INTERDISCIPLINARY GROUP DESIGN PROJECT

UNIVERSITY OF EDINBURGH

SCHOOL OF ENGINEERING

Design of Microsystems - Group 1 Final Report

Designers:
Matilda Bruce, Benjamin Young
Jake Corkhill, Mikhail Gorbunov
Wilson Mojilip, Vojtech Pavlis

Project Coordinator:
Kiran Hosein

System Engineer:
Tej Bhatti

November 12, 2023

Declaration: The work submitted in this report is original except where stated otherwise.

Word Count: 19175

THE UNIVERSITY *of* EDINBURGH

Abstract

The following is an investigation into an autonomous drone system built on the architecture of a Parrot Mambo drone. This was designed to satisfy client requirements by retrieving, transporting, and depositing an egg while navigating a flight course. The implemented design of the machine vision, control systems, and mechanical aspects of the project are detailed. This is supported by an ethical analysis of current drone usage alongside a proposal for future developments following, among others, an open-source methodology.

The drone systems were successful individually with the drone achieving autonomous flight through the entire course in the simulated environment, and successful path following and landing on both straight and curved line sections in classroom 2. Moreover, the egg handling systems were effective in maintaining control over the egg through, take-off, flight, and landing.

However, the implemented design was too computationally intensive for the drone hardware and consequently full systems integration was untenable with the flight and egg manipulation systems remaining separate.

To fulfil client requirements, an open-source design of the drone is proposed encompassing the recommended hardware framework and software development environment. As such, considering this report in conjunction with the attached software files, grabber, and platform, the client can expect a fully-functional drone upon the next design iteration via the proposed system.

Contents

List of Figures

List of Tables

1 | Introduction

1.1 Motivation

The following report details the design of an autonomous drone system based on the Parrot Mambo [1]. The project was performed to meet the client requirements of engineering an autonomous free-flying drone enabling navigation of a course while collecting, transporting, and depositing an egg.

1.2 Background

Underpinning this project is the concept of model-based design [2]. Development is decomposed according to the "V model" [3] and facilitated by the Parrot Mambo and its ability to be programmed through MATLAB and Simulink. This provides a framework to build upon, shifting focus to the design concept itself, and allowing meaningful progress to be made within the 8-week development timeframe.

1.3 Concept

Development was divided into 3 sub-teams: Machine Vision (MV), Control Systems (CS), and Mechanical Design (MD), with the support of a Systems Engineer (SE), and a Project Coordinator (PC). The design concept is based on the goal of each team and the interaction of their contributions.

1.3.1 Machine Vision Concept

The MV team's goal is to generate targets for the drone to follow based on image data gathered from the drone's camera. This decomposes the course into its fundamental components: the path and the landing circles. The path necessitates line detection which is characterised by the identification of the start and endpoint of line segments and their associated direction vector. Whereas circle detection identifies circle presence via circumferences and outputs the coordinates of its centre.

1.3.2 Control Systems Concept

The CS team's goal is to move the drone to its target position. This is conceptualised as minimising an error between the drone's current and intended positions by actuating the drone's control mechanisms, thereby guiding it through the course. The course is staged using a finite state machine (FSM) which identifies the mode of operation the drone should be in for a given section of the course and applies the appropriate control loops.

1.3.3 Mechanical Design Concept

The MD team's goal is to implement a grabber to pick up a quail egg, maintain control during flight through the course, and deposit it at a target. To achieve this, the supplied grabber is evaluated while a base is implemented to hold the egg and index the drone. These facilitate the core client requirements of the project and augment the MV and CS designs, allowing repeatable egg collection.

1.3.4 Shared Design Concept

Machine vision target data is processed by the FSM where a stage of flight is determined. For general flight (path following) the drone position is incremented using the direction vector for the current line segment. Similarly, for landing, the drone's coordinates are incremented until they become the circle's centre coordinates. During landing, the MD developments correct for control imprecision while on base they ensure that the egg is reachable. Combined, these increase the error tolerance of the system.

System integration is managed by the SE who takes leadership of the project from a design standpoint. This allows a unified concept to be compared against the project's requirements, reducing the likelihood that elements will be overlooked.

1.4 Report Structure

An overview of the project and design will be presented via an executive summary. The design process will then be described for each sub-team. Findings will be presented and contrasted against the sub-team requirements enabling sub-system level verification and overall system validation. An ethical review of drone usage will be conducted highlighting the current state of commercial drones and the obligations the client therein. The employed system will be contrasted against open-source alternatives to evaluate development and suggest potential improvements. Finally, conclusions will be made as to the effectiveness of the project and the product.

1.5 Intended Outcome

Through this report, the client will gain an understanding of the societal background surrounding the commissioned work. They will also understand the product's design to better understand its operation. Alternative methodologies will also be presented and compared to the architecture employed. These factors elucidate the design process engendering confidence in the delivered product.

2 | Executive Summary

2.1 Introduction

Meeting client requirement in this project means engineering an autonomous drone capable of picking up, transporting, and delivering an egg. The Parrot Mambo architecture was used as the development platform to facilitate this.

The design approach and concept were to work in 3 distinct sub-teams: Machine Vision (MV), Control Systems (CS) and Mechanical Design (MD) with the support of a Systems Engineer (SE) and Project Coordinator (PC). The proposed drone design would generate spatial targets based on the MV algorithms feeding into a finite state machine. This would stage the flight in turn controlling the PIDs during flight phases and the grabber while landed, this is augmented through the MD systems.

2.2 Responsibilities Matrix

Table 2.1: Responsibilities Matrix

Member	Team/Role	Responsibilities
Matilda Bruce	Machine Vision	Investigation and implementation of circle detection. Design of new line detection algorithm.
Benjamin Young		Investigation and implementation of initial line detection systems. Optimisation and integration with CS.
Jake Corkhill	Control Systems	Incorporation of MV inputs, designed FSM, and created simulated testing space and trials.
Mikhail Gorbunov		PID tuning, sequencing landing, and implemented CS code onto drone hardware.
Vojtech Pavlis	Mechanical Design	Grabber mechanism analysis, landing platform design and testing. Developed precision landing mechanism.
Wilson Mojilip		Investigation of grabber control capabilities. Designed, manufactured, and tested landing platform.
Tej Bhatti	Systems Engineer	Ongoing review of requirements and their fulfilment, open-source research.
Kiran Hosein	Project Coordinator	Organisation, client interaction, scheduling/ monitoring progress, ethical analysis, and open-source review.

2.3 Research Overview

MV: Several algorithms have been developed in the past that aim to extract path features for autonomous vehicle operation. However, these endeavours have had the luxury of high-speed processing hardware by utilising open-source development platforms. The past algorithms are computationally intensive, although the hardware employed for these research systems can perform such processes in real-time.

The difficulty in this project is to implement a similar system on the limited drone hardware. Initial MV research focused on these previously utilised algorithms e.g. developing the line processing algorithms such as the Hough transform method [4]. Whereas, later revisions of the algorithms focused on computational efficiency. Research centred around finding the least computationally intensive MATLAB functions and operations [5] to replicate

the results of these initial research systems alongside down-scaling the optical input and correcting the errors that this may produce.

CS: Proportional-Integral-Derivative (PID) controllers are the primary control mechanism for the drone. Extensive research has been done to tune these optimally for implementation on the drone [6]. A PI-PD approach was investigated with supplementary investigations into Fuzzy PI-PD structures in linearising the control model of the Parrot Mambo Minidrone also being conducted.

A model-based design approach was researched [7] to effectively work in the MATLAB and Simulink environment which caters to this design methodology [2]. This allowed the effective simulation of the drone system which augmented physical testing and was key in producing an effective flight system.

Research into a 6DoF flight model was conducted to simplify the scope the envisioned control systems. Through this investigation it was determined only motion in the x-y-z planes was relevant and as such the 6DoF model could be reduced by half.

MD: The specificity of the egg handling task required more testing and evaluation on the team's part rather than research into grabber and platform systems already developed. Research was done to advise the construction of testing apparatus utilising components that were available at home. The extent of outward research was an investigation into the design specification of the grabber and drone themselves for use in calculations and analytical models.

Supplementary research into aerodynamic effects that occurred close to the ground during landing was also conducted. This determined the presence of vortex generation causing drone instability and distance to the down-wash moving the egg [8]. A supporting study of an electromagnet system to aid in drone centring was undertaken with the components of the envisioned system being detailed [9].

2.4 Design Overview

Illustrated in Fig. 2.1 is the system level diagram. This shows the distinction between the different systems that comprise the drone and their main sub-systems. Dependencies are marked by the direction in which the arrows point, showing that the output from one domain of the system is the input to another.

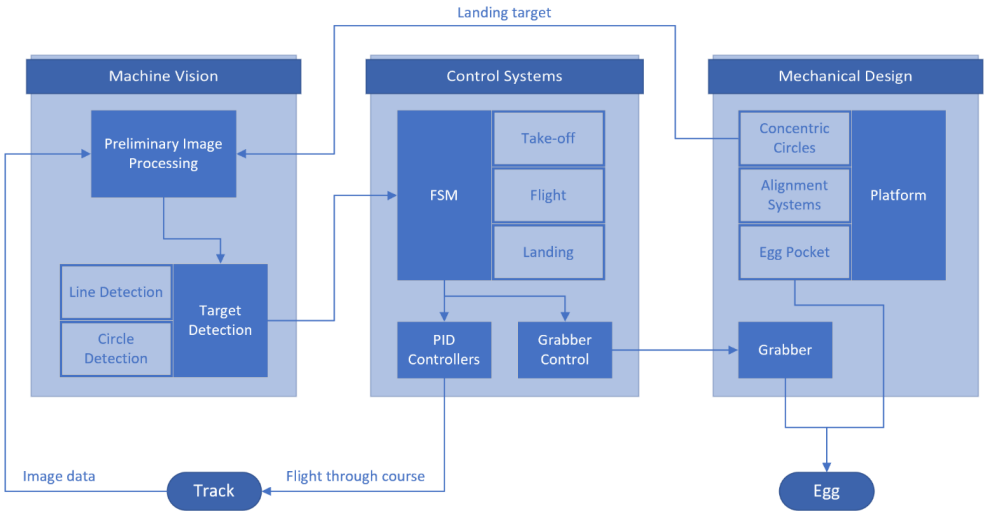


Figure 2.1: System level design overview

2.5 Testing Overview

Due to time and space limitations, the majority of MV and CS testing occurred in simulation. These provided an idealised representation of drone performance. The simulation was optimised to provide a realistic testing environment with a to-scale replication of the assigned track constructed in the simulation environment. This allowed testing to be done with both straight and curved line track segments and with correctly scaled circles. However, when implementing the tested algorithms on the drone hardware, the deficit in processing power required the MV algorithms to undergo considerable optimisation.

Also, due to factors that could not be simulated like processor performance and aerodynamic effects close to the ground, further laborious refactoring of the simulation-validated code was necessitated. For example, this took the form of PID tuning to respond to the discrete inputs caused by the drone's low refresh rate. Moreover, testing was also dependent upon the time of day (as well as lighting conditions), the amount of free space available, and the state of the drone's battery.

Tests on the proprietary grabber involved construction of apparatus like a spring balance and a crush rig. Combined with flight testing of the drone with the grabber, these demonstrated its efficacy in picking up, maintaining control, and depositing the egg. This rendered it unnecessary to develop a specialised grabber. To complement this, the designed platform was tested under take-off and misaligned landing conditions to assess its grip on the egg as well as its centring capacity where it proved effective for both.

2.6 Results

A cursory insight into drone flight performance can be gleaned from the simulated performance of the path following system. The drone completes the full track in a time of 25.80s with a percentage match to the lines of the track of 42.59%. Combined with the 43% reduced settling time from the tuned PID controllers, the drone exhibits fast and stable autonomous completion of the full course.

To assess the optimisation of the MV algorithms that contributed to this, note a reduction in camera resolution from 160x120 (native) to 48x36 pixels (optimised). Combined with the development of an alternative to the Hough transform method, the core vision sub-system of line detection executes 9.2x faster than initially developed.

Finally, the grabber exhibited a gripping force of 0.7N, which was enough to hold the egg securely without exceeding its 6.19N crush force. This is allied with the platform which held the egg under the drone's take-off thrust of 0.5N, and the precision landing systems that were demonstrated to be effective during the presentation and demonstration.

These results validate the drone by confirming that the individual system requirements have been met.

2.7 Discussion and Potential Improvements

Separately, the drone systems are effective in meeting the client requirements. Autonomous flight was successfully demonstrated meaning that specific functions, such as line detection, identification of the landing target, and the sequencing of the FSM operated as intended. Also, egg handling was effective with the demonstration and presentation showing the egg and grabber were reliably indexed and that the grabber could maintain control over the egg throughout all flight phases.

However, when integrating these systems into one cohesive package, the drone's hardware is a limitation. The algorithms of the separate systems are already at their limit of optimisation and a further reduction in their scope would make them cease to function. As such, the delivered drone does not meet the egg delivery requirement.

Through a proposed open-source design, hardware than can effectively execute the developed algorithms is described. This is compounded by refinements to the platform, edge case handling by MV, and refinement from CS. This allows for future developments to be made with the proposed implementation ultimately addressing all the client's requirements.

3 | Implemented Design

3.1 Machine Vision

The role of the machine vision team was to use the drone's inbuilt downward-facing camera to implement circle detection and line following algorithms. The outputs from these algorithms are the errors from the setpoint, *dx* and *dy*, and a signal indicating a circle has been detected. These are then passed onto the control system where they dictate where the drone should move.

3.1.1 Method

To begin the design of the MV system, the MATLAB Parrot MiniDrone Support [5] was analysed. This clarified certain features of the drone, such as the size of the output from the downward-facing camera, as well as giving examples of simple object detection.

The next point of research was on the Hough transform, which is a technique used in image processing to detect lines and circles [4]. MATLAB has a built-in Hough transform function for circle detection called 'imfindcircles', and there is a function on MATLAB's file exchange website called 'circle_hough.m' [10] that was also suitable for use in this system. The function, 'circle_hough.m', requires the number of circles detected to be pre-set, and a second function 'circle_houghpeaks.m' is needed to analyse the results from the first. This led to 'imfindcircles' being chosen, even though it is an in-built function, and therefore cannot be edited easily. For line following, the in-built functions 'houghpeaks' and 'houghlines' were used due to the limited choice of line following algorithms. Rudimentary circle and line following algorithms were written, and these were tested using screenshots of the track provided.

To begin implementing the machine vision algorithms on the drone, a template provided in the Parrot Mambo add-on was used. The template, called "parrot vision external mode", takes advantage of the monitor and tune feature on Simulink designed for monitoring signals from a model while running on hardware real time. This feature was used extensively during development and allowed for code to be run on the drone while the live output was monitored through the video viewer and scopes. An example video viewer output is shown below (Fig. 3.1). The template provided did not include any flight code so drone can be held over objects of interest unlike other flight templates. This allowed for concurrent development of control and machine vision system, because the flight control did not need to be developed before testing of the machine vision system could begin. It can be seen that this algorithm can effectively detect red objects (in white) as shown by the partial track and circle.

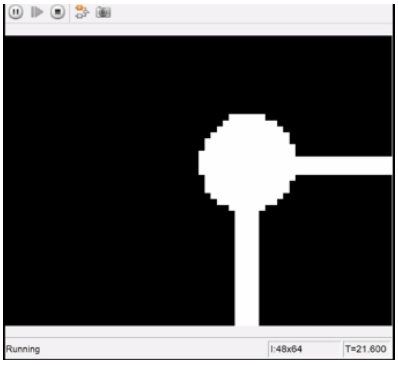


Figure 3.1: Example of video viewer output

The next stage was to examine the designed code in contrast with the design requirements to check if they were satisfied. This meant that, in the case of line following, the error between the detected line and the current drone trajectory must be calculated and output. For the circle detection, this meant that a *true* signal should be output when a circle has been detected. At this stage, these requirements were found to have been met. A final requirement of the drone was that it was to be flown at a height of 1m. To convert the output error from units of pixels to units of metres, a metre per pixel conversion coefficient had to be found. This was done by running an external simulation with the drone, which allowed the output of the drone to be displayed on a laptop. The drone was held at a height of 1m, and a tape measure was extended over the drone's visual range. This distance was measured and then divided by the appropriate resolution, to give a value of 0.007m per pixel.

After these were finished, and testing on the Simulink simulation began, it became clear that many changes were needed. Some errors were encountered when adding MATLAB functions to Simulink such as the unknown size of signals when compiling, therefore dynamic memory allocation had to be enabled. Another error was that, because Simulink's test track had a sharp right turn, the constant forward motion that had been implemented into the line detection code had to be scrapped. In its place, a series of if-statements were inserted to handle this edge case. Once the actual track was imported into the simulation, the curve in the track was registered as a circle, and would cause the drone to transition into the landing state. To combat this, restrictions were placed on the maximum radius that could be searched for. Lastly, the Hough transform line detection did not function, as it was too computationally intensive. This meant that an alternative line detection code had to be designed. The resulting code, that used only basic MATLAB functions and simple algebraic operations, was found to be almost 10x faster than the original, and could function on the simulated drone without issue.

The final stage involved altering the MV system after testing on the hardware. The first issue uncovered was that when running Simulink code with MATLAB function blocks, the fps of the drone was drastically lower than what was expected after simulation. The functions were causing the drone to process images much slower -the video viewer was updating very slowly- most likely due to a lack of processing power. To rectify this, a custom MATLAB function block was added that decreased the resolution of the image. Initially, a scale factor of 0.3 was tested, changing the native 160x120 resolution to 48x36. Another problem found was that the drone flew at a height of 0.8m when instructed to fly at 1m. This meant that the metre per pixel conversion coefficient had to be adjusted accordingly. Through trial and error, a value of 0.005m per pixel was found to remedy the situation.

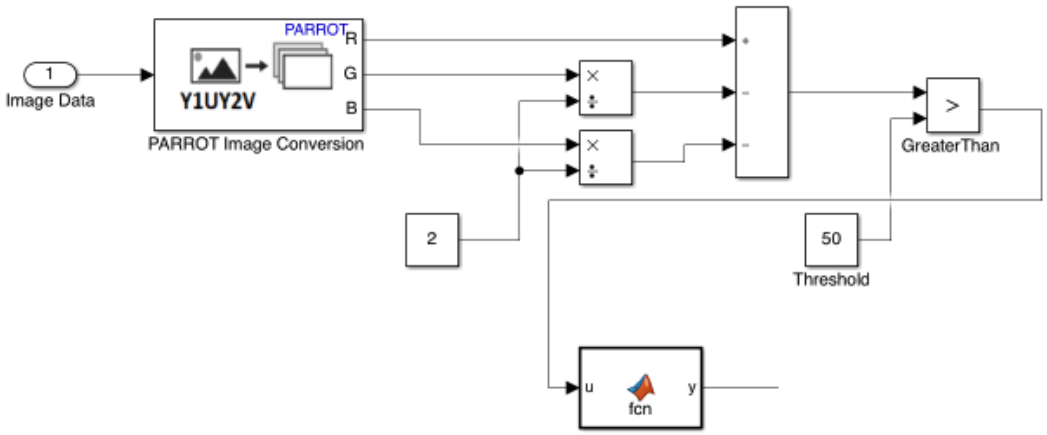


Figure 3.2: Pre-processing section of MV Simulink model

3.1.2 Design Specifications

The same pre-processing is performed on the drone's camera feed for both the line following and circle detection algorithms. This is shown in Fig. 3.2 above.

The first stage involved taking the input from the drone's camera, which was a series of Y1UY2V images and converting these into RGB format. This RGB image is then transformed via a threshold calculation that finds the pixels that have red as the dominant colour. This output of this is a Boolean image, shown in Fig. 3.3, where true denotes a red pixel and false denotes otherwise. Following this, a MATLAB function, 'imresize', is used to decrease the resolution of the image to 0.3 times the original size, which has the advantage of improving the processing speed. The drastic reduction in resolution is needed as the drone has a very limited processing power.

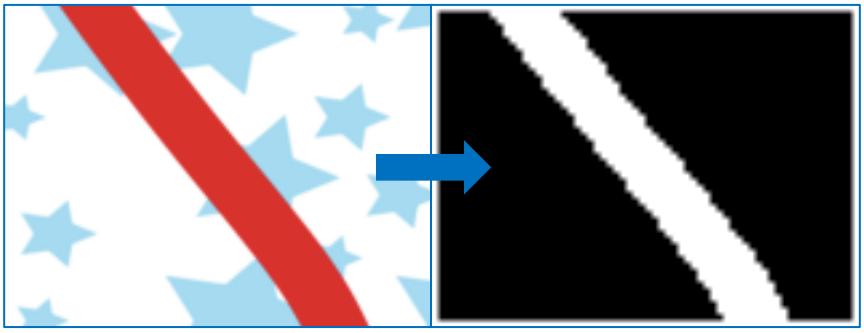


Figure 3.3: Line segment before and after pre-processing

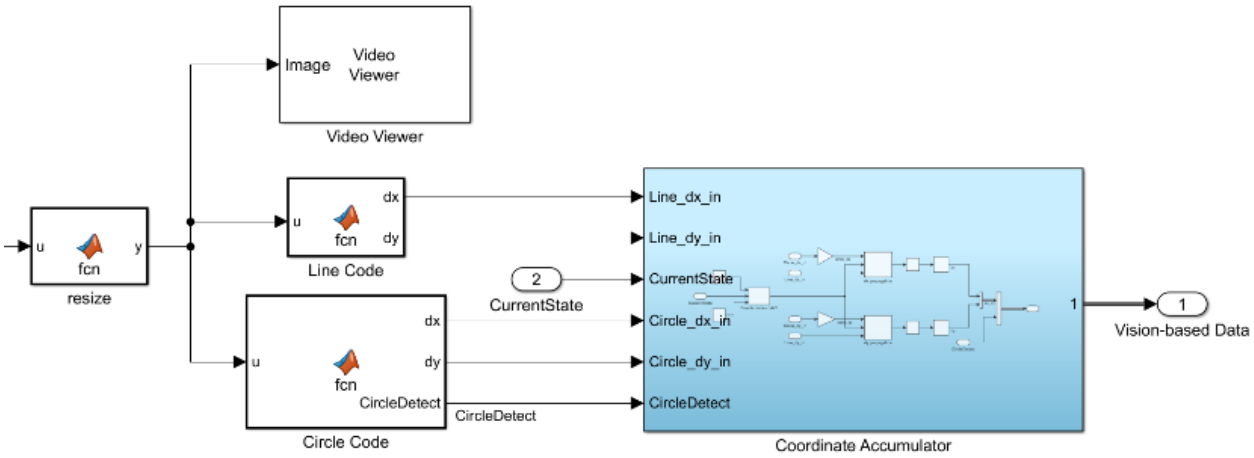


Figure 3.4: Processing section of MV Simulink model

3.1.2.1 Circle Detection

As shown in Fig. 3.4, the resized image is then fed into *Circle code*, the custom circle detection MATLAB function block. This circle detection code makes use of the function 'imfindcircles', with a specified radius range. The maximum radius is calculated by finding the size of the input image, and the minimum radius set to 10 pixels, to stop any artefacts from being wrongly identified as circles [11]. Another input to the function is whether these circles are considered 'bright' or 'dark', and because the red in the image is converted to white, the 'bright' option is used here. The function outputs the radius of the detected circle and the position of its centre.

