ELEC2205 D4 – System Design Exercise

Ben Rowlinson
bdr1g15
27882888
MEng Electronic Engineering
Dr Yoshishige Tsuchiya
17/03/2017

Contents

List of Figures

Nomenclature

DMM	Digital Multimeter
DMP	Digital Motion Processing
ESC	Electronic Speed Controller
Il Matto/Il-Matto/MCU	Il Matto Micro-controller development board
IMU	Inertial Measurement Unit
LiPo	Lithium Polymer
PDB	Power Distribution Board
PID	Proportional, Integral and Differential control
PSU	Power Supply Unit
TYPR	Throttle, Yaw, Pitch, Roll
UAV	Unmanned Aerial Vehicle
YPR	Euler Angles: Yaw, Pitch, Roll

1 Contribution

As team leader, my contribution to the project was more administrative than hands-on. However, this did include the key task of integrating all the design modules together. I also completed some smaller auxiliary tasks involving tuning the PID controller, managing power distribution, interfacing the IR sensor, and measuring battery voltage. In addition to these roles, I was involved to some degree, in the development of every design module to ensure tasks were being completed successfully and would integrate smoothly. I completed all the above tasks correctly, except for tuning the PID control to the point of stable flight being maintained.

2 Specification

2.1 Communications and Control Integration

The communications and control modules must integrate together seamlessly, so that the correct operation of both systems is not affected. The packet transmission time should not exceed 3ms to ensure that neither system is slowed too much by the interface.

2.2 PID Tuning

The PID gains of the controller for all three axes of rotation (Yaw, Pitch, Roll) must be tuned such that drone maintains stable flight and responds accurately to the pilot's commands.

2.3 Power Distribution

Every component on-board must be correctly powered from a single LiPo battery, via a PDB. The controller must be powered by a USB supply from a Laptop.

2.4 IR Proximity Sensor

The IR Sensor must accurately measure the current altitude within its operating range (15 to 100cm) to aid the pilot in landing and cargo acquisition manoeuvres.

2.5 Battery Voltage Measurement

The Battery Voltage Measurement circuit must measure the Battery voltage to an accuracy of ± 0.1V, without drawing a high current (≤ 0.1mA), so the pilot can monitor the power usage of the UAV.

3 Design and Simulation

3.1 Communications and Control Integration

The main point of intersection between the communications and control sub-systems, is the UART interface between the on-board Il Matto and Arduino Leonardo boards. By consulting the communications and control sub-system teams, we drew up an initial control flow diagram for the Leonardo (Figure 1 in Appendix B). I will be focusing on the "Request desired TYPR

data" and "Receive desired TYPR data" sections.
The interface between these modules can be difficult to achieve, without causing each of the sub-systems to experience too much latency to effectively control the UAV. In this design, we took the control system to be more critical to the safe operation of the UAV. As such, the Leonardo board was made the 'Master' in this interface, so it could request new TYPR values when desired. Our design called for a maximum packet transmission time of 3ms. Equation 1 shows how to calculate the time taken to transmit a packet.

$$Time\_Taken = (Number\_Of\_Bytes \times Byte\_Length) \div Baud\_Rate[s] \tag{1}$$

$$Number\_of\_Bytes = 17$$

$$Byte\_Length = 10$$

$$Baud\_Rate = 57600$$

$$Time\_Taken = 2.85ms$$

When the Leonardo wants to receive data it interrupts the Il Matto on the rising edge of a short pulse [8]. This sets a flag [10] on the Il Matto, which then sends the TYPR packet at the next available opportunity, and resets the flag. This has to happen 100 times per second to match the IMU data rate.
Our packet format was dictated by the fact that the serial library chosen (debug.h) could only transmit 8-bit bytes, plus 1 start and 1 stop bit. To simplify the receiving procedure we constructed a 17-byte packet format of:

$$"t\%c\%c\%cy\%c\%c\%cp\%c\%c\%cr\%c\%c\%c\backslash n" \tag{2}$$

The chars 't', 'y', 'p' and 'r' are identifiers for TPYR respectively. '%c' is a char 0-9, with the 10-bit ADC value from the X-Y potentiometers being represented by "%c%c%c". The packet is closed by a newline character '\n'. The Arduino then leaves the receiving function [7] after changing the desired TYPR values, as per Figure 1. The ADC values only range from 256 to 768, therefore only "%c%c%c" is needed to represent the full range of ADC values we are expecting. This is by no means an optimal solution but significantly eases the receiving procedure on the Leonardo, as the input string can be read in 1 char at time into a string. The string is then converted to an int and assigned to the correct TYPR value. A similar procedure could be used to update PID gain values.

