COMSATS University, Islamabad Pakistan

Project Name

By

Mirza Muhammad Bin Adnan Baig	**CIIT/FA15-BCS-048/ISB**
Hamza Ahmad	**CIIT/FA15-BCS-069/ISB**
Muhammad Ussama Irfan	**CIIT/FA15-BCS-088/ISB**

Supervisor

Dr. Tahir Mustafa Madni

Bachelor of Science in Computer Science (2015-2019)

The candidate confirms that the work submitted is their own and appropriate credit has been given where reference has been made to the work of others.

COMSATS University, Islamabad Pakistan

Autonomous Quadcopter

A project presented to

COMSATS Institute of Information Technology, Islamabad

In partial fulfillment

of the requirement for the degree of

Bachelor of Science in Computer Science (2015-2019)

By

Mirza Muhammad Bin Adnan Baig	CIIT/FA15-BCS-048/ISB
Hamza Ahmad	CIIT/FA15-BCS-069/ISB
Muhammad Ussama Irfan	CIIT/FA15-BCS-088/ISB

DECLARATION

We hereby declare that this software, neither whole nor as a part has been copied out from any source. It is further declared that we have developed this software and accompanied report entirely on the basis of our personal efforts. If any part of this project is proved to be copied out from any source or found to be reproduction of some other. We will stand by the consequences. No Portion of the work presented has been submitted of any application for any other degree or qualification of this or any other university or institute of learning.

Mirza Muhammad Bin Adnan Baig	Hamza Ahmad
---------------------------	---------------------------

Muhammad Ussama Irfan

CERTIFICATE OF APPROVAL

It is to certify that the final year project of BS (CS) “Autonomous Quadcopter” was developed by **Mirza Muhammad Bin Adnan Baig (CIIT/FA15-BCS-048)**, **Hamza Ahmad (CIIT/FA15-BCS-069) and Muhammad Ussama Irfan (CIIT/FA15-BCS-088)** under the supervision of “Dr. Tahir Mustafa Madni” and that in his opinion; it is fully adequate, in scope and quality for the degree of Bachelors of Science in Computer Sciences.

Supervisor

External Examiner

Head of Department
(Department of Computer Science)

Executive Summary

For past decade quadcopters have become common and day by day its needs are increasing. But still most of the quadcopters are being controller from radio transmitters and some of them through smartphones. There has been a lot of development throughout the years but still must be controlled by humans. Therefore, it all depends upon human capabilities and piloting skills on how he control the quadcopter. Common or consumer grade quadcopters cannot perform computationally intensive and time sensitive Computer Vision algorithms because they do not have enough computational power. Most of the people use live camera feed and sensor data to external computers for computations and decision making. Then computer send decisions to the drone for execution which produce delay between actions performed by drone due to which risk of failure and collision is high.

To prevent these problems, quadcopter must be made capable so it is aware of its surroundings and take reasonable actions autonomously because of which human factor will be out of equation which was causing delay. It can be done by using extremely powerful Computer Vision techniques such as Visual SLAM and VIO. Quadcopter will be able to fly autonomously and be able to make logical decision depending upon its environment.

Autonomous Quadcopter will not only be able to fly autonomously but can also be assigned different tasks like tracking, 2D mapping of an area and waypoint navigation. These tasks can be assigned through Mission Ground Control application. While performing all the mention tasks quadcopter will able to detect and avoid obstacles and will also be able to perform emergency handling.

Acknowledgement

All praise is to Almighty Allah who bestowed upon us a minute portion of His boundless knowledge by virtue of which we were able to accomplish this challenging task.

We are greatly indebted to our project supervisor "Dr. Tahir Mustafa Madni". Without his personal supervision, advice and valuable guidance, completion of this project would have been doubtful. We are deeply indebted to him for their encouragement and continual help during this work.

And we are also thankful to our parents and family who have been a constant source of encouragement for us and brought us the values of honesty & hard work.

Mirza Muhammad Bind Adnan Baig   Hamza Ahmad

---------------------------   ---------------------------

Muhammad Ussama Irfan

Abbreviations

SRS	Software Require Specification
SDS	Software Design Specification
SDD	Software Design Description
SE1	Software Engineering 1
ICP	Introduction to Computer Programming
OOP	Object Oriented Programming
2D	2-Dimensional
AI	Artificial Intelligence
UC	Use Case
FR	Functional Requirement
NFR	Non-Functional Requirement
GPS	Global Positioning System
Lat	Latitude
Long	Longitude

Table of Contents

List of Figures

1 Introduction

A system in which user will be able to assign different tasks to the quadcopter through Mission Ground Control Application. Task that can be assign includes tracking an object autonomously, navigating from one place to another autonomously, navigating over an area while automatically capturing images for generating 2D maps and while performing all these tasks the quadcopter will autonomously detect and avoid obstacles.

1.1 Brief

The Project that is being developed is called "Autonomous Quadcopter". A quadcopter will be built with NVIDIA Jetson TX2, which is the most power-efficient embedded AI computing device, onboard for executing computationally intensive and time sensitive Computer Vision, Navigational and Control Algorithms in real-time. These algorithms will help the quadcopter to fly autonomously and complete missions while making all decisions itself. With the help of these algorithms the quadcopter will be able to detect any potential obstacles in its path and avoid them when necessary, autonomously follow an object being tracked, autonomously navigate from a starting position to destination, autonomously navigate over an area while automatically capturing images for generating 2D maps. The user will use Mission Ground Control Application to control and assign any mission to the quadcopter.

1.2 Relevance to Course Modules

The project that I built reflects a lot of courses directly and indirectly, that I studied during BSCS.

1.2.1 Introduction to Computers and Programming

This course provided us with all the basic programming fundamentals. It help us how to build logic and perform different tasks.

1.2.2 Object Oriented Programming

Like ICP, OOP also enhance our programming skills. It enhance our knowledge about computer programming and other aspects related to it.

1.2.3 Software Engineering

Software Engineering related concepts was taught by this subject. It taught us how to select appropriate software process for our project. Project scope, SRS, SDS were developed thanks to the skills gained because of this subject.

1.2.4 Electronics

As one of the modules of project is Building the quadcopter so it was necessary to have prior knowledge about different electrical components and there working and how to connect them.

1.2.5 Artificial Intelligence

There are a lot of AI algorithms used in our project. Without knowledge about this course it would not be possible for us to understand and implement these algorithms.

1.3 Project Background

Quadcopters have been the most popular flying machine so far. It has been through tremendous amounts of incremental and iterative improvements throughout the years but still the quadcopters must be controlled by humans. Most of the people buy commercially available Ready to Fly quadcopters to train their selves to fly. Most of the consumer and even professional grade drones must still be controlled from Radio Transmitter or smartphone. Some of the consumer and professional grade drones are regarded as intelligent and smart by their respective companies yet they still don’t have any form of Obstacle Detection and Obstacle Avoidance. These quadcopters don’t even have enough computational power onboard to execute computationally intensive and time sensitive Computer Vision algorithms.

What we are trying to build is an Autonomous Quadcopter which basically doesn’t require continuous inputs or assistance, from any external sources, to fly. Depending upon the situation, it can take necessary decisions on its own while flying. The degree of autonomy depends upon the strength, effectiveness and efficiency of the state-of-the-art computer vision algorithms that are programmed inside the quadcopter. The product that will be developed will help in making a quadcopter aware of its surroundings by using extremely powerful and cutting-edge Computer Vision techniques such as Visual Simultaneous Localization and Mapping (Visual SLAM) and Visual Inertial Odometry (VIO). These techniques will not only enable the quadcopter to fly autonomously but also enable it to make logical decisions depending upon its environment and situation. The quadcopter will be built with NVIDIA Jetson TX2, which is the most power-efficient embedded AI computing device, onboard for executing computationally intensive and time sensitive Computer Vision, Navigational and Control Algorithms in real-time. These algorithms will help the quadcopter to fly autonomously and complete missions while making all decisions itself.

1.4 Related System Analysis

The system related to the proposed system comprises of Quadcopters from Consumer and Professional Drones Manufacturers like **DJI, Parrot**, **Syma**, **JJRC** etc**.** Most of their drones must still be controlled from Radio Transmitter or smartphone. Some of their products are regarded as intelligent and smart by their respective companies yet they still don’t have any form of Obstacle Detection and Obstacle Avoidance. Names of some popular and expensive products from **Parrot**, **DJI** and **Syma**, their weakness and proposed solution are mentioned in the following table.

Product Name	**Weaknesses**	**Proposed Project Solution**
Drones from Parrot like Parrot Bebop 2, Parrot Mambo and Parrot Anafi etc.	• Must be controlled by human pilot. • No obstacle detection and obstacle avoidance technique used in any quadcopters. • Object Tracking technique uses target’s smartphone’s GPS sensor to locate the target. • No powerful hardware for onboard computation and decision making.	• Fully Autonomous Flight without any external existence. • Intelligent Obstacle Detection and Obstacle Avoidance Techniques. • Effective and Robust Visual Tracking System. • Contains Powerful embedded system for real-time processing and decision making.
Drones from DJI like the Mavic Air and Mavic Pro, Spark, Phantom 3 and Phantom 4 etc.	• Must be controlled by human pilot. • Average Object Tracking technique used in some quadcopters. • No powerful hardware for onboard computation and decision making.	• Fully Autonomous Flight without any external existence. • Effective and Robust Visual Tracking System. • Contains Powerful embedded system for real-time processing and decision making.

Drones from Syma like X25 Pro, X8 Pro, X23W, X22W, X5UW-D etc.	• Must be controlled by human pilot. • No obstacle detection and obstacle avoidance techniques used in all quadcopters. • Object Tracking technique uses target's smartphone's GPS sensor to locate the target. • No powerful hardware for onboard computation and decision making.	• Fully Autonomous Flight without any external existence. • Intelligent Obstacle Detection and Obstacle Avoidance Techniques. • Effective and Robust Visual Tracking System. • Contains Powerful embedded system for real-time processing and decision making.