The outputs of *Circle code* are the position errors, *dx* and *dy*, and a Boolean signal called *CircleDetect*. The errors *dx* and *dy* are calculated by comparing the position of the centre of the circle with the centre of the image, where the drone is assumed to be positioned upon. *CircleDetect* is a signal that goes high when a circle has been detected. This is used, along with the Control System's FSM and a Simulink switch block, to switch between the circle errors *dx* and *dy* and the line errors *dx* and *dy*.

3.1.2.2 Line Following

The binarized and resized image is also fed into *Line code*, a third and final custom MATLAB function block, which outputs position errors, *dx* and *dy*. Again, the drone is assumed to be directly over the middle of the image. The first step in the line following process is to take slices from two rows of the image, the first through the middle of the image and the second a third of the way from the top. As the binarized image is a matrix, these are in vector form. A function is used to find the indices of *true* values in these vectors, as this corresponds to the red track in the image. Once these values are found, the midpoint of the red track can be calculated. The vertical and horizontal distances from the red in the centre of the image, to the red in the top section of the image are calculated. As the units of these are indices, the error value is in pixels. Before this can be output to the Control System's accumulator, they must be converted into metres using the meter per pixel conversion coefficient. These converted error values are the line errors *dx* and *dy*.

Figure 3.5 below shows the process outlined above on a screenshot of the drone's camera taken while running a simulation.

The final part of the *Line code* contains multiple if-statements for different edge cases. In the case that there is no red found in the upper slice of the image, there will be no horizontal or vertical movement. However, if then there is no red found in the centre slice of the image, but red is found in the upper slice, or vice versa, there can still be movement. In the case described above, red is found in both slices of the image.

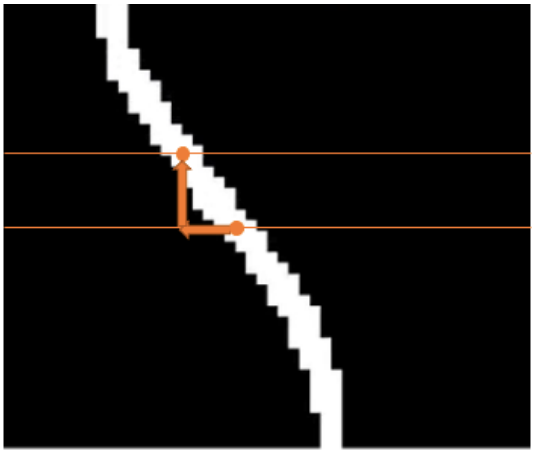


Figure 3.5: Line following code visualised on a segment of track

3.1.3 Results

Preliminary algorithms were tested on screenshots of the track's PDF file. Screenshots of multiple sections of the track were used in order to validate algorithms under as many scenarios as possible including both straight line and curved line sections. This acted as proof of concept for the line and circle detection prior to testing on drone hardware. The results of this are shown on Fig. 3.6.

The performance on the drone improved substantially after adding the image resize function. The effect of scaling the image resolution on the drone frame rate is seen in Fig. 3.7. As seen on the figure, the refresh rate when the image is not resized would not work with our line and circle detection algorithms. This is because the drone would continue to move and not update its set-point, causing it to drift off the track. The frame rate starts to become feasible at a scale factor of less than 0.4. While further downsizing improves performance, it comes at the cost of degrading the performance of the circle detection, as noise has a larger impact.

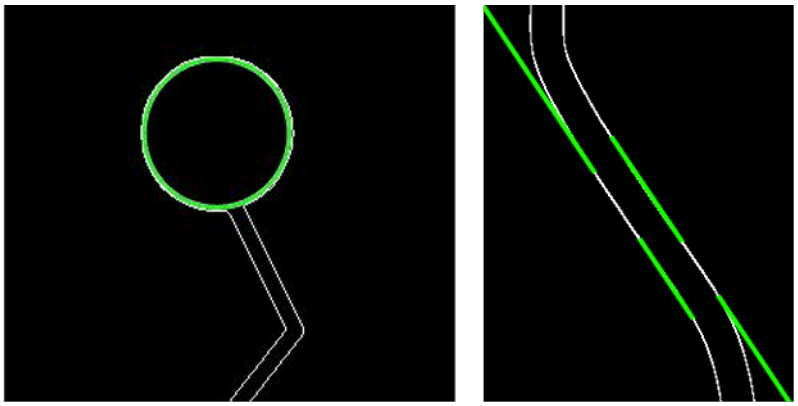


Figure 3.6: Detected circle and line segments

After the new line following code was written, it was compared to the original Hough line code, to determine that it would be faster, and therefore less computationally intensive. To test this, the functions were both run 7x on the same line segment, using the 'timeit' function. It can be seen, from Table 3.1 below, that the new code is 9.2x faster on average. While this is not an exact scenario that would occur real-time in the drone, it is a good representation for comparison.

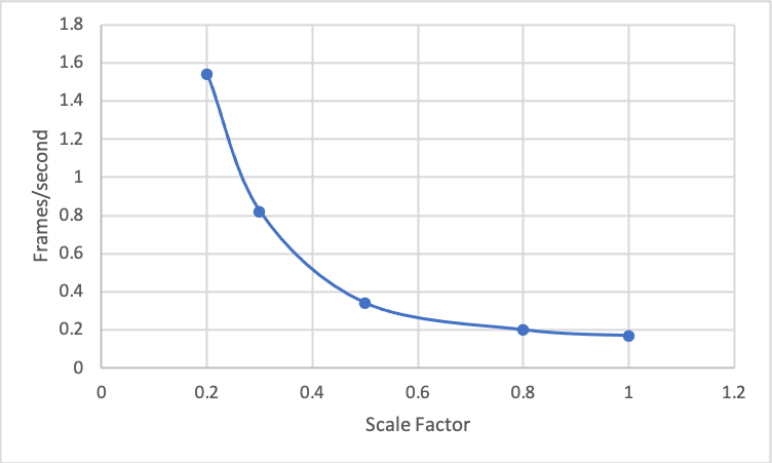


Figure 3.7: Frames per second against image scale factor

Figure 3.8 shows the outputs *CircleDetect* and the accumulated *dy* from the machine vision MATLAB functions as the drone progresses through the simulation. This demonstrates both the circle detection and line following code working. The dashed black line shows the *CircleDetect* flag being activated and the *pos-y* line shows how the drone's y-axis set point changes as it moves through the track.

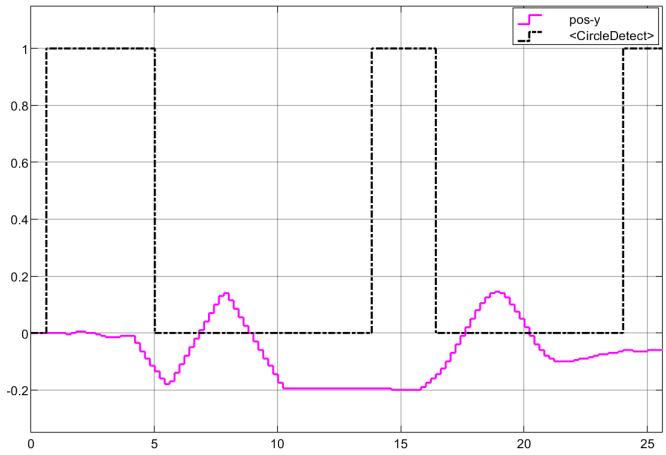


Figure 3.8: Output from Machine Vision MATLAB Functions

Table 3.1: Runtimes for Hough and Alternative line following algorithms

Algorithm	Runtimes (ms)							
	1	2	3	4	5	6	7	Average
Hough Transform Line Code	11.8	13.0	11.4	16.6	15.3	16.5	12.2	13.8
Alternative Line Code	1.50	1.90	0.90	1.50	1.40	1.80	1.50	1.50

3.1.4 Discussion

The implemented algorithms have key limitations. A lot of focus was placed on simplicity of the algorithms due to time and hardware limitations. This resulted in the algorithms not being able to adapt to all of the edge cases that occurred when testing. A key issue faced by algorithms was drift, as the line following code did not include code to keep the drone directly over the track. This did not seem to be an issue in the simulation but when it came to testing on the hardware the low refresh rate caused the drone to drift at corners before updating to the new line direction. This would sometimes lead to the drone not being able to see any track or when the drone approached the circle, the circle would sometimes not be in full view. While the circle detection worked under good lighting conditions, the circle detection would also fail at times due to poor lighting conditions and noise which caused the drone's video feed to be so distorted that the circle was no longer recognisable.

While the track following and circle detection worked, we were not able to implement a reliably accurate landing. This was due to a lot of issues, including the inaccuracy of altitude sensors as the height decreased. To solve this issue, a landing platform was designed in conjunction with the Mechanical Design sub-team. The platform featured a red circle that, in theory, would be used to calculate a more accurate landing position. However, it was found that at the lower resolution required, the circles on the platform could not be identified due to the distortion that occurred.

Design decisions had to be made to compromise between processing speed and accuracy - most noticeable in the resolution choice of the video feed. In order to reduce processing power to a level that the drone could handle the resolution had to be decreased. This had impacts on the performance of circle detection. With higher resolution, the Hough circle detection performs better. However, as this enhances the issues discussed earlier with drift, a compromise had to be made with the resolution to optimise overall performance.

A key limitation to our solution is that it will not work on angles greater than 90 degrees. This is because our algorithm ignores anything in the lower half of the image. There are no turns in the track provided that are larger than 90 degrees, therefore, a simpler solution, suited to the track, was pursued. A way to regulate this, in a production setting, would be to specify that path profiles with turns that are greater than 90 degrees are not suitable for use with this product.

3.2 Control Systems

The requirement for the Control System's section of the project relates to the simulation and hardware testing of the complete system code through the incorporation of Machine Vision image identification algorithms. The scope of this section includes a definition of the MathWork's Simulink working environments, a description of the finalised design, analysis of results achieved during artificial and hardware testing, finalised by a discussion on the limitations of the project and possible avenues for improvement.

3.2.1 Method

As mentioned previously, system designs are programmed using the Simulink Support Package for Parrot Minidrones, which is an add-on hardware package provided by MATLAB, based off of the Aerospace Blockset developed by MIT [12]. To complete the commissioned task within the time constraints, it was divided down into its key components;

1. Completion of stable hovering conditions at constant altitude.
2. Movement within 6 degrees of freedom.
3. Controlled landing at variant velocities.
4. Flight characteristics whilst carrying the payload.
5. Integration of MV algorithms.
6. Full scale testing with the system track and egg collection.

Referring to the mentioned Simulink package, the only block which is deployed on the hardware and requires alteration is *flightControlSystem*, with the other sub-system blocks providing the dynamics of the drone and its environment within simulation. Regarding the simulation and its parameters, they were first validated through topic research [7] followed by individual testing, with the linear environmental parameters specified and the dynamic air-frame model used for the drone itself. As a conclusion, the simulation in comparison to a practical model was defined to be accurate. Going back to the *flightControlSystem* block, for it to be loaded onto the hardware it first needs to be generated into C-code followed by binary code conversion. The process of compiling is done by a pre-supplied library included within the package.

After demonstration of flight hovering and motion in 6 degrees of freedom to the client it was decided to restrict the scope of development and focus on motion in the x, y and z planes only. This was elected in an effort to optimize both the code size and developing time. To complete the remaining sub-tasks and provide the MV team with a working test platform, the CS team was split into simulation design and hardware implementation respectively. The alterations were made to the Simulink code, configuration parameters, as well as to the 3D visualisation by merging a full-scale track provided by the client into the MATLAb 3D Virtual Reality World Editor. Considering the design requirements and physical constraints, a finite state machine was introduced, preceded by x and y coordinate drone translation, along with a switch for motor operation and another for altitude sensor selection.

Results provided after implementation of additional features, including both circle and line detection algorithms into the hardware, culminated in an overall system design that evidently exceeded the drone's processor capabilities and greatly impeded operational performance. Nevertheless, if those codes are to run separately the performance becomes closer to the simulated alternative. To overcome this issue saturator gain alterations and camera resolution require adjustments are required. Moreover, the *flightControlSystem* block was redesigned with the aim to leave only essential operations and minimise total CPU usage.

As an outcome, following a final presentation and the client's feedback, sluggish flight response was observed around the course track. For possible future development, analogous iterations for position reference control were considered and a PI-PD approach was reviewed yet not implemented.

3.2.2 Design Specifications

The predominant system code design is situated within the *FlightControlSystem* sub-block, illustrated on the following page (Fig. 3.10), which takes its coordinate inputs directly from Vision-based data. This section of the system also incorporates inputs from the drone sensory outputs including information relating to the Inertial Measurement Unit (IMU), ultrasound, pressure, and optical flow sensors which all gets processed within the *estimator* sub-block.

The Finite State Machine (*DecisionBasedFSM*) incorporates the outputs created by the path finding algorithms in the MV code and propagates the x and y coordinates dependent on what state it is currently located in. These coordinate signals are transmitted through to the *flightController* (light green sub-system) where they are compared to the estimated values for position reference, and the desired motion in the 3D plane is achieved. The state machine also dictates the output signals for the grabber arm activation, take-off flag, whether the path movement should be based upon the line or circle sections of the code, as well as, a signal enabling the rotation of the rotors for safety and power conservation purposes.

The state machine has been based around the completion of the task with an emphasis placed on design simplicity as to limit hardware performance complications, as well as, aid design apprehension and ease of communication with the client. As indicated in the decision-based flowchart in Fig. 3.9, after the drone has successfully taken off *[S_1]* it begins by following the line detection algorithm whilst constantly searching for a red circle *[S_2]*. Once a circle has been detected the system now switches to the centering code *[S_3]* as to hover directly over the circle ready for the land sequence *[S_4]* to be initiated. Following this and dependent upon which circle the drone is located at along the track, it will either perform an egg collection *[S_5]* and propagate through *[S_1]* again, or it is located at the final stage of the track and egg delivery *[S_6]* occurs completing the task and deactivating the drone.

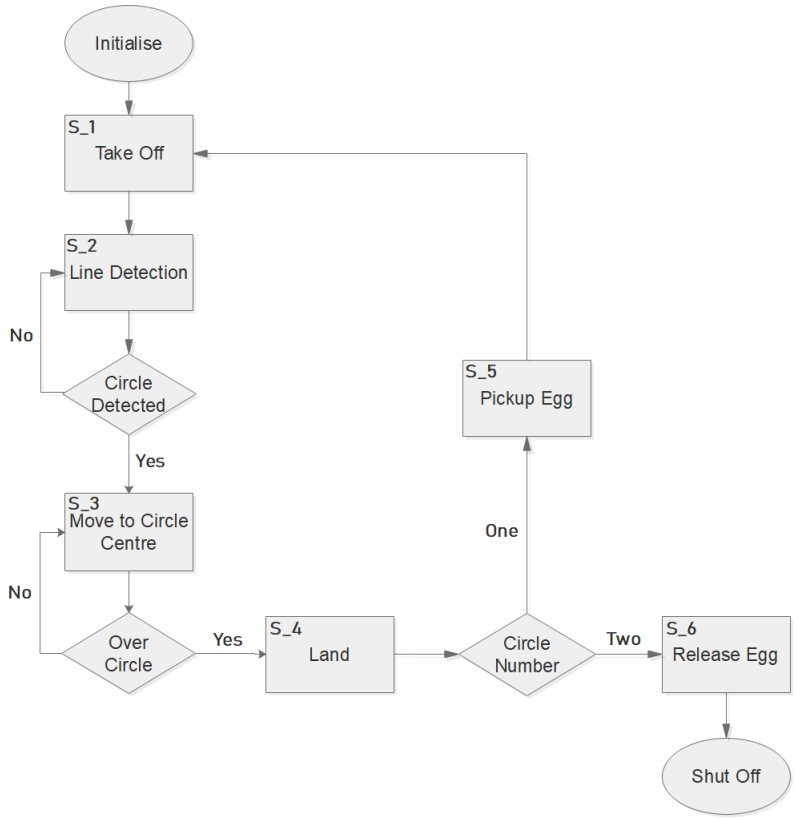


Figure 3.9: Simplified System Block Diagram

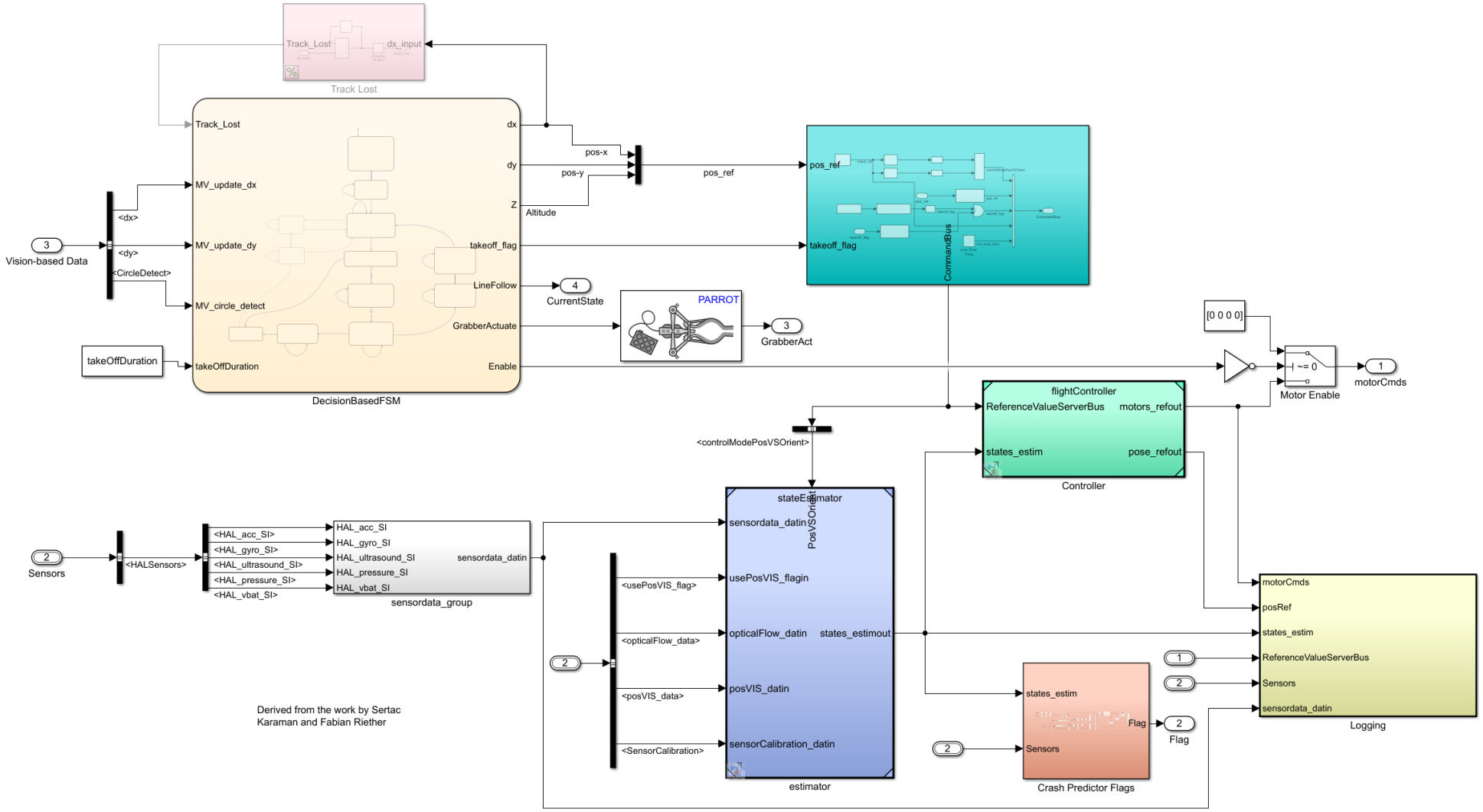


Figure 3.10: Flight Controller block

Translating this flow chart into the FSM, illustrated in Fig. 3.11, provides the decision making capability of the system which follows the chart arrows only when the conditional statement within the square brackets is true, and outputs the conditional action in curly brackets immediately. When in a specified block, the outputs contained within said block are concurrently being output as long as that block is still held true.

