COMSATS University Islamabad,
Park Road, Chak Shahzad, Islamabad Pakistan

SOFTWARE DESIGN DESCRIPTION

(SDD DOCUMENT)

for

AUTONOMOUS QUADCOPTER

Version 1.0

By

Mirza Muhammad Bin Adnan Baig **CU/FA15-BCS-048/ISB**

Hamza Ahmad **CU/FA15-BCS-069/ISB**

Muhammad Ussama Irfan **CU/FA15-BCS-088/ISB**

Supervisor

Dr. Tahir Mustafa Madni

Bachelor of Science in Computer Science (2015-2019)

Table of Contents

Revision History

Name	Date	Reason for changes	Version

Application Evaluation History

Comments (by committee) ***include the ones given at scope time both in doc and presentation**	**Action Taken**

Supervised by

Dr. Tahir Mustafa Madni

Signature_____

1. Introduction

The proposed system comprises a set of Computer Vision, Navigation and Controlling based programs necessary for making an autonomous quadcopter. All processing, execution and decision will be made using onboard embedded system namely NVIDIA Jetson TX2. This will enable the quadcopter to process all data onboard and make decisions asynchronously without encountering any delays.

The user will use Mission Ground Control Application to assign any mission to the quadcopter. The user must select a mission from the Mission Menu and then complete the mission requirements to assign the mission to the quadcopter. Mission requirements will vary from mission to mission. For tracking-based mission, the user must select a target from live video feed to be tracked and follow by the quadcopter. For waypoints navigation-based mission, the user must select waypoint by clicking on Google Earth Map at different locations. The user could also specify in which order the waypoints should be traversed and the height to maintained throughout the flight. For mapping missions, the user must select an area from the Google Earth that is to be mapped and the height to be maintained while mapping.

The quadcopter will be able to track an object (specified by user) and autonomously follow it. The quadcopter will be able to autonomously navigate from a starting position to destination (specified by user). The quadcopter will be able to autonomously navigate over an area (specified by the user) according to a generated flight path while automatically capturing images that will then be used to generate a Google Earth like 2D map. The quadcopter will perform all these tasks while detecting any potential obstacles in its flight path and avoiding them when necessary without any external assistance.

The scope that has covered so far with respect to modules is described below:

- Tracking Algorithm has been created, tested and ready to deploy.
- The building of the quadcopter is in progress as some necessary hardware components have not arrived till now.
- A Simulation of the Quadcopter has been created in Unity which will later be used to test navigational and control algorithms that will later be deployed onto the actual quadcopter.
- Work on creating our own and testing open source algorithms for Obstacle Detection is in progress.

The complete list of modules with their description is as follows:

1.1.1 Building the Quadcopter:

This module comprises of building up the quadcopter. A custom quadcopter must be built so that it can accommodate the NVIDIA Jetson TX2 within its frame and be able to fly firmly and steadily under its weight.

1.1.2 Flight Testing the Quadcopter:

This module comprises of a series of flight tests that are performed upon the quadcopter in real-time to ensure that it can perform certain flight modes like Altitude Hold Mode, Loiter Mode and Stabilized Mode which are coded into it. If the quadcopter can perform the above coded modes, then it can be assumed that the quadcopter is aerodynamically stable.

1.1.3 Collision Detection:

This module comprises of testing Computer Vision techniques like Visual SLAM and Stereo Depth Mapping to determine obstacles that are potential candidates for collision.

1.1.4 Collision Avoidance:

This module comprises of testing navigation and control algorithms necessary to avoid obstacles that are potentially capable of collision soon. These algorithms will not only guarantee avoidance from collisions but also maintain stabilized flight during the evasion process.

1.1.5 Tracking and Navigation:

This module comprises of tracking and navigation algorithms. The tracking algorithms will enable the quadcopter to track specific objects, specified by the user, in the live video feed. The navigation algorithms will control the trajectory, speed and angles of the quadcopter to constantly follow the tracked subject. The quadcopter will follow the tracked subject while constantly detecting and avoiding any potential obstacles.

1.1.6 Waypoints Navigation:

This module comprises of a task of navigating the quadcopter from initial point to final point. These points are set using GPS coordinates from maps like Google Earth or Google Maps. The user will be able to select any number of GPS coordinates for the quadcopter for the quadcopter to navigate to or from. The quadcopter will navigate to or from all specified GPS coordinates while constantly detecting and avoiding any potential obstacles.

1.1.7 2D Mapping:

This module comprises of task of mapping an area by taking multiple pictures of the area and stitching them up together to form a Google Earth like map. The user will be able to select the area to be mapped, then the program will generate a flight path and points at which images will be captured that will tend to cover the entire area to be mapped. The quadcopter will automatically takeoff and travel to the starting point. After reaching the starting position the quadcopter will start travelling on the generated flight path while automatically capturing images at generated capture points. After the quadcopter has completed its flight path it will return to the takeoff position and land automatically. The captured images will then be stitched together to form a Google Earth like map.

