

The AgroFlight Soil Sensing Drone

A Technical Report submitted to the Department of Electrical and Computer Engineering

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

Anthony Zaitsev
Spring Semester 2022

Technical Project Team Members


Anubhav Acharya

Sean Benish

Zach Khan

Gabriel Mallari

On my honor as a University Student, I have neither given nor received
unauthorized aid on this assignment as defined by the Honor Guidelines
for Thesis-Related Assignments

Signature  Date May 8, 2022
Anthony Zaitsev

Approved  Date May 12, 2022
Technical Advisor, Harry Powell, Department of Electrical and Computer Engineering

Statement of Work

Zach Khan

Zach's primary responsibilities were to design the printed circuit board (PCB) for the flight computer and to select drone hardware. He used online part checklists to find necessary components like propellers, motors, electronic speed controllers, flight controllers, power distribution boards, batteries, and a frame that was used as the baseline. His parts research verified compatibility, like picking motors that both wouldn't be stressed by the power needed for the propellers and couldn't draw more power than the battery could supply. Mass and power calculations were used to project flight times and thrust to aid soil sensor injection.

The PCB design process started with compiling components required for the drone not already found in the drone hardware. Zach selected components like a distance sensor, soil sensor, and MOSFETs for soil sensor actuation. After working with Anthony to make a hierarchical schematic of the entire flight computer, they worked to find real world parts and linked their footprints to components. He performed board layout constrained by the size of the MCU development board, established a bill-of-materials (BOM) for all of the parts, and ordered them from suppliers who had the right parts in stock. He tested both revisions of the board, and corrected mistakes from the buck converter, serial communication lines, and solenoids.

Sean Benish

Sean focused on developing the software and drivers necessary to make communication between the controller input and the flight controller possible. In the desktop realm, he developed the library implemented by the GUI that reads data from any configured USB controller as well as the two-way communication between the ground station and the drone via the Simple Drone Communication Protocol (SUMD), also designed by him. In the embedded section, he selected components desirable for the Wi-Fi module, GPS module, antennas, and microcontroller. Once selected, he devised methods for communicating between the microcontroller and Wi-Fi module as well as the microcontroller and the flight controller, which he would later implement in software. Furthermore, Sean helped bring the other developers up to speed on using TI-RTOS and became proficient in its operation. Lastly, the development of the soil sensor drivers, solenoid drivers, and testing of all of the microcontroller's tasks were completed by him.

Gabriel Mallari

Gabriel's primary responsibility was implementing the graphical user interface (GUI) for the ground control station application. The main purpose of the application is to display and send GPS, soil nutrition, altitude, and battery life data back and forth between the application and the drone. A worker thread class was implemented to enable the app to continuously request data from the drone. The majority of the semester was spent developing the application. It was programmed in C++ using the QT framework and QML. Having no experience with this framework and unique language, a majority of the semester was spent learning and understanding the language's syntax and libraries. Both tools were user friendly, heavily documented, and had an extensive community which aided in the development process. Gabriel

was also responsible for registering the drone and obtaining approval for Part 107 flight for testing.

Anubhav Acharya

Over the course of the semester, Anubhav focused mainly on software and hardware assembly for the drone. In software, he integrated the GPS Module into TI RTOS. He wrote the task that would read in data from the GPS Module and store it in a buffer. He also wrote the task that would take the read data, which is a combination of 8 different sentences, and then go through it extracting and storing information into a data structure. This extracted data had many different useful values like the latitude, the longitude, ground speed, time, date, etc. and was then later sent to the ground control station over Wi-Fi. In terms of software, Anubhav also configured the flight controller telemetry through Betaflight Configurator while also analyzing the protocol used for telemetry. This analysis was useful when decoding the telemetry data within the flight computer. The telemetry used is called Light Telemetry (LTM).

In terms of hardware selection, the research into launchpad selection was also done by Anubhav. He analyzed multiple different devkits like the ST Electronics Nucleo boards, several TI MSP and Tiva-C boards for relevant hardware specifications and price. The team ended up using the TI Tiva-C devkit which has 256 kilobytes of flash memory and 32 kilobytes of SRAM. Anubhav, also did some 3d design work in Fusion 360. He redesigned the top and bottom plates and then printed them using a 3D printer. He also later worked with Anthony on assembling the parts to build the drone. Most of the work done by Anubhav was on the software side, TI RTOS, and 3d design and assembly.

Anthony Zaitsev

Anthony's work focused on hardware planning, board design, and design of the physical frame. He worked with Zach to select drone parts that would optimize thrust to carry the load of the soil sensor at a reduced cost. At the start of the project, he was involved in researching hardware and firmware solutions to design the mechanical architecture. For the PCB design, Anthony was primarily responsible for designing the communication and configuration path of the WF121 Wi-Fi module as well as understanding the configuration process. He worked closely with Zach to design the circuitry for the solenoid and the soil sensor, but was especially involved in PCB design. Anthony foot printed parts, aided in layout, and helped optimize trace lengths. Anthony and Zach worked together to pick components and connectors to place on the board. After the board was tested, Anthony's efforts were concerted towards designing the frame and landing gear for the drone. He spent lots of time iteratively modeling and 3D printing parts for the frame to ensure the drone could land without damaging the soil sensor, but also embed the sensor into the ground. Anthony worked closely with Anu to 3D print parts and hand assemble the frame.

Table of Contents

Statement of Work	2
Table of Figures	6
Table of Tables	7
Abstract	8
Background	8
Project Selection	8
Similar Products	8
Project Novelty	9
Previous Coursework	9
Constraints	10
Design Constraints	10
CPU Limitations	10
Software Availability	10
Manufacturing Limitations	10
Economic and Cost Constraints	11
External Standards	12
Barr C	12
NEMA	12
RoHS	12
IPC	12
I2C	12
Soil Sensor Communication	13
Legal	15
Tools Employed	16
Printed Circuit Board	16
Drone Frame	16
Embedded Software	17
GUI Software	17
Ethical, Social, and Economic Concerns	17
Environmental Impact and Sustainability	17
Health and Safety	18
Ethical Issues	18

Intellectual Property Issues	19
Similar Patents	19
Drone Frame	19
Detailed Technical Description of Project.....	20
The Wi-Fi Module	20
Chip Selection	21
Hardware Implementation	21
Drone Flight Computer Software Interface	21
The GPS Module.....	21
Hardware Implementation	23
Drone Flight Computer Software Interface	23
The Soil NPK Sensor	23
Sensor Selection.....	24
Frame and the Landing System.....	24
Preliminary Decisions and Workflow.....	24
Top Plate and Bottom Plate	25
First Tier.....	25
Second Tier	26
Landing Gear	26
The Drone Flight Computer.....	32
Launchpad Selection.....	33
TI RTOS.....	33
Hardware Design of the Flight Computer Expansion Board	41
Graphical User Interface Application	52
GUI Stack Research.....	52
GUI Application Development	52
Wi-Fi Drone Interface Library	55
Interface	55
SDCP.....	56
Helper Programs	58
The Drone Flight Controller	58
Software Suite for the Controller.....	59
Betaflight.....	59
The Drone Hardware.....	60

Project Timeline.....	61
Test Plan.....	64
Hardware.....	64
Software	64
Final Results.....	68
Costs.....	70
Future Work	71
References	73
Appendix.....	80
Appendix 1: Parts Ordered.....	80
Appendix 2: Drone Parts Unit and Bulk Cost.....	86
Appendix 3: PCB Test Plan	89
Appendix 4: Drone Certificate of Registration	90

Table of Figures

Figure 1: Soil Sensor Modbus Inquiry and Response Frame Structure.....	13
Figure 2: GPS Module Sentences, In Order.....	22
Figure 3: S500 Drone Frame Chosen as Starting Design	24
Figure 4: First Frame Tier.....	26
Figure 5: Second Frame Tier	26
Figure 6: Soil Sensor Mount.....	27
Figure 7: Solenoid Holder.....	28
Figure 8: Sliding Leg with Feet	29
Figure 9: Fixed Leg with Bottom Plate Mount.....	30
Figure 10: Landing Gear in Extended Position.....	31
Figure 11: Landing Gear in Contracted Position	31
Figure 12: Physical Assembly of Landing Gear	32
Figure 13: Embedded Software Overview.....	34
Figure 14: Connection Layer Overview	34
Figure 15: BGAPI Packet Format.....	35
Figure 16: Connection Layer State Machine	36
Figure 17: Telemetry Layer Overview	37
Figure 18: LTM Attitude Frame Format.....	37
Figure 19: LTM Status Frame Format	38
Figure 20: Sampling Layer Overview.....	39
Figure 21: Soil Sensor Sampling Request Packet.....	40
Figure 22: Soil Sensor Sampling Response Packet	40
Figure 23: Flight Director Overview	41

Figure 24: Flight Computer Expansion Board Ultiboard Layout	42
Figure 25: Recommended Buck Converter Circuitry	43
Figure 26: Buck Converter Schematic	43
Figure 27: WF121-E-V2C Wi-Fi Module	44
Figure 28: WF121 Circuit Implementation.....	44
Figure 29: ICSP Header for WF121 Configuration.....	45
Figure 30: Physical Dimensions of WF121: WF121 Physical Specification	46
Figure 31: GN-8720 Circuit Implementation	46
Figure 32: Prohibition Area for GN-8720	47
Figure 33: MCU Connections on Schematic	48
Figure 34: RS-485 Half-Duplex Transceiver Design	49
Figure 35: Solenoid Circuitry	51
Figure 36: GUI Workflow	53
Figure 37: Drone Application GUI.....	54
Figure 38: SDCP Communication Overview	56
Figure 39: SDCP Communication Overview	57
Figure 40: Xcopter Calculator for Drone Hardware	60
Figure 41: Preliminary Gantt Chart	61
Figure 42: Finalized Gantt Chart	62
Figure 43: Team Member Responsibilities.....	63
Figure 44: Test Plan – Wi-Fi Module	65
Figure 45: Test Plan – SDCP	65
Figure 46: Test Plan – Flight Controller	66
Figure 47: Test Plan – Telemetry.....	66
Figure 48: Test Plan – GPS Module	67
Figure 49: Test Plan – Soil Sensor.....	67
Figure 50: Final Drone Assembly.....	69

Table of Tables

Table 1: Soil Sensor MODBUS Addresses	39
Table 2: SUMD Packet	40
Table 3: RTOS Actions and Priorities	41
Table 4: Summary of Wi-Fi Communication Ports.....	45
Table 5: Summary of GPS Communication Ports	47
Table 6: Summary of Component Connections to MCU.....	48
Table 7: Summary of RS-485 Communication Ports	50
Table 8: Summary of Solenoid Communication Ports	51
Table 9: OP_JOY Packet	57
Table 10: OP_TEL Packet	58

Abstract

The AgroFlight is a drone system capable of collecting soil nutrition data at remote coordinates and wirelessly returning the data to a ground control station for analysis. AgroFlight is split into three primary functional systems – the flight and probe system, the flight computer, and the ground station. The flight and probe system entails the physical drone. It flies using hardware optimized for stable, long-distance flight as well as a landing gear to deploy a soil sensor at an operator's will. The flight computer is a logic device onboard the drone that commands flight instructions as well as collects data to be sent to the ground station. The ground station is a graphical interface used to control drone flight as well as display soil data and flight information. AgroFlight is capable of performing regular and repeatable soil sensing flights to create a map of soil nutrition over a large land area. This makes AgroFlight an ideal product for farmers interested in tracking their soil nutrition to make better resource management decisions.

Background

Project Selection

In the agricultural industry, farmers regularly collect soil samples to test a field's nutritional content to manage fertilization and predict crop yield. The land management decisions made from the soil analysis significantly influence farmers' use of fertilizer resources and crop planning [1], both of which influence incurred costs and profit. Though traditionally done only once every few years, tighter margins across the industry are pushing some farmers to conduct more regular soil tests to optimize their crop production. For large-scale farms, this can be a difficult task; tests are currently collected at a rate of anywhere from 1 sample per 20 acres of land down to 1 sample per 2 acres [2]. This means a very large farm 2000 acres in size would have to collect anywhere from 100 to 1000 samples.

AgroFlight is the most effective way to accomplish large-scale semi-automated soil testing. Drones are a cost-effective measure to traverse many acres of land in a short period of time and have the potential to conduct timely and regular tests without much effort by the user. A drone is ideal for a farm setting since there are few obstacles that could pose danger to flight as well as obscure GPS signals for flight navigation. Automating flight navigation also has the advantage of collecting a large quantity of datapoints for this application.

The need for a product to perform automated soil testing stems from the industry trend of using new tech to optimize farming. Farmers are already utilizing technologies for real-time monitoring of moisture, smart irrigation, and even self-driving GPS tractors to farm more efficiently. Streamlining soil testing would be another step towards sustainability by enabling a higher level of precision.

Similar Products

Soil testing is traditionally conducted by manually digging samples of soil and sending the specimens to a lab for analysis. The process is laborious and time consuming. Furthermore, bookkeeping the samples is another challenge since each sample has to be individually bagged

and labeled. AgroFlight automatically tags a soil test with the GPS coordinates giving farmers instant access to a map of their soil nutrition. Though lab samples can test for a more comprehensive list of nutrients, the most important influences for crop yield are nitrogen, phosphorus, and potassium. AgroFlight is equipped with an electronic soil sensor that can collect these exact nutrients. While AgroFlight isn't intended to completely replace the traditional method of testing, it gives farmers another tool to aid in land-management that can more rapidly track nutritional changes.

Automated drone flight has been implemented in many industries. The most notable use of automated drone flight can be found from companies that offer surveillance services. Drones created by Airobotics are deployed in worksites and perform autonomous flight for functions such as surveying and mapping, infrastructure inspection, and stockpile management [3]. These drones implement a variety of sophisticated sensors to perform their surveying functions. Drone technology has also been implemented in agriculture. The UAV industry leading company, DJI, sells a drone product that irrigates a field with fertilizer [4]. It can hold and spray 30L of liquid around the entire perimeter of the vehicle.

Project Novelty

Few drone products with automated capabilities are currently on the market that implement external systems to interact with their environment; most focus exclusively on ariel imagery [5]. Common drone products are deployed on a fixed route then returned to home base without ever landing. AgroFlight is unique in that landing away from home base is a necessary part of its mission. Take-off and landing algorithms must therefore be implemented in the flight computer.

Though the agricultural industry is also seeing a rise in drone usage, most products are similarly centered around imagery analysis [6]. Drones that do perform a physical task, such as those that plant seeds and spray crops. AgroFlight will be one of the only agricultural products that collect data through direct interaction with its environment.

Previous Coursework

This project will draw from experience from several core and elective courses in the electrical and computer engineering (ECE) department. ECE Fundamentals 1, 2, & 3 (ECE 2630, 2660, 3750) provides knowledge needed to implement a proprietary microcontroller for use as the flight computer. Experience in developing circuitry and PCB foot printing is necessary to interface communications and power AgroFlight's electronics. Embedded Computing & Robotics 1 & 2 (ECE 3501, 3502) provide experience in using microcontrollers (MCU); one of the biggest challenges of the project is using a MCU that must be configured to run handle data from multiple systems as well as configuring interrupt service routines (ISR). An ISR will have to be written to respond to the distance sensor data when landing the drone. Timers will also have to be configured to perform timed tasks within the automated flight and sampling control system. For instance, the process of actuating the soil sensors will be timed. Data from the soil samples will also have to be captured and storage into registers so it can then be communicated back to the user. Digital filtering techniques will also be implemented for using sensors (distance, soil). Real Time Embedded Systems (ECE 4501) develops skills in real time

processing of signals and controls which will be used on the flight computer as part of flight control. A lot of processes will have to run concurrently; instructions must be sent to the motors while also reading the distance sensor data. Other communications from the flight controller might have to be monitored to aid operating flight navigation and landing in real time. Finally, Radio Frequency (RF) Circuit Design and Wireless Systems (ECE 4209) will aid in the selecting of a GPS module and a corresponding antenna.

Constraints

Design Constraints

CPU Limitations

For this project, performance, familiarity, and connectivity was a must. The TM4C123GH6PM microprocessor fills in the performance category with its 80 MHz system clock, 256 KB of single-cycle flash memory, and 32 KB of single-cycle SRAM [7]. Furthermore, this microprocessor (attached to the EK-TM4C123GXL evaluation board) can use up to 6 of its UART interfaces, allowing for a multitude of connections with peripherals [8]. Lastly, this board already has support with TI-RTOS, including peripheral drivers, and has been used before by group members [9]. As many things needed to be implemented for this project to work and they had to be implemented well, using external, existing, reliable code served as the tool needed to wrangle the scope of the project into a feasible range.