1.5 Methodology and Software Lifecycle for This Project

1.5.1 Programming Paradigm:

The Procedural Programming Paradigm is selected as a software design methodology for the development of our system. As our project requires working with an Embedded System namely NVIDIA Jetson TX2, so it is obligatory for us to use Procedural instead of Object Oriented as a necessary programming paradigm. Most embedded systems are quite small, and things like memory space and computational resources are to be carefully managed. I can't have something I don't understand come by asynchronously (or worse, non-deterministically) and collect my garbage. While such a thing might indeed provide a more comfortable environment for writing a Windows application, those aren't, as a rule, real-time operations. The goal in real-time embedded systems should be that every byte and every cycle is known and accounted for. Embedded software needs clear program organization, so it won't collapse under its own weight. But embedded software often has other stringent requirements that demand a design with low overhead, tight memory and real-time constraints that may argue against on-demand allocation of objects at run time.

1.5.2 Software Process Methodology:

The selected software development methodology is Incremental method because of its nature to decompose the required product into several components depending upon functionality, each of which is designed and built separately. The incremental software development method is a method of software development where the product is designed, implemented and tested incrementally. Project requirements are divided into separate modules and each of them are developed separately. It is beneficiary to use Incremental model in developing large applications as it allows to complete each functionality of the required product incrementally and to add additional functionality later.

2 Problem Definition

2.1 Problem Statement

Quadcopter have been around for decades now and most of them are still being controlled from radio transmitters and some are being controlled from smartphones. There have been tremendous amounts of incremental and iterative improvements throughout the years but still quadcopters must be controlled by humans. Therefore, it can be assumed that the limit to a quadcopter's ability depends upon the skillfulness of its pilot. Consumer grade quadcopters don't have enough computational power onboard to execute computationally intensive and time sensitive Computer Vision algorithms. Researchers, enthusiasts and hobbyist usually send live camera feed and sensor data to external computers for computation and decision making. Computed decisions are then sent to the drone for execution. This process introduces a seemingly small but significant delay between consecutive actions performed by the drone, and this simultaneously introduces a high degree of risk of failure and collision.

2.2 Deliverables and Development Requirements

2.2.1 Project Report

A complete project report that includes Software Requirement Specifications, Software Design Specifications, and Test Cases.

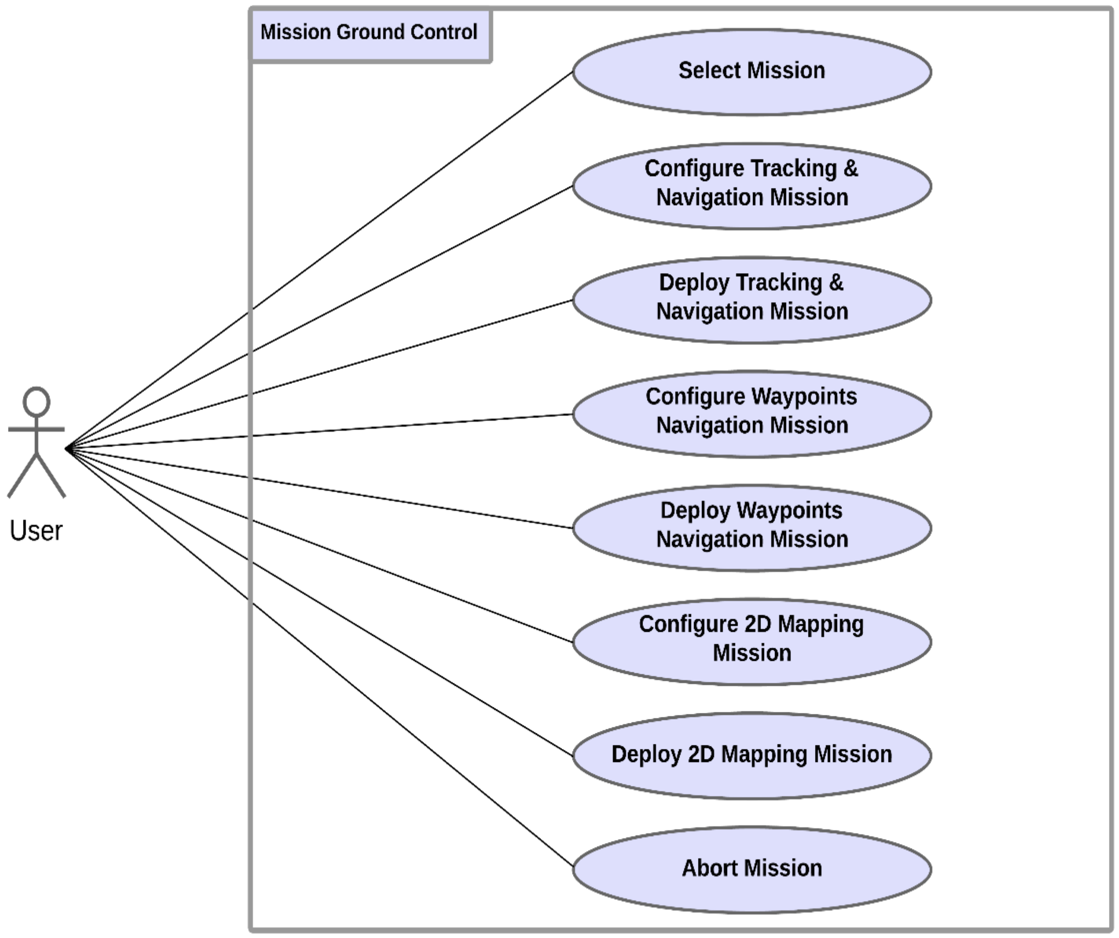
2.2.2 Source Code

A CD including the source code of the application

3 Requirement Analysis

As the user will interact with the Mission Ground Control System that will act as a middleware in establishing the communication between the user and the quadcopter, so use cases are selected as the primary requirement identifying technique.

3.1 Use Cases Diagram(s)



Mission Ground Control
Select Mission
Configure Tracking & Navigation Mission
Deploy Tracking & Navigation Mission
Configure Waypoints Navigation Mission
Deploy Waypoints Navigation Mission
Configure 2D Mapping Mission
Deploy 2D Mapping Mission
Abort Mission
User

Figure 3.1: Use Case Diagram

3.2 Detailed Use Case

3.2.1 UC-1: Select Mission

Use Case ID:	UC-1
Use Case Name:	Select Mission
Actors:	User
Description:	User Selects the mission to be performed by the quadcopter.
Trigger:	User will select the desired mission from the Mission Menu.
Preconditions:	Nil
Postconditions:	User will be directed to the selected mission screen for further details.

Normal Flow:	1. User shall navigate to the mission selection menu 2. User will select the desired mission from the Mission Menu. 3. User will be directed to the selected mission screen for further details.
Alternative Flows:	N/A
Exceptions:	N/A
Business Rules	N/A
Assumptions:	Nil

3.2.2 UC-2: Configure Tracking & Navigation Mission

Use Case ID:	UC-2
Use Case Name:	Configure Tracking & Navigation Mission
Actors:	User
Description:	User must provide necessary information regarding the Tracking & Navigation mission.
Trigger:	User selects the Tracking & Navigation Mission from the Mission Menu.
Preconditions:	Nil
Postconditions:	Necessary information regarding the mission has been confirmed. The User can then deploy the mission onto the quadcopter.
Normal Flow:	1. User selects the Tracking & Navigation Mission from the Mission Menu. 2. User must select the height from the ground to be maintained while tracking the target from the drop-down menu. 3. User must select the distance from the target to be maintained from the drop-down menu. 4. The User can then deploy the mission onto the quadcopter.
Alternative Flows:	N/A
Exceptions:	N/A
Business Rules	N/A
Assumptions:	Nil

3.2.3 UC-3: Deploy Tracking & Navigation Mission

Use Case ID:	UC-3
Use Case Name:	Deploy Tracking & Navigation Mission
Actors:	User
Description:	After the user has provided necessary mission information, the user opted to deploy the mission onto the Quadcopter.
Trigger:	User selects the Execute Mission on Quadcopter option.
Preconditions:	User must have provided all the necessary mission relevant

	information.
Postconditions:	Quadcopter started Tracking and Following the target specified by the user.
Normal Flow:	1. After the user has provided necessary mission information, user selects the Execute Mission on Quadcopter option. 2. Mission information sent to the quadcopter. 3. Quadcopter takes off and reach the height specified by the user. 4. User selects the target object to track and follow by selecting it from the Live Camera Feed screen in the Mission Ground Control. 5. Quadcopter starts tracking and following the target while maintaining user specified distance from target.
Alternative Flows:	N/A
Exceptions:	N/A
Business Rules	N/A
Assumptions:	Nil

3.2.4 UC-4: Configure Waypoints Navigation Mission

Use Case ID:	UC-4
Use Case Name:	Configure Waypoints Navigation Mission
Actors:	User
Description:	User must provide necessary information regarding the Waypoints Navigation mission.
Trigger:	User selects the Waypoints Navigation Mission from the Mission Menu.
Preconditions:	Nil
Postconditions:	Necessary information regarding the mission has been confirmed. The User can then deploy the mission onto the quadcopter.
Normal Flow:	1. User selects the Waypoints Navigation Mission from the Mission Menu. 2. User must select the height to be maintained during the mission from the drop-down menu. 3. User must select the waypoints by clicking on desired points on the map screen. 4. User must specify the order in which the waypoints should be visited. 5. The User can then deploy the mission onto the quadcopter.
Alternative Flows:	N/A
Exceptions:	N/A
Business Rules	N/A
Assumptions:	Nil

3.2.5 UC-5: Deploy Waypoints Navigation Mission

Use Case ID:	UC-5
Use Case Name:	Deploy waypoints Navigation Mission
Actors:	User
Description:	After the user has provided necessary mission information, the user opted to deploy the mission onto the Quadcopter.
Trigger:	User selects the Execute Mission on Quadcopter option.
Preconditions:	User must have provided all the necessary mission relevant information.
Postconditions:	Quadcopter navigates through all the waypoints and lands at the last waypoint.
Normal Flow:	1. After the user has provided necessary mission information, user selects the Execute Mission on Quadcopter option. 2. Mission information sent to the quadcopter. 3. Quadcopter takes off and reach the height specified by the user. 4. Quadcopter visits all waypoints and land at the last waypoint.
Alternative Flows:	N/A
Exceptions:	N/A
Business Rules	N/A
Assumptions:	Nil