3.2 PID Tuning

In order to gauge how to tune our PID controller, we researched PID gain values for similar Quad-copter configurations, as well as the methods for fine tuning the control [2][3]. Since the PID gains vary considerably from one drone to the next, these researched values and methods were used as initial values which we would later refine during the tuning stage. We wanted to implement a feature which would allow us to update PID gains 'on the fly' but we were not able to get this to function correctly.

3.3 Power Distribution

For power distribution we wanted to use a single 11.1V 3s LiPo battery to power all on-board components. This required a PDB which could handle the current spikes due to rapid changes in thrust but also supply a smooth 5V to the Arduino and Il Matto. The Il Matto already contains a 3.3V voltage regulator which needs a minimum 4.5V input to operate correctly. The current requirements on the PDB meant I had to supplement the tri-pad board with additional cable and solder. This ensured we could safely operate the motors at high current without damaging the LiPo or the rest of the UAV. This extra cable added weight to the drone. The schematic for the PDB is found in Appendix C of the Team Report.

3.4 IR Proximity Sensor

The Sharp GP2Y0A21YK0F IR Proximity Sensor has a proprietary IC which handles the signal processing involved in estimating the proximity. This module then outputs an analogue voltage when powered, which corresponds to the distance according to Figure 2 in the IR Sensor Datasheet[1]. This output means the IR sensor effectively acts like potentiometer. A test program was written which performs a 10-bit ADC conversion and outputs the ADC value over a UART interface to a serial terminal (PuTTY).
The ADC value is not the actual distance so must be converted to a distance in cm. The approximate conversion and relation is shown in Equation 3 and Equation 4 respectively.

$$V_{IR} = \frac{adc_{val}}{adc_{res}} \times V_{cc}[V] \tag{3}$$

$$V_{cc} = 3.3V$$

$$adc_{res} = 2^{10} = 1024$$

$$distance \approx \frac{k_{IR}}{V_{IR}}[cm] \tag{4}$$

The constant k can vary from sensor to sensor so a calibration routine must be implemented to be able to collect reasonable data. Any distance readings below 15cm or above 100cm can be disregarded by the pilot, as this is outside the accurate operating range of the sensor. The measurement only takes place when the on-board Il Matto transmits telemetry back to ground control.

3.5 Battery Voltage Measurement

The battery voltage of up to 12.6V must be stepped down to a safe input ($\leq$3.3V) for the Il Matto ADC input. This is achieved by taking a divided voltage into the Il Matto. This measurement only takes place when the on-board Il Matto transmits telemetry.

$$V_{adc} = \frac{R_2}{R_1 + R_2} \times V_{LiPo}[V] \tag{5}$$
$$R_1 = 100k\Omega$$

$$R_2 = 33k\Omega$$

The voltage into the Il Matto is converted to a 10-bit integer and transmitted back to ground control, where it is converted back to a float.

3.6 Schematic

As part of the design process, I compiled our design plan into a high-level block diagram and later an EAGLE schematic of the complete electronic design. This EAGLE schematic allowed us to accurately plan out our on-board layout and ensure that we had enough power connections for every component. This schematic was also used as a reference by the team to ensure consistent assembly of the circuitry. (Appendix C of Team Report)

4 Testing and Results

4.1 Communications and Control Integration

I wrote a pair of test programs (onboard_serial_test_ard_3.ino and onboard_serial_test_final.c) investigating each side of the interface to verify the operation and speed of the protocol described in 3.1. The protocol could not use packets which exceeded 3ms in time. The Channel 1 signal on these oscilloscope traces is the short pulse transmission from the Leonardo [8] which requests the TYPR data [9], shown on Channel 2 (Figure 2 and Figure 3 in Appendix B).

We initially tested with a baud rate of 9,600, however we found this to be too slow to transmit the 17 byte packets in the specified time of 3ms (One packet took 17.71ms @ 9,600). A reduced packet size of 13-bytes is still not able to achieve the correct data rate (Figure 2 in Appendix B). Switching to a baud rate of 57,600 we were able to transmit a packet in 2.95ms (Figure 3 in Appendix B).