Software Availability

National Instruments' Multisim [10] and Ultiboard [11] were tools utilized for board design. UVA provides active licenses for students to use these tools for free. Texas Instruments' Code Composer Studio™ (CCS) [12] was an integrated development environment (IDE) used to implement embedded code for the microcontroller. QT Group's QT Creator [13] is another IDE employed to write code for the graphical user interface of the drone application. The mentioned development environments can be downloaded at no cost. The drone's frame was designed using Autodesk Fusion 360 [14], available to students for free with the creation of an account using a school email address. The 3D mesh flies were sliced using Ultimaker Cura [15] slicer which is available for free to anyone.

Manufacturing Limitations

Circuit Board

Since the manufacturing of the PCB was exported to first Advanced Circuits [16] and later JLCPCB [17], the board had to be designed to meet their machining capabilities. The board had to be an even number of layers, and due to budget constraints, this was further limited to being only two layers. Other limitations included making sure traces, vias, and pads weren't too close to each other, as their process couldn't etch copper shapes within 0.001" of other copper.

Once manufactured, the PCB was populated by WWW Electronics Inc (3W) [18]. As a redundancy in case 3W wasn't available, it was decided that the PCB needed to be operatable with the equipment in the National Instruments Laboratory, which had solder irons with tips that

wouldn't be compatible with extremely small parts. For this reason, all components were footprinted with parts no smaller than 1206 SMD components, which refers to components that are 0.12"x0.06".

Drone Frame

Since rapid mechanical prototyping was necessary to design a working frame within the project timeline, 3D printing was chosen as the primary technique to manufacture the Agroflicht's frame. The printers used for this project were two personally owned hobby printers – the Ender 3 Pro [19] and the Tevo Tarantula [20]. Though generally precise, machine tools were necessary for refining printed parts since abrasive artifacts remained around edges of the prints as well as where support material was used to print overhangs. For the instance of the drone's landing gear rails, the print artifacts interfered with the functionality by not allowing the landing gear to slide smoothly. The soil sensor assembly could not be manufactured with enough precision to predict where the mounting holes would be on the bottom plate, so the holes had to be marked and drilled into the bottom plate manually.

Ease of printing was the most important design constraint placed on the frame. Structures with overhangs, complex geometries, and protrusions were minimized to reduce print complexity. Though severe overhangs can be printed with support structure, necessary for some parts, the supports do not always remove cleanly which can impair functionality. The Tevo Tarantula in particular had difficulty printing overhangs due to the lack of a cooling fan on the print nozzle. A cooling fan helps the plastic to solidify quickly which helps overhang structures form without warping and drooping. Complex designs also increased print time because they mandate a lengthier nozzle route. Parts were therefore designed with simple angles and curves. The best parts to print typically have a long, flat plane with all complexity leaving this plane. This is so the part can be laid flat to establish a strong first layer to the print.

Since 3D printers were necessary to this project, the frame material was limited to what could be printed. Both printers were best equipped to use PLA, one of the most common and easy filaments. Though PLA is considered to be strong and ductile, it is not as strong as other potential candidates, such as metals or carbon fiber. These materials could potentially withstand more force, be more ridged, and more light weight.

Economic and Cost Constraints

The biggest cost constraint came as part of the capstone course's \$500 budget per team. This played an important role in part selection, manufacturing, and testing/configuration methods. As the project was completed in Fall 2021, the world was (hopefully) in the tail end of the COVID-19 pandemic and still feeling the effects of Silicon shortages and supply chain issues [21]. This heavily impacted bulk order periods where the entire class submitted orders for common distributors as well as PCBs. A lack of hardware delayed the integration and testing, which meant certain features of the drone had to be cut for a functional product delivered on time. With the limited budget, the drone was designed to meet the minimum requirements of a standard drone plus the soil sensor and associated frame modifications.

Another constraint is the audience of the drone. As an agricultural tool, it would be built for farms and sold through relevant product lines like John Deere Agriculture [22]. However, as a product that falls decisively under nice-to-haves, it should not be an expensive product. AgroFlight is priced similarly to Gator upgrades like their Agricultural Management Solutions (AMS) [23] and will likely attract a similar audience given the type of data they both produce.

External Standards

Barr C

The Barr Group's Embedded C Coding Standard [105] are programming guidelines that help products in the embedded domain adhere to good quality code. It strives to avoid common mistakes, improve readability, and increase accessibility of all sorts of data types and functions in C and C++. As a technology that is completely dependent on the microcontroller, these standards are important to implement to provide ease in testing and debugging code.

NEMA

Since the drone will be operated outdoors to sample soil, National Electrical Manufacturers Association (NEMA) Type 3 standards [90] will be followed to ensure the protection of the equipment inside from solid foreign objects and ingress of water, including falling direct and rain.

RoHS

The Restriction of Hazardous Substances (RoHS) is a standard that originated in Europe that restricted the use of 10 hazardous materials found in electrical and electronic products including robots, drones, 3D printers, and other medical devices [95]. Most components used in the drone are RoHS compliant, and these reduces the impact of electrical waste on the environment and human health.

IPC

Because the drone features a custom flight computer built on a standalone PCB, the engineers will follow the Institute for Printed Circuits (IPC) standards [91] on design and testing. IPC-6012 standard specifies the quality and performance for a rigid circuit board [91], and the PCB falls under class 2 which include PCBs that are service electronic products like television, computer, or air conditioner [92]. IPC-A-600J standards details the acceptance criteria for printed wiring boards which cover material surface, conductor width, spacing, plating, and soldering [93].

I2C

The flight controller has standard I²C pins that are used to export sensor data (gyro, accelerometer, magnetometer). Although this data is used internally to keep the drone stable, it can be helpful for the flight computer to collect and report back to the user. Modules like the Wi-Fi transceiver and GPS will use UART pins on the flight computer while others like the IR distance sensor can report data through GPIO pins.

IEEE 802.11

The wireless communications on the drone will adhere to the IEEE 802.11 standards for wireless LAN technology. The WF121 Wi-Fi module is IEEE 802.11n, 802.11g, and 802.11n compatible [85]. The device is also fully CE, FCC, and IC certified [85].

Soil Sensor Communication

The soil sensor communicates using the Modbus communication protocol using the RS-485 electrical standard [96]. RS-485 is a standard for the physical layer used to drive and receive a differential signal pair to and from the sensor. The signals are typically sent over a twisted pair cabling since noise sources will couple equally into both signal lines and be rejected by the differential receiver. This is especially important for the drone since an antenna is present on board to send wireless signals. The driver signal provides a minimum differential output of 1.5 V, but standard receivers can detect a differential input down to 200 mV. The lines should always be terminated with a 120 Ohm resistor to maximize signal strength and minimize reflections.

The Modbus protocol is a serial request-response communication standard where a master can initiate transactions with a responder [97]. The master communicates with the responder through the query-response cycle, where the master sends a query frame addressed to responder over a common bus, and the responder returns a message frame that includes the data bytes requested by the master. The RS-NPK-01-TR soil sensor specifies structures for the inquiry frame as well as the response frame. Figure 1 shows the frame structure specified by the datasheet that can be obtained by contacting customer support at Renke [24].

Inquiry frame

address code	function c ode	starting address	Data length	Checksum low byte	Checksum hi gh byte
0x01	0x03	0x00 0x1E	0x00 0x01	0xE4	0x0C

Response frame

address code	function c ode	Returns the nu mber of valid b ytes	Nitrogen conte nt	Checksum low byte	Checksum hi gh byte
0x01	0x03	0x02	0x00 0x20	0xB9	0x9C

Figure 1: Soil Sensor Modbus Inquiry and Response Frame Structure

RCRS Regulations

Drones by definition are a Radio Control Radio Service (RCRS), a non-commercial short-distance radio service for wirelessly controlling the operation of device which include model aircrafts [84]. This means that drones are subjected to FCC regulation for RCRS's in accordance to Title 47 of the Code of Federal Regulations (C.F.R.) Part 95 [84]. Relevant rules include [84]:

1. §95.725 - RCRS stations must not cause interference to:
 - a. authorized radio operations in the 72-76 MHz band
 - b. broadcast television reception on TV Channels 4 or 5
 - c. Industrial, scientific, or medical devices operating in the 26-28 MHz band
 - d. Fixed and mobile stations in other services operating on the same or adjacent channels
2. §95.731 - RCRS can be used in the following ways:
 - a. Control of model crafts and devices
 - i. Channels in 72 MHz frequency band can only be used to control and operate model aircraft
 - ii. Channels in 75 MHz frequency band can only be used to control and operate model surface craft
 - iii. Channels in 26-28 MHz frequency band can be used to control and operate any kind of device
3. §95.767 - RCRS transmitter power must not exceed the limits specified:
 - a. 72 and 75 MHz frequency bands – Power must not exceed 0.75 W
 - b. 26-28 MHz frequency bands – Power must not exceed 4 W
4. §95.773 - Occupied bandwidth cannot exceed 8 kHz for any emission type
5. §95.779 - RCRS transmitter types must satisfy unwanted emission limits:
 - a. 26-28 MHz frequency band. For an RCRS transmitter operating in the 26-28 MHz frequency band, the power of unwanted emissions must be attenuated below the transmitter output power in Watts (P) by at least:
 - i. 25 dB (decibels) in the frequency band 4 kHz to 8 kHz removed from the channel center frequency
 - ii. 35 dB in the frequency band 8 kHz to 20 kHz removed from the channel center frequency
 - iii. $43 + 10 \log (P)$ dB in any frequency band removed from the channel center frequency by more than 20 kHz
 - b. 72 and 75 MHz frequency bands. For an RCRS transmitter operating in the 72 and/or 75 MHz frequency bands, the power of unwanted emissions must be attenuated below the transmitter output power in Watts (P) by at least:
 - i. 25 dB (decibels) in the frequency band 4 kHz to 8 kHz removed from the channel center frequency

- ii. 45 dB in the frequency band 8 kHz to 10 kHz removed from the channel center frequency
- iii. 55 dB in the frequency band 10 kHz to 20 kHz removed from the channel center frequency
- iv. $56 + 10 \log (P)$ dB in any frequency band removed from the channel center frequency by more than 20 kHz.

Legal

The Federal Aviation Administration is the main governing body for regulating the use of drones in the United States. In order to legally fly AgroFlight in the United States, it must be registered at the FAA DroneZone website [25]. Part 107 registration costs \$5 per drone and is valid for 3 years [60]. After registering the drone, a label containing the drone's unique registration number should be placed on it. Drones registered under Part 107 can only be operated by certified remote pilots. This requires passing the initial aeronautical knowledge exam: "Unmanned Aircraft General – Small (UAG)" [26].

Once the drone is registered with the FAA and the remote pilot certification is obtained, the pilot can fly for recreational and commercial reasons. Pilots flying under the FAA must follow the regulations set by the rule 14 CFR (Code of Federal Regulations) Part 107 [27]. These most relevant operating requires include:

§107.15 Condition for safe operation

- a. Do not operate unmanned aircraft system (UAS) if it's not in the condition for safe operation

§107.17 Medical Condition

- a. Do not operate or supervise the flight of an UAS if anyone has a physical or mental condition that would interfere with the safe operation of it

§107.23 Hazardous operation

- a. Do not operate the UAS in a careless or reckless manner that endangers the life of property of another person

§107.25 Operation from a moving vehicle or aircraft

- a. No one may operate a UAS from a moving aircraft unless you are flying it over a sparsely populated area and it does not involve transportation of property for compensation or hire

§107.29 Operation at night

- a. Remote pilot in command should have anti-collision light visible for at least 3 miles to avoid collision

§107.31 Visual line of sight aircraft operation

- a. Drone should be in the pilot's line of sight throughout the entire flight

§107.39 Operation over human beings

- a. Do not fly UAS over human beings unless the human is directly participating in the operation of the UAS

The drone was registered by the UVA Operations Manager the certificate of registration is shown in Appendix 4. Because UVA does not approve recreational flights [28], the drone was registered under Part 107. Additionally, a flight request must be made when flying the drone on the behalf of the university [29]. This flight request must detail the date of operation, location, flight plan, and remote pilot in command. Also, an assistant professor with a remote pilot certificate volunteered to supervise a test flight of AgroFlight enabling the team to fly under Part 107. The flight test was performed at Milton Airfield, an established UAS flight zone under the safety protocols defined by the Part 107 regulations.

Tools Employed

Printed Circuit Board

The schematic for the Flight Computer Expansion Board was designed and simulated using Multisim. The printed circuit board design was done by forward annotating the schematic to Ultiboard. When a component wasn't available in the master library, Multisim's component wizard and Ultiboard's part wizard were used to create components and footprints. After the board layout was complete, it was exported and then reformatted by the university department's computer aided manufacturing (CAM) software to be compatible with bulk ordering.

All components of the board could be tested using the VirtualBench [30] as a power supply and signal generator. Measurements were taken using the oscilloscope and logic analyzer.

Drone Frame

The structural components of the drone's frame were all manufactured using the Tevo Tarantula and the Creality Ender-3 Pro 3D printers. Both of these machines are FDM (Fused Deposition Modeling) printers, meaning they employ additive manufacturing techniques. The 3D models are built by laying down thin layers of melted plastic filament in succession. The parts were designed in Fusion 360, a cloud-based 3D modeling software. This was chosen since it allows for collaboration on the 3D models. The models are prepared for printing by exporting them in the STL format into the Cura slicer. A slicer is a software that converts 3D models to printing instructions in the form of a G-code file. These are specialized files generated for use by a specific 3D printer and allow for configuring settings to optimize a part's structure as well as the printer's performance. These software are freely available for personal use.

The frame is held together using machine screws and nuts. Along with the standard hand tools for assembly, the parts had to be refined using machine tools. A belt sander was used to refine

imperfections. A drill press and rotary cutter were used to add screw holes that could not be reasonably implemented with the 3D printers. The machine tools are freely available to students in UVA's Lacy Hall.

Embedded Software

All embedded software was developed with TI-RTOS [9] using Code Composer Studio 11 [31]. To lessen the workload of developing a robust, multithreaded scheduler in addition to hardware drivers, a real-time operating system (RTOS) was chosen for the microcontroller. In addition, a RTOS allows one to design a system in which the timing constraints for system-critical tasks can be guaranteed (see rate monotonic scheduling [32]). When selecting an RTOS to use for the project, TI-RTOS was selected because of its integration with Code Composer Studio and its plethora of documentation in TI-Resource Explorer [33]. No one but Sean took operating systems before, so the concepts for multithreaded systems like semaphore, mailboxes, etc. had to be taught to the other developers. Finally, using XDCTools (a built-in tool for TI-RTOS that cross-compiles OS code across different microcontrollers [9]) statically, when threads and resources must be created at compile-time, was a challenge that eventually became conquered.

GUI Software

The GUI application was developed using C++ [34] and the QT framework [35]. QT is a C++ framework used to create desktop applications. This software was chosen for several reasons. It has cross-platform compatibility across Windows [36] and Linux [37] operating systems (OS). This enables developers to program, compile, and run their applications regardless of the type of OS making collaboration more efficient. It also effectively extends the C++ language to make GUI development more efficient with features like signals and slots. Signals and slots effectively integrate the QT widgets, or frontend UI components, with the objects and models maintained on the backend. The user interface of the application was developed using QT Modeling Language (QML) [38], a programming language used for user interface specification. This language is seamlessly integrated into QT and has readable, declarative, JavaScript Object Notation (JSON) [39]-like syntax with JavaScript [40] support making it easy to learn. Additionally, the OpenStreetMap plugin [41] was used to provide a map interface for the application. The QT company also provides an IDE for application development called QT creator enabling developers to run and test the application easily.

Ethical, Social, and Economic Concerns

Environmental Impact and Sustainability

The rechargeable Lithium-polymer (LiPo) batteries pose an environmental concern. Even though LiPo batteries are manufactured with a smaller environmental footprint compared to Lithium-ion batteries used in phones, mining for the raw materials required to assemble them leads to significant aquatic eutrophication which may negatively impact biodiversity and contaminate drinking water [42]. Proper disposal is needed to prevent the metal pollutants these batteries contain from contributing to human toxicity and ecotoxicity. Consumers should recycle these batteries at certified battery electronics recyclers rather than being discarded in the trash or put

into municipal recycling bins [43]. For businesses disposing these batteries in large quantities, the EPA recommends they manage them under “universal waste” regulations in Title 40 of the Code of Federal (CFR) part 27. To mitigate the negative impacts of these batteries, it is encouraged for consumers to completely discharge the batteries using the LiPo charger and ship them to the nearest battery disposal center. With this in mind, consumers shipping these batteries must comply with United States Postal Service (USPS) [44] for U.S. mail shipments and Department of Transportation (DOT) [45] for shipments with other carriers [46].