1.1.8 Mission Ground Control:

This module comprises of a fully featured ground station application for controlling and assigning tasks to the quadcopter. With Mission Ground Control, flight information can be converted for use in Google Earth. Mission Ground Control gives you the ability to view your flight path over the terrain map in Google Earth. Tracking and Waypoint Navigation tasks will be assigned using the Mission Ground Control. The drone can also be programmed to take off and land autonomously and fly using different flight modes using the Mission Ground Control. The user can also program other flight parameters such as speed and altitude of the drone over each waypoint. The Mission specifications will be sent to the drone via telemetry connection.

1.1.9 Emergency Handling:

This module comprises of a set of strategies for handling emergencies. These strategies include:

- Knowing how much battery could be consumed while starting a waypoint navigation to determine whether the drone could successfully complete its mission provided a certain amount of charged battery.
- If a drone is unable to complete a waypoint navigation mission due to a windstorm or 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.

2. Design methodology and software process model

The project underdevelopment makes use of the following design methodologies, software process model and programming paradigm:

2.1 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.

2.2 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.

3. System overview

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.

3.1 Architectural design

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.

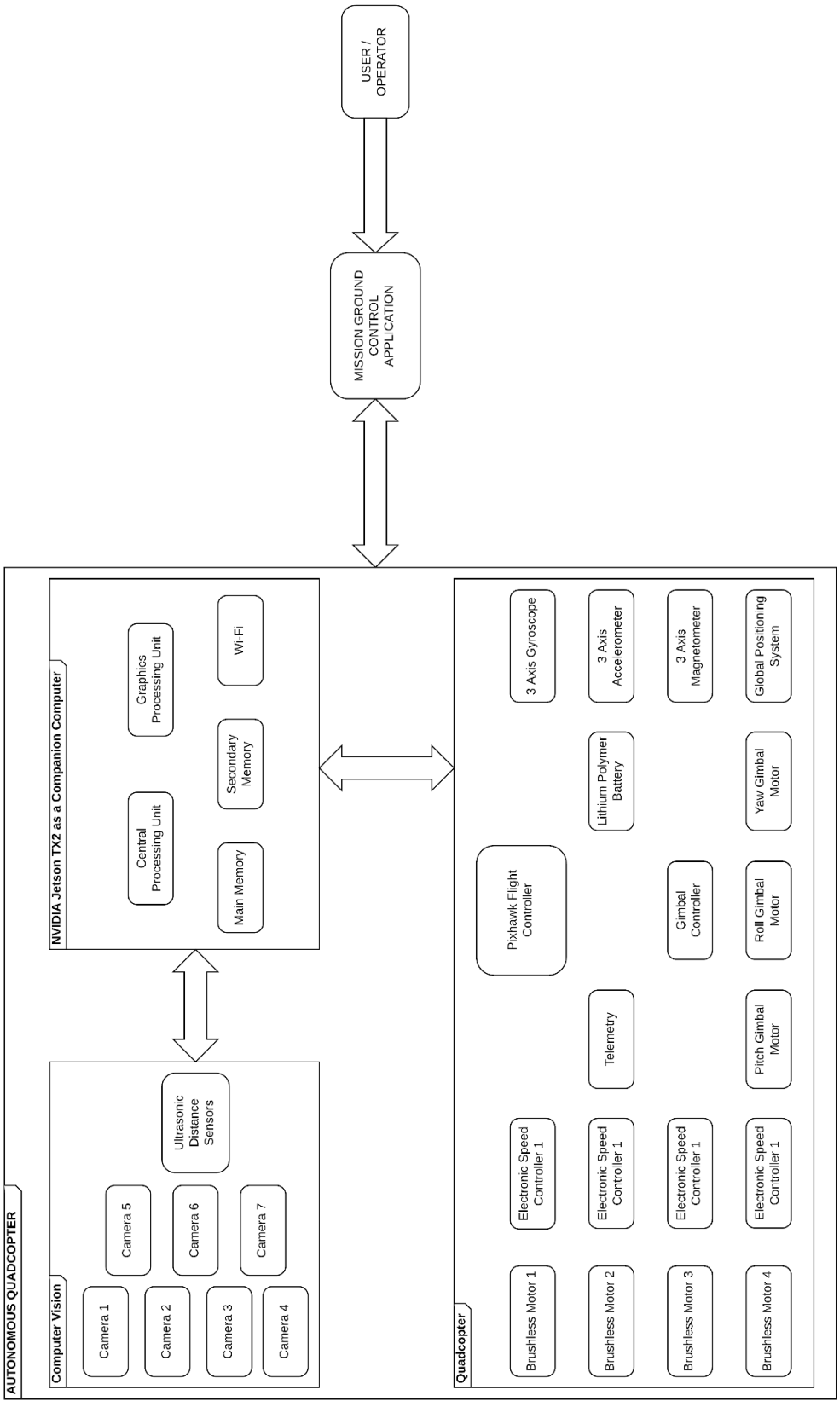

Figure 1 – Architecture Design Diagram

3.2 Process flow

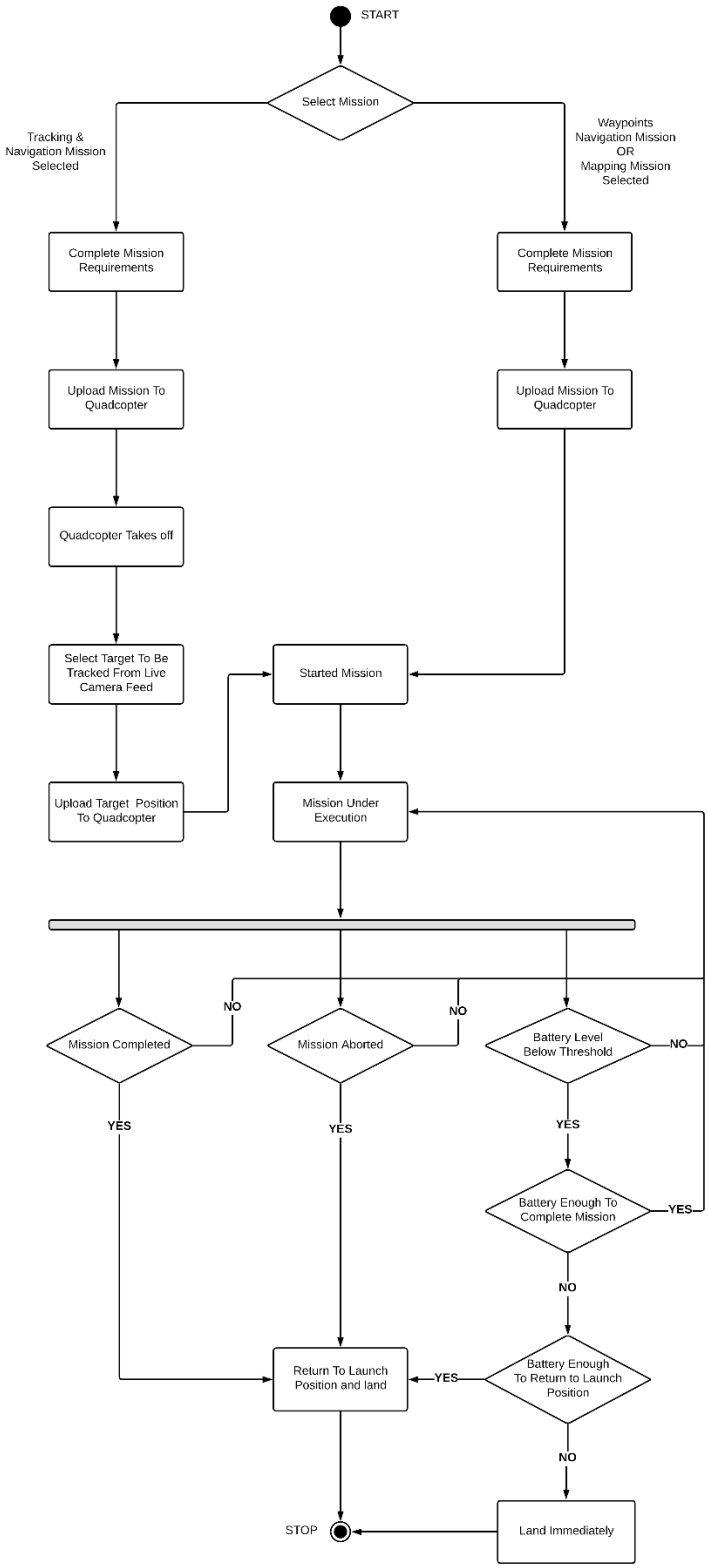

Figure 2 – Activity Diagram

4. Design models [along with descriptions]

Design Model of our system includes State Machine Diagram and Data Flow diagrams.

