	2.10	255.00
2.20	119.47	2.20	126.75	2.05	255.00	2.10	255.00

Rotor1		Rotor2		Rotor3		Rotor4	
Thrust	Input	Thrust	Input	Thrust	Input	Thrust	Input
2.25	122.67	2.25	131.24	2.05	255.00	2.10	255.00
2.30	125.87	2.30	139.52	2.05	255.00	2.10	255.00
2.35	129.04	2.35	144.28	2.05	255.00	2.10	255.00
2.40	131.71	2.40	145.58	2.05	255.00	2.10	255.00
2.45	134.16	2.45	146.27	2.05	255.00	2.10	255.00
2.50	136.86	2.50	146.96	2.05	255.00	2.10	255.00
2.55	139.69	2.55	147.65	2.05	255.00	2.10	255.00
2.60	143.89	2.60	148.34	2.05	255.00	2.10	255.00
2.65	148.07	2.65	149.03	2.05	255.00	2.10	255.00
2.70	153.19	2.70	149.72	2.05	255.00	2.10	255.00
2.75	180.82	2.75	152.37	2.05	255.00	2.10	255.00
2.80	194.93	2.80	160.73	2.05	255.00	2.10	255.00
2.85	198.54	2.85	167.33	2.05	255.00	2.10	255.00
2.90	207.92	2.90	176.78	2.05	255.00	2.10	255.00
2.95	221.16	2.95	190.11	2.05	255.00	2.10	255.00
3.00	251.32	3.00	192.41	2.05	255.00	2.10	255.00
3.03	255.00	3.05	194.71	2.05	255.00	2.10	255.00
3.03	255.00	3.10	205.31	2.05	255.00	2.10	255.00
3.03	255.00	3.15	215.26	2.05	255.00	2.10	255.00
3.03	255.00	3.20	247.82	2.05	255.00	2.10	255.00
3.03	255.00	3.23	255.00	2.05	255.00	2.10	255.00
3.03	255.00	3.23	255.00	2.05	255.00	2.10	255.00
3.03	255.00	3.23	255.00	2.05	255.00	2.10	255.00
3.03	255.00	3.23	255.00	2.05	255.00	2.10	255.00

D.2 Torque

Rotor1		Rotor2		Rotor3		Rotor4	
Thrust	Input	Thrust	Input	Thrust	Input	Thrust	Input
0.00×10^{-00}	0.00	0.00×10^{-00}	0.00	0.00×10^{-00}	0.00	0×10^{-00}	0.00
1.00×10^{-05}	16.28	1.00×10^{-05}	26.06	2.00×10^{-06}	36.10	2×10^{-06}	27.38
2.00×10^{-05}	20.69	2.00×10^{-05}	30.60	4.00×10^{-06}	37.37	4×10^{-06}	29.77
3.00×10^{-05}	23.56	3.00×10^{-05}	33.79	6.00×10^{-06}	38.63	6×10^{-06}	31.55
4.00×10^{-05}	26.03	4.00×10^{-05}	36.37	8.00×10^{-06}	39.89	8×10^{-06}	33.27
5.00×10^{-05}	28.10	5.00×10^{-05}	38.58	1.00×10^{-05}	41.30	1×10^{-05}	34.99
6.00×10^{-05}	30.14	6.00×10^{-05}	40.91	1.20×10^{-05}	42.72	1×10^{-05}	36.61
7.00×10^{-05}	31.87	7.00×10^{-05}	43.45	1.40×10^{-05}	44.14	1×10^{-05}	38.23
8.00×10^{-05}	33.59	8.00×10^{-05}	46.01	1.60×10^{-05}	46.17	2×10^{-05}	39.85
9.00×10^{-05}	35.28	9.00×10^{-05}	48.61	1.80×10^{-05}	49.13	2×10^{-05}	41.30
1.00×10^{-04}	36.88	1.00×10^{-04}	51.22	2.00×10^{-05}	60.89	2×10^{-05}	42.73
1.10×10^{-04}	38.47	1.10×10^{-04}	53.83	2.20×10^{-05}	62.25	2×10^{-05}	44.16