The drone’s motors are brushless which has the added environmental benefit of being more efficient and longer lasting [47]. The decreased degradation of parts leads to fewer parts being produced and the increased energy efficiency reduces the environmental load of these motors [48]. The chassis is 3D-printed using Acrylonitrile Butadiene Styrene (ABS) plastic, which is non-biodegradable, but can be recycled [49]. The other electronic components needed to assemble the drone (ESC, flight controller, flight computer, sensor, motor, PDB, etc.) should be recycled at the nearest electronics recycler. Through these measures, the impact of mass production of the drones could have on the environment can be minimized.

Health and Safety

The operation of an unmanned aircraft can pose many safety risks to the user and people surrounding them. As an electronic device that produces high mechanical rotation speed using high currents, its acceleration can produce a force that might cause a non-fatal injury to the individual. Another consideration is that if the drone loses power during flight, it could crash and damage itself along with people and property below it. Because of the safety concerns posed by drones, the FAA requires a remote pilot certification of some sort depending on the use case for the drone. Within the context of the capstone project, the drone will only need to adhere to the rules defined by Title 49 of the United States Code (49 U.S.C.) for Recreational Flyers since the drones are flown for research purposes [50]. For farmers and other business, however, operators must be an FAA-Certified Drone Pilot, and this is achieved by passing an FAA administered Knowledge Test [51]. After obtaining these certifications, the operators should follow the safety guidelines outlined by governing bodies under the FAA. The aircraft should be flown within visual site of the person operating it. The drone should also avoid flying over people if possible. The aircraft should flight at or below 400 feet in Class G airspace to prevent interference with other manned aircraft [50]. In addition to these regulations, the drone application should be used to monitor the location, altitude, and battery life of the drone to ensure safe operation. Another safety concern is exposure to electrical components of the drone. Exposure to such a high-powered system could result in electrocution at lethal levels. This system and the LiPo batteries pose a fire hazard and must be handled accordingly.

Ethical Issues

One concern regarding the implementation of drones on farms is the displacement of the migrant farm workforce. There exist agricultural drones that perform soil mapping, surveying, monitoring, crop-dusting, and spraying [52]. The automation of these tasks, typically performed by temporary migrant farm workers, could lead to decreased job security for them. Within the context of the capstone project, the drone is primarily responsible for a part of the soil testing

process which is collecting the NPK data. AgroFlight should not have an impact on the employment opportunities of farm workers because soil testing is conducted by the soil technician. Rather than displacing or replacing soil technicians in the workforce, the drones aid them in the collection of soil data as there is a need for soil collection and data analysis in the research labs. Future expansion on this drone should be considered though as more robust drone technology is integrated onto farms.

Another major concern regarding the use of drone technology is privacy. AgroFlight uses GPS technology to monitor the drone's location during operation. Even though the intent was to enable farmers to know the nutrient contents of different areas of their farmland, there are concerns about having this technology constantly monitoring one's location.

Intellectual Property Issues

Similar Patents

The first patent [102] found that shares a similar theme to AgroFlight is a drone designed to spray crops in agricultural fields. It's a substantially different design than the traditional drones that inspired the team's frame design, with the two halves of the drone split by a rigid tube that both connects them and has places for the liquid contents to be sprayed out over a wide region. It does feature similar drone hardware such as a battery and ESC. However, much of the patentability of the drone came from its liquid storage and discharge process. It stores liquid in one of the structural halves and electronically controls the flow down the tube to be sprayed out.

Another agriculturally oriented drone [103] patented is one that seeks to replace some of the heavy machinery commonly used on fields. It's a combination of an airship (like a small blimp) and a drone, which provides good mobility and great cargo capabilities. As a sort of all-in-one solution, it strives to replace tasks like crop dusting, seed planting, and soil fertilizing. This is a creative application of drone technology in the agricultural field, similar to AgroFlight. However, with such different use cases as well as a vastly different propulsion system, the team does not need to worry about infringing on their patent.

The last patent researched is the most similar to the project – a drone that can collect soil [104]. It was actually the team's initial idea before being refining to the task specified in this report. This quadcopter, like AgroFlight, is a remotely controlled vehicle to interact with soil over long ranges. However, the missions are sample-and-return style which means that it cannot collect data from site to site. The non-standard drone hardware includes a drill to impact the ground and different method of adjusting its frame to bring the drill down to the soil. Furthermore, it states a very different goal of being used to collect soil samples from dangerous or inaccessible places.

Drone Frame

The drone arms, bottom plate, and top plate are all modified versions of a drone frame found on the public file sharing site GrabCAD [53]. The design, titled "S500" frame were created by user Velimir who uploaded the design on July 7th, 2020. The files were downloaded on September 18th, 2021 for use and reference for this project. Section 4 of GrabCAD's terms of use specify that all content submitted by contributors is for non-commercial use only, unless otherwise

agreed to in writing by the contributor. However, GrabCAD dismisses responsibility for any intellectual property enforcement for usage outside of the website. If AgroFlight was to be manufactured for commercial purposes, a written agreement of approval would have to be made with Velimir, or novel parts would have to be designed for implementation. Since this project is for academic purposes, Up in Frames claims fair use of this model.

Detailed Technical Description of Project

The following section provides a technical description of every technology and design implemented in the AgroFlight. This section begins with background on part selection and their functionality, which set the framework for the project architecture. The section then provides a comprehensive technical description of AgroFlight.

The Wi-Fi Module

After selecting the AgroFlight project at the beginning of the semester, one of the first issues tackled was communication. Because project was a drone, having it tethered to a ground station at all times during the operation was massively impractical. As the group looked for answers, many technologies that could have been used for establishing communication between the drone and ground station were considered.

The first one system was a simple radio setup. The drone as well as the ground station would incorporate a transceiver that would allow them to communicate with each other, over impressive distances. Even though the rate at which data could be sent across a radio channel would be limited, this seemed like a good option. Deeper investigation revealed a glaring problem: latency. Off the shelf radio transceivers are not known for their ability to process data at a blazing rate and given the time constraints, designing one with enough power for the system was not an option. The drone and the ground station need to be able to communicate with each other quickly. Hundreds of milliseconds of latency could mean that the drone doesn't get the instructions fast enough to avoid obstacles in its path.

It seemed that drone makes did actually use radio signals as the communication channel between the drone and the controller. The difference between these and the regular off the shelf radio transceivers was the hardware capability. The drone industry had essentially developed their own mode of radio communication by tweaking the protocol and seriously upgrading the hardware for these transceivers. Flight controller software like Betaflight [54] have built-in support for these types of transceivers as well. If not for the budget of the project, embarking on this path would have significantly reduced the complexity of the project.

Looking further, the team realized that the well-known and ubiquitous IEEE 802.11 [55] Wi-Fi standard could be the solution for the project. The standard can be used to send massive amounts of data across the channel with minimal latency overheads. However, the range of the drone would be reduced be one or two orders of magnitude. Improvements in latency and pre-existing libraries were more than enough to outweigh the range limitations though. 802.11ac over UDP

was chosen as the standard for the communication protocol over which the drone and the ground station would send data to each other.

Chip Selection

After selecting the wireless specification for the project, the next hurdle was to select the hardware. Three out of five group members had taken RF design classes and were aware of the challenges around designing such hardware. Taking that into consideration, Wi-Fi modules that were factory ready to work when powered up were researched. Many results featured modules that had antennas incorporated into their designs. Choosing something like this would mean that all of the difficult RF design work could be avoided, and the team could focus on other parts of the project. A module-based solution would involve connecting the Wi-Fi module onto the flight computer through a universal asynchronous receiver-transmitter (UART) [56] channel and essentially forgetting about it.

However, the modules with inbuilt antennas were too expensive for the given budget. Discrete Wi-Fi chips that could be incorporated into the flight computer itself were explored next. On average, the results yielded chips that were five times less than the previous ones. A suitable candidate was finally found – the Bluegiga WF121 [57] Wi-Fi chip. This device was perfect for the project: a chip that was designed for IoT use and implemented the user datagram protocol (UDP) [58] stack by default. Another added bonus of using this chip was the selection of antennas. While the chip has inbuilt circuit to connect an antenna, the makers allowed users to choose the type of antenna to use. Amongst the two types of antennas, active and passive, the passive design was chosen as it was simple and much less strenuous to implement than the active one. For these reasons, the WF121 was chosen as the hardware for the Wi-Fi implementation.

Hardware Implementation

Hardware implementation was completed with the aid of manufacturer provided documentation with suggested circuit designs. These guidelines, specifically the passive antenna variants, were implemented it on the board layout for the flight computer. This is the design that is on the flight computer in the final version of the drone. The design is described in the Drone Flight Computer section.

Drone Flight Computer Software Interface

The Wi-Fi chip, like everything else in the system, uses UART to communicate with the flight computer. The flight computer is using the TI RTOS system and all of the critical processes are running as threads. One of these threads is responsible for reading in the data sent by the Wi-Fi chip over UART and another one is responsible for converting this raw stream of data to usable information. Another one of these threads is also responsible for sending telemetry data back to the ground control station.

The GPS Module

Since the infancy of the project, GPS was planned to be an integral component of the drone for many reasons. First, it could provide system status like the ground speed and altitude. This would serve as a nice redundancy check against the data provided by the controller itself. The second

and the most important reason for the use of a GPS module was to get the location coordinates and the time of soil data collection. The data would then be used to build a catalog to provide the user with a detailed graph of the nutrients around the field being tested.

Like the Wi-Fi module, the right hardware that could do this without much trouble needed to be selected. It was not difficult to come across the GN-8720 [59] GPS module. From what was gathered from the hardware documentation, was basically an embedded system on its own. It didn't just perform as a GPS module providing geographical information but could take commands and respond to them. Additional documentation provided details of the working protocol for the GPS module. It would send information to the flight computer over UART in the form of National Marine Electronics Association (NMEA) [60] sentences. An example of an NMEA sentence is shown below:

\$GPGGA,002059.942,3459.82,S,13830.07,E,1,4,1.7,2.0,M,,,*,2C

This sentence is the GGA sentence and has the longitude, the latitude, what quadrant of the earth the drone is in, the magnetic variation, and etc. The delimiters are the commas that appear at multiple different locations in the sentence. The actual data is between the \$ and the * characters within the sentence and the last two characters are the checksum for the data contained in between those characters. This checksum is calculated by using an exclusive or operation over the data, in order from beginning to the end. This checksum is always a two-byte hexadecimal number that follows every sentence sent by the module.

The GGA is one of the many different sentences that the GPS module has the capability of sending to the flight computer. The entire list of sentences in the order that they would appear in the periodic output blob from the module is shown in the Figure 2 below.

RMC
GNS
GGA
GLL
VTG
GST
GBS
GFA
GSA
ZDA
GSV

Figure 2: GPS Module Sentences, In Order

By default, the GPS module outputs eight sentences: RMC, GNS, GST, GSA, ZDA, and three GSV over the UART and it happens every second on the second. The RMC, GST, and the ZDA

sentences provided everything needed for the drone's operation, and the rest were turned off. The RMC sentence had the longitude, the latitude, and the quadrant of the drone's location on the Earth. The GST sentence had the many different errors associated with the latitude and the longitude data, and the ZDA sentence contained the current time and date.

Not only did the GPS Module send us data easily, but it could also be sent commands using the proprietary command defined in the protocol information document from the manufacturer. One advantage this provided was the ability to make the GPS module stop its periodic output and go to sleep with the following command.

\$PERDAPI,STOP*2F

The following sentence would start the periodic output behavior of the GPS module.

\$PERDAPI,START*2C

With this, the GPS Module can always be in a predictable state. This becomes very important for the project as it incorporates a real-time operating system that ensures a flying equipment does not become a free-falling projectile on a collision course with objects on the ground.

Hardware Implementation

The documentation found surrounding the GPS Module came straight from the manufacturer and was followed as the correct approach to integrating it into the design. One of these documents had the suggested circuit for the chip and the antenna which was used in the board layout. The implementation is described in the Drone Flight Computer section.

Drone Flight Computer Software Interface

The GPS Module is an embedded system on its own that interacts through UART. The only mode of communication the flight computer has with the GPS module is through serial connection. There are three threads running in the RTOS for the sole purpose of the communication. One of the threads reads, at a regular interval of 1 second, the UART port for 8 NMEA sentences sent over by the module. This thread then posts a semaphore that the convert thread would be waiting on. As soon as the convert thread gets this message, it extracts the useful data from the blurb of sentences and waits for another post on the same semaphore. This process of pending and posting on a semaphore repeats continuously as long as the flight computer is running.

The third thread is a write thread and as the name suggests, this thread writes the data provided to it onto the serial port. This data will be received by the GPS module and will behave accordingly. To make the sending of commands easier, a checksum calculating algorithm as a function was implemented within the computer. Altogether, the flight computer now had an efficient bi-directional communication channel with the GPS Module.

The Soil NPK Sensor

The name of the project, AgroFlight, suggests that it's involved in making something that will be useful to farmers working on large fields. Initially, the project was about collecting soil from different parts of the field and bring the samples back to a central place on the fields. This would

make the farmers life in the field easy as they won't have to roam around a massive field manually collecting the samples facing the weather. The drones could do it for them and all they would have had to do was send the samples in to a lab for testing.

However, research showed that the equipment needed to design such a system was significantly beyond the budget of the project. The mechanical technology was also heavier than the combined thrust provided by the four motors on the drone.

Sensor Selection

At this point, other objectives that could both be feasible to accomplish over the span of the semester and stay within the budget were explored. In a meeting with the team's faculty advisor, the team was guided to think about what else was important to farmers. NPK concentration in a farmland is one of the most important metrics the farmer has to be wary about. Probing the soil for those elements using a sensor seemed like a suitable application for the drone. Further analysis yielded relatively low-cost electrochemistry sensors that could calculate the concentration of those essential elements on farmland. A sensor that read the NPK concentration from the soil using its three prongs was chosen due to part availability and budget restrictions. This sensor communicates with the flight computer using the RS485 protocol [61].

Frame and the Landing System

Preliminary Decisions and Workflow

A preexisting 3D model of a drone frame was selected from GrabCAD as a starting frame design for AgroFlight. User Velimir uploaded the "S500" quadcopter design which was chosen due to a longer than average arm length of 17.5 cm that would fit AgroFlight's 5-inch (12.7 cm) propeller length. Figure 3 shows the original S500 drone frame design.

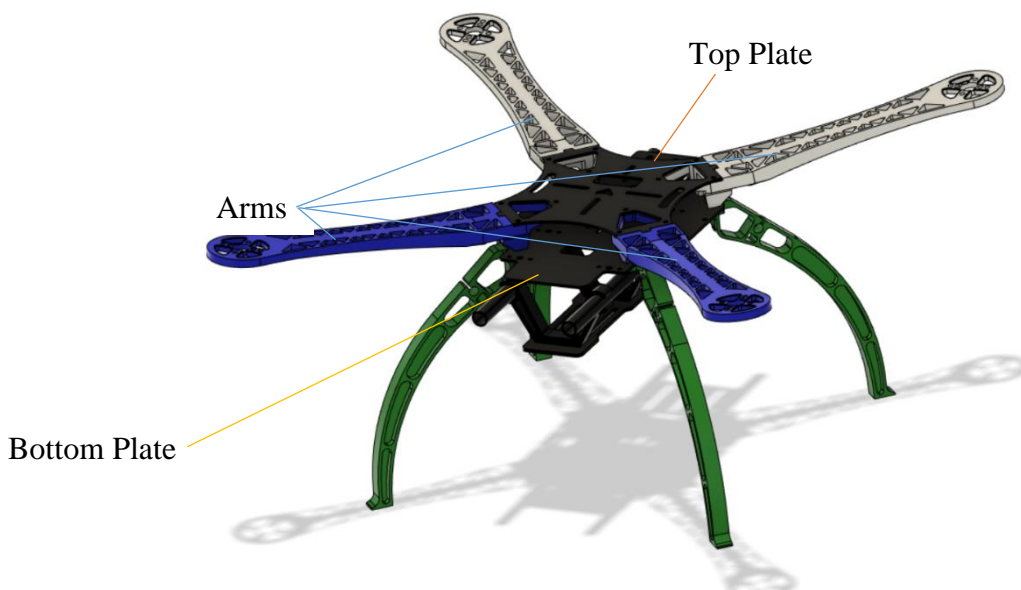


Figure 3: S500 Drone Frame Chosen as Starting Design

Significant deviations were made to the S500, mostly by way of exclusion of parts. Only the arms, top plate, and bottom plate were preserved from the original design. These individual parts were isolated in Fusion 360, modified to meet the requirements, and then exported as STL [62] files for 3D printing. The landing gear was custom designed and will be discussed later in this section. The general workflow followed to get parts created for the drone was to take measurements, create a model, 3D print the piece, test fit, and redesign if needed. Once all the parts were printed, the pieces were refined or modified at UVA's machine shop, Lacy Hall, [63] to fit together. The final form looks different than the original module. Instead of being a simple single tiered design, the drone became a multi-tiered flying soil sensor with a complex landing system in place.