3.2.6 UC-6: Configure 2D Mapping Mission

Use Case ID:	UC-6
Use Case Name:	Configure 2D Mapping Mission
Actors:	User
Description:	User must provide necessary information regarding the 2D Mapping mission.
Trigger:	User selects the 2D Mapping Mission from the Mission Menu.
Preconditions:	Nil
Postconditions:	Necessary information regarding the mission has been confirmed. The User can then deploy the mission onto the quadcopter.
Normal Flow:	1. User selects the 2D Mapping Mission from the Mission Menu. 2. User must select the height to be maintained during the mission from the drop-down menu. 3. User must select the area to be mapped be making a bounding box on the map screen. 4. The User can then deploy the mission onto the quadcopter.
Alternative Flows:	N/A
Exceptions:	N/A
Business Rules	N/A
Assumptions:	Nil

3.2.7 UC-7: Deploy 2D Mapping Mission

Use Case ID:	UC-7
Use Case Name:	Deploy 2D Mapping Mission
Actors:	User
Description:	After the user has provided necessary mission information, the user opted to deploy the mission onto the Quadcopter.
Trigger:	User selects the Execute Mission on Quadcopter option.
Preconditions:	User must have provided all the necessary mission relevant information.
Postconditions:	Quadcopter navigates over the area while capturing images. After the area has been completely navigated the simulated quadcopter return to its initial position and lands.
Normal Flow:	1. After the user has provided necessary mission information, user selects the Execute Mission on Quadcopter option. 2. Mission information sent to the quadcopter. 3. Quadcopter takes off and reach the height specified by the user. 4. Quadcopter navigates over the area selected by the user while automatically taking pictures at regular intervals. 5. After completely navigating over the area the Quadcopter return to its initial position and lands.
Alternative Flows:	N/A
Exceptions:	N/A
Business Rules	N/A
Assumptions:	Nil

3.2.8 UC-8: Abort Mission

Use Case ID:	UC-8
Use Case Name:	Abort Mission
Actors:	User
Description:	User can abort any mission by pressing the Escape button and the Quadcopter will return and land at the take-off location.
Trigger:	User pressed the Escape Button on the keyboard.
Preconditions:	Quadcopter must be executing / carrying out a mission.
Postconditions:	Quadcopter will return to the take-off location and land.
Normal Flow:	1. While a Quadcopter is carrying out a mission, user presses the Escape button on the keyboard. 2. Quadcopter Aborts any mission currently being carrying out. 3. Quadcopter returns to the take-off location and lands.
Alternative Flows:	N/A
Exceptions:	N/A
Business Rules	N/A
Assumptions:	Nil

3.3 Functional Requirements

3.3.1 FR-1: Track the Target object

Identifier	FR-1
Title	Track the Target Object
Requirement	The quadcopter shall be able to track or localize the target object in every frame of the camera.
Source	Supervisor
Rationale	The quadcopter should be able to successfully track the target object, specified by the user, by calculating the position of the target object in every frame coming from the camera.
Business Rule (if required)	Nil
Dependencies	Nil
Priority	High

3.3.2 FR-2: Follow the Target Object

Identifier	FR-2
Title	Follow the Target Object
Requirement	The quadcopter shall be able to follow the target object by analyzing the position of the target object, received from FR-1, of every consecutive frame. The quadcopter should adjust its position, angle and speed in order to keep the target object at the center of the frame.
Source	Supervisor
Rationale	The quadcopter should follow the target object by adjusting its position, angle and speed accordingly.
Business Rule (if required)	Nil
Dependencies	FR-1
Priority	High

3.3.3 FR-3: Sending Live Video Feed

Identifier	FR-3
Title	Sending Live Video Feed
Requirement	The quadcopter shall be able to send live video feed to the Mission Ground Control software.
Source	Supervisor
Rationale	The user should be able to monitor the quadcopter's movements using the live video feed.
Business Rule (if required)	Nil
Dependencies	Nil
Priority	High

3.3.4 FR-4: Sending Sensors Readings

Identifier	FR-4
Title	Sending Sensors Readings
Requirement	The quadcopter shall be able to send its sensors readings to the Mission Ground Control software.
Source	Supervisor
Rationale	The user should always be able to monitor the state of the quadcopter.
Business Rule (if required)	Nil
Dependencies	Nil
Priority	High

3.3.5 FR-5: Saving Waypoints

Identifier	FR-5
Title	Saving Waypoints
Requirement	The Quadcopter shall be able to save the GPS waypoints and their order of traversal, selected by the user from the map in the Mission Ground Control.
Source	Supervisor
Rationale	The user would want the quadcopter to save all waypoints and their order of traversal in its memory.
Business Rule (if required)	Nil
Dependencies	Nil
Priority	High

3.3.6 FR-6: Traversing Waypoints

Identifier	FR-6
Title	Traversing Waypoints
Requirement	The Quadcopter shall be able to visit all the GPS waypoints in the order specified by the user.
Source	Supervisor
Rationale	The user would want the quadcopter to visit all the waypoints in user specified order.
Business Rule (if required)	Nil
Dependencies	FR-5
Priority	High

3.3.7 FR-7: Detecting and Avoiding Obstacles

Identifier	FR-7
Title	Detecting and Avoiding Obstacles
Requirement	The quadcopter shall be able to detect any potential obstacles and avoid
	them when necessary.
Source	Supervisor
Rationale	The user should not have to worry about the quadcopter crashing into obstacles during the assigned mission.
Business Rule (if required)	Nil
Dependencies	FR-1
Priority	High

3.3.8 FR-8: Generating Flight Plan

Identifier	FR-8
Title	Generating Flight Plan
Requirement	The Mission Ground Control system shall generate an appropriate flight plan, of the area selected by the user, for the quadcopter to follow.
Source	Supervisor
Rationale	The Mission Ground Control system should generate an appropriate flight plan that the quadcopter will follow in order to efficiently cover and take pictures of the entire area.
Business Rule (if required)	Nil
Dependencies	Nil
Priority	High

3.3.9 FR-9: Deploying Mission

Identifier	FR-9
Title	Deploying Mission
Requirement	The Mission Ground Control system shall be able to deploy / assign the selected mission, with all its requirements completed, to the quadcopter for it to start.
Source	Supervisor
Rationale	The user would want its mission uploaded / deployed onto the quadcopter after he has completed all the mission requirements.
Business Rule (if required)	Nil
Dependencies	Nil
Priority	High

3.3.10 FR-10: Handling Emergencies

Identifier	FR-10
Title	Handling Emergencies
Requirement	The Quadcopter shall be able to regularly check the battery level before and during the assigned mission and perform safety measure when necessary. These emergency situations include determining how much battery would be needed to complete a waypoint navigation mission or if a drone is unable to complete a waypoint navigation mission due to loss of GPS connection or low battery because of trying to fly to a point in a windstorm, it should check if it has enough battery to return home or find a suitable place to land and send GPS location of its landing spot so that it can be located by the user.
Source	Supervisor
Rationale	The user would want the Quadcopter to deal with emergency situations related to the battery level by itself.
Business Rule (if required)	Nil
Dependencies	Nil
Priority	High

3.3.11 FR-11: Aborting Mission

Identifier	FR-11
Title	Handling Emergencies
Requirement	The Quadcopter would stop executing any mission after the user has confirmed aborting of mission. The Quadcopter would then immediately return to the position from where it started its flight and land.
Source	Supervisor
Rationale	The user would want the Quadcopter to return and land automatically after the user has aborted the mission.
Business Rule (if required)	Nil
Dependencies	Nil
Priority	High

3.4 Non-Functional Requirements

3.4.1 NFR-1: Usability

Identifier	NFR-1
Title	Usability
Requirement	System will be designed using consistent and standardized approaches so that it is easy to memorize, and user will face no difficulty in using the system. Power users will learn in a duration of 15 minutes as it will be similar to other mission planner or ground control systems in the market in terms of flow.

3.4.2 NRF-2: Modularity

Identifier	NFR-2
Title	Modularity
Requirement	System will be developed in modular way. Modularity increases efficiency of the system. Modular systems are also reusable. Modular development is fast and meaningful also. As system is developed using modular approach so it makes our system modules reusable also.

3.4.3 NFR-3: Performance

Identifier	NFR-3
Title	Performance
Requirement	The performance of the system is highly considered for the critical environment in which the system will deployed. Efficient and Effective algorithms would be executed on the NVIDIA Jetson TX2 resulting in the high performance of application and quick response timings.

4 Design and Architecture

4.1 System Architecture

The system architecture we selected for our system is Component based because our system is a prototype and major parts of this system can be reused and they can also be replaced by other components in future without breaking the system altogether. Following are the major components which exchange information between each other:

- **Mission Ground Control:**
  This application acts as a middleware between the user and the quadcopter. It allows the user to select a mission, complete the necessary mission requirements, deploy the mission onto the quadcopter and monitor the quadcopter's state, while it is executing the assigned mission, from the live sensor data and video feed that is being transferred from the quadcopter in real time.

- **Quadcopter:**
  The quadcopter will be responsible to receive the mission from the Mission Ground Control, deliver the mission to the Companion Computer for further processing. The Companion Computer which in our case is the NVIDIA Jetson TX2 will return control signals after processing that will then be fed into the Flight Controller of the quadcopter which will generate necessary signals to control the speed of each motor. The quadcopter will also be responsible for delivering sensors (3 Axis Gyroscope, 3 Axis Accelerometer, 3 Axis Magnetometer and GPS) data to the NVIDIA Jetson TX2.

- **NVIDIA Jetson TX2:**
  NVIDIA Jetson TX2 acts as the companion computer for our quadcopter. A companion computer is a device that travels on-board the vehicle and controls/communicates with the autopilot over a low-latency link. Apps running on a companion computer can perform computationally intensive or time-sensitive tasks and add much greater intelligence than is provided by the Flight Controller alone. Jetson TX2 would be responsible for executing all the computationally intensive Computer Vision, Navigation and Control Algorithms necessary for making the quadcopter autonomous.

The Architecture Diagram of our system is given on the next page.

