

**DEAKIN University**

**Symbiosis Project – 2018**

Project Handover (Review)

**Team #26**

Tom Guilfoyle - tguilfoy@deakin.edu.au

Paul Hammond - phammon@deakin.edu.au

Sherry He - hezixu@deakin.edu.au

Ricky Lam - waik@deakin.edu.au

Kevin Li - rui@deakin.edu.au

Joshua Pujol - jpujol@deakin.edu.au

Justin Taylor - jktaylor@deakin.edu.au

Table of Contents

[Client 1](#_Toc520075805)

[Project Supervisors 1](#_Toc520075806)

[Project Liaison 1](#_Toc520075807)

[Project Team 1](#_Toc520075808)

[Document Purpose 1](#_Toc520075809)

[Project Description 1](#_Toc520075810)

[Timing Chart 2](#_Toc520075811)

[Planned Work 1](#_Toc520075812)

[Findings 1](#_Toc520075813)

[Justification 1](#_Toc520075814)

[Actions 1](#_Toc520075815)

[Prototype Flight-Control Solution – Record of Changes 5](#_Toc520075816)

[Overview 5](#_Toc520075817)

[High-level Architecture – Prototype Solution 6](#_Toc520075818)

[User Manual – Flight-control Software Prototype 7](#_Toc520075819)

[Downloading the Flight-Control Prototype Code 8](#_Toc520075820)

[Downloading the GUI-Application 9](#_Toc520075821)

[Running the Flight-Control Prototype Application 9](#_Toc520075822)

[Source Code 13](#_Toc520075823)

[Team Structure Review 14](#_Toc520075824)

[Findings 14](#_Toc520075825)

[Justification 14](#_Toc520075826)

[Actions 14](#_Toc520075827)

[Trello Board Updates 14](#_Toc520075828)

[Interim Sprint 14](#_Toc520075829)

[Sprint 3 15](#_Toc520075830)

[Open Issues Review 16](#_Toc520075831)

[Time 16](#_Toc520075832)

[Findings 16](#_Toc520075833)

[Justification 17](#_Toc520075834)

[Actions 17](#_Toc520075835)

[Resource Management 17](#_Toc520075836)

[Occupational Health and Safety 17](#_Toc520075837)

[Findings 17](#_Toc520075838)

[Justification 18](#_Toc520075839)

[Action 18](#_Toc520075840)

[Drone Fragility / Damage 18](#_Toc520075841)

[Findings 18](#_Toc520075842)

[Justification 18](#_Toc520075843)

[Actions 18](#_Toc520075844)

[Drone Battery Life 19](#_Toc520075845)

[Findings 19](#_Toc520075846)

[Justification 19](#_Toc520075847)

[Actions 19](#_Toc520075848)

[Synchronicity 19](#_Toc520075849)

[Findings 19](#_Toc520075850)

[Justification 19](#_Toc520075851)

[Action 20](#_Toc520075852)

# Client

Deakin Motion Capture Lab (DML), Building B1.81, Deakin University, Burwood

* Professor Stefan Greuter - [stefan.greuter@deakin.edu.au](mailto:stefan.greuter@deakin.edu.au)
* Dr Jordan Vincent - [jordan.vincent@deakin.edu.au](mailto:jordan.vincent@deakin.edu.au)

# Project Supervisors

* Dr Greg Bowtell - [g.bowtell@deakin.edu.au](mailto:g.bowtell@deakin.edu.au)
* Dr Tim Wilkin - [tim.wilkin@deakin.edu.au](mailto:tim.wilkin@deakin.edu.au)

# Project Liaison

* Dr Jan Carlo Barca - [jan.barca@deakin.edu.au](mailto:jan.barca@deakin.edu.au)

# Project Team

|  |  |  |
| --- | --- | --- |
| **Student Id** | **Student Name** | **Role** |
| 216189236 | Joshua Pujol | Group 1 |
| 217087364 | Thomas Guilfoyle | Group 1 |
| 217197255 | Kevin Li | Group 2 |
| 215194586 | Ricky Lam | Group 2 |
| 213089336 | Justin Taylor | Group 1 |
| 217196992 | Sherry He | Group 2 |
| 216171484 | Paul Hammond | Group 1 |

# Document Purpose

This document is intended to provide a summary of changes relating to Symbiosis Team’s work on the CrazyFlie 2.0 Drone Flight-Control solution, as part of the review of the project handover document completed for SIT374 in trimester 1.

These changes have occurred since the submission date of the handover document.

# Project Description

Refer to handover documentation.

# Timing Chart

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date Starting** | **9-Jul** | **16-Jul** | **23-Jul** | **30-Jul** | **6-Aug** | **13-Aug** | **20-Aug** | **27-Aug** | **3-Sep** | **10-Sep** | **17-Sep** | **24-Sep** |
| **Week** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |
| **Interim Sprint** |  |  |  |  |  |  |  |  |  |  |  |  |
| Learn OptiTrack Calibration Process |  |  |  |  |  |  |  |  |  |  |  |  |
| Expand Prototype solution to incorporate C# / Unity Functionality |  |  |  |  |  |  |  |  |  |  |  |  |
| Demonstrate to Client @ DML |  |  |  |  |  |  |  |  |  |  |  |  |
| **Sprint 3** |  |  |  |  |  |  |  |  |  |  |  |  |
| Develop CrazyFlie / OptiTrack tracking functionality |  |  |  |  |  |  |  |  |  |  |  |  |
| **Sprint 4** |  |  |  |  |  |  |  |  |  |  |  |  |
| Debug and finalise software |  |  |  |  |  |  |  |  |  |  |  |  |
| Prepare Documentation |  |  |  |  |  |  |  |  |  |  |  |  |
| Application Delivery |  |  |  |  |  |  |  |  |  |  |  |  |

Figure 1: Proposed Timing for SIT302 – Trimester 2 (Updated 21/7/201

# Planned Work

Predominantly, changes to planned work relate only to the prototype solution. The following section describes those changes.