The arms, top plate, and bottom plate were 3D printed with 100% infill since they experience the most load. The remaining parts were printed with between 30-60% infill.

Top Plate and Bottom Plate

The top layer is a 3 mm thick slab that has slots to connect all of the four arms. The arms have tabs that they fit into and are secured in place by 3 nuts and bolts each.

The bolt on the top of the triangle shape on the wings is an #4 SAE bolt and is ½ inch long. The other two bolts are 2-inch #6 SAE bolts that run through the entire vertical cross section of the drone. These bolts are the support pillars for the tiered design of the drone. The tiers and the legs of the drone attach to these bolts with eight in total to make a sturdy design. The top plate has an arrow shaped hole in the center which is a visual indicator of the direction the drone is facing.

The top plate had to be redesigned to a larger height since it was not intended to be 3D printed but rather machined from metal. Extraneous openings were also removed to increase rigidity. This process was replicated for the bottom plate, which was also printed to a 3 mm height. The sides of the bottom plate were also extended to give slightly more room to house parts.

First Tier

The first tier sits just below the top plate of the drone. The top plate, in addition to providing slots for the wings, also acts as a protective shield for all the layers below it. The first tier is a thicker version of the S500 frame's bottom plate. Though not as thick as Agrofliht's bottom plate, it is sufficient to house the PDB, the ESC and the drone flight controller. The ESC sits in the center of this tier with the PDB with the flight controller surrounding it. These components are mounted using ½ inch #4 SAE bolts where applicable. The wires coming out from the ESC and going into the motors are clamped down rather than mounting the board directly to the layer. Figure 4 shows the first tier.



Figure 4: First Frame Tier

This is done so that the stress from vibrations and movements would be experienced by the thicker wires rather than the weak soldered connections which have the potential of shorting if damaged.

Second Tier

The second tier houses the flight computer i.e. the launchpad and the expansion header board stack-up. Figure 5 shows the second tier.



Figure 5: Second Frame Tier

This is also the bottom layer of the drone and as such, the legs attach to this layer. The bottom layer was printed 3 mm to ensure structural strength when pressing against the legs. The 3-inch bolts also run through to the leg attachment points on the underside of the bottom layer to make the legs more ridged. The underside of the bottom layer also fits the soil sensor attachment and the drone battery. The bottom plate was modified with slots to hold Velcro straps to hold the battery to make it easier to change between charges.

Landing Gear

Agroflight's landing gear encompasses the mount for the soil sensor, the sliding legs and the solenoid holder. The landing gear functions by mounting the soil sensor to the bottom plate with two-piece legs that can retract and push the sensor into the ground. The legs slide with each other along a rail but are held in the lengthened form by force of a rubber bands. The legs are fixed in the extended or retracted position by a solenoid that holds a pin against the sliding leg.

Soil Sensor Mount

The soil sensor is mounted to the bottom plate with a 4-piece system. A two-piece holder secures the soil sensor independently from the top and bottom each with two flanges. This way force is applied to the top of the sensor when deploying into the ground and from the bottom when retracting the sensor. These flanges fit into the rail between two identical mounting posts that secure to the bottom plate. The soil sensor mounting system is detailed in Figure 6.

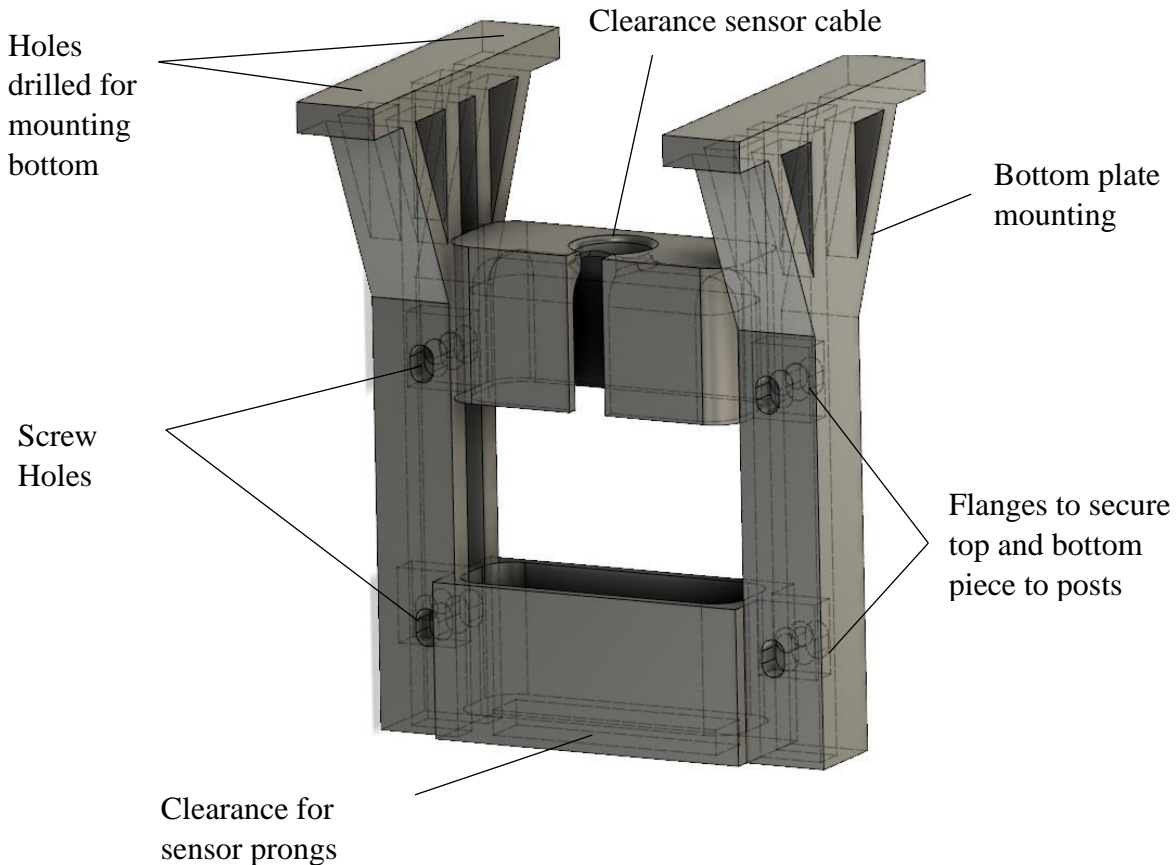


Figure 6: Soil Sensor Mount

Solenoid Holder

The solenoid is a push-pull solenoid that actuates a pin as part of a locking mechanism described in the next section. The solenoid pushes by default using a spring. The pulling mechanism is electrically actuated by inducing an electromagnet that pulls the metal pin into the solenoid. The solenoid holder has tabs that lock into the flanges on the fixed legs. The solenoid holder is a little box that houses the solenoid with the pin actuating through a hole in the box. Figure 7 shows the solenoid holder.

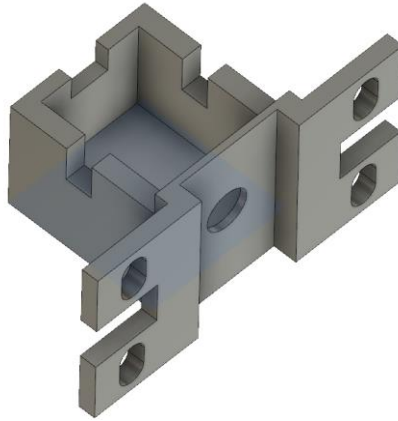


Figure 7: Solenoid Holder

The solenoid is held in the holder with zip ties.

The solenoid holder is dimensioned such that the distance from the legs allows the pin to lock the landing gear's sliding mechanism, but also the pin can be retracted once the solenoid is actuated.

Contracting Legs

The legs are built as a two-piece rail system that forcibly extend outward to prevent the soil sensor prongs from being damaged upon landing. The “fixed leg” is connected to the drone body and the “sliding leg” makes contact with the ground. A rubber band pulley keeps the sliding leg extended and builds tension when the legs are forcibly contract with downward thrust from the propellers. The rubber band attaches to the top of the fixed leg, wraps through a clearance in the fixed leg, then attaches to the top of the sliding leg to create downward tension against the sliding leg. The sliding leg has two locking holes used for fixing the legs in either the extended or contracted position. A solenoid mounts to the fixed leg and locks the sliding leg in place by actuating a pin into the hole of the sliding leg to prevent it from moving. In the solenoid's resting mode, the spring pushes the pin against the sliding leg, ensuring it stays locked in place, or it will lock once the hole moves into place.

Figure 8 shows the sliding leg attached with the landing feet. The feet have two axes of length for extra stability when landing.

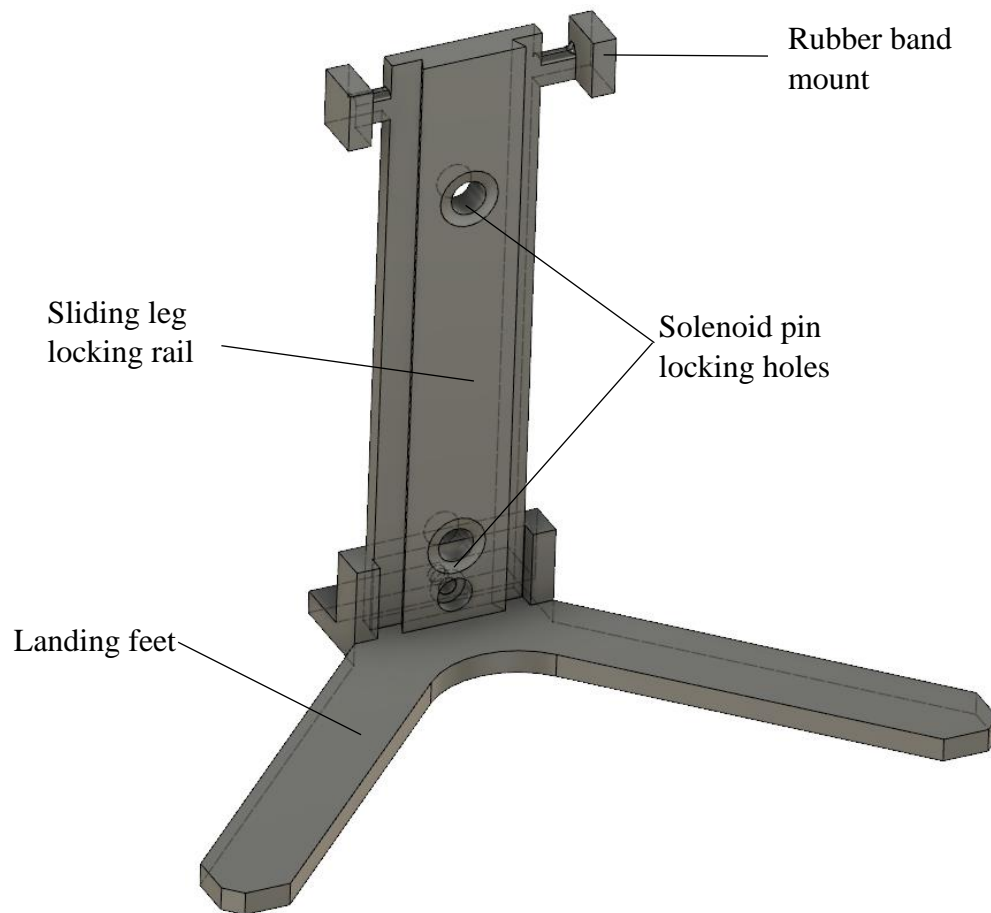
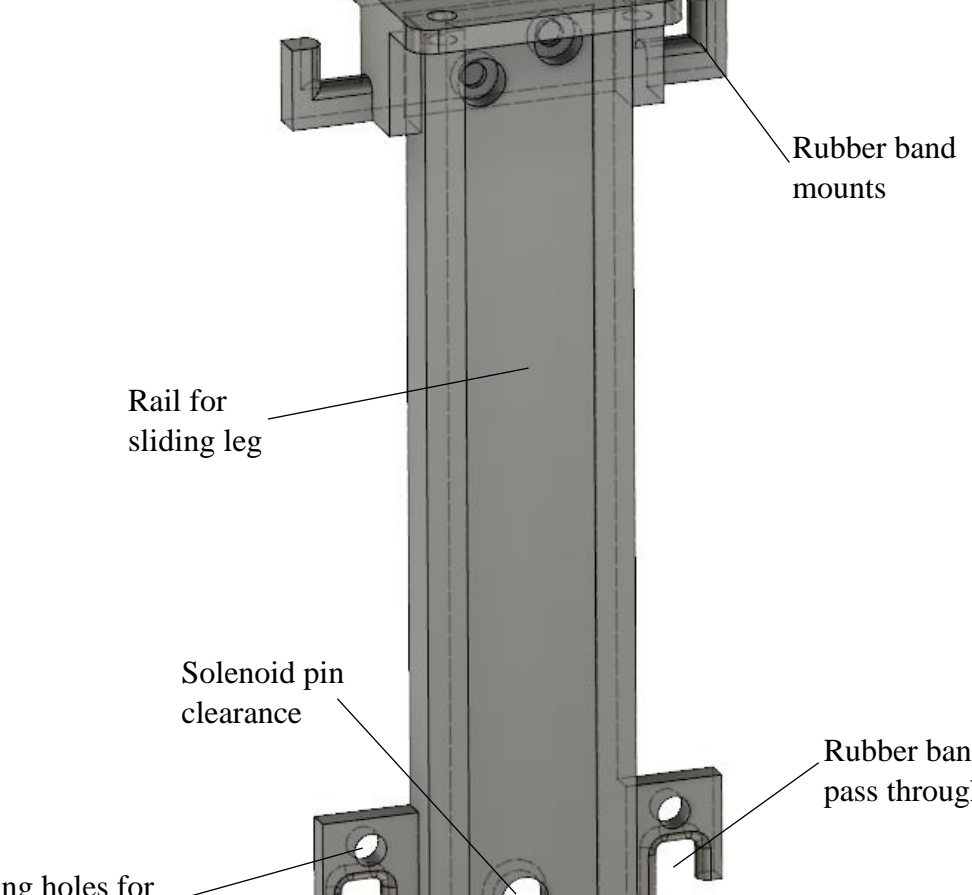


Figure 8: Sliding Leg with Feet



leg and the leg mount is shown in Figure 9 below.

Leg mount to bottom plate

Rubber band mounts

Rail for sliding leg

Solenoid pin clearance

Rubber band pass through

Mounting holes for solenoid holder

Figure 9: Fixed Leg with Bottom Plate Mount

Figure 10 shows the full landing gear assembly in the extended position. In this form, the soil sensor prongs are fixed in place with about 1.5 cm of clearance from the ground. This is the landing gear's default position. It can be held in place with the solenoid's locking mechanism.

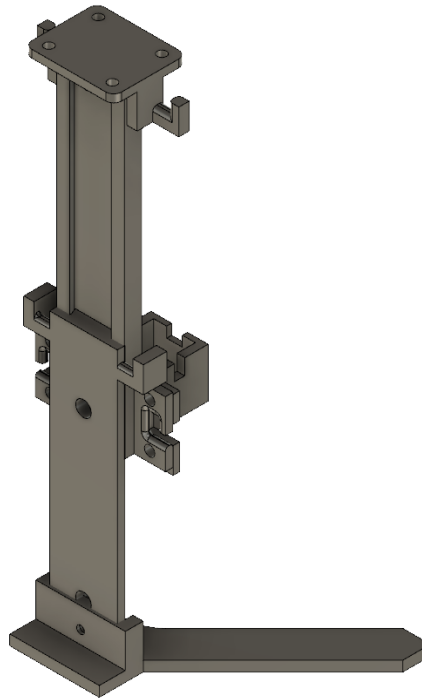


Figure 10: Landing Gear in Extended Position

Figure 11 shows the landing gear in the contracted position. In this form, the soil sensor prongs are plunged 5.4 cm into the ground. This position is held using the solenoid.

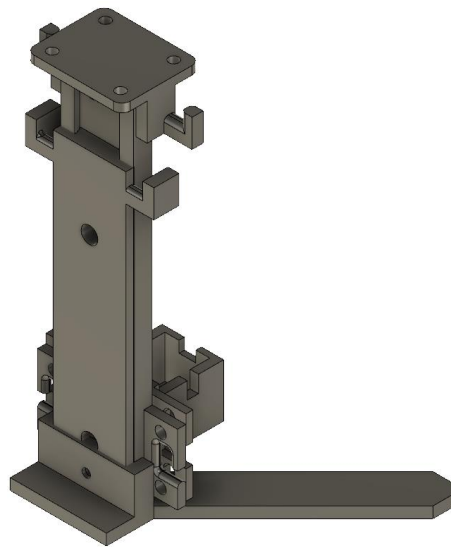


Figure 11: Landing Gear in Contracted Position

Figure 12 displays the assembled landing gear. As shown, the rubber band holds the legs in the extended position.