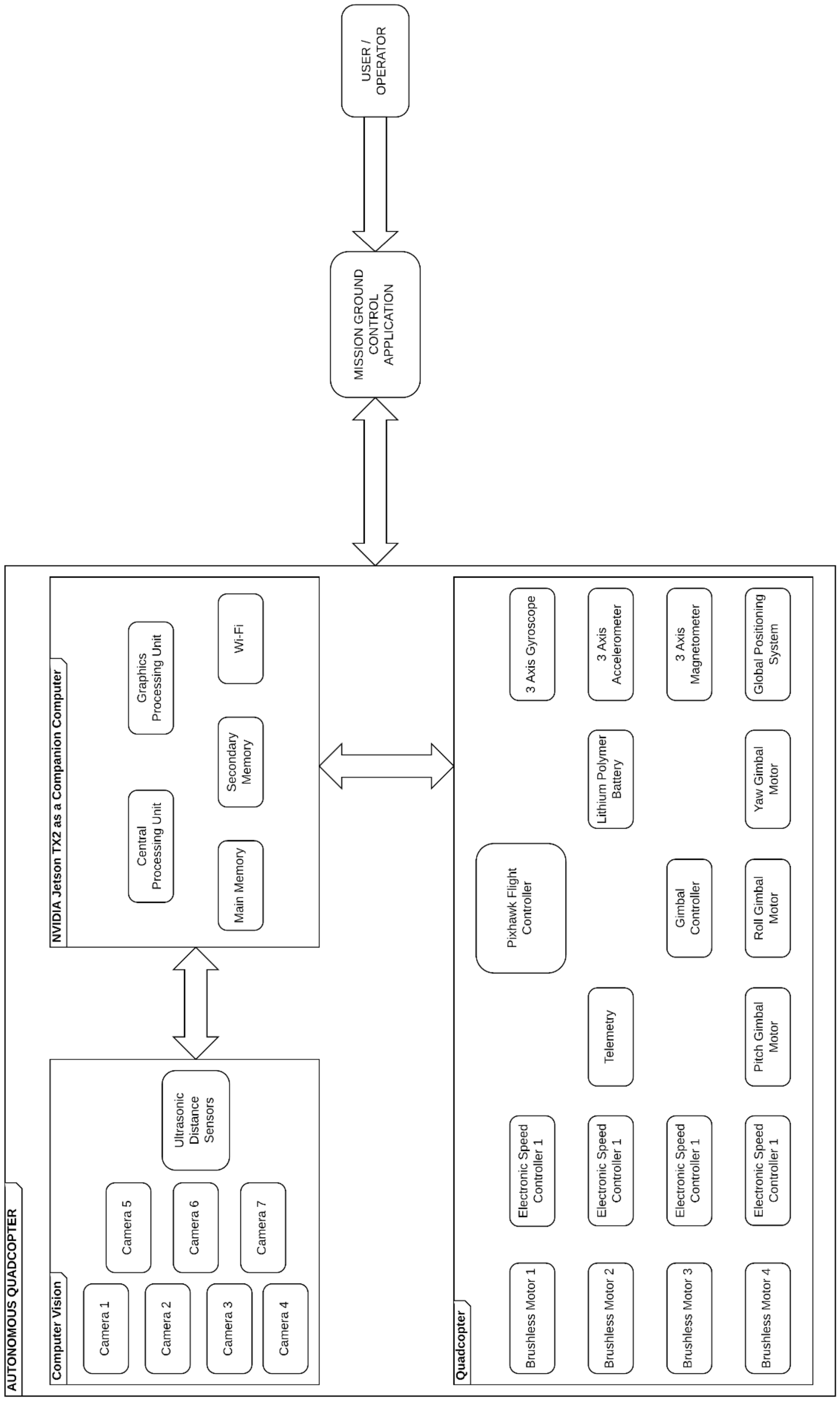


Figure 4.1 – Architecture Design Diagram

4.2 Process Flow/Representation

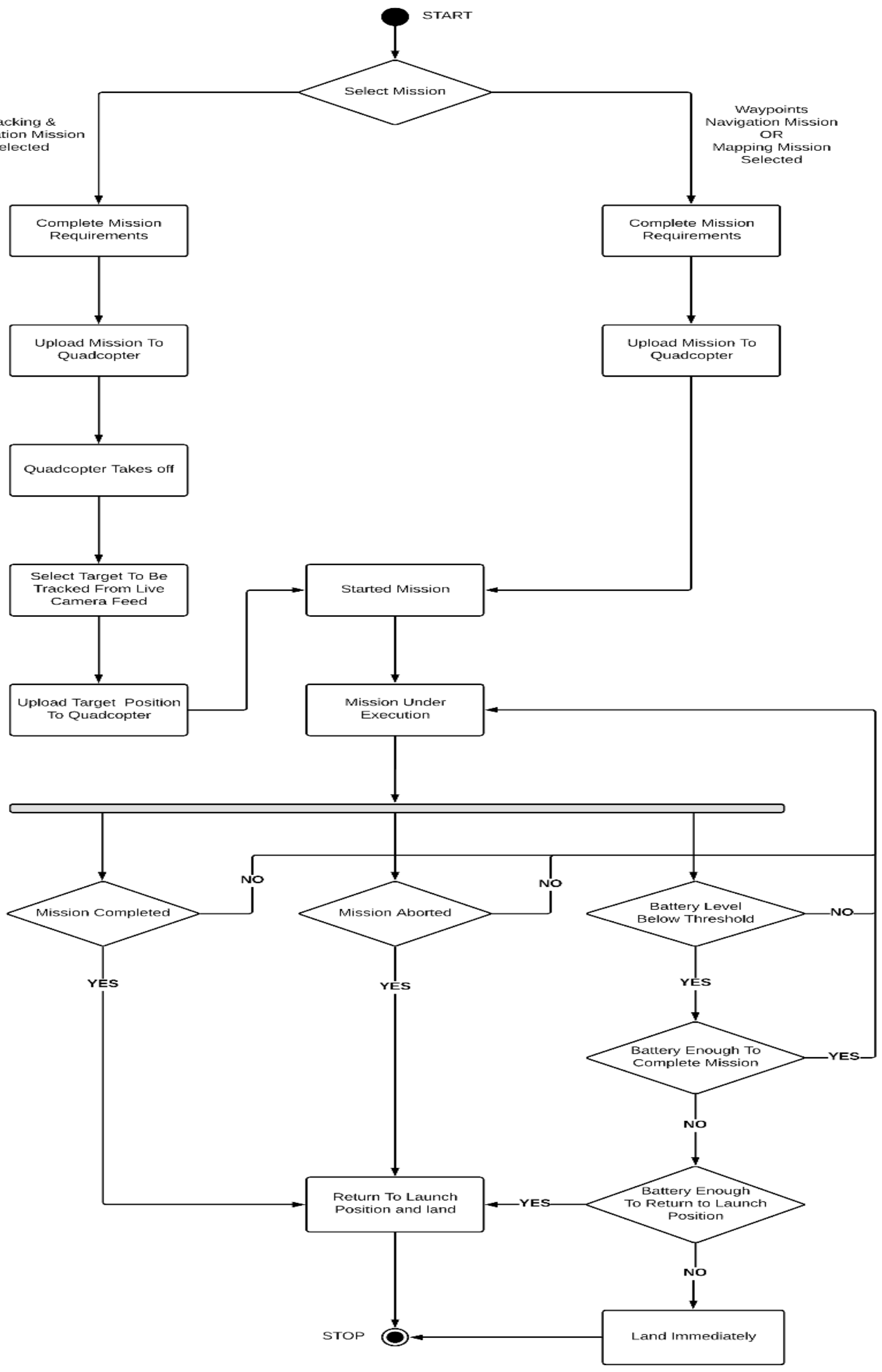

START
Select Mission
Tracking & Navigation Mission Selected
Waypoints Navigation Mission OR Mapping Mission Selected
Complete Mission Requirements
Complete Mission Requirements
Upload Mission To Quadcopter
Upload Mission To Quadcopter
Quadcopter Takes off
Select Target To Be Tracked From Live Camera Feed
Started Mission
Upload Target Position To Quadcopter
Mission Under Execution
NO
NO
Mission Completed
Mission Aborted
Battery Level Below Threshold
NO
YES
YES
YES
Battery Enough To Complete Mission
YES
NO
Return To Launch Position and land
YES
Battery Enough To Return to Launch Position
NO
STOP
Land Immediately