## Findings

After a successful demonstration of the prototype solution to the client in week one (12th July), it was decided by the team and our supervisor (Tim) that we should set a new project goal and continue to expand development on the prototype to incorporate the desired user input functionality directly from Unity (the prototype solution utilised a console-based interface for user input). This has been captured in an “Interim Sprint” prior to commencement of Sprint 3 (which will see continued development of flight control software to utilise the OptiTrack motion capture system for external tracking of the drone in flight).

## Justification

This change to planned work is intended to aid our team in progressing development of the final solution, which requires integration of the flight control system with the Unity software. By utilising the prototype (an already proven, and familiar, working solution), our team can completely focus on integrating user-input functionality in Unity (the unknown). This “divide and conquer” approach helps to break the larger problem into smaller, more manageable problems, resulting in less likelihood of errors and a better chance of a successful outcome. Once this development phase is completed, it can then be easily incorporated into the development of the final solution during the upcoming Sprints. The flow-on effect is also a more efficient allocation of team resources during those critical sprint phases.

## Actions

The prototype solution has been written in Python while Unity utilises the C# language. As such, development in week two has seen the creation of a C#-based application designed to act as a “bridge”, automatically (and directly) able to run the python-based prototype solution within the C# framework (Figure 2). The C# application utilises a GUI (Graphical User Interface) and can run natively in Windows, omitting the need for the Bitcraze Virtual Machine Environment or the user having to go through overly complicated installation and running steps, therefore vastly simplifying the operation of the prototype software solution for the end-user.

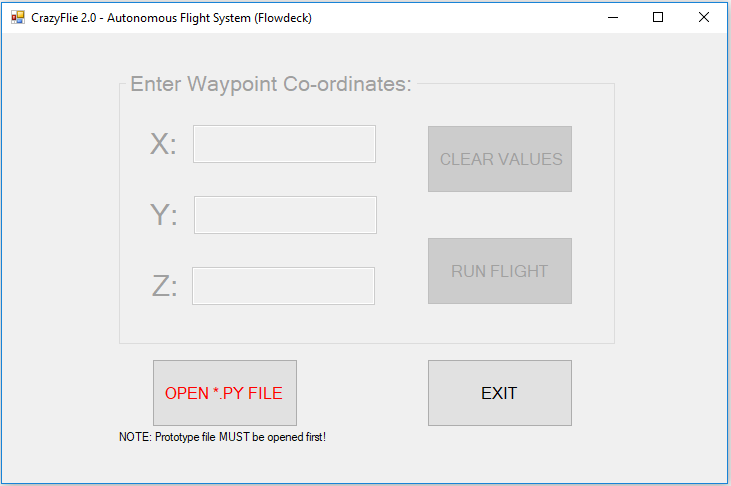


Figure 2: C#-based GUI application which runs the Python-based Prototype Flight-Control Solution

Additionally, user-input error checking functionality has been incorporated into the GUI to prevent incorrect or erroneous user inputs being mistakenly sent to the CrazyFlie drone, which could result in a drone crash during flight (Figure 3). The user must enter positive integer or decimal numbers for x, y, and z before the program will allow the flight control software to fly the drone.

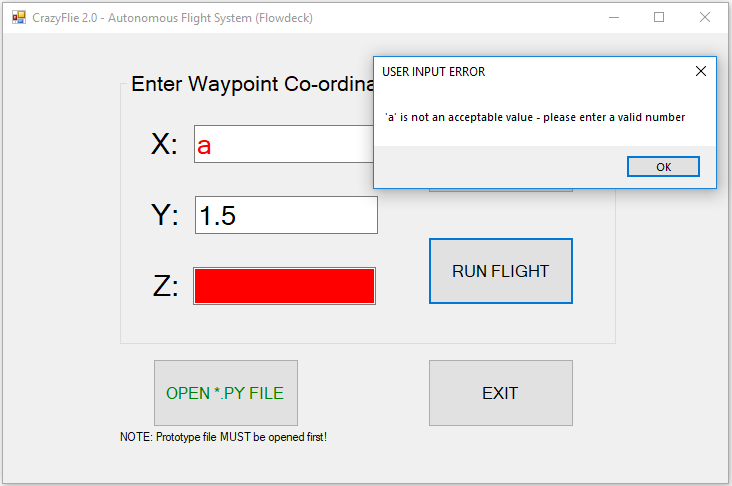


Figure 3: Built-in error checking to prevent incorrect user commands being sent to drone flight program.

To use the application, the user simply selects the Python file from its location on the user’s local machine (eg: C:\DroneFlight\proto\_v3.py) and inputs the desired co-ordinate for the drone to fly to. Clicking “RUN FLIGHT” will execute the prototype flight-control software (Figure 4).

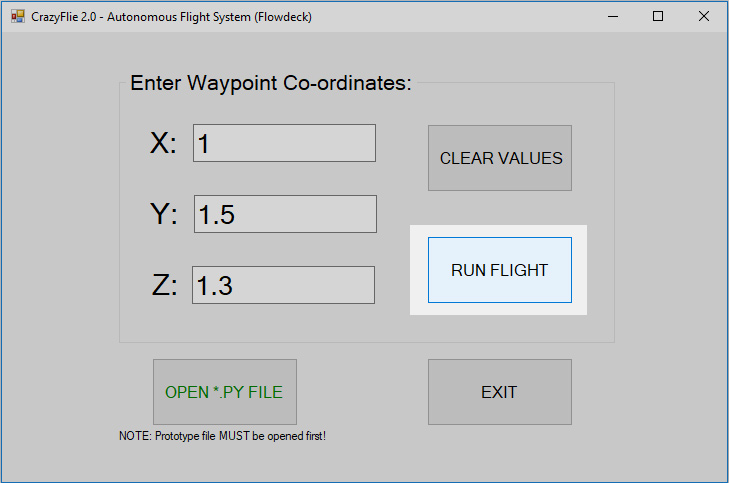


Figure 4: Once the prototype file is opened and desired waypoints entered, clicking “RUN FLIGHT” executes the Python-based Prototype Flight-Control Solution and flies the drone automatically.