Figure 12: Physical Assembly of Landing Gear

The Drone Flight Computer

Agroflight's primary differentiation from a standard drone starts at the stack of communications and controls. Normal drones utilize an off the shelf radio transceiver and a controller to interface with. These devices would be used to control the drone remotely. Due to the unique purpose of AgroFlight, a Wi-Fi communicating computer application was needed as the remote, which also served as the ground station. This is not standard in the world of drones and the only ones to have done this are drone companies like DJI who employ proprietary solutions that allow the users to use an application on their smartphones to control the drone. On the controller side, the app meant that a custom solution for the transceiver on the drone was needed. This is where the concept of the flight computer came in. The computer is a combination of the TI EK-TM4C123GXL and a custom designed expansion header for the launchpad. The devkit runs all of the required functions as tasks in the TI RTOS and is the bridge that connects the flight

controller with the ground station. Not only that, but it also employs the logic for automatic landing and takeoff. A lot of hard work had to be put into this part of the project since a real-time operating system was a new concept for the group. Also, the documentation on TI RTOS was lacking in a few major areas. It doesn't describe the system behavior in much detail, if at all, and as such, experimentation was the chief method of understanding the workings of the RTOS.

Launchpad Selection

The launchpad is what houses the microcontroller and as such, there are some very stringent requirements that it has to meet. First and foremost, it has to be adequately fast at performing the required tasks as this will determine if the drone runs into an obstacle or not. Secondly, it has to have enough flash memory to store the application that will be deployed to it. Since an RTOS was determined to be an integral and necessary part of the project, candidates big enough fit the application needed to be found. Thirdly, it has to be easy to interface it with something custom – in this case, the flight computer expansion header. The fourth requirement is having enough communication channels to interface with all the components used in the expansion header as well as the flight controller. All of these requirements were uncompromisable in the search for a suitable launchpad.

The first contender was the STM32F070RB [64] launchpad. This launchpad has 128 kilobytes of flash memory and 16 kilobytes of SRAM. The raw specs look good and the board itself looks good in the pictures in Mouser. However, there were concerns regarding the board layout itself. Most of the communication channels were multiplexed, and this could cause unforeseeable complexities down the line. Also, having settled on using TI RTOS meant a launchpad from TI would be much better as it would guarantee compatibility. Finally, the EK-TM4C123GXL [65] launchpad was selected. This launchpad has 256 kilobytes of flash memory and 32 kilobytes of SRAM on it. In terms of raw specifications, this one is slightly better than the ST Electronics devkit board. In the TI devkit, however, the communication channels are not multiplexed and compatibility with TI RTOS is a guarantee. Another factor was that the entire team had experience with TI development environment Code Composer Studio which meant that writing code, debugging, and flashing were not going to be big problems. After choosing the MCU, design of the expansion header board began.

TI RTOS

The embedded software exists to facilitate communication between the different peripheral modules, the flight computer, and the GUI. Within the framework that TI-RTOS provides, the embedded microprocessor can enable communication while meeting time constraints and prioritizing some actions over others. Furthermore, synchronization objects allowed interfaces to be designed for layers that could be implemented later or in parallel. As a result of this framework, the overview of the systems in the embedded software is provided in **Error!**

Reference source not found.¹³. In the following subsections, the individual layers and the flight director will be explored.

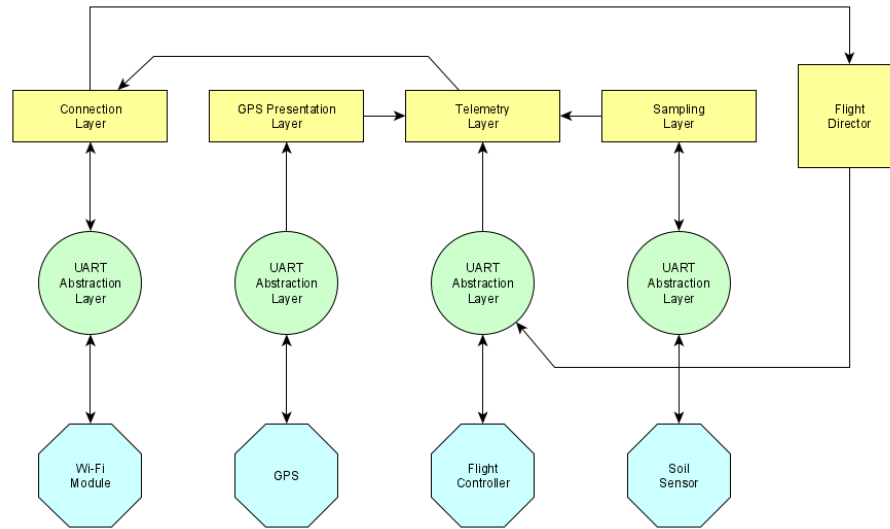


Figure 13: Embedded Software Overview

Connection Layer

Described in **Error! Reference source not found.**4, the connection layer handles the communication with the WF-121E Wi-Fi module as well as the SDCCP protocol needed to keep a reliable connection between the drone and the GUI. This module communicates to UART 1 of the microcontroller (connects to WF-121E) and with the flight director as well as the telemetry layer.

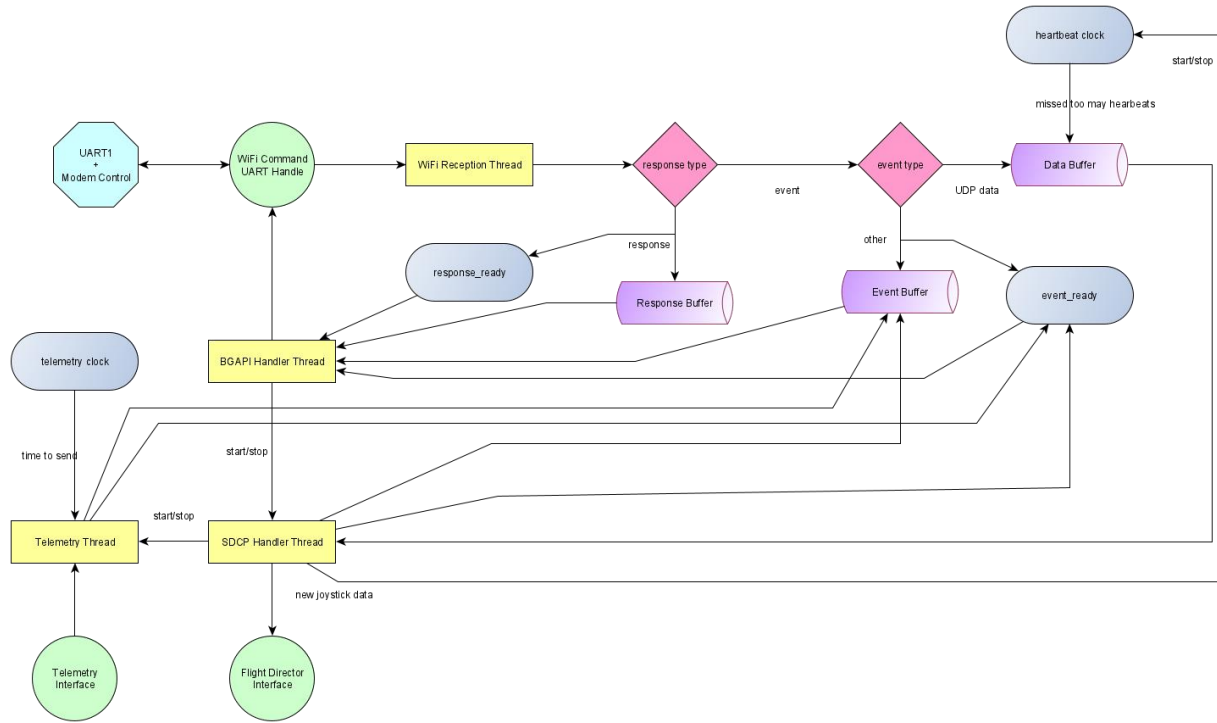


Figure 14: Connection Layer Overview

Data from the WF-121E is sent to the Wi-Fi reception thread in the BGAPI format [66]. In this format, a data packet contains a 4-byte header with identifying information, payload length, and payload type, followed by 0 or more bytes of payload information. Figure 15 shows the BGAPI packet format from the WF-121E’s protocol datasheet [66]. To ensure data is consistent and to detect bugs, the Wi-Fi reception thread begins by “header tracking”, a technique where data is streamed in, byte-by-byte, until the header becomes valid. Without this method, any unexpected payload or dropped bytes would case the communication channel to fail, the header appearing before or after where the software expects it to be. Thus, header tracking allows for this bad communication state to either drop excess data or drop the bad packet. With the expecting a periodic communication in both the UDP channel and from the BGAPI Handler thread, the dropped packet will either being retried or replaced quite quickly. After a packet has been processed, it is sent to one of 3 buffers: the response buffer if the packet was a BGAPI response, the event buffer if it was a BGAPI response, or the data buffer if it was a UDP data packet. When sent to the response buffer, the “response_ready” semaphore becomes posted. Similarly, the “event_ready” semaphore becomes posted when sent to the event buffer.

Octet	Octet bits	Length	Description	Notes
Octet 0	7	1 bit	Message Type (MT)	0: Command/Response 1: Event
...	6:3	4 bits	Technology Type (TT)	0000: Bluetooth 4.0 single mode 0001: Wi-Fi
...	2:0	3 bits	Length High (LH)	Payload length (high bits)
Octet 1	7:0	8 bits	Length Low (LL)	Payload length (low bits)
Octet 2	7:0	8 bits	Class ID (CID)	Command class ID
Octet 3	7:0	8 bits	Command ID (CMD)	Command ID
Octet 4-n	-	0 - 2048 Bytes	Payload (PL)	Up to 2048 bytes of payload

Figure 15: BGAPI Packet Format

When a response or event is received from the Wi-Fi module, it is sent to the BGAPI handler thread. In this thread, the microcontroller performs the long handshaking processes to communicate with the Wi-Fi module and to connect to a wireless network. This thread operates a “transmit-response-event” loop, where it first transmits to the module (if needed), waits and processes its response (if needed), and then handles any events sent before restarting the loop. With an upper bound to response waiting, this loop ensures that missed communication is resent until an understood response is returned. To know what to send or what to look for, the state of the Wi-Fi module is kept in a state machine, each step progressing on the machine until it is operational. This machine is shown in Figure 16. Furthermore, this machine handles SDCCP communication to the Wi-Fi module, as multiple threads could violate the transmit-and-wait rule of the WF-121e module [66]. It performs this task by treating data to be sent as an event and handing the necessary calls accordingly.

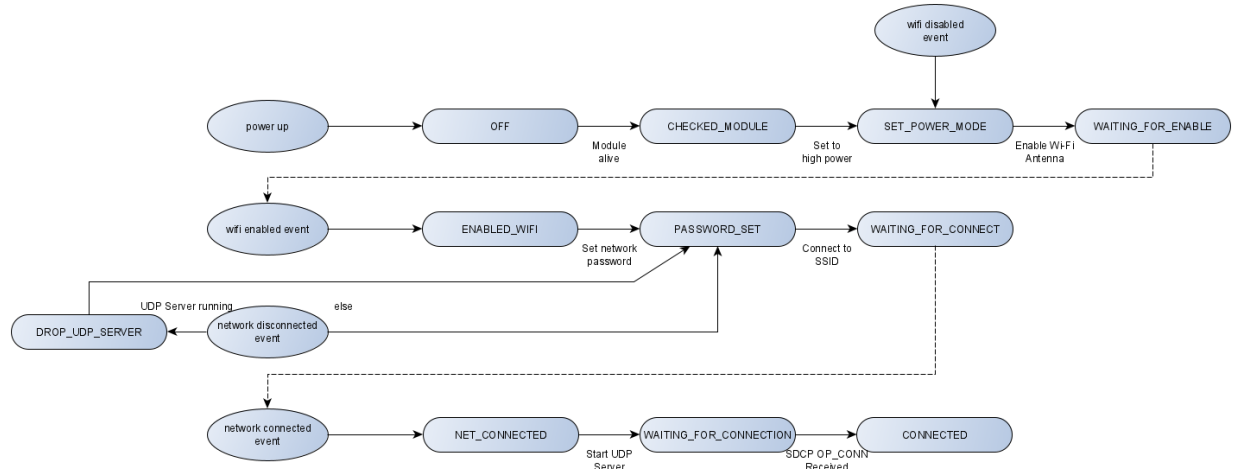


Figure 16: Connection Layer State Machine

When connected with the GUI, the SDCCP thread begins to run. This thread responds to data sent from the GUI to the drone as well as posting transmission events to the BGAPI handler thread.

Moreover, this thread manages the lifetime of the heartbeat clock, a requirement of SDCP, and will notify it when data has been received. If the clock does not receive information within SDCP's maximum timeframe (150 ms), the SDCP handler thread will forcibly receive a drop request. Lastly, the telemetry thread sends a telemetry transmit event every second. For more information on SDCP, reference the protocol in the *Wi-Fi Drone Interface Library* section.

Telemetry Layer

Depicted in Figure 17, the telemetry layer is vastly simpler than the connection layer. Telemetry from the flight controller comes in the form of LTM packets, a binary stream. Data in this protocol starts with the ASCII characters "\$T" followed by a single ASCII character to identify payload type [67]. For this project, the attitude ("A") and status ("S") frames are of interest. Figure 18 and Figure 19 show the format of the payload. After the payload, an XOR, 8-bit CRC algorithm is used to validate the packet. Unlike most other protocol supported by Betaflight, LTM requires no request to receive telemetry, a necessary requirement as the microcontroller's UART does not output the 5 V logic-level needed for the flight controller's telemetry UART port. The telemetry handler thread simple decodes this data, grabs other data from the soil and GPS layers, and populated a telemetry packet in the telemetry buffer for the next telemetry request.

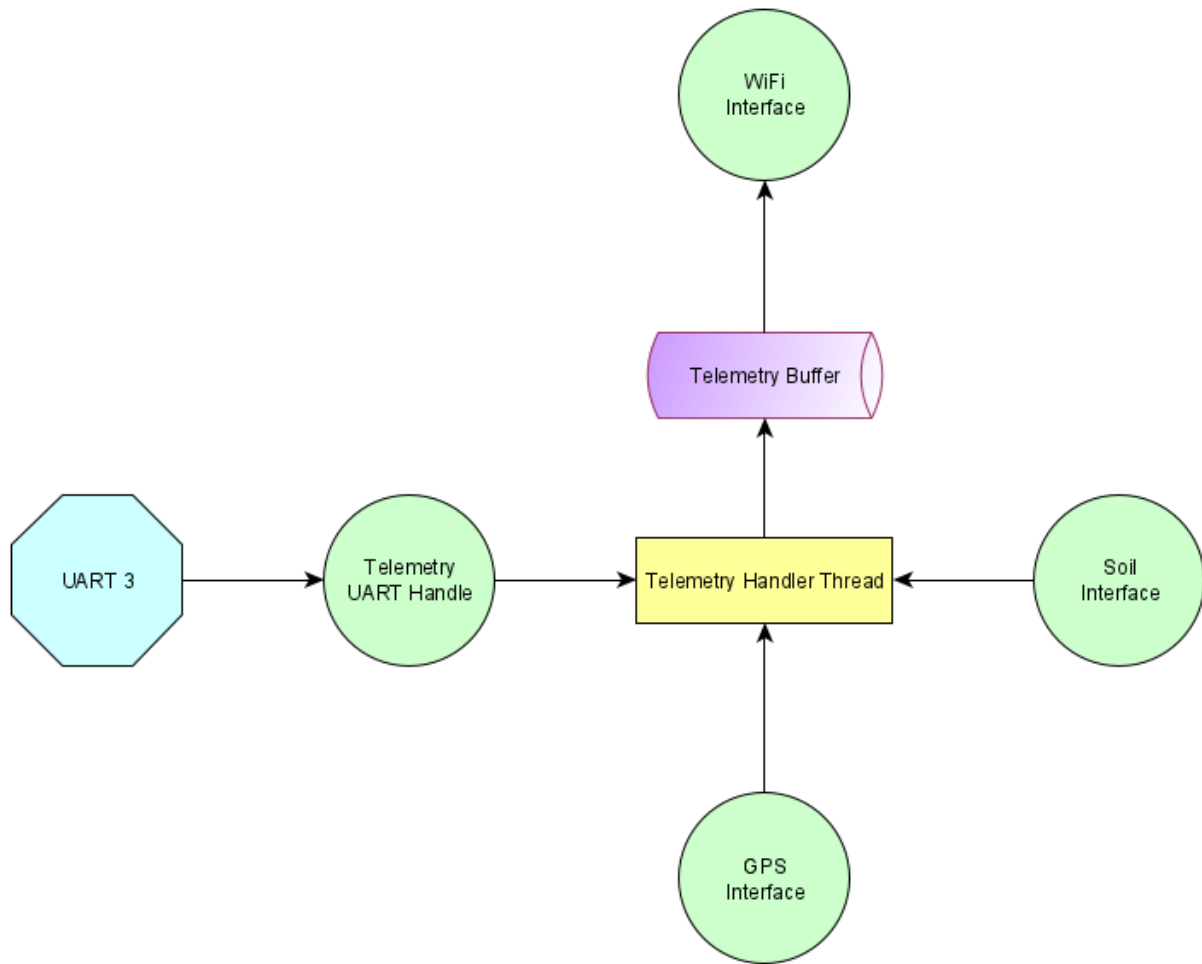


Figure 17: Telemetry Layer Overview

Attitude Frame (A)