Figure 4.2 – Activity Diagram

4.3 Design Models

4.3.1 State Machine Diagram:

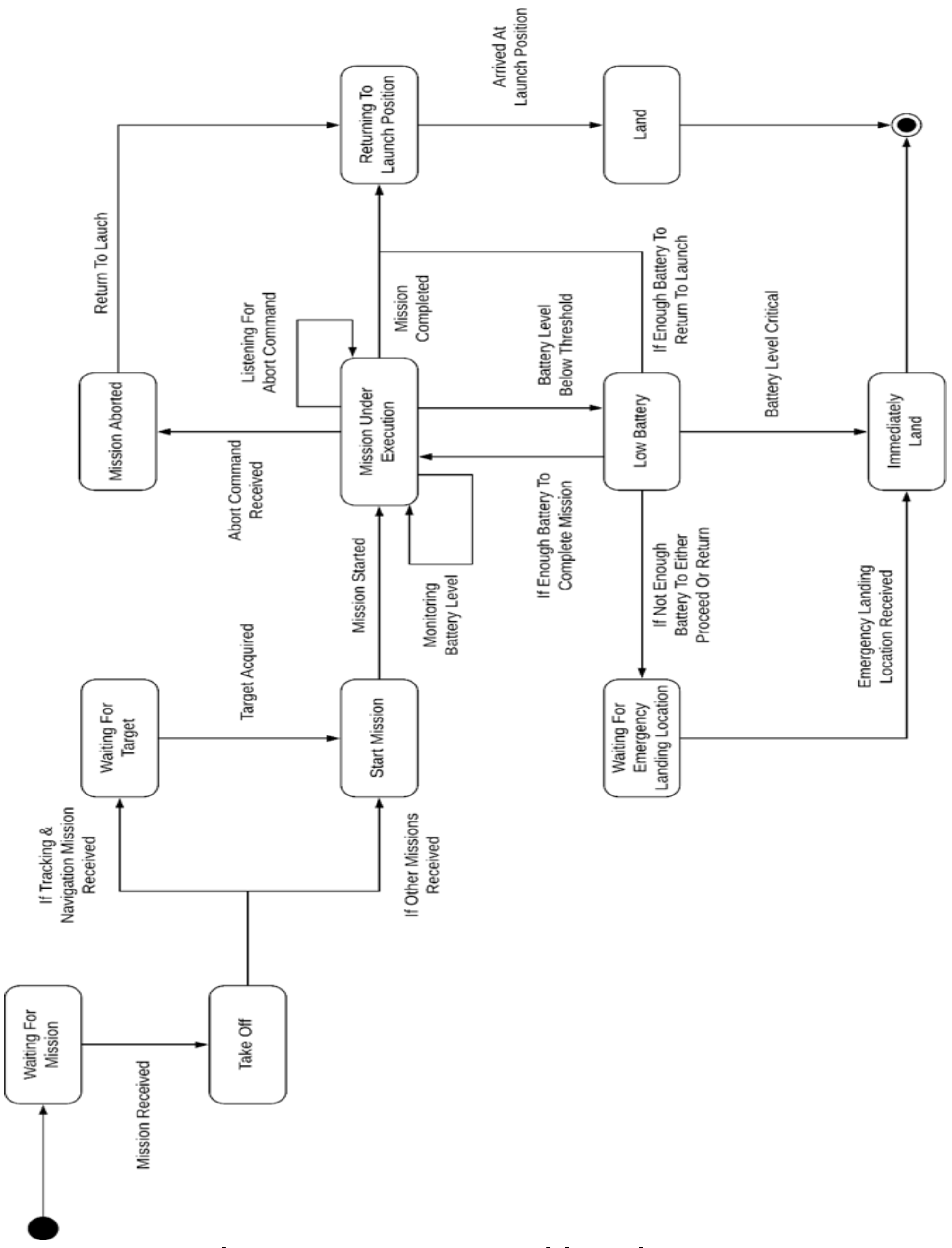

Waiting For Mission
Mission Received
Take Off
If Tracking & Navigation Mission Received
If Other Missions Received
Waiting For Target
Target Acquired
Start Mission
Mission Started
Monitoring Battery Level
Mission Under Execution
Listening For Abort Command
Abort Command Received
Mission Aborted
Return To Lauch
Mission Completed
Returning To Launch Position
Arrived At Launch Position
Land
Battery Level Below Threshold
If Enough Battery To Complete Mission
Low Battery
If Enough Battery To Return To Launch
If Not Enough Battery To Either Proceed Or Return
Waiting For Emergency Landing Location
Battery Level Critical
Immediately Land
Emergency Landing Location Received

Figure 4.3.1 – State Machine Diagram

4.3.2 Data Flow Diagram:

- **Data Flow Diagram Level 0:**

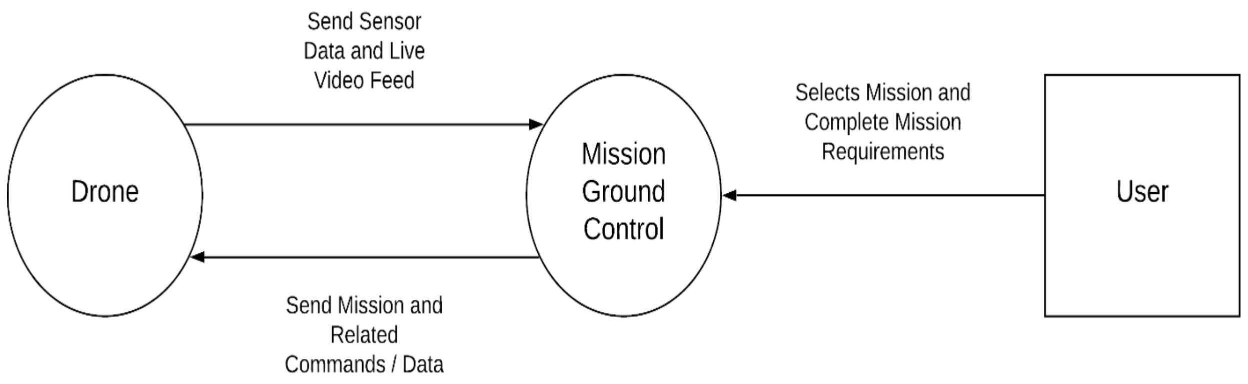

Send Sensor Data and Live Video Feed
Drone
Mission Ground Control
Selects Mission and Complete Mission Requirements
User
Send Mission and Related Commands / Data

Figure 4.3.2 (a) – Data Flow Diagram Level 0

- **Data Flow Diagram Level 1:**

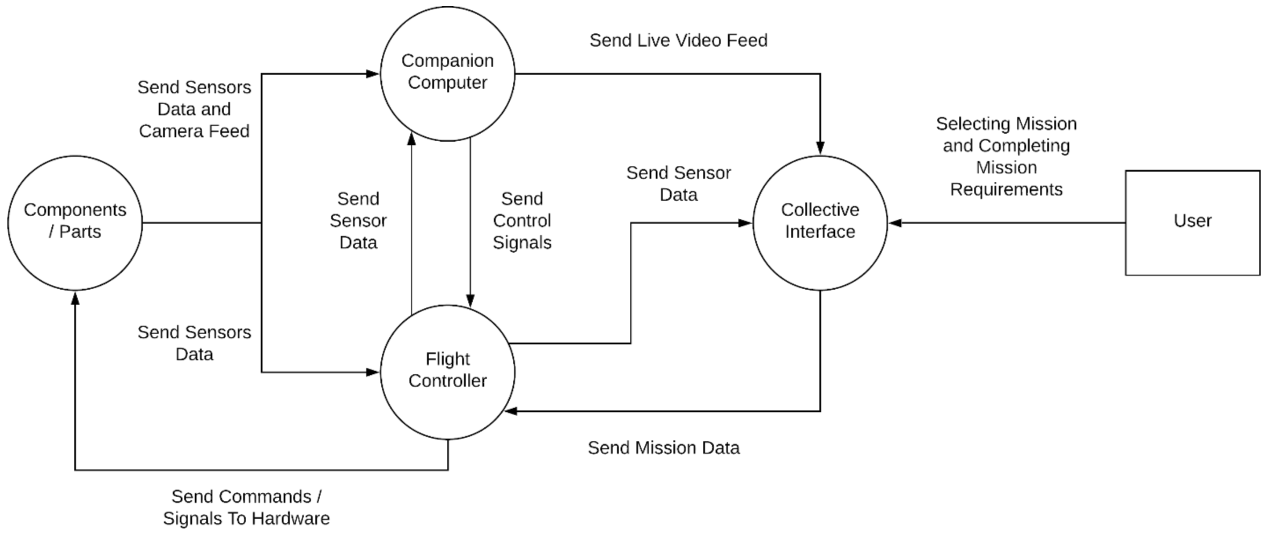


Figure 4.3.2 (b) – Data Flow Diagram Level 1

- **Data Flow Diagram Level 2:**

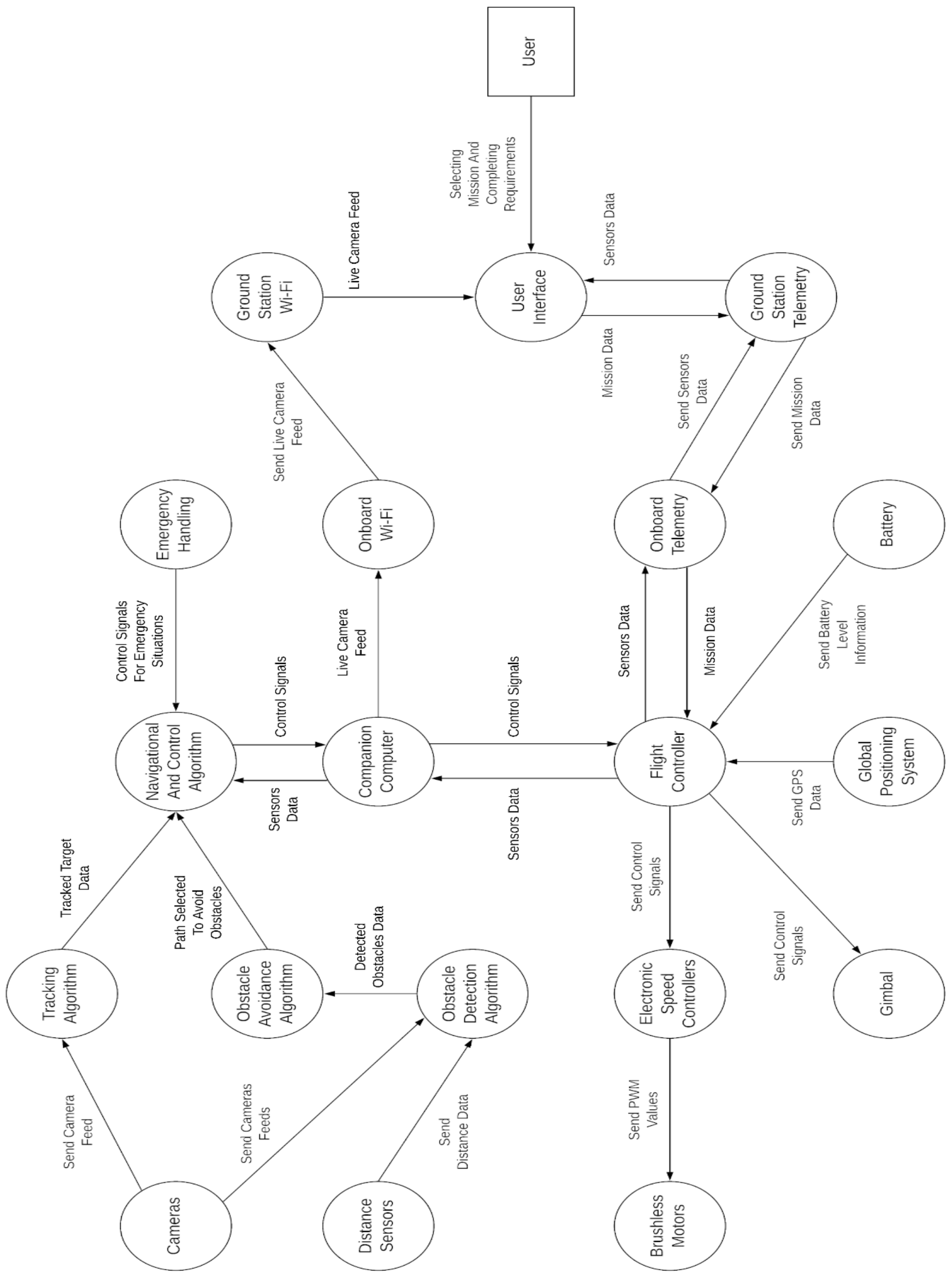


Figure 4.3.2 (c) – Data Flow Diagram Level 2

5 Implementation

5.1 External APIs

Table 1: Details of APIs used in the project

Name of API	Description of API	Purpose of usage	List down the function/class name in which it is used
DroneKit	DroneKit-Python allows developers to create apps that run on an onboard companion computer and communicate with the ArduPilot flight controller using a low-latency link.	DroneKit is used in this project to control the quadcopter.	• GOTO() • Simple_Takeoff() • Change_Yaw() • Set_Position_target_local_ned()
OpenCV	OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.	OpenCV is used in this project to aid and handle the Computer Vision algorithms used in this project.	• VideoCapture() • Read() • imshow() • waitKey()
OpenDroneMap	In a nutshell, it's a program that takes images as input and produces a variety of georeferenced assets as output, such as maps and 3D models.	OpenDroneMap is used to stitch individual images together to create a map.	• create_task() • info() • wait_for_completion() • download_assets()