After each successful flight, new co-ordinate values can be entered by the user and the drone flight re-initiated.

Having successfully completed this task, we have proven that it is possible to run the python code from a C# application. The next logical step is to see if we can adapt the C# application to work within Unity, allowing a user to input desired positional co-ordinates (x, y, z) into the Unity interface, which is then passed into the prototype software via the C# application / adapted C# script to fly the drone , in line with the client’s requirements (Figure 5).

Upon completion, the expanded prototype solution will be demonstrated to the client in the Deakin Motion-Capture Lab (DML) for review and feedback: anticipated to occur during Week 3.

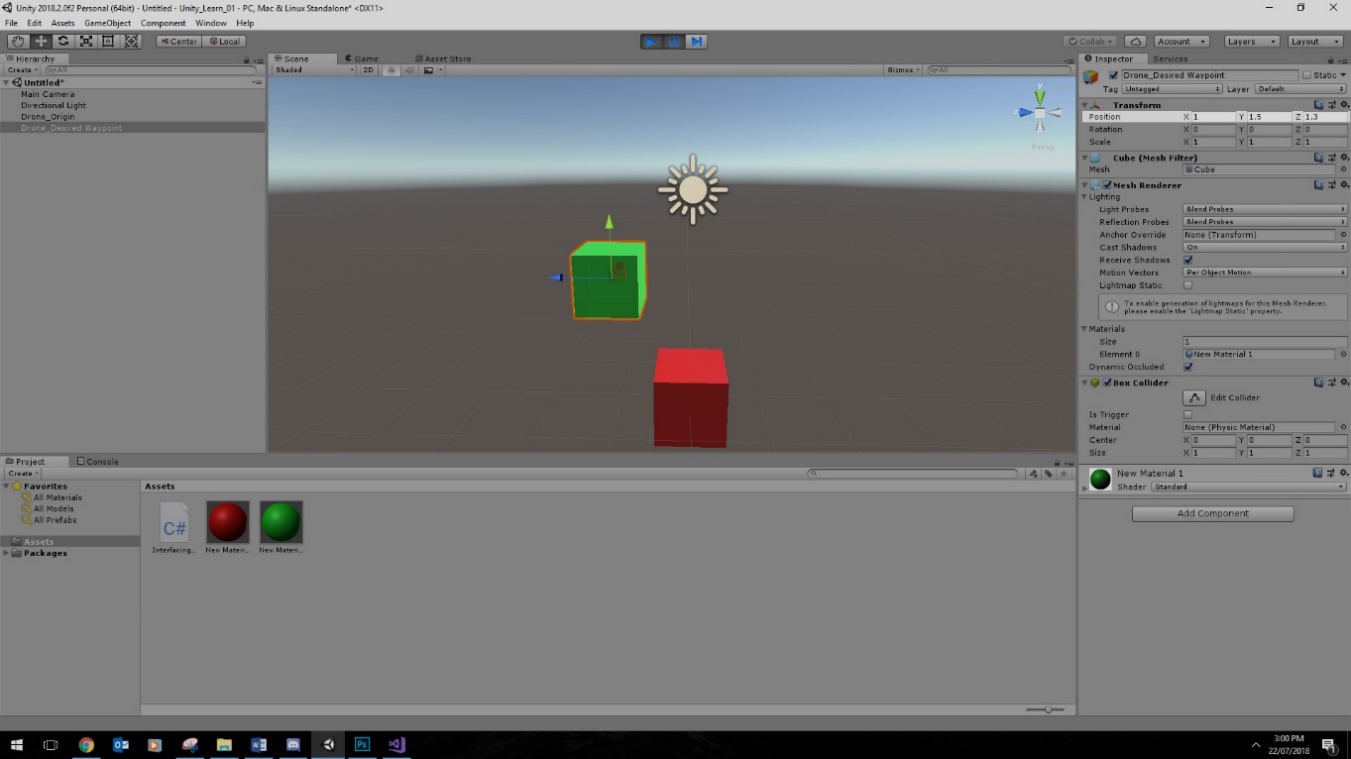


Figure 5: Unity interface: desired Position: x, y, z values inputted directly through the interface.

# Prototype Flight-Control Solution – Record of Changes

## Overview

The information provided in the following section describes changes to the prototype flight-control solution for automatic control of the CrazyFlie 2.0 drone, namely the implementation of a C#-GUI based application to run the Python code and the omission of the Bitcraze Virtual Machine environment. Information on the original prototype solution can be reviewed in the handover documentation.

The final solution for our client will integrate the DML's OptiTrack, Motive and Unity systems with the CrazyFlie Flight-Control software developed by our team, to enable the DML operators to set desired waypoints (positional 3-D co-ordinates) and automatically fly the drone to those waypoints within the DML environment, without the need for manual flight-control inputs.

## High-level Architecture – Prototype Solution



Figure 6: High-level Architecture: Prototype Flight-Control Solution

# 

# User Manual – Flight-control Software Prototype

Prior to running the Symbiosis Flight-Control software prototype, the following hardware and software must be installed.