4.1 State Machine Diagram:

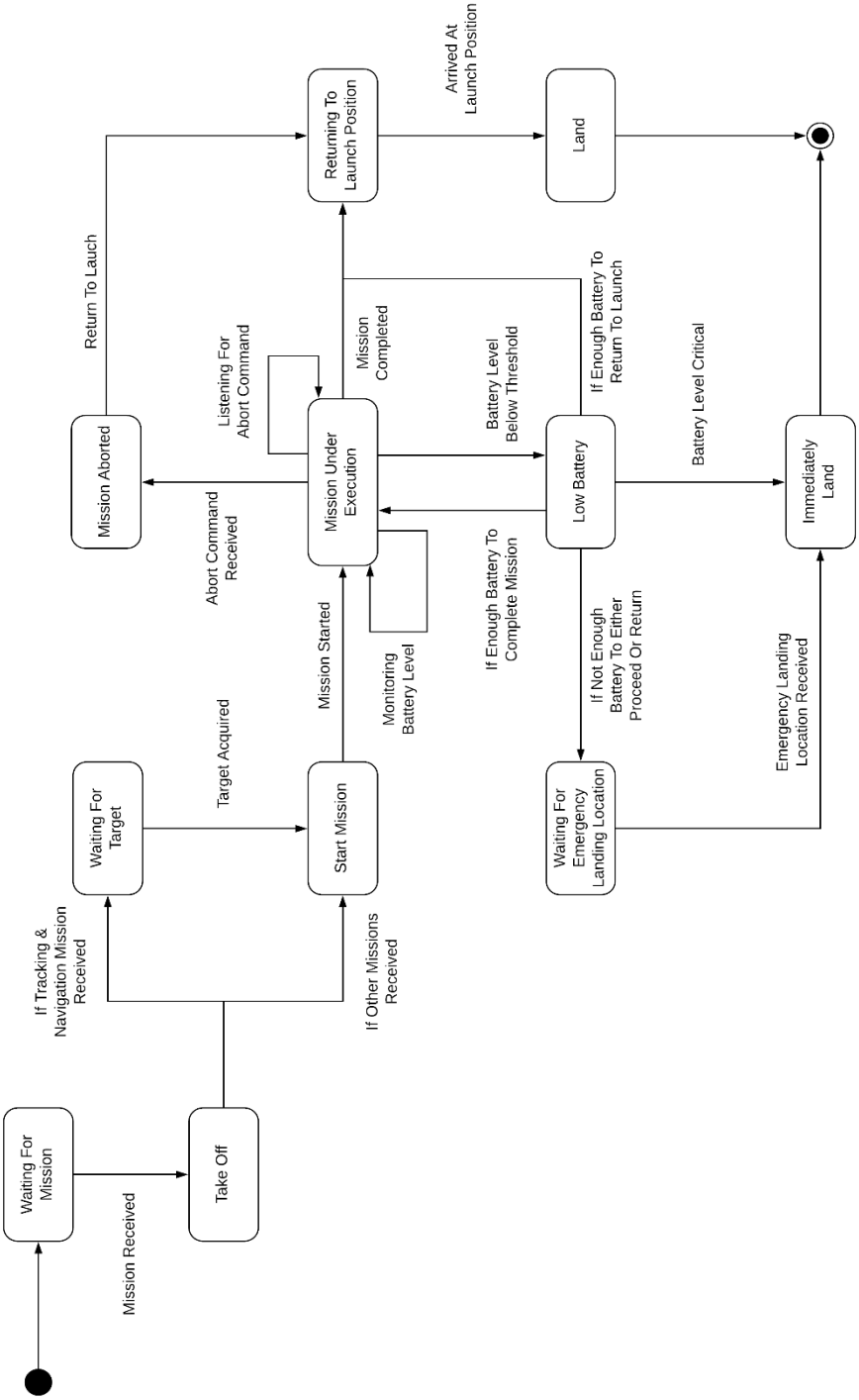

Figure 3 – State Machine Diagram

4.2 Data Flow Diagram:

4.2.1 Data Flow Diagram Level 0:

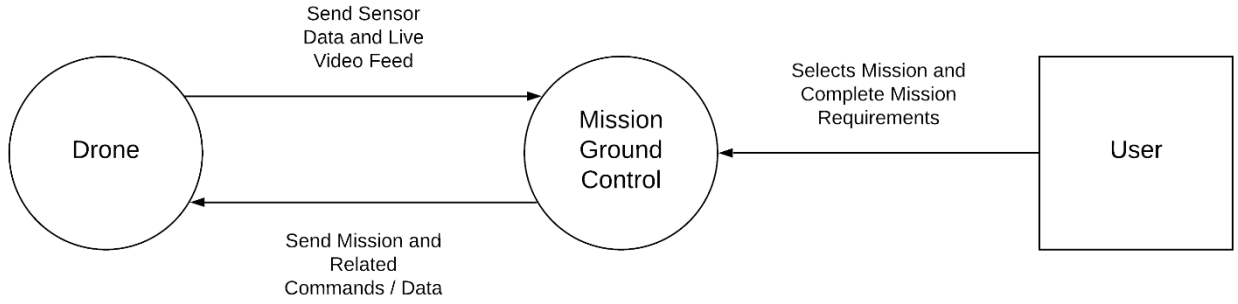

Figure 4 – Data Flow Diagram Level 0

4.2.2 Data Flow Diagram Level 1:

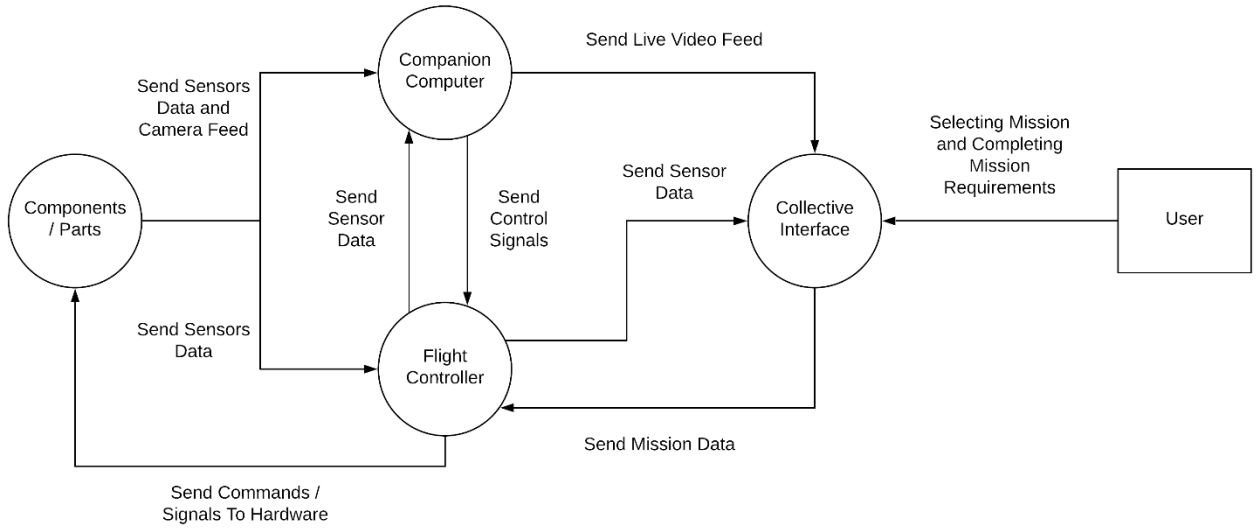

Figure 5 – Data Flow Diagram Level 1

4.2.3 Data Flow Diagram Level 2:

