Investigative approach for the development of an augmented reality experience using an FPV multi-rotor.

Gary Sangwell

SAN12366949

BSc. (Hons) Games Computing

School of Computer Science

University of Lincoln

Supervisor: **Dr John C Murray**

Acknowledgements

Without the continued commitment and support of others, this project would not have nearly been as successful as it was, and so I would like to extend my thanks to all those involved.

Dr. John C Murray – For both your time and continued devotion throughout, without your support (and your kit!) this project wouldn’t be what it is.

Mr Ashley Williamson – For your moral support throughout this project, both in the fun times and the hard times, and for the continued supply of caffeine and sanity.

Miss Holly Moscrop – For your continued support throughout the project, your dedication and commitment in the stressful times and for dealing with all my late nights at the labs working on this project.

Friends and Family – For all your support, encouragement and enthusiasm, both throughout this project and my time at University.

Abstract

Multi-rotors have been increasing in popularity recently, in both commercial and consumer areas. Recent advances in virtual reality technology have made new applications of virtual reality possible, and opened up a range of new possibilities for the application of augmented reality. Virtual reality, augmented reality and multi-rotor technology have all been employed here, in order to investigate the development of an augmented reality first person view experience. A first person view multi-rotor platform has been developed, along with accompanying software for required communication and video transmission. An initial prototype of an augmented reality experience is presented. Future work would iterate on the prototype, and then use this to evaluate the user's experience, in order to gauge the value of augmented reality in FPV multi-rotors.

Table of Contents

Table of Figures

1 Project Background

The purpose of this section is to introduce the chosen project, briefly explain why this is an interesting area for study, and provide the rationale and motivation for the project.

1.1 Introduction and Motivation

Virtual Reality (VR) is an artificial environment presented to the user through the use of hardware, such as head mounted displays (HMDs), in such a way that they suspend belief and accept it as if it were a real environment. Ludow (2015) discusses the emerging applications and directions for the use of virtual reality, in which he states that 'technologies are used to create a 3D environment which the users experience through sensory perception, physical movement and communication' (Ludlow, 2015). During his discussion, Ludow discusses how the VR can be applied to a number of applications for use in education and simulation. For example, Jayaram *et al.* (1997) presented a research effort aimed at creating a virtual assembly design environment, using VR to assist designers in creating designs. They discuss how VR was valuable in assisting the designer, and how full implementations could significantly improve the design process; this emphasises the wide variety of applications for VR, and how it can be used in a context outside of traditional use.

Augmented Reality (AR) is integration of digital information into the user's environment in real time, in such a way that a composite view of both the real and the virtual is created. Azuma *et al.* (2001) provide a discussion on the recent advances in AR, in which they first define the 'basic goal' of using AR; to enhance the user's perception of and interaction with the real world, through the supplement of 3D virtual objects that appear to exist in the same space as the real world. During their discussion, the authors highlight current technology for creating AR applications, as well as a wide variety of use cases for both entertainment as well as serious uses such as education and collaboration.

Whilst still only in its infancy, VR is new a technology moving forward at an increasing fast pace, and both hardware and software for the creation of VR is improving continuously. In addition to this, the use of VR technology has led to the increase of development in the area of AR, opening the door for a whole host of new possible applications previously not possible. This wide variety of possible applications makes both VR and AR technology an interesting area of study.

Multi-rotors have come a long way since their early conception, with both commercially available consumer and DIY platforms now affordable at a wide variety of price ranges. Multi-rotors are now not only used in the commercial sector, but there is also an increasing area of interest around hobbyist multi-rotor applications. This has driven the development of the technology, and led to the increase of affordable platforms, both for entertainment, commercial and research, equipped with a wide variety of sensors for all kinds of applications. As a result of this, new sports and experiences using these multi-rotors have been conceived; one well known and emerging example is first person view (FPV) multi-rotor racing.

As discussed above, the application of VR and AR to other areas has proven successful in the past, not only increasing the value and immersion of the experience for users, but also as a platform for improvement. This drives the idea that AR and VR could be used to further the experience and applications possible with multi-rotors, opening new avenues for exploration that wouldn't otherwise be possible. This would be beneficial for both entertainment, whereby VR and AR could be used to create new immersive games or experiences using a multi-rotor, but also serious applications such as in the areas of aerial search and rescue or simulation and training.

2 Academic Literature and Background Research

2.1 Literature Review

Both multi-rotors, virtual reality and augmented reality are becoming increasing popular, due to an increasing audience and therefore drive in development, making them interesting areas for research. Whilst little in the way of research has been undertaken using both augmented reality and multi-rotors, this section attempts to identify similar and relevant literature to the project being undertaken here.

Ikeuchi *et al.* (2014) explored the idea of using a multi-rotor to create an interactive experience for users, with the aim to enhance the somatic sensation to fly in the sky. In order to achieve this, they used the AR Drone, the Microsoft Kinect and the Oculus Rift HMD. Video is sent from the AR Drone and displayed in the HMD, whilst arm movements are tracked using the Kinect, allowing the user to use natural gestures to control the AR Drone as if they were actually flying in the sky. The authors discuss how the integration of the virtual and physical world is a popular research topic, and how the use of this significantly increased the immersive experience; this reinforces the idea that the use of AR can be used to improve a user's experience.

Thon *et al.* (2013) explored the concept of using AR and a multi-rotor, in order to develop a serious game for use within the confines of a museum; this allowed visitors to explore the museums inner courtyard whilst it was closed for renovation. Visitors controlled the drone, the AR Drone by Parrot, from outside the museum, in an aim to move around and shoot the targets placed around the Forum. A short informational video challenging common beliefs about the museum was displayed each time a target was shot. Whilst the aim of the research was to allow the museum to communicate on its future renovation, the authors discuss how all ages of players were attracted to the game, and how in future work the game will be used to evaluate how the players' experience of the museum changed through the use of AR. This shows that the uses of AR can appeal to a wide range of ages and audience, and how this use of AR and a multi-rotor can be an appealing area of research, not only for developing serious applications, but also for assessing user experience in research.

Wen and Kang (2014) presented their ongoing work for using an Unmanned Aerial Vehicle (UAV) in construction scenarios, along with augmented reality to enhance the application of the UAV. They discuss how real images from the UAV are integrated with virtual scenes to provide engineers with augmented views to the construction site, and how this will increase the possibility of discovering unfound problems. They state how the proposed system aided engineers in visualising the field environment, and how the application of AR creates new possibilities for the planning process, not normally possible from other existing methods. Future work is discussed, mentioning how the UAV-AR solution could be applied in other areas of construction, and how the application of this would potentially benefit these areas. This research helps to highlight the value of the integration of AR and multi-rotor technology, showing how it can improve a user's experience in a wide variety of applications.

Ramasubramanian (2015) explores futuristic applications in environmental sensing by integrating cross-reality with semi-autonomous vehicles, and the cross-reality principles of augmented reality are enabled through the use of an unmanned-aerial-vehicle (UAV). The work presents a system that was developed to track and move a UAV in the physical world, with goals of using this for sensing and visualisation. During the discussion of the work, the author demonstrates how augmented reality is applied to environmental sensing, being used to provide the pilot with visual information within their view. This again demonstrates the possibility of using both augmented reality and multi-rotors for creating solutions not otherwise possible, for applications outside of the norm for this technology.

Bondyra *et al.* (2015) present a custom built multi-rotor known as the Falcon V5, designed as a versatile research platform, based on their previous work developing the Hornet robot. The authors highlight some of the major disadvantages of the platform, such as flight time and lifting capacity, and how the application of a custom design can be used to overcome these issues, through the tailoring of design to the application. In their discussion, they mention how modern 3D printing techniques enable the design of a custom multi-rotor platform, and how these techniques were able to produce replacement parts rapidly with low cost. They also mention how 3D printing can be used to adjust the strength of the platform, through the variation of infill amount, meaning weight versus strength can be balanced as required. This research shows the possible applications for 3D printing in research similar to that being undertaken, and how this technique can be used to develop hardware to compliment the research. The use of 3D printing allows for quick and iterative development of producible parts, allowing research to progress much quicker than if 3D printing was not available.

Krajnik *et al.* (2011) discuss the use of the AR Drone quadrotor as a suitable platform usable for both research and education. In their discussion of the platform, they describe both the hardware and software specification, as well as several issues with equipment, ability and performance of the drone. The authors also describe experiments in which the drone has been used, and introduce a freely available software package useful for overcoming initial problems, to further demonstrate its suitability for use in research. This discussion shows how a common multi-rotor platform is used in multiple areas of research, and demonstrates why such a platform is ideal for investigation. This furthers the idea that a multi-rotor platform is a worthwhile platform for investigation in the project.

Abeywardena *at al*. (2014) present the design and development of an open source multi-rotor platform for research. During their discussion of the design and development, the authors identify that whilst commercial, off-the-shelf solutions are fast to setup and provide a ready-made platform, they have limited extensibility for allowing researchers to modify them for scientific work. The authors evaluate several commercially available platforms, in terms of openness, extensibility and performance, before proposing a custom multi-rotor platform specifically geared towards research. This research demonstrates how multi-rotors are increasingly used in research and development, and how issues with commercially available platforms can be overcome through the development of a custom platform; concerns that are important to consider with this project.

3 Aims and Objectives

3.1 Aims

The aim of this project is to carry out investigative development of a hardware and software solution, providing an augmented reality FPV experience using an FPV multi-rotor and VR technology.

3.2 Objectives

1. Acquisition and evaluation of a selection of multi-rotors, to determine suitability for the project in terms of hardware requirements.
    a. A form factor that is suitable for the project is required; changes in development are likely to occur and so the chosen platform needs to suitable to accommodate these changes.
    b. The multi-rotor needs to be capable of having various hardware components mounted, and therefore needs enough thrust to fly with this additional weight in a responsive and safe manner.
2. Acquisition and familiarisation of suitable VR equipment and associated SDKs.
    a. Interfacing with the VR equipment in order to display video and read any IMU data from it correctly.
    b. Experimenting with the creation of virtual objects in VR space.
    c. Experimenting with augmentation in VR.
3. Augmentation of virtual objects into the video feed.
    a. Augment basic objects into the video feed to ensure that the hardware is capable of doing so in real time suitable for use during flight.
    b. Consideration of both on board and off board processing.
4. Hardware and software for communicate with the flight controller on-board the multi-rotor.
    a. Experimenting with various input devices for software control of a multi-rotor.
    b. Reading data from the flight-controller such as telemetry information.
    c. An appropriate input method will be chosen to create a familiar and intuitive control method.
5. Coordinate system localisation of IMU telemetry data and VR space.
    a. Crucial for ensuring that the virtual world lines up correctly with the real world and that virtual objects are shown correctly on screen based on where they should be in the real world.
    b. Will require investigation into localisation techniques suitable for a multi-rotor, such as GPS, dead reckoning or Visual Odometry.
6. Implementation of an interactive experience.

   a. A playable experience that the pilot can interact with by using the multi-rotor in an open space.
   b. Emphasis on creating an experience that would not normally be possible without using VR.

Part way into the project it became clear that due to issues that had arose with the completion of certain objectives, and slippage of the project, some objectives were not possible to complete in the given timeframe. These objectives are detailed below.

1. Carry out the first user study to collate usability data on traditional use of multi-rotors in the context of immersion and enjoyment.
   a. Will provide usability data that can then be compared to usability data collected in the second user study, in order to make a reflective comparison.
2. Carry out the second user study to collate usability data on the use of augmented reality in multi-rotors, in terms of immersion and enjoyment.
   a. Critical assessment of augmented reality in multi-rotors allows for the review of augmented reality as a solution to improve immersion and enjoyment.

As the primary aim of the project was to develop a hardware and software solution for an augmented reality experience, it was decided that the time available would be better suited developing the solution rather than carrying out user studies. This is also partly down to the fact that in order to carry out user studies, and evaluate the solution in terms of immersion and enjoyment, the solution would first need to be fully developed.

4 Methodology

4.1 Project Management and Software Methodology

Project management methodologies, such as PRINCE2 or Agile methodologies, help to provide structure and methods for more effective project management, in order to maximise the chance of success during project development. Whilst several well-known project management methodologies exist, most of these focus on projects being undertaken by larger teams, rather than a sole individual. For this reason, most of the project management methodologies that are discussed here would be unsuitable to apply in their entirety to this project, and therefore choosing some individual aspects from these methodologies that better suit and compliment the development of this project was more appropriate. As Charvat (2003) states, adopting an incorrect methodology can be detrimental to the progression of a project, leading to issues such as project management burnout, or time and cost slippage.

Agile methodology is an alternative to traditional project management methodology, which is often used within software development. Unlike more rigid methodologies, Agile allows teams to react and respond to unpredictable changes that arise during project development, through an incremental and iterative approach. Karlesky and Vander Voord (2008) state that Agile project management offers solutions to common and persistent problems with traditional project management, summarising with how Agile is well equipped to aid in managing risk, scope, budgets and schedules. This is further evident through twelve principles behind the agile methodology, stated in the Agile Manifesto (Agilemanifesto.org, 2016). Due to the nature of this project, an Agile development methodology was used to allow for flexibility in project development.

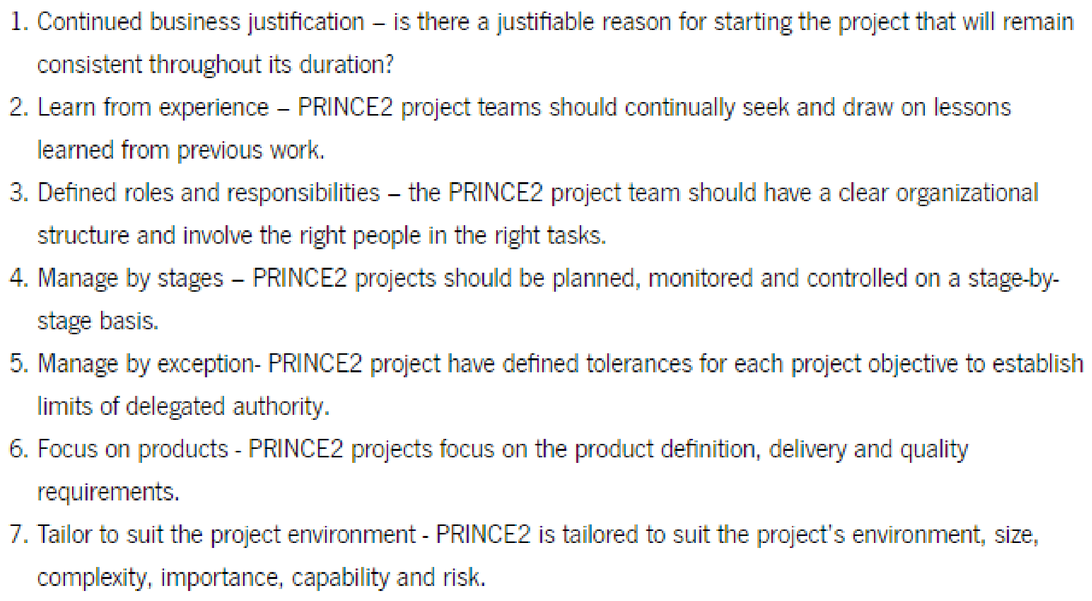
Extreme Programming (XP) is a software development methodology, designed to improve software quality and responsiveness to changing requirements. XP is inherently an agile software development methodology, and development is done in short cycles advocating frequent releases. Lists of 12 practices are used to help guide development, some of which were used during the development of this project. Figure 1 details what practices from the XP methodology were used.

- Simple design – the best design is the easiest one that works.
- Refactoring – Code should be continuously refactored.
- Coding standards – All code must be written in the same style using the same formatting.

Figure 1 - XP practices used during project development

SCRUM is an agile methodology that is commonly applied to software development, suiting projects with changing or emerging requirements. Development is performed through a series of small iterations called sprints, in which a small subset of requirements are implemented and tested. As the development undertaken in this project is by a single individual rather than a team, the methodology is not applicable in its entirety, and some of the processes involved in SCRUM wouldn't be necessary. However despite this, the overall idea of the SCRUM methodology is applicable to this project.

PRINCE2 (AXELOS, 2016), an acronym short for Projects in Controlled Environments, is a process-based method for effective project management. PRINCE2 is a widely recognised and utilised project management methodology both in the private sector, as well as internationally and often considered the de facto standard for project management methodologies. The key features of the methodology are outlined below in Figure 2.



1. Continued business justification – is there a justifiable reason for starting the project that will remain consistent throughout its duration?
2. Learn from experience – PRINCE2 project teams should continually seek and draw on lessons learned from previous work.
3. Defined roles and responsibilities – the PRINCE2 project team should have a clear organizational structure and involve the right people in the right tasks.
4. Manage by stages – PRINCE2 projects should be planned, monitored and controlled on a stage-by-stage basis.
5. Manage by exception- PRINCE2 project have defined tolerances for each project objective to establish limits of delegated authority.
6. Focus on products - PRINCE2 projects focus on the product definition, delivery and quality requirements.
7. Tailor to suit the project environment - PRINCE2 is tailored to suit the project's environment, size, complexity, importance, capability and risk.

Figure 2 - Key features of the PRINCE2 project management methodology (AXELOS, 2016)

Whilst being very process-based in its method, PRINCE2 is a product-driven methodology, focusing on producing deliverables and adjusting to changes in requirements at a level appropriate to the project. As with other methodologies discussed, PRINCE2 is not applicable to this project in its entirety, but some of the key features behind its process can be used to guide development; these include dividing the project into manageable stages, a focus on both the product quality and delivery and the idea of tailoring the methodology to suit the project environment.

Al-Zoabi (2008) investigated the effect of applying PRINCE2 on XP projects, discussing how the agility and flexibility in agile methodologies may badly affect project development. The objective was to extract the best out of both XP and PRINCE2, through the combination of principles from both methodologies; taking advantage of XP's flexibility and adaptability with PRINCE2's value of control.

The author concludes with how the combination was used successfully in real life projects, to deliver systems both on time and within budget, along with key lessons learnt from the experience. During the development of this project, an approach similar to the above was taken; the practices and principles discussed above, from both PRINCE2 as well as agile methodologies, were combined in order to better support development, and maximise the chance of project success.

4.2 Risk Matrix

Assessing the possible risks associated with a project is important to consider in order to ensure that any problems or issues that could arise during the project, are handled in an appropriate manner, thus minimising their potential impact on the project. Datta and Mukherjee (2001) discuss the development of a risk management matrix, in order to facilitate better project planning. Through their proposal and testing of a methodology suitable for identifying and quantifying risks at an early stage, they identify how the success of a project, in terms of budgeted time and cost, largely depends on the early identification of immediate risks to the project. One method that can be used for assessing project risks is a risk matrix, a 'simple, easy to use, structure process that identifies the risks that are most critical to the program' (Lansdowne, Z. F., 1999). Originally devised by the acquisition reengineering team at the Air Force Electronics System Center, the use of a risk matrix allows for the early identification of project risks, and their subsequent quantification of both likelihood and project impact.

4.2.1 Initial Risk Matrix

Before the project began, a comprehensive risk matrix was drawn up detailing the risks associated with the project, along with quantification for their likelihood and impact. This matrix also includes a formal response column, which details how to prevent and mitigate each potential risk, as well as how to acknowledge the risk has occurred. In order to quantify both the risk likelihood as well as the risk impact, a very low to very high scale has been used.

Risk	Likelihood	Impact	Cause	Response
Participant is hurt during testing.	Very Low: Steps will be taken to ensure the safety of participants at all times.	Very High	A collision with the multi-rotor and a participant during testing could result in injury.	Prevention: Isolated operation with physical barriers separating the participant and the multi-rotor at all times during operation. Only participants with experience using multi-rotors will be chosen. Fly only within line of sight and with the project research present at all times.

				Mitigation: Be aware of how to seek medical assistance if needed. Use suitable size multi-rotor as larger multi-rotors increase risk, due to large form factor, propellers and weight. Acknowledgement: Respond to any injury to the participant immediately, and take the appropriate course of action to inform the required members of staff.
Multi-rotor propellers break.	Medium: Propellers are made from fragile material and spin at high RPM, therefore any collision with the propeller and the environment could cause breakage.	Low	Fragile propellers. Any collision between surfaces or other objects could cause multi-rotor propellers to break.	Prevention: Ensure the multi-rotor is only used in a safe and hazard free environment. Only operate within line of sight. Handle with care. Mitigation: Have an adequate supply of spare propellers, purchasing more when running low so an unexpected break will not halt project progress. Visually inspect propellers prior to each flight to check for damage or imperfections that could

				result in a break mid-flight. Acknowledgement: Replace any broken propellers and inspect for any additional damage.
Other multi-rotor components or equipment in use breaks.	Low	Low to Medium	Multiple impacts or heavy impacts could result in damage to more expensive components or other equipment on board.	Prevention: Only use the multi-rotor within line of sight and away from any potential hazard. Ensure adequate design to protect any delicate components. Use legs to raise the body of the multi-rotor from the ground to lessen impacts. Mitigation: Obtain spare or replacement equipment where possible, or tools to repair damage equipment. Research alternative equipment if spare or replacement is not possible. Acknowledgement: Repair the equipment or replace if possible. If repair or replacement is not possible then use alternative components or equipment. For example, if

				the multi-rotor is broken beyond repair, the AR Drone could be used instead.
Damage to other property.	Low	Low	Unexpected behaviour of the multi-rotor causing impact with other property.	Prevention: Ensure multi-rotor is only used in a clear environment and away from other property where possible. Mitigation: Use the multi-rotor sensibly and at suitable speeds when around other property. Acknowledgement: Report any damage for assessment.
Project delays.	Low	Low to Medium	Part of the project could take longer than expected. Damage to components or equipment.	Prevention: Clear project plan, with adequate time allocated to each object in the project. Mitigation: Reassess project plan. Acknowledgement: Continue with the project following the revised plan.
Equipment needed is unavailable.	Low	High	Unable to source equipment for	Prevention: Adequate research to ensure any equipment is

			project due to lack of stock or price.	available and affordable. Mitigation: Research readily available and affordable alternatives. Acknowledgements: Use a cheaper or more readily available alternative.
Loss of data.	Low	High	File corruption of either source code or project report. Storage medium is lost, stolen or broken.	Prevention: Use versioning tools such as Git to maintain source code. Use of Cloud based storage for documents. Ensure backups of all files are taken frequently. Mitigation: Keep storage medium safe and secure. Small iterative changes to minimise loss. Acknowledgement: Recover files from a previous back to minimise loss.
Incompatible SDKs.	Low	Medium	SDK is not compatible with other software. Poorly written SDK causes issues with	Prevention: Initial research into the hardware chosen to ensure the SDK will be compatible.

			other software.	Mitigation: Use of the adapter pattern to create a compatible interface for the SDK. Acknowledgement: Use other hardware with a more suitable SDK.
Adverse health and safety effects such as motion sickness or migraines caused by VR equipment during development.	Low	Medium	Prolonged use of VR equipment.	Prevention: Plan development to minimise use of VR equipment. Take regular breaks during use. Mitigation: Take a break from development or work on another part of the project if any effects start to show. Acknowledgement: Work on another part of the project that does not involve the VR equipment.
EA2 ethics approval, for involvement of participants during evaluation, is denied.	Low: Ethics forms will be prepared and all risk considered and mitigated.	High	Ethics committee feels inadequate steps have been taken to prevent risk to participants.	Prevention: Fully prepare ethics form and consider all risk to participants, mitigating according. Mitigation: Re-evaluate risk to participants and migration

				of those risks for resubmission. Acknowledgement: Continue with project without participants for testing and evaluation of solution.

4.2.2 Additional Risks

Shortly after the risk matrix was complete, it became clear that some risks to the project hadn't been identified; this risks are shown in the risk matrix below, following the same format as the first risk matrix. The combination of these two risk matrices show the full set of project risks considered prior to the undertaking of this project.

Risk	Likelihood	Impact	Cause	Response
Issues with the localisation of the multi-rotor.	Medium: Whilst initial research suggested that localisation is possible, issues could still arise if methods don't work as well as they are presented.	Very High	Method used for localisation is either unfeasible due to hardware constraints, or the method does not perform as originally expected.	Prevention: Consider the options for localisation carefully before implementation. Mitigation: Consideration of other possible methods or solutions that are available. Reconsideration of the approach taken in order to implement AR such as markers. Acknowledgement: Use a different method of localisation that is feasible, or use a different

				approach to AR if no other methods are feasible.
Issues acquiring a suitable multi-rotor platform for the project	Low	High	As the project as a list of requirements for the platform, available platforms might not be suitable for use.	Prevention: Adequate research into available platforms that would be suitable. Mitigation: Investigate other alternative directions for the project, if a platform is not available. For example, the concept could be developed without the platform. Acknowledgement: Approach the project in a different direction that does not require the platform.
Illness	Low	Medium – High	Illness could be random and unexpected, or work induced if over worked and breaks are not taken.	Prevention: Proactive about health and observant of any signs of illness. Mitigate: Seek medical help if needed, and ensure that adequate rest is taken until the illness passes. Acknowledgement: Stop development until

				the illness passes, or continue only with parts that will not worsen or prolong the illness.
Low motivation	Low: High level of interest and passion in the project should mean motivation is not an issue.	High	Signification issues or problems experienced during the project could result in low motivation, especially if the project falls far behind.	Prevention: Be aware of how the project is progressing, and have alternative solutions for if things fail. Mitigate: Take a short break from the project, before reassessing the project as a whole, in order to determine how to move on. Acknowledgement: Continue with the project as best as possible, seeking guidance from supervisor if possible.