5.2 User Interface

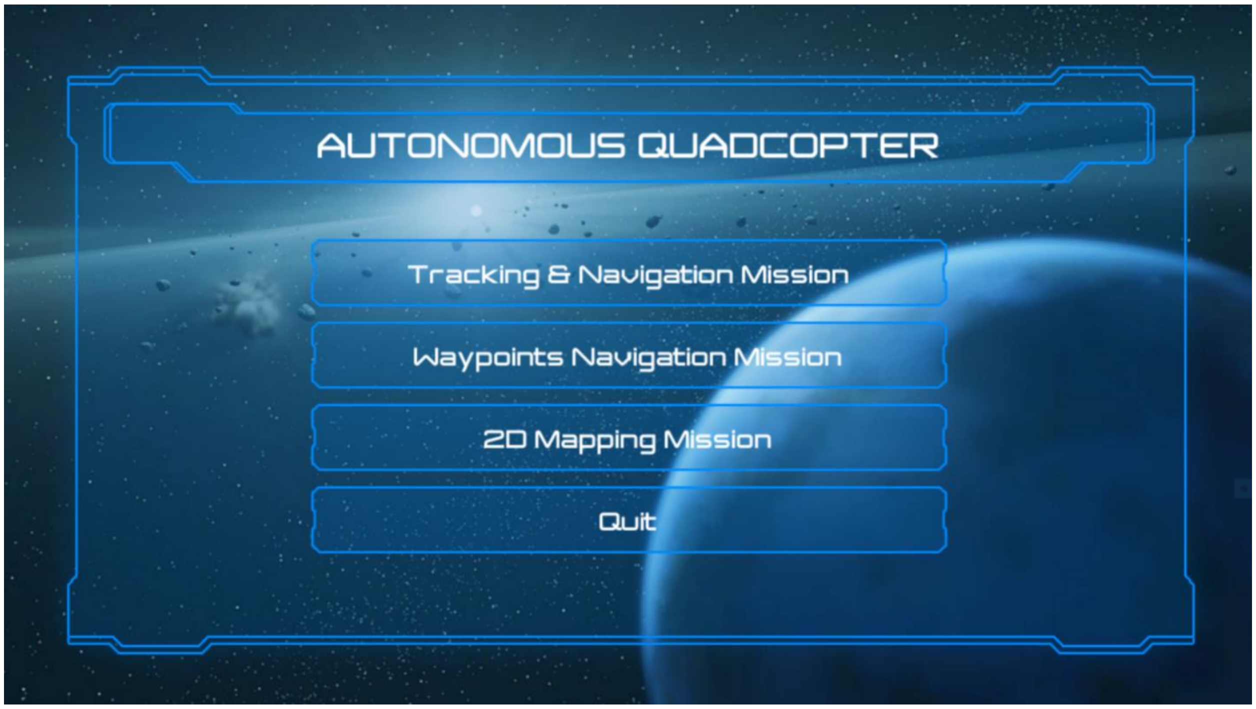


Figure 5.1: Mission Selection Screen

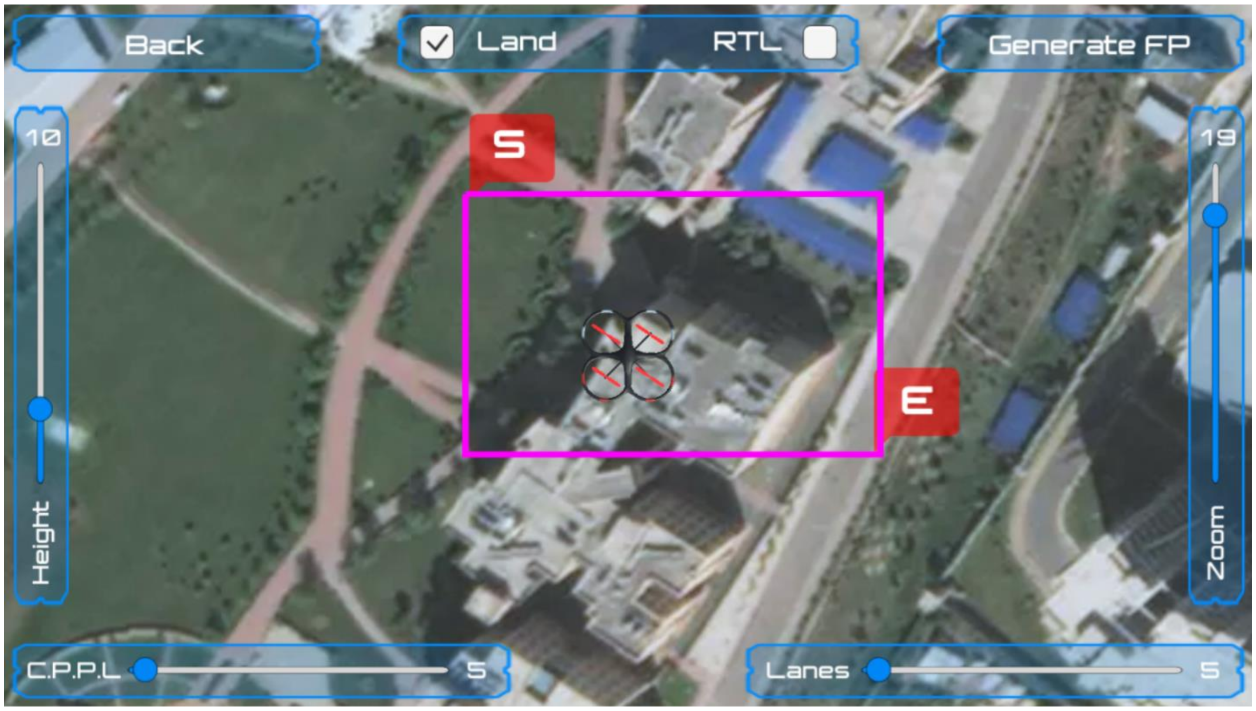


Figure 5.2: 2D Mapping Area Selection Screen

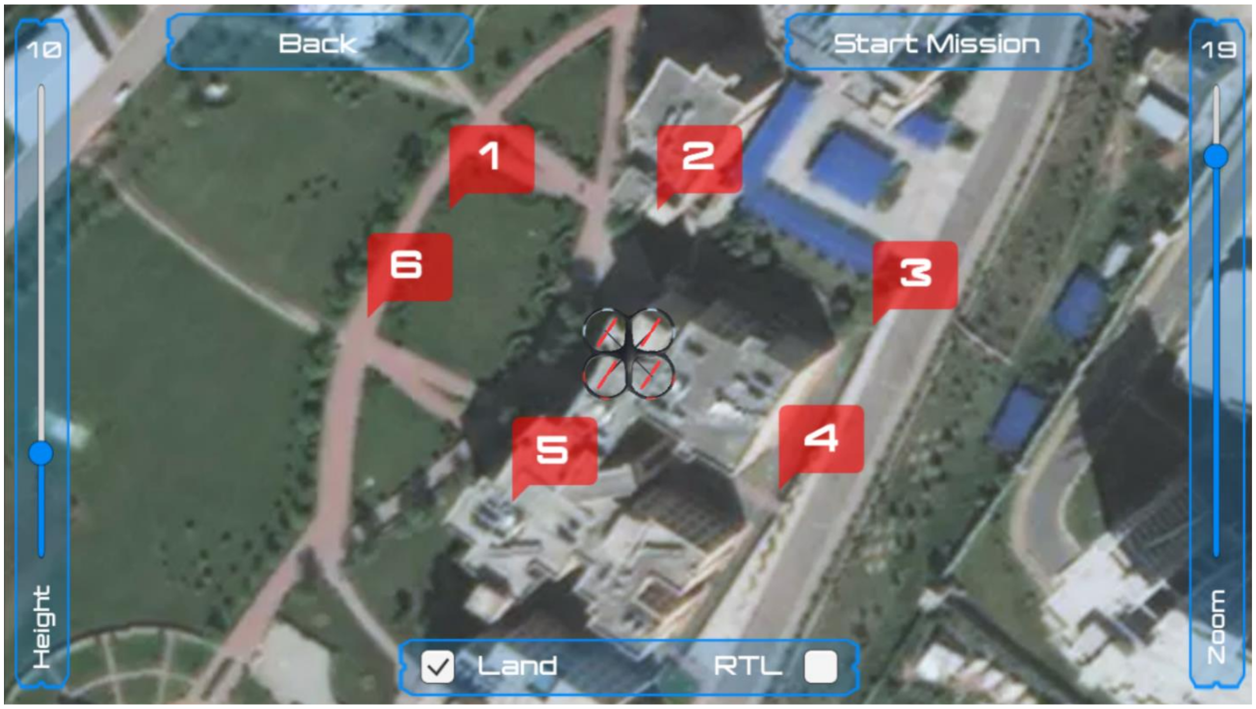


Figure 5.3: Waypoints Selection Screen

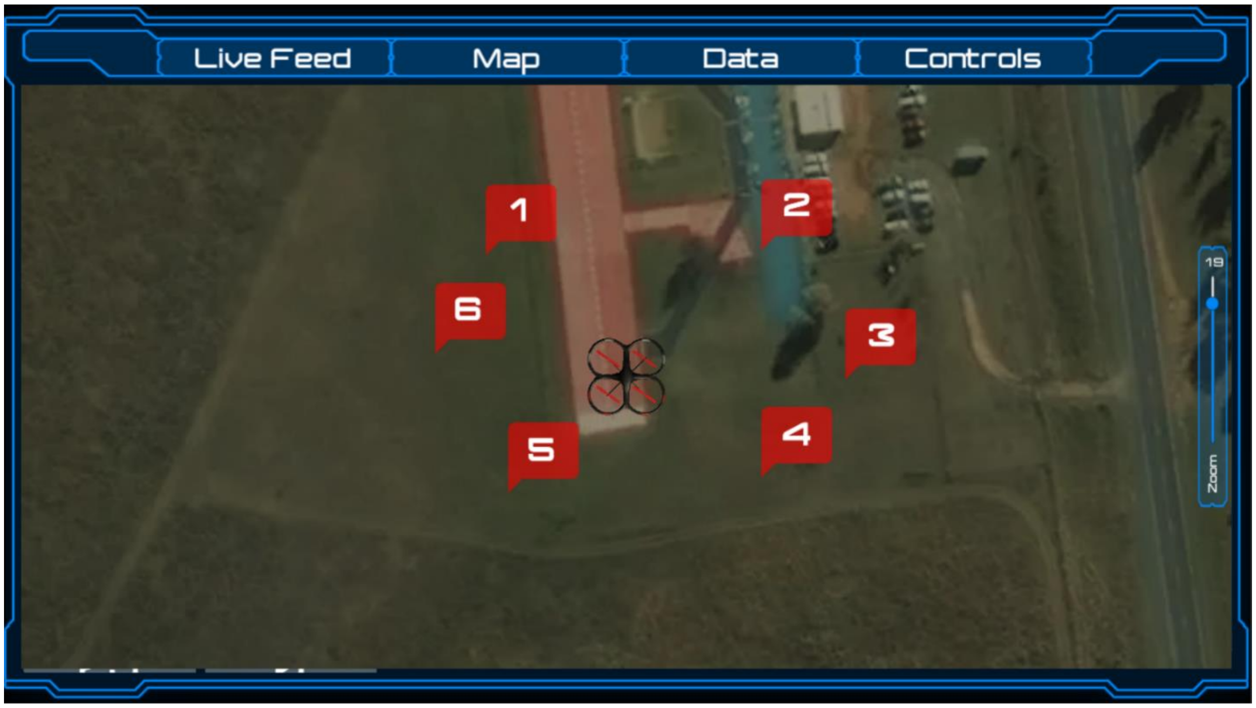
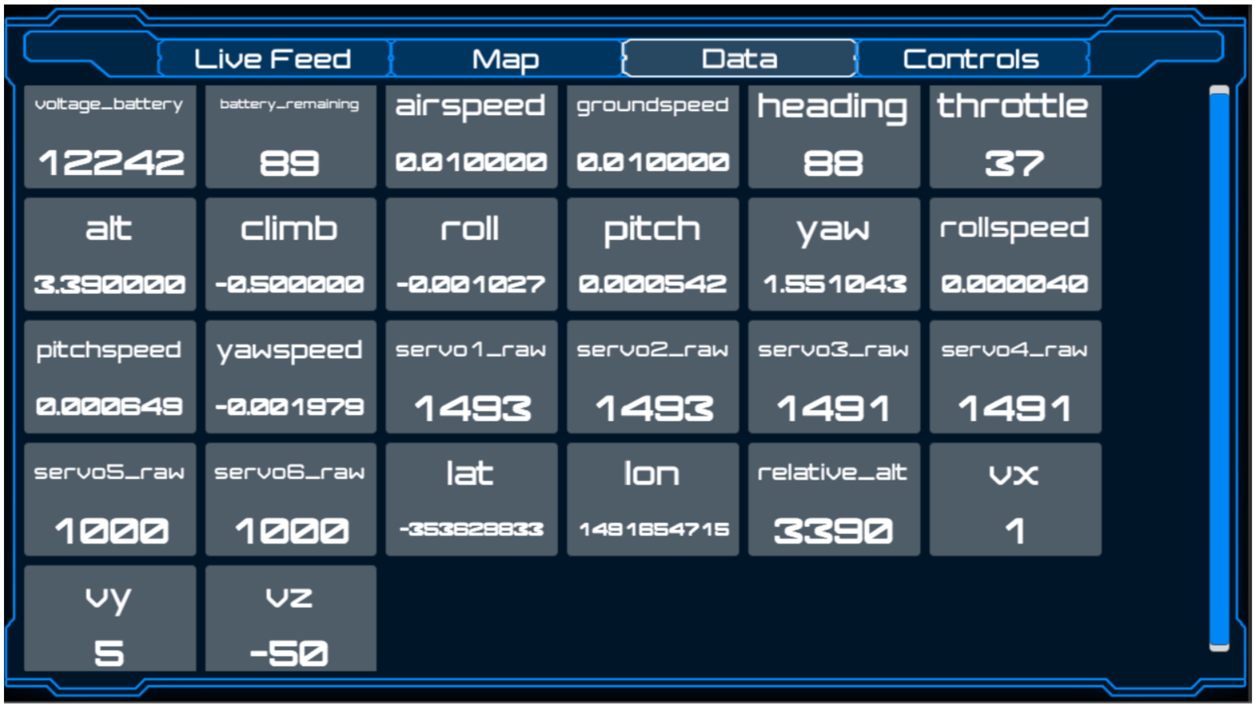


Figure 5.4: Main Map Screen



voltage_battery	battery_remaining	airspeed	groundspeed	heading	throttle
12242	89	0.010000	0.010000	88	37
alt	climb	roll	pitch	yaw	rollspeed
3.390000	-0.500000	-0.001027	0.000542	1.551043	0.000040
pitchspeed	yawspeed	servo1_raw	servo2_raw	servo3_raw	servo4_raw
0.000649	-0.001979	1493	1493	1491	1491