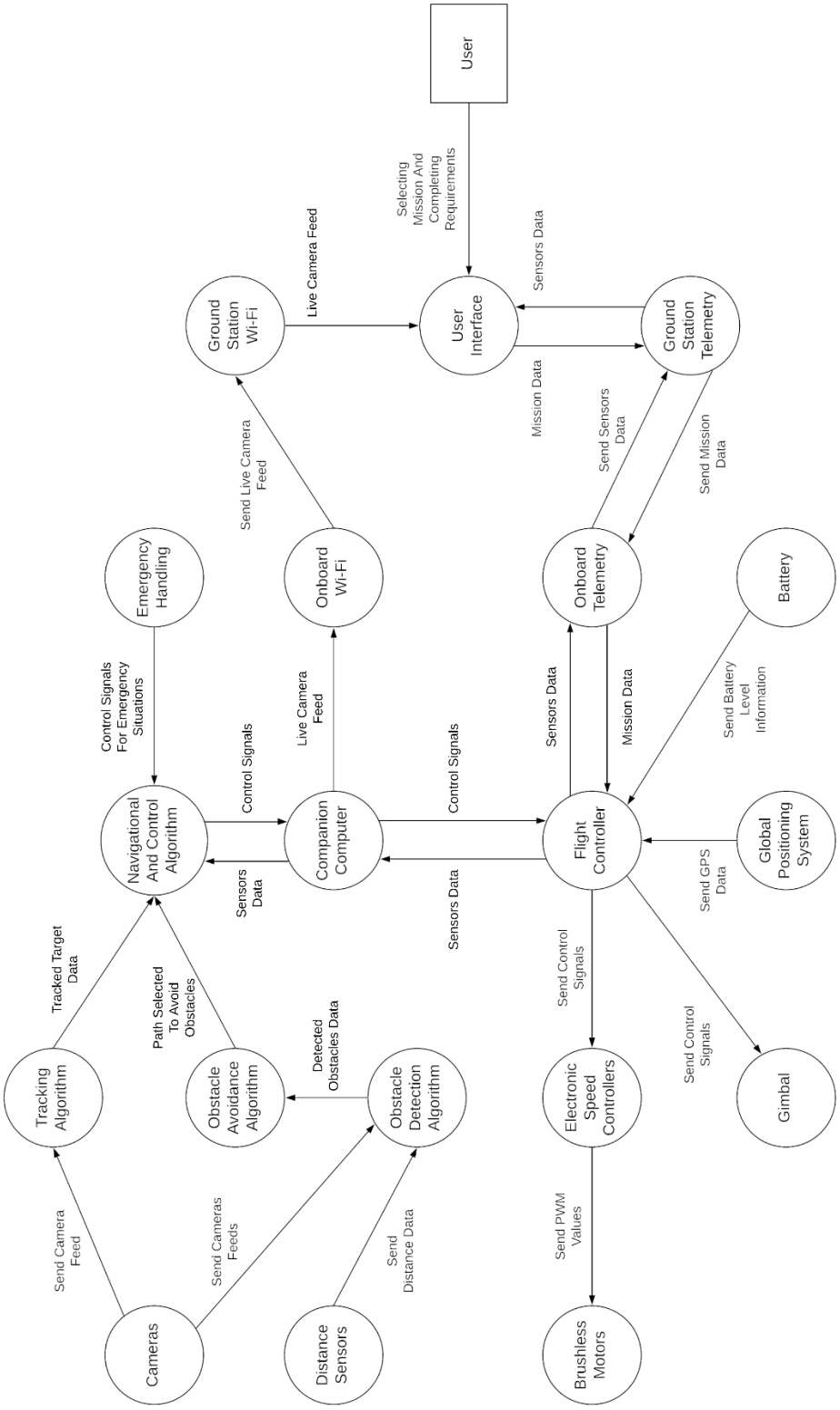

Figure 6 – Data Flow Diagram Level 2

5. Data design

Our System does not incorporate or utilize any form of databases or any other data storage technique. All the data will be generated at runtime and our system will directly use this generated data to perform specified tasks, so there is no need for any data design.