**Hardware required**:

* Bitcraze CrazyFlie 2.0 drone (updated to latest firmware: 2018.01.1)
* Bitcraze CrazyRadio USB device (placed in a USB 2.0 port)
* Bitcraze Flowdeck (mounted to drone)

**Software required**:

* Python
  + <https://www.python.org/downloads/>
  + Ensure python.exe is added to Windows PATH (during installation)
* Zadig USB Driver Software (Windows-based USB drivers for CrazyRadio)
  + <https://zadig.akeo.ie/>
  + Version 2.3 installed
  + NOTE: select “libusb” driver
* GUI-based user application
  + User-selects flight-control prototype file
  + Takes user input values for x, y, z
  + Runs the flight-control prototype (python-source code)
  + For the GUI Application, please refer to Symbiosis GitHub repository:
* <https://github.com/SymbiosisTeam/Prototype-GUI-Executable>
* Python-based flight control source code (current version: proto\_v3.py)
* For the Prototype Flight Control Solution, please refer to Symbiosis GitHub repository:
  + <https://github.com/SymbiosisTeam/Prototype-Solution/blob/master/proto_v3.py>

## Downloading the Flight-Control Prototype Code

Connect to Team Symbiosis GitHub repository and download latest version of the Python-based prototype file and GUI-based application (Figure 7).

* Link: <https://github.com/SymbiosisTeam/Prototype-Solution>
* Recommended to download file to a root directory created off the local drive,

eg: C:\DroneApp

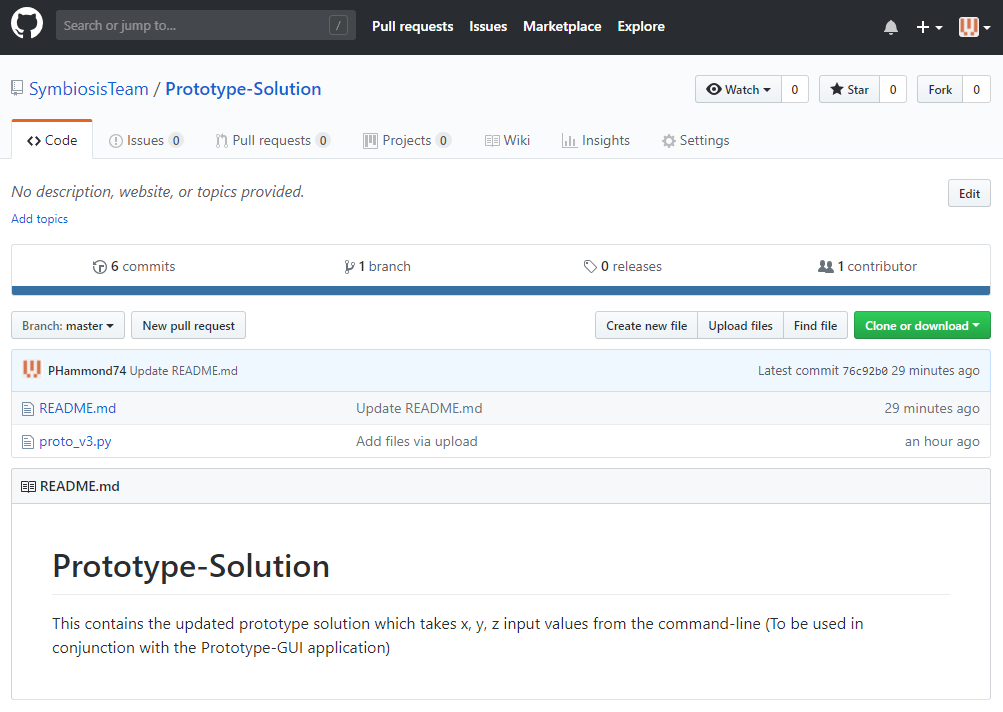


Figure 7: Team Symbiosis Github Repository for Prototype Source Code

## Downloading the GUI-Application

Connect to Team Symbiosis GitHub repository and download latest version of the GUI-based application (Figure 8).

* Link: <https://github.com/SymbiosisTeam/Prototype-GUI-Executable>
* Recommended to download both files to same directory, eg: C:\DroneApp

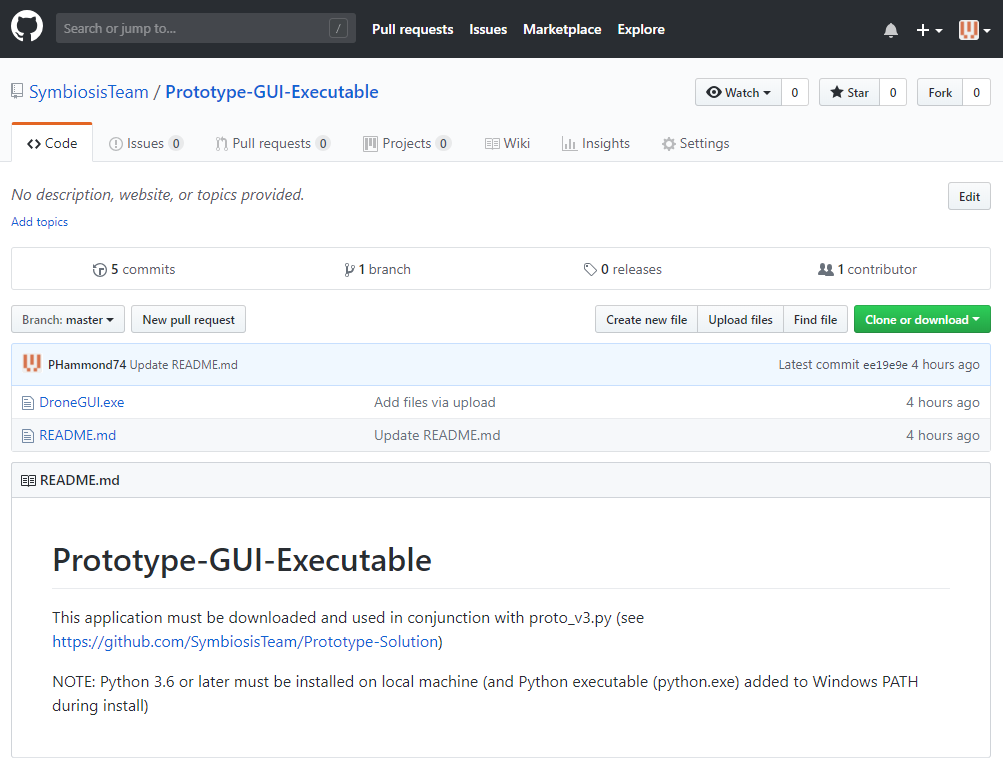


Figure 8: Team Symbiosis Github Repository for GUI-based Application

## Running the Flight-Control Prototype Application

NOTES:

* To avoid potential damage to drone and / or surrounding environment, ensure the drone is only flown in a sufficiently sized, obstacle-free space. User designated co-ordinate points selected must be accurate and exist within the flight zone environment.
* The prototype solution works relative to the drone’s location (origin) at the time the program is executed.
  + For this prototype solution to work accurately, the drone must be placed at a user-designated origin, and oriented to face intended forward flight direction (as shown in Figure 9).