We attempted faster baud rates, such as 115,200, but we found the data was more likely to be corrupted, so 57,600 baud was settled on.

These tests meant that when we wanted to perform a critical path test, we could simply insert my interfacing program into the existing communications and control sub-systems, knowing that it already worked correctly.

4.2 PID Tuning

Initially, our approach to testing our fully integrated UAV was to tether it down and apply throttle to see if it could achieve a stable flight. This testing method proved fruitless as the UAV was never under its own control, but instead was heavily influenced by the tethering set-up (Figure 4 in Appendix B).

We concluded that we ought to let the drone fly independently of tethers in a wide open space. Unfortunately, this led to the back legs and feet snapping off a number of times due to a consistent pitch backwards on take-off and landing.

We re-evaluated our approach to tuning the PID controller. Instead of trying to tune all the axes of rotation concurrently, we decided to suspend the drone from string such that only one axis was rotated around (Dubbed Balance testing : Figure 5 in Appendix B). This practice was taken from [4].

The centre of mass in these tests was initially placed below the axis of rotation leading to incorrect PID values. We realised this error and continued testing with the centre of mass now above the axis of rotation (Figure 6 in Appendix B). This gave us an inherently more unstable initial condition, which offered a better test of our control system. Again, we saw the consistent pitch backward being exhibited by the drone.

After an exhaustive debugging procedure, including IMU re-calibration, adding YPR offsets, and re-positioning the IMU, we concluded that the problem lay in one of the ESCs being un-calibrated with respect to the other three and as such never provided the balanced thrust required to maintain stable flight. After this was rectified we continued with the balance testing and soon established some viable PID gains for Pitch and Roll. We were able to perform tuning of the Pitch and Roll PID values but were unable to tune the Yaw control before the final hand-in. A fully tuned, independent flight test has not yet been performed on the UAV. Table 1 shows our Final PID values.

	p	i	d
y	0	0	0
p	0.4	0.001	20
r	0.2	0.001	20

Table 1: Final PID Gains

4.3 Power Distribution

Initially to verify the correct operation of the PDB I used a bench DMM to check for continuity errors, then powered the ESCs without motors connected. The 5V regulator was then added and tested using the above procedure, this time measuring the regulated output. Next, I performed a full power distribution test with all components connected, using a current limited (500mA) PSU to protect from unexpected current surges, but without driving the motors. Following these successful tests, I powered all the components including the motors simultaneously, again with a current limit (5A). Finally, a full PDB test with the LiPo battery was performed which was successful on the first attempt and remained operational throughout testing.

4.4 IR Proximity Sensor

To calibrate the IR sensor I placed a planar surface at 75cm from the sensor, measured the ADC value at that distance. I then scaled the conversion to the desired value (75cm), using Equation 3 and Equation 4, to find the value of:

$$k_{IR} \approx 24 \tag{6}$$

for our particular sensor. This gave us a reasonably good approximation of the current altitude, if the drone was in operating range. To improve our altitude measuring capabilities we could have included an ultra-sound module but our research showed that the very noisy environment created by the motors severely affects the the accuracy of the ultrasound modules. Another option could be using a barometric sensor to detect altitude change due to changing air pressure.

4.5 Battery Voltage Measurement

To test the operation of the Battery Voltage Measurement on the final design, we integrated my test code into the existing drone communications code . The telemetry code was already available to me so we could insert my code easily. We tested the measurements at range of voltages from 11.1-12.6V and all these results fell into the desired range of ±0.1V

5 Management and Team Working

5.1 Approach to Management

As team leader, my approach to managing the team was to designate specific roles to the team as early on as possible, whilst ensuring everyone was content with their place in the team. Built into this strategy was a sense of redundancy, in that no one was solely responsible for a particular sub-system. This meant that if a team member was away from meetings or working sessions, we would still be able to discuss problems reasonably well. Fortunately the entire team was very responsive to this and were consistent in attending meetings and lab time. We worked cohesively as a single unit throughout the duration of the project with no personal conflicts or arguing. The only disagreements were due to different approaches to the same engineering problem, which lent itself to a very healthy working environment. I felt it was important that the team met and worked together face-to-face at every opportunity, but also allowed for a period of time away from the project.