Payload: 6 bytes

- Pitch, int16, [degrees]
- Roll, int16, [degrees]
- Heading, int16, [degrees]

Figure 18: LTM Attitude Frame Format

Status Frame (S)

Payload: 7 bytes

- Vbat, uint16, [mV]
- Current, uint16, [mA]
- RSSI, uchar
- Airspeed, uchar8, [m/s]
- Status, uchar
 - bit 0: Armed
 - bit 1: Failsafe
 - bits 2-6 : Flight mode:
 - 0 : Manual
 - 1 : Rate
 - 2 : Angle
 - 3 : Horizon
 - 4 : Acro
 - 5 : Stabilized1
 - 6 : Stabilized2
 - 7 : Stabilized3
 - 8 : Altitude Hold
 - 9 : GPS Hold
 - 10 : Waypoints
 - 11 : Head free
 - 12 : Circle
 - 13 : RTH
 - 14 : Follow me
 - 15 : Land
 - 16 : Fly by wire A
 - 17 : Fly by wire B
 - 18 : Cruise
 - 19 : Unknown

Figure 19: LTM Status Frame Format

GPS Presentation Layer

As the GPS decoding protocol has been specified before, the GPS presentation layer acts nearly identically to the telemetry layer—a UART interface to receive data, a handler thread to process data, and a buffer to hold on to said data. Thus, reference the telemetry layer and the forementioned GPS decoding process to understand the GPS presentation layer.

Sampling Layer

The sampling layer, shown in Figure 20, handles communication with the soil sensor and presents information to the telemetry interface for further processing. Unlike the GPS and telemetry handlers, the soil sampling thread must send a request to the soil sensor in order to receive data, as specified by the MODBUS protocol [68]. To read nutrient content from the soil, the registers of the soil sensor is sampled with a packet like Figure 21. Registers mapping to soil content are shown in Table 1. A response packet like the one in **Error! Not a valid bookmark self-reference.** will be returned and processed. To use the UART-to-RS-485 converter, however, the DE/RE pin must be pulled up before transmission and pulled down after [61!!].

Table 1: Soil Sensor MODBUS Addresses

Nutrient to Sample	Nitrogen	Phosphorus	Potassium
Address	0x001E	0x001F	0x0020

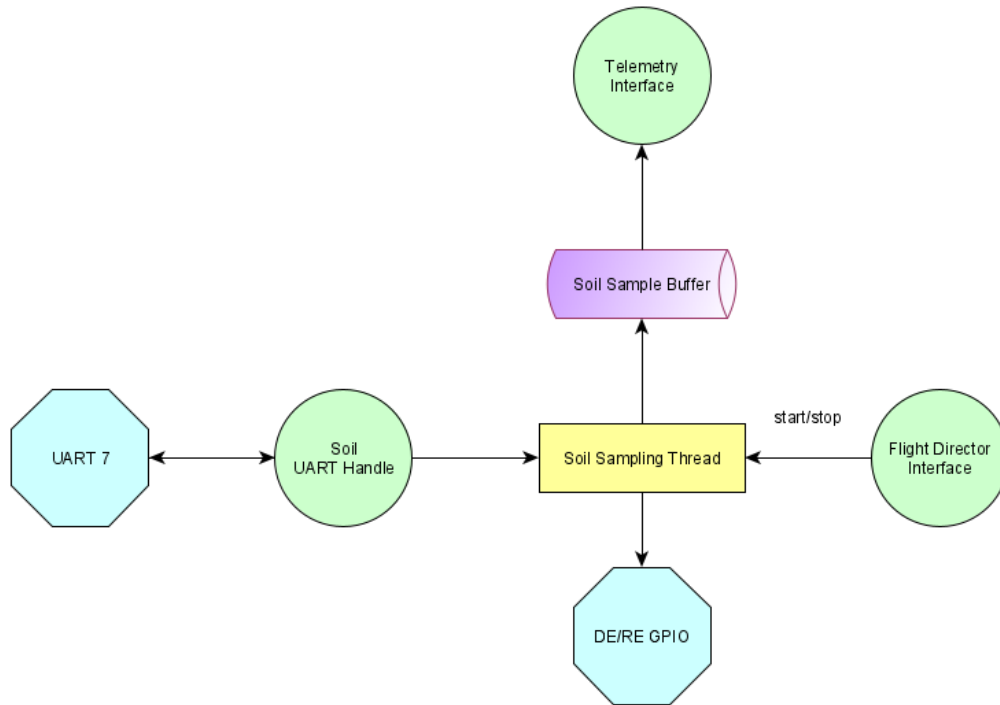


Figure 20: Sampling Layer Overview

address code	function c ode	starting address	Data length	Checksum low byte	Checksum hi gh byte
0x01	0x03	0x00 0x1E	0x00 0x01	0xE4	0x0C

Figure 21: Soil Sensor Sampling Request Packet

address code	function c ode	Returns the nu mber of valid b ytes	Nitrogen conte nt	Checksum low byte	Checksum hi gh byte
0x01	0x03	0x02	0x00 0x20	0xB9	0x9C

Figure 22: Soil Sensor Sampling Response Packet

Flight Director

The most important module, the flight director (see Figure 23) sends flight instructions from the microcontrollers and flight controller. To do this, the controller converts the joystick data into a SUMD serial RX packet, the 115200-baud, 8N1 format [69] of which is shown in Table 2. Note that channel values range from 0x1C20 and 0x41A0 (representing -150% and 150%). When the connection layer gets a joystick response, it uses a function call to update the presented packet in the packet buffer. Then, when the SUMD clock tells the director to run (once every 10 milliseconds [69]), the packet is sent to the flight controller, no extra processing required.

Table 2: SUMD Packet

Header Identifier 0xA8	Failsafe Transmission 0x81	Number of channels 0x08	Channel 1 0x2E 0xE0	Channel 2 0x2E 0xE0
Channel 3 0x2E 0xE0		Channel 4 0x2E 0xE0	Channel 5 0x2E 0xE0	Channel 6 0x2E
0xE0	Channel 7 0x2E 0xE0	Channel 8 0x2E 0xE0	CRC-16-XMODEM 0xF9 0x0F	

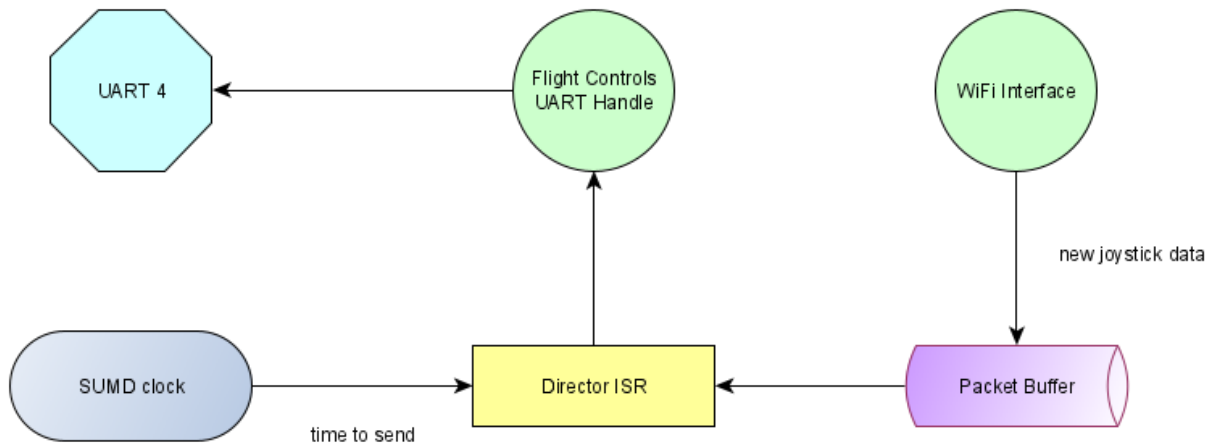


Figure 23: Flight Director Overview

Priority Scheduling

As the last section, Table 3 shows the priorities of different actions against one another. Some actions, like the flight director, are interrupt service routines while others exist as tasks. In TI-RTOS, tasks get a priority level and are scheduled in the order of highest priority first %. Only when a higher-priority thread becomes blocked can a lower-priority thread run. For threads that deal with processing UART data (wifi's reception thread, telemetry reception thread, etc.), they always have a higher priority than their processing thread, but they are not always the highest. For example, missing data on a GPS routine is not as critical as missing timing on UDP data handling (SDCP handler thread) or telemetry handling (Telemetry Handler Thread). As a result, different layers have different priorities, with the flight director preempting all, the connection layer going next, then telemetry, sampling, and finally GPS.

Table 3: RTOS Actions and Priorities

Action	Priority
Director ISR	Interrupt
Wi-Fi Reception Thread	13
WBGAPI Handler Thread	12
SDCP Handler Thread	10
Telemetry Handler Thread	7
Soil Sampling Thread	6
GPS Reading	2

Hardware Design of the Flight Computer Expansion Board

The following section details the schematic and board design of the expansion board used as part of the flight computer. The expansion board attaches to the two 2x10 pin headers on the launchpad to add necessary functionality to the MCU. Notably, the Expansion Board houses the

Wi-Fi module, GPS module, it adds interfaces for the soil sensor and distance sensor, as well as implements hardware to drive the solenoids as part of the landing gear. The Ultiboard PCB layout of the final version of the Expansion Board is shown below in Figure 24.

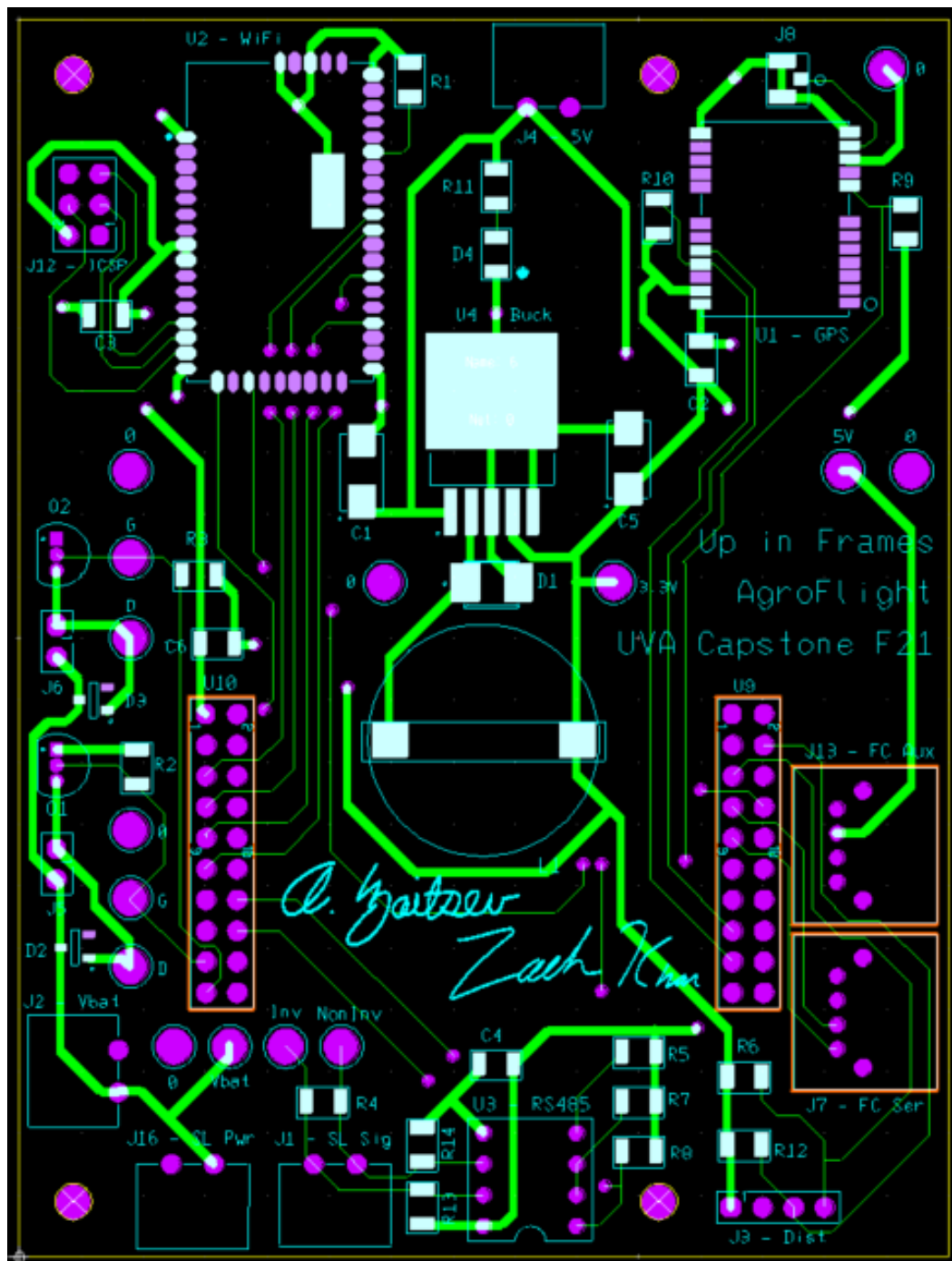


Figure 24: Flight Computer Expansion Board Ultiboard Layout

The expansion board design can be split into the following sections: power, communications, sensors, and the solenoid interface.

Power Hardware

The expansion header receives power from two inputs; wires running from the power distribution board supply the expansion header with 11.1 V battery as well as to a 5 V source. The 11.1 V line is used to supply power to the soil sensor as well as the solenoids. The 5 V is used to power the RS485 conversion between the soil sensor and the flight controller. The rest of the modules on the board are powered with 3.3 V. Since there is no 3.3 V source available, the 5 V source was stepped down using a TI buck converter. The buck converter is a TI LM2596SX-3.3/NOPB [70]. The recommended circuitry from TI to achieve a regulated output is shown in Figure 25 below.

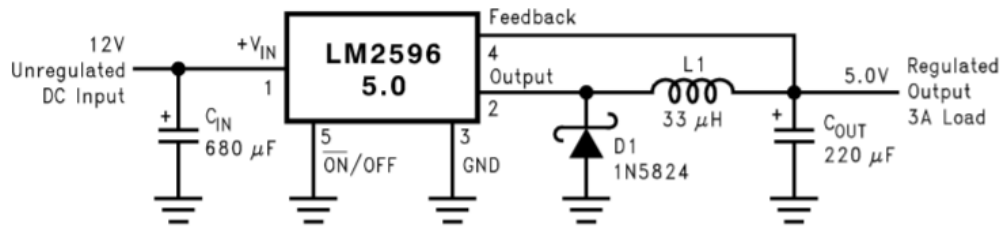


Figure 25: Recommended Buck Converter Circuitry

The recommended circuitry shows an example for a 5.0 V regulated output, but alternative component values were provided for designing a 3.3 V regulated output.

Inductor selection for L1 was particularly important since it regulates the output voltage and handles a large amount of current. The inductor selected is the Coilcraft DO5022P-153MLD [71]. This is a 15 uH inductor capable of high current draw – up to 4.0 A RMS.

The recommended topology and component values were followed creating the schematic for the board as shown in Figure 26 below.