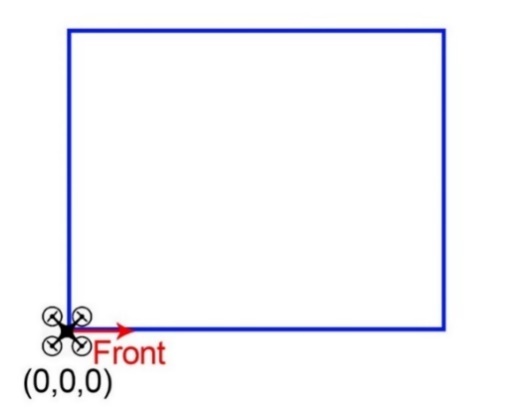


Figure 9: The Origin (0, 0, 0) location relative to the flight space envelope

Step 1: Plug CrazyRadio controller into a USB port

* + - NOTE: Do not use a USB 3.0 port is this may have compatibility issues affecting communication with drone.

Step 2: Place Drone on a flat surface and switch Drone on.

* + - NOTE: Drone must be placed on a flat surface prior to being switched on. Failure to do so may affect the flow-deck initialisation phase and alter flight characteristics, resulting in a crash. Once initialised, drone may be moved / placed in a different location without issue.

Step 3: Place Drone at desired starting location (origin) (refer to Figure 9).

* + - NOTE: Program is designed to fly drone to the user-designated waypoint in relation to Drone’s position at commencement of flight (origin)
      * x = forward, y = left, z = up

Step 4: Open GUI-Application (DroneGUI.exe) (refer previous section)

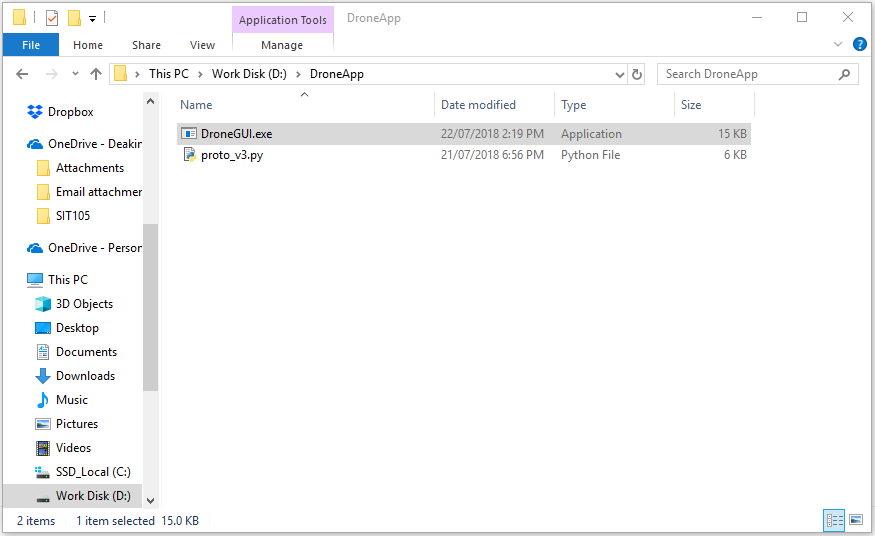
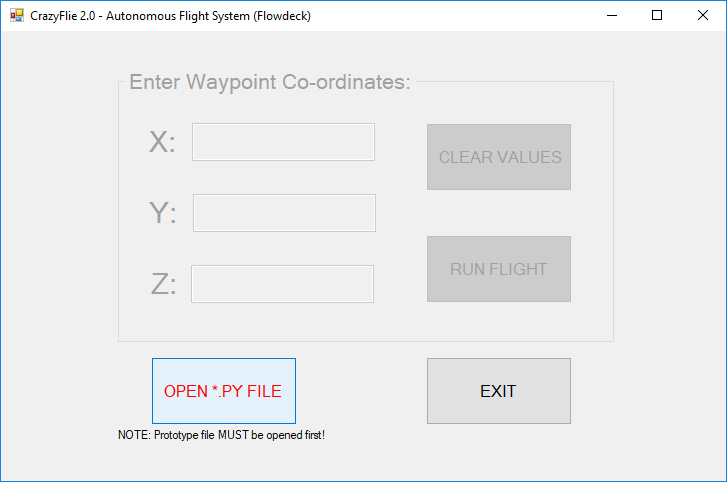


Figure 10: Open DroneGUI.exe

Step 5: Select OPEN \*.PY FILE

* + - * Use the open file dialog to navigate to the appropriate directory and open the prototype python file (python\_v\*.py)