5.2 Designating Technical Roles

After a couple of days of initial research into the project, it became clear who was suitable for certain roles within the team. I partitioned the team such that one pair was responsible for communications (J Trickey and M Ibrahim) and another pair (L Gray and J Hindmarsh) were responsible for control. I worked closely with both pairs throughout the project, with them consulting me to ensure consistency in the design and keeping me up to date with their progress. This included resolving any issues quickly and efficiently. This meant when we came to integrate modules together I already had a firm grasp of the entire system functionality, easing the integration process.

5.3 Management Roles

I designated the team to complete various administrative roles within the team which would continue throughout the project. These included Treasurer(L Gray), Minutes(J Trickey), Documentation/Photography(M Ibrahim) and External Communications(J Hindmarsh). Each of the team members had these auxiliary roles to fulfil.

5.4 Project Planning

With regard to the planning process we identified a critical path for the construction of the drone electronics, that would enable basic functionality. The critical path was defined as: "The Communications module sending instructions over a radio channel to an on-board receiver, which buffers and forwards the data to the control element, on request. The control element processes these and the IMU data to afford control to the motors." Up until the point we could perform a full critical system integration, we worked on some of the extra functions, if and only if the critical task had already been performed. We planned for full critical system integration to be finalised on Thursday 9th March and from that point onwards we would concentrate on tuning our PID controller and adding extra functionality if possible.

5.5 Risk Management

We drew up a fairly broad risk assessment, based on our intensive research early on in the project. Fortunately, very little went wrong in our approach, that we didn't expect to go wrong. The chassis did shatter a number of times during flight and tethered tests, but we always ensured we were able to bring the drone back to flying condition or left enough time to manufacture a new frame. Any expected long term absences (more than 24hrs) were declared to me by the team in the first few days of the project so we could work around the team members' schedule without compromising the progress of the project.

6 Critical Evaluation and Reflection

6.1 On My Contribution

As team leader I felt it was important for me to take a step back and consider the project as a whole rather than focusing too much on performing a particular task. On reflection, I feel that I could have contributed more of my technical skills to the project. What I did contribute technically, however, did affect the success and functionality of our system. Retrospectively, I should have re-evaluated our approach to flight testing procedure, and approached it in a more structured and thoughtful way. If we had initially used the balance-test approach to tuning the PID controller, then we may have put ourselves in a better position to achieve stable flight. This is one of the key things I have learnt during this project; how to conceive functional and informative testing procedures. If we had applied more thought to this earlier on, we would have put ourselves in a better position.

6.2 On the Team's Progress

We produced a product which achieved many of the core goals we set out for ourselves in the project planning stage. Many of the elements of the design attempted were in fact completed to some degree - Wireless 2-way communications, self-built chassis, functional control etc. The main goal we did not achieve was to sustain stable, independent flight.

6.3 On Management

With regard to my managerial approach, I would say that it was a true pleasure to manage such a dedicated, motivated, and intelligent team. We always worked together as a unit. They made my experience on the project very enjoyable.

6.4 Overall Reflections

Overall, I am pleased with how we developed the project. Although we were not able to sustain a stable flight in the testing phase, the process we went through to get to that stage was challenging and fulfilling.

A Bibliography

References

[1] Sharp GP2Y0A21YK0F Datasheet
`http://www.sharpsma.com/webfm_send/1489`

[2] PID tuning guide
`http://iflyquad.com/2016/01/05/depthpidtuningguide/`

[3] Quadcopter Control Theory
`http://andrew.gibiansky.com/downloads/pdf/Quadcopter%20Dynamics,%20Simulation,%20and%20Control.pdf`

[4] How to recognise incorrect PI Gains
`https://www.youtube.com/watch?v=YNzqTGEl2xQ`

[5] Battery and IR read and transmit
`drone_serial_comms.c line 180`

[6] Battery and IR receive
`test5_basestation_comms.c line 219`

[7] serialEvent
`accelGyro.ino line 315,360,526`

[8] Pulse Function
`onboard_serial_test_ard_3.ino line 106`

[9] TYPR transmit
`onboard_serial_test_final.c line 64`

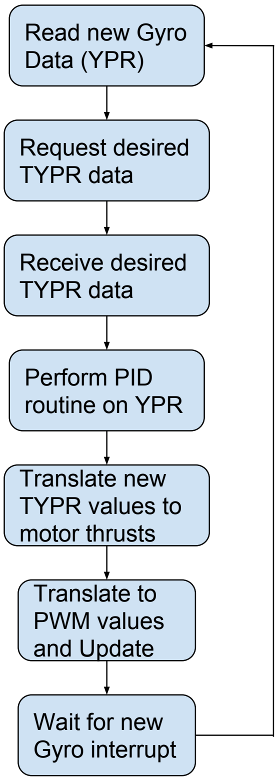
[10] Serial Flag
`onboard_serial_test_final.c line 18`