servo5_raw	servo6_raw	lat	lon	relative_alt	vx
1000	1000	-353628833	1491654715	3390	1
vy	vz				
5	-50				

Figure 5.5: Main Data Screen

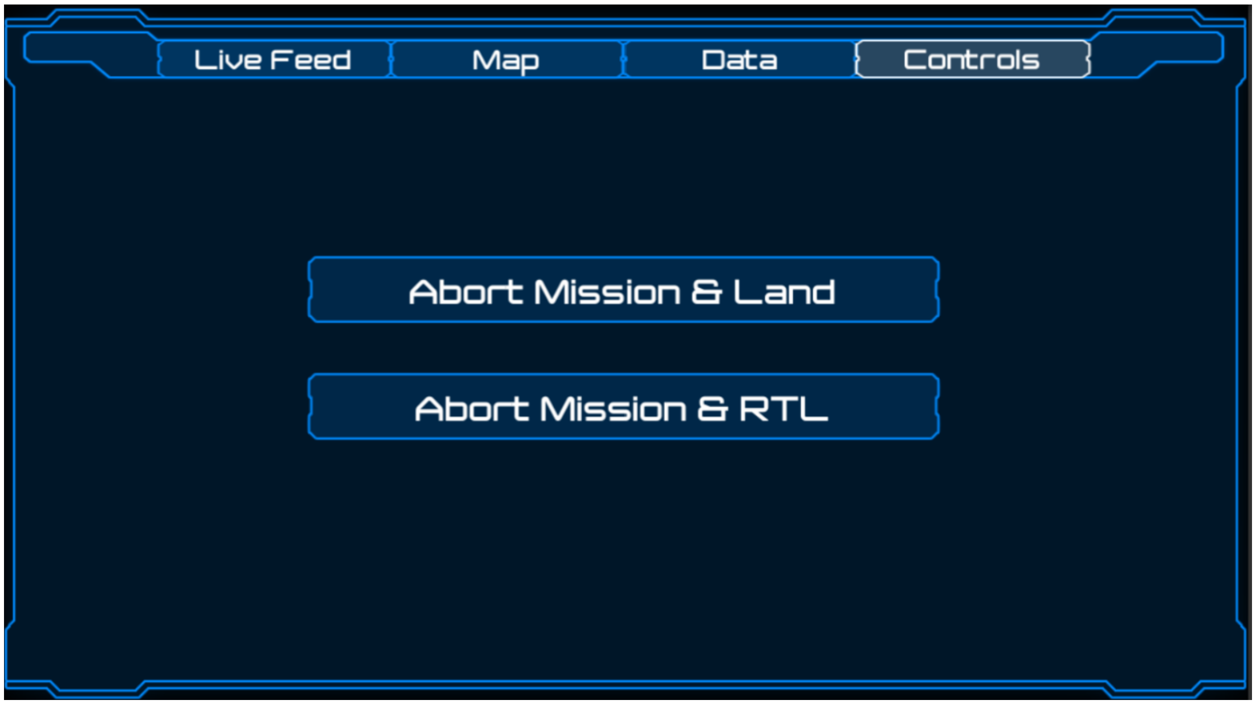


Figure 5.6: Main Controls Screen

6 Testing and Evaluation

6.1 Manual Testing

6.1.1 Unit Testing

Unit Testing 1: Take off
Testing Objective: To ensure that the quadcopter successfully takes off.

No.	Test case/Test script	Attribute and value	Expected result	Result
1.	It is to make sure that quadcopter take off and see whether it is stable or not while airborne.	• Altitude	It is stable in air.	Pass

Unit Testing 2: Land Safely
Testing Objective: To ensure that quadcopter Lands safely after taking off.