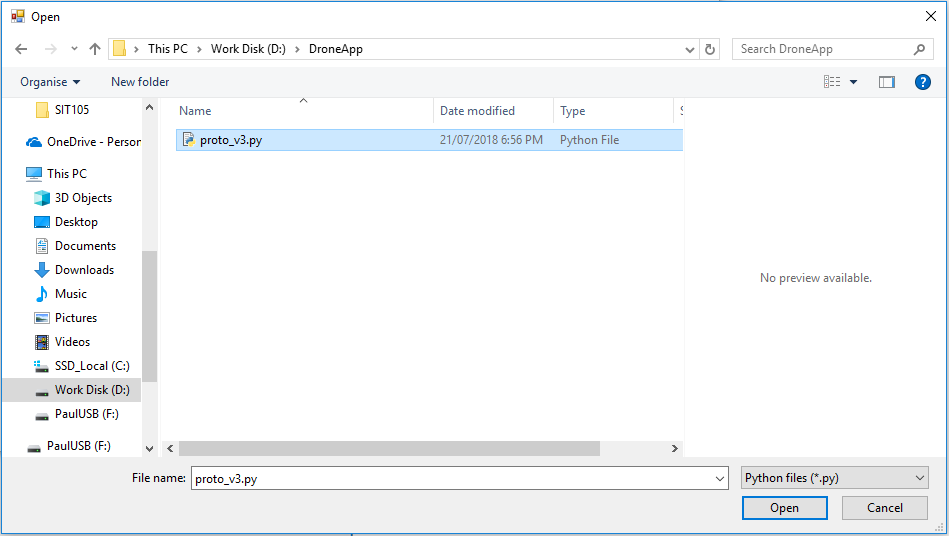


Figure 11: Opening proto\_v3.py

* + - * Once the appropriate python file is opened, the co-ordinate fields are unlocked and co-ordinate values can then be inputted by the user (Figure 12).

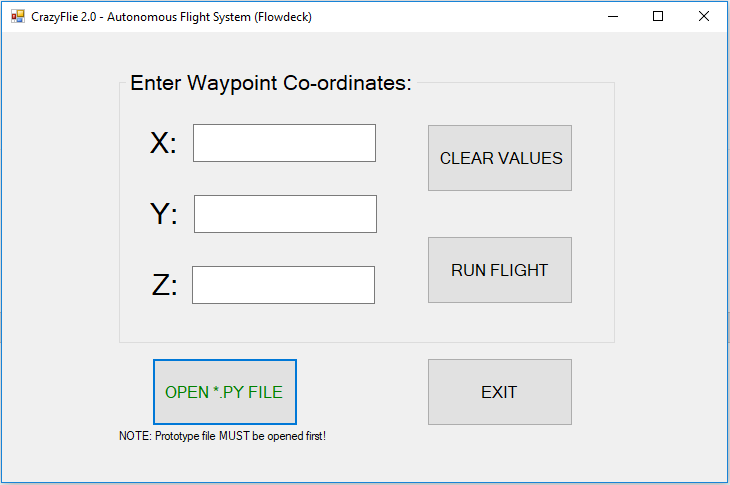


Figure 12: proto\_v3.py file is opened, unlocking access to co-ordinate fields

Step 6: Enter desired waypoint (x, y, z) co-ordinate in the respective fields

* + - NOTE: Values must be positive integers or decimals
    - Values can be cleared by clicking “CLEAR VALUES”

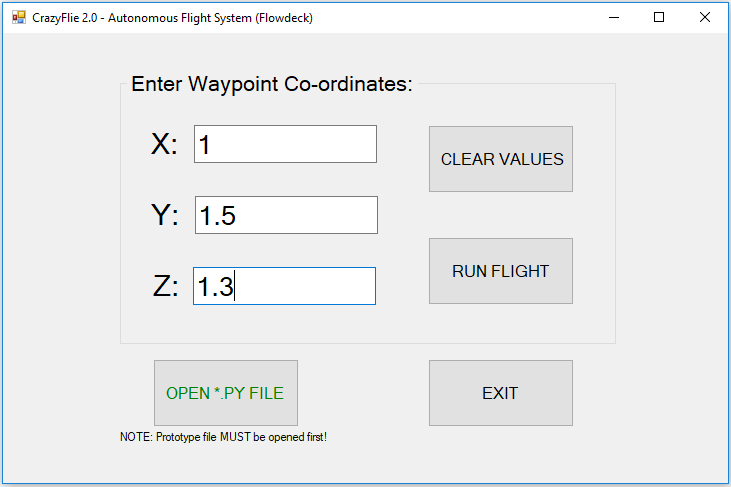


Figure 13: Enter co-ordinate values

Step 7: Click “RUN FLIGHT” button to fly drone

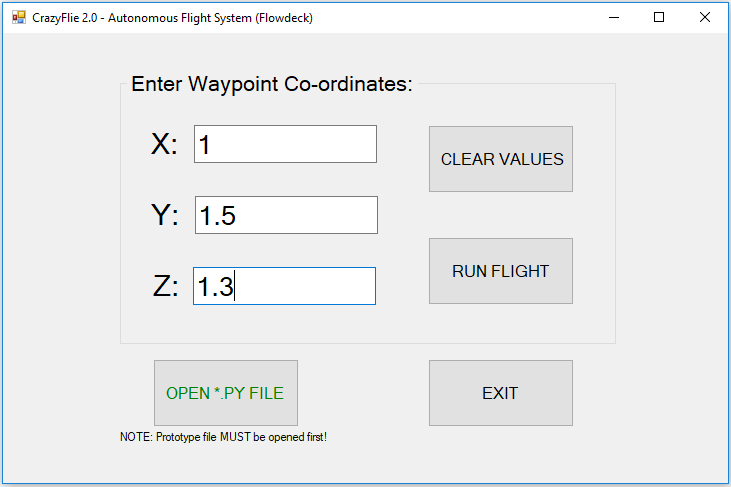


Figure 14: Click “RUN FLIGHT” to fly drone

Step 8: Additional flights can be run by repeating Steps 6 – 7.

Step 9: Exit Application by clicking “EXIT” and turn off drone.