B Figures

Appendix including all figures used in this report

Figure 1: Control Flow Diagram

Diagram showing the flow of control in the Control Micro-controller



Read new Gyro Data (YPR)
Request desired TYPR data
Receive desired TYPR data
Perform PID routine on YPR
Translate new TYPR values to motor thrusts
Translate to PWM values and Update
Wait for new Gyro interrupt

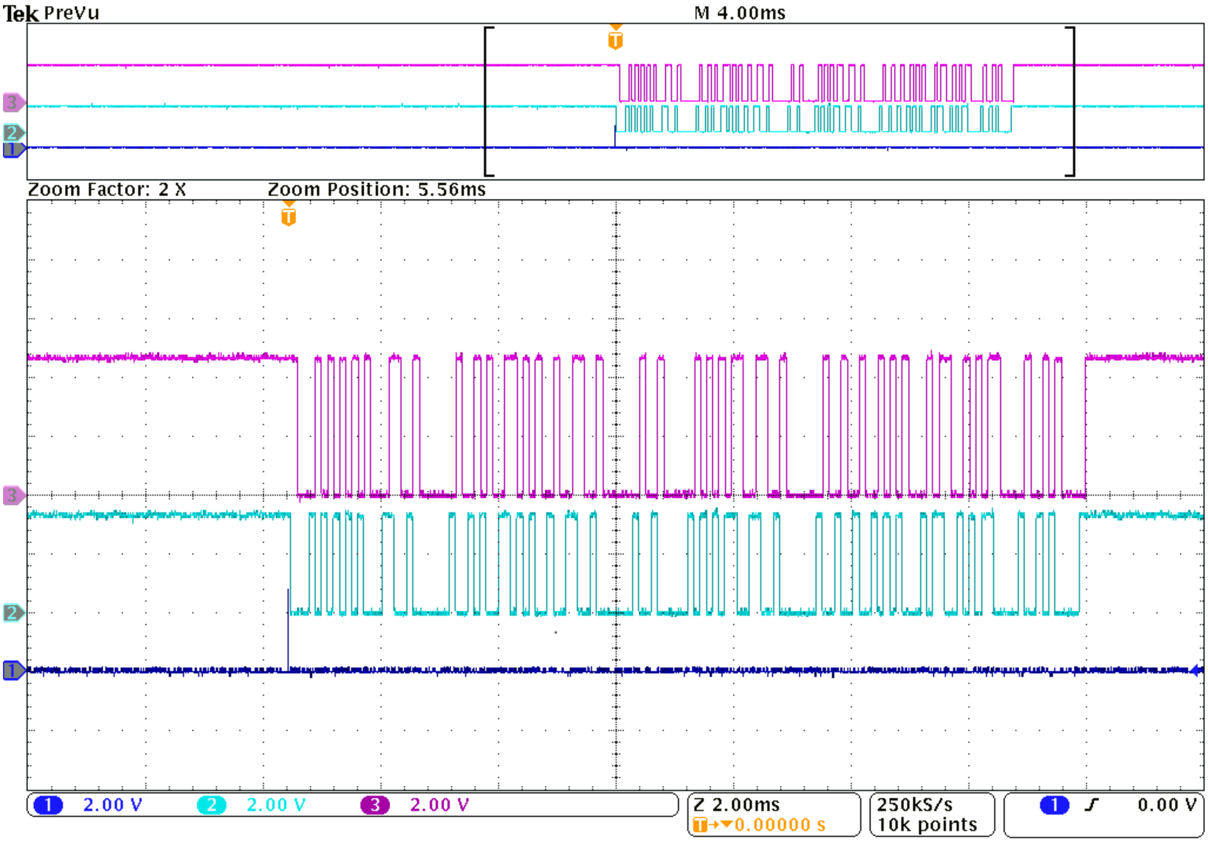


Figure 2: On-board Serial Test @ 9600 Baud

With 13-byte packet. Time taken = 13.4ms. ”t00y00p00r00\n”
Channel 1: Transmission request from Leonardo, Channel 2: UART Transmission from Il Matto, Channel 3: Ignore

