Aerial Pathfinding Reconnaissance

Jason Gilman
Max Griffith
Mark Johnson
Dilanka Weerasinghe

INTERFACE CONTROL DOCUMENT

REVISION 1 – Draft
20 September 2020

INTERFACE CONTROL DOCUMENT
FOR
Aerial Pathfinding Reconnaissance

PREPARED BY:

Jason Gilman Date

Max Griffith Date

Mark Johnson Date

Dilanka Weerasinghe Date

APPROVED BY:

Jason Gilman Date

Prof. S. Kalafatis Date

T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
1	9/20/2020	Jason Gilman		Draft Release

Table of Contents

List of Figures

1. Overview

This document will provide specifics into how the Aerial Pathfinding Reconnaissance System (APRS) will be implemented. It will also describe the APRS subsystems physically and electrically, providing information on how the system will function. The document will define the interaction between the subsystems, and the requirements to enable functionality.

2. Definitions

2.1. Definitions

REST	Representational State Transfer
APRS	Aerial Pathfinding Reconnaissance System
CV	Computer Vision

3. Physical Interface

3.1. Weight

3.1.1. Weight of Aerial Drone

The aerial drone system will weigh less than 3680g. This is to allow an average person to be able to transport and setup the drone. This weight includes the drone chassis, sensors, battery, power distribution components, and microcontroller.

3.1.2. Weight of Land Drone

The land drone system will weigh less than 6.1kg. The land drone system will need to be placed at the user's defined start point, so the land drone's weight was defined so that a user could transport and setup the drone, similar to the aerial drone. The weight includes all necessary parts for the land drone's functionality.

3.2. Dimensions

3.2.1. Dimension of Aerial Drone

The sensors, battery, power distribution components, and microcontroller onboard the drone will fit within the footprint of the drone's chassis. The drone has a diagonal wheelbase of 66.04cm with a vertical footprint of the drone is 38.1cm.

3.2.2. Dimension of Land Drone

The land drone's chassis will have motors, wheels, a motor controller, and a microcontroller mounted to it. The land drone can have a volume ranging from 0.03m^3 to 0.07m3.

3.3. System Setup Locations

The following section will define the requirements for where each system will need to be placed in order for the Aerial Pathfinding Reconnaissance System to function properly.

3.3.1. Placement of Aerial Drone

The aerial drone must be placed on a level surface with no obstructing objects within 0.5m. of the zone. The drone must be placed such that a wireless local area network connection hosted on the drone can be connected to by the local machine running the system's application. This placement will be used for takeoff and landing of the aerial drone, as well as transmission of data to the local machine.

3.3.2. Placement of Land Drone

The land drone must be placed such that a wireless local area network connection hosted on the land drone can be connected to by the local machine. The land drone should not be placed on a surface with a slope greater than 30 degrees or 58% grade. This placement will be used as the drone's start point.

3.3.3. Placement of the Local Machine

The user’s local machine must be in proximity to the aerial and land drones initial locations such that the machine can connect to wireless local area networks hosted by each machine.

4. Electrical Interface

4.1. Primary Input Power

4.1.1. Aerial Drone

The standard power voltage provided to quadcopters and drones is 22.2V. The drone will be using a single 22.2V lithium ion battery with a capacity of 5000maH. A lithium ion battery was chosen due to its weight and efficiency in comparison to other battery types such as lead acid. A single one of these batteries will power the aerial drone in normal operation and for longer flights it may take 2 of these batteries in parallel. Below, in Figure 1, is a graph detailing the voltage drop off of a lithium battery versus a Lead-Acid.