4.3 Tools

4.3.1 Version Control System

For managing source code and project files throughout the development of the project, a version control system (VCS) was required. Many version control systems are available, including Git and SVN. Git (Git-scm.com, 2016) allows for a complete history of source code to be maintained throughout development, and for the development to be structured through the use of feature branches and commits. As the author had previous experience with using these features of Git to better structure project development, Git was chosen over other available version control systems.

Whilst traditionally being used to develop new features separately from the main code base, prior to being merged back onto the master branch, feature branches were also used to allow completely different parts of the project to be worked on independently and in parallel, allowing for more focussed development and testing. One example of this is during the development of the MSP protocol implementation; this was developed on a separate branch from the main code base, and later merged back into the development branch when complete.

GitHub (GitHub, 2016), a freely available remote repository host for Git, was the platform chosen in order to maintain all source code developed during the project. This platform was chosen due to existing familiarity with the platform from prior work and experience, and because it offered remote hosting of local git repositories, making the repositories available online. This was something which was advantageous when developing from both Home and University.

4.3.2 Cloud Storage

4.3.2.1 Dropbox

Dropbox (Dropbox, 2016) is a personal cloud storage service, frequently used for the online storage, backup and sharing of multimedia. This service was used throughout the development of the artefact for the project in order to store photos and videos documenting the development process, as seen throughout this document. Dropbox was chosen over other cloud storage solutions due to its wide variety of supported platforms, allowing for seamless integration between mobile, desktop and online devices. This saved time when documenting the development with pictures and videos, which could be uploaded straight from the mobile device they were taken on. As the media is stored in the cloud, development could be done from multiple locations without having to worry about development evidence being fragmented between multiple devices and locations.

4.3.2.2 One Drive

OneDrive (Onedrive.live.com, 2016) is an online file hosting service, which allows users to store files and then sync them across multiple devices and platforms. Integration into applications such as Microsoft Word and Microsoft Excel, mean documents can be worked on just as if they were local to the application, and any changes made are automatically saved to the cloud and mirrored across all devices seamlessly. This was highly beneficial during the documentation of the project, as it allowed the document to be worked on at both University and Home without requiring to manually keep changes made in sync. In addition to this, the services also doubled as a backup for the project document, ensuring that an up-to-date copy was always available in the result of data loss.

4.3.3 Integrated Development Environments

Several Integrated Development Environments (IDE) were utilised during the development of the project throughout its lifetime, due to extensive features provided for assisting in code development. As multiple different libraries were required to develop the project, many of which the author had no prior experience with, automatic code completion and definition peek were invaluable features decreasing the learning curve of these libraries significantly, and thereby reducing the need to lookup documentation for the library as often. Extensive syntax highlighting allowed for easier visual inspection of source code, by presenting cleaner and more structured code, something which is more important to consider when developing in C++ versus other languages that are less complex.

Additionally, the tools provided for runtime debugging helped to assist in development, often allowing bugs or errors to be identified quicker than if an integrated debugger had not been available. One key feature provided by integrated debuggers, that was heavily used, were breakpoints; these allowed for runtime inspection of variable values and stepping through program execution in order to identify program flow. This made debugging program errors quicker and easier, as code execution could be traced from before any exception was raised during runtime. This proved especially useful when debugging the high definition (HD) video transmission system due to FEC encoding algorithm complexity making standard forms of inspection less effective.

Integrated compilers simplify the process of building and linking programs, and instead use project and solution files to setup the build configuration. This automates the build process for compiling applications, which is particularly beneficial for large and complex applications which link to numerous libraries; requiring the inclusion of a significant list of compile-time flags. Multiple build configurations can also be setup and swapped between with minimal effort, such as Debug and Release, as well as processor architecture. The use of multiple build configurations allowed a more optimized Release configuration to be used during performance testing, and Debug modes with all symbols to be used during development debugging as outlined in debug methods above.

4.3.3.1 Visual Studio

Visual Studio 2015 (Visualstudio.com, 2016) (VS2015) was chosen as the main development environment due to the extensive features offered by the IDE, as discussed above, as well as the fact the IDE is considered the de facto standard for Windows development environments.

4.3.3.2 NetBeans

During development of source code for the Raspberry Pi, NetBeans (Netbeans.org, 2016) was employed due to the built in support for remote compilation, which resulted in a much smoother and faster development process for the Raspberry Pi hardware, reducing time taken to develop for the hardware. The NetBeans IDE was run on a full desktop PC rather than the Raspberry Pi itself, and as a result the graphical and processing limitations did not introduce difficulties or slow development. Source code was uploaded automatically and compiled on the Raspberry Pi seamlessly, and debugging abilities were still available just as if the code was being executed locally. For this reason alone, NetBeans stood out above all other common IDEs when developing for the Raspberry Pi.

4.3.4 Software Development Kits and Libraries

Throughout development of the project, several different libraries and SDKs were used to provide needed functionality for different parts of the project.

4.3.4.1 Oculus Rift SDK

The open source Oculus Rift SDK (Developer.oculus.com, 2016) was required in order to interface with the Oculus Rift Head Mounted Display (HMD), which is currently only available in beta release for the Windows operating system (OS) due to the drop of support for Linux and Mac platforms by Oculus. Whilst attempts have been made to continue developing the SDK for the Linux platform, it is less developed and maintained than the official Oculus SDK, due to the restrictions on budget, developers and time that is inherent with open source development. In addition, the newer SDK available not only provides better features, but also sports a much more refined interface which allows for cleaner and more appropriate architecture when developing application for the Rift. For these reasons, the newer version of the SDK available on Windows was chosen.

4.3.4.2 Libpcap, Winpcap and Airpcap libraries

One objective of the project was investigation into suitable medium for video transmission from the aerial platform to the ground station, which would later be required to augment virtual objects into. Initially analogue video transmission equipment was used, however limitations with analogue video formats, specifically low resolution and low frame rate, prompted investigation into other methods that would provide a better experience for the user. Libpcap (TCPDUMP, 2016) is a system independent interface for the transmission and capture of networking packets, which was required to implement the Wi-Fi based video transmission system. Libpcap was chosen over other possible alternatives due to the cross platform development required; a Raspberry Pi B+ was used on-board the multi-rotor to capture, process and send the video, whereas the ground station was running Windows in order to support the Oculus Rift. By using an interface that was implemented cross platform, development time was reduced as much of the same code was used both for transmission as well as receiving. The Winpcap (Winpcap.org, 2016) library was the Windows implementation of the Libpcap interface that was used for receiving side. In addition, the Windows implementation for Libpcap had a lack of support for an unassociated mode of wireless transmission known as monitor mode, and as a result a third library was required. The Airpcap (Support.riverbed.com, 2016) library was chosen as a result, due to the fact it was the only available solution that provided functionality for monitor mode on Windows.

4.3.4.3 Monocular Visual Odometry libraries

Various Monocular Visual Odometry libraries were investigated and considered for use during development of the project, as a possible solution to the issue of localising the aerial platform within the real world. One objective of the project was to localise the two coordinate spaces, the real world and the virtual world, in order to successfully augment virtual objects that appeared as if they were physical to the user. Initially, GPS was considered, but this was quickly dismissed due to the development environment being the literal definition of a GPS denied environment. Hardware constraints meant that a stereoscopic camera was not available, and therefore Monocular Visual Odometry algorithms had to be used instead of Stereoscopic Visual Odometry algorithms.

One such example considered was the Semi-direct Visual Odometry library (GitHub, 2016), which during initial research looked to provide a reasonable solution to the problem of localisation. In the paper discussing the algorithm implemented by the library, the authors discuss and compare the algorithm will the well-known PTAM algorithm, and conclude how it is 'particularly useful for state-estimation on-board MAVs' (Forster *et al.,* 2014) and how 'high frame-rate motion estimation, combined with an outlier resistant probabilistic mapping method, provides increased robustness in scenes of little, repetitive, and high-frequency texture.' (Forster *et al.,* 2014).

4.3.4.4 Aruco

As the project progressed, issues with the available implementations of Monocular Visual Odometry algorithms meant a different approach to augmented reality had to be used. During initial research into existing augmented reality applications, AR marker based libraries were identified as a possible solution for augmented virtual objects into the real world. This approach has proved to be popular with other AR applications, due to both its effectiveness and simplistic approach. Two popular implementations that are available include Aruco (GitHub, 2013), a minimal C++ library for the detection of augmented reality markers, and ARToolkit (Artoolkit.org, 2016), a cross platform SDK for the development of augmented reality applications.

The Aruco library was favoured for use in development due to being a minimalistic library, providing a simple C++ interface that could be implemented into the existing code with relative ease. This was important to consider as development of the project had already fell behind schedule at this point, due to the issues experienced with the Monocular Visual Odometry libraries. This meant that whilst other alternatives, including the ARToolkit, provided a more comprehensive set of features, the additional complexity of learning and implementing a new SDK would have resulted in additional delays.

4.3.4.5 CMake and CCMake

CMake (Cmake.org, 2016) is an open-source system that manages the build process through the use of simple configuration files, known as CMakeLists, placed within each source directory. CCMake (Cmake.org, 2016), a GUI based build configurator, allows for the easy configuration of these CMake files, simplifying the process for configuring larger projects and libraries. CMake files are automatically generated from the configuration set by the user, and can then be used with tools such as Make in order to build. During development on Linux and Raspbian, these tools were used in order to generate build files for code compilation. As some libraries used during this phase of development had to be built from source code, the use of these tools was essential.

4.3.5 Utilities

4.3.5.1 Cleanflight Configurator

The Cleanflight Configurator (GitHub, 2016) is a Google Chrome Application that is used for the configuration, setup and flashing of new firmware for Cleanflight based flight controllers. Currently, the Cleanflight Configurator is the recommended method for configuring these flight controllers, and as such its use was required during the development of the platform hardware. As the Cleanflight Configurator is a Google Chrome Application, it is cross platform and works on both Windows, Linux and Mac operating systems; this proved beneficial whenever development was being done on Linux as the hardware could tuned, tweaked or reconfigured without requiring a reboot into Windows. This is unlike other flight controller configuration tools, such as the older MultiWii tool, that was only available under Windows. The configurator also proved invaluable for debugging issues with the platform, such as incorrect control mappings, as well as for tuning the PID loops for stable flight.

4.3.6 Hardware

4.3.6.1 3D Printing and Thingiverse repository

3D printing was used throughout the development of the project, as the main method of producing parts for both the multi-rotor platforms. 3D printing was used to produce both frames at low cost, with the ability to quickly have parts reprinted if any modifications were needed. This allowed the development of the platform to move along quickly, as the parts could be modified and reprinted quickly, as well replacement or additional parts printed if necessary.

One example of this is an antenna mount that was printed part way into the build. Initially, it was not considered necessary, but during the construction of the platform it became clear there were limited mounting options available for the antennas. The benefits of 3D printing, as previously discussed in the literature review on Page XX, allowed for the flexibility for this change in design. In addition to this, 3D printers were available throughout the entirety of the project for printing any parts that were required for the hardware build.

The online repository, Thingiverse (Thingiverse.com, 2016), is a community driven platform for the sharing and development of 3D printable models. This provided a host of 3D printable designs to search through to identify existing models that were suitable for use in the project, ranging from full frames for the multi-rotor platform to small mounts for antenna aerials. As the platform is largely community driven, users can post comments and feedback, as well as pictures and discussion about their prints, often detailing issues or recommended print settings that help to improve the print quality. This helped to limit the number of issues with 3D printed parts, as potential issues could be identified and mitigated before any printing was even started.

4.3.6.2 eCalc.ch

eCalc (Ecalc.ch, 2016) is an online tool for calculating theoretical flight characteristics, such as flight time, thrust to weight ratio (TWR) and required throttle for hover, from the component specification entered by the user. This was utilised during the initial design stages of the 500 size multi-rotor platform, in order to ensure the components chosen would be adequate for flight, reducing time and material waste. Without this, time would have had to of been spent manually calculating these flight characteristics, such as thrust to weight ratios from motor and propeller data sheets, and then expected flight time calculated accordingly. The use of an online tool simplified this process, and allowed multiple variations of component configurations to be assessed quickly with minimal effort.

4.3.6.3 Blackmagicdesign Intensity

The Blackmagicdesign Intensity (Blackmagicdesign.com, 2016) is a USB 3.0 video capture and playback device, allowing the capture of both HDMI and analogue video directly, using dedicated built in hardware. This device was required during development in order to capture the analogue video provided by the multi-rotor, and provide it in a suitable format for use with OpenGL and the Oculus Rift. This particular device was chosen over other alternatives as it was both available from the department for use during the project, and was more than adequate in terms of specification due to being high end consumer grade hardware. In addition to this, an SDK was also available that provided support for graphics libraries, including OpenGL, reducing the complexity of integrating the device into the existing code.

4.3.6.4 Oculus Rift DK2

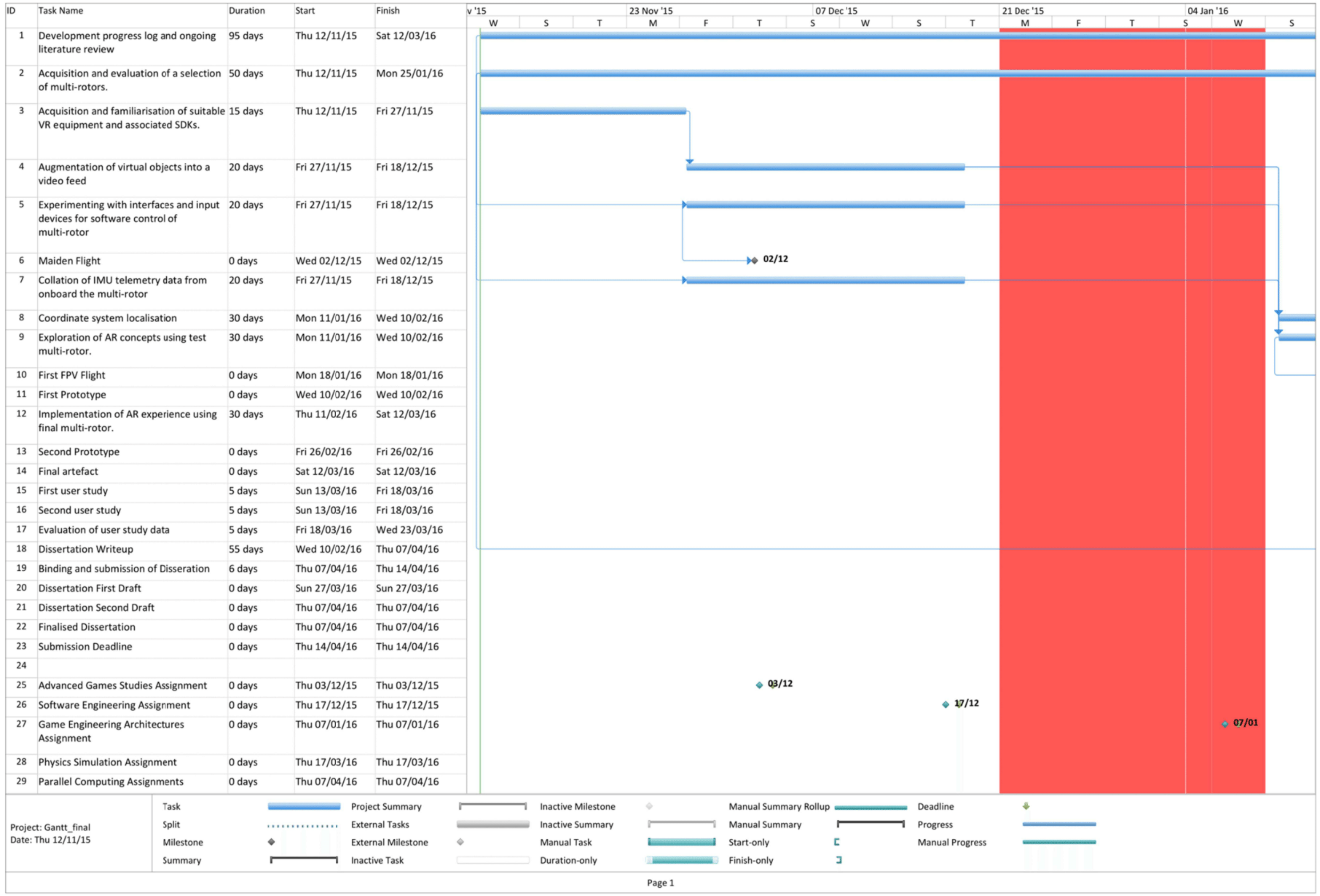
The Oculus Rift is a virtual reality head mounted display, developed and produced by Oculus VR, and recently released for consumer release. It was widely considered the de facto standard for VR at the time of this project, providing a better experience than some of the other more limited hardware alternatives, and therefore was the chosen VR hardware for this project. The Oculus Rift DK2 specifically was used in the project, over the original Oculus Rift DK1; whilst both were available for use during development, the DK2 provides a much superior experience, with not only a higher resolution display, but also better head tracking and hardware for decreasing motion sickness. This was important to consider, as the device was used at multiple stages in development, all of which could have been delayed or affected.

4.3.6.5 Raspberry Pi

The Raspberry Pi is a low cost, single board computer, primarily designed for use in education. The device is capable of running a wide variety of applications, including graphical user interfaces, and offers a plethora of input and output (IO) options for developing hardware projects. The Raspberry Pi was used in this project on-board the multi-rotor, for high definition video transmission, and was chosen over other possible alternatives, due to its low cost and availability, as well as its support for hardware accelerated H264 video encoding. The device is equipped with a Camera Serial Interface (CSI), which facilitates the connection of a small camera to the main on-board processor. This type of interface, and the support for hardware H264 encoding, was beneficial to the project as it allowed for high resolution video to be captured and encoded with low levels of latency, something which wasn’t possible on other devices that lacked these features.

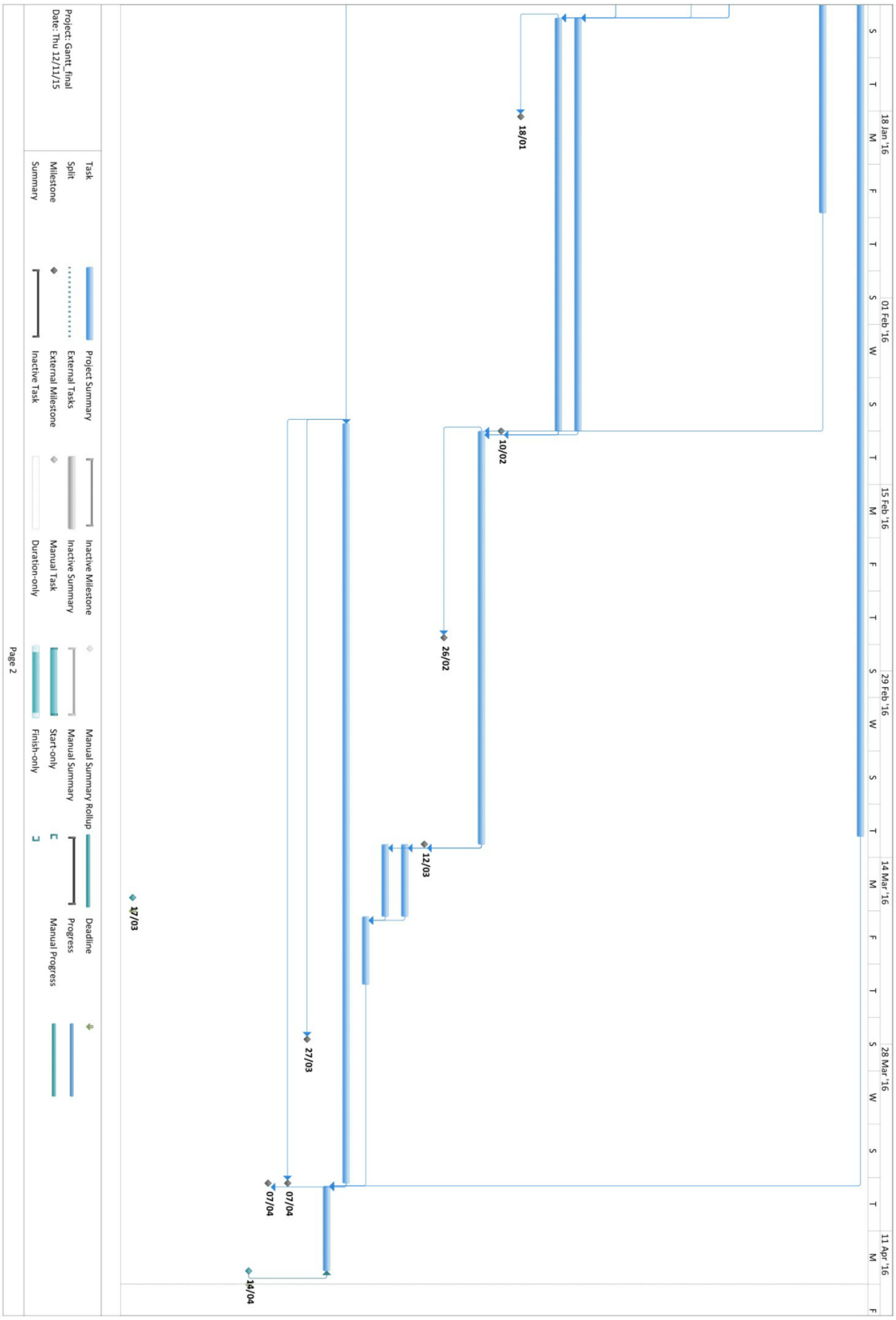
4.4 Gantt Chart

During the proposal for the project, an initial Gantt chart shown below was created. However, as the development followed an agile methodology, as discussed in section 4.1 Project Management and Software Methodology, the Gantt chart was not really applied. This is because the methodology chosen followed an iterative approach towards development, in order to accommodate the unforeseen changes anticipated during development. Karlesky and Vander Voord (2008) discuss the common issues with traditional project management, how agile project management offers solutions to these common and persistent problems, and how Gantt charts rarely reflect reality. This is reflective of the issues experienced in following the Gantt chart in this project; the agile methodology assisted in the progression of development in this project, which would have been hindered if the Gantt chart was followed rigorously. As a result of this, no further Gantt charts were created, and instead the chosen methodology was used to guide the project.