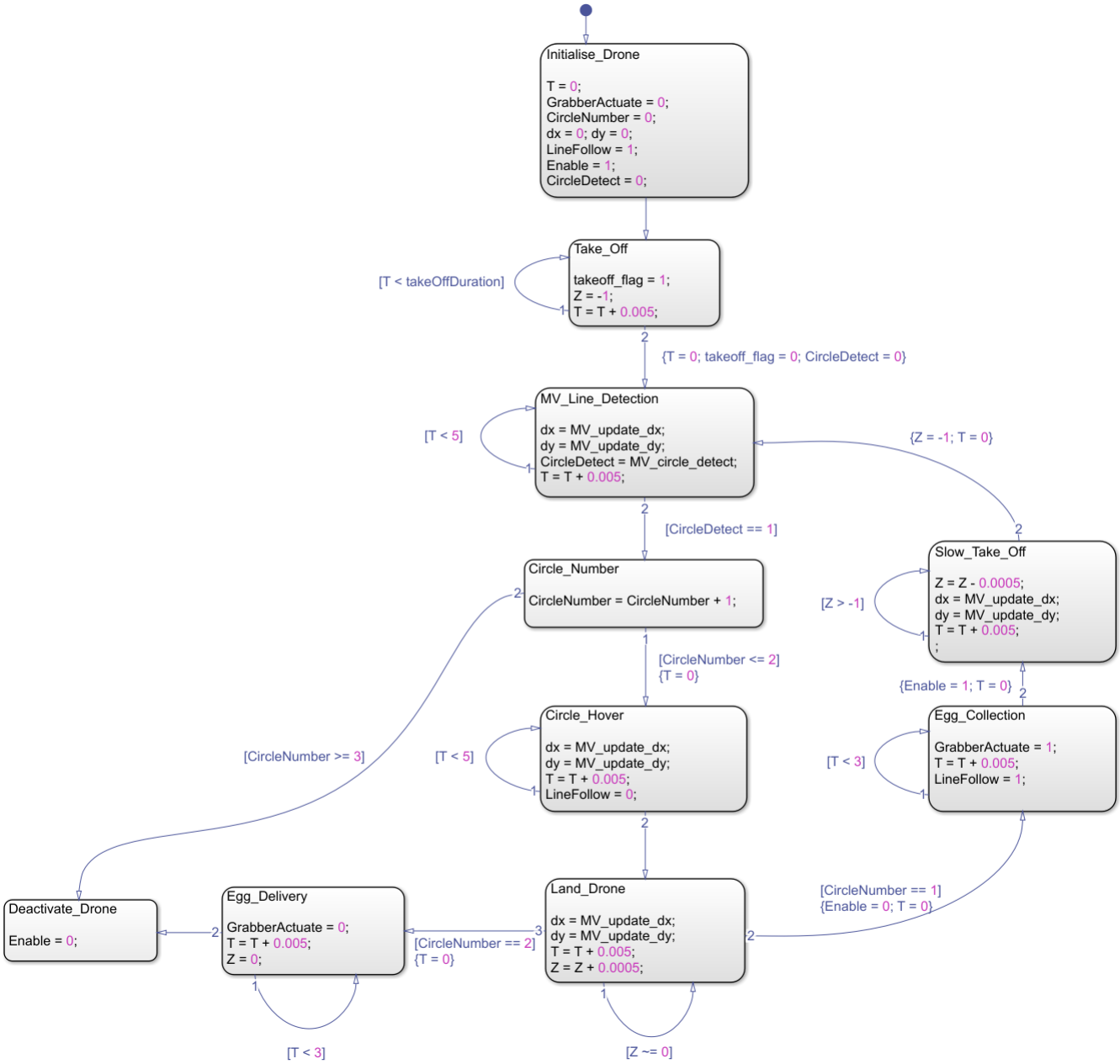


Figure 3.11: Finite state machine

There are some minor alterations between the design flow chart and the FSM incorporated into the system, most notably in the form of the *Circle_Number*, *Circle_Hover*, and *Slow_Take_Off* states which did not feature in the simplified chart. An additional safety feature in the form of the second transitional path from the circle number state leads directly to the deactivation of the drone as this indicates that the drone has detected an erroneous circle and task completion is no longer possible. An operational summary of all the states are provided;

- *Initialise_Drone*: The default state, once the system has been activated it initialises all the required system states and sets them to their desired values. Although not specifically a requirement, it ensures that all variables used within the FSM are functionally stable from the start of operation. This also sets the x and y coordinate system origin to that of the drone's initial starting position on the first circle.
- *Take_Off*: Incorporating the controller's built in take-off flags, it sets the desired flight altitude and waits within that block for the take-off duration time, allowing for the take-off procedure to be completed both quickly and successfully.

- *MV_Line_Detection*: The propagation signals from the line following algorithm are used to update the concurrent position reference of the drone allowing it to follow the red track lines effectively. Whilst in this state the detection code is constantly searching for circles within the track and will output high when a circle is asserted. This state is held for a period of 5 seconds as to allow the drone to progress out of sight of the take-off circles and eliminate an unwanted circle detection.
- *Circle_Number*: Upon detection of a circle, this state acts as a tally counter storing the current circle number within it. It only increments by one integer as it is a non-conditional state, with two possible transition paths, if the circle number is upwards or equal to three it sends the system straight to drone deactivation as an erroneous circle detection has occurred.
- *Circle_Hover*: Within this state the line follow signal has been set to low, switching the coordinate update system to that of the circle algorithm, which tends the drone towards its geometric centre. This state is held for 5 seconds providing the system with enough time to gain stable hovering conditions.
- *Land_Drone*: Whilst maintaining the coordinate updating to that of the circle centre, the drone gradually decreases its altitude at a rate of $0.1ms^{-1}$ and remains within this state until the altitude is 0m. At this rate the landing sequence should take 10 seconds.
- *Egg_Collection*: The drone's rotors are turned off as the drone now rests on the egg platform, with the grabber signal being sent high, this state is held for 3 seconds providing sufficient time for complete grabber actuation. The system coordinates are now being updated as per the line detection algorithms.
- *Slow_Take_off*: The rotors are once again activated and the drone is instructed to increase its altitude at a rate of $0.1ms^{-1}$ until it has reached its flight height of 1 metre. This ensures that the inertia on the payload is minimised and the risks of the egg slipping out of the grabber are also decreased.
- *Egg_Delivery*: If located at the second circle, the grabber signal is sent low and deactuation occurs thus releasing the egg, this state is held for 3 seconds as to allow complete egg release.
- *Deactivate_Drone*: The rotor signals are once again set to zero, concluding the flight of the system at the third circle post egg delivery.

As mentioned above, the FSM has a total of three input signals originating from the MV section of code, referred to as *MV_update_dx*, *MV_update_dy* and *MV_circle_detect* respectively. Noting the particular section of code below (Fig. 3.12), the specific x and y coordinates are determined through the use of a switch, enabled by the *CurrentState* signal which takes its inputs directly from the *LineFollow* output of the FSM. Thus when instructed to be following the line section of code, the leftmost switch outputs a Boolean value of false to the right dx/y propagation switches, which in turn output only the line algorithm updates. Prior to being sent to the FSM, the individual values for dx and dy are sent through a set of saturators limiting their maximum increase potential to 0.04m per cycle, as well as accumulators set at half gain which continuously provide the position reference of the system with regards to the original starting point.

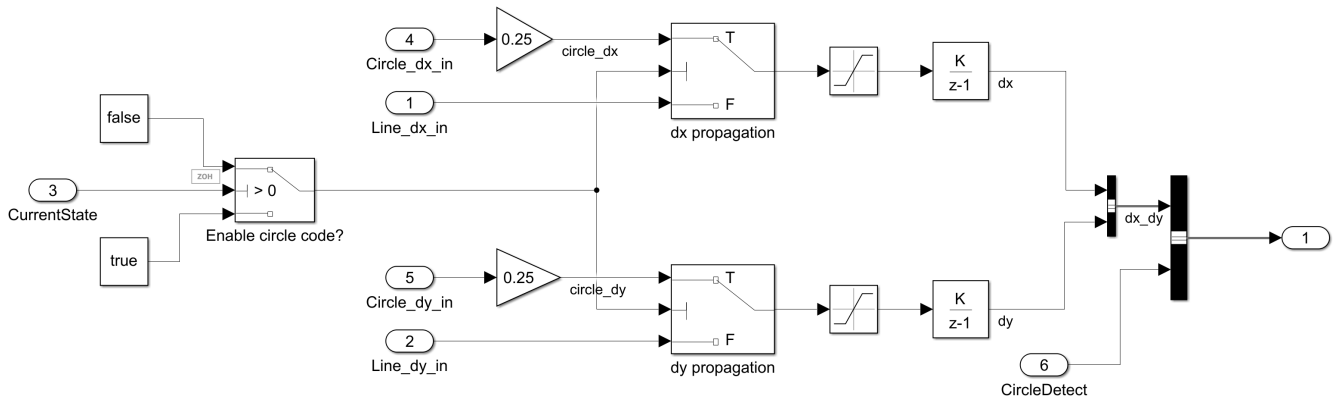


Figure 3.12: Translation of MV Algorithms into Propagation Coordinates

Alterations to the provided Simulink code were made in order to eliminate a pre-existing set altitude landing point that determined at an altitude of 0.44m a landing sequence was to be initiated. With this flag removed, it was noticed in simulation that when flying at altitudes roughly at and below this 0.44m mark, flight characteristics became unstable and major drifting in the x-y coordinates occurred. Further investigation revealed this to be due to limitations of the ultrasound height estimations illustrated by the orange trace in Fig. 3.13, whereby at values of below 0.44m the sonar output is held constant, with the combined height estimation being a differential value based on the responses of both the sonar and pressure sensors.

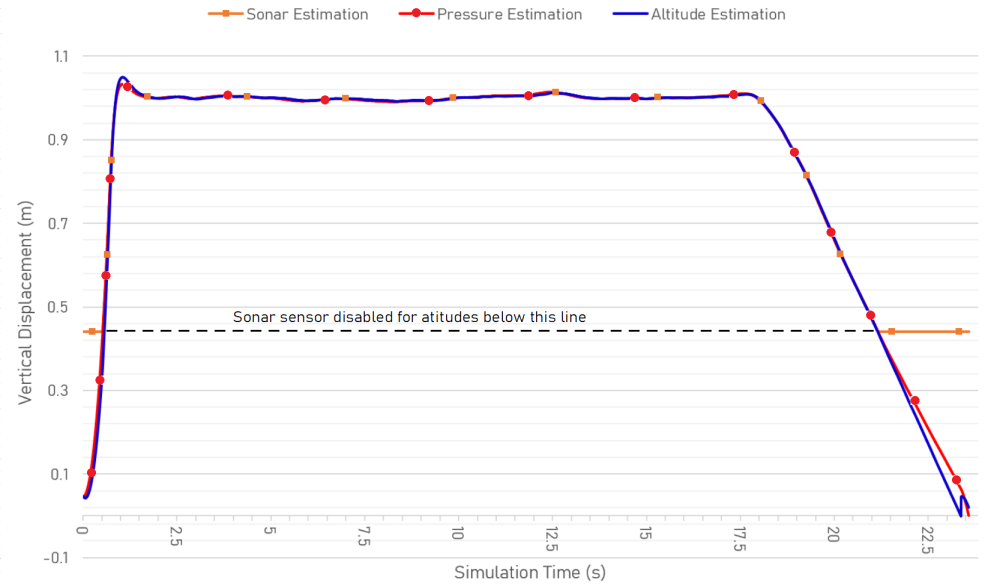


Figure 3.13: Simulated Response of Altitude Estimation

This was remedied through the addition of the blue section of code within the *Altitude Estimation Code* sub-block (Fig. 3.14), which states that when flying at altitudes below 0.44m, the height estimation is no longer considering inputs from the sonar sensor and bases it solely upon the pressure sensor. The current design also incorporates a linear altitude algorithm after this point where it no longer tracks the pressure estimation line and instead follows a linear decremental response. As the system is not set to fly at these altitudes for long periods of time, this solution is valued as acceptable.

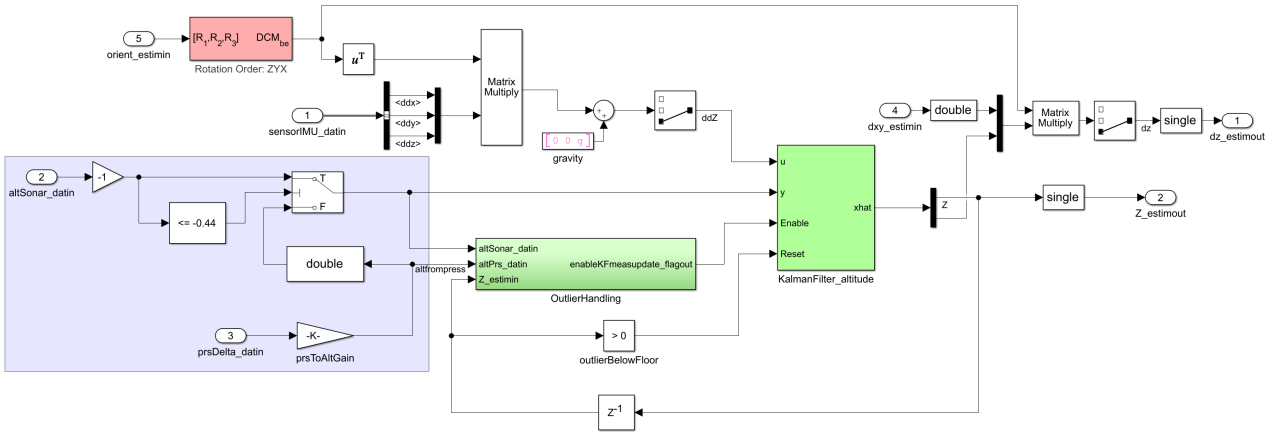


Figure 3.14: Altitude Estimation Code

Flight stability is provided through Proportional-Integral-Derivative (PID) controllers within the *flightController* sub-block, whereby closed loop feedback is incorporated to calculate the error between the desired and the actual values. These control systems prove to be very effective in providing stable flight as the system continuously alters the individual commands providing 4 degrees of freedom (DoF) including roll, pitch, yaw and thrust. As the quadcopter is underactuated in nature, it requires the combination of both its roll and pitch to move in the horizontal x-y plane, thus necessitating the inclusion of an outer loop position controller, effectively providing flight characteristics with a total of 6 DoF. An overview of the implemented control design is included;

- *Position Reference X-Y*: Designated by the MV algorithms, these inputs are fed through two closed loop systems and make up part of the Roll and Pitch signals.
- *Orientation Reference*: Provided by the Inertial Measurement Unit (IMU), this creates the estimated values for the orientation coordinates.
- *Yaw = 0*: Designated by the user to be set to a value of zero degrees, this keeps the drone facing forward along the track at all times.
- *Altitude*: Used to control stable take-off and landing conditions, this signal directly affects the thrust PID controller within the inner loop.

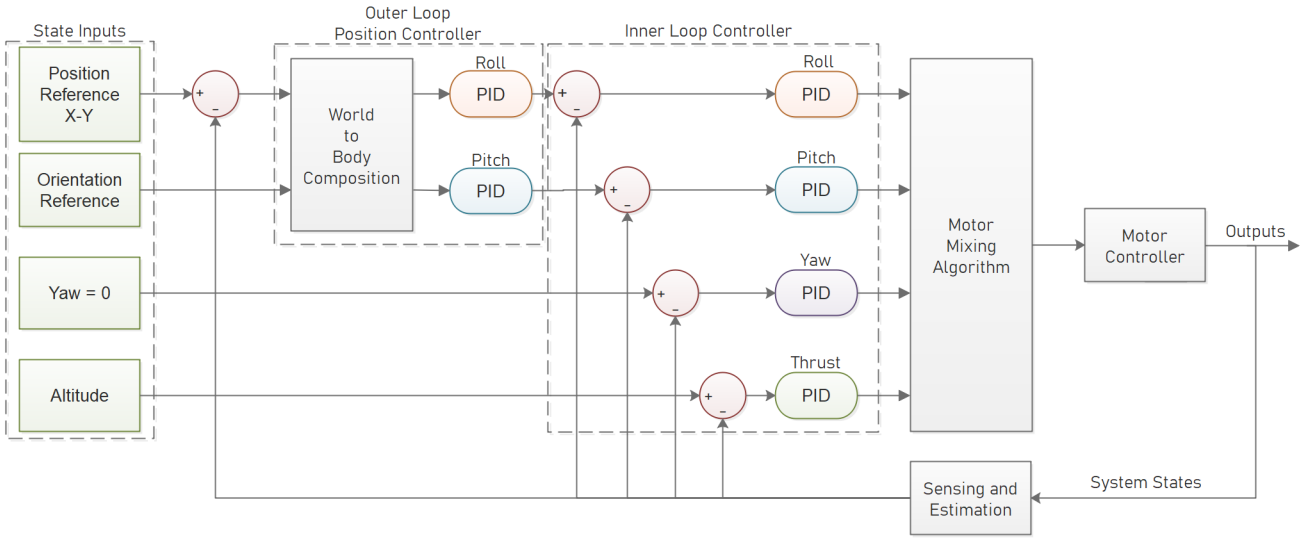


Figure 3.15: PID Overview

After the 4 DoF signals are formed, they are fed into the *Motor Mixing Algorithm* (MMA) which combines them into a single 4x1 matrix of desired motor power values. Subsequently, these are passed through the *Motor Controller* which limits the maximum power output of the individual motors through saturator blocks, whilst setting the individual rotational directions to [1, -1, 1, -1] as to eliminate torque spinning.

A final design feature to mention, are the pre-included safety features and data checks which ensure that all sensor data is correct and operational. These include, but are not limited to:

1. Geofencing errors that check if the drone's x-y coordinate positioning system is functioning under normal operating conditions, essentially ensuring that neither coordinate estimate changes by a value greater than 10 within a single time period. This would be indicative of a sudden impact and would be characterised by a large change in vehicle acceleration, handled by the IMU sensor.
2. Optical flow errors are handled in a similar way, whereby if there is a limited number of successful image reads, or the positional data conflicts with that of the other sensory unit estimations, then the safety flag is initiated.
3. Within the *DataHandling* block, the optical flow sensor is disabled for very low altitudes, large angular rates and acceleration values as these are proven to be troublesome for the processor to effectively compute.

3.2.3 Results

Initial drone take-off and hover simulations showed major overshoot in the vertical axis, with a total height reach of 3.34m being achieved as opposed to the desired value of 1m, providing unacceptable responses. To eliminate this the *takeOffDuration* was reduced to a value 0.68 seconds and the *takeOffGain* was also reduced to a value of 0.30, with the original and corrected flight responses illustrated below in Fig. 3.16. The altitude PID had its derivative component increased to a value of 0.60 as well as this further decreased the total overshoot. The improved thrust PID not only decreased the maximum overshoot 99.7%, but it also decreased the total settle time by a margin of 1.23s providing exceptionally better system take-off responses.

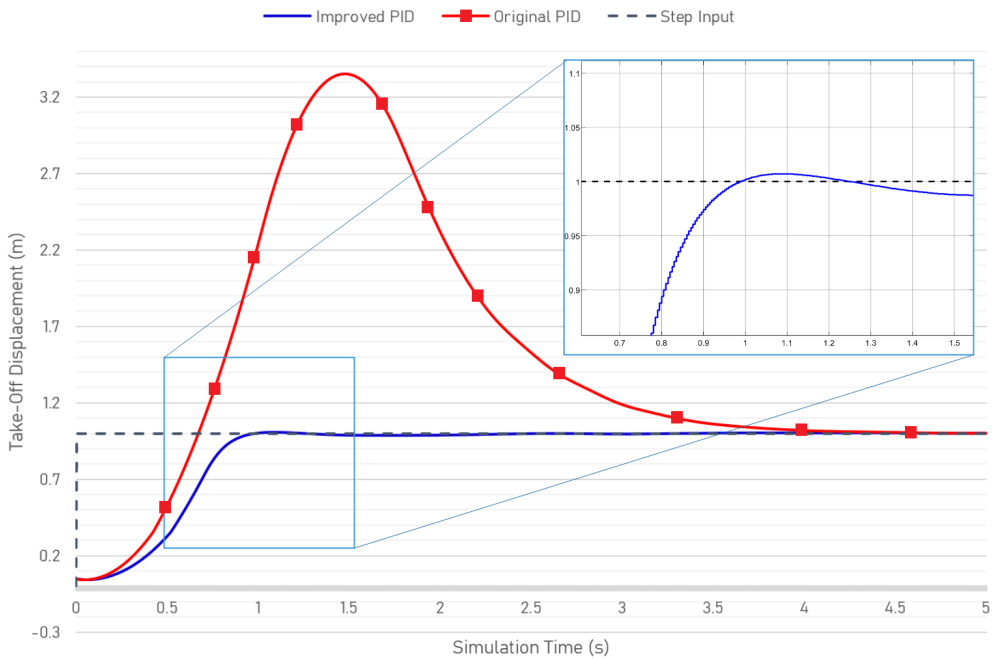


Figure 3.16: Take-Off PID Comparison

Table 3.2: Simulation Results

PID	Percentage Overshoot	Time to Settle
Original	234% (+2.340m)	3.80s
Improved	0.7% (+0.007m)	2.57s

The same tuning procedure was applied to the other PID controllers with their respective values indicated in Table 3.2. A notable change was observed in the X-Y position reference PID, whereby when subjected to a 1m step input in the x-direction (ie forwards along the track), the drone would overshoot by 0.4m and then proceed to teeter back and forth as it tries to locate itself above its desired point. This was altered so that the overshoot was minimised as much as possible whilst eliminating the oscillatory behaviour for smooth and stable flight. This is illustrated in Fig. 3.16 where although it takes the drone twice as long to reach its designated location, due to the much lower value of proportional gain in its controller, it actually reaches steady hovering conditions in a lower set time overall.

The complete system gains for all PID controllers are listed in Table 3.3. These values were calculated using the Ziegler-Nichols method [6] for PID tuning, followed by incremental tweaking as to achieve optimal flight responses for the designated task, considering that sometimes the fastest response is not always the most suitable.

Table 3.3: PID Values

Parameter	Proportional	Integral	Derivative
X-Y Position	0.10	-	0.13
Altitude	0.80	-	0.60
Yaw	0.003	-	0.0012
Pitch	0.013	0.010	0.0020
Roll	0.010	0.010	0.0028

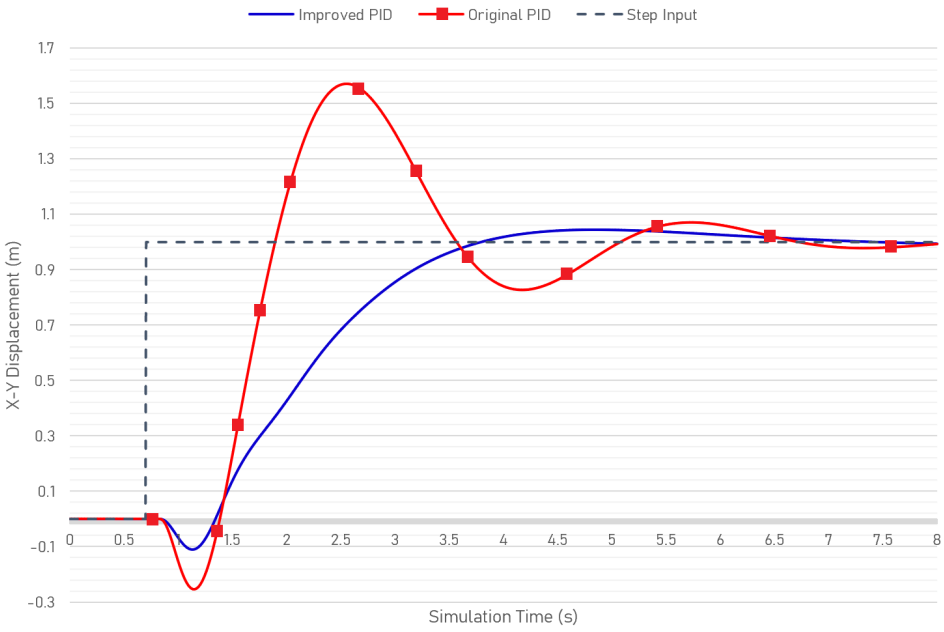


Figure 3.17: X-Y PID Comparison

In order to test the effectiveness of the path finding abilities of the MV algorithms, the simulation environment pre-loaded with the Simulink project files was altered as to now include a scaled version of the flight track used in the final task. All testing done henceforth included take-off from within the first circle and propagation along the track.