1.20×10^{-04}	40.07	1.20×10^{-04}	57.64	2.40×10^{-05}	63.61	2×10^{-05}	45.75
1.30×10^{-04}	41.74	1.30×10^{-04}	62.14	2.60×10^{-05}	64.98	3×10^{-05}	47.57
1.40×10^{-04}	43.41	1.40×10^{-04}	65.59	2.80×10^{-05}	65.83	3×10^{-05}	49.40
1.50×10^{-04}	45.10	1.50×10^{-04}	67.43	3.00×10^{-05}	66.67	3×10^{-05}	52.32
1.60×10^{-04}	47.15	1.60×10^{-04}	69.26	3.20×10^{-05}	67.51	3×10^{-05}	55.89
1.70×10^{-04}	49.21	1.70×10^{-04}	71.24	3.40×10^{-05}	68.35	3×10^{-05}	59.85
1.80×10^{-04}	51.73	1.80×10^{-04}	73.31	3.60×10^{-05}	69.19	4×10^{-05}	62.59
1.90×10^{-04}	54.54	1.90×10^{-04}	75.65	3.80×10^{-05}	70.04	4×10^{-05}	65.19
2.00×10^{-04}	57.93	2.00×10^{-04}	79.12	4.00×10^{-05}	70.98	4×10^{-05}	67.00
2.10×10^{-04}	61.45	2.10×10^{-04}	81.72	4.20×10^{-05}	71.92	4×10^{-05}	68.81
2.20×10^{-04}	64.99	2.20×10^{-04}	84.02	4.40×10^{-05}	72.86	4×10^{-05}	70.51
2.30×10^{-04}	68.32	2.30×10^{-04}	86.90	4.60×10^{-05}	73.80	5×10^{-05}	71.98
2.40×10^{-04}	71.60	2.40×10^{-04}	90.27	4.80×10^{-05}	74.74	5×10^{-05}	73.45
2.50×10^{-04}	74.81	2.50×10^{-04}	95.08	5.00×10^{-05}	76.26	5×10^{-05}	74.92
2.60×10^{-04}	78.10	2.60×10^{-04}	101.67	5.20×10^{-05}	77.98	5×10^{-05}	76.52
2.70×10^{-04}	81.60	2.70×10^{-04}	106.88	5.40×10^{-05}	79.70	5×10^{-05}	78.13
2.80×10^{-04}	85.52	2.80×10^{-04}	112.22	5.60×10^{-05}	85.26	6×10^{-05}	79.75
2.90×10^{-04}	90.41	2.90×10^{-04}	118.24	5.80×10^{-05}	88.40	6×10^{-05}	82.33
3.00×10^{-04}	96.26	3.00×10^{-04}	126.60	6.00×10^{-05}	91.10	6×10^{-05}	85.10
3.10×10^{-04}	101.82	3.10×10^{-04}	132.14	6.20×10^{-05}	93.34	6×10^{-05}	88.21
3.20×10^{-04}	106.69	3.20×10^{-04}	138.01	6.40×10^{-05}	95.71	6×10^{-05}	90.66
3.30×10^{-04}	111.12	3.30×10^{-04}	148.42	6.60×10^{-05}	98.45	7×10^{-05}	92.24
3.40×10^{-04}	115.25	3.40×10^{-04}	157.93	6.80×10^{-05}	100.86	7×10^{-05}	93.81
3.50×10^{-04}	119.20	3.50×10^{-04}	173.14	7.00×10^{-05}	102.84	7×10^{-05}	95.41
3.60×10^{-04}	123.04	3.60×10^{-04}	192.80	7.20×10^{-05}	104.82	7×10^{-05}	97.09
3.70×10^{-04}	126.85	3.70×10^{-04}	215.84	7.40×10^{-05}	106.51	7×10^{-05}	98.76
3.80×10^{-04}	130.69	3.80×10^{-04}	217.51	7.60×10^{-05}	108.17	8×10^{-05}	103.32
3.90×10^{-04}	134.68	3.90×10^{-04}	219.18	7.80×10^{-05}	109.83	8×10^{-05}	111.16
4.00×10^{-04}	139.10	4.00×10^{-04}	220.89	8.00×10^{-05}	112.78	8×10^{-05}	112.50
4.10×10^{-04}	144.14	4.10×10^{-04}	222.64	8.20×10^{-05}	116.64	8×10^{-05}	113.84