ID	Task Name	Duration	Start	Finish
1	Development progress log and ongoing literature review	95 days	Thu 12/11/15	Sat 12/03/16
2	Acquisition and evaluation of a selection of multi-rotors.	50 days	Thu 12/11/15	Mon 25/01/16
3	Acquisition and familiarisation of suitable VR equipment and associated SDKs.	15 days	Thu 12/11/15	Fri 27/11/15
4	Augmentation of virtual objects into a video feed	20 days	Fri 27/11/15	Fri 18/12/15
5	Experimenting with interfaces and input devices for software control of multi-rotor	20 days	Fri 27/11/15	Fri 18/12/15
6	Maiden Flight	0 days	Wed 02/12/15	Wed 02/12/15
7	Collation of IMU telemetry data from onboard the multi-rotor	20 days	Fri 27/11/15	Fri 18/12/15
8	Coordinate system localisation	30 days	Mon 11/01/16	Wed 10/02/16
9	Exploration of AR concepts using test multi-rotor.	30 days	Mon 11/01/16	Wed 10/02/16
10	First FPV Flight	0 days	Mon 18/01/16	Mon 18/01/16
11	First Prototype	0 days	Wed 10/02/16	Wed 10/02/16
12	Implementation of AR experience using final multi-rotor.	30 days	Thu 11/02/16	Sat 12/03/16
13	Second Prototype	0 days	Fri 26/02/16	Fri 26/02/16
14	Final artefact	0 days	Sat 12/03/16	Sat 12/03/16
15	First user study	5 days	Sun 13/03/16	Fri 18/03/16
16	Second user study	5 days	Sun 13/03/16	Fri 18/03/16
17	Evaluation of user study data	5 days	Fri 18/03/16	Wed 23/03/16
18	Dissertation Writeup	55 days	Wed 10/02/16	Thu 07/04/16
19	Binding and submission of Disseration	6 days	Thu 07/04/16	Thu 14/04/16
20	Dissertation First Draft	0 days	Sun 27/03/16	Sun 27/03/16
21	Dissertation Second Draft	0 days	Thu 07/04/16	Thu 07/04/16
22	Finalised Dissertation	0 days	Thu 07/04/16	Thu 07/04/16
23	Submission Deadline	0 days	Thu 14/04/16	Thu 14/04/16
24				
25	Advanced Games Studies Assignment	0 days	Thu 03/12/15	Thu 03/12/15
26	Software Engineering Assignment	0 days	Thu 17/12/15	Thu 17/12/15
27	Game Engineering Architectures Assignment	0 days	Thu 07/01/16	Thu 07/01/16
28	Physics Simulation Assignment	0 days	Thu 17/03/16	Thu 17/03/16
29	Parallel Computing Assignments	0 days	Thu 07/04/16	Thu 07/04/16

Project: Gantt_final
Date: Thu 12/11/15


18 Jan '16
01 Feb '16
15 Feb '16
29 Feb '16
14 Mar '16
28 Mar '16
11 Apr '16
S T M F T S W S T M F T S W S T M F T S W S T M F
18/01
10/02
26/02
12/03
17/03
27/03
07/04
07/04
14/04
Project: Gantt_final
Date: Thu 12/11/15
Task
Split
Milestone
Summary
Project Summary
External Tasks
External Milestone
Inactive Task
Inactive Milestone
Inactive Summary
Manual Task
Duration-only
Manual Summary Rollup
Manual Summary
Start-only
Finish-only
Deadline
Progress
Manual Progress
Page 2

5 Development of Artefact

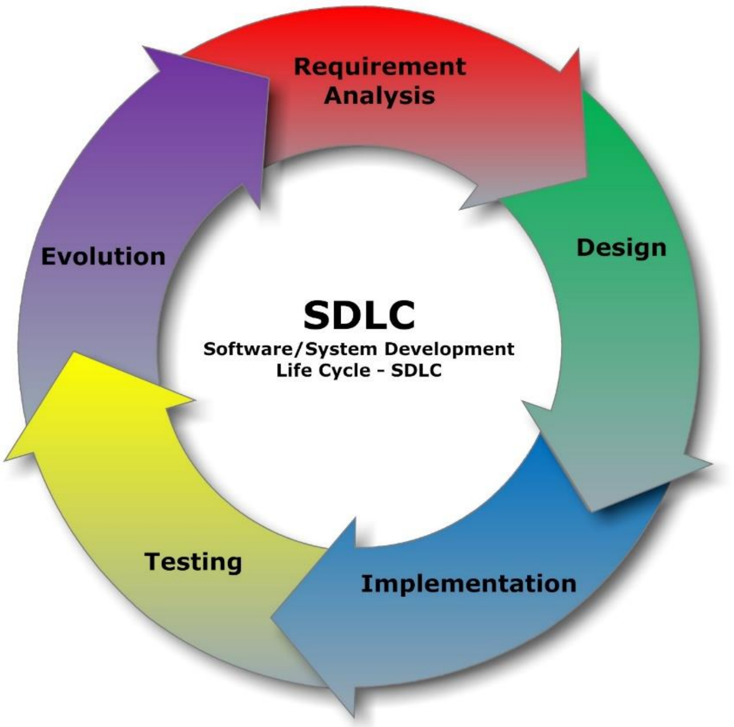


Figure 3 - Software Development Lifecycle (Stylusinc.com, 2016)

The software development lifecycle (SDLC) is a process used to structure the development of software, consisting of 5 key stages shown above in Figure 3. Ruparelia (2010), in their discussion of the main software development lifecycle models, highlights that the SDLC is a framework or process that considers the structure of the stages involved in the development of an application. As one of the requirements for this project was to undertake and document at least 2 stages of the SDLC, it is crucial that its role in supporting development is identified. As discussed previously in the section 4.1 Project Management and Software Methodology, this project followed a more agile approach, developing in iterations focusing on deliverables. During each of these iterations, the software development lifecycle was followed in order to structure development, and this section aims to show how this process was used and applied throughout.

5.1 Initial Research

Before any development of the artefact was started, some initial research was required to meet some of the objectives laid out in section 3.2 Objectives. This research included an investigation and evaluation of hardware for both the multi-rotor and VR platform.

5.1.1 Multi-rotor Platform

One of the initial objectives of the project was the investigation and evaluation of a selection of multi-rotor platforms, in order to determine suitability for the project in terms of hardware requirements. In order to meet this objective, a list of hardware requirements for the platform had to be drawn up, detailed below in Figure 4. In order to evaluate each platform, this list of requirements was used as a basis for their suitability; the evaluation involved the use of the multi-rotor, either directly by the author or through observation of its use. In addition to this, the technical specifications, including both the frame design, size and payload capacity, were considered in relation to these requirements.

- Affordable within the scope of the project.
- Adequate size and form factor for mounting hardware required.
- Stable when flying.
- Can be modified to react to changes in development.

Figure 4 - List of requirements for multi-rotor platform.

One platform that was investigated was the commercially available consumer DJI Phantom 2 (Dji.com, 2015). The Phantom 2 was however dismissed early on for several reasons. Firstly, the price point for a suitable version of the Phantom 2 was too high in relation to the scope of the project, coming in at around £500. Whilst cheaper versions are available, these did not include any options or hardware for mounting a camera, meaning additional design work would have been required. The gimbals that are included in the more expensive versions of the Phantom 2 are limited in terms of the cameras that they are able to use; the GoPro series of cameras is required. This again, would have either meant additional design work to modify the platform to fit different cameras, or additional expenditure. The form factor of the Phantom 2, whilst nice for consumers, does not lend itself to research based applications; the enclosed casing and design prevents easy modifications and repair, and limited mounting options for hardware could have hindered the projects progress.

Figure 5 - DJI Phantom 2 with H4-3D gimbal (Dji.com, 2015)

Along with the Phantom 2, DJI offer the Inspire 1 (Dji.com, 2016) multi-rotor, which was also investigated during initial research. This again was dismissed early on during investigation, for similar reasons to the Phantom 2. Whilst the Inspire 1 includes both a gimbal and a camera unlike the Phantom 2, the price point of the Inspire 1 is again out of the budget for the project, at around £2300.

Figure 6 - DJI Inpsire 1 (Dji.com, 2016)

After evaluating commercially available off-the-shelf solutions, and determining their unsuitability, investigation moved towards DIY platforms. One such platform looked at was the Team Black Sheep (TBS) Discovery (Team-blacksheep.com, 2016), a commercially available multi-rotor frame in kit form; additional components including a battery and motor system are required, all of which have recommended specifications given by Team Black Sheep, making the platform suitable as an entry level DIY kit. Whilst better suited than the previously discussed multi-rotor platforms, the TBS Discovery was also dismissed as a viable platform due to cost, and limitations with the form factor. Despite the fact the platform is a DIY kit, and can be modified for customisation to an extent, the integrated PCB and design of the frame is tailored towards the recommended build, and therefore could be limiting in terms of the requirements for this project.

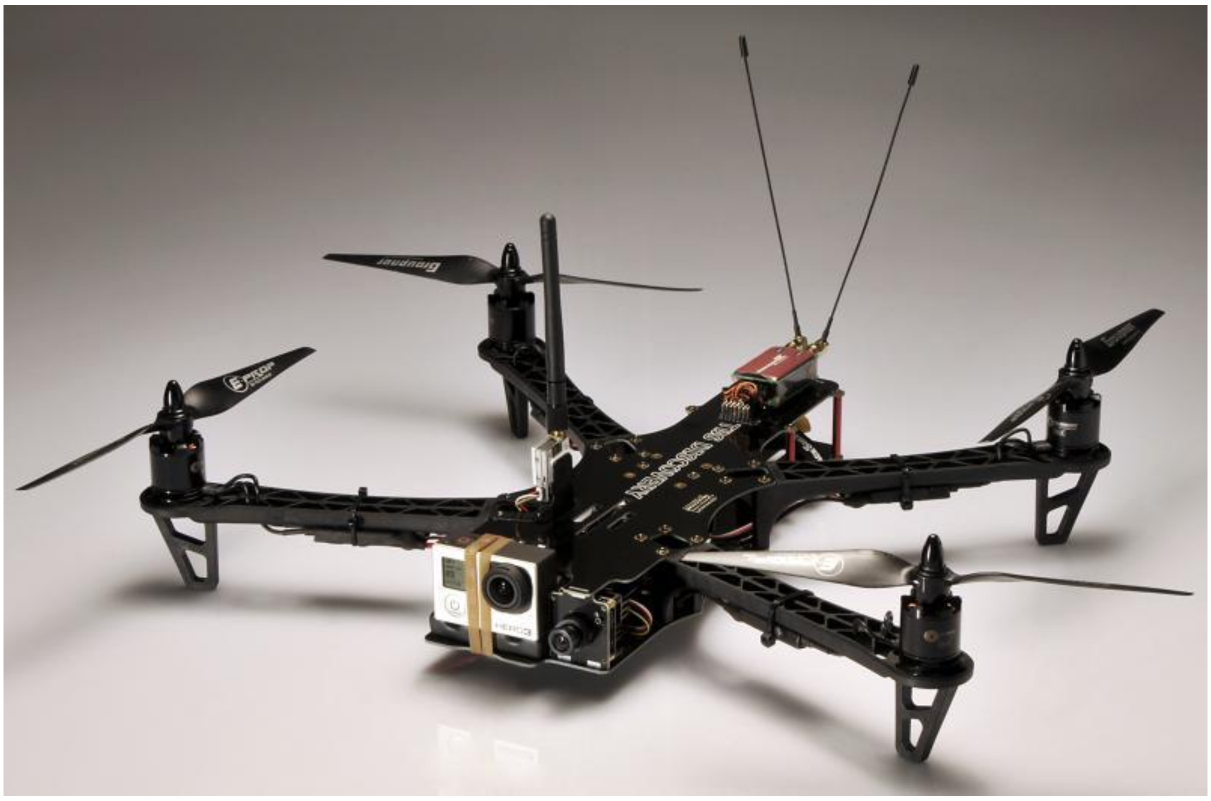


Figure 7 - TBS Discovery (Team-blacksheep.com, 2016)

During the proposal for the project, 3D printing was identified as a viable technique for producing parts needed for the multi-rotor platform, for reasons outlined in the section 4.3.6.1 3D Printing and Thingiverse repository. As a result of this, another platform that was considered was a 3D printed tri-copter, built from a freely available frame design found on Thingiverse (Thingiverse.com, 2016). The platform was primarily a FPV platform, with a lightweight and inexpensive design, meeting some of the requirements laid out. However, after spending several hours configuring the platform, including trying multiple versions of firmware and tuning the PIDs, the platform still appeared unstable during certain manoeuvres. This was down to an inherent problem with the tri-copter configuration; the rear two rotors are replaced with a single tilt rotor, which led to difficulties tuning the multi-rotor for stable flight, leading to issues such as oscillations when returning to a level position from a yaw rotation. In addition to this, whilst built from a 3D printed design, that could be easily modified using modelling software, the design was still limiting in terms of space and options to mount additional hardware.

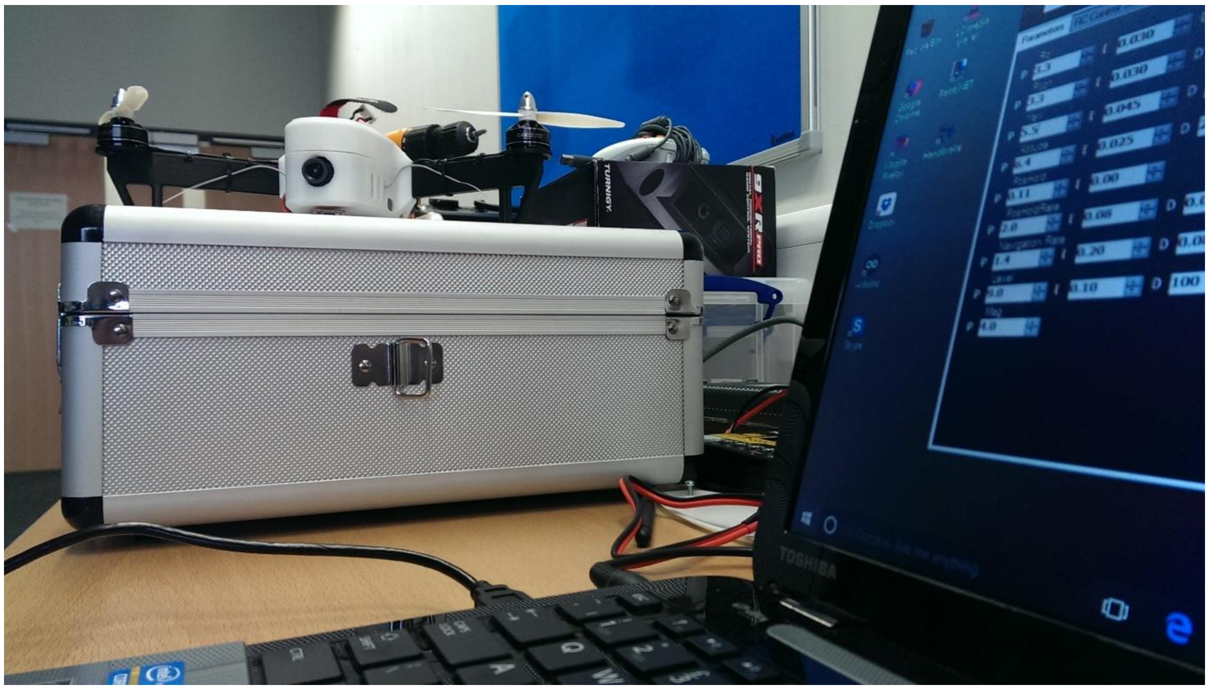


Figure 8 – Tuning the PIDs for the 3D printed tri-copter.

After evaluating multiple possibilities for the multi-rotor platform, it was determined that a custom platform would be built using 3D printing techniques, in order to fully meet the requirements laid out. Whilst the options looked at often met several of the requirements, it was felt that a custom built platform would be less limiting during development, allowing for the design to be better tailored towards the development.

5.1.2 Virtual Reality

The second objective of the project was the acquisition and familiarisation with suitable VR equipment and associated SDKs. Before this objective could be completed, some initial research into available VR technology was undertaken.

One commercially available and affordable entry into VR that was initially considered is Google Cardboard (Google.co.uk, 2016). Google Cardboard is a low cost VR platform developed by Google that uses a head mounted viewer along with a smartphone to create an affordable HMD, and was initially considered due to the low cost entry level hardware that is available. An SDK is available for both the iPhone Operating System (iOS), the Android OS and Unity, to develop applications for use with the HMD. Whilst this could be considered viable solution, Google Cardboard was dismissed for use in this project for several reasons. Firstly, the platform is much less developed than other VR platforms, offering an experience that is subpar to alternatives such as the Oculus Rift. Additional complexity would have been introduced by using a mobile device as the platform for development; multiple smartphones are available for use with the platform, with varying performance and screen specifications. On top of this, other components required for the project would have been either difficult or impossible to implement on a mobile device, due to constraints with both software and hardware.

Figure 9 - Various head mounted viewers available for Google Cardboard (Google.co.uk, 2016)

The Oculus Rift (Oculus.com, 2016) is a commercial available HMD for VR applications, recently released for consumer consumption. It was widely considered the de facto standard for VR at the time of the project, with support for development in OpenGL, DirectX and Unity. An SDK is available for the platform, with a wide range of resources such as documentation and tutorials. The Oculus Rift was chosen as the VR platform for this project because it was available to the author, and significant support was available for development.

Figure 10 - Oculus Rift DK2 HMD (Oculus.com, 2016)

5.2 Hovership MHQ2

After initial research was complete, development on the project artefact began. It was decided that the first iteration of development would focus on building the initial platform needed for development. Requirements for the first iteration were already laid out in the previous section 5.1.1 Multi-rotor Platform, and so no further requirements gathering was required.

Several possible designs available from Thingiverse were considered for the platform, based on existing hardware that was available. The Hovership MHQ2, shown in Figure 11, was the chosen design for the platform due its popularity; with over 36,000 downloads and 90 makes, there was a degree of confidence that the design was clearly well developed and therefore any possible issues when building the platform should be minimal.

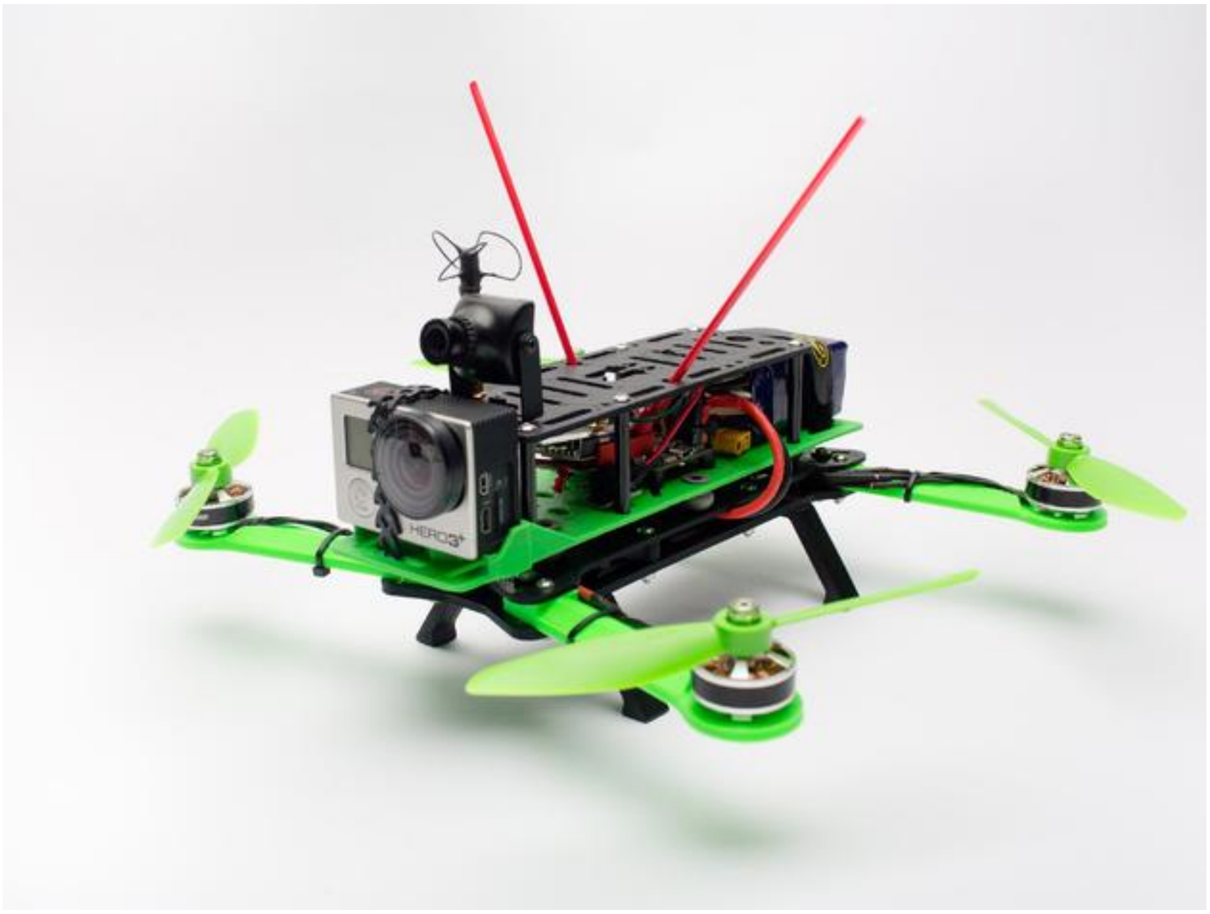


Figure 11 - Hovership MHQ2 (Thingiverse.com, 2014)

During the build, issues with the original print settings for the 3D printed parts were noticed; arms were originally printed at 20% infill, which resulted in considerable flex which could cause issues when the arm was subjected to a moment during flight. These arms, and subsequent parts, were then printed at a higher infill of 40%, which provided more than enough rigidity for the frame.

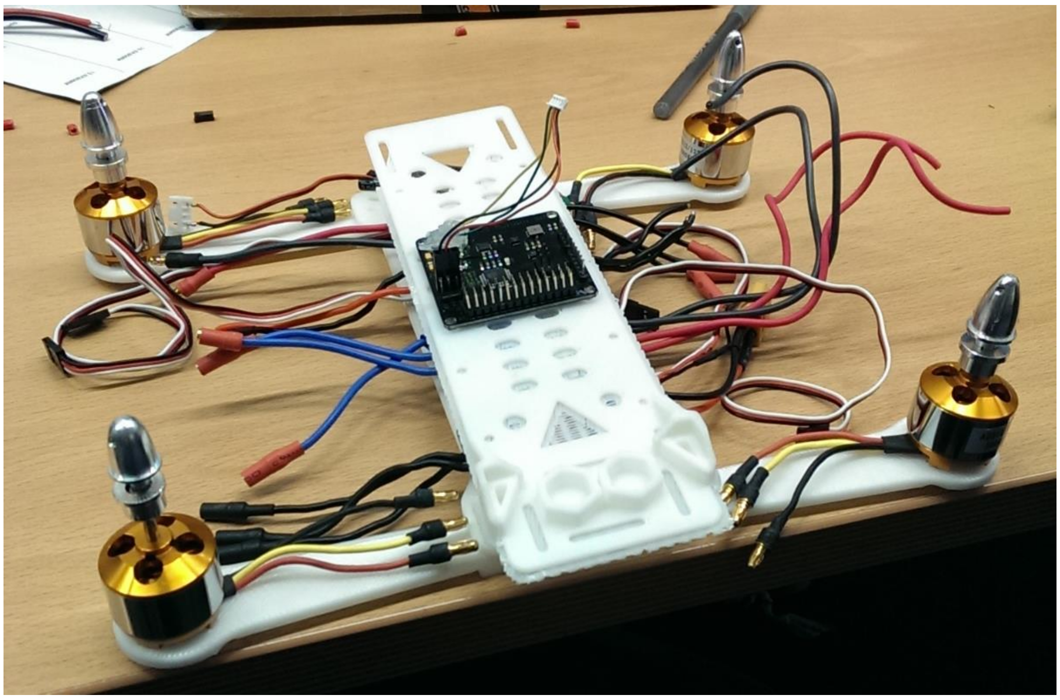


Figure 12 - Testing the layout of components on the 3D printed MHQ2 frame.

Other issues were also encountered during the build, including issues with the limited documentation available for the components in use due to being old hardware. This prolonged development slightly, as it took time to become familiar with specifications, and configurations than was first expected. Space issues were also a concern, as the components were not really aimed at this size frame, and required out of the box thinking to resolve.

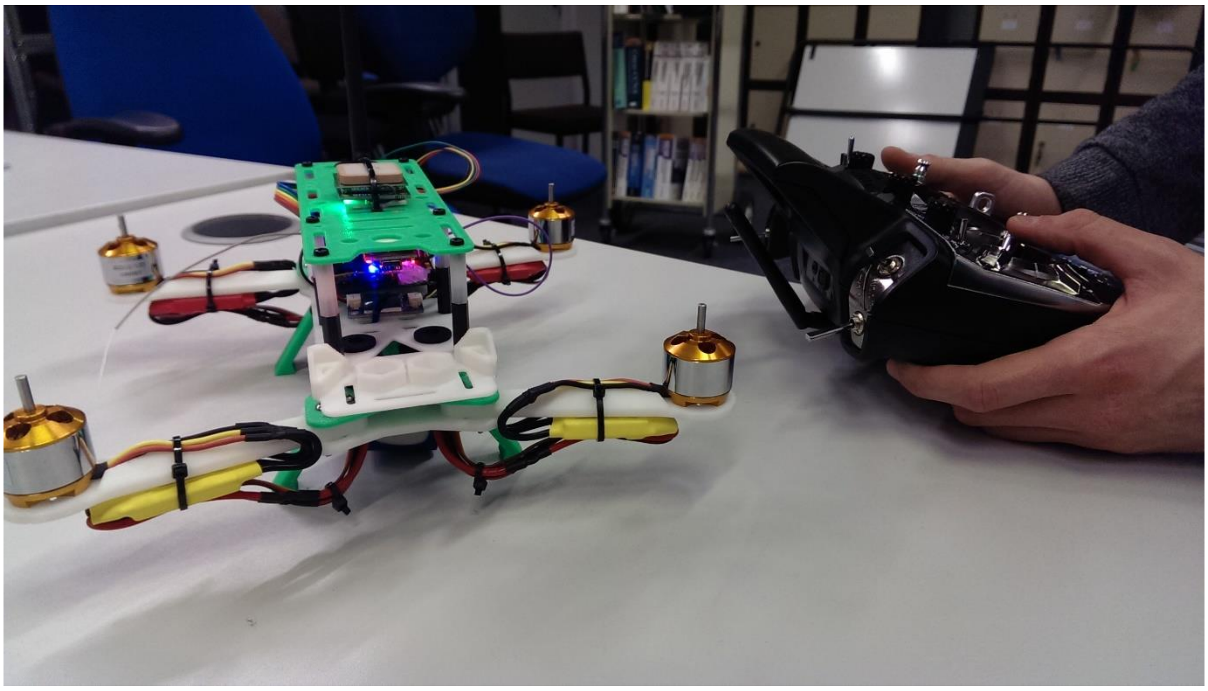


Figure 13 - Fully built MHQ2 without propellers during testing.