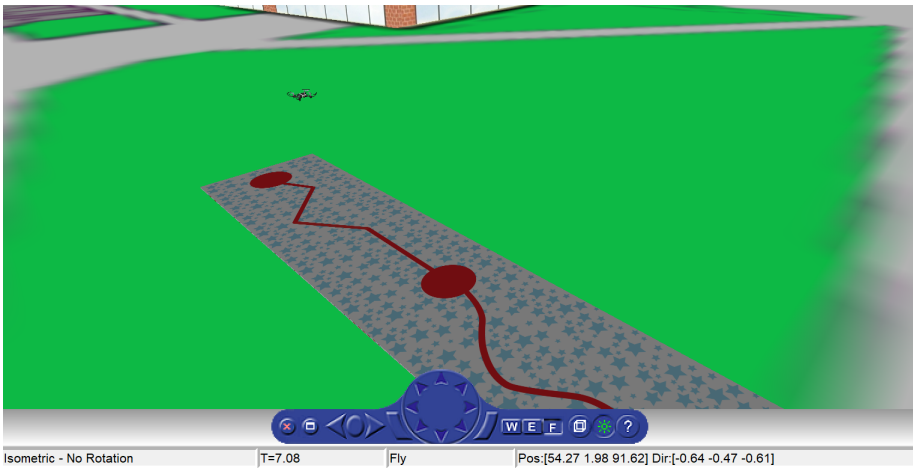


Figure 3.18: Simulation Environment

Circle recognition occurs effectively at the desired flight altitude of 1m, with monitoring of the system signals clearly indicating that the MV circle detection algorithm works successfully. The Simulink package allows the user to view concurrent updates of the 3D simulation, the image recognition video feed sent to the MV algorithms, selected signal scope plotting, and, live indication of what state the FSM is currently running.

All individual aspects of the state machine were tested in separate sections, these were separated by the landing at the second circle for reasons which will be mentioned in the discussion section of the report. The FSM response through the simulated testing was proven to be successful with no alterations being required, the system was also tested with an additional circle to inspect the effectiveness of the safety turn-off feature which also operated successfully. One issue with this method of shutting off the drone is that although it produces an instantaneous response, a drop from this flight height could incur the probability of damage to the egg payload.

The path finding abilities of the system are proven to be excellent. When solely testing the MV line propagation algorithm without circle detection in place, the drone is able to accurately locate the red path and promptly update the position reference sent through the controller. Multiple tests were performed to objectively analyse the effectiveness of the system by altering the speed at which the propagation saturators were set to. This test was replicated for both directions of the track layout.

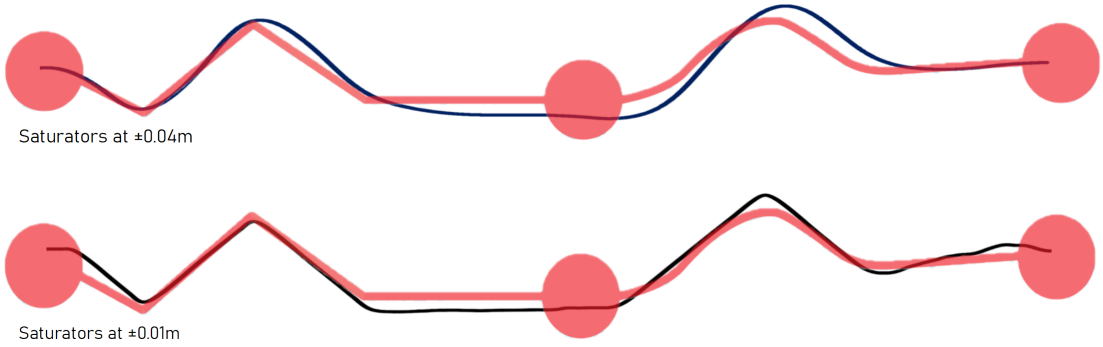


Figure 3.19: Path Following Comparison

Table 3.4: Path Follow Results

Saturator Limit	Track Completion Time	Percentage Match
0.04m	25.80s	42.59%
0.01m	95.28s	61.11%

Noting the results in Table 3.4, although the path tracking abilities presented with the 0.01m saturation limit provide a quantitatively more accurate and precise flight pattern, the much greater completion time is not an acceptable trade-off. Thus, the final system incorporates the 0.04m saturation limits for the x-y propagation code, albeit at a slight loss of tracking accuracy. The percentage match statistics, although they may appear strikingly low, are in fact positive as this is based off of the amount of time the drone is above the red part of the track at an altitude of 1m, which in itself is only 35mm wide.

When transitioning the testing onto the physical hardware, it became evidently apparent that the drone's processing capabilities struggled heavily with the computational load placed on it by the detection algorithms and the scale of the FSM design. An attempt to run the complete design showed very sluggish refresh rates within the image processing video viewer, this was slightly improved by reducing the camera resolution to 0.3x that of the original down to 48x36 pixels. Any lower than this resolution and the circle detection code became unreliable. Alterations to the dx and dy propagation commands from the line detection algorithm were increased through the incorporation of proportional gains to counteract the slow image processing rate, as indicated in Fig. 3.20. Although much slower than the simulations, the drone was capable of following the line detection algorithm and did manage to reach the end of the track on numerous occasions. One downside, however, is that intermittently it would propagate outside of the viewing range of the track due to this increased dy gain and no longer receive any directional movement inputs.

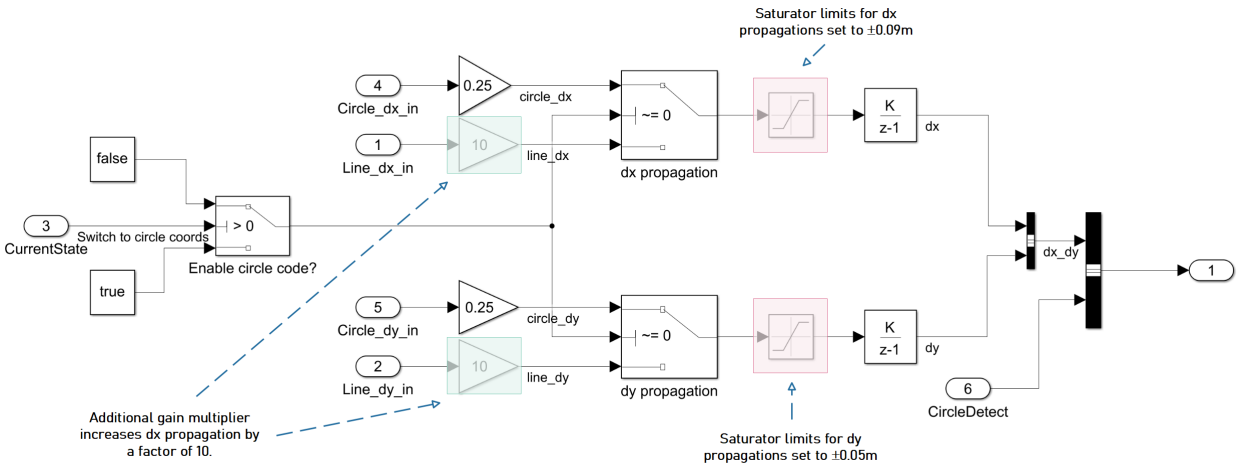


Figure 3.20: Alterations to Propagation Gains

There was an evident height mismatch between actual flight and the requested hovering altitude, with the drone electing to fly at values roughly 80% of those intended. The reasons for this are still not fully understood as an inspection of the flight data logs shows that the altitude sensors correctly record hover heights in the region of

0.788m when a value of 1m was requested.

A final issue that was recorded during flight testing, was that at low altitudes roughly below 0.25m, the drone can exhibit drifting in the horizontal x-y planes. Referencing the figure below, when the drone was instructed to hold a static height of 0.2m above its take-off point, the slow drifting pattern began immediately with both the x-y coordinates changing in a similar manner. Note that this phenomenon is not as severe when landing at the regular descent velocity.

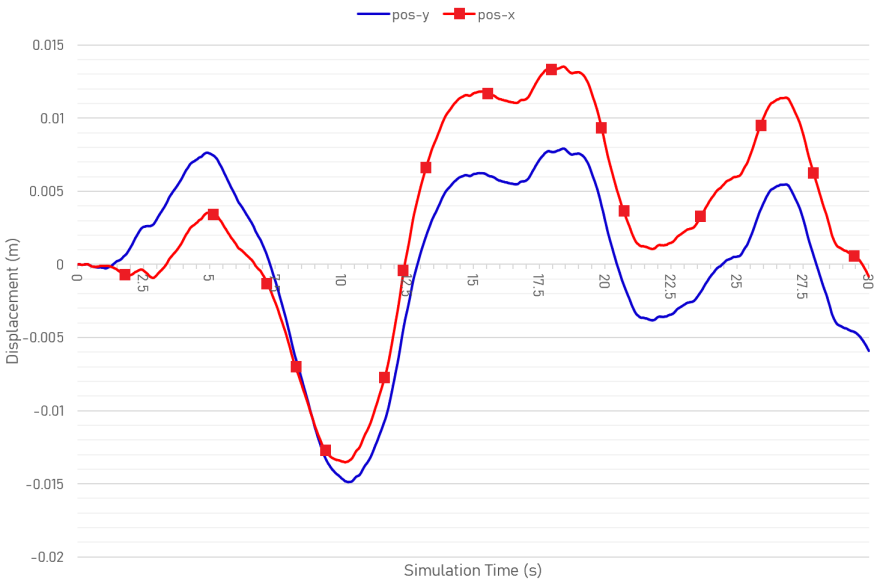


Figure 3.21: Low Altitude Drift Pattern

Based on our experiments and the outcome of the research made by Charles University in Prague [13] the issues could be;

1. The blowback effect from the rotors at these low altitudes creates a cushion of high pressure below the drone, effectively applying a force vector opposing the intended motion.
2. At these altitudes the camera cannot efficiently comprehend the image captured and causes optical flow issues, the drone no longer has realistic data for its current position and thus provides erroneous x-y propagation inputs.
3. Although the ultrasonic sensor has been deactivated at these heights, data could be being sent through to other sections of the state estimation code and is interfering with flight characteristics.
4. When above the circle at low altitudes, the image processing no longer sees the geometric circular shape and the dx and dy propagation signals sent to the position reference of the system become incorrect.

This issue could be counteracted by shutting off signals to the rotors when the drone reaches this low altitude and having it land on the platform unassisted. If it was to free-fall from an altitude of 0.25m at a descending velocity of $0.1ms^{-1}$, it would impact the ground at an estimated velocity of $2.217ms^{-1}$ which is low enough as to not cause damage to the drone itself or its payload. This solution was not tested due to time constraints and as such landing at standard velocities was incorporated.

3.2.4 Discussion

Testing of the system with the circle detection through the incorporation of the FSM included previously, showed that the drone can effectively take-off at the first circle, propagate along the line, as well as, detect and land successfully at the second circle. This is the maximum system testing which can be achieved through the current simulation suite as floor contact asserts a simulation failure signal, forcing it to cease immediately. Although this feature can be disabled, the way within which the proprietary drone visualisation block has been included into the 3D world editor, means that geometric collisions between the drone and any object within the simulation

are not permitted. Along with this, there is also no inclusion of the grabber within the model, thus there is no possibility of testing the design with regards to the collection of the egg.

The drone fundamentally lacks the computational power required to execute the entire design code, the hardware pre-supplied by the manufacturer is not upgradeable and the only methods of improving performance can be achieved through simplification of the code and its processing requirements. Provided that the simulation executes the code effectively, the possibility of having a host computer perform the computational processing and communicating with the drone may alleviate this issue. In this scenario, the drone would send the images from its camera feed along with its sensory data to the host, this would then calculate the required motor commands which subsequently get sent back to the drone for implementation. A simplified illustration of this is provided below in Fig. 3.22.

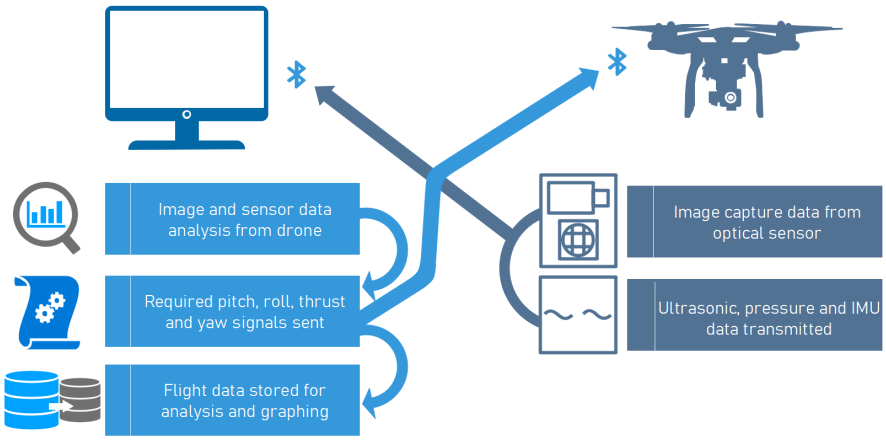


Figure 3.22: Alternative Control Proposition

One issue which was mentioned previously was that if the drone were to stray too far from the track and lose sight of it, the way it is currently designed it is instructed to hover in its position indefinitely. One possible solution to this would be the inclusion of a “search and find” state, this would activate if the line detection provided a zero propagation output for certain periods of time and the drone would be instructed to hover in a small 0.5m pattern to the left and then back across the right until it once again finds the track. If it does not locate evidence of the track within a certain time frame it would then be instructed to enter the shut off state.

In order to initiate the search and find states, a new signal was created labelled *Track_Lost* (indicated in red in Fig. D.1) which checks for concurrent updates in the dx propagation signal using a derivative and compares that value to zero. Thus, when there are no longer any dx updates, the signal is enabled indicating the loss of the track. Note in the updated FSM, step inputs are used to execute the search states as this section of code is located after the position reference accumulators, however, as the propagation PIDs have been tuned to limit overshoot this will not result in unstable movement. This functionality will not be added into the final design as it will only serve to increase the computational requirements placed on the hardware, it is merely included to serve as an example of a possible solution.

For safety reasons the pre-supplied code includes an IMU sensor which sends a crash flag upon sudden change in its inertia, as would be expected when subjected to a sudden impact. This flag could, in theory, be activated upon a successful landing by accident and it may prove necessary to neglect the usage of this flag when within the landing state.

There were multiple issues which arose from the physical testing space, whereby, optimal flight conditions require large unobstructed areas with bright lighting. Due to restrictions in the availability of such locations, consistent testing was not feasible, with a noticeable reduction in the drone’s path finding abilities occurring as the amount of external daylight decreased.

The built-in sensors are lacking in terms of their precision and flexibility, and the system as a whole would benefit greatly from the inclusion of GPS tracking and LiDAR functionality. This would improve the flight position stability and make egg detection possible, although it would inadvertently add additional computational load onto the drone’s processor. Although in its current state, the drone is technically capable of meeting the task requirements, its ultimate chances of success could be improved with these additional sensors.

3.3 Mechanical Design

The role of the Mechanical Design team was to suggest the mechanical method of manipulating the egg to succeed in the task. This involved a review of the stock grabber that was supplied with the drone for the task of this project, as well as a study of the alternatives to this. Several tests were performed on the grabber and the quail egg to determine the force that the grabber can deliver and the force that the egg can withstand. Additionally, the team was responsible for developing a landing platform that helps the drone to target and land precisely enough to initiate grabbing of the egg, as well as securing the egg in place. Finally, the implementation of a precision landing mechanism is discussed to improve upon the landing platform subsystem.

3.3.1 Method

The work of the Mechanical Team is split into several stages that are described as follows:

- Quail egg review
- Parrot Mambo Grabber review
- Landing platform development
- Precision landing mechanism
- Implementation and overall integration of the system

Firstly, the supplied components are reviewed to understand the mechanics and interaction between individual components of the system. This involves an investigation into how the grabber performs as a stock unit, studying the physical properties of quail eggs, and studying the interaction between the grabber, quail egg, and overall drone operation. Next, the design, manufacture, and testing of a landing platform system is discussed. Augmenting the landing platform system, a precision landing mechanism is discussed. Finally, some sustainability considerations as well as observations on the work done by the Mechanical Design team and its pertinence to a mass-produced and commercial application is discussed.

3.3.2 Quail egg review

With reference to the task, a quail egg is to be collected, flown along a track, and deposited at the end of the track without breakage and autonomously by a supplied Parrot Mambo drone. To this end, the strength of quail eggs were experimentally determined to determine the stresses it could handle when grabbed. It was decided that the egg must be grabbed parallel to its minor axis to achieve consistent grabbing, shown in Fig. 3.23.

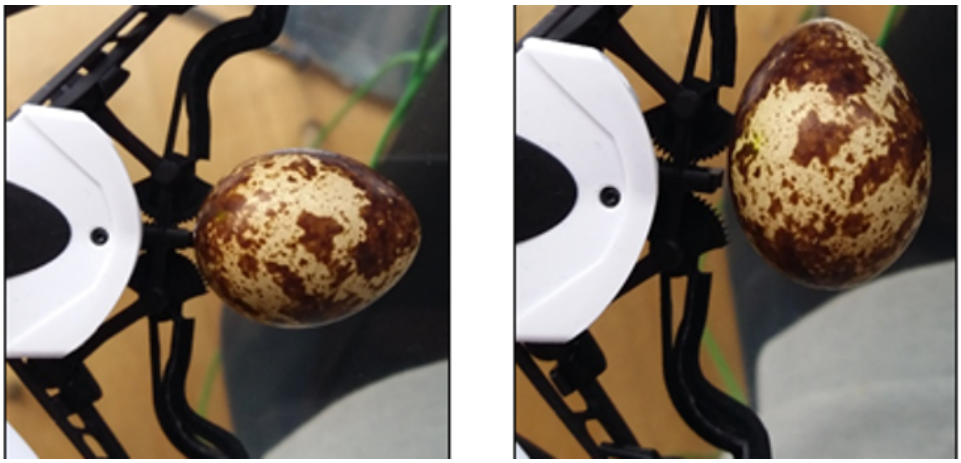


Figure 3.23: Egg Orientation: Minor axis (left), Major axis (right)

A simple test rig is used to measure the maximal force of the egg before failure. The egg is secured in place as a container is placed on top of the rig. Water is then added to the container to increase the force on the egg due to gravity gradually (See Fig. 3.24). The strength of the egg is exceeded when a crack was observed to form, thus adding of water is stopped and the container is weighed to get the total mass of the failure load.

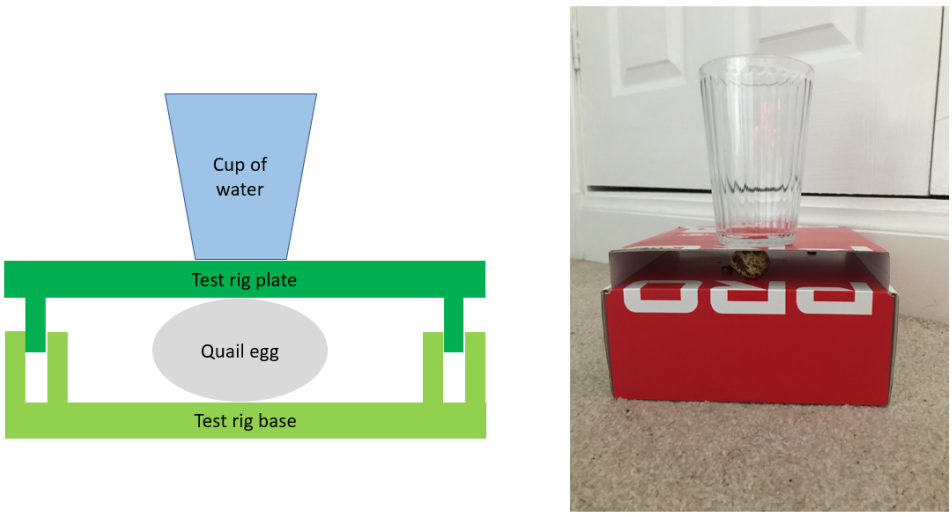


Figure 3.24: Egg strength Test Rig: Diagram (left), Real Testing (right)

The strength of the egg shell is determined using Newton's Second Law shown below in Eq. 3.1.

$$F = ma \tag{3.1}$$

Where m is the mass of the load, a is the gravitational acceleration and F is the force that the egg can withstand before failure.

The test results and properties of the egg are presented in the Table 3.5.

Table 3.5: Physical Egg Properties

Average physical properties	Value	Absolute error
Mass of the load [g]	631	0.5
Force to break the egg [N]	6.19	0.005
Mass of the egg [g]	2	0.5
Diameter of the egg [mm]	25	1
Height of the egg [mm]	40	1

The absolute errors are due to the measuring instruments. The egg is measured using a ruler that has increments of 1mm, giving an absolute error of 0.5mm. Since both ends of the measurement yields the same absolute error, this gives the total absolute error of 1mm. A similar error analysis is done for the scale which has increments of 1g resulting in an absolute error of 0.5g. According to [1] the drone's grabber attachment is capable of carrying a load of maximum weight of 4g. The egg is therefore a reasonable load to be carried for this task.

3.3.3 Grabber evaluation

3.3.3.1 Functional requirements of the grabber

The functional requirements of the grabber required to maximise the chance of carrying out the task successfully are outlined below.