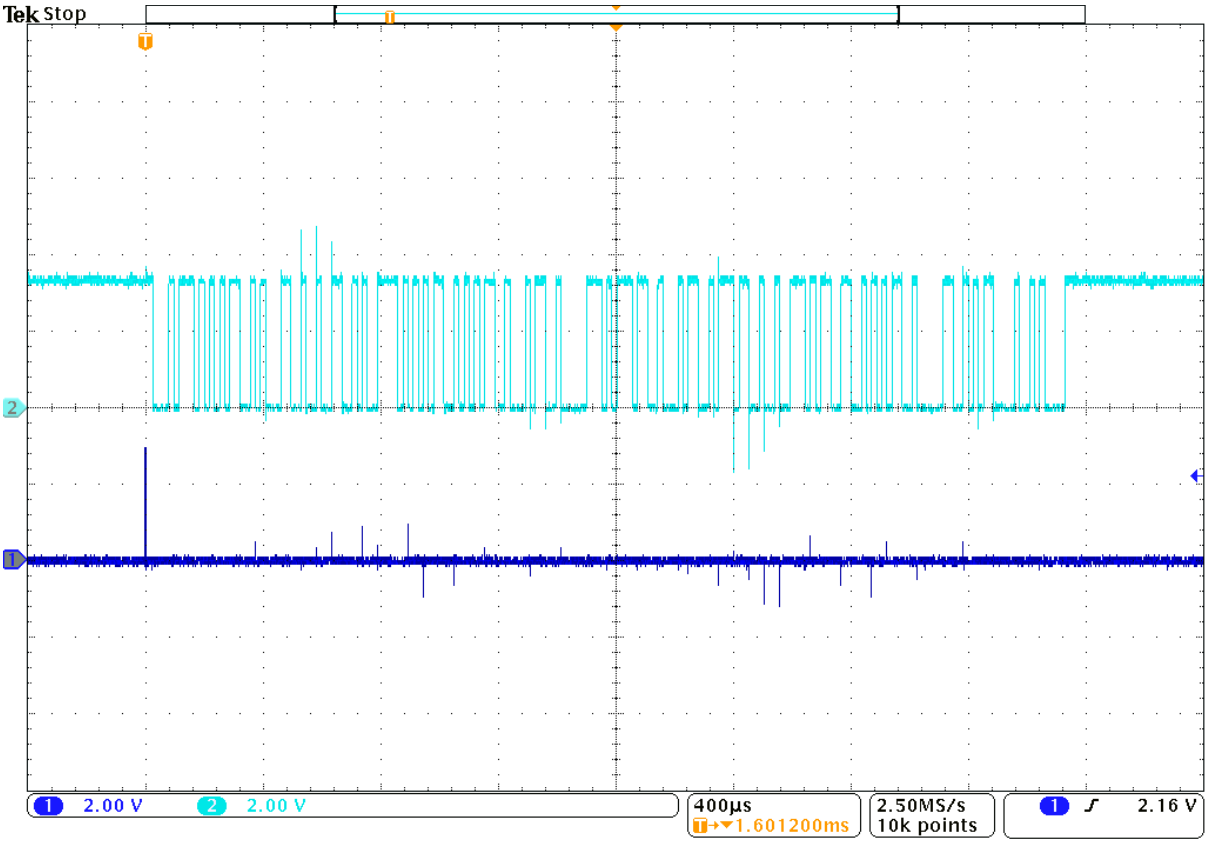


Figure 3: On-board Serial Test @ 57600 Baud

With 17-byte packet. Time taken = 2.95ms. ”t512y554p747r350\n”
Channel 1: Transmission request from Leonardo, Channel 2: UART Transmission from Il Matto