No.	Test case/Test script	Attribute and value	Expected result	Result
1.	It is to ensure that quadcopter land safely.	-	It lands safely	Pass

Unit Testing 3: Quadcopter Movement
Testing Objective: To ensure that quadcopter move in every direction

No.	Test case/Test script	Attribute and value	Expected result	Result
1.	It is to ensure that quadcopter move in all direction and is balanced.	• Altitude • Duration • x-velocity • y-velocity • z-velocity	It moves in all direction.	Pass

Unit Testing 4: Yaw check
Testing Objective: To ensure that quadcopter changes its yaw correctly.

No.	Test case/Test script	Attribute and value	Expected result	Result
1.	It is to ensure that quadcopter changes its yaw angle correctly.	• Altitude • Yaw_Angle	It changes its yaw angle correctly	Pass

Unit Testing 5: Position check
Testing Objective: To ensure that quadcopter move with respect to position.

No.	Test case/Test script	Attribute and value	Expected result	Result
1.	It is to ensure that quadcopter move towards position.	• Altitude • Position-x • Position-y • Position-z	It successfully moved with respect to its current Position.	Pass

Unit Testing 6: Velocity check with yaw angle
Testing Objective: To ensure that quadcopter move with velocity and changes its yaw as well.

No.	Test case/Test script	Attribute and value	Expected result	Result
1.	It is to ensure that quadcopter will move while changes its yaw.	• Altitude • Duration • Velocity-x • Velocity-y • Velocity-z • Yaw_Angle	It successfully moved and changed its yaw with respect to its current Position.	Pass

Unit Testing 7: Checking GPS Based Navigation
Testing Objective: To ensure that quadcopter goes to specified GPS points.

No.	Test case/Test script	Attribute and value	Expected result	Result
1.	It is to ensure that quadcopter goes to specified GPS location.	• Current_GPS_Location • Current_Latitude • Current_Longitude • Destination_Latitude • Destination_Longitude • Altitude.	It goes to the specified location.	Pass

Unit Testing 8: Checking Waypoint Navigation
Testing Objective: To ensure that quadcopter goes to multiple GPS locations.

No.	Test case/Test script	Attribute and value	Expected result	Result
1.	It is to ensure that quadcopter goes to multiple specified GPS locations.	• Current_GPS_Location • Current_Latitude • Current_Longitude • Destination_Latitudes • Destination_Longitudes • Altitude	It successfully navigates to all specified GPS waypoints.	Pass

Unit Testing 9: Generate flight path for 2D Mapping
Testing Objective: To ensure that quadcopter will generate flight path for 2D mapping.

No.	Test case/Test script	Attribute and value	Expected result	Result
1.	It is to ensure that 2D map module will generate a flight path for a specified area whose 2D map is to be generate.	Current_Latitude Current_Longitude Target_Altitude Capture_Points_Per_Lane Number_Of_Lanes Top_Right_Latitude Top_Right_Longitude Top_Left_Latitude Top_Left_Longitude Bottom_Right_Latitude	The System successfully generates flight plan of the area whose 2D map is to be made.	Pass

		Bottom_Right_Longitude Bottom_Left_Latitude Bottom_Left_Longitude		

Unit Testing 10: Note Capture points
Testing Objective: To ensure that quadcopter will note capture points for 2D mapping.

No.	Test case/Test script	Attribute and value	Expected result	Result
1.	It is to ensure that 2D map module will keep capturing point in memory.	• Current_Latitude • Current_Longitude • Target_Altitude • Capture_Points_Per_Lane • No_of_Lanes • Top_right_lat • Top_right_long • Top_left_lat • Top_left_long • Bottom_right_lat • Bottom_right_long • Bottom_left_lat • Bottom_left_long	It goes to destination keep the GPS coordinates of the capture points in memory.	Pass

Unit Testing 11: Generate 2d map.
Testing Objective: To ensure that quadcopter will generate 2D map of specified area.

No.	Test case/Test script	Attribute and value	Expected result	Result
1.	It is to ensure quadcopter will move along the flight path and capture images from capture points and then generate 2D map.	• Flight_Plan	It goes to destination latitude and longitude of the area whose 2D map is to be made and generate its flight path.	Pass

Unit Testing 13: Tracking.
Testing Objective: To ensure that quadcopter is tracking the Target.

No.	Test case/Test script	Attribute and value	Expected result	Result
1.	It is to ensure quadcopter will track the Target and follow it.	• ROI	The quadcopter is tracking and following the Target.	Pass

Unit Testing 14: Obstacle detection.
Testing Objective: To ensure that algorithm is detecting the obstacles correctly.

No.	Test case/Test script	Attribute and value	Expected result	Result
1.	It is to ensure algorithm is detecting the obstacles correctly.	• Window_Size • Number_Of_Disparities • Minimum_Disparities	Algorithm is detecting the obstacles correctly.	Pass

Unit Testing 15: Obstacle avoidance.
Testing Objective: To ensure that quadcopter is avoiding the obstacles.

No.	Test case/Test script	Attribute and value	Expected result	Result
1.	It is to ensure that quadcopter is avoiding the obstacles correctly.	• Disparity_Threshold	Quadcopter is doing it well	Pass

Unit Testing 16: Check Battery status
Testing Objective: To check the battery status.

No.	Test case/Test script	Attribute and value	Expected result	Result
1.	It is to check the battery status.	• Critical_Battery_voltage • Critical_Battery_Level	Battery status is being checked correctly	Pass

Unit Testing 17: Emergency Handling testing
Testing Objective: To ensure that quadcopter is landing safely in case of low battery.

No.	Test case/Test script	Attribute and value	Expected result	Result
1.	It is to ensure that quadcopter is landing safely or return to launch position in case of low battery status if battery is not enough to complete the task.	• Critical_Battery_voltage • Critical_Battery_Level	Quadcopter is performing this task finely.	Pass

6.1.2 Functional Testing

Functional Testing 1: Module: Flight Testing
Objective: To ensure that quadcopter is moving as expected.

No.	Test case/Test script	Attribute and value	Expected result	Result
1.	Upward movement of quadcopter.	• Takeoff_Altitude	Quadcopter is going upward.	Pass
2.	Quadcopter movement toward its North.	• Target_Altitude • Velocity_X	Quadcopter is moving towards its north.	Pass
3.	Quadcopter movement toward its East.	• Target_Altitude • Velocity_Y	Quadcopter is moving towards its East.	Pass
4.	Quadcopter movement toward its South.	• Target_Altitude • Velocity_X	Quadcopter is moving towards its South.	Pass
5.	Quadcopter movement toward its West.	• Target_Altitude • Velocity_Y	Quadcopter is moving towards its West.	Pass
6.	Quadcopter Yaw movement	• Yaw_Angle	Quadcopter is changing its yaw angles	Pass

Functional Testing 2: Module: Emergency Handling
Objective: To ensure that quadcopter is handling itself well in case of emergency.

No.	Test case/Test script	Attribute and value	Expected result	Result
1.	Checking battery status.	• Critical_Battery_Voltage • Critical_Battery_Level	Battery status is checked correctly.	Pass
2.	Return to launch position if battery is not enough to complete the task but is enough to come back.	• Critical_Battery_Voltage • Critical_Battery_Level	Returned to launch position and land safely.	Pass
3.	Emergency Landing in case of low battery	• Critical_Battery_Voltage • Critical_Battery_Level	Quadcopter is landing in	Pass

	and not enough battery to return to launch position.		case of low battery	

Functional Testing 3: Module: Waypoint Navigation
Objective: To ensure that quadcopter is navigating to different GPS locations.

No.	Test case/Test script	Attribute and value	Expected result	Result
1.	Quadcopter take off	• Altitude	Quadcopter take off to specified height.	Pass
2.	Goto function to move quadcopter to different waypoints.	• Target_location	Quadcopter will go to the target	Pass
3.	Get distance in meters as it will check continuously whether quadcopter has reached the target location or not.	• Current_location • Target_location	Distance is being calculated correctly	Pass

Functional Testing 4: Module: 2D Mapping
Objective: To ensure that quadcopter is generating 2D Map of specified area.

No.	Test case/Test script	Attribute and value	Expected result	Result
1.	Create Flight plan for capturing images	• Top_Left_Latitude • Top_Left_Longitude • Bottom_Left_Latitude • Bottom_Left_Longitude • Top_Right_Latitude • Top_Right_Longitude • Bottom_Right_Latitude • Bottom_Right_Longitude • Point_To_Point_Distance • Lane_To_Lane_Distance	Quadcopter will generate flight path for capturing images	Pass
2.	Follow flight path	• Flight_Plan	Quadcopter will follow the flight plan	Pass

Functional Testing 5: Module: Tracking and Navigation
Objective: To ensure that quadcopter is tracking and navigating the target.

No.	Test case/Test script	Attribute and value	Expected result	Result
1.	Quadcopter is tracking the specified person.	• Camera_Feed • Target_ROI	Quadcopter will track the specified person	Pass
2.	To ensure quadcopter is navigating the target	• Velocity_X • Velocity_Y • Velocity_Z • Position_X • Position_Y • Position_Z • Yaw	Quadcopter will follow the target person	Pass

7 Conclusion and Future Work

7.1 Conclusion

The System under discussion has been completed successfully. The User has the ability to assign missions to the Quadcopter, monitor the system's behavior and abort the mission when necessary. The system is capable of navigating autonomously under suitable conditions and carrying out all mission / tasks assigned by the user.

7.2 Future Work

The System is only limited to a single Quadcopter but can be diversified in the future. I plan to work on making the system able to control swarms of quadcopter and also to study more advanced and sophisticated Computer Vision and Artificial Intelligence Algorithms to further increase and flourish the functionalities and possibilities of the system.

8 References

- https://hackernoon.com/quadcopter-physics-explained-468ee44ba40b
- https://www.wired.com/2017/05/the-physics-of-drones/
- https://www.droneomega.com/drone-applications/
- https://www.instructables.com/id/Overview-of-Quadcopter-Components-How-to-Select-Pa/
- http://quadcoptergarage.com/quadcopter-parts-list-what-you-need-to-build-a-diy-quadcopter/
- http://www.quadcopteracademy.com/quadcopter-parts-what-are-they-and-what-do-they-do/
- https://www.gnebehay.com/cmt/
- https://www.opendronemap.org/pyodm/
- https://www.pyimagesearch.com/
- https://www.learnopencv.com/
- http://python.dronekit.io/
- https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-tx2/
- https://devtalk.nvidia.com/default/board/188/jetson-tx2/