1. Function
   - 1.1. Grabs the quail egg with precision and in one attempt
   - 1.2. Maintains sufficient grip of the quail egg over the course of the flight
   - 1.3. Does not break or induce breakage of the quail egg
2. Flight characteristics
   - 2.1. Does not or minimally impacts drone flight performance

3. Integration
    3.1. Controlled and can be interfaced by the overall drone system
    3.2. Control system uses minimal amount of computing power
4. Safety
    4.1. Geometry of object causes minimal harm to living objects in case of collision
5. End consumer use
    5.1. Must be fastened securely and with minimal or no need for tools

To this end, the stock grabber module is reviewed and compared against the list of functional requirements needed of it. There are three decisions that can be made with respect to the grabber:

i Develop grabber attachments to enhance the capabilities of the stock grabber

ii Develop an entirely new grabber attachment

iii Use the stock grabber as-is

Developing an entirely new grabber attachment was not a feasible choice due to the limited control capabilities of the grabber system within the Simulink environment, which will be discussed in the following section. If a new grabber were to be developed, it would need to be controlled externally, adding unnecessary complexity to the task. Developing grabber attachments would increase the chances of the grabber completing the task successfully as it can enhance grabbing precision as well as maintain grip on the egg with either a larger grabbing surface or simply carrying the egg in a basket-style attachment, however the augmentation must be justified against the added mass and complexity of balancing the centre of gravity and lift of the drone. The final suggestion and decision from the Mechanical Design team was to use the stock grabber as-is. This is justified in the following sections.

3.3.3.2 Grabber control analysis

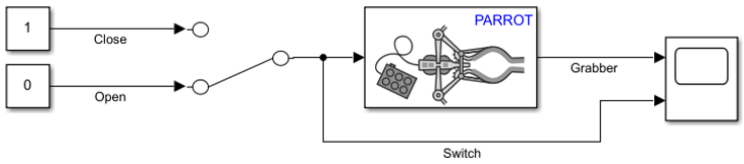


Figure 3.25: Provided Simulink grabber control block

This section discusses an analysis of the grabber control capabilities. Shown in Fig. 3.25 is the provided grabber control block within the Simulink environment. The grabber block is commanded by inputting a boolean input, with a 1 to close and 0 to open. The grabber block also outputs its current state.

The analysis has two objectives, that is to determine if the grabber is capable of detecting that it has grabbed an object, and to determine if it were possible to control the grabber directly by pulsing the actuating motor via sending command signals directly to the motor. To test the former, the experimental setup shown in Fig. 3.25 which measures the current state of the grabber and the inputted command signal against time is tested with and without an egg. If a time discrepancy exists in the two situations, it can be concluded that the grabber can detect that it has grabbed an object. The result of this test is shown in Table 3.6. A $0.01s$ average time difference which translates to a roughly 1.17% percentage difference is insignificant and concludes that no sensing capabilities are present in the grabber. From researching the documentation and Simulink blocks, no method of directly controlling the grabber was found. This therefore rules out the feasibility of designing an entirely new grabber as it would need an external control method which would also need to be integrated with the native code of the overall drone.

Table 3.6: Grabber control analysis

Attempt	Grabber cycle time (s)	
	Without egg	With egg
1	0.89	0.83
2	0.84	0.86
3	0.84	0.84
4	0.84	0.89
5	0.83	0.88
6	0.85	0.83
7	0.85	0.87
8	0.89	0.83
9	0.84	0.83
10	0.88	0.85
11	0.87	0.89
12	0.84	0.84
Average (s)	0.86	0.85

3.3.3.3 Grabber grip force analysis

The maximum gripping force must be measured to prove whether the stock grabber is capable of holding the egg without crushing it, and maintaining the grip on the egg throughout the flight. First, the maximum grip force is tested experimentally using a ballpoint pen spring as there is no data publicly available. Since the spring constant k is not known, an experimental procedure which consists of a hanging object with a known mass attached to the spring is done. Hooke's Law is used to compute the spring constant (see Eq. 3.4). The ballpoint pen spring is then fixed at one end while the other end is connected to one grabber arm in the closed position. The grabber is actuated resulting in pulling the spring which changes its length (see Fig. 3.26). The length difference measured along with the spring constant calculated earlier is used to determine the gripping force of the grabber.

$$F = k\triangle x \tag{3.2}$$

Where k is the spring Constant, $\triangle x$ is the change in length of the spring and F is the force of the spring.

The appropriate value are inserted to Eq. 3.4 for calculating the spring constant k and the gripping force of the grabber.

$$k = \frac{F}{\triangle x} = \frac{Mg}{L_1 - L_0} = \frac{124 * 10^{-3} * 9.81}{(33 - 26) * 10^{-3}} = 173.3\frac{N}{m} \tag{3.3}$$

$$F_{grabber} = k(L_2 - L_0) = 173.3 * (30 - 26) * 10^{-3} = 0.70N \tag{3.4}$$

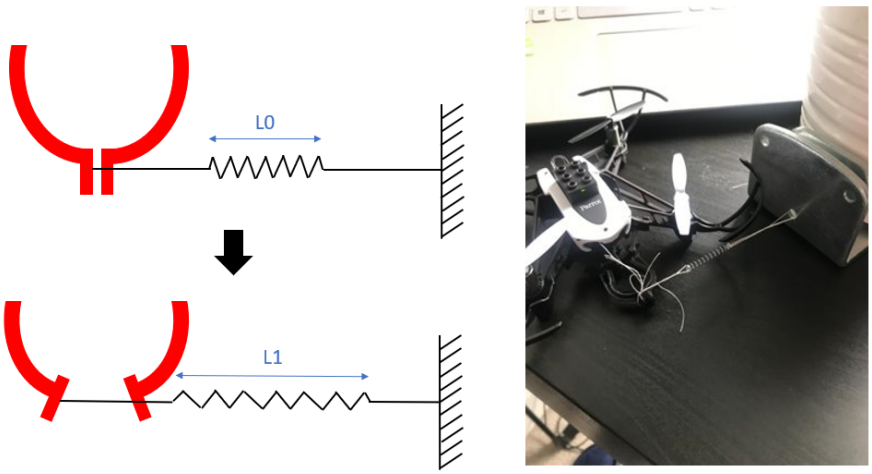


Figure 3.26: Gripping Force Test Rig: Diagram (left), Real Testing (right)

The results of this test are presented in Table 3.7.

Table 3.7: Results of the grabber grip force tests

Aim of the Analysis	Physical Property	Value	Absolute error
Spring constant	Mass of the load [g]	124	0.5
	Force of the load [N]	1.216	0.5
	Initial Length of the spring *L0* [mm]	26	1
	Final Length of the spring *L1* [mm]	33	1
	Elongation [mm]	7	2
	Spring constant [N/m]	173.7	50.4
Grabber Griping Force	Initial Length of the spring *L0* [mm]	26	1
	Final Length of the spring *L2* [mm]	30	1
	Elongation [mm]	4	2
	Gripping force of the grabber $F_{grabber}$[N]	0.70	0.55

The gripping force of the grabber is calculated to be 0.70N. However, a high relative error was observed when measuring the length of the spring (both initial and final) as the values were small with absolute errors being relatively high resulting in significant discrepancies for the gripping force of the grabber.

It can be seen from the Tables 3.7 and 3.5 that the egg can withstand $6.19 \pm 0.005N$, which is significantly higher compared to the $0.70 \pm 0.55N$ force exerted by the grabber. The worst case scenario shows a safety factor of 4.95, justifying that the provided grabber will not break the egg when collecting it.

3.3.3.4 Analysis of grabber-egg interaction during flight

After collecting the egg, the drone must carry it along the track to its final destination. This section outlines the analysis performed on the capability of the grabber to maintain grip on the egg during flight. Two major forces act against this effort, that is:

- Inertial force on the egg as the drone accelerates upwards during takeoff
- Gravitational acceleration on the egg causing slippage during flight

An additional force is neglected here, that is the aerodynamic forces induced on the egg due to the flow of air downstream of the drone's rotors. This assumption is justified as it can be seen in Fig. 3.27 that the airflow downstream of the rotors do not coincide with the egg. Vorticity effects causing flow field to expand downstream of the rotors can also be neglected due to the eggs positioning vertically with respect to the drone's rotors. The study conducted by Yoon et al, 2017 [8], agrees with this observation. Qualitatively this was found to be the case, though further CFD analyses may be necessary to test this assumption.

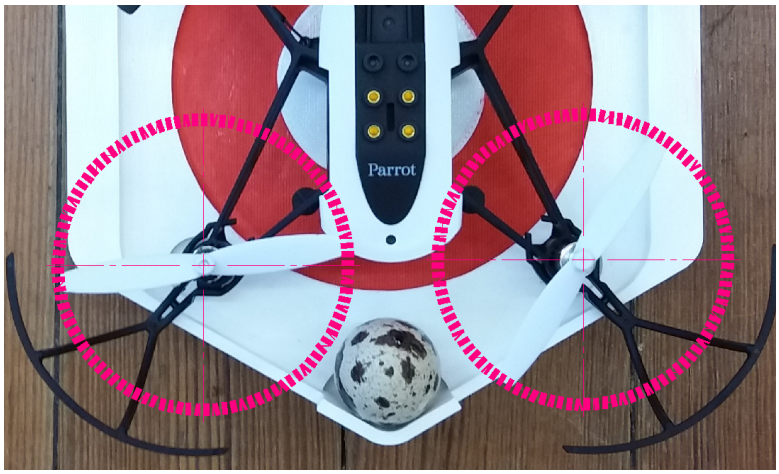


Figure 3.27: Rotor flow profile

Figure 3.28 shows the two main forces acting on the egg during the lift off. As this is the most critical part for the grabber to hold the egg because both forces act downwards, it is sufficient to analyse only this part of the flight.

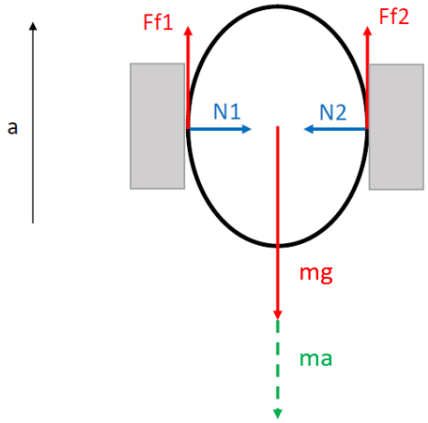


Figure 3.28: Free body diagram - Grabber holding the egg

The grabber motor stops generating force after it has grabbed the egg. Deformation of the foam material resulting in a spring force on the grabber arms presses on the egg and as a consequence frictional force is generated and holds the egg in the grabber. The frictional force can be calculated as follows.

$$F_f = N\mu \tag{3.5}$$

Where F_f is the frictional force, N is the normal force acting on the egg and μ is the coefficient of friction between the egg and the grabber.

To prevent the egg from sliding off the grabber arms, maximum frictional force has to be greater than the two forces acting downwards (inertia and gravitational force).

$$F_{fmax} = m(a + g) \tag{3.6}$$

From the free body diagram in Fig. 3.28 we conclude that

$$F_{fmax} = F_{f1} + F_{f2} = (N_1 + N_2)\mu \tag{3.7}$$

$$N_1 = N_2 = 0.7 \pm 0.55N \tag{3.8}$$

The coefficient of friction μ is found to be 0.463 [14]. By using Eq. 3.7 and 3.8 the maximum friction force and total force acting downwards on the egg are calculated as follows. The acceleration for the take-off is set to 0.8 ms^{-2} by the Control Systems team.

$$F_{fmax} = (N_1 + N_2)\mu = (0.7 + 0.7) * 0.463 = 0.648N \tag{3.9}$$

$$m(a + g) = 2 * 10^{-3}(0.8 + 9.81) = 0.021N \tag{3.10}$$

Clearly $0.648N > 0.021N$, therefore the egg stays in the grabber during the flight.

In summary, the stock grabber has been evaluated and was found to be sufficient for the task and therefore can be used without any design changes to achieve the client's goal. Moreover, this rules out completely the need to develop an external grabber attachment as the benefits do not outweigh the added mass (and therefore increased energy consumption) and added complexity of balancing the drone's centre of gravity and lift due to the added mass being located further out from the centre compared to the stock grabber.

3.3.4 Landing Platform

3.3.4.1 Overview

During the egg collection phase of the task, two major problems were identified. Firstly, for a fully autonomous procedure, the egg must be placed such that sensing the location of the egg is possible for the done, allowing the drone to land towards it for collection. Secondly, the airflow downstream of the drone's rotors as it lands pushes the egg off-target from the landing zone, necessitating a method of securing the egg in the intended collection location. To this end, it was decided that solutions to both these problems could be incorporated into a landing platform. The functional requirements of the landing platform are outlined below.

1 Function

1.1. Remains secure to the ground during drone landing and takeoff

1.2. Secures egg in intended collection spot during operation

1.3. Provides sensible element that allows drone to detect the platform

2 Integration

2.1. Efficient sensing of location by the overall system

3 End consumer use

3.1. Ease of use to the client

3.3.4.2 Method

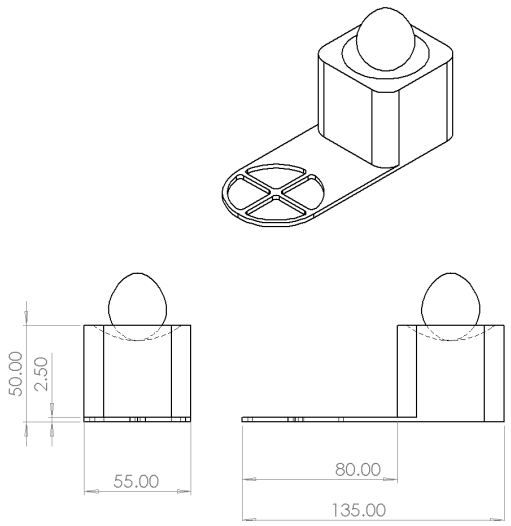


Figure 3.29: Initial landing platform design

The design was first approached by attempting to complete the task as simply as possible. Shown in Fig. 3.29 is the initial design of the landing platform. The egg is secured in a divot on a raised platform. The drone will then hover over the cross feature as it collects the egg. This design was unfeasible as it required the drone to hover reasonably still as it collects the egg, requiring a high degree of landing accuracy from its control system. It was also found that the drone tended to hover due to a cushion of air forming underneath it [13]. Further, the distance between the target cross to the egg pocket was unmeasured and untested at this stage, meaning that if the design were printed and found to perform unsatisfactorily at this stage it would be a waste of resources. Finally, the block-like shape of the platform may act as a bluff body, experiencing aerodynamic forces as the drone comes down to land thereby pushing it off-target.

The next design was intended to be a device to obtain the necessary measurements and information to carry over into the next design iteration. This was an important step to minimise wasted time and resources as it allowed for the collection of as much data as possible regarding the landing platform and its required elements. To a lesser extent, the design also explored the possibility of minimising the size and therefore cost of the landing platform. The design is shown in Fig. 3.30. It is a compact, 'matchbox' design with a part that slides

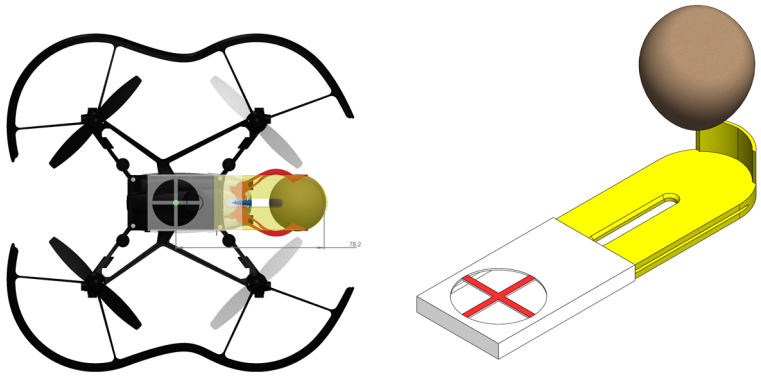


Figure 3.30: Second iteration of the landing platform. 78mm measured distance (Left), Isometric view (Right). CAD model of the drone was provided by Parrot.

into the other. Shown in the same figure is the measurement that was intended to be collected during testing. The reference distance, measured from the drone's camera (coinciding with the centre of the cross) to the inside edge of the guard, was a design distance that was intended to be fixed in the next iteration. Further observations are that the drone would sit at an angle to the base due to the platform interfering with the grabber module, as well as the platform being so lightweight that the inside guard acted as a drag machine which caused the drone to sail forwards when the rotors were powered. A real-life image of this is shown in Fig. 3.31.

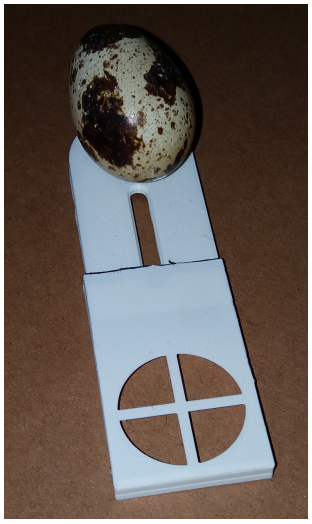


Figure 3.31: 3D-printed second version of the platform with a quail egg

On the third iteration, a helipad-style landing platform design was chosen. This was done mainly due to the drift effect experienced when flying in low (roughly 0.3m) altitudes. The design also incorporated an improvement to its egg-securing feature, now featuring an indentation on the egg pockets to further retain the egg. Another major change is the painted red circle forming a concentric circle pattern which acts as a detectable feature by the drone to trigger the landing and collection procedures. The drone is also lowered into the platform by way of indentations on where the feet lay in the platform, the depth of which sets the vertical height of the drone. The design is shown in Fig. 3.32. One major flaw identified was that the egg still did not remain firmly in place during operation. The measured distance of 78mm was found to be suitable as the grabber succeeded in grabbing the egg always. Finally, success in circle detection was demonstrated in a test, the output image seen in Fig. 3.33. The distinct concentric circle pattern seen acted as a trigger to initiate the landing and collection phases of the task.

The final design is shown in Fig. 3.34. The finalised design features a through hole on the egg pockets which allows airflow and prevents blow-back from reflected air, keeping the egg in place. Two minor changes are the indented inner circle which now fully clears the grabber, as well as a filleted profile on the drone feet guides to slide the drone into place thus helping it to land with a higher success rate. Success of the landing platform was demonstrated by manual testing of the drone. It was found that the egg remained securely in place as the drone took off, as well as during hovering of the drone over the egg. Further, it was found that accurate landing of the drone on the platform was possible, though inconsistent. In later sections, the design of a precision landing

mechanism is discussed to augment this system. On a final note, a technical drawing of the finalised landing platform is presented in the appendix of this document.

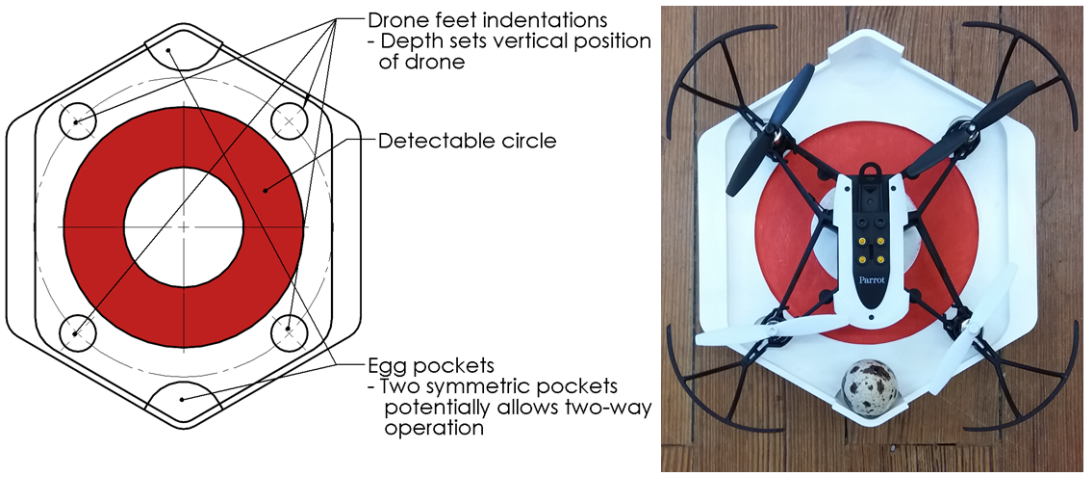


Figure 3.32: Third version of the landing platform. Highlighted key features (Left), 3D-printed (Right)

Figure 3.33: Image output of the drone flying over the platform

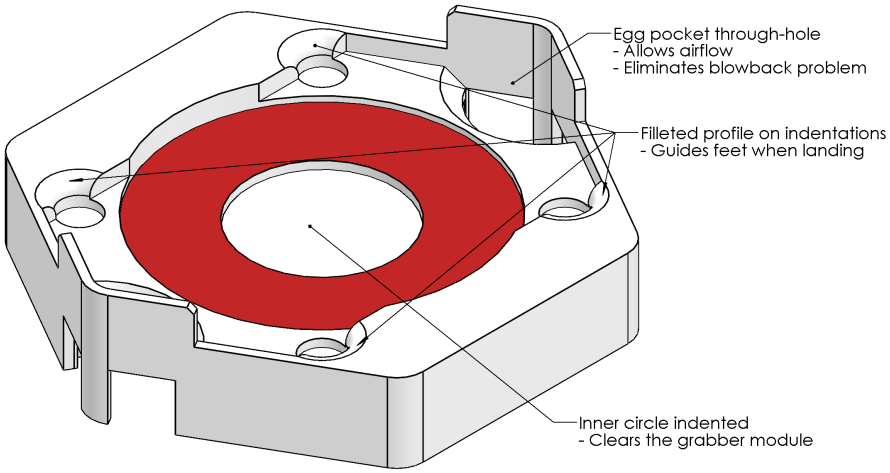


Figure 3.34: Final version of the landing platform

3.3.5 Precision Landing Mechanism

After some testing on the landing procedure was done it has been observed that the landing is not consistent meaning the drone does not consistently land on the same spot in each landing. The platform features cone shaped pockets for the feet of the drone to land in which mitigates the landing inaccuracy slightly. However, testing has shown that there were times that the landing was not precise enough for the grabber to reach the egg. Since the position of the drone is critical for collecting the egg from the platform, an aligning mechanism is developed to improve landing accuracy.