6. Algorithm & Implementation

The following is a list of some of the algorithms that will be used in our project. These algorithms have been studied thoroughly and proved to be useful for our project. Only the Tracking Algorithm has been finalized and the other is subjected to improvement or change in the future. More algorithms for obstacle detection and other modules will be studied and finalized in the near future.

6.1 Clustering of Static-Adaptive Correspondences for Deformable Object Tracking (CMT):

The main idea behind CMT is to break down the object of interest into tiny parts, known as keypoints. In each frame, the algorithm tries to find again the keypoints that were already there in the initial selection of the object of interest. The algorithms do this by employing two different kind of methods. First, it tracks keypoints from the previous frame to the current frame by estimating what is known as its optic flow. Second, it matches keypoints globally by comparing their descriptors. As both of these methods are error-prone, the algorithms also employ a novel way of looking for consensus within the found keypoints by letting each keypoint vote for the object center.

6.2 Semi Global Block Matching Algorithm (SGBM):

Semi Global Block Matching Algorithm is used to calculate disparity. Disparity refers to the difference in location of an object in corresponding two (left and right) images as seen by the left and right eye which is created due to parallax (eyes' horizontal separation). The brain uses this disparity to calculate depth information from the two-dimensional images. In short, the disparity of a pixel is equal to the shift value that leads to minimum sum-of-squared-differences for that pixel.

7. Software requirements traceability matrix

Table 1 - Requirements Traceability Matrix

Req. Number	Ref. Item	Design Component	Component Items
FR01	DFD	DFD Level 2	Tracker
FR02	DFD	DFD Level 2	Navigational & Control Algorithms
FR03	DFD	DFD Level 2	Onboard Wi-Fi
FR04	DFD	DFD Level 2	Onboard Telemetry
FR05	DFD	DFD Level 2	Flight Controller

FR06	DFD	DFD Level 2	Companion Computer
FR07	DFD	DFD Level 2	Obstacle Detection & Avoidance Algorithm
FR08	DFD	DFD Level 2	Companion Computer
FR09	DFD	DFD Level 2	Ground Station Telemetry
FR10	DFD	DFD Level 2	Emergency Handling
FR11	Activity Diagram	Activity Diagram	Mission Aborted

8. Human interface design

8.1 Screen images

Following are the mockups of the Mission Ground Control Application:

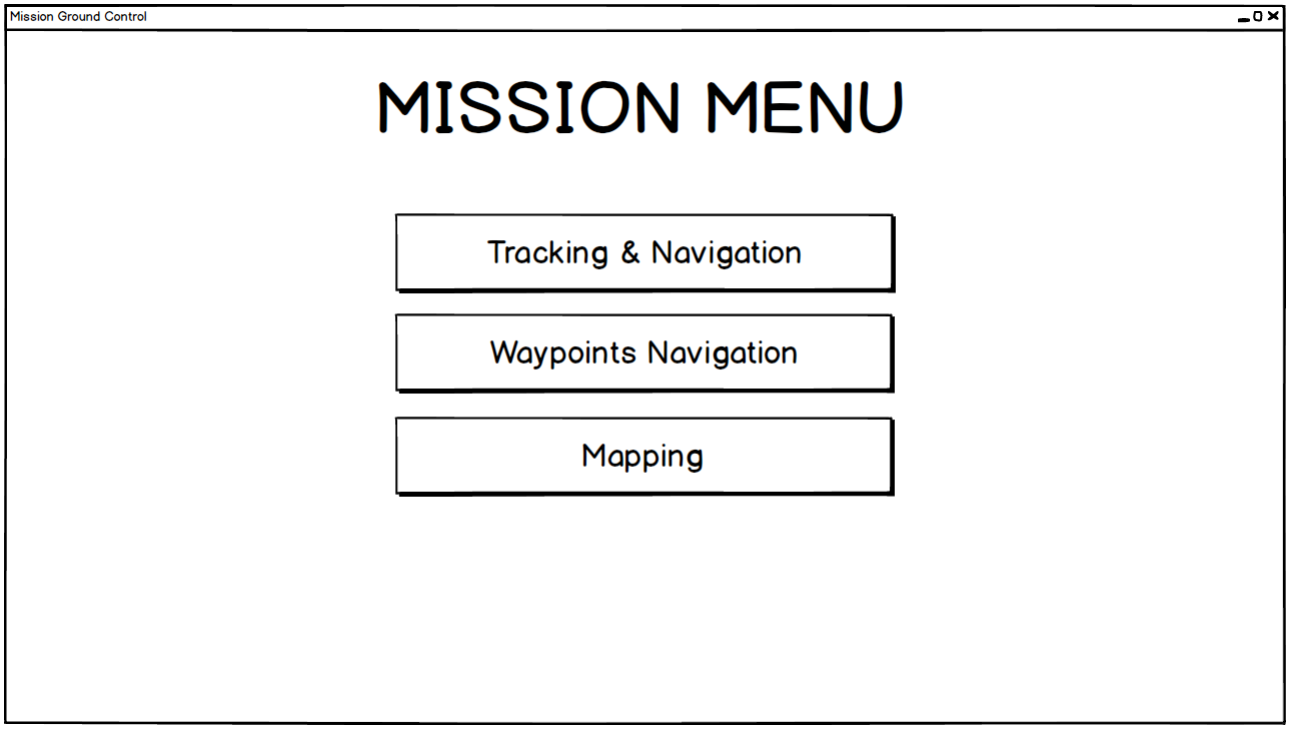

Figure 7 - Mission Menu Screen

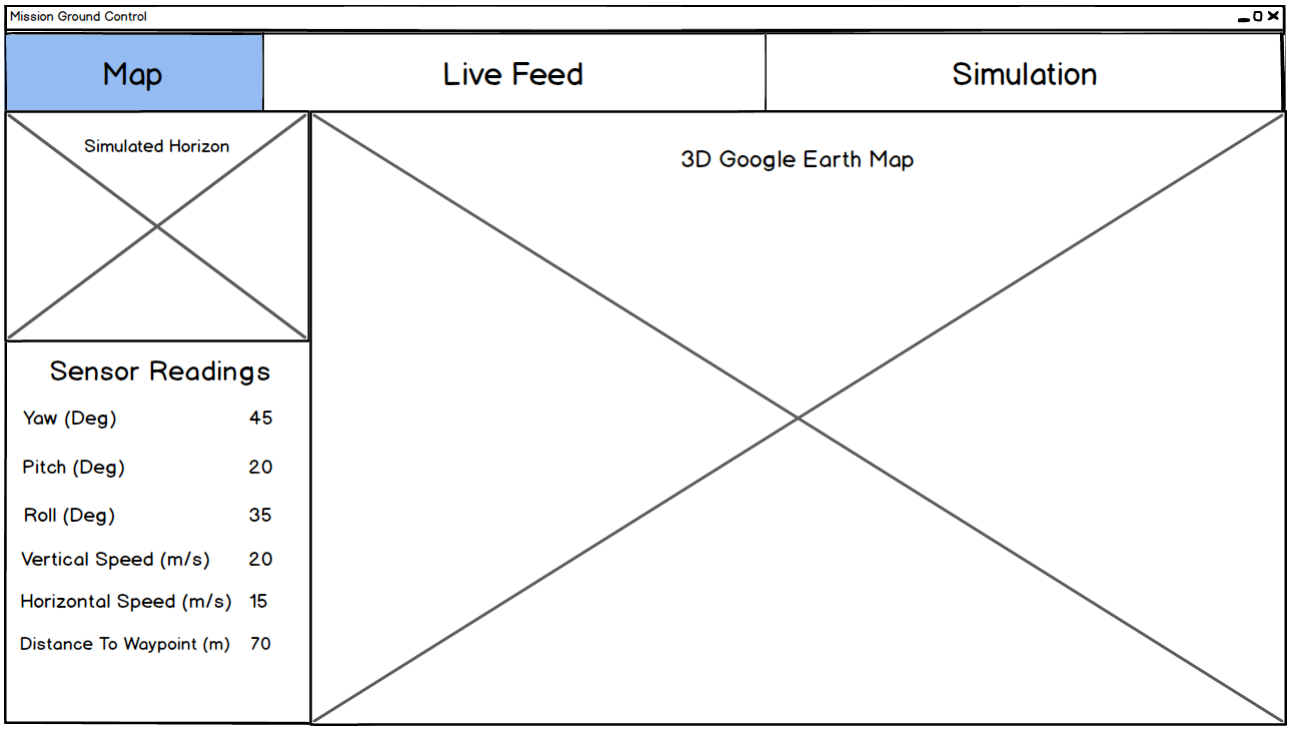

Figure 8 - Map Screen