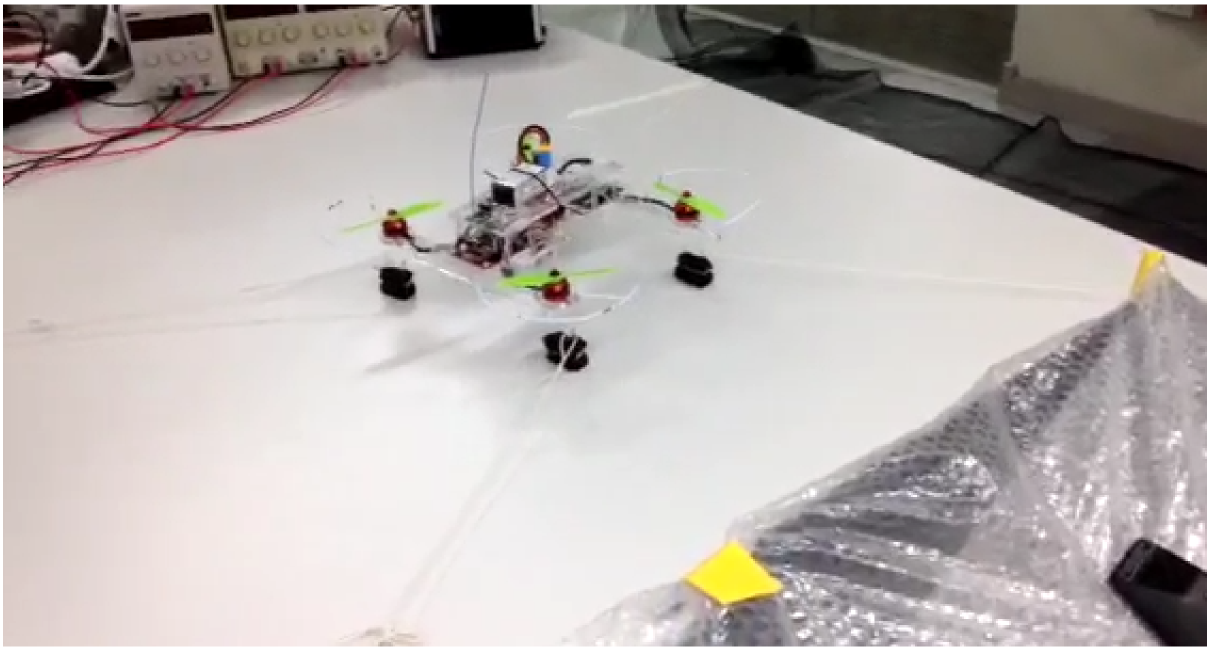


Figure 4: Tethered Flight Set-up

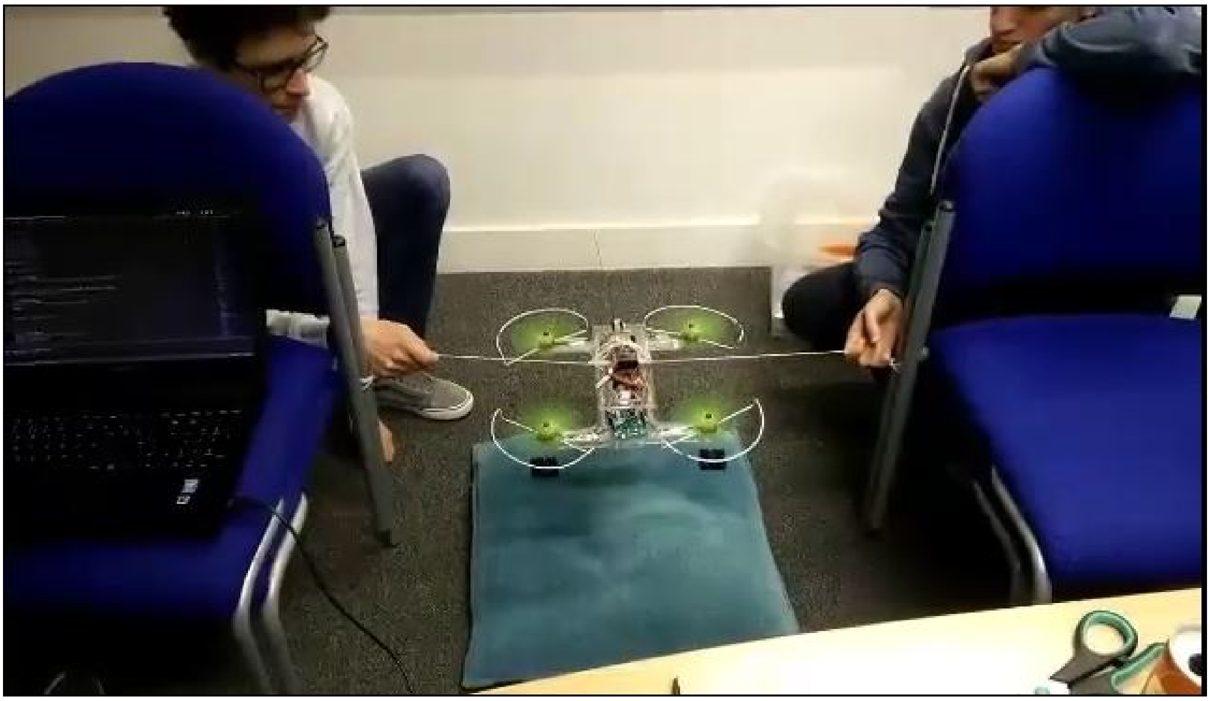


Figure 5: Initial Balance Test Set-up

This Set-up tests the Pitch control and is configured to have the