1 Function

1.1. Secures the drone in one specific place that is ideal for the egg collection

1.2. Can be used for repeatable landing

2 Integration

2.1. Implementation within the landing platform

3 End consumer use

3.1. Ease of use to the client

3.2. Reasonable price

3.3.5.1 Permanent magnets

The first and simplest method to fix the issue mentioned above is to position a set of permanent magnets in each corner of the platform where the feet of the drone sit upon landing. Once the drone approaches the platform, it is attracted by the magnets and aligns to an ideal position to commence egg pick-up. Since there are no metal parts on the bottom of the drone that would be attracted, four 3D-printed cylinders that are attached to the bottom of the drone's feet are designed. At the bottom of the cylinder there is a pocket to hold a metal piece that is attracted by the permanent magnets on the platform (See Fig. 3.35). The cylindrical holders are attached to the drone using double-sided tape. One cylindrical holder including a metal piece weights 0.5g resulting in a total 2g additional mass to the drone.

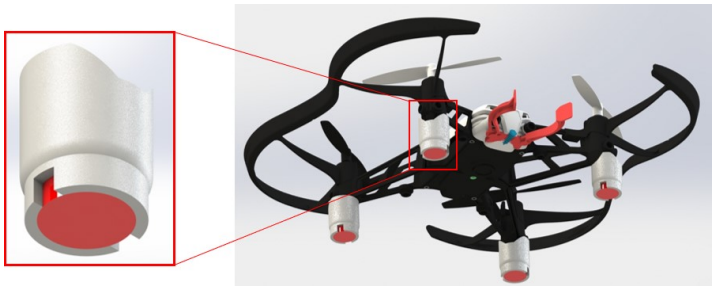


Figure 3.35: Drone with a detail view on a cylindrical metal holder

The limitation of this design is that the drone has to overcome the attractive force from the permanent magnets on the platform during lift-off. The drone's maximum thrust is only 51g (0.50N) [15]. This means that the maximum pull force of a magnet can only be 0.125N if using 4 magnets (one for each foot of the drone) or 0.25N if using 2 magnets (placed in diagonal corners). Such strong magnets might not be able to attract the drone from a distance. To prove this, an FEA for magnetic fields using FEMM software is done. The permanent magnet used in the analysis is made of out Y25 ferrite which is a material commonly used for small ring permanent magnets [16]. The magnetic flux field is plotted in FEMM and presented in Fig. 3.36.

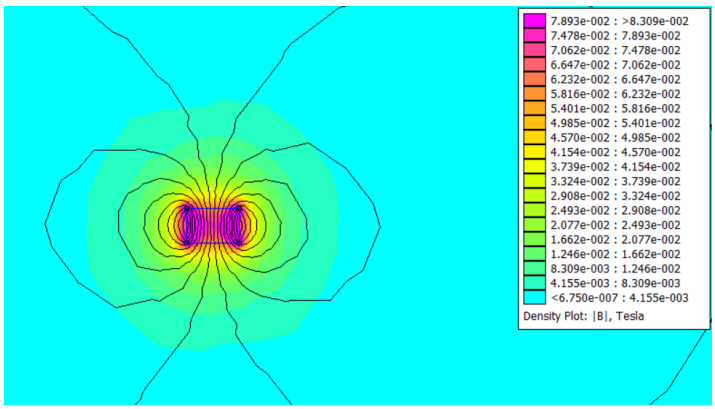


Figure 3.36: Magnetic Flux Field (B) around a permanent magnet

The magnet analysed is a $15mm$ diameter and $10mm$ height ring magnet. The plot above gives information about the magnetic flux surrounding it. By using Eq. 3.11, the force around the magnet can be calculated.

$$F = \frac{B^2 A}{2\mu_0} \tag{3.11}$$

Where B is the magnetic flux density, A is the surface area of the magnet and μ_0 is the constant permeability of space.

A few points in the y-direction (upwards) from the magnet are analysed to see what forces are present there.

Table 3.8: Magnetic forces around a permanent magnet

y distance [m]	Magnetic Flux [T]	Force [N]
0.005	$34.414*10^{-3}$	$83.274*10^{-3}$
0.01	$18.723*10^{-3}$	$24.647*10^{-3}$
0.05	$2.061*10^{-3}$	$0.299*10^{-3}$
0.1	$0.508*10^{-3}$	$1.82*10^{-5}$

From Table 3.8 it can be seen that there are small forces present around the magnet. At the surface of the magnet the maximum force is calculated to be $3.8N$. This means that the drone would not be able to takeoff from the platform. At the same time, the attractive forces are not strong enough to help the drone to align, therefore, this solution is not very feasible for improving the precision of landing.

3.3.5.2 Electromagnets

Since the permanent magnet option is not feasible, the use of electromagnets is considered instead. The advantage of this system is that the electromagnets can be activated and deactivated using a controller, therefore, their strength can be rated much higher than that of permanent magnets. There are no risks for the drone not being able to lift-off due to the ability of the circuit to be disengaged once the drone sits on the platform.

The control system is designed such that a sensor detects the drone when it flies above the platform (ready to land). Consequently, the electromagnets that attract the drone to help it land are actuated. Once the drone lands and collects the egg, the electromagnets are switched off and the drone is free takeoff. Figure 3.37 shows the sequence of the control system.

Figure 3.37: Sequence of the control system

An Arduino board is used to control the system. There are two sensors within the circuit; an ultrasound receiver that detects the drone being above the platform and a photoresistor that checks the change in light once the drone lands indicating a successful landing.

Figure 3.38 shows the circuit diagram generated using Fritzing Software. To be able to test the Arduino module, instead of using the actual drone for the ultrasound generation, an ultrasound sensor including both the transmitter and receiver is used. Pin 10 sends signal to the transmitter to emit ultrasound waves every $1ms$. The receiver senses the waves as they reflect from the obstacle above. The signal is received on pin 9 and the distance between the object and the sensor is calculated. Once the distance is less than $50cm$, the Arduino sends current to the circuit with the electromagnets through pin 12. The LED turns on to indicate that electromagnets are actuated. Once the drone lands on the platform, it creates a shadow on top of the photoresistor that senses the changes of light using an analogue reading. Once the value changes and exceeds a designed threshold, the Arduino cuts off the current on pin 12 (the circuit with the electromagnets) after a little delay to make sure the drone has time to align. At this point, the circuit stays turned off and thus there are no electromagnetic forces preventing the drone from lift-off.

Figure 3.39 shows the actual Arduino circuit that was built to perform a few tests to prove that this concept works. The actual code for the Arduino is attached in the appendix.

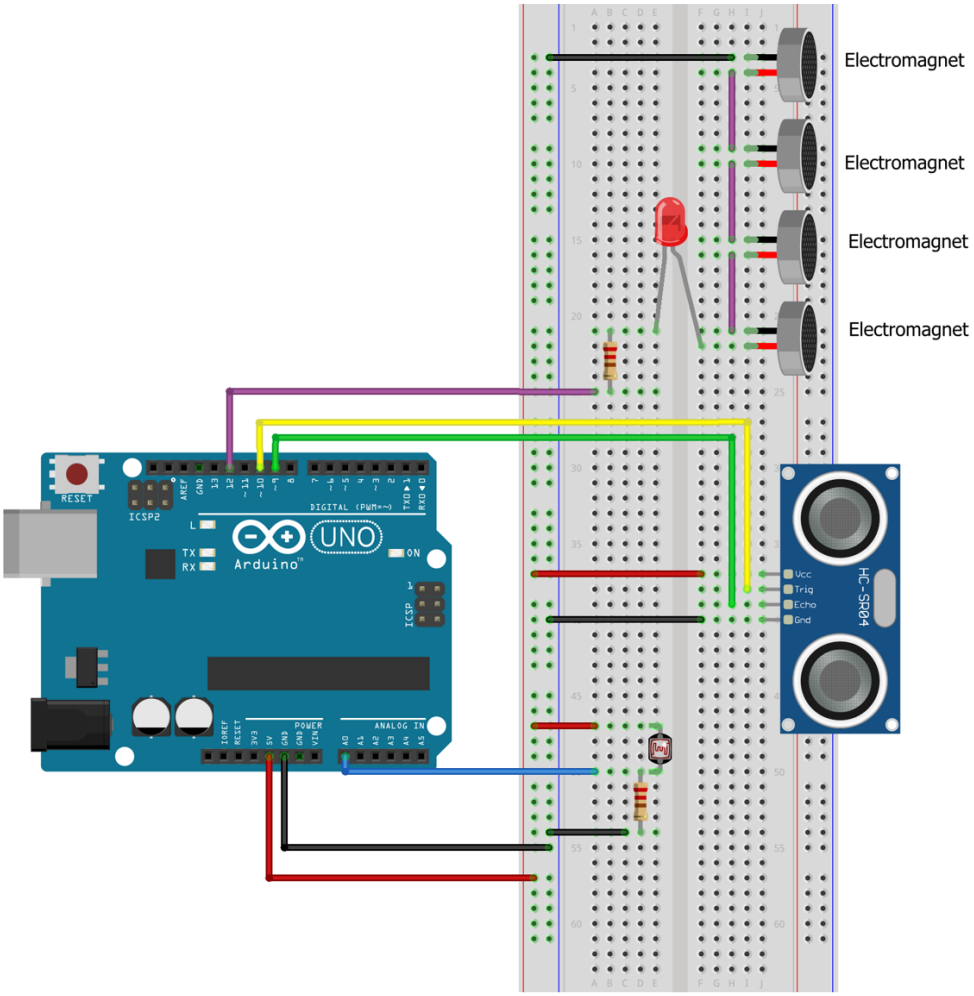


Figure 3.38: Control circuit diagram

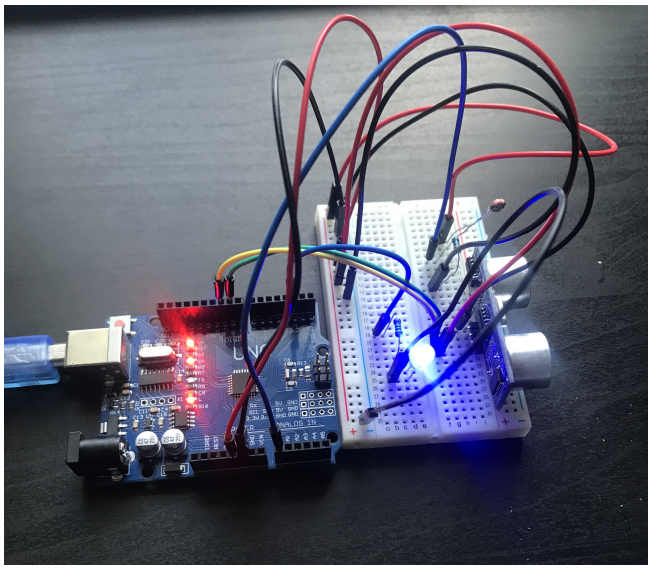


Figure 3.39: Control circuit testing device

To build this module, there are several components that must be sourced. For our application the majority of components were our own therefore no cost was incurred. However, for the client's application the cost analysis has to be taken into account. Table 3.9 gives a cost breakdown of components needed to build such a circuit.

Table 3.9: Precision landing module cost breakdown

Component	Price per item [£/item]	Quantity	Total Price [£]
Arduino Uno	4.19	1	4.19
9V Battery	2.00	1	2.00
Electromagnet (5V)	9.99	4	39.96
Ultrasonic Receiver	2.95	1	2.95
Photoresistor	1.02	1	1.02
Others (resistors, wires, LED etc.)	2.00	1	2.00

1 Arduino Uno is used for our application. To reduce the cost an equivalent Uno board with the same capabilities can be used [9]

2 The electromagnets used have to be rated to 5V DC because that is the maximum voltage that the Arduino board can deliver [9]. The strongest electromagnets available for this voltage have 5kg holding force [17].

3 The ultrasonic sensor consists of the transmitter and the receiver [18].

4 The photoresistor reacts to light changes [19].

The total price for the precision landing mechanism sums up to 52.12£. For the particular application it is significantly costlier than the other components and therefore this might be a crucial factor for not developing this system. However, the costs can be reduced by using aftermarket or secondhand parts.

To implement the sensors and electromagnets into the whole system, a few changes to the landing platform must be made. The core features of the platform remain unchanged (see Fig. 3.40).

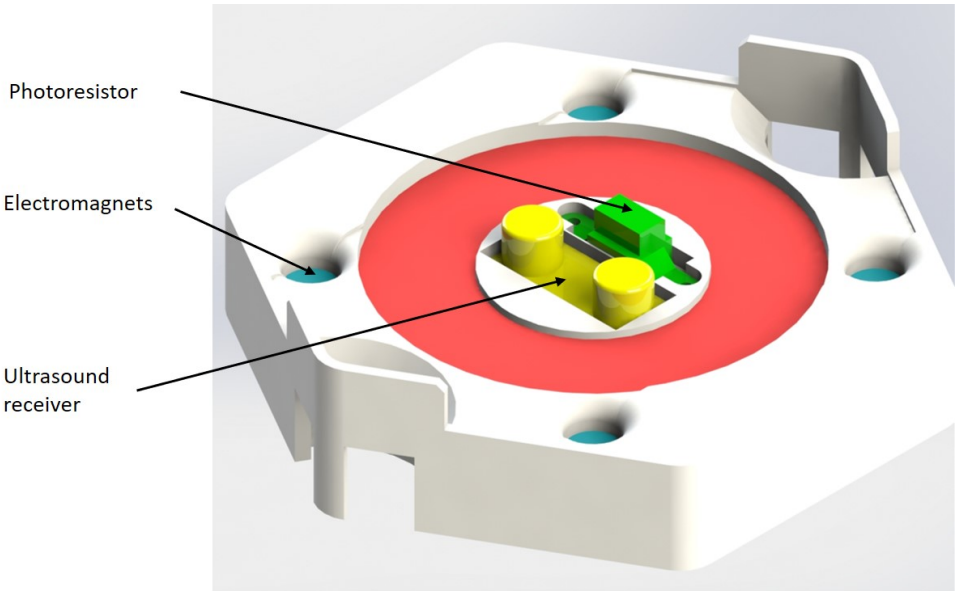


Figure 3.40: Landing platform with implemented precision mechanism

3.3.6 Discussion

The decision to use the stock grabber was justified as it has been shown that consistent grabbing of the egg was achieved, along with evidence that the grabber can safely manipulate the egg with no risk of losing grip during the task or in breaking or inducing breakage to the egg. A landing platform system has also been designed and demonstrated to be successful in its intended task, along with plans laid out to further augment the system with a precision landing mechanism.

To reiterate, the use of the stock grabber was justified as it succeeded in performing the task without added complexity or added mass, however the current system works only for quail eggs or similarly shaped objects. Modifying the design will be necessary should the client wish to extend the different types of payload applicable to the task, which may necessitate the development of different grabber extensions or different pockets on the

landing platform to better fit different objects. Further, there is scope for optimisation with the drone's cargo carrying capabilities. Should the end use case of the drone is to be a dedicated 'delivery' drone, it should be noted that the drone can output a maximum of 51g of thrust [15] with a maximum flight time of around 7 minutes [12] depending on the weight carried or computational power required of the drone. This means that for this application, this project should be used as a starting point for further research rather than a working design as there is likely no appreciable or practical use due to the drone's hardware limitations.

The landing platform is made of 3D-printed PLA filament, which suits the purpose of this project due to the complicated geometry and availability of both the process and material. If the product were to be mass-produced, a different manufacturing process such as vacuum forming or injection moulding may be a better alternative due to the long manufacturing times inherent to the 3D-printing process. From the perspective of sustainability, the PLA material used gives some degree of recyclability for the landing platform. With a simplified design, wood may also be a good sustainable alternative material choice for the landing platform.

The precision landing mechanism must be first tested to prove its functionality. Due to time constraints this has not been done and therefore the theoretical stage has been reached only. In theory, the precision landing module can implemented for applications using different drones not just the supplied Parrot Mambo. Since the magnet solution is used, a set of metal pieces must be implemented within the drone to provide a medium for the magnetic attraction. This results in putting extra weight of 2g on the drone. Further testing over the track needs to be done to analyse the effect of this additional weight on the overall flight performance. The sensors used within the electromagnet solution may be easily changed depending on the nature of the task, however, the principle remains the same. Instead of using an ultrasound sensor that might interfere with the Parrot Mambo drone ultrasonic height detection, a LiDAR or other type of sensor may be considered. The photoresistor is feasible for this specific application because the presence of light during the flight is essential for the drone's orientation. Alternatively, an infrared reflective sensor could you be used in case the photoresistor does not suit the client's requirements. For applications with heavier drones, the electromagnetic alignment system might not be feasible due to the greater weight that would need to be attracted, necessitating more powerful magnets that will be required a greater amount of electrical power. The option of using 2 magnets placed in the diagonal corners might be sufficient enough to align the drone, however, further testing to prove this assumption would be required.

4 | Project Validation

The scope of the drone system is limited to an indoor setting, with good lighting conditions, and preferably no wind interference. While the system in isolation from physical constraints and imbalances has proven to be fundamentally correct, at the time of writing it is not a solution that has truly been built to its true potential, due to technical limitations, time restrictions, and lack of a consistent testing space.

The supplied grabber has been demonstrated to be an ideal solution to the project. Mass optimisation could only be achieved by redesigning and manufacturing a completely new grabber, adding complexity to integration of the grabber due to limited control capabilities. Further, the consistent success in grabbing the egg, demonstrated in the presentation and flight demonstration, justifies the choice of using the supplied grabber. In conjunction, the developed landing platform has been demonstrated to mitigate the aerodynamic effects causing the egg to not remain firmly in place, also providing a secure base for the drone to land and takeoff from. The development of the precision landing mechanism shows promise in increasing the project's chances of success.

Moreover, the lack of data on the sensors' and the drone's hardware in general compromise the integrity of validating the landing sequence. Until the very end the data from the altitude sensors tends to demean the estimation of the height of the drone, perhaps due to errors arising in the confluence of the sonar and pressure sensors. However, we stand firm with our simulations to confirm that there is an adequate degree of path following and navigation.

When the drone follows the red track, it moves left or right, along the y-axis, and forward, along the x-axis. This means that the drone is capable of following tracks that have both straight corners and curved edges. However, if the given track had a 90 degree bend or a curved segment that looped once around itself, the path following algorithm would need to be rectified. This is mainly due to the fact that the drone's yaw is not adjusted for track alignment, which was a design decision made to meet the requirements of the track given.

The majority of testing was performed on the Simulink simulation which did not account for noise and shadowing, this meant that further issues were discovered when hardware testing began. One of these issues was that, occasionally the drone would not detect the circles on the track.

In the context of the 5-meter track, the battery life is appropriate for covering several runs in a single session. However, when considering that the project requires hours of judicious testing and verification for each phase of the problem, a more obvious issue presents itself. This was slightly resolved by the availability of the second battery, but even then, a few genuine considerations were made to plan the testing before going ahead to conduct a few test flights over the track. Additionally, this becomes a more significant concern when the length of the track increases beyond 10m.

As such, the delivered system is validated for the testing environment of classroom 2, however, if intended to be used in a more flexible "delivery" context, we cannot fully validate our solution. If granted the luxury of time and perhaps additional processing resources, more thorough testing could be achieved and a more adaptable framework could be developed. This would ensure the project requirements have been met alongside introducing a capacity for adaptability to unforeseen environments without further design.

5 | Ethics

5.1 Introduction to Drone Ethics

As a disruptive technology [20], drones carry with them ethical concerns inherent to both their nature and the opportunities they provide to their users. This can be seen in the microcosm of the project drone described above, alongside the wider use of commercially available drones. Concerns range from the cost savings a drone may offer a business to the litany of remote sensors they put into a private user's hands. To provide the client with more context concerning drone operation, the main areas of ethical debate will be discussed focusing on the civilian usage of drones. Safeguarding measures and regulations currently employed will be detailed, alongside an analysis of how these apply to the project drone.

5.2 Privacy

The remote cameras often integrated into drones pose a threat to privacy as they serve as a covert method of gathering images, especially against a person who is in a private place. This is viewed by the UK's Civil Aviation Authority (CAA) as a breach of data protection law [21] and carries with it punishments appropriate to the severity of the breach. Ethically, this extends to operating drones for filming purposes in public spaces. While legal, before the proliferation of drones, public recording came with the expectation of the camera and operator being located together. As such, if the drone operator is unable to be easily identified, their ability to record is legal, although ethically questionable.

The consensus is that in public, consent should be sought by the drone operator from the people who are intended to be recorded. Failing this, their identities should be made ambiguous in the footage by for example, blurring their faces either through the nature of the shot or in post-production. Meanwhile, the drone operator should be easily identifiable and approachable [21]. In any case, the recorded video should be handled following data protection law, notably, it should only be retained for the minimum time necessary for its purpose; disposed of appropriately when no longer required; and stored securely throughout use [22].

The project drone includes a camera, however, the images generated are processed internally, outputting a low-resolution rendering of the world space in black and white. Such an output cannot identify a person and simultaneous high-resolution recording of the camera feed is negated by the drone's limited processing power. As such, the drone does not pose a threat to individual privacy, however, the aforementioned laws must be followed regardless during operation.

5.3 Safety Hazards

Drones pose a hazard to public safety by virtue of being an object with mass and mobility. As such, they can be flown into areas that endanger lives. A notable example of this was the 2018 incident where reported sightings of drones near Gatwick Airport forced a shutdown with approximately 760 flights affected [23]. This issue has only grown with a 2019 flight into Gatwick being redirected as a result of drone encroachment on the flight path [24]. Protestors have planned to use a similar strategy to hold an airport hostage as expressed by climate campaigners in 2019 [25]. This could extend to disruption of military installations and power stations threatening national security if unchecked.

These incidents prompted safety measures such as the Drone Code which provides guidelines to drone users, aiming to avoid inadvertent trespassing and potential incidents [21]. Notably, drones are excluded within a 5km radius of airports with additional 5km long and 1km wide restricted areas covering the flight path of aircraft

onto the runways. To combat malicious activity, drone detection and interdiction methods have been developed. These include visual, acoustic, and RF detection of trespassing drones which aim to identify drones before they become a serious threat. These are combined with signal jamming or kinetic attack and capture of drones to neutralise them once they have been confirmed as a danger, but before they can do harm [26].

Also, as with the project drone, commercially available drones can carry payloads. They can lift a mass, giving it gravitational potential energy and then release it; in the wrong hands becoming a weapon. Although, this is limited by the carrying capabilities of the drone and the degree of control the pilot has over it. To combat this possibility, in addition to established no-fly zones, drone flight within 150m of gatherings of 1000 people or more is prohibited [27].

To protect against these and further drone safety hazards, a licence is required for the operation of drones over 250g [27]. The operator must pass an online exam to become licensed and must renew this annually costing £9 [28]. This provides a baseline level of knowledge for safe drone operation and thereby allows the user to be held accountable for their actions in case laws are broken. Also, the necessary renewal keeps the user updated with ever-changing drone technology and laws, while also funding the programme.