Once assembled, the platform was configured and PIDs were tuned, in an attempt to achieve stable flight. During testing, issues were noticed with the chosen combination of components, which were preventing the multi-rotor from achieving flight, even at high levels of throttle. Investigation into this issue revealed that the choice of motor and propeller combination was providing an inadequate thrust-to-weight ratio, even when using aggressively pitched or 3-bladed propellers. Whilst the platform was suitable, and all but one of the requirements lay out, the issues faced with achieving a stable flight meant further work would be necessary in order to complete the final platform for the project.

5.3 MultiWii Serial Protocol

Despite issues encountered with the platform during the first iteration, development was still able to continue, and the second iteration focused on developing the software and hardware needed to communicate with the multi-rotor. The requirements for this software, shown in Figure X, were inferred from the project objectives.

- Successfully connect to and disconnect from the multi-rotor flight controller.
- Successfully read telemetry data, such as IMU data, from the flight controller.
- Successfully send control signals to the flight controller.

Figure 14 - Software requirements for multi-rotor communication.

In order to successfully communicate with the flight-controller, the MultiWii Serial Protocol (MSP) (Multiwii.com, 2016) had to be used; MSP is the de facto standard for interacting with MultiWii based flight-controllers. Some existing implementations of the protocol were available, but these were either incomplete or not available in the language being used, and instead the protocol was implemented in C++. In order to achieve this, the design of the protocol had to first be examined; this consisted of 3 types of messages: commands, requests and responses.

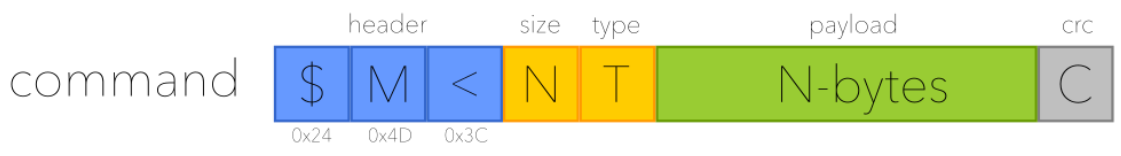


Figure 15 - Structure of an MSP command

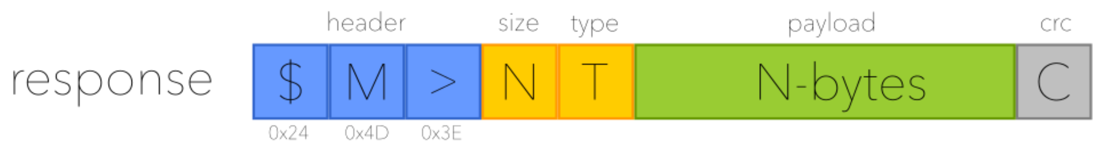


Figure 16 - Structure of an MSP response

Figure 18 and Figure 16 both show the breakdown of two types of MSP messages into their constitute parts, which consist of the header, the size, the message type, a payload if required, and a single bit cyclic redundancy check (CRC). The header follows a strict format that is used to identify that the data being sent is the start of a message, as well as the direction of the message, depending on whether is it either a command or a response. A command is used to update the flight controller with new data such as RC input values, and a response is the result of a request to the flight controller for information. Figure 17 shows the slightly differing structure of a request message, which is comparable to the command message just without the option for a payload.

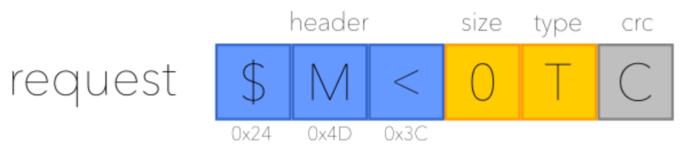


Figure 17 - Structure of an MSP request

As well as the header, the protocol also uses a single bit CRC at the end of each message, to ensure the messages being transferred between the two devices are correctly sent and received. Figure 18 shows the formula for calculating this CRC bit. Due to the safety critical nature of the protocol, using a form of error detection is essential, as the protocol can be used to send RC input values to the flight controller, and therefore control the multi-rotor. If errors are not correctly detected in received messages, then issues could arise if incomplete or scrambled data is received, a problem that is inherent with the types of communication hardware normally used to transmit data to and from a multi-rotor.

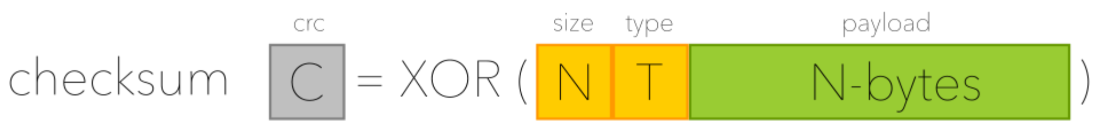


Figure 18 – Formula for CRC checksum used in MSP messages

In order to develop the MSP implementation, a hardware setup was needed first. During early development, this simply consisted of using a MultiWii flight controller board and a Bluetooth module. By using a simplified hardware setup, additional complexity was removed from the development, and focus could be placed on simply developing the software. As development progressed, a more robust solution was used on-board the multi-rotor. This solution consisted of a 900Mhz telemetry transmitter, which was mounted to the multi-rotor frame and connected to the flight-controller, and a USB telemetry receiver.

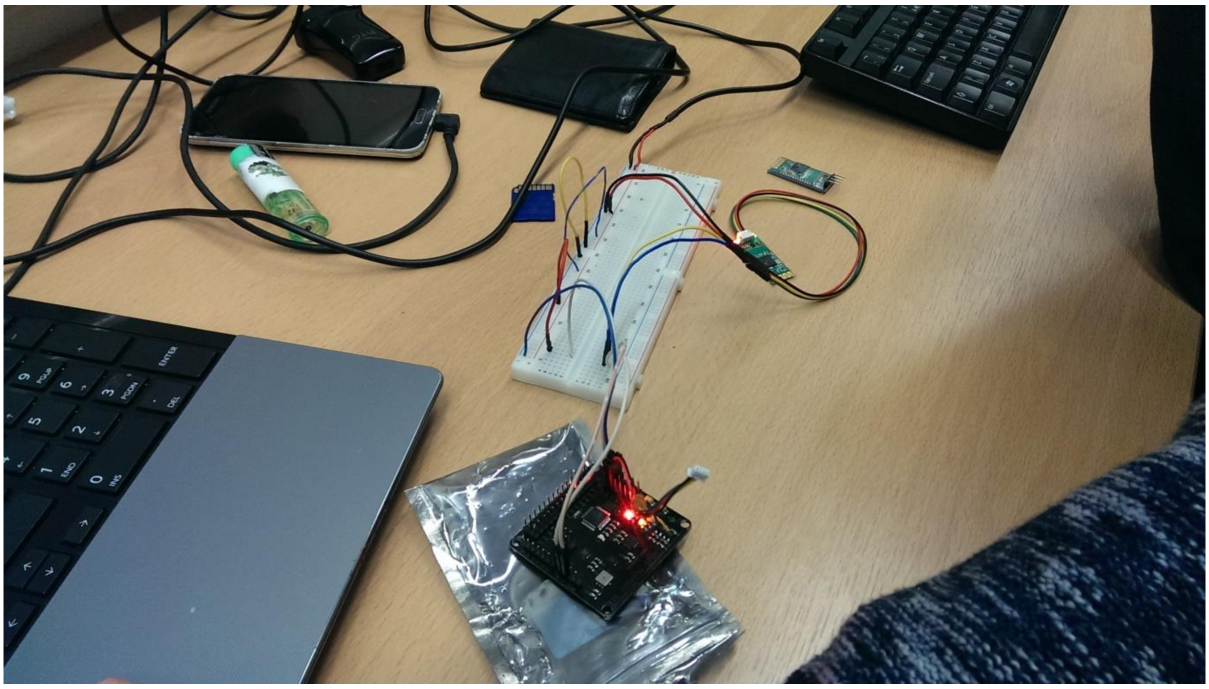


Figure 19 - Simplified hardware setup for MSP development

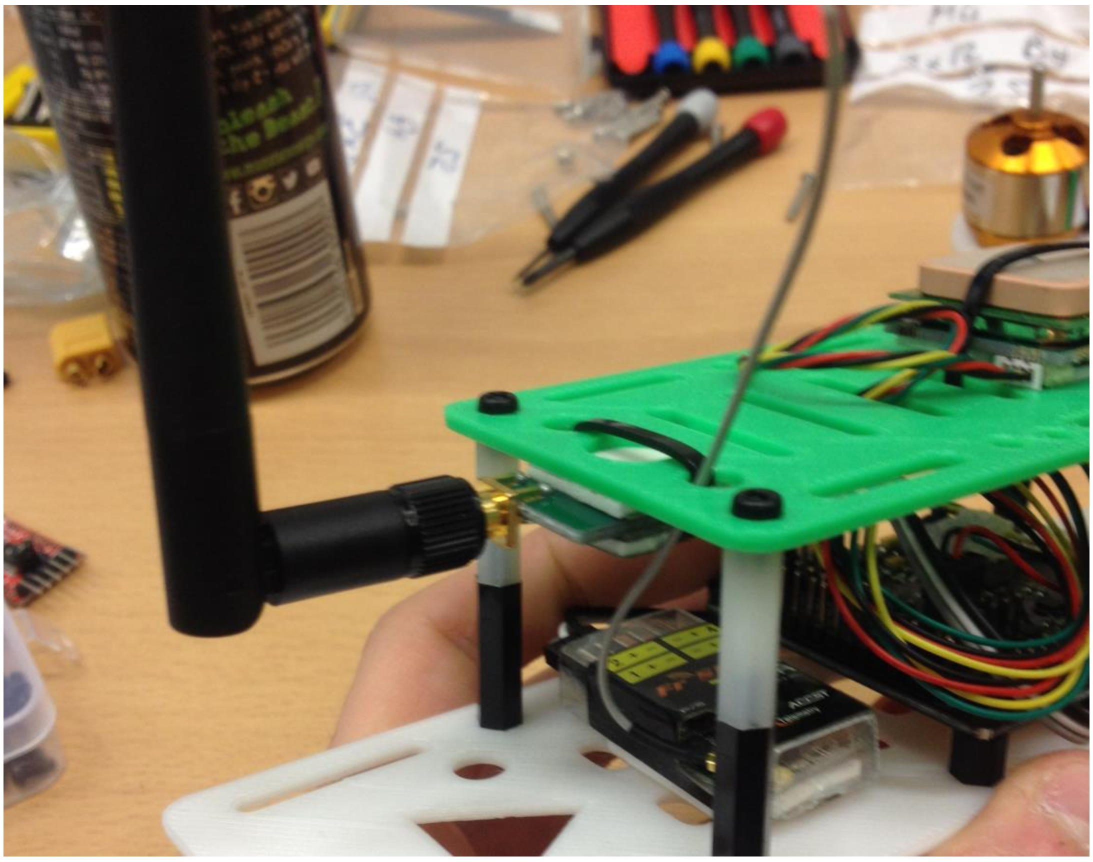
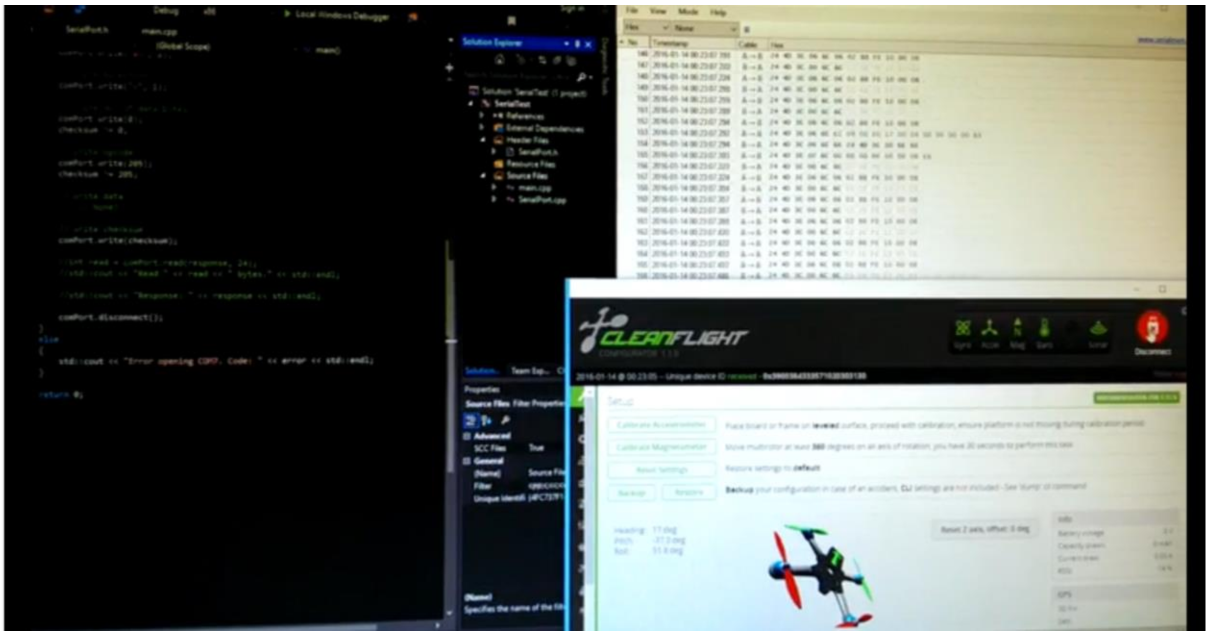


Figure 20 - Telemetry transmitter mounted to multi-rotor during MSP development

In order to test the implementation, code was written in order to use the protocol to send and receive messages to the flight-controller. These tests included connecting to the flight-controller, reading the flight-controller board information, reading telemetry IMU data, and sending RC signals. Figure 21 shows one method that was used during development to test the communication between the developed code and the flight controller, through the inspection of the serial packets being transmitted between the two.



```
comPort.write('<', 1);

comPort.write(0);
checksum ^= 0;

comPort.write(205);
checksum ^= 205;

comPort.write(checksum);

//int read = comPort.read(response, 24);
//std::cout << "Read " << read << " bytes." << std::endl;

//std::cout << "Response: " << response << std::endl;

comPort.disconnect();
}
else
{
    std::cout << "Error opening COM7. Code: " << error << std::endl;
}

return 0;
```

Figure 21 – Visual confirmation of communication during development of the MSP implementation.

5.4 Input Devices

Once the implementation of the MSP protocol was complete, it was realised that the implementation could be used in order to experiment with possible input devices, required for meeting one of the project objectives. The list of requirements detailed in Figure 22 below, derived from this project objective, details how the suitability of the input devices were evaluated to determine suitability for the use in this project.

- Adequate number of inputs to safely control all aspects of the multi-rotor.
- Able to provide intuitive control of the multi-rotor.
- Provide a familiar feel to the user.

Figure 22 - List of requirements used to evaluate possible input devices.

One input device, the Turnigy 9XR Pro (Turnigy9xr.com, 2016) transmitter shown in Figure 23, was previously used during the development of the MHQ2 platform. As a result of this, it was unnecessary to carry out additional testing; the transmitter met all of the requirements laid out, and was therefore a suitable possibility for use in this project.

Figure 23 - Turnigy 9XR Pro transmitter (Turnigy9xr.com, 2016)

The next input device that was tested was an Xbox 360 Controller (Xbox.com, 2016), chosen for its similarity in layout to a RC transmitter, and also being familiar due to popularity. Whilst this was successful, and it was possible to control the multi-rotor with the Xbox 360 controller, it was determined that the control felt very inconsistent and unsafe. Investigation into this issue revealed that it was caused by noisy input from the Xbox 360 controller, and therefore due to this it would not be suitable for use in the project.

Figure 24 - Xbox 360 Controller (Xbox.com, 2016)

Whilst other possible input devices could have been tested, it was assumed that these were more than likely to have very similar issues to the Xbox 360 controller; traditional RC transmitters have gimbals with high sensitivity potentiometers, in order to provide precise input, and this level of precision would have been hard to duplicate with other available input devices. Due to this, it was decided that a normal RC transmitter would be used for the project, as it provided much more precise input and therefore much more finesse when controlling the multi-rotor.

5.5 VR Familiarisation

In order to successfully develop the augmented reality experience, one of the objectives that needed to be completed was the development of VR software. In order to meet this objective, a series of requirements that needed to be met were drawn up, and are shown in Figure 25.

- Successfully display content on the HMD.
- Read IMU data needed for head tracking from the HMD.
- Create and display objects in a basic VR world.
- Augmented a real-time video stream into the VR world.

Figure 25 - Requirements list for meeting the objective

Once the requirements were set, a series of C++ classes were written in order to encapsulate much of the complexity of the Oculus Rift SDK, exposing an interface that was much easier for the rest of the application to. In order to do this, a design pattern known as the Adapter pattern was used. This design pattern is a common pattern that is often used in order to provide a new and more suitable interface, which still provides similar functionality to the old interface. Figure 26 shows how this adapter was then tested; a simple cube was rendered and displayed on the HMD to confirm the adapter was working as expected.

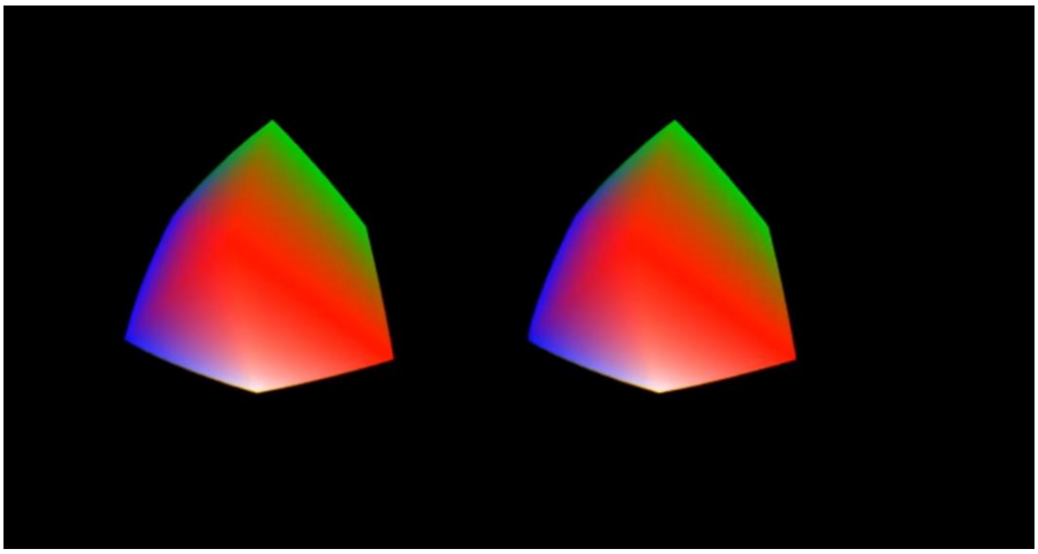
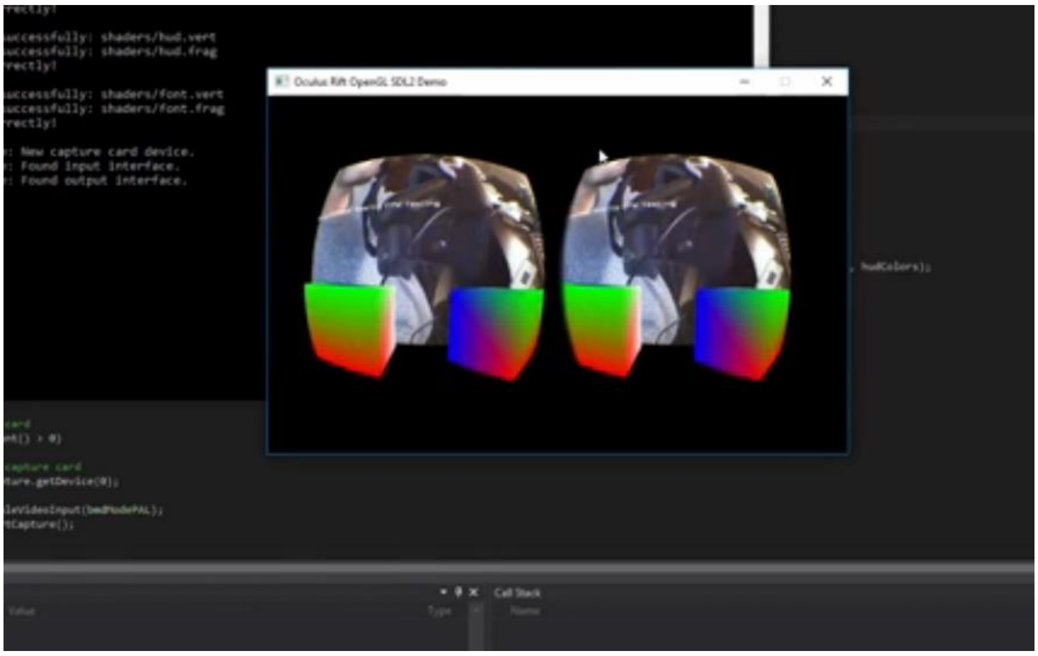


Figure 26 - Coloured cube shown in Oculus Rift HMD.

The next step of the VR development consisted of creating a small VR demo scene that the user could move and look around. This would be used to confirm that IMU data was being read correctly from the HMD, and then used to orient the camera in the VR world. Once head tracking was working correctly, development moved to the augmentation of a video stream into the VR world. In order to do this, a simple RGB webcam was used instead of the multi-rotor, to simplify the setup required.

Figure 27 shows the video stream of the real world being augmented into the VR world on a simple quad in the VR world.



```
rectly!

uccessfully: shaders/hud.vert
uccessfully: shaders/hud.frag
rectly!

uccessfully: shaders/font.vert
uccessfully: shaders/font.frag
rectly!

: New capture card device.
: Found input interface.
: Found output interface.
```

```
card
nt() > 0)

capture card
ture.getDevice(0);

leVideoInput(bmdModePAL);
tCapture();
```

```
, hudColors);
```

Figure 27 - Initial augmented of video stream into VR world

5.6 Spyda 500

Due to issues faced with the MHQ2 Hovership, detailed in section 5.2 Hovership MHQ2, the decision was made to develop a second, more suitable platform for the project. The Spyda 500 (Thingiverse.com, 2013) is a FPV multi-rotor frame based on the popular Dead Cat design, with a wide front providing better FPV footage by keeping the rotors out of the view, and a larger body design that provides much more room for mounting hardware. When compared with the original requirements for the multi-rotor platform the frame seemed suitable, and so specification for components were decided, in order to gauge the total cost of the platform.

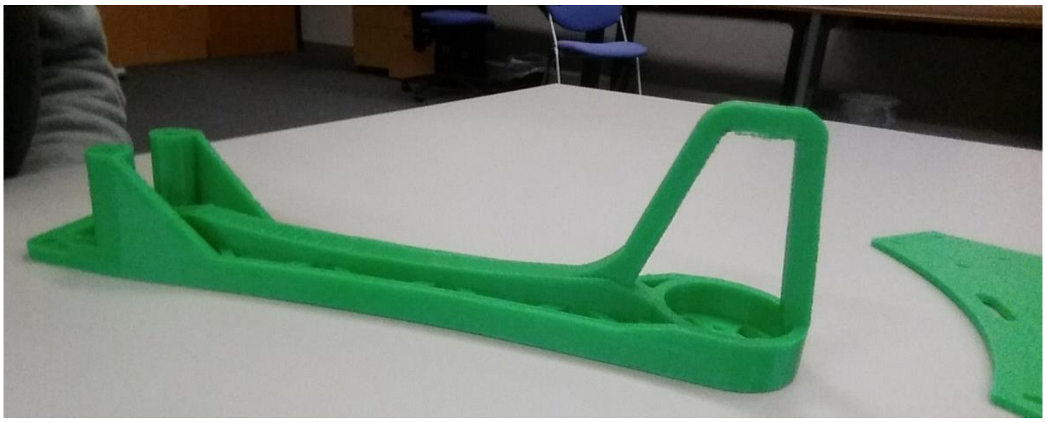


Figure 28 - 3D printed leg for Spyda 500 frame

In order to calculate what components would be suitable, an online calculator called eCalc was used to test possible component configurations, and calculate theoretical flight characteristics. Once the components had been decided on, and the platform was deemed affordable within the scope of the project, the frame was 3D printed using ABS plastic. Figure 28 shows some individual parts of the frame after they had been 3D printed, and Figure 29 shows a mock assembly of the fully printed frame.