centre of mass below the axis of rotation.

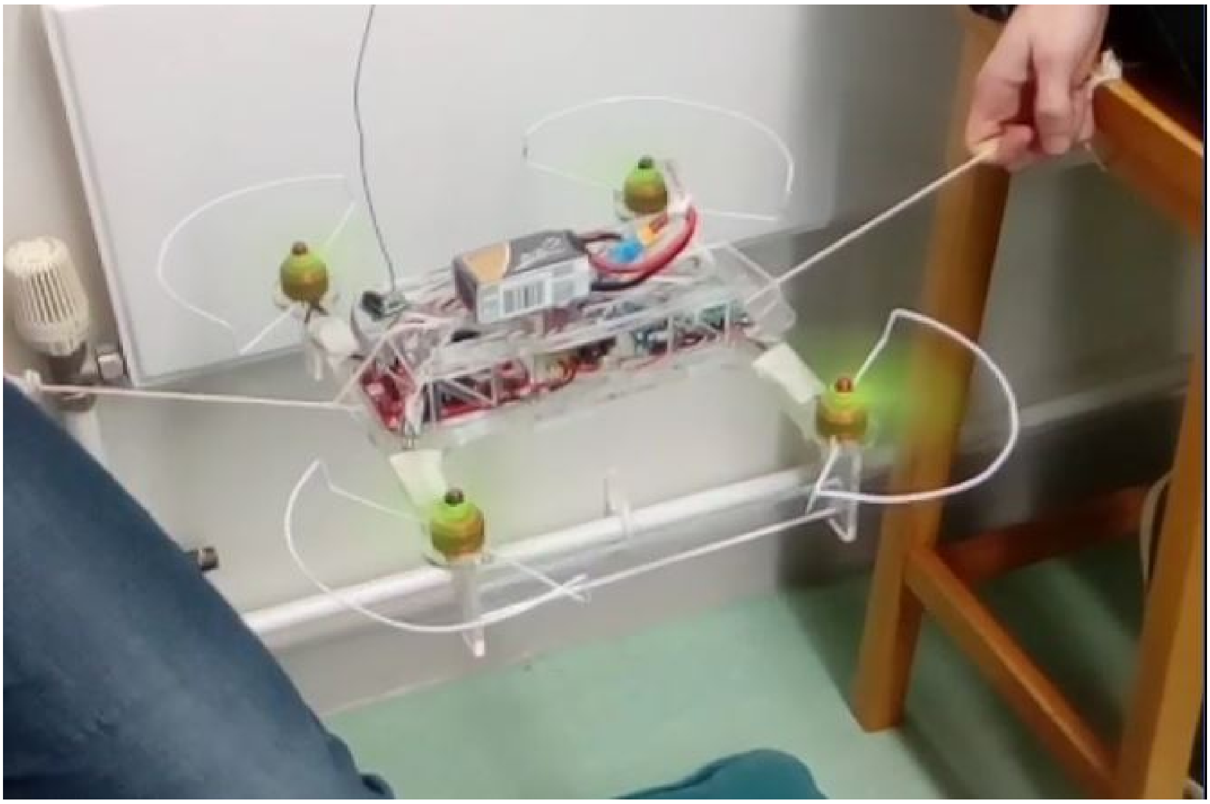


Figure 6: Final Balance Test Set-up

This Set-up tests the Roll control and is configured to have the centre of mass above the axis of rotation.