## Source Code

For the Prototype Flight Control Solution, please refer to Symbiosis GitHub repository:

<https://github.com/SymbiosisTeam/Prototype-Solution>

For the GUI Application Executable File, please refer to Symbiosis GitHub repository:

<https://github.com/SymbiosisTeam/Prototype-GUI-Executable>

For the GUI Application Source Code, please refer to Symbiosis GitHub repository:

<https://github.com/SymbiosisTeam/Prototype-GUI>

# 

# Team Structure Review

## Findings

Following on from a successful completion of Sprints 1 and 2 in trimester 1, the need to maintain sub-groups (Group 1 and Group 2) within the team is no longer required.

## Justification

Sprint 3 will require the amalgamation of the team’s knowledge and experience of working with both the CrazyFlie Drone and the Unity platforms to begin developing the OptiTrack aspect of the flight-control solution. The interim sprint has seen our team bring forward development on the Unity functionality to work with our prototype flight control solution, which has drawn upon the work carried out by both sub-groups during Sprints 1 and 2.

## Actions

During the interim sprint period, our team held a team meeting where we discussed dissolving the sub-group structure and we restructured ourselves into a single team. As we begin planning for Sprint 3, it is expected that individual team members will take on responsibilities for some assigned key tasks, whilst other tasks may be carried out by multiple members or the whole team (where required).

# Trello Board Updates

## Interim Sprint

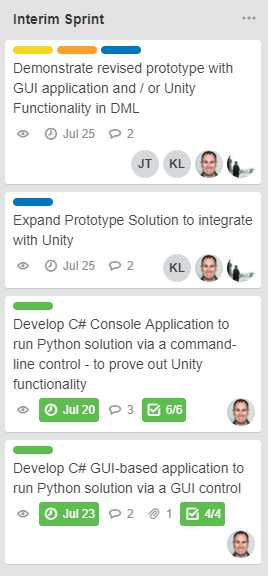


Figure 15: Interim Sprint Trello Board

## Sprint 3

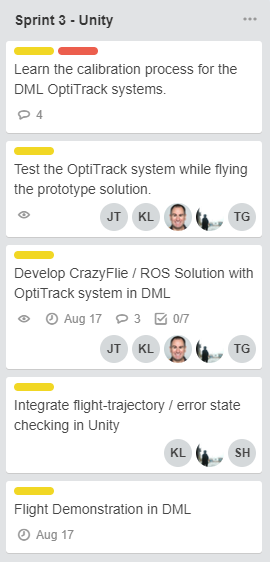


Figure 16: Sprint 3 Tasks

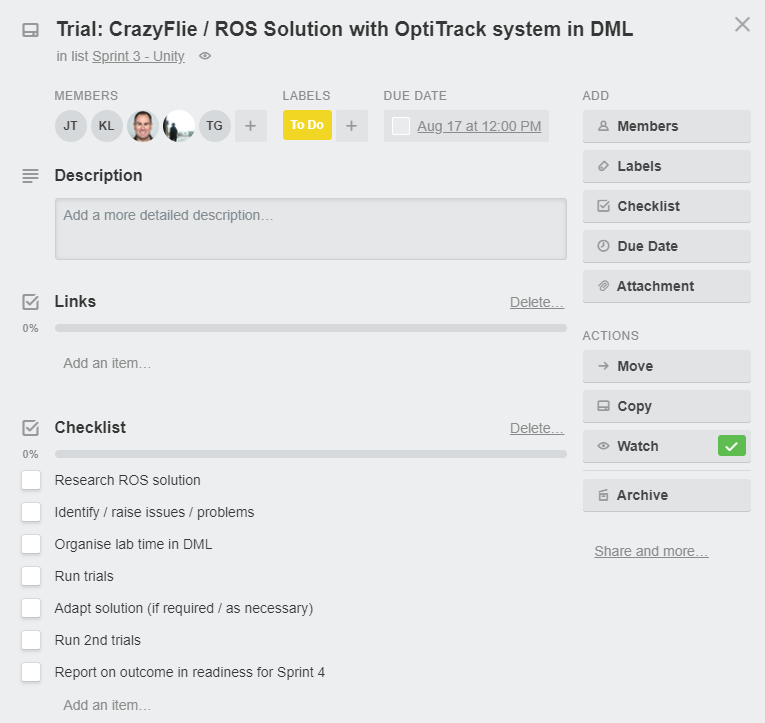


Figure 17: Sprint 3 – Primary Task Breakdown

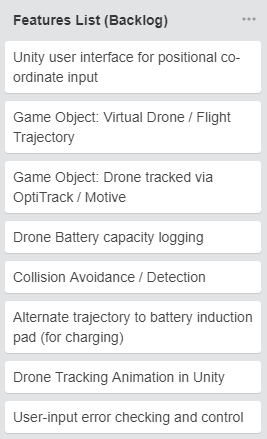


Figure 18: Proposed Features list expected to be implemented during Sprints 3 and 4

# Open Issues Review

## Time

### Findings

Although we have made successful progress with our Symbiosis project during the first trimester, as well as developing a more detailed, clear and practical plan, time is still the primary issue which our team recognises as having significant impact on the continued progress of our project during trimester 2.

Some team members are having to manage a full-time study load while others are currently unavailable (or have limited availability) due to being overseas. We all have additional responsibilities (such as work and family, and other study units) that may take priority at certain times and needs to be carefully managed.

Rather than wait for commencement of Sprint 3, our team took active steps to arrange a meeting with the client (to demonstrate the prototype solution) in the first week of the second trimester. After that successful outcome, we decided to establish objectives for an “Interim Sprint” over the first three weeks, working on further development of the prototype, whilst bringing forward planned work that was due to be carried out over the upcoming sprints. We have made significant progress with our project in that time. Our continued aim is to maintain this momentum and capitalise on as much available time as we can to ensure our project remains on schedule.

For Sprint 3, from week 4 to week 7, we are expecting to develop the external tracking capability of the OptiTrack system in the DML, integrating it with our improved flight-control system. Depending on the outcome of Sprint 3, we may then incorporate our step goals for the final sprint, such as the battery induction pad, object collision avoidance, and multiple drone cycling.