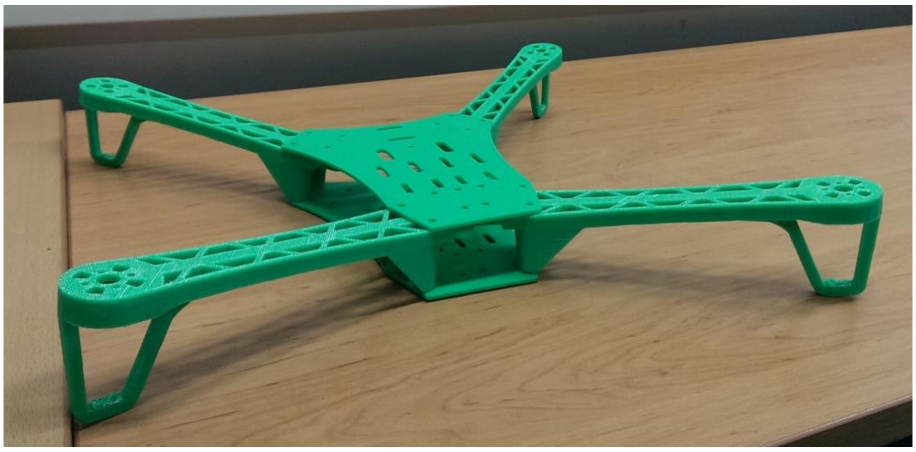


Figure 29 - 3D printed Spyda 500 frame

Once the frame had been printed, the Cleanflight configurator was used to configure the flight controller. This included ensuring that the correct frame layout was selected, as well as ensuring receiver and input signals were setup correctly. Once the board was configured, the components were mounted and wired, and multi-rotor was then powered up without propellers in order to calibrate and verify the correct direction for each motor. As the motors and ESCs were connected using bullet connectors, any motor spinning the wrong direction was easily correctly by simply swapping 2 of the 3 cables. Upon verifying the motors were all spinning the correct direction, some basic testing without propellers was carried out. This included ensuring that all input responded as expected, and that the multi-rotor was safe to fly. Lastly, any on-board sensors such as gyroscopes and magnetometers were calibrated, and the propellers were fitted. Figure 30 shows the completed Spyda 500, along with Bluetooth connectivity to a smart phone; this was used in the next step during initial testing and tuning of the platform.

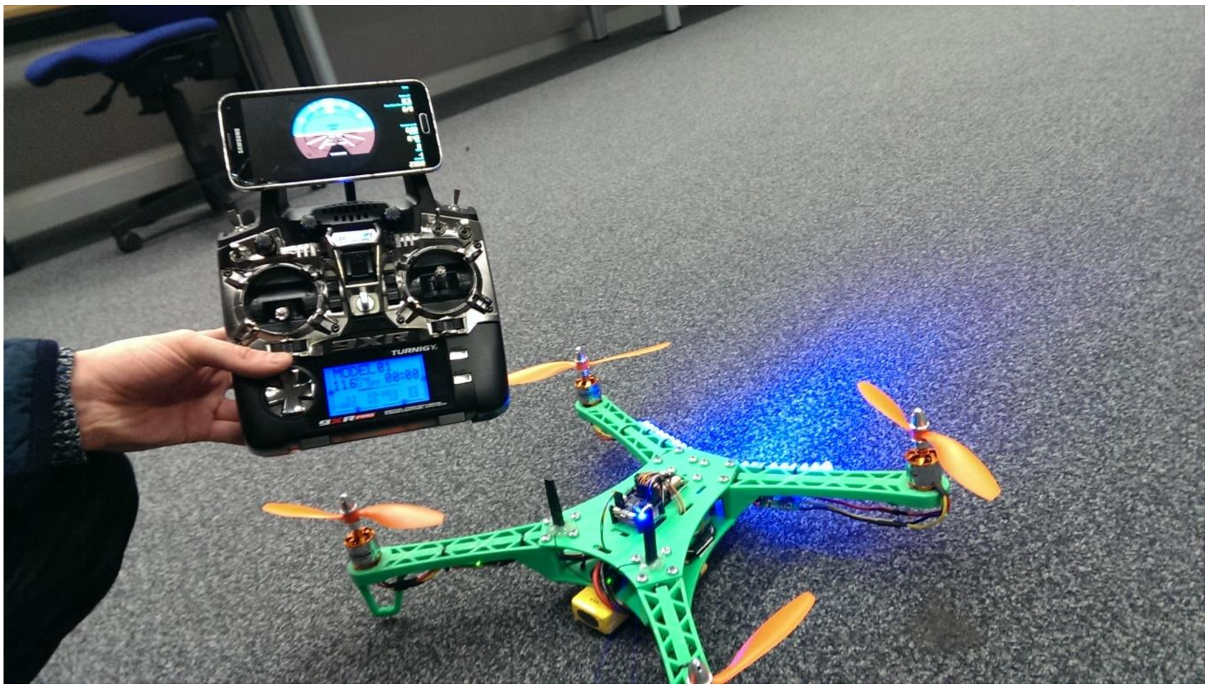


Figure 30 – Completed 3D printed Spyda 500 connected to smart phone for testing

The last requirement of the multi-rotor platform yet to be met was the achievement of stable flight. In order to evaluate this, initial testing of the platform was carried out, and any adjustments to the PID tuning were made. The use of a smart phone during this stage made changes to settings possible on the fly, rather than having to be connected to a PC, thus reducing the time taken to configure the platform. Overall, very little changes were needed in the way of tuning the PID loops, as the preconfigured settings in the Cleanflight configurator already worked well, and the multi-rotor was already very stable in flight.

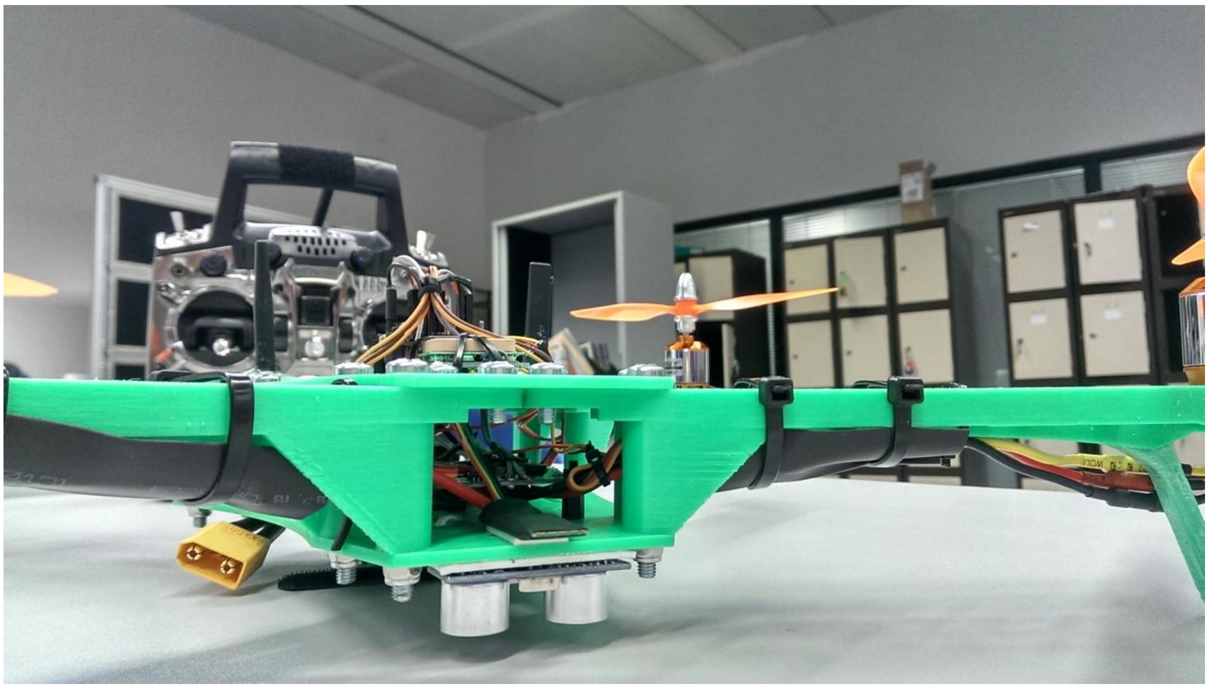


Figure 31 – Completed and tuned 3D printed Spyda 500

Further use of the platform outdoors was carried out, both for familiarisation with flying a multi-rotor, as well as to carry out extended testing in order to ensure there were no issues with the platform. During this, one possible issue that was highlighted was the use of a 3D printed frame, and the design of the frame chosen; the legs of the frame were rather weak, and often broke even under light impact. As a temporary fix acetone was used; this was applied to the broken areas in order to slightly melt the plastic, before the piece was held back in place whilst the solvent evaporated, resulting in the two pieces being fused back together. As a temporary solution this worked fine, however as this was used more and more, concern was raised that the use of acetone was damaging the honey comb like structure inside the leg. This was reducing the legs overall strength and increasing the likelihood the leg would break again, which eventually would require the whole arm to be reprinted. One solution that was briefly considered was to modify the design of the arm in order to have detachable legs, which would mean that only a new leg would have to be printed, rather than the whole arm. This was dismissed due to time constraints, and instead extra care was taken with the platform to minimise damage to the legs.

5.7 Video Transmission

In order to successfully create an AR experience with the multi-rotor, a video feed was essential, and in order to achieve this, both a hardware and software solution was required. To achieve this, a few simple requirements for the iteration were laid out, as presented in Figure 32.

- Successfully transmit video from the multi-rotor.
- Successfully receive and display video from the multi-rotor on screen.
- Ensure latency, resolution and framerate are suitable for the application.

Figure 32 - Requirements for the video transmission

5.7.1 Analogue

To simplify development, the existing Tri-copter seen in Figure 33 which was already equipped for analogue FPV was used. This allowed the development of using analogue video for augmented reality to be tried and tested, without first having to modify the multi-rotor platform and equipping FPV gear.

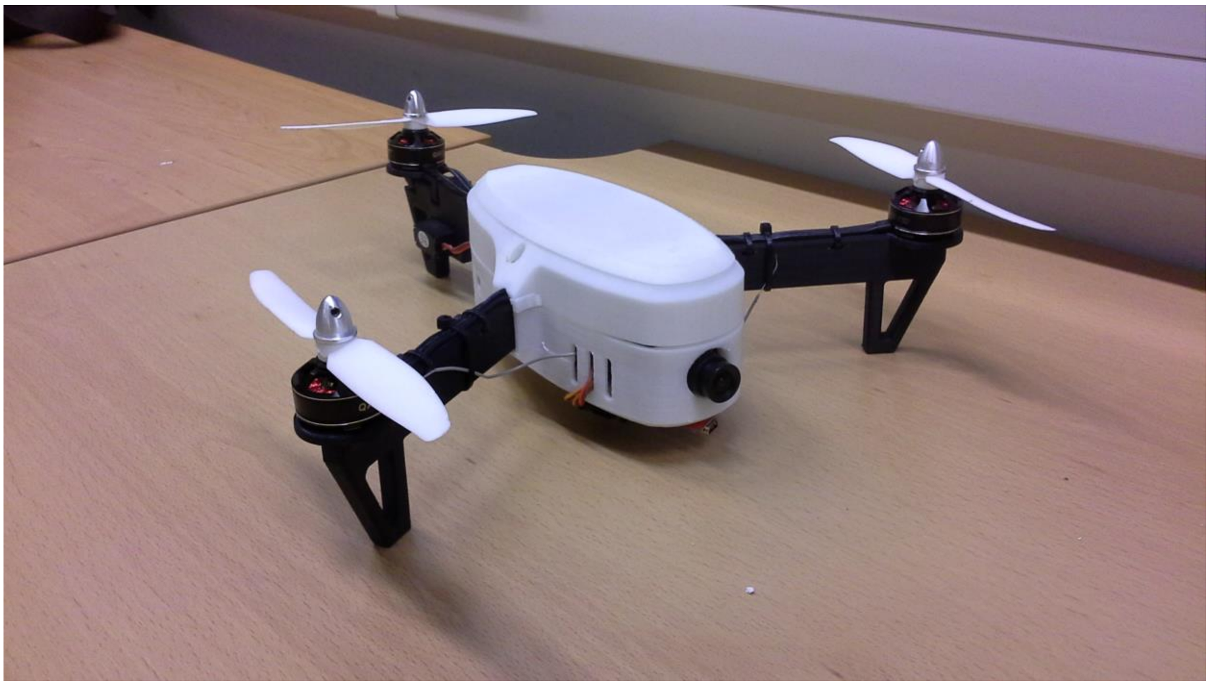


Figure 33 – 3D printed FPV tri-copter used for development of analogue video transmission

In order to convert the video into a format suitable for use with the Oculus Rift, a video capture card was required to convert from analogue to digital. For the reasons mentioned in section 4.3.6.3 Blackmagicdesign Intensity, the video capture device chosen was the Blackmagicdesign Intensity, shown in Figure 34.

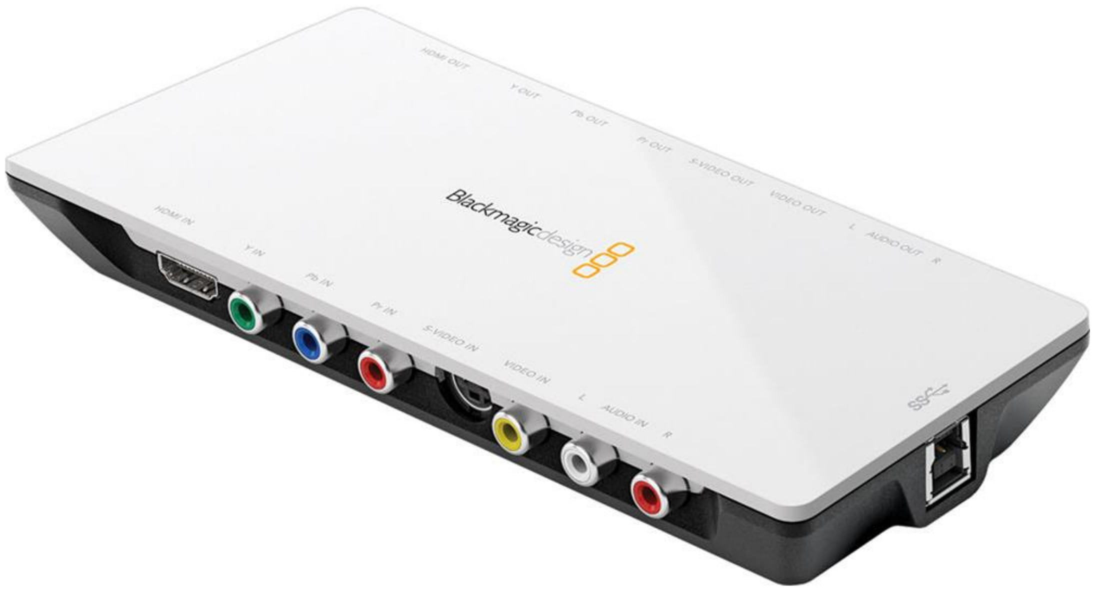


Figure 34 - Blackmagicdesign Intensity video capture card used for analogue to digital video conversion (Blackmagicdesign.com, 2016)

The capture card, along with an analogue video transmitter, was the final hardware setup required for receiving the analogue video and converting it for use with the Oculus Rift. The arrangement of this setup is shown in Figure 35.

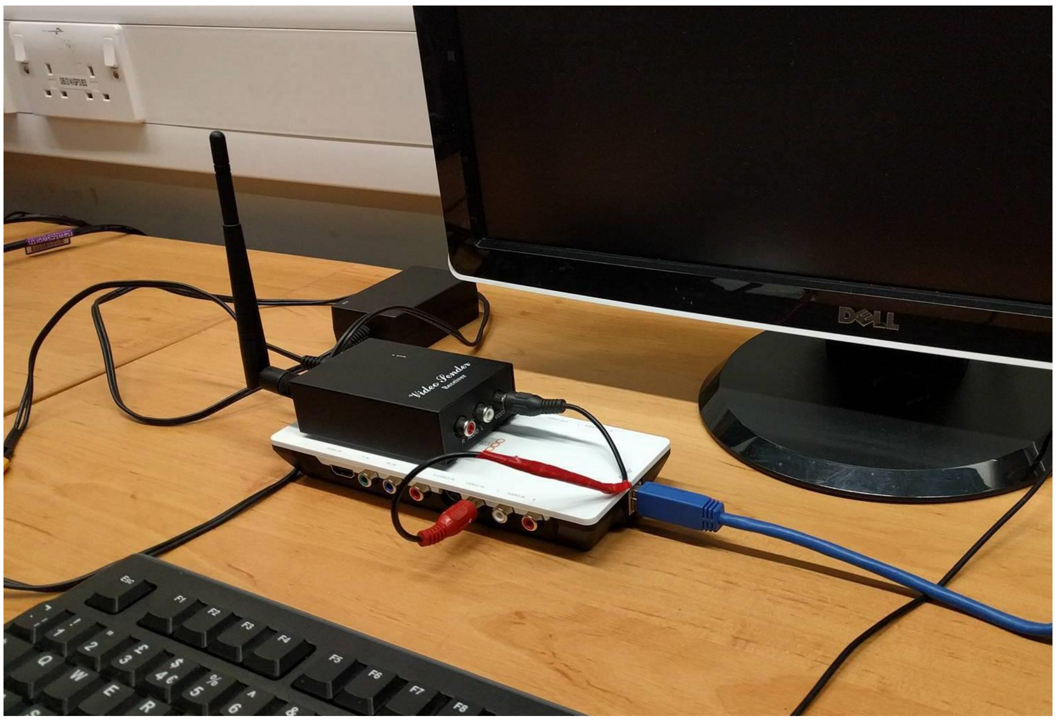


Figure 35 - Hardware setup used for receiving analogue video

Once the hardware setup was assembled, attention turned to the development of software for interfacing with the Intensity video capture card. As the Intensity is a high-end consumer device, an SDK was available online, aimed at developers creating applications using the hardware. This SDK exposed the device through a C++ interface, that was then used to retrieve the individual frames of the analogue video; this was implemented through the use of GPUDirect (NVIDIA Developer, 2015), which the Blackmagicdesign SDK had built in support for. Issues were originally faced with the design of the SDK, which required a call back function, which was called from a separate thread, to retrieve the frame contents; OpenGL is difficult to implement across multiple threads, requiring context switching and thread blocking. This was overcome through careful design consideration, and an intermediate buffer that was simpler to multi-thread was used.

Once the software had been implemented, some basic latency testing was carried out to assess the level of latency the pilot would experience when flying the multi-rotor. This was an important consideration to ensure the safe operation of the multi-rotor; as multi-rotors can be flown at speed, even moderately high levels of latency would cause issues with control. Despite the multi-rotor used in this project would only be used at low speed, the scope of the project includes applications such as FPV racing. A basic approach to testing the latency of the analogue video solution was used, which involved using an onscreen stopwatch, which was then viewed through the video transmission system, and displayed on a second screen. A screenshot could then be taken, which would capture the time difference between the two images; this was done multiple times in order to achieve a rough average. The solution averaged around a 80ms delay, which was considerable acceptable for the project; however the FPV racing application mentioned before might require a lower degree of latency to be effective.

5.7.2 Digital

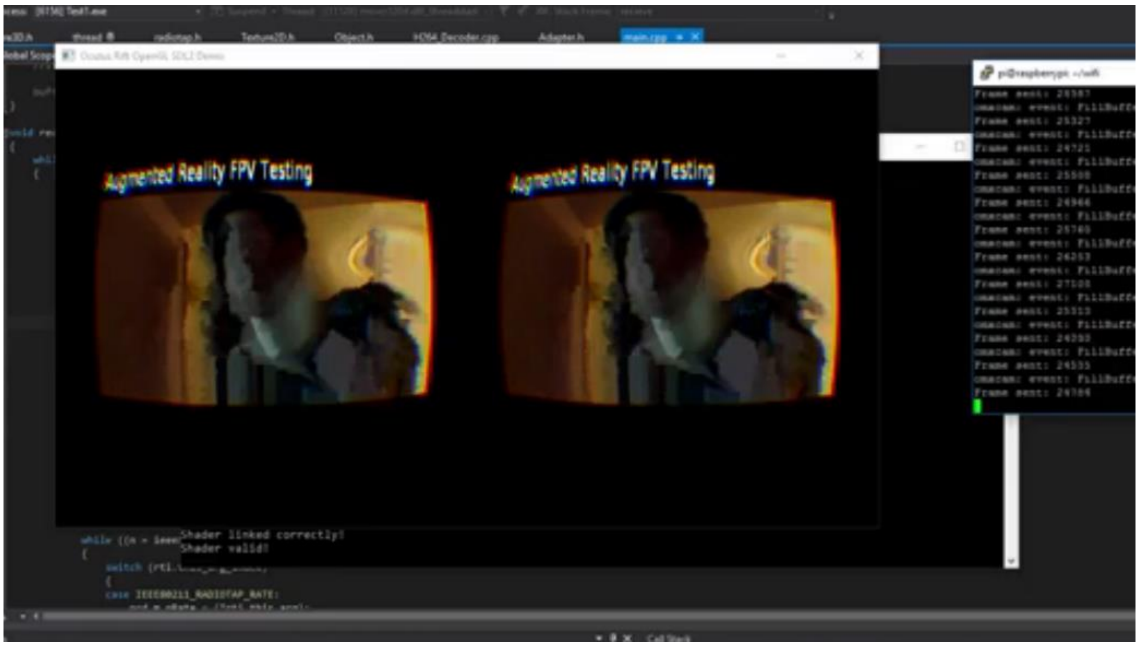
During the development of the analogue solution, concern over the resolution and frame rate of the analogue video formats available, prompted investigation into other methods of video transmission that could resolve these concerns. One possible solution that was considered was commercially available high definition video transmitters, including both the DJI Lightbridge (Dji.com, 2016) and the Connex HD Video Downlink (Connex.amimon.com, 2016). Both of these solutions were however dismissed, due to the cost of the devices being out of the scope of this project.

Figure 36 - DJI Lightbridge (Dji.com, 2016)

Instead of opting for a consumer device, a custom solution for transmitting high definition video was developed. In order to achieve this, several components had to be considered. A method of wireless data transfer was needed, with a throughput suitable for the bitrate of the desired video format; Wi-Fi was chosen immediately as it was a suitable communication medium, which has been used in other wireless video streaming devices successfully. This then involved looking at hardware suitable for mounting on the multi-rotor to capture and send video over Wi-Fi. The single board Raspberry Pi model B was chosen, for the reasons outlined in the section 4.3.6.5 Raspberry Pi.

Figure 37 - Raspberry Pi Model B (Raspberry Pi, 2016)

During implementation, issues were noticed with considerable packet loss, which resulted in major artefacts in the video received that appeared to worsen over time. An example of these artefacts is shown in Figure 38. Investigation into the issue revealed that packet loss was causing artefacts rather than dropped frames due to the compression format used.



```
pi@raspberrypi: ~/wifi
Frame sent: 25587
omxcam: event: FillBuffe
Frame sent: 25327
omxcam: event: FillBuffe
Frame sent: 24721
omxcam: event: FillBuffe
Frame sent: 25508
omxcam: event: FillBuffe
Frame sent: 24966
omxcam: event: FillBuffe
Frame sent: 25740
omxcam: event: FillBuffe
Frame sent: 26253
omxcam: event: FillBuffe
Frame sent: 27108
omxcam: event: FillBuffe
Frame sent: 25313
omxcam: event: FillBuffe
Frame sent: 24755
omxcam: event: FillBuffe
Frame sent: 24535
omxcam: event: FillBuffe
Frame sent: 24786
```

```
Shader linked correctly!
Shader valid!
```

Figure 38 - Artefacts in H264 encoded video transmission

The H264 compression format used consists of multiple types of frames, including both I and P frames. Whilst the theory behind the types of frames used in H264 encoding is both complex and out of scope for this project, the principle is that full resolution key frames are interleaved with multiple lower quality differential frames, as can be seen in Figure 39. This meant that artefacts increased significantly over time, due to packet loss in some intermediate frames, up until a new key frame was received. Figure 40 shows a visual representation of how both I and P frames are encoded.