The project drone is 77g with grabber attached and has a payload of 4.3g. As such it does not require a licence to operate and does not pose a threat to public safety. However, its operation should be limited to indoors to avoid the risk of damage due to environmental factors, loss of control in a public space, or trespassing into a no-fly zone.

5.4 Automation

While implemented on a small scale in this project, the large scale and well-funded development of autonomous drones serve to provide both benefits and pitfalls in modern society. Free-flying drones offer the ability to automate repetitive tasks and allow access to dangerous locations without endangering human lives. However, with the possibility of increased production efficiency and reduced overhead costs, companies may consider employing drones wherever possible.

This means that autonomous drones are a threat to job security for many, namely in the infrastructure, agriculture, delivery, and transport sectors, potentially replacing $127 billion of human labour [29]. To companies in these sectors, drones are the next step in service and supply chain development with a case in point being Amazon Prime Air. Through this venture, Amazon aims to deliver small packages within 30 minutes of their purchase via drone. It is estimated that package delivery via drone will cost $0.05-$0.07 per trip while an equivalent human package delivery would cost $1.20 [30]. This shows the potential cost savings for a company which thereby incentivises drone usage in place of human labour.

This phenomenon is not new however, throughout history disruptive technologies have replaced human labour most notably during the industrial revolution and the rise of the internet. Automation presents a similar paradigm shift meaning its repercussions will be similarly far-reaching. Ethically, there always has been much debate on this matter. Acknowledging the past waves of technological advancement, technologies will be adopted regardless of their human impact if they are commercially valuable. The debate now centres around providing workers with replacement jobs or methods that can be applied to ensure their continued livelihoods.

This can take the form of a universal basic income, where for example people of working age would be paid $1000 per month as proposed by US Presidential Candidate Andrew Yang [31]. While automation puts human labour in general at risk, it disproportionately affects the less educated [32] meaning that employers must seek to provide education and retraining, particularly for lower-skilled workers. While no concrete solutions have been implemented, the threat of automation and by extension autonomous drones has been identified. As predicted by PWC, autonomy will become pervasive in the 2030s [32] setting a deadline for a solution to the aforementioned issues.

The project drone's level of automation does not render it a threat to society, however, it displays what can be achieved with little to no time and budget. When extrapolated to a commercial level and with ample funding and development time, autonomous drones quickly become disruptive to the sectors they are implemented in. Their capabilities only increase with time, and as a technology, pose a threat to the livelihoods of many.

5.5 Summary of Drone Ethics

With new technology, the ethical impact cannot be judged immediately as it takes time to understand the capabilities presented and how they can be exploited. Accordingly, while sometimes based on the precedent set by previous technologies, legislation and ethical guidelines are usually reactionary. However, as drones have established themselves on the commercial market, guidelines have matured and taken shape to form a robust framework for drone operation.

The project drone's capabilities are indicative of the state of drones years prior, falling into a low weight class and possessing limited camera and processing hardware. Resultantly, its operation has already been codified and the client faces little ethical quandary as to its operation.

6 | Open-source Drone Development

6.1 Introduction to Open-source Systems

While the Parrot Mambo and its supporting architecture were employed for the project, notably for their availability, ease of use, and shallow learning curve, other solutions are available. A distinction can be made between the proprietary system that the project focused on and an open-source system. For this project, open-source will be considered as open-source software as defined by OSS-Watch [33] combined with modular hardware that is designed to allow core components to be interchanged [34].

Such a system presents advantages over the currently employed architecture from a development and end-user perspective, however, it lacks features integral to the developed drone. This chapter will contrast the two approaches and assess what system the drone should utilise if redeveloped.

6.2 Limitations of the Parrot Mambo

6.2.1 Computational Limitations

The computational hardware of the Parrot Mambo lacked the processing power to interpret the camera's video feed at an acceptable frame rate in its default form. The resolution was decreased to increase frame rate with a compromise being achieved at 48*36 pixels. In the simulated environment fig 3.14, the simulation runs at a video refresh rate of 30 frames per second correlating to smooth path following. However, in physical testing, the drone exhibited a frame rate of $\sim$ 1fps.

In this state line detection was operable, although, with the integration of circle detection the frame rate decreased further, not refreshing on the live viewer. This integration also reduced the consistency of path following and introduced random errors where the drone would stray away from the path. However, increasing the frame rate to improve reliability was now impossible as this would come at the cost of image fidelity making the track indistinguishable.

To integrate the egg transport system to the already stressed hardware was therefore unfeasible. Any additional procedures would reduce the frame rate to a state where drift between *dx* and *dy* inputs would have a larger effect than the inputs themselves.

To further these issues, due to the lack of system datasheets, it is impossible to relate specifications like processor clock speed and amount of RAM to an operable camera resolution and frame rate. Further optimisation cannot be made as we are not privy to details like the processor's instruction set to tailor our code to the fastest operations for the processor. Additionally, without this information, it is difficult to know the performance threshold that must be crossed to run the developed software.

The optimisation and refactoring process described by MV in section 3.1.4 was essential to delivering the drone in its current state. However, this prolonged development and delayed systems integration and testing. While this is also a time limitation issue, the restrictive hardware is the root cause of these delays. Therefore, development time would be better spent adding the functionality integral to meeting client requirements instead of chasing diminishing returns trying to optimise for the hardware.

6.2.2 Connection Limitations

From a mechanical design perspective, there were limited possibilities for optimising the grabber as it used a proprietary connection format. This connector lacked data signals only utilising a positive and negative wire,

limiting its functionality to an on-off switch. This narrowed the scope of grabber development to retrofitting the arms themselves, and only facilitated rudimentary grabber controls. For instance, the presence of the egg could not be determined by the grabber. While the grabber did not need to be modified for the implemented design, developing more robust sensing with the grabber could reduce the reliance on other systems like the platform or the landing state in the FSM. This would increase the adaptability of the drone to different courses but would also increase computational overhead.

6.2.3 Software Environment Limitations

Software development utilised MATLAB whereas the Parrot Mambo executes code in C necessitating translation of the code. The downloaded MATLAB add-ons perform this task, however, during this code generation stage, many unexpected errors are also produced. As such, errors relating to managing data types, non-constant variables, and indexing required refactoring already running code.

These issues only become present once code is ported to the drone. This delays project progress as code verified through simulation cannot be trusted to run on drone hardware. These errors are pernicious as they occur in the critical system integration and testing stage which means that any delays in this activity will affect the overall project completion.

6.3 Open-source System Advantages

Open-source systems excel in modularity and upgradability. Open-source hardware is interchangeable allowing an engineer to utilise off-the-shelf components to meet project requirements at a system and subsystem level. This extends to software development as languages can be chosen for their suitability to the task at hand e.g. a low-level language like C++ for close to real-time computing in the drone's image processor.

Moreover, the software and the development environment itself is well documented and there are many resources to be leveraged. This would make for a quicker and simpler design experience due to the additional support and precedent for development set.

This paradigm of adding components as required provides a low barrier to entry from both a cost and complexity standpoint. Only necessary components need to be included minimising wasted hardware and thereby costs. Development is expedited as only the necessary components need to be understood and implemented saving time bypassing irrelevant but built-in systems. To further this, if the requirements then outstrip the components, further hardware can be added as needed. This is especially true to "future-proof" the design with the aid of increased computational power and decreased packaging.

Lastly, ownership of a developed product is clear with an open-source system. With proprietary systems, there may be legal ramifications to the modification of a company's intellectual property. Open-source hardware and software are intended to be modified with the licensee having the right to distribute and sell the developed product [35].

6.4 Open-source System Disadvantages

While open-source systems are more adaptable during development, proprietary systems once developed are better optimised and packaged for the intended task. Proprietary systems are made with components that are designed and manufactured specifically for their requirements. An open-source methodology would thereby manifest itself as a heavier and thus more fragile drone. The caveat is that within this project, a proprietary system is not being designed, rather one is being adapted to a fulfil a role outside the scope of its initial design. To this end, the optimisation and packaging of the proprietary system hinder development.

A proprietary system benefits from an increased level of security when compared to an open-source system [36] as with an open-source system the source code is freely available for anyone to read. While this allows vulnerabilities to be highlighted and fixed, if identified covertly these vulnerabilities can be used to undermine the system. For an open-source drone this may be problematic as if hijacked, the drone may be used as a weapon as discussed in section 5.3. Proprietary systems hide their source code through methods like obfuscation and encryption. While vulnerabilities may exist, they are harder to find than with open-source, with patches and updates available from the creator company also. This makes close-source as safe if not more than open-source.

Furthermore, for hardware, open-source safety is difficult to assess. A manufacturer can verify all the com-

ponents that comprise a proprietary system and certify that they work together. This cannot be said for an open-source system that has numerous interchangeable components. The number of component combinations is too large for the manufacturer to certify so they can only adhere to relevant communication protocols like USB and standards like those described by ISO. The expectation is that the other components intended to interchange within this framework meet the relevant standards also. As such the user takes a risk when interchanging components in an open-source system.

6.5 Implementation of an Open-source Framework

Considering the limitations detailed in section 6.2, a hypothetical open-source implementation of the drone can be explored. Hardware and software improvements over the Parrot Mambo are detailed alongside a design overview of their implementation.

6.5.1 Hardware

The Pixhawk series of flight controllers find their application best suited to autonomous drone flight, especially with the use of onboard machine vision as the means of path identification. The Pixhawk 4 is the most widely used flight controller with advanced autopilot capabilities for both academic and commercial purposes. The following would be the key hardware items required to assemble the same drone system, in an open-source framework:

- The Pixhawk 4 Flight Controller board itself which houses the main Flight Management Unit (FMU) processor and enough ports to connect to a variety of peripherals, including the motors. This board weighs 15.8g and should be mounted on the central structure of the quadcopter. It comes inbuilt with an accelerometer, a gyroscope, a magnetometer, and a barometer.

- The Power Management Board that houses the power supply, which is a Lithium Polymer (LiPo) battery, the Electronic Speed Controllers (ESCs) as well as the servos.

- The GPS/Compass module, which should be mounted on the frame but away from other electronics to reduce interference.

- A Raspberry Pi microcontroller which runs Robot Operating System installed with the MAVLink protocol and is compatible with the PX4 FMU. This would be the key component for introducing machine vision and grabber control capabilities to the drone system.

- Any lightweight and non-fragile camera that works with the ROS in synchronization with the IMU as the optical flow input. The combination of the video feed with the IMU information to the machine vision algorithms would be crucial to pattern recognition, and in turn, identifying the track and the landing circles themselves. Additionally, the hardware design would allow for more computationally intensive processes, such as the extraction of 3D geometrical features of the environment, and thus the quail egg itself would be detectable to the drone system.

6.5.2 Software

At the centre of all the drone's operations is the PX4 open-source autopilot flight stack software. This can control the quadcopter itself and is the interface between the Raspberry Pi controller and the proprietary sensors. It also part of the broader Dronecode platform which includes the QGroundControl station and MAVlink.

Mission planning and monitoring of the drone's flight status and location is done through QGroundControl installed on a separate laptop or PC. This flashes the PX4 firmware onto the Pixhawk 4 FMU to allow for drone setup, tuning the flight parameters and for viewing live (real-time) flight details to monitor whether the drone performs autonomously as expected.

MAVLink is a robust and actively accessed library that provides the API (which is cross-platform and works for C++, iOS, Python, Android, etc.) to manage the drone. This is also run on a companion computer (like a remote PC/laptop/mobile device) to program machine vision and route planning onto the drone system.

6.5.3 Open-source Design Overview

The open-source design would utilise a similar software structure to the current drone, as such the relevant changes are those to the hardware. These are illustrated in Fig. 6.1:

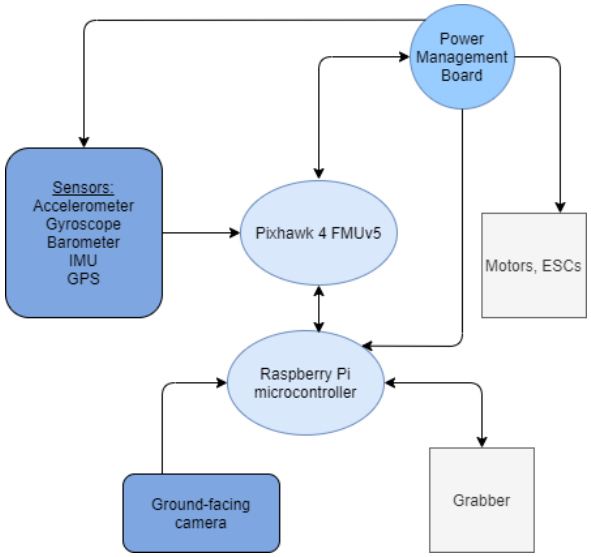


Figure 6.1: Open-source design overview

6.6 Evaluation of Proposed Open-source Design

Based on our experience with the Parrot Mambo the following should be made clear:

- There should still be considerations made in optimizing the algorithms to run on-board the drone because it is inevitable that the drone may crash several times. The open-source solution is more fragile and less robust to crashing compared to the system we worked and endured with. When the code crashed the Simulink model, the drone became unresponsive and was unable to continue flight. Translating to this to a possible open-source solution would mean that multiple emergency landing protocols need to be programmed to lessen the occurrence of crashes.
- The design should be modular, both in hardware assembly and in the software stack to grant systematic testing and debugging and allow for the sub-teams to work more independently.
- There are already a number of user-created solutions and code in the PX4 platform. This will allow for development time and energy to be spent on adding the functionality specifically required for this project rather than to create custom solutions for routine areas.

6.7 Summary of Open-source Drone Development

The current proprietary architecture used to develop the drone limited the scope of the developed product. This was largely due to a lack of processing power hindering systems integration. As such we have proposed an open-source design that will allow us to add additional computational power while maintaining the ease of use of the Parrot Mambo system. For the client, the advantages of an open-source system outweigh the disadvantages as the delivery of a working product is paramount. The expandable capabilities an open-source drone offers ensures requirements can be met within project schedule either through software optimisation or hardware substitution. For the client security is a minor concern considering the drone is intended to operate indoors on a 5m long track. As such the PX4 open-source framework is recommended for the further development of this project.

7 | Conclusions

7.1 The Delivered Product

The project drone as delivered meets the client's requirements in a number of key areas. Autonomous flight through the course has been achieved with the drone taking-off; following either the straight or curved line section; and landing in the second circle. This can be attributed to the successful implementation of image pre-processing followed by line following and circle detection on the drone hardware. These algorithms have been optimised to utilise the limited hardware available. Optimisation involved reducing the camera's resolution from 160x120 to 48x36 pixels and circumventing the traditional Hough Transform algorithm. These combined made the core vision sub-system, line following algorithm, 9.2x faster than initially executed.

This data feeds into a working FSM which effectively stages the flight based on where in the track the drone is, gleaned from the number of circles passed. This controls the drone during the appropriate stage of flight through the tuned PID controllers. These notably have been developed to deal with discrete inputs, a result of the low camera refresh rate, and exhibit a 43% reduced settling time when compared to the standard controllers.

With the MV and CS software combined, the drone exhibits a full track completion time of 25.80s with a percentage match to the track of 42.59% in the simulated environment as described in Table 3.4. This combined with the physical demonstrations of drone navigation in CR2 should instil confidence in the client that the free-flying requirement of the drone has been met.

Moreover, the drone was successfully able to manipulate the egg through the take-off, flight, and landing phases. Egg handling itself is attributable to the supplied grabber. This was assessed for efficacy, notably meeting requirements with a gripping force of 0.7N, which was enough to hold the egg firmly without exceeding the 6.19N crush force.

To augment the landing and take-off phases, a platform was created. This held the egg under the drone's take-off thrust of 0.5N maintaining a repeatable egg location for the grabber to index with. Also, precision landing features were implemented, namely a concentric circle targeting pattern and conical feet indentations. These reduced the precision needed in landing by leveraging the existing circle detection code to maintain a target while descending, and by guiding the drone's feet into the intended position upon landing, even if misaligned.

These systems as demonstrated in CR2 and shown in the presentation confirm that the egg handling requirements have been met.

7.2 Shortcomings

Despite the flight systems being operational, the drone in its delivered state is computationally limited. Actively optimising the resolution meant that by running line and circle detection concurrently, the occasional low frame rate issues can lead to the drone drifting off course between refresh cycles. Computational hardware upgrades are not possible due to the system architecture and no further meaningful optimisation can be performed. This means that by integrating the flight and egg handling systems, they would individually cease to function. As such the drone's implied requirement of delivering the egg to the final circle has not been met.

The strict optimisation of the MV algorithms has led to a degree of inflexibility where for instance the drone cannot cope with turns greater than 90 degrees. A similar inflexibility can be seen in the grabber. While capable for the client's requirements, if the drone is intended to be used in a more general-purpose role, the implemented grabber and potentially the platform pocket will require modification.

While extensive testing was performed through simulation, the simulated environment did not have the functionality to add the grabber, egg, or world collisions. This meant that full systems integration was dependent on

physical testing. This relegated systems integration to 80 minutes per week in CR2, coupled with the difficulty of this task due to hardware limitations, it meant that creating the cohesive package that was initially intended was not possible.

These shortcomings point to future developments for the project centering around hardware upgrades. These are identified as 2 primary and 1 supplementary method.

7.3 Further Developments

A primary method would include a host computer that could be utilised to process the drone's camera and sensor data via Bluetooth connection. This would offload the necessary computation meaning the drone's processing power could be devoted to the actuation of its control mechanisms. This would allow a near real-time response to track features, assuming a powerful computer and stable connection.

To improve the adaptability of the drone system, a refined platform would supplement the drone though the use of electromagnets controlled by ultrasound or LiDAR sensors. This would enable the drone to be pulled into the base from a landing state, further increasing its tolerance to landing misalignment. This also widens the scope for drone payload geometries as potentially the pocket could be neglected, with the payload being held in place via an active mechanism.

Finally, combining both performance and adaptability, the drone itself could be redesigned using an open-source methodology. This would allow the use of powerful, and importantly, upgradable onboard computational hardware, in the proposed design utilising a Raspberry Pi. This addresses the primary limitation of the project drone and ensures that if in future the requirements outstrip the hardware, that the limiting component can be replaced. Similarly, if the task at hand calls for a supplementary component like an additional sensor, a forward-facing camera, or a custom grabber, the drone as a platform is capable of readily incorporating these features.

The proposed open-source design ensures that in conjunction with the developed hardware and software, the client's requirements for an autonomous egg transportation drone have been met.

Bibliography

[1] Parrot, "Parrot mambo documentation," 2020. [Online]. Available: https://support.parrot.com/uk/support/products/parrot-mambo (visited on 11/24/2020).

[2] Mathworks, "Model-based design with simulink," 2020. [Online]. Available: https://uk.mathworks.com/help/simulink/gs/model-based-design.html (visited on 10/15/2020).

[3] K. Forsberg and H. Mooz, "The relationship of system engineering to the project cycle," vol. 1, no. 1, pp. 57–65, 1991.

[4] H. A. Elsalamony, "Detecting distorted and benign blood cells using the hough transform based on neural networks and decision trees," pp. 457–473, 2015.

[5] Mathworks, "Getting started with image processing algorithms for parrot minidrones," 2020. [Online]. Available: https://uk.mathworks.com/help/supportpkg/parrot/ref/getting-started-with-parrot-minidrone-vision.html (visited on 10/01/2020).

[6] S. M. Sam and T. S. Angel, "Performance optimization of pid controllers using fuzzy logic," in *2017 IEEE International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM)*, 2017, pp. 438–442. DOI: 10.1109/ICSTM.2017.8089200.

[7] A. Talaeizadeh, E. Najafi, H. N. Pishkenari, and A. Alasty, "Deployment of model-based design approach for a mini-quadcopter," in *2019 7th International Conference on Robotics and Mechatronics (ICRoM)*, IEEE, 2019, pp. 291–296.

[8] S. Yoon, P. V. Diaz, D. D. Boyd Jr, W. M. Chan, and C. R. Theodore, "Computational aerodynamic modeling of small quadcopter vehicles," in *American Helicopter Society (AHS) 73rd Annual Forum Fort Worth, Texas*, 2017.

[9] Banggood, "2pcs uno r3 atmega328p development board no cable geekcreit for arduino - products that work with official arduino boards," [Online]. Available: https://uk.banggood.com/2Pcs-UNO-R3-ATmega328P-Development-Board-No-Cable-p-1427596.html?utm_source=googleshopping&utm_medium=cpc_organic&gmcCountry=GB&utm_content=minha&utm_campaign=minha-gbg-en-pc&currency=GBP&createTmp=1&utm_source=googleshopping&utm_medium=cpc_bgs&utm_content=lijing&utm_campaign=ssc-gbg-all-newsutom-0928&ad_id=468370263002&gclid=Cj0KCQiAk53-BRD0ARIsAJuNhpvKTyJwq2MllKC0iDGA-P5jjXdVVsJ6KriMyGRwZvyaGJSct0UL93AaAghOEALw_wcB&cur_warehouse=CN (visited on 12/01/2020).

[10] D. Young, "Hough transform for circles," 2016. [Online]. Available: https://uk.mathworks.com/matlabcentral/%20fileexchange/26978-hough-transform-for-circles (visited on 10/07/2020).

[11] Mathworks, "Imfindcircles - find circles using circular hough transform," 2020. [Online]. Available: https://uk.mathworks.com/help/images/ref/imfindcircles.html (visited on 11/07/2020).

[12] ——, "Parrot minidrones support from simulink," 2020. [Online]. Available: https://uk.mathworks.com/hardware-support/parrot-minidrones.html (visited on 10/02/2020).

[13] R. Barták, A. Hrasko, and D. Obdrzalek, "On autonomous landing of ar. drone: Hands-on experience," in *The Twenty-Seventh International Flairs Conference*, 2014.

[14] MatWeb, "Overview of materials for thermoset polyurethane foam, unreinforced," [Online]. Available: http://www.matweb.com/search/DataSheet.aspx?MatGUID=91d44cae736e4b36bcba94720654eeae&ckck=1 (visited on 12/01/2020).

[15] R. W. Deters, O. D. Dantsker, S. Kleinke, N. Norman, and M. Selig, "Static performance results of propellers used on nano, micro, and mini quadrotors," in *2018 Applied Aerodynamics Conference*, 2018, p. 4122.