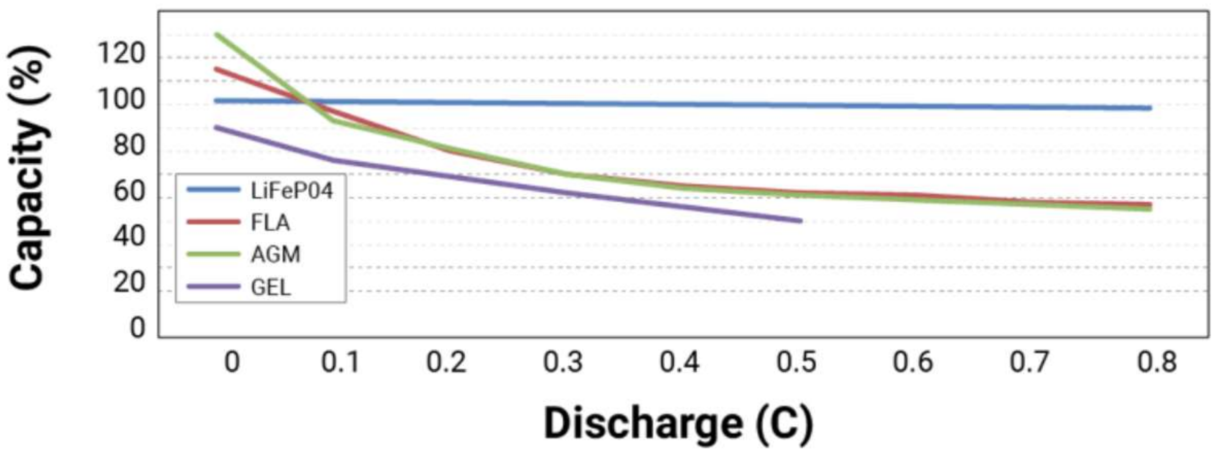

Figure 1. Lithium-Ion vs Common Lead Acid

The batteries will be attached to the drone with a custom fitting bracket that will securely hold the batteries during flight. The batteries must be fitted correctly to ensure no electrical malfunction occurs in the system.

4.1.2. Aerial Drone

The land drone will be using a high capacity lithium ion battery. To see why a lithium ion battery was chosen, see the above description on the benefits of lithium ion vs other battery types. Because of the varying power inputs on the digital controllers attached to this drone a series of boost and buck converters will be used in addition to the 4.2V battery. The batteries will be attached to the drone using a custom fitting bracket that will securely hold the batteries during land travel.

4.2. Signal Interfaces

4.2.1. Aerial Drone Connection

The onboard SDK for the Tarot 650 V2.1 utilizes serial ports on the PCB to communicate with the drone controller. The drone controller has high level methods to communicate with the telemetry module and flight controller. The microprocessor will issue commands to either the TTL UART port or the USB port. Several broadcasts can be received from the drone. Due to the large amount of data generated by the drone this application can record the data to any controller at the users need. The drone can offload tasks using the OSDK to the controller to perform our autonomous flight and data retrieval tasks.

4.2.2. Aerial Drone µC

In accordance with the requirements of the Tarot a handful of controllers can be utilized. A stm32 or Arduino can send the required information back and forth from the drone and to the server. The stm32 has a faster processor and thus can handle more tasks. Because of the low frequency of the broadcasts incoming from the drone either controller will fulfill the project goals. The microcontroller will run the Linux operating system to communicate with the drone. The drone can also be directed with a Mission Controller Program without any object avoidance application.

4.2.3. Land Drone Connection

This drone will be using a stm32 or Arduino to handle control of the motor controller and other system connections. The land drone will utilize a sabretooth (ii) motor controller to handle power distribution and control of the motors. The Land Drone will have a wireless connection to the user's local machine using a WIFI module further outlined in Section 6.

4.3. Sensors

4.3.1. Aerial Drone Sensors

4.3.1.1. Camera

A camera will be used by the µC to record video of the ground. This camera will be directed straight down and will not have any connection to the flight controller on the drone. Only to the 3rd party controller (stm32/Arduino).

4.3.1.2. Ultrasonic / Lidar

Attached to the front of the drone will be a series of ultrasonic or lidar sensors to handle object detection and perform anti-collision movements. Because the drone will only be performing small movements in accordance with telemetry data gathered it will only travel forward and rotate. Therefore, we do not need 360 vision, instead the drone will see the immediate area in front and maneuver around obstacles slowly.

4.3.1.3. GPS

The flight controller will collect telemetry data such as GPS. The highly accurate GPS location data is what is used to send the drone to specific locations. The µC will send coordinates to the flight controller to follow.