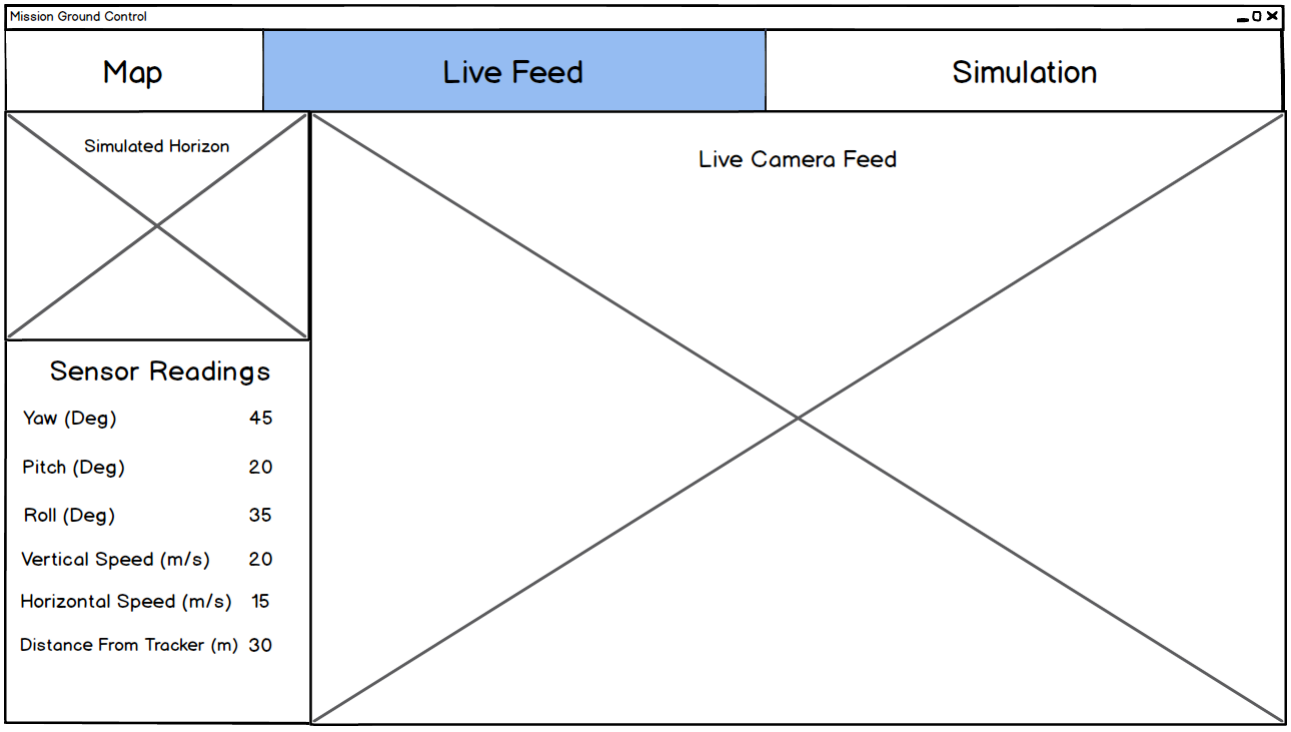

Figure 9 - Live Feed Screen

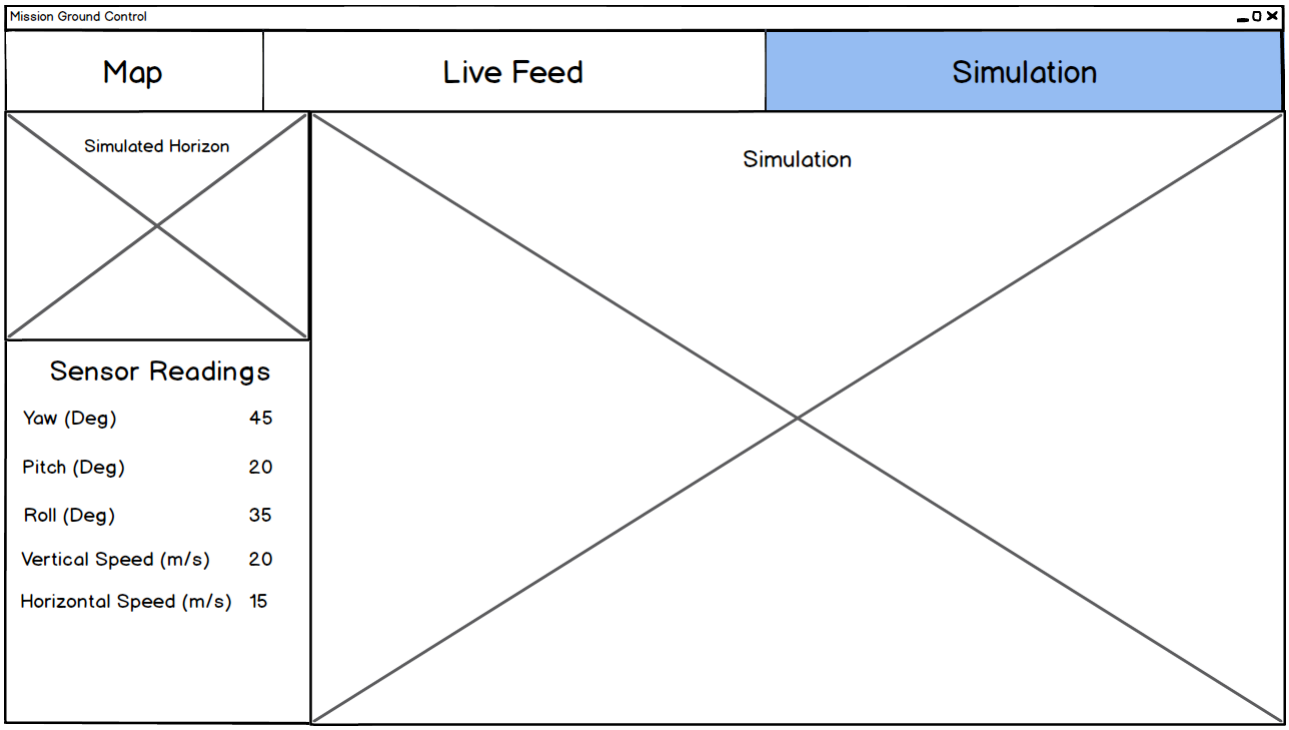

Figure 10 - Simulation Screen