[16] First4Magnets, "Grades of ferrite magnets," [Online]. Available: https://www.first4magnets.com/tech-centre-i61/information-and-articles-i70/ferrite-magnet-information-i83/grades-of-ferrite-magnets-i106 (visited on 12/01/2020).

[17] CoolComponents, "5v electromagnet - 5 kg holding force (p25/20)," [Online]. Available: https://coolcomponents.co.uk/products/5v-electromagnet-5-kg-holding-force-p25-20?variant=13205204631613&utm_medium=cpc&utm_source=google&utm_campaign=Google%5C%20Shopping&gclid (visited on 12/01/2020).

[18] Flytron, "Hc-sr04 ultrasonic sensor," [Online]. Available: https://store.flytron.com/products/hc-sr04-ultrasonic-sensor%5C?variant=32071510851715%5C&currency=GBP&utm_medium=product_sync&utm_source=google%5C&utm_content=sag_organic%5C&utm_campaign=sag_organic&utm_campaign=gs-2020-09-07&utm_source=google&utm_medium=smart_campaign (visited on 12/01/2020).

[19] Farnell, "161 photo cell," [Online]. Available: https://uk.farnell.com/adafruit-industries/161/accessory-type-photo-cell/dp/2419162?gclid=Cj0KCQiAk53-BRD0ARIsAJuNhpv4-3f3mgaetVVtTppqSXmT91oZABMuV-GjiTptXWZ4NAjiDIbXMowaAnYdEALw_wcB&gross_price=true&mckv=s_dc%7Cpcrid%7C483687683215%7Ckword%7C%7Cmatch%7C%7Cplid%7C%7Cslid%7C%7Cproduct%7C2419162%7Cpgrid%7C112844087463%7Cptaid%7Caud-899162869380:pla-300681412013%7C (visited on 12/01/2020).

[20] S. Beninger, "The disruptive potential of drones," *Marketing Letters*, vol. 31, pp. 315–319, 2020.

[21] U. C. A. Authority, "The drone and model aircraft code," 2019. [Online]. Available: https://register-drones.caa.co.uk/drone-code/protecting-peoples-privacy (visited on 11/25/2020).

[22] I. C. Office, "In the picture: A data protection code of practice for surveillance cameras and personal information," 2017. [Online]. Available: https://ico.org.uk/media/for-organisations/documents/1542/cctv-code-of-practice.pdf (visited on 11/25/2020).

[23] BBC, "Gatwick airport: Drones ground flights," 2018. [Online]. Available: https://www.bbc.co.uk/news/uk-england-sussex-46623754 (visited on 11/20/2020).

[24] ——, "Gatwick-bound plane forced to avoid 'high risk' drone," 2019. [Online]. Available: https://www.bbc.co.uk/news/uk-england-sussex-49494817 (visited on 11/20/2020).

[25] ——, "Heathrow drone protest: Airport says plans 'criminal and counterproductive'," 2019. [Online]. Available: https://www.bbc.co.uk/news/uk-england-london-49509852 (visited on 11/20/2020).

[26] G. Lykou, "Defending airports from uas: A survey on cyber-attacks and counter-drone sensing technologies," *Senors*, vol. 20, 2020.

[27] U. C. A. Authority, "The drone code," 2020. [Online]. Available: https://dronesafe.uk/wp-content/uploads/2019/11/Drone-Code_October2019.pdf (visited on 10/25/2020).

[28] ——, "Registering yourself to use a drone or model aircraft," 2020. [Online]. Available: https://register-drones.caa.co.uk/drone-code/protecting-peoples-privacy (visited on 10/25/2020).

[29] C. Weller, "Drones could replace $127 billion worth of human labor," 2016. [Online]. Available: https://www.businessinsider.com/drones-could-replace-127-billion-of-human-labor-2016-5?r=US&IR=T#:~:text=A%5C%20new%5C%20report%5C%20from%5C%20PwC,and%5C%20services%5C%20across%5C%20several%5C%20industries.&text=Drones%5C%20are%5C%20a%5C%20cheap%5C%2C%5C%20versatile,a%5C%20solid%5C%20cost%5C%2Dcutting%5C% (visited on 10/25/2020).

[30] A. Welch, "A cost-benefit analysis of amazon prime air," *University of Tennessee at Chattanooga*, 2015.

[31] A. Yang, "The freedom dividend, defined," 2020. [Online]. Available: https://www.yang2020.com/what-is-freedom-dividend-faq/ (visited on 10/25/2020).

[32] PWC, "How will automation impact jobs?," 2015. [Online]. Available: https://www.pwc.co.uk/services/economics/insights/the-impact-of-automation-on-jobs.htmL (visited on 10/21/2020).

[33] OSSWatch, "Open source software development," 2020. [Online]. Available: http://oss-watch.ac.uk/resources/softwaredevelopment (visited on 10/15/2020).

[34] A. Gibb, "Building open source hardware: Diy manufacturing for hackers and makers," 2015.

[35] R. Wilson, "Open source development - an introduction to ownership and licensing issues," 2013. (visited on 10/15/2020).

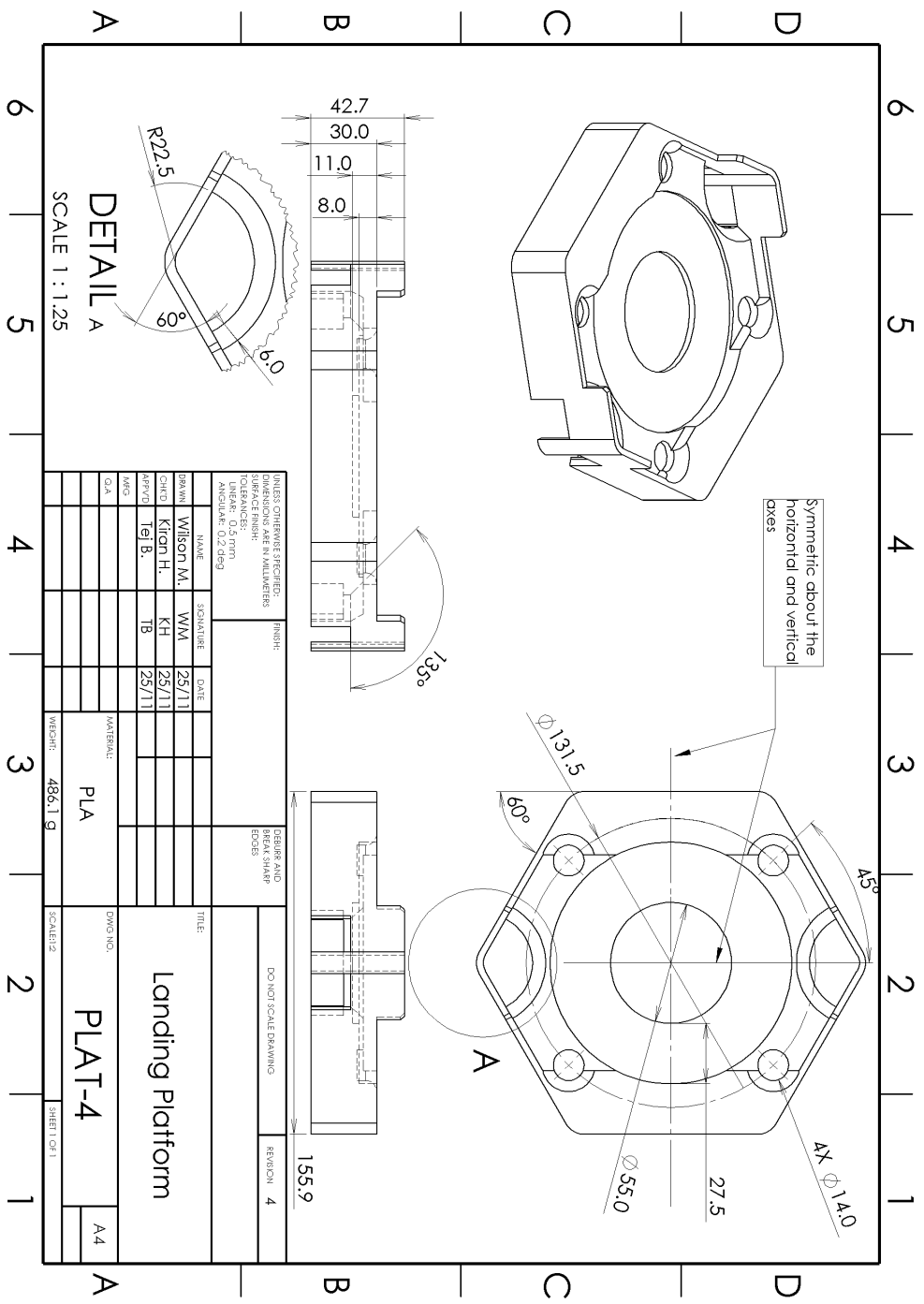
[36] H. Collins, "Is open source software more secure than proprietary products?," 2009. [Online]. Available: https://www.govtech.com/security/Is-Open-Source-Software-More-Secure.html (visited on 10/10/2020).

A | Original Hough Line Detection Code

Listing A.1: hough_line.m

```
% The purpose of this code is to detect lines in an image, and convert said lines into dx and ...
    % dy errors, which dictate the drone's movement.
%
% Written on 6/10/2020 by Benjamin Young, Matilda Bruce
% Last change:  20/10/2020

%The input u is the pre-processed image from the drone's camera feed
function [dx,dy] = hough_line(u)

BW = edge(u,'canny');

se = strel('line',3,100);
clean_u = imdilate(BW,se);

%hough transform performed on processed image
[H,theta,rho] = hough(clean_u);
peaks  = houghpeaks(H,2);

%lines extracted from hough transform
lines = houghlines(Ibw,theta,rho,peaks);

hold on

%convert lines to start and end coordinate points
for k = 1:numel(lines)
    y1 = lines(k).point1(1);
    x1 = lines(k).point1(2);
    y2 = lines(k).point2(1);
    x2 = lines(k).point2(2);
end

L=0.4*60;
y=0.4*80;
x=0.4*60;

angle = lines(1).theta;

%calculating dy and dx values from the angle of the first line.
dy=0.005*(y+(L*cosd(angle)));
dx=0.005*(x-(L*sind(angle)));
end
```

B | Technical drawing of landing platform



C | Precision landing mechanism: Arduino code and components used

Listing C.1: Control module.ino

```
/*
This code engages the electromagnet when the ultrasound sensor detect ...
    a flying drone above the platform.
The electromagnets get disabled once the drone successfully lands ...
    using a photoresistor so that the drone can lift-off again.

Written on 6/11/2020 by Vojtech Pavlis
Last change:  30/11/2020
*/
// Define pins for all sensors
    int const trigPin = 10; // digital pin (ultrasound transmitter)
    int const echoPin = 9; // digital pin (ultrasound receiver)
    int const ElectrPin = 12; // digital pin (electromagnets)
    int const PhotoResistor = A0; // analogue reading (photoresistor)

void setup(){
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT); // trig pin will have pulses output
  pinMode(echoPin, INPUT); // echo pin should be input to get pulse width
  pinMode(ElectrPin, OUTPUT); // ElectrPin controls the electromagnets
  pinMode(PhotoResistor, INPUT); // PhotoResistor input to get light ...
      changes
}

void loop()
  {
  // Duration will be the input pulse width and distance will be the ...
      distance   to the obstacle in centimeters
  int duration, distance, value, current;
  // Output pulse with 1ms width on trigPin
  digitalWrite(trigPin, HIGH);
  delay(1);
  digitalWrite(trigPin, LOW);
  // Measure the pulse input in echo pin
  duration = pulseIn(echoPin, HIGH);
  // Distance is half the duration devided by 29.1 (from datasheet)
  distance = (duration/2) / 29.1;

  // if distance less than 0.5 meter and more than 0
  if (distance <= 50 && distance >= 0) {
    // Current is sent through pin 2 to actuate the electromagnets
    digitalWrite(ElectrPin, HIGH);
    delay(50);
```

```
40      current = 1; // set to 1 to indicate there is current through pin 2
41      }
42
43    // if there is current through pin 2 then the photoresistor starts ...
          reading
44    if (current == 1) {
45     int analogValue = analogRead(A0);
46     Serial.print("Analog reading = "); // analog readings are printed
47     Serial.println(analogValue);
48
49     // if the value fet under 500 (gets darker) then switch off the ...
          electromagnets after a little delay
50     if (analogValue < 500) {
51       delay(2000); // delay set to 2 sec
52       digitalWrite(ElectrPin, LOW);
53       current=0;
54       delay(15000); // 15 sec delay giving the drone enough time to leave
55
56          }
57        }
58      }
```

Key electrical components used

Figure C.1: Electrical components: Arduino UNO (left) and 5V electromagnet (right)

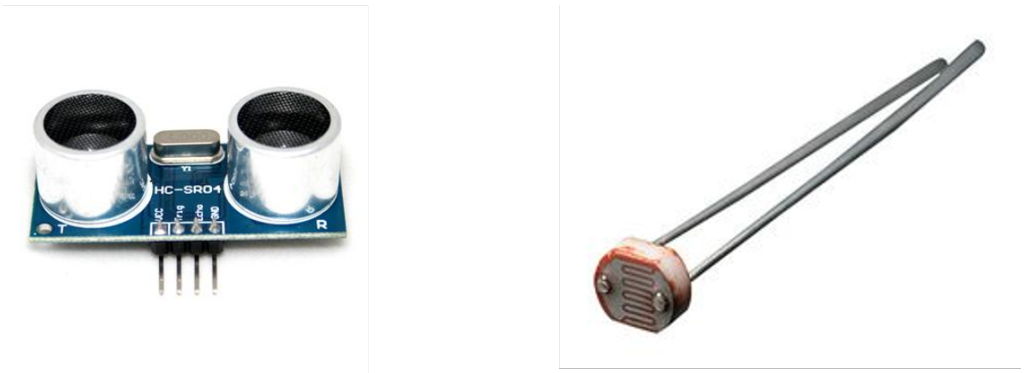


Figure C.2: Electrical components: Ultrasonic sensor (left) and photoresistor (right)

D | Possible code improvements

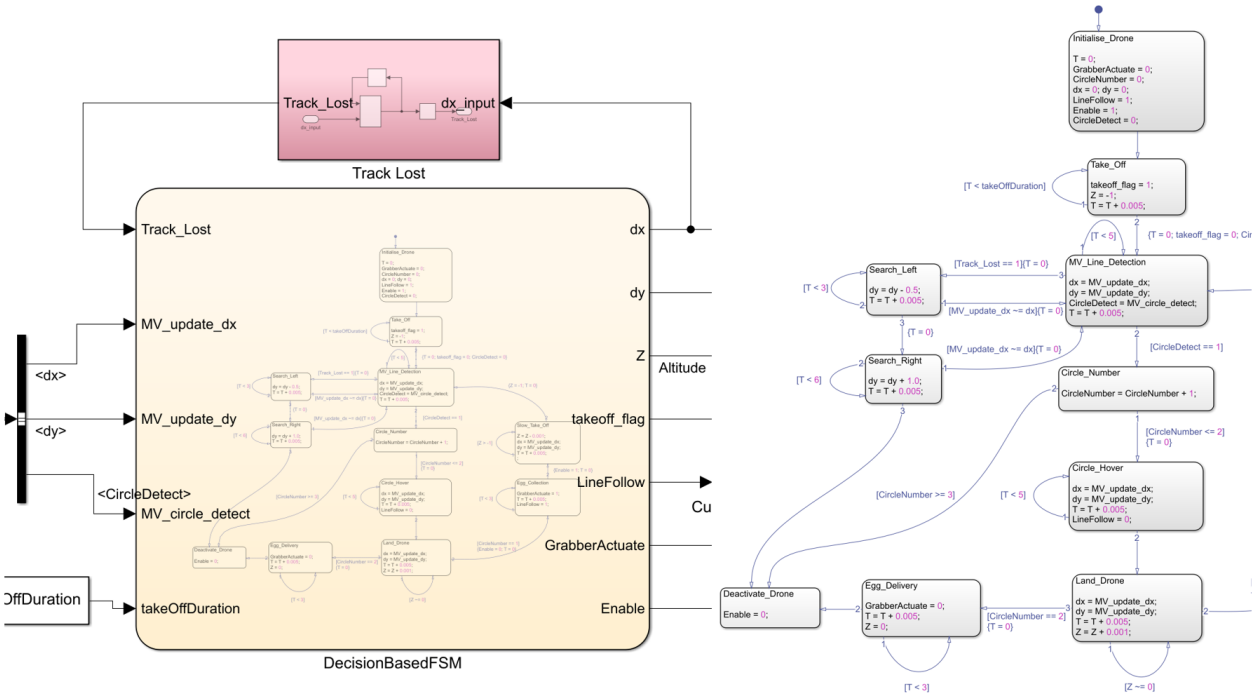


Figure D.1: Search and Find Code Inclusions

PI-PD tuning

In the following context, the reasoning for implementing the PI-PD controller will be reviewed and validated by the use of real-world experimental results provided by the Istanbul Technical University [37]. The study shows the derivation of equations of motion for a linear model based on the drone's non-linear model with the linear model input. The methods of linearization using Taylor series approximation were used which resulted in the following transfer function:

$$G_z(s) = \frac{z(s)}{F_B^{thrust}(s)} = \frac{25}{s^2} \tag{D.1}$$

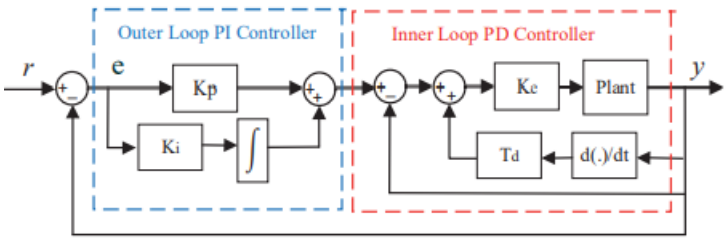


Figure D.2: A controller structure for PI-PD

From experimental results that the team conducted, the initial PID set up was giving a constant steady state error for amplitude. The research implemented outer loop PI controller on the feedback path to tackle that error. To determine values for K_c and T_d, the pole placement design method is used to achieve design criteria for both the overshoot value (OS) and settling time.

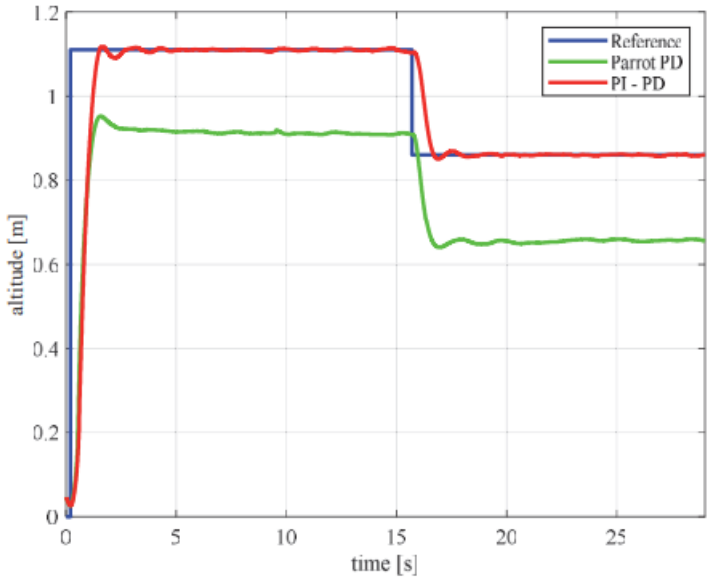


Figure D.3: Experimental results for the altitude reference trajectory

E | Failure mode and effects analysis

Item	The system or component that is affected
Failure mode	The method by which the component fails
Cause	The root cause of the failure
Effects	What happens to the drone as a consequence of the failure
Risk	Calculated automatically from Severity, Occurrence
Comment	Once a failure has been detected, what is the immediate reaction of a drone's operator to mitigate the risk What precautions do you take whilst doing this?

Table E.1: FMEA definitions of column headers

Contact: Kiran Hosein (s1795175@sms.ed.ac.uk)

Item	Failure mode	Cause	Effects	Severity	Occurrence	Risk	Comment
IMU	Anomalous or no data flow.	Accelerometer, gyroscope is faulty or damaged.	Uncontrolled motion could result in a crush. Drone does not react plausible to control input.	3	1	3	A safety flag will be triggered. Deactivate drone wirelessly.
Ultrasonic sensor	Sensor is faulty. Anomalous data.	Sensor is defected or damaged.	Uncontrolled motion could result in a crush. Drone does not react plausible to control input.	3	3	6	A safety flag will be triggered. Deactivate drone wirelessly.
Pressure sensor	Sensor is faulty. Anomalous data.	Sensor is defected or damaged.	Uncontrolled motion bellow 0.5m, which could result in a crush. Drone does not react plausible to control input.	2	2	4	A safety flag will be triggered. Deactivate drone wirelessly.
Camera	Anomalous or no data flow.	Camera is faulty or damaged.	Drone will take off and hover. Drone does not react plausible to control input.	1	1	1	Drone will deactivates as soon as the battery will be discharged. Deactivate drone wirelessly.
Grabber	Grabber is damaged.	Grabber is damaged due to previous crashes.	A grabber or its part can fall apart.	1	1	1	Check the grabber before attaching to the drone. If it has cracks contact Parrot Mambo support team for a replacing part.
Grabber controller	Grabber does not react plausible to control input.	Wiring is damaged. Controller is not securely attached to the drone.	Green led is not on. Grabber remains deactivated.	1	1	1	Check wiring. Deactivate drone wirelessly. Secure the controller to the drone before activating.
Blade	Unexpected behaviour.	Blade is not attached to the motor's shaft. Blade is defected.	Uncontrolled motion could result in a crush. Blade can fly off.	5	3	15	Secure blades before activating. A safety flag will be triggered. Deactivate drone wirelessly.
Accumulator	Cell voltage or current is below operating range. Drone's accumulator connector is damaged.	Faulty battery. Battery is discharged.	Drone is not activated.	1	2	2	Drone will remain deactivated.

Table E.2: Failure mode and effects analysis

Rating	Severity	Occurrence
1	No injuries may be caused, but general safety is affected by this failure	Failure occurrence is very unlikely
2	Light injuries may be caused by this failure	Relatively few failure occurrence
3	Medium injuries may be caused by this failure	Occasional failure occurrence
4	Heavy injuries may be caused by this failure	Frequent failure occurrence
5	Fatal injuries may be caused by this failure	Persistent failure occurrence

Table E.3: Key for Severity, Occurrence and Detection ratings