4.3.2. Land Drone Sensors

4.3.2.1. GPS

The µC will collect telemetry data such as GPS. The drone will use the GPS to determine its location in accordance with the map provided and relay this to the user.

4.4. User Control Interface

4.4.1. Physical Interface

Both the land and aerial drones will indicate to the user the effective battery power and ready state. Both systems will include an indication light for powered on and system ready. The system ready indicator will show the user that it is prepared for remote instruction and operation.

5. Software Interface

5.1. Computer Vision and Mapping

5.1.1. Inputs

The Computer Vision and Mapping subsystem will take one or more videos, in MP4 format. For each field-of-view of the ground along the path, there are expected to be two clips which hold the following properties:

- The length of these videos should be long enough for a computer vision (CV) algorithm to have enough visual information to eliminate moving objects from detection.
    - For example, to ignore pedestrians, the clips' length should be at minimum the time it takes for a pedestrian to move a few feet from their original position.
- These clips should be taken sufficiently far apart (depending on the drone's altitude) to provide strong enough visual cues to infer depth with a second CV algorithm.
    - For example, a drone at 10m altitude will likely want to record their two clips at least 1m away from each other for improved depth estimation.
- Each clip's metadata should include a timestamp so that the correct sequence of video inputs can be determined.

Furthermore, the drone's flight altitude and land vehicle's wheel radius must be known to determine the scale at which obstacles are to be detected.

5.1.2. Error Conditions

The Computer Vision and Mapping subsystem will report an error through standard Python exceptions when a top-down map of the environment cannot be produced.

5.1.3. Outputs

The Computer Vision and Mapping subsystem will output a two-dimensional Boolean matrix representing the locations of obstacles, with each element representing approximately one pixel of the final top-down image of the environment. In this case, the term "obstacle" refers to any region which presents too great of a change in depth, for which a path through that region is not possible for the land vehicle.

The Computer Vision and Mapping subsystem will also output a top-down color image of the surveyed environment intended for use in detecting the starting location of the ground vehicle.

5.2. Pathfinding and Ground Vehicle Detection

5.2.1. Inputs

The input of this system will be the two-dimensional Boolean matrix that represents the location of obstacles as outlined above.

5.2.2. Error Conditions

An error occurs if and only if no possible path is discovered from the Boolean map. The pathfinding and ground vehicle subsystem will report the error though standard Python exceptions and throw the exception to the user interface subsystem.

5.2.3. Outputs

There will be two outputs of this system: one pertaining to the map and the other pertaining to the drone.

The Boolean map will be changed throughout a greedy algorithm. The greedy algorithm will work as follows:

- With respect to a final path heading from South to North, the greedy algorithm will choose the straightest path that is valid. Once a northern path is deemed non-traversable, the drone will select a west path and repeat this process. If neither are deemed traversable, the drone will select an east path and repeat the process.

- If anytime during this calculation a dead end is spotted, the algorithm will mark the current index as false and travel back to the previous position and start the algorithm again. Once the end index is located, the algorithm will be complete, and the path will be sent to the drone.

- If there is no valid path, the Boolean map will be marked as entirely false by the algorithm itself.

The goal of this greedy algorithm is to find the shortest locally optimal path in a quick manner.

The second output will be the path of the drone. Once this new Boolean map is sent to the drone, the drone will move to the end position following the map that has been sent if there is a possible path. If no possible path is found, the error condition is met, and the drone will respond accordingly.

6. Communications / Device Interface Protocols

6.1. Wireless Communications

A combination of communication through wireless local area networks (WLANs) and the internet will be used. Communication between the user's local machine and the drones will be through the use of a WLANs. If the user meets the requirement of internet access on their local machine and wishes to offload processing to a GPU accelerated server, the internet will be used to send information from the user's machine to the server. All wireless communication will follow IEEE specification 802.11.

6.2. Host Device

A local machine, such as a laptop, provided by the user will serve as the host device for the system's desktop application. This device will allow for networking throughout the system. Optionally, the system's software analysis will occur locally or on a GPU accelerated server.

6.3. REST Architecture Constraints

The system will implement a RESTful API so that software can be utilized within the systems operation. GET and POST application state transitions will be used to send and receive data throughout the system.