### Justification

Our team has reflected on our experiences working on the Symbiosis project during trimester 1. We learnt from issues which arose during that period, such as hardware delays (which significantly impacted on our development of the prototype solution), some inter-team communication problems, as well as the steep learning curve associated with the CrazyFlie platform and the Unity software.

In addition, focusing on the integration of flight-system and Unity during the “Interim Sprint” will also benefit our team by requiring less work during the later sprints (when we may also be facing other, unexpected issues). contributing to less required work during the later sprints lot to easier management and less bugs for our symbiosis team during final solution implementation.

Careful planning of the work to be done (and allocation of resources) for the upcoming third (and fourth) sprint is crucial if we are to remain on track.

### Actions

We have also made positive changes by taking the initiative and making an immediate start at the commencement of the second trimester.

Our team will continue to monitor our progress and aim to identify any potential roadblocks which could affect our progress through the third and fourth sprints. We will utilise tools such as timing charts, the Trello board, Discord, and meeting minutes to ensure any issues are identified and rectified as early as possible.

We will also endeavour to keep our communication effective and open, referring to and utilising the tools described above. Regular weekly meetings were effective and contributed significantly towards our trimester 1 success. We will continue to undertake weekly team meetings over trimester 2, and any additional meetings as required.

Each team member recognises that they need to show initiative and take full responsibility to ensure their work is completed correctly and on time, whilst always being prepared to offer additional support where needed for the rest of the team.

## Resource Management

## Occupational Health and Safety

### Findings

During Sprint 1 in SIT374, the client advised our team that they intended to develop a 3-D printed frame mountable to the drone to help improve convergence for the motion capture system as well as act as a protective guard for Occupational Health and Safety (OH&S) requirements. It was also anticipated that the guard would help make the drone more durable. The client has since advised that they will no longer pursue this goal.

### Justification

The client has determined that a guard not required for OH&S regulations, nor do they expect to have issues with convergence of the motion capture markers (based on trials in the DML). In addition, durability is no longer a major concern, with additional drones (and spare parts) on order, in conjunction with the successful performance of the autonomous flight-control system.

### Actions

No further action required.

## Drone Fragility / Damage

### Findings

During development of the prototype, some drone crashes have been experienced with occasional damage to drone parts (rotors) requiring replacement. This reinforced an issue raised during the project proposal that related to the, then, availability of a single drone. Whilst there are some spare parts included in the package, there is not an adequate supply readily available in the event of sever accidental damage or multiple incidents of damage. In addition, with potentially long hardware shipping delays due to fluctuating stock levels, there was a risk of project delay (and added costs), which had to be allowed for as part of the project development.

### Justification

Spare parts are required. Even if crashes do not occur, parts will still be subject to wear. Having enough parts available will avoid unnecessary project delays, albeit at increased project cost.

Autonomous control of drone flight eliminates the need to rely on manual flight control of drone, significantly reducing risk of accidents. With the prototype working successfully, it is expected that implementation of highly accurate, external tracking via the OptiTrack system, combined with fully autonomous drone control and user-input error checking, that risk of accidents will be almost non-existent.

### Actions

User-input error checking has been built into the prototype application to eliminate an incorrect input which could cause the drone to crash. This must be carried over into the Unity functionality and the final solution.

The client has ordered additional spare parts to cover accidents / crashes during flight trials. The client has also ordered additional drones, which could be utilised in place of a severely damaged or unrepairable drone.

Team to acquire a fishing net (as per good suggestion from Tim) as an additional safety measure to keep on hand during flight trials in case of unexpected deviation from flight trajectory. A falling drone caught in a net will suffer much less chance of damage than hitting a solid floor or other obstacle.

## Drone Battery Life

### Findings

During development of the prototype, through numerous flight trials, it has been determined that the CrazyFlie 2.0 drone has a significantly limited battery life of approximately 6 to 7 minutes, depending on usage, with a re-charge time of 30-plus minutes. This will certainly be a major issue for the client (and the final solution) if they continue to rely on a single drone. The team needs to collaborate with the client to identify and implement a suitable solution, such as cycling through multiple drones to cover the larger downtime period.

### Justification

Unless a more efficient battery and / or recharging system can be acquired, the most feasible solution is to utilise a continuous cycle of multiple drones. The alternative is that the client may only be able to use the drone in 6-minute spurts followed by 30-minute downtime while recharging, which would not be acceptable for their needs.

### Actions

As the client has confirmed they have ordered additional drones, the team has included the implementation of multiple drone-cycling functionality into a step goal, to be developed subject to time and resource availability and the successful outcome of implementing the OptiTrack system into the flight-control solution (the primary project goal).

Tim, our supervisor, has advised that he’s placed a request for Deakin’s electrical engineering department to develop a more efficient battery. This is not expected to have any impact on our team’s progress with the project, however.

## 

## Synchronicity

### Findings

During SIT374, the team raised some concern with the need to rely on different applications, based on different languages (C# vs Python), having the ability to communicate with each other. Research and development was required to be undertaken once the basic prototype solution was developed to prove out the concept.

### Justification

Not having experience with cross-application functionality, the team needed to allocate some additional time and resource towards conducting research, developing and testing a C# application that could directly communicate with (and automatically control) the python-based flight-control program. It was decided that this should be carried out in an Interim Sprint, prior to commencement of Sprint 3, after successfully completing the prototype solution (and demonstrating to the client).

### Actions

During the Interim Sprint, a C# application has been successfully developed and tested, confirming to the team that it is possible to achieve cross-functional communication and control of programs between C# and Python. Building on this achievement, the prototype solution has been further expanded to incorporate GUI functionality and simplify the installation and setup process required to fly the drone.

Current development is still underway in adapting the C# application into the Unity framework, to allow user-input directly from the Unity interface.