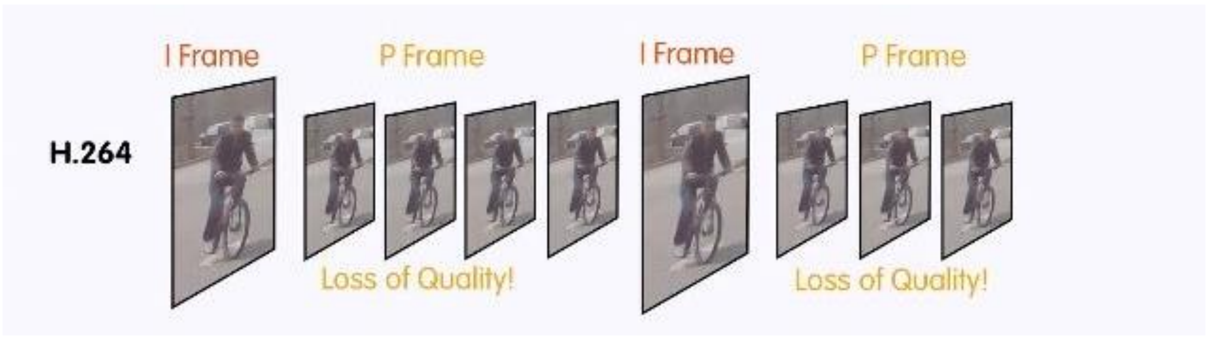


Figure 39 - H264 frame type sequencing (Hamp, 2014)

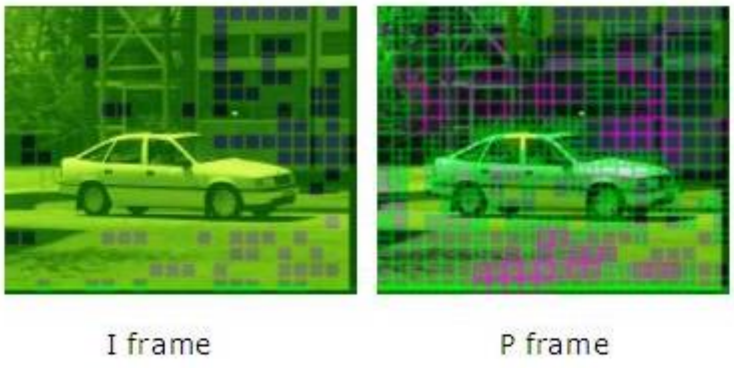


Figure 40 - Visual representation of I and P frames REF!!!

To decrease the effect of packet loss on the video stream, different techniques for providing data redundancy were considered in order to improve the quality of the video transmission. One method initially considered was simple retransmission; this method of redundancy consists of sending each packet multiple times, so in the event of a packet loss, at least one copy of the packet will arrive. Whilst a viable and straight forward method to implement, there were concerns over the effect this would have on latency, as each frame would effectively be sent multiple times, and therefore the throughput of data would significantly increase. Even a single retransmission would double the throughput requirements, and therefore double the time taken to transmit a single frame.

Forward error correction (FEC) is a method used in data transmission for error control, whereby the transmitter sends redundant data along with the original data, so that the receiver can then recognise only the portion of data that contains no apparent errors. This technique is often used for controlling errors in data transmission send over noisy or unreliable communication mediums.

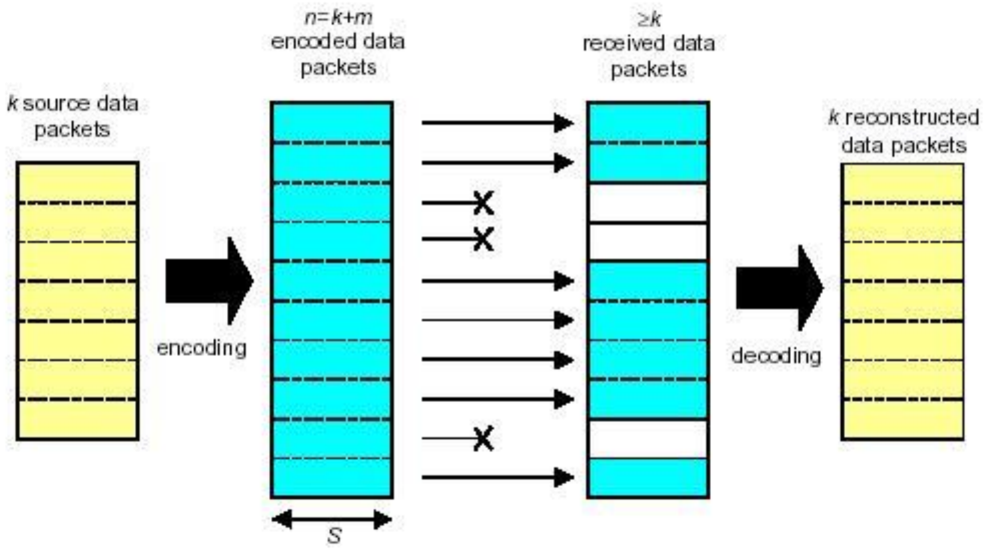


Figure 41 - Overview of a basic FEC algorithm (Cs.technion.ac.il, 2016)

Figure 41 shows a visual illustration of a basic FEC algorithm, in which k data packets are encoded using m redundant packets, resulting in a total of n packets for transmission. Of those n packets, only k or more are then required at the receiving end to reconstruct the original data packets. This algorithm allows for a variable amount of redundant data packets, with the reliability increasing as the number of redundant packets increases. Whilst this type of algorithm can provide similar levels of redundancy to simple retransmission, the data overhead can be considerably smaller and therefore the impact on latency minimal. This is because unlike retransmission, in which every packet has to be retransmitted to provide redundancy, FEC can recover the original data from any k of the transmitted packets, by encoding extra information required for retrieval should a fault occur.

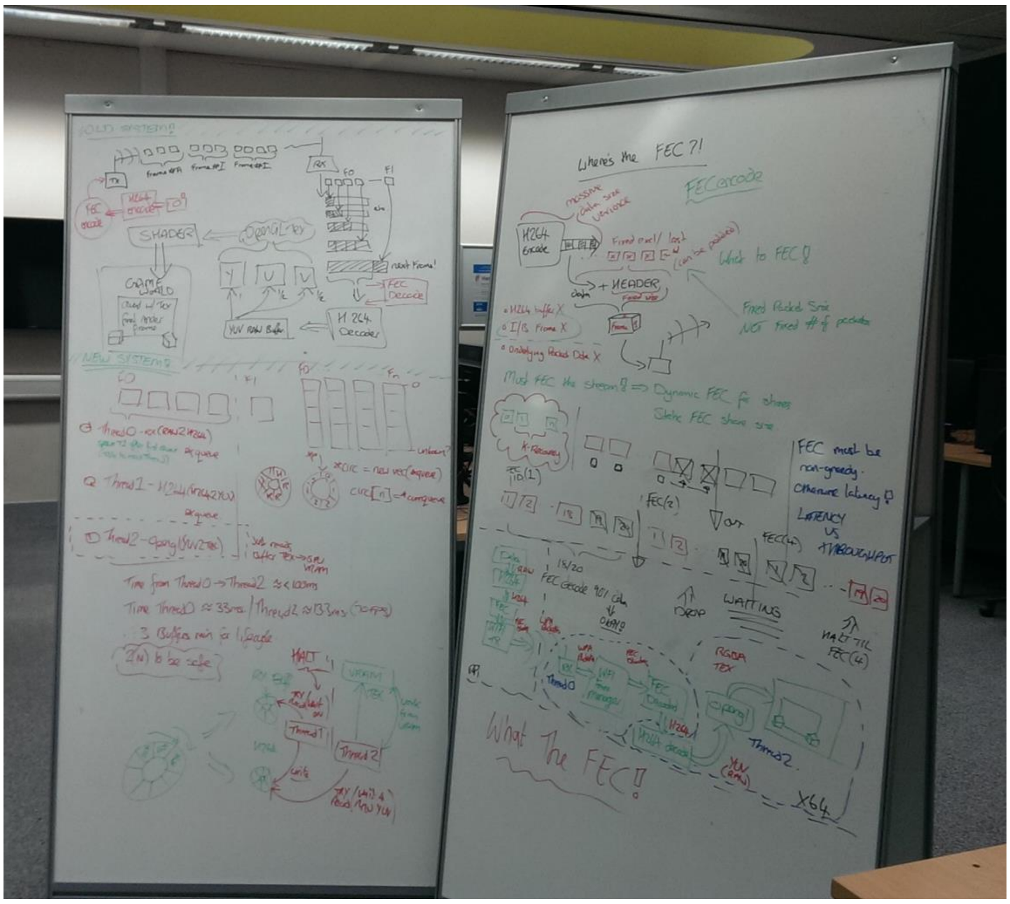


Figure 42 - Whiteboard planning for FEC encoding of H264 video

To successfully implement FEC to improve reliability of video transmission, some design consideration was required. Figure 42 shows some conceptual design work done prior to the implementation of the FEC algorithm, and how multiple circular buffers were required in order to successfully implement the algorithm utilising multiple threads.

During the implementation, some problems were encountered with the development of the algorithm; these mainly consisted of issues with the multi-threaded design that was used, that resulted in either dead locks or buffer synchronisation errors. The features offer by the Visual Studio debugger, as discussed in section 4.3.3 Integrated Development Environments, proved invaluable when trying to pinpoint the cause of the issues, saving vital development time. In addition to this, an issue was experienced with the implementation of the Winpcap library, which meant development was moved to a Linux based operating system, and the Libpcap library used. Whilst the issue was never resolved, investigation seemed to indicate it was an internal buffer issue, and not an issue with any of the developed code.

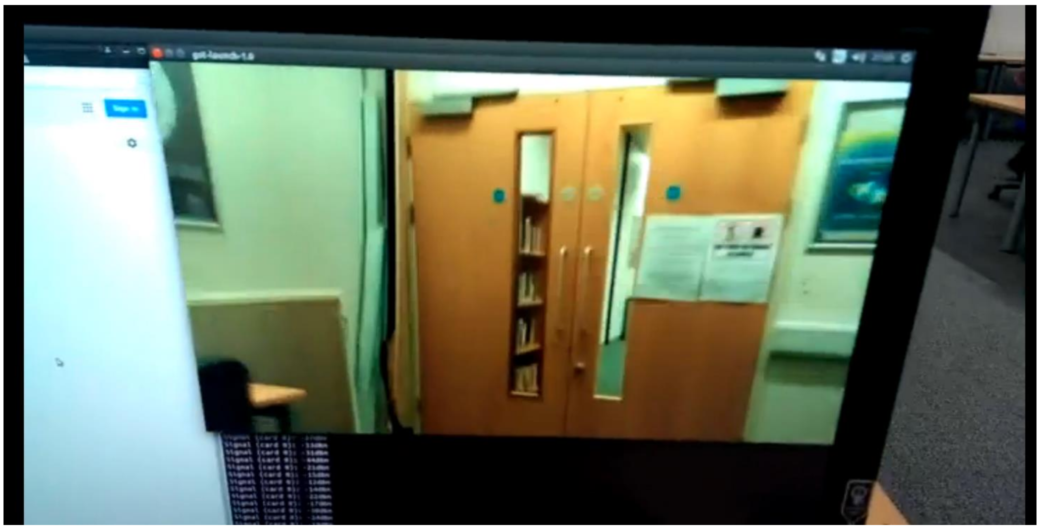


Figure 43 - H264 encoded video transmitted using FEC

However, once an implementation was complete, it was tested to compare the quality to the original version. As is shown in Figure 43, the quality was drastically improved, and no significant artefacts were visible, even when testing at range in an enclosed environment with multiple walls and doors between the transmitter and receiver. After confirming that the quality of transmission had improved, the same basic test for latency was used to determine the latency of the system. Whilst slightly higher than the latency achieved with the analogue setup, averaging around 120ms, it was deemed adequate enough for this application.

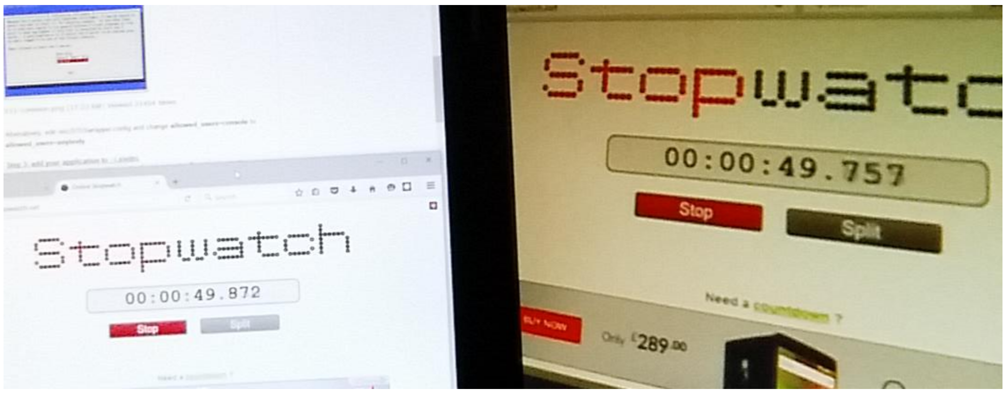


Figure 44 - Crude method for testing latency used

5.7.3 Interference Issues

As the digital video solution provided similar levels of latency when compared to the analogue video solution, yet at much higher resolution and bitrates, the digital video solution was chosen as the more suitable solution to take forward. However, during additional testing it became apparent that the 2.4GHz Wi-Fi adapters in use were causing major interference issues with the 2.4GHz RC receiver. This wasn't identified during initial testing, which is likely because the platform was in close proximity to the RC transmitter at all times during this stage of development.

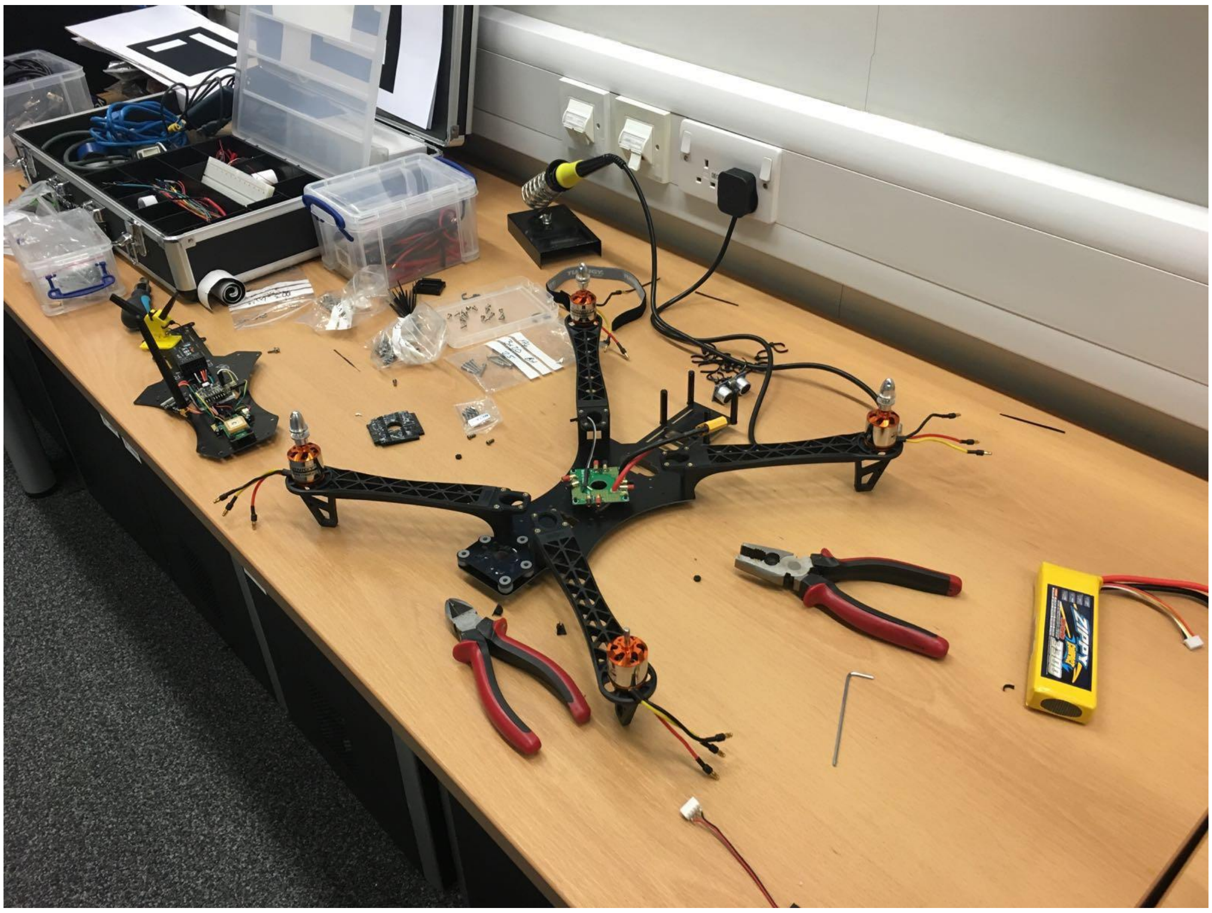


Figure 45 - Rebuilding the Spyda 500

The interference issues experienced resulted in irreparable damage to the multi-rotor platform, and as a result the platform had to be rebuilt. Whilst 3D printing techniques were used to develop the first platform, a commercially available multi-rotor frame was chosen the second time round; this frame was chosen as it was a similar design, and thus still appropriate for the project. This decision was made in an effort to save development time, as whilst 3D printing does provide a method for rapidly prototyping parts, current commercially available printers are still relatively slow at printing larger components. The 3D printed multi-rotor frame originally used took over 24 continuous hours of printing to produce, which would have delayed the development of the project further. The development of the replacement platform was nearly identical to the original platform, with only minor differences in the assembly of the frame and mounting configuration of the components. One issue noticed with the original platform was the lack of options for mounting the PCB antennas for the RC receiver. These antennas are meant to be mounted at a 90 degree angle from each other, which was difficult to achieve on the original platform. When the platform was rebuilt 3D printing techniques were used to print an antenna mount, the FRSky X8R Holder (Thingiverse.com, 2015), in order to better mount the antennas.

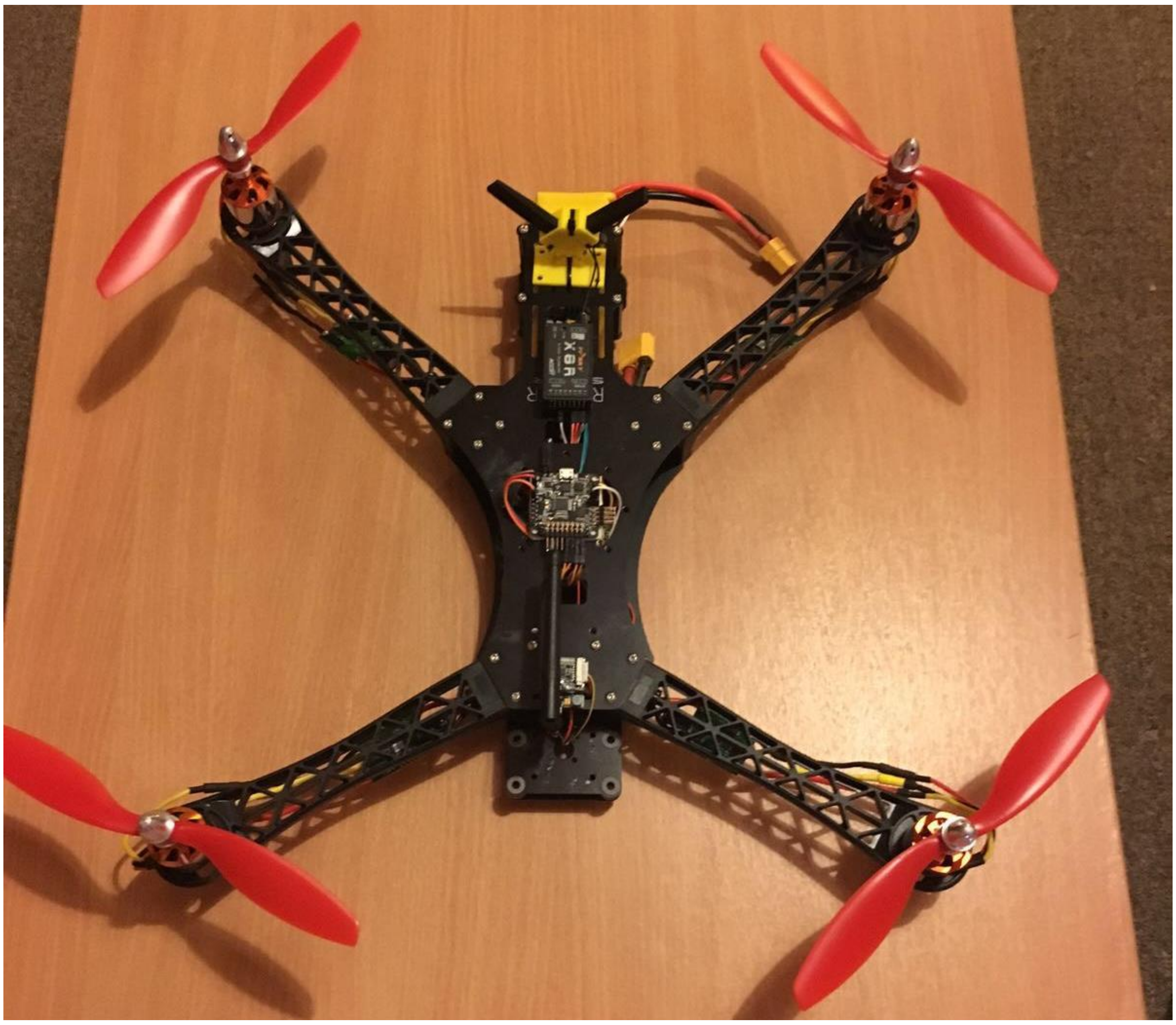


Figure 46 - Completed rebuild of the Spyda 500

5.8 Localisation

Once the ground work for the artefact had been laid, and the hardware and software needed for both the real world and the VR world were working, a method of localisation was needed in order to localise the coordinate systems of the two worlds. To fulfil this project objective, the requirements in Figure 47 were drawn up, in order to guide the design and implementation.

- Successfully localise the multi-rotor in the VR world.
- Display virtual objects so they appear as if they were physical to the user.

Figure 47 - Requirements for coordinate system localisation

5.8.1 Monocular Visual Odometry

One initial idea for localisation was to use a form of Visual Odometry, in order to localise the multi-rotor in relation to its start position. Visual Odometry is a process that is used to determine position and orientation of a robot or platform, through the analysis of associated camera images; two popular forms are Monocular Visual Odometry, using a single camera, or Stereo Visual Odometry, using either a stereoscopic or multiple cameras.

Due to the issues faced with video transmission in section 5.7.3 Interference Issues, analogue video transmission had to be used. As a result of this, the use of multiple cameras would have significantly increased the complexity of the hardware, requiring one transmitter, receiver and video capture device for each camera. A stereo camera was briefly considered, but dismissed due to both concerns over the resolution being insufficient when shared between 2 eyes, as well as the availability and cost concerns of acquiring a suitable stereo camera. This meant that the only option that was feasible was to use a Monocular Visual Odometry solution.

Forster *et al.* (2014) proposed a semi-direct Monocular Visual Odometry algorithm, for state-estimation of a micro-aerial-vehicle (MAV) in a GPS denied environment. The proposed method eliminated the need for feature extraction and robust matching techniques, by operating on pixel intensities, using a probabilistic mapping method to estimate 3D points, resulting in fewer outliers and more reliable points. The algorithm is described as precise, robust and faster than other current methods, especially in scenes of repetitive or little texture; Figure 48 shows how the algorithm supposedly detects features in scenes of high-frequency texture. An open source implementation the algorithm, known as Semi-direct Visual Odometry (SVO) (GitHub, 2016), is available on Github, and was chosen as the library to implement Monocular Visual Odometry in the project due to its development being aimed at the state-estimation of aerial platforms.