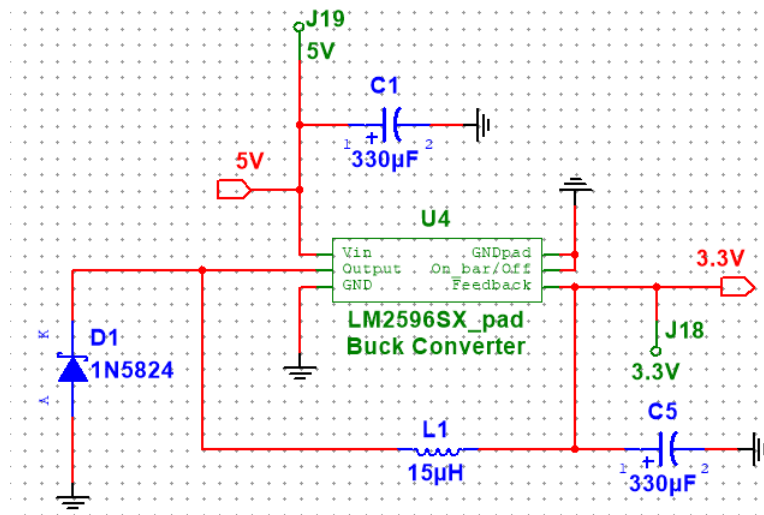


Figure 26: Buck Converter Schematic

This 3.3 V line is used to power the GPS and Wi-Fi modules as well as the MCU.

Communications Hardware

This section covers the design of powering and connecting the Wi-Fi and GPS modules to the MCU.

Wi-Fi Communications Hardware

Design for the WF121-E-V2C Wi-Fi module was done per recommendations of the datasheet provided by Silicon Labs. The module houses a U. FL connector for the connecting an external antenna. This mitigates the need for any RF traces and impedance matching on the expansion header. Figure 27 shows the WF121-E-V2C with the on-board U. FL connector.

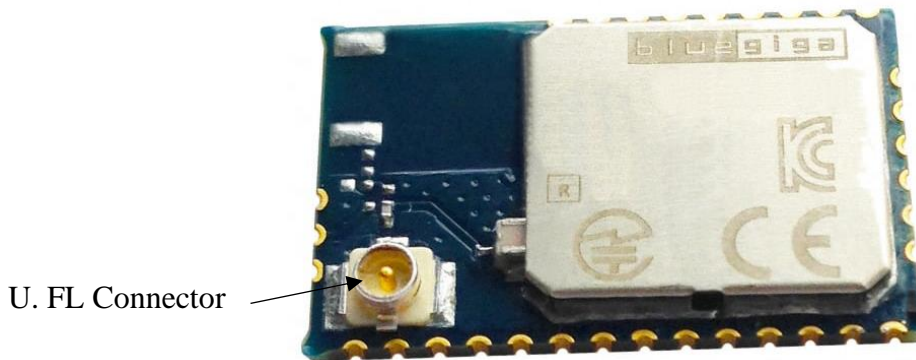


Figure 27: WF121-E-V2C Wi-Fi Module

The antenna has a +1 dBi gain and is made for Bluetooth applications.

Figure 28 shows the implementation of the WF121 module in the expansion board.

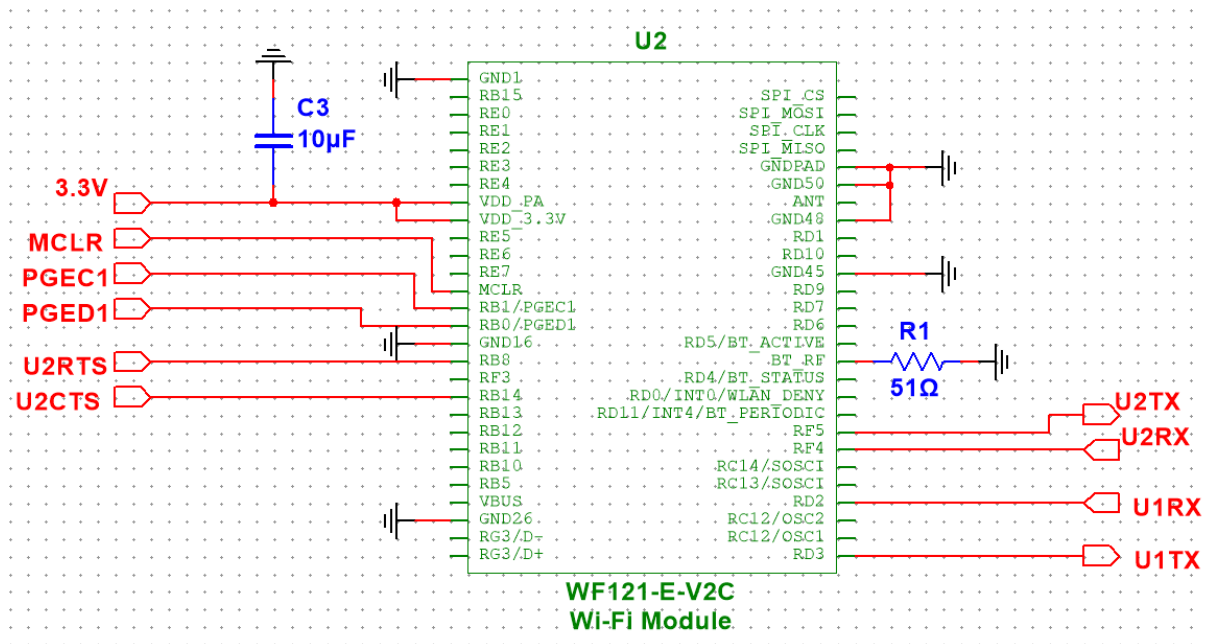


Figure 28: WF121 Circuit Implementation

The WF121 Wi-Fi module can be configured using an ICSP port. ICSP is In Circuit Serial Programming; it is a method of serially programming done via a 6-port header implemented on the expansion board. Figure 29 shows the connections to the module for the ICSP header.

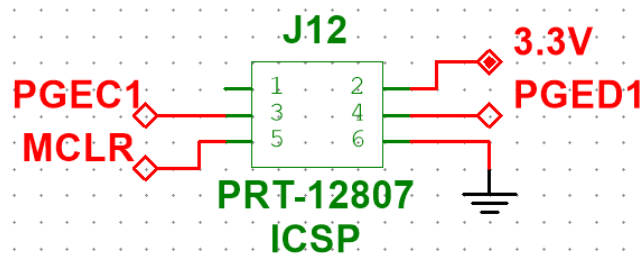


Figure 29: ICSP Header for WF121 Configuration

Although this header was made available, the default factory configuration was used since it provides the necessary UART settings to send TX and RX signals to the MCU since the MCU was configured to communicate using UART.

The expansion header establishes communication between the WF121 and the MCU by implementing two UART modules; two RX buses read the wireless data received from the antenna, and two TX bus transmits data sent by the MCU to the antenna to be sent wirelessly back to the ground station. Due to changes to the design, only UART 1 is used to transmit and receive between the MCU. This channel was chosen since UART 1 is available from the factory settings and does not require reconfiguration.

The module is powered using the regulated 3.3 V output. The 3.3 V line also powers VDD_PA, the RF power amplifier used to amplify signals for the antenna. Per the recommendation of the WF121 datasheet, a 10 uF bypass capacitor is placed on the power lines. Since Bluetooth was not implemented for this module, the Bluetooth antenna port was shorted to ground through a 51 Ohm capacitor.

Table 4 summarizes the communication connections used for the breakout board's Wi-Fi module.

Table 4: Summary of Wi-Fi Communication Ports

Port Name	Use	Pin
MCLR	ICSP RST	ICSP Pin 5
PGEC1	ICSP SCK	ICSP Pin 3
PGED1	ICSP MOSI	ICSP Pin 4
WF_U1TX	TX (MCU to Antenna)	MCU Pin PE4
WF_U1RX	RX (Antenna to MCU)	MCU Pin PE5

The only layout restrictions for the WF121 module resulted from pads on the underside. Figure 30 shows the module's physical dimensions. Traces were not drawn through pads 52, 53, 54, and 55 to avoid any interference.

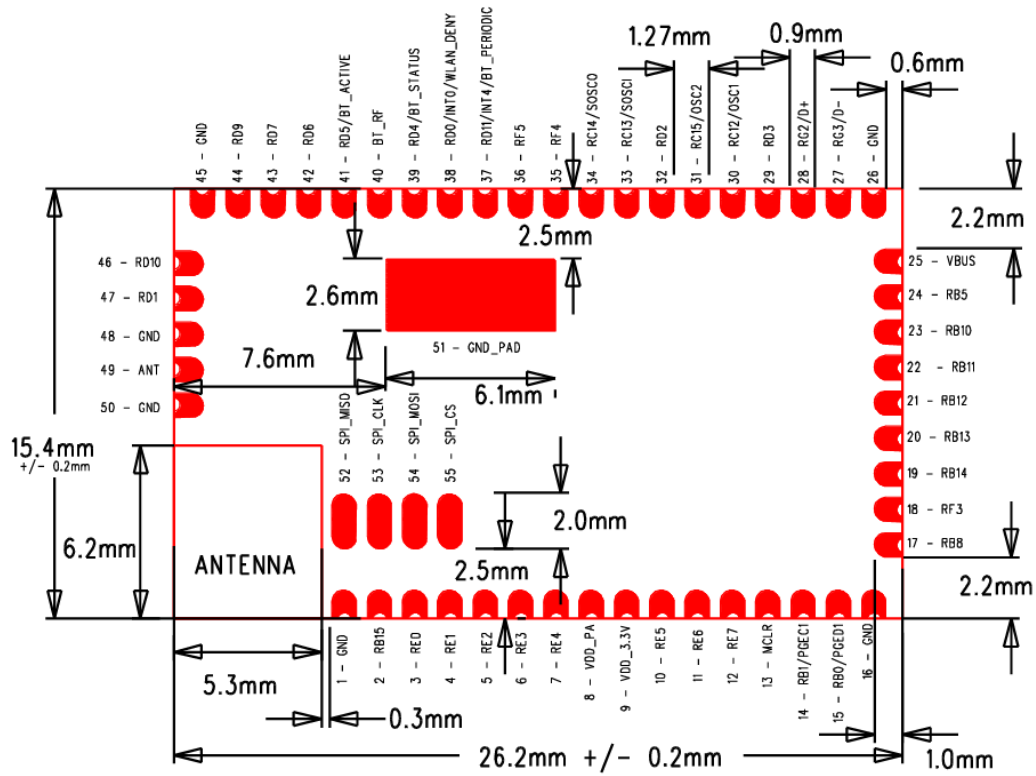


Figure 30: Physical Dimensions of WF121: WF121 Physical Specification

GPS Communications Hardware

Design for the GN-8720 GPS module was done per recommendations of the datasheet provided by the manufacturer, Kaga Electronics. Figure 31 shows the implementation of the GN-8720 module on the flight computer's expansion board.

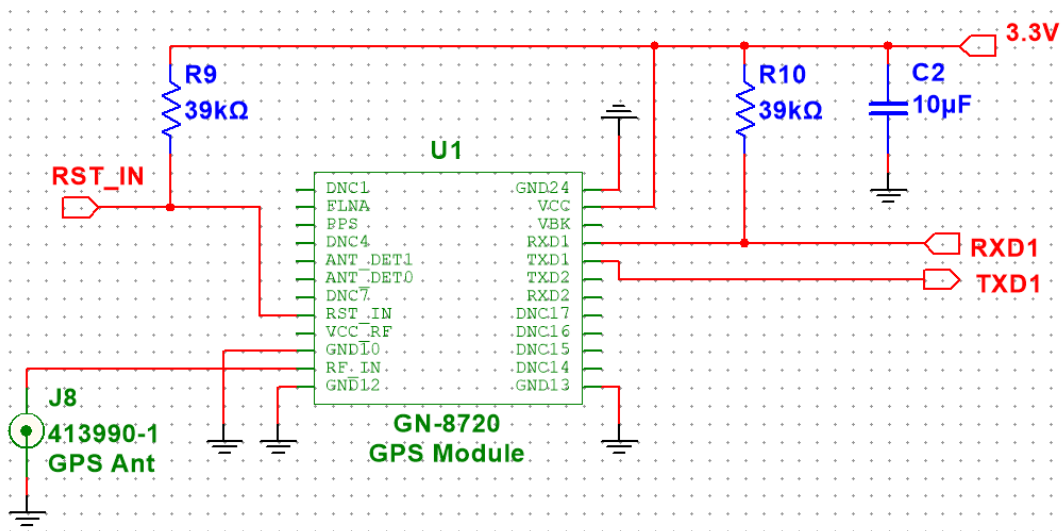


Figure 31: GN-8720 Circuit Implementation

Wireless communication for the GPS module is done using an external antenna. The module was designed for a passive antenna using the recommendation of the datasheet. Kaga Electronics recommends use of a 50 Ohm antenna. Since there is no on-chip connector, an external U. FL connector is placed on the expansion board. The selected antenna is the Molex 2065600100 which matches the 50 Ohm impedance [98]. No design recommendations were made for the layout of the connector, but it was placed as close as possible to pin 11, RF_IN.

On-board communication with the MCU is established using connections to TXD1 and RXD1, GN-8720 pins 20 and 21 respectively. The MCU uses UART pins to connect to the TX and RX pins since the GPS module is factory programmed to communicate over UART. A reset pin is also brought-up should it be needed. Pull-up resistors are connected to the RXD1 and RST_IN lines per the datasheet recommendation. A 10 uF bypass capacitor is also placed next to the VCC pin.

Table 5 summarizes the communication connections for the breakout board's GN-8720 module.

Table 5: Summary of GPS Communication Ports

Port Name	Use	Pin
RST_IN	GPS RST	MCU Pin PF4
RXD1	RX (Antenna to MCU)	MCU Pin PD7
TXD1	TX (MCU to Antenna)	MCU Pin PD6

The datasheet does specify an exclusion zone underneath the GPS module where no traces are to be placed. Figure 32 shows the exclusion zone for the GN-8720.

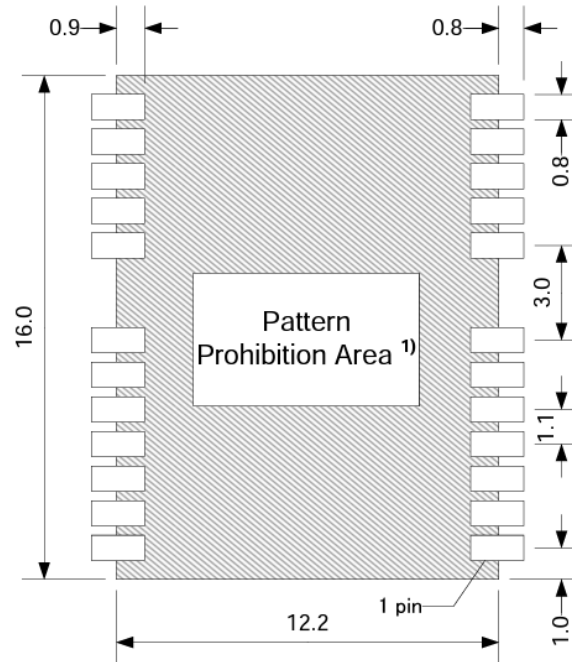


Figure 32: Prohibition Area for GN-8720

MCU Headers

All the communication signals were connected to headers fitted to the flight computer's MCU. The layout of the communication pins is shown in the schematic in Figure 33.

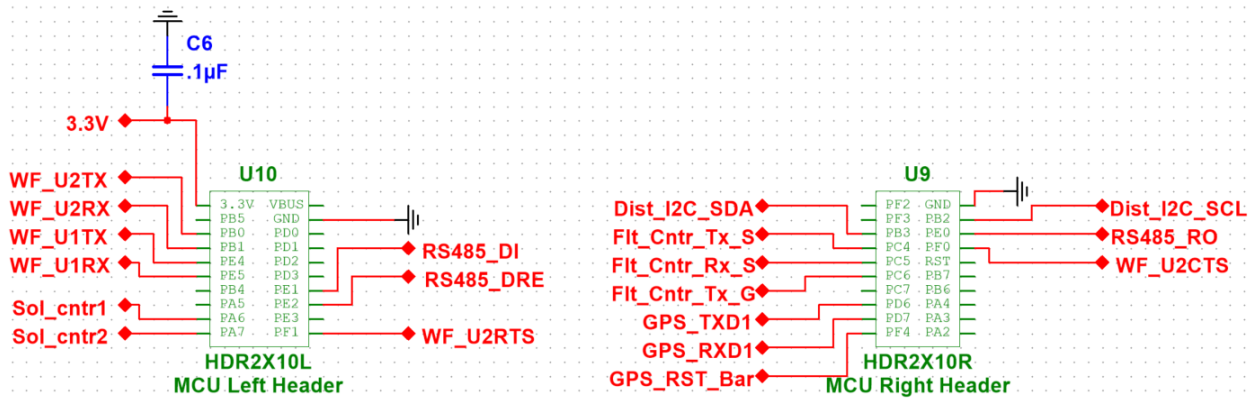


Figure 33: MCU Connections on Schematic

Table 6: Summary of Component Connections to MCU

System	Pin on TIVA	Configure for	Pin on component
GPS (U1)	PD6	U2Rx	TXD1
	PD7	U2Tx	RXD1
	PF4	GPIO	Reset (bar)
WiFi (U2)	PB0	U1Rx	U2TX
	PB1	U1Tx	U2RX
	PE4	U5Rx	U1TX
	PE5	U5Tx	U1RX
	PF0	U1RTS	U2CTS
	PF1	U1CTS	U2RTS
RS485 (U3)	PE0	U7Rx	RO
	PE1	U7Tx	DI
	PE2	GPIO	DE, RE (bar)
Distance (J3)	PB2	I2C0SCL	SCL
	PB3	I2C0SDA	SDA
Solenoid (J5)	PA6	GPIO	Gate
Solenoid (J6)	PA7	GPIO	Gate
Flight Controller Serial (J7)	PC4	U4Rx	U3TX
	PC5	U4Tx	U3RX
Flight Controller Aux (J13)	PC6	U3Rx	U1TX

Soil Sensor RS-485 Hardware

The NPK soil sensor operates using the RS-485 electrical characteristic standard with the Modbus serial communication protocol. In order to use the MCU to communicate with the soil sensor, the MAX485ESA+ transceiver is implemented to send and decode RS-485 signals [99]. The MAX485 transceiver is manufactured by Analog Devices/Maxim Integrated. A half-duplex, as specified by the datasheet, is selected since the Modbus can operate using a request-response communication; transmission and reception do not occur simultaneously. Figure 34 shows the half-duplex design using the RS-485 module. It should be noted that this module is powered using a 5 V VCC line.