4.20×10^{-04}	150.20	4.20×10^{-04}	224.40	8.40×10^{-05}	120.56	8×10^{-05}	115.14
4.30×10^{-04}	158.97	4.30×10^{-04}	228.91	8.60×10^{-05}	121.86	9×10^{-05}	116.18
4.40×10^{-04}	173.11	4.40×10^{-04}	255.00	8.80×10^{-05}	123.15	9×10^{-05}	117.22

Rotor1		Rotor2		Rotor3		Rotor4	
Thrust	Input	Thrust	Input	Thrust	Input	Thrust	Input
4.50×10^{-04}	181.65	4.40×10^{-04}	255.00	9.00×10^{-05}	124.45	9×10^{-05}	118.26
4.60×10^{-04}	188.29	4.40×10^{-04}	255.00	9.20×10^{-05}	125.65	9×10^{-05}	119.29
4.70×10^{-04}	197.61	4.40×10^{-04}	255.00	9.40×10^{-05}	126.78	9×10^{-05}	120.45
4.80×10^{-04}	205.89	4.40×10^{-04}	255.00	9.60×10^{-05}	127.91	1×10^{-04}	121.85
4.90×10^{-04}	210.65	4.40×10^{-04}	255.00	9.80×10^{-05}	129.04	1×10^{-04}	123.25
5.00×10^{-04}	214.85	4.40×10^{-04}	255.00	1.00×10^{-04}	131.98	1×10^{-04}	124.65
5.10×10^{-04}	219.19	4.40×10^{-04}	255.00	1.02×10^{-04}	145.77	1×10^{-04}	126.91
5.18×10^{-04}	255.00	4.40×10^{-04}	255.00	1.04×10^{-04}	148.16	1×10^{-04}	129.47
5.18×10^{-04}	255.00	4.40×10^{-04}	255.00	1.06×10^{-04}	150.62	1×10^{-04}	133.88
5.18×10^{-04}	255.00	4.40×10^{-04}	255.00	1.08×10^{-04}	153.28	1×10^{-04}	138.06
5.18×10^{-04}	255.00	4.40×10^{-04}	255.00	1.10×10^{-04}	156.82	1×10^{-04}	141.34
				1.12×10^{-04}	162.52	1.12×10^{-04}	143.95
				1.14×10^{-04}	167.47	1.14×10^{-04}	146.46
				1.16×10^{-04}	171.33	1.16×10^{-04}	148.90
				1.18×10^{-04}	175.13	1.18×10^{-04}	151.83
				1.20×10^{-04}	179.92	1.20×10^{-04}	155.22
				1.22×10^{-04}	183.09	1.22×10^{-04}	160.01
				1.24×10^{-04}	185.64	1.24×10^{-04}	164.84
				1.26×10^{-04}	187.25	1.26×10^{-04}	193.02
				1.28×10^{-04}	188.86	1.28×10^{-04}	220.04
				1.30×10^{-04}	190.37	1.30×10^{-04}	222.09
				1.32×10^{-04}	191.65	1.32×10^{-04}	224.14
				1.34×10^{-04}	192.92	1.34×10^{-04}	226.94
				1.36×10^{-04}	194.20	1.36×10^{-04}	255.00
				1.38×10^{-04}	195.61	1.36×10^{-04}	255.00
				1.40×10^{-04}	197.26	1.36×10^{-04}	255.00
				1.42×10^{-04}	198.90	1.36×10^{-04}	255.00
				1.44×10^{-04}	202.81	1.36×10^{-04}	255.00
				1.46×10^{-04}	221.53	1.36×10^{-04}	255.00
				1.48×10^{-04}	223.59	1.36×10^{-04}	255.00
				1.50×10^{-04}	226.80	1.36×10^{-04}	255.00
				1.52×10^{-04}	238.94	1.36×10^{-04}	255.00
				1.54×10^{-04}	241.53	1.36×10^{-04}	255.00
				1.56×10^{-04}	243.27	1.36×10^{-04}	255.00
				1.58×10^{-04}	245.04	1.36×10^{-04}	255.00
				1.59×10^{-04}	255.00	1.36×10^{-04}	255.00