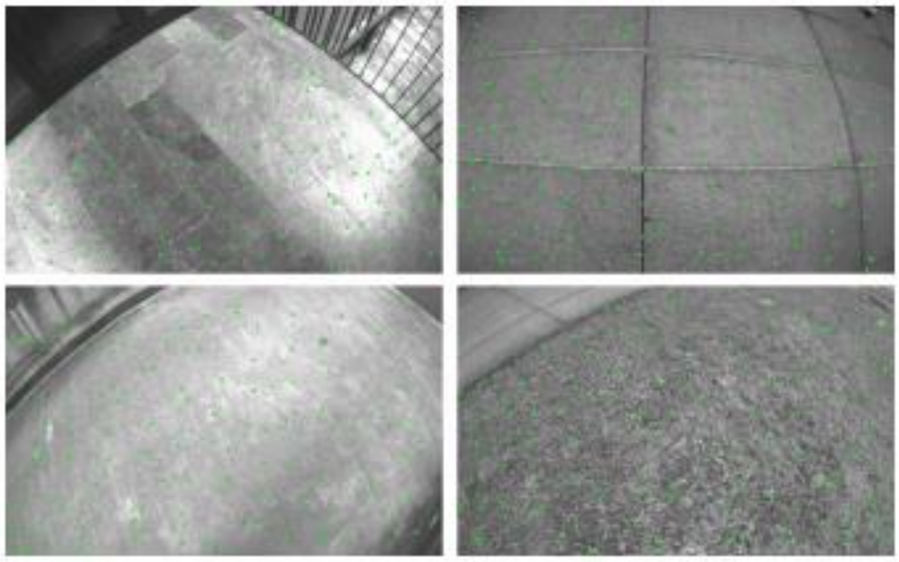


Figure 48 - SVO tracking features in scenes of high-frequency texture (Forster et al., 2014)

During implementation, several issues were encountered with SVO not working as expected; when implementing the library and using it to estimate the state of a camera, it led to erratic and inconsistent results. Significant development time was spent investigating these issues, in an attempt to identify the cause of the issue. Originally, an RGB webcam was used to capture the frames needed for the algorithm, and it was thought that both the low resolution, as well as the low frame rate could be the cause of the issue. To resolve this, various alternative cameras were tried, including those with both higher resolutions, as well higher frame rates. One of the cameras tried, the Point Grey Firefly MV (Ptgrey.com, 2016) shown in Figure 49, was a high-fps machine vision camera, specifically designed for a wide variety of imaging applications, and as a result should have been suitable for use with the algorithm. However, despite trying multiple different cameras, none of them were able to resolve the issue.

Figure 49 - Point Grey Firefly MV machine vision camera (Ptgrey.com, 2016)

Investigation into the source code provided and the debugging of the developed application revealed that the cause of the issue was the implementation in the open-source version of the algorithm. For unknown reasons, the implementation was unable only ever comparing subsequent frames to an original key frame, rather than intermittently updated key frames; this meant that as soon as the camera had moved more than a frame distance, all tracking was lost and highly erratic results were produced. At this stage of development, significant time had already been spent in an attempt to resolve the issue with SVO, resulting in the project falling behind schedule considerably. As a result of this, the use of Monocular Visual Odometry was abandoned, and other methods for augmented reality localisation were investigated instead.

5.8.2 Aruco

Due to the issues faced with the chosen implementation of Monocular Visual Odometry, another method of AR localisation was needed. From initial research and knowledge, AR markers were identified as a viable solution for localising VR objects in the real world. This approach is fundamentally different from using a localisation method, as rather than knowing where the multi-rotor is in the real-world and using that to augment objects, this approach uses markers that can then be identified in the captured frame. These markers are a two dimensional symbol, similar to that of a QR code, that can be uniquely identified and used to calculate transforms. These transforms are then used to move objects in the planar space of the marker, which makes them appear as if they were physical in the real world.

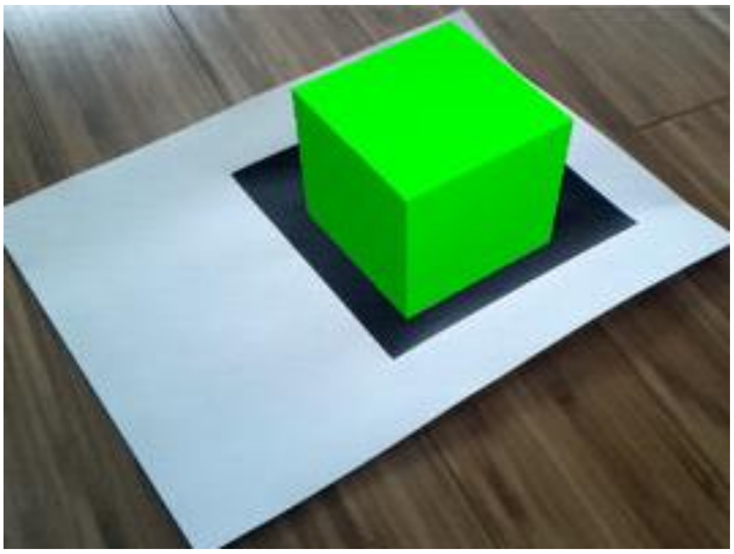


Figure 50 - Example of AR marker used to place virtual object in the real world

Two libraries were looked at and considered in order to implement this, both ARToolkit and Aruco, as discussed in section 5.8.2 Aruco. Aruco was the library chosen for this implementation due to its simplicity, as the development needed to move forward quickly. During the implementation, source code was written in order to successfully identify and outline AR markers in the real world. To do this, an RGB webcam was used to provide a video feed, and paper markers were printed for placement in the real world. Figure shows these markers placed in the real world, and then being correctly identified and outlined.

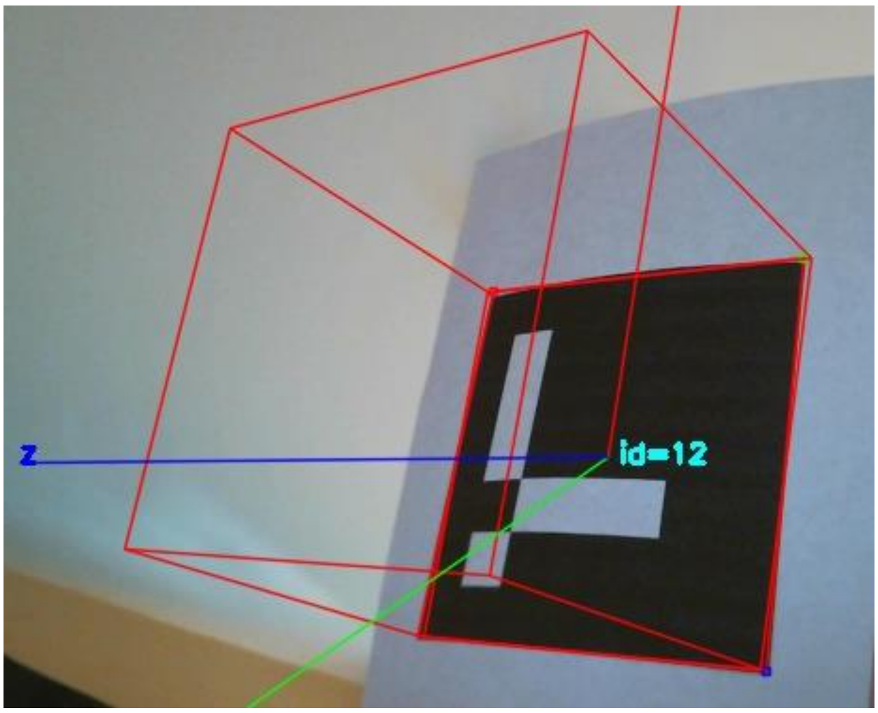


Figure 51 - AR marker being identified and outlined

Once markers could be identified, objects could then be drawn in their planar space, creating the illusion the object was in the real world. To achieve this, the transform from object space to planar space had to be calculated, which required knowing the intrinsic properties of the camera. In order to calculate these, a tool is provided that is used in combination with a special checkboard; the checkboard is held in front of the camera in different orientations and positions, and the resulted frames are used by the tool to calculate these parameters.

Difficulties were experienced during this part of the implementation with the transform matrix returned by Aruco; this at first did not appear to be working correctly, as when the transform was applied, the object was not appearing in the position it should have. This turned out to be an issue with the ordering of the matrices; OpenGL uses column-major matrix ordering, whereas OpenCV, on which Aruco is built upon, uses row-major matrix ordering. Once this issue was identified, and the matrix was transposed, objects were successfully displayed in the planar space of the marker.

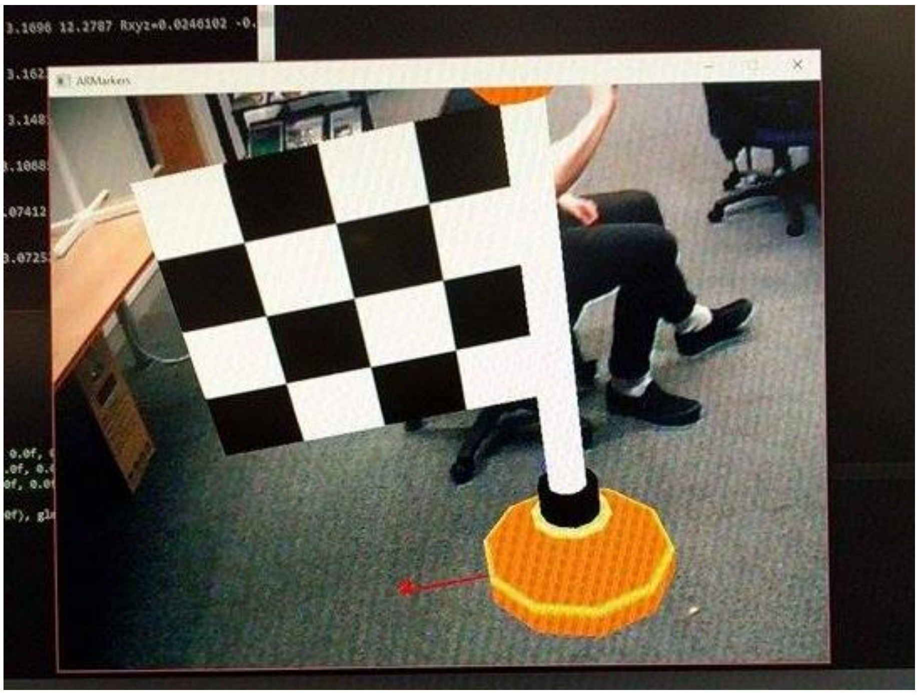


Figure 52 - AR marker used to draw virtual object in the real world

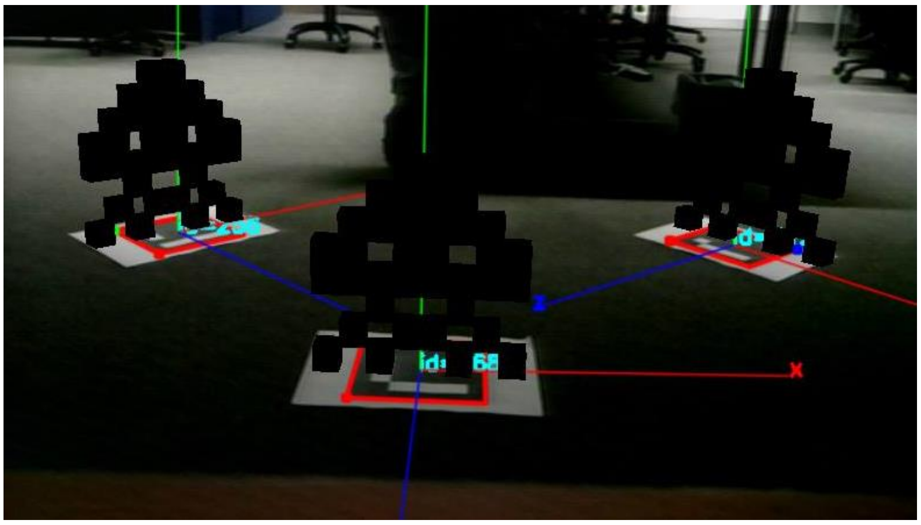


Figure 53 – Multiple objects drawn in the real world using AR markers

One issue noticed with this approach however was the loss of marker identification when the camera was moved, which was a serious concern as the intended application was a FPV multi-rotor. At first, this issue was thought to be due to motion blur inherent with the rolling shutter in the RGB webcam used, and so the camera on the multi-rotor platform shown in Figure 54 was used instead. This camera used a global shutter instead, meaning the amount of motion blur present in the video should be lower.

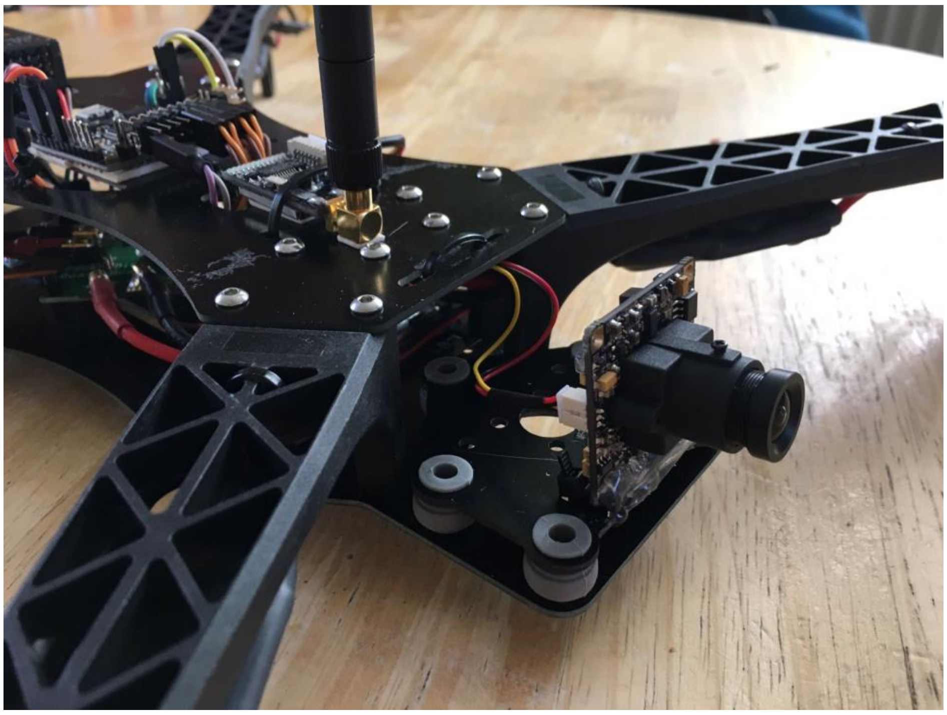


Figure 54 - Global shutter camera mounted on the multi-rotor

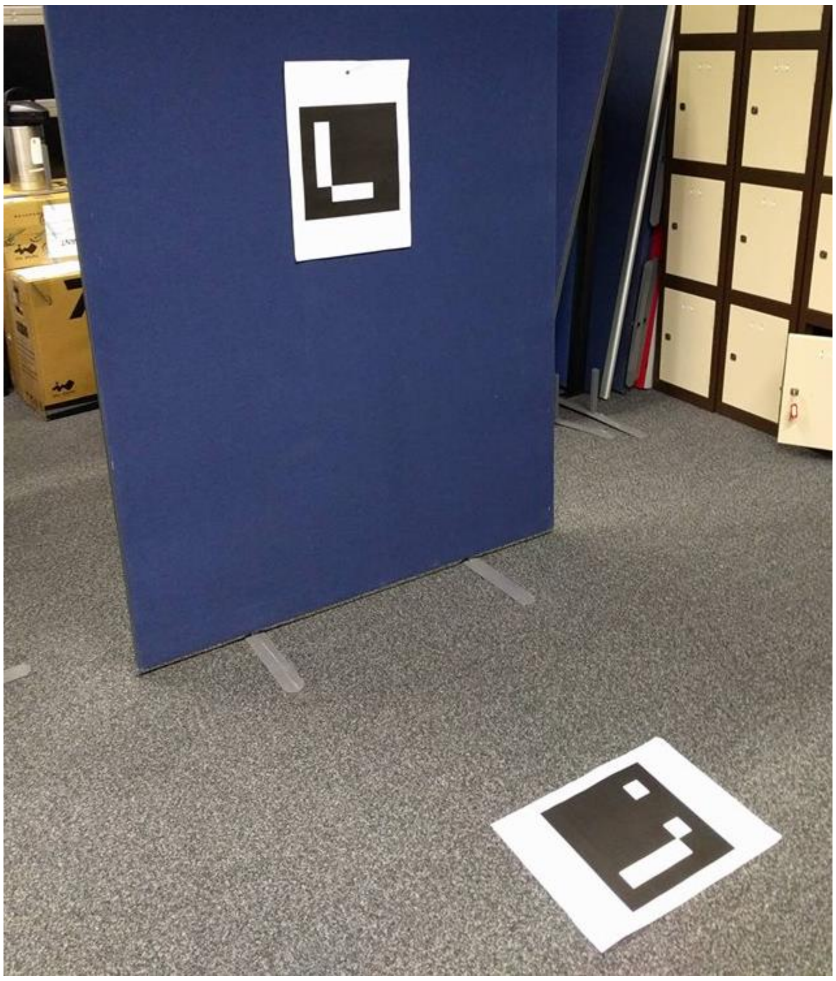


Figure 55 - Setup of Aruco markers used during testing with the multi-rotor

This was carried out using the setup shown in Figure 55, and the multi-rotor was flown in sight of the markers to test the identification of the markers when using the platform. Figure 56 shows how the marker was identified and augmented when the platform was at rest. The use of the global shutter camera however did not improve the ability for the library to track the markers, and the issue still remained; the vibrations and movement of the multi-rotor meant the markers could not be identified successfully. Other approaches were considered to solve this, including the use of a gimbal on-board the multi-rotor to smooth out sudden movements and vibrations. Unfortunately, due to the time left for completing the project, this was not possible as the added complexity would have stopped any form of final artefact being produced.

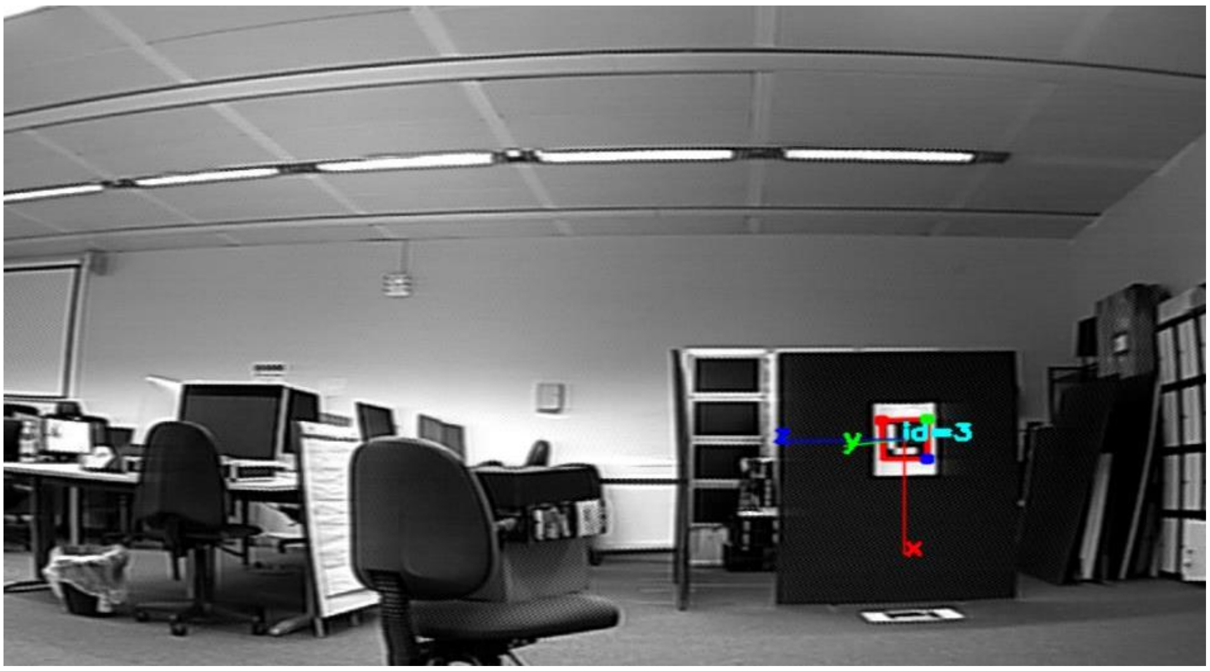


Figure 56 - AR marker identified and outlined from a multi-rotor when not moving

5.9 Final Implementation

Despite the issues with the AR markers not being identified properly, development was started on the artefact anyway, so at least a proof of concept was implemented. As this was only a first prototype, no design consideration was taken before starting to implement the artefact. Instead, the code developed so far was combined in order to display AR, using the AR markers from the previous section, in the virtual world developed using the Oculus Rift.

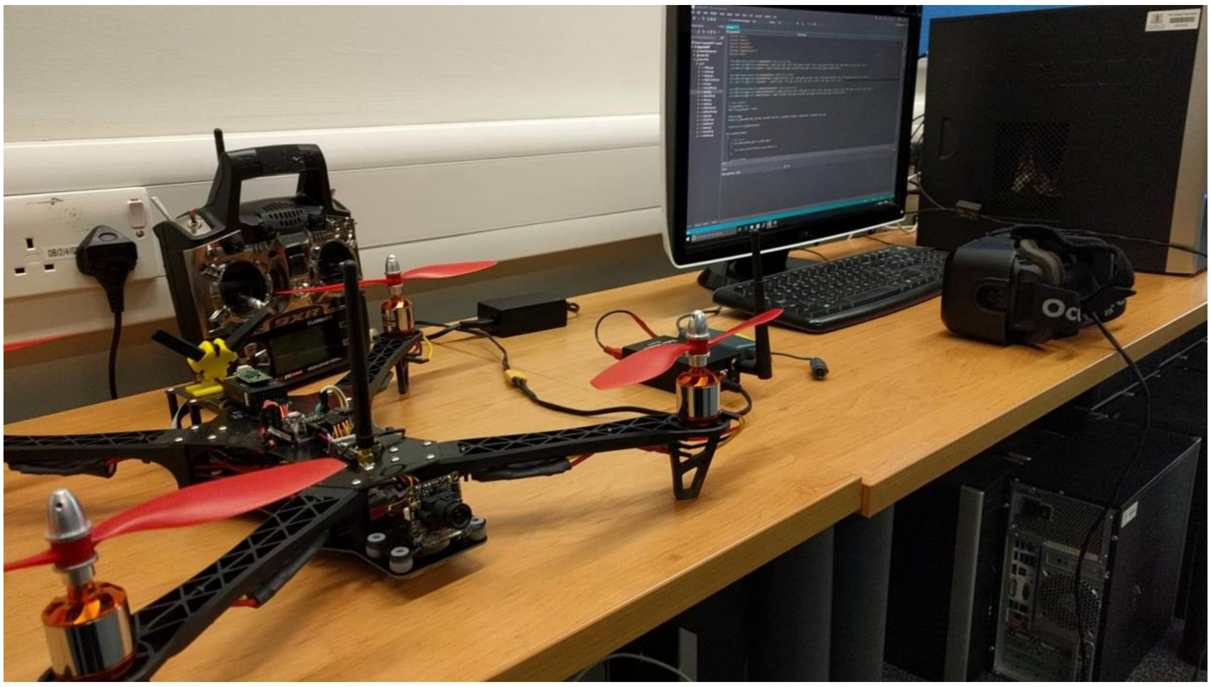


Figure 57 - Final hardware setup used to develop first prototype of artefact

Originally, the idea was to fully immerse the user in the real world, but this proved difficult as issues with the correct application and ordering of matrices from both the Aruco and Oculus meant that virtual objects would not appear as expected. In addition to this, the fact that only a single camera was used, instead of a stereo camera, meant that the same view point had to be used for each eye. This made the virtual objects appear unnatural, as the real world appeared flat whilst they appeared 3 dimensional.

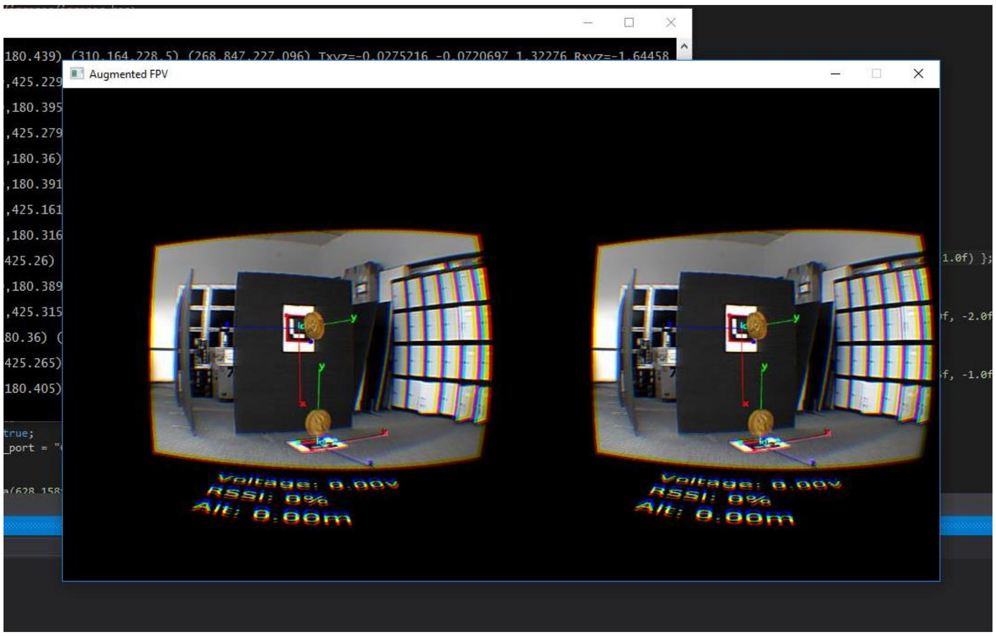


Figure 58 - Initial prototype of a virtual cockpit showing the augmented world

Instead, the augmented real world was presented on a flat plane inside of the virtual world, as if the user was in a virtual cockpit looking out a window at the augmented world, and telemetry information was provided when the user physically looked down using the Oculus head tracking. This is presented in an extremely basic and limited format as seen in Figure 58, as this was only meant to be the first prototype, used just to check all functionality correctly worked together, and would have required further iterations to develop further into the finished artefact.

6 Evaluation

6.1 Evaluation Methodology

Despite the project not reaching a stage suitable for formal evaluation, a methodology was considered during development to assess how users felt augmented reality improved their experience with an FPV multi-rotor. This evaluation method would involve the use of 2 user studies, both of which would use standardised questionnaires to gain participant feedback. Appropriate questionnaires to use would have included those typically used in areas of Games User Research which assess a participant's experience, in terms of immersion, enjoyment and engagement.

Once the user studies were carried out, the results collected from the participants would have been used to evaluate whether augmented reality improved their experience with an FPV multi-rotor. As standardised questionnaires would have been used, the two studies could be compared with each other, allowing conclusions to be drawn from the quantitative and qualitative data available. In addition to this, thematic analysis could have been used on any qualitative data gained, identifying key themes in the participants' responses for later discussion.

6.2 Results

As the project did not reach a stage suitable for participant testing, and therefore no formal evaluation can be carried out, the only method suitable for evaluating this project is self-evaluation. At the beginning of the project a list of objectives to be met, were detailed in section 3 Aims and Objectives. These objectives are the basis for the self-evaluation of the project, which has been carried out continuously through the development of this project; this is detailed where required in section 5 Development of Artefact.

Overall, compared to these objectives, the project met all those that were required, although not to the high standard expected from the onset of this project. However, the aim of this project was investigative development, and so despite not fully completing the objectives to build the final artefact, the project has met its originally intended aim.

7 Critical reflection

Critical reflection is a reasoning process used in order to understand the meaning and value of an experience. The Gibbs reflective cycle (Lia, n.d.) is a theoretical model that is often used as a framework for carrying out critical reflection, consisting of 6 distinct stages. The aims of using the Gibb's reflective cycle is to challenge assumptions, explore new ideas or approaches, and promote self-improvement and to link practice with theory. The Gibbs reflective cycle will be used here in order to critically evaluate the project undertaken.

7.1 Description

During the development and subsequent documentation of this project, investigation was undertaken into the possibility of using virtual reality technology, combined with the use a multi-rotor platform, in order to create an augmented reality FPV experience.

This involved the familiarisation with various VR and multi-rotor hardware, both for evaluation as well as development. A small multi-rotor platform was developed, although problems with control and lack of thrust prevented the platform being suitable. This was however subsequently used to develop software for communication, after which a second platform was developed using 3D printing, which better suited the development.

Both analogue and digital video transmission methods were investigated, during which issues arose with both methods and the second platform was considerably damaged and had to be rebuilt. A method for localisation of the multi-rotor using a single camera was tried, but despite numerous efforts the method would not work correctly. Instead, a method using AR markers was employed.

Several of these unexpected issues, required a new approach to be taken in order to move the project forward, which not only caused project slippage, but also affected the successful completion of some of the original aims laid out. As a result of this, some objectives of the project had to be dropped early on into the project, in order to focus on the main crux of the development. By the end of the development, an initial prototype meeting the objectives laid out was developed, however a further iteration would have been required in order to finish the artefact. In addition to this, issues were still present with the method used for augmented reality; these could however have been addressed given more time.

7.2 Feelings

Before the project, I felt very confident that I could successfully complete the project; this was due to both the project proposal grade, as well as my own confidence in skill and ability. Whilst I thoroughly enjoyed the undertaking of the project as a whole, and feel that I wouldn't change the experience, some parts were enjoyed much more than others.

I thoroughly enjoyed building the multi-rotor platforms, as I not only gained knowledge and experience from it, but also felt a sense of pride that I had built something that was capable of physically flying. On the flip side of this, I felt rather frustrated when the MHQ2 Hovership platform wouldn't fly, despite the numerous methods tried to resolve the issue. Other parts of the project that were enjoyable to me included much of the software development, as the language being used was C++. This was well known to me, and so I was able to write much better code than if an unknown language was used, something which affords a sense of pride.

When trying to implement the Monocular Visual Odometry library during which issues were encountered trying to get it working at all, I was both considerably frustrated and extremely worried the project wouldn't work out. At the same time to this, I also felt very determined to get the implementation working, something which later was detrimental to the progression of the project. At multiple times during the development, I felt exhausted from the amount of work put in, which often consisted of long nights working in the lab. I felt I had very little progress to show for all the time dedicated to the project in terms of meeting the project objectives, which was both disheartening and demotivating.

Once I had switched to using AR markers, I began to think that at least I would be able to finish at least most of the project objectives, though issues with motion blur and having to use an undesired method caused considerable frustration. Towards the end of development when I had an initial prototype for the artefact, I felt more confident that the project would at least be partially successful, despite the issues with tracking AR markers still being present. Overall at the end of the project I felt that I had learnt a considerable amount of technical knowledge from the project, despite not fully completing what I set out to achieve.

7.3 Evaluation

During the project, there were multiple parts that I consider to have gone well, resulting in several working components that met the requirements set out. One thing I think that went well was the development and implementation of the MSP protocol; this piece of software was rather straight forward to develop, and no real issues were encountered during its development. As well as this, I also think the development and build of the Spyda 500 platform also went well, and the platform flew very stable without any real problems configuring or tuning it. When developing the VR software, I feel that no real issues were encountered and that this phase of the project went well and completed on time. The last key part of the project I feel went well was the development of an analogue video transmission system. When developing this, the hardware used worked well first time, and there were minimal issues encountered as well.

As well as things that went well during the project, I also consider that some things went wrong, or didn't work. The first problems experienced during the project were the issues with the Hovership MHQ2 having inadequate thrust of stable flight; whilst the development didn't exactly go wrong, and a finished platform was produced, the end result did not function correctly as specified in the requirements. Another thing that went wrong during the project was the irreparable damage to the Spyda 500 platform, as well as the major issues experienced with the Monocular Visual Odometry libraries not working as expected. Other things that went wrong during the project include the issues experienced with the AR marker tracking, as well as the incomplete final artefact, and the fact only an initial prototype of the artefact was developed in time.

Overall, the project ended with only an incomplete initial prototype that whilst technically met the aim and objectives of the project, wasn't a complete solution and still required further work. Issues were still present in the artefact produced with the tracking of the AR markers.

7.4 Analysis

I think that the development of the MHQ2 multi-rotor didn't meet the requirements properly due to insufficient planning and design of the platform prior to the build. The components chosen weren't suitable, and as a result the platform wasn't able to fly. This led to delays later on in the project, as a second platform had to be developed in order to carry on development of the final artefact. This could have been avoided by spending more time designing the platform, ensuring that the components used were in fact adequate and suitable for the design.

The damage caused to the Spyda 500 platform was a result of issues with 2.4GHz Wi-Fi interfering with the 2.4Ghz transmitter and receiver used for control; this wasn't initially obviously as an issue, until the platform was used at greater range. As a result of this, additional expenditure for replacement components, as well as additional time to rebuild the platform was needed, resulting in the project slipping behind schedule. This could have been avoided by using a different frequency for control of the multi-rotor, and through better assessment of the risk of interference; testing could have been done to verify whether interference would be an issue, before the platform was used.

Issues experienced with the Monocular Visual Odometry libraries used were the result of poor and incomplete implementations of the algorithms available chosen, due to the fact that it was out of scope of the project to implement these algorithms from scratch. I think however, that too much time was spent trying to rectify these issues prior to moving on from them, which resulted in delays in the project and issues finishing the final artefact on time. This could have been mitigated better by accepting the fact the implementation was not going to work sooner, rather than wasting multiple weeks of development time continuously trying ways to fix the implementations.

As a result of the other issues, the final artefact produced was only an initial prototype rather than a final implementation. This was caused by both delays from previous issues, as well as time wasted on development not strictly needed for the project. I could have avoided this by better following the project plan, and not allowing as much feature creep and time waste during the development. In addition, the agile project methodology chosen allowed for this to happen, and so maybe a more rigid methodology would have help to prevent it.

When developing the MSP protocol implementation, I think that it went well because lots of time was taken to carefully consider the design of both the hardware and software required. The protocol was examined and researched in good depth, to ensure a full understanding of how it worked prior to development. The hardware used was also rather straight forward, and prior knowledge helped with the design and setup. This meant that the software could be used in the final artefact for providing visual feedback to the user about the multi-rotor, something which is beneficial when FPV flying. As well as this, the fact this part of the project went well helped to keep hope the project would be successful, despite issues encountered elsewhere. This could have been improved further by implementing the protocol as a standalone library, which would have made integration into the artefact more straight forward later on.

I think the design and build of the Spyda 500 platform went well, as adequate time was spent planning the build, as well as choosing components and ensuring they were suitable for the chosen frame. In addition, lots of time was spent looking at possible frame designs, and weighing them against the requirements list for the platform. This meant that once the platform was built, the multi-rotor would be suitable for use, and any issues with it would hopefully be minimal. Decisions were made early on and prior to the build that ensured the platform would be sufficient. This eased later development, as the platform accommodated changing requirement, especially when developing video transmission solutions; plenty of space on-board allowed for a Raspberry Pi to be used, something that was never originally considered.

The development of both video transmission systems also went well, and despite the issues experienced, both solutions did work, just each had a drawback associated. The development of both solutions went well because of both the consideration took before their development, as well as the ability to adapt to changes in their design and implementation afforded by the project methodology chosen. This led to more choice when choosing what video transmission system to use for the final artefact, and options if things changed later on. Despite going well, the solutions could have been improved by looking at possible ways to overcome their draw backs, different frequency Wi-Fi for example.

7.5 Conclusion

Looking back on my project, it is clear that from the offset it was both over scoped and ambitious, which was to the detriment of the project. Whilst initially it seemed plausible, parts of the project took longer than expected, or did not work, leading to slippage. Whilst the knowledge I gained during the project is invaluable, failures during the project affected motivation to continue development, and often when something went wrong, I refused to move on straight away. This resulted in the waste of a lot of development, as it was spent trying to solve issues rather than continue development.

Feature creep was also an issue during the project, and a large portion of the development time was spent developing something which wasn't even used in the final artefact, due to interference issues experienced. If I had followed the project plan more closely, and not lost sight of the goal, the overall aim of the project may have been better achieved. This project helped teach me about my own strengths, as well as my own weaknesses; the refusal to give up on something was damaging, and is something that I could have done differently.

During most of the project I tended to focus less on the overall aim of the project, and more on developing something interesting and fun. This again is something I learnt about myself during the project, and if I had kept focus on the project at hand rather than, being distracted by something I am heavily interested in, I would have benefited both myself and the project. Despite the project was not fully completed, all of the objectives set out were met at least to some degree, but not to the standard I would have liked.

7.6 Action Plan

If I were to undertake work similar to this again, there a several things that I would do differently in order to better achieve whatever aim is laid out. Firstly, it was evident during both development, as well as during this report, that I over scoped the project, picking an ambitious project that required everything to go correctly in order to achieve all the objectives laid out. Instead, a more relevantly scoped project would be chosen, focusing more on delivering a successful project, rather than over stretching what is possible in the given timeframe; picking something that would be successful over something that I would like to be successful.

In addition to this, my critical evaluation identifies that spending too much time trying to resolve issues in development negatively impacted the project. If similar work was done again, I would learn from this experience, and better evaluate when to move on when things don't go as expected. Focus during this project was often on parts deemed fun, rather than focusing on fulfilling the overall goal. This again negatively impacted the development, as it caused project slippage when too much time was spent on parts I enjoyed more. In order to improve from this experience, I would learn to apply better focus to the project as a whole, rather than specific parts of interest to me.

When things went wrong during the project, I often became demotivated to continue development, often without realising, because I wanted to use a specific implementation or method for the project rather than move on. This led to the issue mentioned above, which pushed the project further and further behind schedule. To improve on this, I would better realise when I was becoming demotivated, and address this issue earlier, learning when it is time to move on rather than sticking to what I would like.

8 Future Work

As the project was subject to strict time constraints, some objectives had to be pulled in order to meet the aim laid out. If given more time, I would first continue to address the issues with the final artefact, exploring better methods for localisation or AR markers, or methods and solutions to resolve the issue with the current implementation of AR markers. Along with this, iterative development of the final artefact would be continued, in order to develop a polished solution that could then be used for the assessment of using AR with multi-rotor in terms of user experience.

After this project, work would take the direction of assessing how valuable the development of AR is to the experience possible with multi-rotor. One of the original objectives was to assess this through the use of user studies, which would provide great insight into the benefits of using AR with multi-rotors, and whether it provides a better or more engaging experience for the user.

Other considerations for future work include directions that this investigation could take. This could include exploring other avenues and applications for AR and multi-rotor use, and rather than for entertainment and enjoyment, serious applications for this technology could be explored. On top of this, there are ethical considerations for the direction this work could head in; whilst currently limited in terms of what is possible, as technology improves this type of work could find itself used in areas such a military applications, who have already explored the adoption of VR and AR in other forms.

9 References

A.Quarter.To.Seven. (2015). MSP - The MultiWii Serial Protocol - A.Quarter.To.Seven. [online] Available at: http://www.stefanocottafavi.com/msp-the-multiwii-serial-protocol/ [Accessed 11 Apr. 2016].

Abeywardena, D., Pounds, P., Hunt, D. and Dissanayake, G. (n.d.). Design and Development of ReCOPTER: An Open source ROS-based Multi-rotor Platform for Research.

Agilemanifesto.org. (2016). Manifesto for Agile Software Development. [online] Available at: http://agilemanifesto.org/ [Accessed 11 Apr. 2016].

Al-Zoabi, Z. (2008). Introducing discipline to XP: Applying PRINCE2 on XP projects. pp.1--7.

Artoolkit.org. (2016). Open Source Augmented Reality SDK | ARToolKit.org. [online] Available at: http://artoolkit.org/ [Accessed 10 Apr. 2016].

AXELOS. (2016). What is PRINCE2? | PRINCE2 | AXELOS. [online] Available at: https://www.axelos.com/best-practice-solutions/prince2/what-is-prince2 [Accessed 13 Apr. 2016].

Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S. and MacIntyre, B. (2001). Recent advances in augmented reality. Computer Graphics and Applications, IEEE, 21(6), pp.34--47.

Blackmagicdesign.com. (2016). Blackmagic Design: Intensity. [online] Available at: https://www.blackmagicdesign.com/uk/products/intensity [Accessed 12 Apr. 2016].

Bondyra, A., Gardecki, S., G\kasior, P. and Kasi\'nski, A. (2015). Falcon: A Compact Multirotor Flying Platform with High Load Capability. pp.35--44.

Charvat, J. (2003). Project management methodologies: selecting, implementing, and supporting methodologies and processes for projects. John Wiley \& Sons.

Cmake.org. (2016). CMake. [online] Available at: https://cmake.org/ [Accessed 12 Apr. 2016].

Connex.amimon.com. (2016). CONNEX. [online] Available at: http://connex.amimon.com/connex [Accessed 12 Apr. 2016].

Cs.technion.ac.il. (2016). Forward Error correction (FEC) for SCTP in Satellite environment. [online] Available at: http://www.cs.technion.ac.il/Courses/Computer-Networks-Lab/projects/summer2001/SCTP/Final_report/sctp%20-%20final.htm [Accessed 12 Apr. 2016].

Datta, S. and Mukherjee, S. (2001). Developing a risk management matrix for effective project planning-an empirical study. Project Management Journal, 32(2), pp.45--57.

Developer.oculus.com. (2016). Developer Center — Home | Oculus. [online] Available at: https://developer.oculus.com [Accessed 12 Apr. 2016].

Dji.com. (2015). Phantom 2 - The Spirit Of Flight. [online] Available at: http://www.dji.com/product/phantom-2 [Accessed 11 Apr. 2016].

Dji.com. (2016). DJI Lightbridge - revolutionary 2.4G Full HD digital video downlink. [online] Available at: http://www.dji.com/product/dji-lightbridge [Accessed 12 Apr. 2016].

Dji.com. (2016). Inspire 1 V2.0 - Everything you need for aerial filmmaking | DJI. [online] Available at: http://www.dji.com/product/inspire-1 [Accessed 11 Apr. 2016].

Dropbox. (2016). Dropbox. [online] Available at: https://www.dropbox.com/ [Accessed 12 Apr. 2016].

Ecalc.ch. (2016). eCalc - the most reliable RC Calculator on the Web. [online] Available at: http://www.ecalc.ch/ [Accessed 12 Apr. 2016].

Forster, C., Pizzoli, M. and Scaramuzza, D. (2014). SVO: Fast semi-direct monocular visual odometry. pp.15--22.

GitHub. (2013). rmsalinas/aruco. [online] Available at: https://github.com/rmsalinas/aruco.git [Accessed 10 Apr. 2016].

GitHub. (2016). Build software better, together. [online] Available at: https://github.com/ [Accessed 12 Apr. 2016].

GitHub. (2016). cleanflight/cleanflight-configurator. [online] Available at: https://github.com/cleanflight/cleanflight-configurator [Accessed 12 Apr. 2016].

GitHub. (2016). uzh-rpg/rpg_svo. [online] Available at: https://github.com/uzh-rpg/rpg_svo [Accessed 10 Apr. 2016].

Git-scm.com. (2016). Git. [online] Available at: https://git-scm.com/ [Accessed 12 Apr. 2016].

Google.co.uk. (2016). Google Cardboard – Google. [online] Available at: https://www.google.co.uk/get/cardboard/ [Accessed 11 Apr. 2016].

Hamp, T. (2014). The Difference Between H.264 and MxPEG - VoIP Insider. [online] VoIP Insider. Available at: http://www.voipsupply.com/blog/voip-insider/the-difference-between-h-264-and-mxpeg/ [Accessed 12 Apr. 2016].

Ikeuchi, K., Otsuka, T., Yoshii, A., Sakamoto, M. and Nakajima, T. (2014). KinecDrone: enhancing somatic sensation to fly in the sky with Kinect and AR. Drone. p.53.

Karlesky, M. and Vander Voord, M. (2008). Agile project management. ESC, 247(267), p.4.

Krajn\'\ik, T., Von\'asek, V., Fi\vser, D. and Faigl, J. (2011). AR-drone as a platform for robotic research and education. pp.172--186.

Lansdowne, Z. (1999). Risk matrix: an approach for prioritizing risks and tracking risk mitigation progress. Proceedings of the 30th Annual Project Management Institute, Philadelphia, PA, October, pp.10--16.

Lia, P. (n.d.). Using Gibbs' Reflective Cycle. 1st ed. [ebook] London: King's College London, pp.1-5. Available at: https://www.kcl.ac.uk/campuslife/services/disability/service/Using-Gibbs-Reflective-Cycle-in-Coursework.pdf [Accessed 13 Apr. 2016].

Ludlow, B. (2015). Virtual Reality: Emerging Applications and Future Directions. Rural Special Education Quarterly, 34(3), p.3.

Multiwii.com. (2016). Multiwii Serial Protocol - MultiWii. [online] Available at: http://www.multiwii.com/wiki/index.php?title=Multiwii_Serial_Protocol [Accessed 11 Apr. 2016].

Netbeans.org. (2016). Welcome to NetBeans. [online] Available at: https://netbeans.org/ [Accessed 12 Apr. 2016].

NVIDIA Developer. (2015). NVIDIA GPUDirect. [online] Available at: https://developer.nvidia.com/gpudirect [Accessed 12 Apr. 2016].

Oculus.com. (2016). Oculus Rift Development Kit 2 (DK2) | Oculus. [online] Available at: https://www.oculus.com/en-us/dk2/ [Accessed 11 Apr. 2016].

Onedrive.live.com. (2016). Welcome to OneDrive. [online] Available at: https://onedrive.live.com/about/en-us/ [Accessed 12 Apr. 2016].

Ptgrey.com. (2016). Firefly MV USB 2.0 cameras for industrial, life science, traffic, and security applications. Point Grey USB 3.0, Gigabit Ethernet and FireWire Machine Vision Cameras. [online] Available at: https://www.ptgrey.com/firefly-mv-usb2-cameras [Accessed 13 Apr. 2016].

Ramasubramanian, V. (2015). Quadrasense: immersive UAV-based cross-reality environmental sensor networks.

Raspberry Pi. (2016). Raspberry Pi 1 Model B. [online] Available at: https://www.raspberrypi.org/products/model-b/ [Accessed 12 Apr. 2016].

Ruparelia, N. (2010). Software development lifecycle models. ACM SIGSOFT Software Engineering Notes, 35(3), pp.8--13.

Stylusinc.com. (2016). Software Development Life Cycle (SDLC), Process & Business Models | Stylusinc- Technology Consultants and Solutions. [online] Available at: http://www.stylusinc.com/BI/the-software-development-life-cycle-sdlc/ [Accessed 11 Apr. 2016].

Support.riverbed.com. (2016). Riverbed AirPcap. [online] Available at: https://support.riverbed.com/content/support/software/steelcentral-npm/airpcap.html [Accessed 12 Apr. 2016].

TCPDUMP. (2016). TCPDUMP/LIBPCAP public repository. [online] Available at: http://www.tcpdump.org/ [Accessed 12 Apr. 2016].

Team-blacksheep.com. (2016). Team BlackSheep Online Store - TBS DISCOVERY (top / bottom plate). [online] Available at: http://www.team-blacksheep.com/products/product:98 [Accessed 11 Apr. 2016].

Thingiverse.com. (2013). Spyda 500 Quadcopter by Gyrobot. [online] Available at: http://www.thingiverse.com/thing:160607 [Accessed 13 Apr. 2016].

Thingiverse.com. (2014). Hovership MHQ2 by Hovership. [online] Available at: http://www.thingiverse.com/thing:511668 [Accessed 11 Apr. 2016].

Thingiverse.com. (2015). Frsky X8R Holder - Hinged (BC Chipmunk 170) by Bungeecow. [online] Available at: http://www.thingiverse.com/thing:981288 [Accessed 13 Apr. 2016].

Thingiverse.com. (2016). MakerBot Thingiverse. [online] Available at: http://www.thingiverse.com/ [Accessed 10 Apr. 2016].

Thon, S., Serena-Allier, D., Salvetat, C. and Lacotte, F. (2013). Flying a drone in a museum: An augmented-reality cultural serious game in Provence. pp.669--676.

Turnigy9xr.com. (2016). Turnigy® 9XR Transmitter. [online] Available at: http://www.turnigy9xr.com/ [Accessed 11 Apr. 2016].

Visualstudio.com. (2016). Overview of Visual Studio 2015 Products. [online] Available at: https://www.visualstudio.com/en-us/products/vs-2015-product-editions.aspx [Accessed 12 Apr. 2016].

Wen, M. and Kang, S. (2014). Augmented Reality and Unmanned Aerial Vehicle Assist in Construction Management. Proc., Computing in Civil and Building Engineering (2014), pp.1570--1577.

Winpcap.org. (2016). WinPcap - Home. [online] Available at: http://www.winpcap.org/ [Accessed 12 Apr. 2016].

Xbox.com. (2016). Xbox 360 Controllers and Remotes. [online] Available at: http://www.xbox.com/en-GB/xbox-360/accessories/controllers [Accessed 11 Apr. 2016].