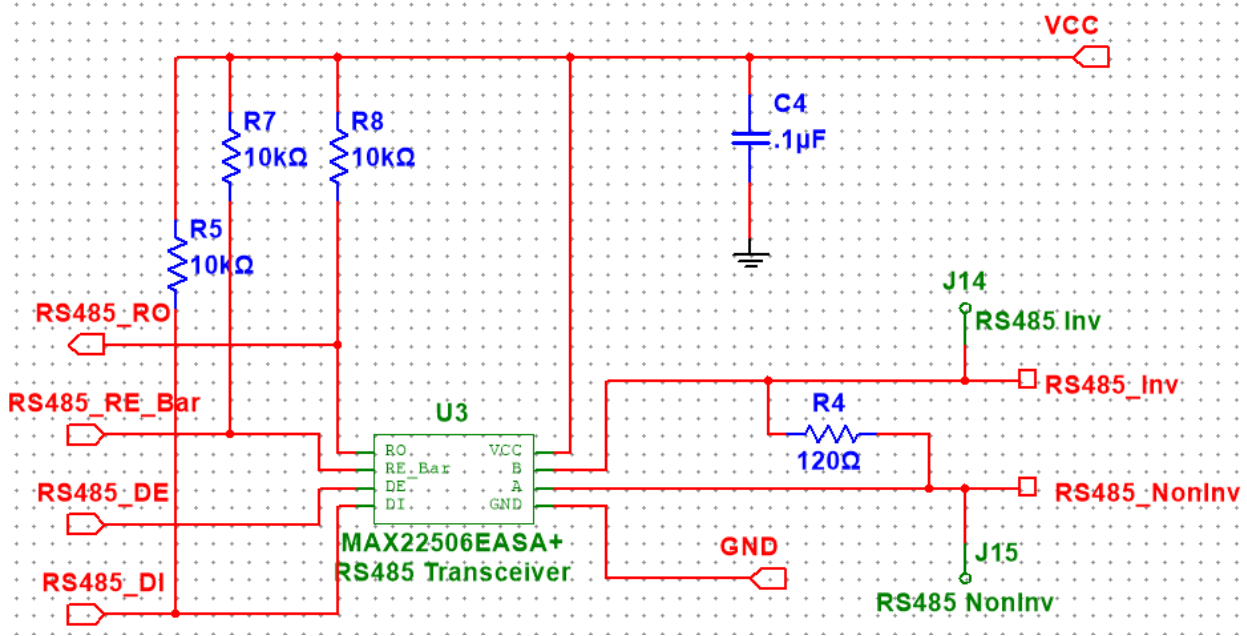


Figure 34: RS-485 Half-Duplex Transceiver Design

The module functions by having the MCU set the transceiver mode by writing to the RS485_RE_Bar and RS485_DE ports, which are tied to a single port on the MCU. A “low” output designates the module is receiving data and “high” designates the MCU is sending a request for the sensor’s readings. Since RS-485 is a differential signal, the inverted and non-inverted lines are terminated with a resistor to eliminate any signal reflection. To read data, the MCU sets the select high, then a request frame is sent over RS485_DI. The request frame is converted to RS-485 format and sent over the differential lines RS485_Inv and RS485_NonInv. Once the soil sensor receives the request, it will respond in less than 1 second. In that time, the select line then has to be changed to low as the sensor then sends the response with the real-time soil data through the differential RS485 inverting/non-inverting lines, which is converted to a standard digital format and received by the MCU through the RS485_RO port.

Table 7: Summary of RS-485 Communication Ports

Port Name	Electrical Protocol	Use	Pin
RS485_RO	Standard Digital	RX Sensor Data	MCU Pin PE0
RS485_RE_Bar	Standard Digital	Transceiver Mode Select	MCU Pin PE2
RS485_DE	Standard Digital	Transceiver Mode Select	MCU Pin PE2
RS485_DI	Standard Digital	TX Request	MCU Pin PE1
RS485_Inv	RS-485	Differential Signal	RS485B
RS485_NonInv	RS-485	Differential Signal	RS485A

Pull-up resistors are added to RS485_DI, RS485_RE_Bar, and RS485_RO lines per the recommendation of the datasheet. The final board has place for a pull-up resistor for RS485_Inv and a pull-down resistor for RS485_NonInv; however, those components were desoldered from the board because their presence interferes with the functionality. A bypass capacitor is also placed close to the module's VCC per the recommendation of the datasheet.

Distance Sensor Hardware

Initially, a discrete infrared distance sensor was sought, with the intention of placing it on the bottom of the PCB and adapting the frame for the sensor to have access to the ground. However, this became a problem when the CAD was adapted to house both the battery and soil sensor at the bottom. The search was refocused to finding an independent module that could be mounted anywhere and connected to the expansion header. Eventually, the team selected the SEN0245 board [72], which used the VL53L0X distance sensor [73] and provided a connection over I2C. This compact breakout board would be able to mount onto any remaining free space on the drone's electronics platforms and be connected to an I2C port on the MCU.

Hardware for Solenoid Interface

Two solenoids are attached to the landing gear to keep the drone legs locked out when landing which prevents damage to the soil sensor. The solenoid actuates (retracts) when battery voltage is applied, which unlocks the drone's legs allowing the soil sensor to be punctured into the ground. The premise for the design of the solenoid interface is to use a transistor to switch on battery voltage to the solenoid using a PWM signal from the MCU as the on/off. Figure 35 shows the circuit design to operate the solenoid using the MCU. An identical circuit is implemented for the second solenoid. The system's description will be spoken in terms of solenoid 1 but equivalent corresponding signals exist for solenoid 2.

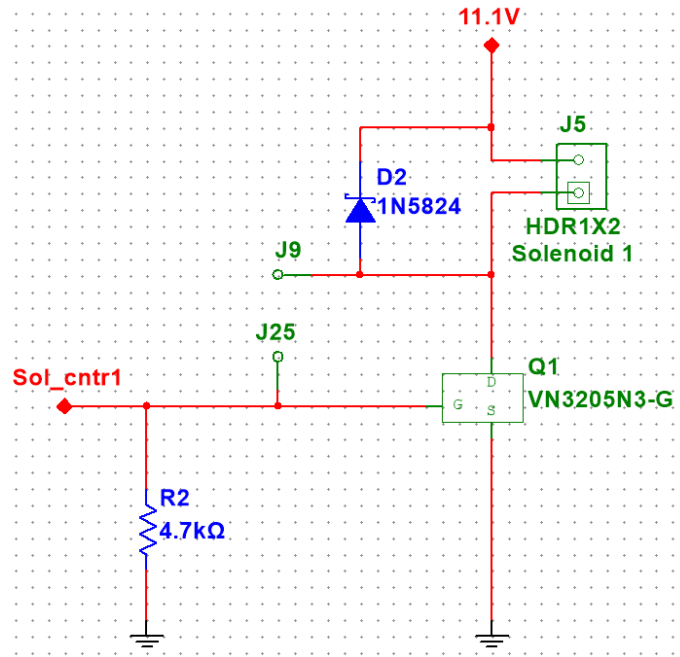


Figure 35: Solenoid Circuitry

MOSFETs were chosen as the switching mechanism to power the solenoid. The VN3205N3-G MOSFET made by Microchip Technology was selected since it fit the power parameters available on the board [100]. The MOSFET's gate threshold voltage ($V_{GS(TH)}$) is 0.8 V, which means it can be switched by a standard 3.3 V GPIO pin from the MCU. Furthermore, the MOSFET is capable of continuous drain-source current of 2 A up to 25 V. This is more than enough to support the 11.1 V battery voltage as well as 500 mA of current due to the solenoid's 21.93 Ohm internal resistance. A pull-down resistor is placed at the gate to ensure a controlled state, preventing unintentional actuation.

The solenoid is placed at the transistor's drain terminal in parallel with Nexperia USA PMEG4005ET,215 Schottky diode. The diode functions as a catch diode in order to safely reroute the voltage spike caused by a change in current, like when powering on or off. Although the MOSFETS have high tolerances, these diodes added a layer of protection to ensure the drone's ability to land safely.

Table 8 provides a summary of the communication lines with the MCU between both solenoid interfaces.

Table 8: Summary of Solenoid Communication Ports

Port Name	Use	Pin
Sol_cnr1	Power switch solenoid 1	MCU Pin PA6
Sol_cnr2	Power switch solenoid 2	MCU Pin PA7

Graphical User Interface Application

The graphical user interface application is an important software component of the Capstone project. The purpose of the application is to enable the end user to monitor the status of the drone during operation, while also allowing them to control it. While this application can be characterized as a ground station control application, the main purpose of this application is to monitor the telemetry, altitude, battery, and NPK data sent by the drone over Wi-Fi. The application was developed using C++, Qt, and QML on the QT Creator IDE.

GUI Stack Research

The development process began with research into a suitable tech stack for desktop application development, and the first one discussed was the Tkinter [74] library in Python [75]. Even though it was a popular and well-established library for GUI development, it was not chosen for a couple reasons. The library was too high up the stack to always have granular control of the system. Tkinter is designed to make it easy for anyone to design a cross platform GUI and have it running with backend code in very little time, but the abstractions python provides were not necessary to the development of this application. Another setback is the exact byte calculations of the variables to be sent over the network to the drone. The process of sending exact byte sizes over the network using python is complex, so another language was preferred.

Because the developers on the team were familiar with C and C++, additional research was conducted in C++ GUI libraries. The list was narrowed down to ImGui [76] and Qt, commonly recommended C++ libraries for desktop development. ImGui provided the benefit of having all the tools necessary to quickly get a functional GUI, but the documentation and community support was sparse. Since no one on the team had experience designing a GUI using C++, having abundant documentation and an active community was essential. Qt provided these benefits and is considered the industry standard when it comes to C++ GUI design. It's used by major automatic manufacturers like BMW and Tesla for the operating systems that run in their vehicles. Applications developed using Qt are much faster compared to similar ones designed using Tkinter or any other python libraries.

Qt is also compatible with QML (Qt Modeling Language), a markup language similar to JSON. This allows us to use styling markup languages to edit and set visual styles for many default widgets provided by Qt making handling requests with JavaScript possible. It doesn't increase application performance though, but it does improve ease of design and visual upgrades.

GUI Application Development

The desktop application for the drone was implemented over a tutorial project provided by the Qt company, Plane Spotter [77]. The purpose of the tutorial was to demonstrate the integration of location and positioning C++ data types into QML and vice versa [77]. The example displays a map of Europe, generated using the OpenStreetMap plugin, and 2 clickable planes at Oslo and Berlin. These planes travel back and forth from these two cities, but with modification of the existing models, it can be used to receive and display telemetry data on a map in real time. Additionally, tutorial projects provided by Qt are dual-licensed under commercial and open-source licenses [78]. The commercial license gives full rights to create and distribute software

made with Qt without open-source obligations, which is necessary when considering mass production of AgroFlight [78].

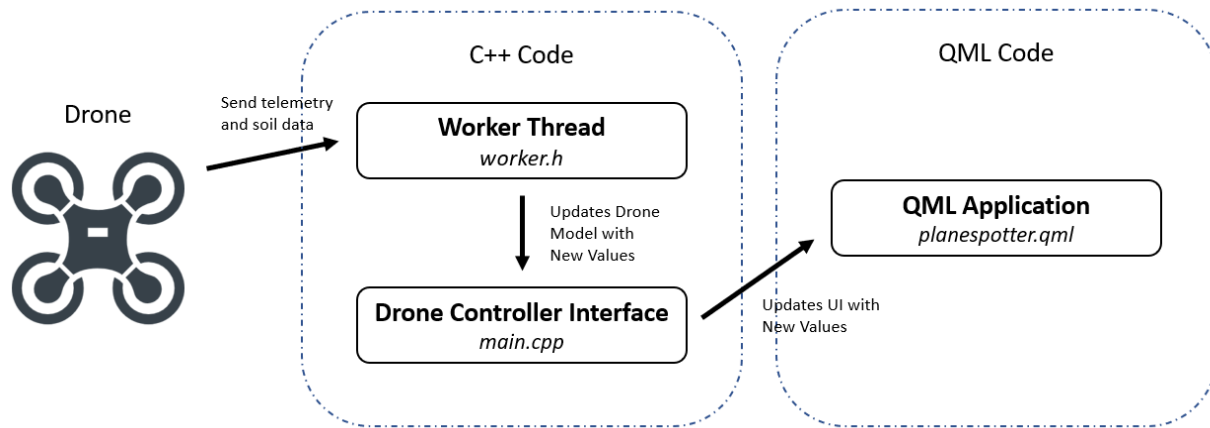


Figure 36: GUI Workflow

Using the plane-spotter application as the foundation for the GUI application, the code was split into two parts as shown in the above workflow. The backend code for the application was implemented using C++, which contains the classes and threads needed to receive and store relevant data to display on the frontend QML interface. In the main.cpp file, it contains the DroneController class to store and send relevant operation data and the main method used to execute the application. The DroneController class has several fields including the current position, battery percentage, nitrogen value, phosphorus value, potassium value, and the altitude. Each of these fields are private, so the appropriate getters and setters were implemented to safely manipulate the data. The data type used to represent the position field is the QGeoCoordinate class, a QT class that represents an object's location on the earth. It contains the latitude and longitude fields that are used to display the drone's location on the map on the frontend.

After creating the drone controller class, a workerThread field was added to it to enable the data fields to be updated continuously. The main purpose of this thread is to continuously request and receive data from the drone to update the drone controller values in real time. The data type used to represent this thread is the QThread class, a platform independent way of managing threads [79]. Additionally, a worker class was implementing in the worker.h file, and it contains a method called collectData. This method requests connection to the drone, receives data from it, and sends it to the DroneController interface. The connections established to make the data transfer process possible was implemented in the constructor for the DroneController class.

In order to establish communication between different objects in QT, the framework utilizes a feature called Signals and Slots. This acts as an alternative to event-driven programming and the use of callback methods used in other frameworks. Often in GUI programming, when the state of one widget is changed, another widget would want to be notified. In QT, this functionality is achieved by creating a signal for one widget and a slot for the other widget. A slot is essentially a function that is called in response to a signal. This enables widgets to emit a signal when it's

state changes and for the slot of another widget to be called in response to it. This concept is applied when establishing the major connections between the worker class, worker thread, and the drone controller class. When the worker thread is finished executing, the worker object should be deleted. The DroneController's operate signal is connected to the worker class's collectData slot to enable the drone to notify the worker that it wants new data. The worker class's resultReady signal is connected to the DroneController's updateData slot to enable the worker to notify the DroneController that it has new data ready for it. Establishing these connections and starting the thread enables the DroneController class to continuously update its fields while the drone is in operation.

The main method in main.cpp executes the GUI application. An instance of the DroneController class is created which prompts the thread to start running. Then, the drone's operate signal is emitted which starts the process of obtaining new position and NPK data from the drone. Then, the drone controller object is passed from the backend to the frontend by retrieving the QML engine and setting the drone context object. This enables the DroneController object to be used in the frontend QML code for continuous reference. After loading the QML file, the application's execute method is run which opens up the application in a window as shown in Figure 37 below.

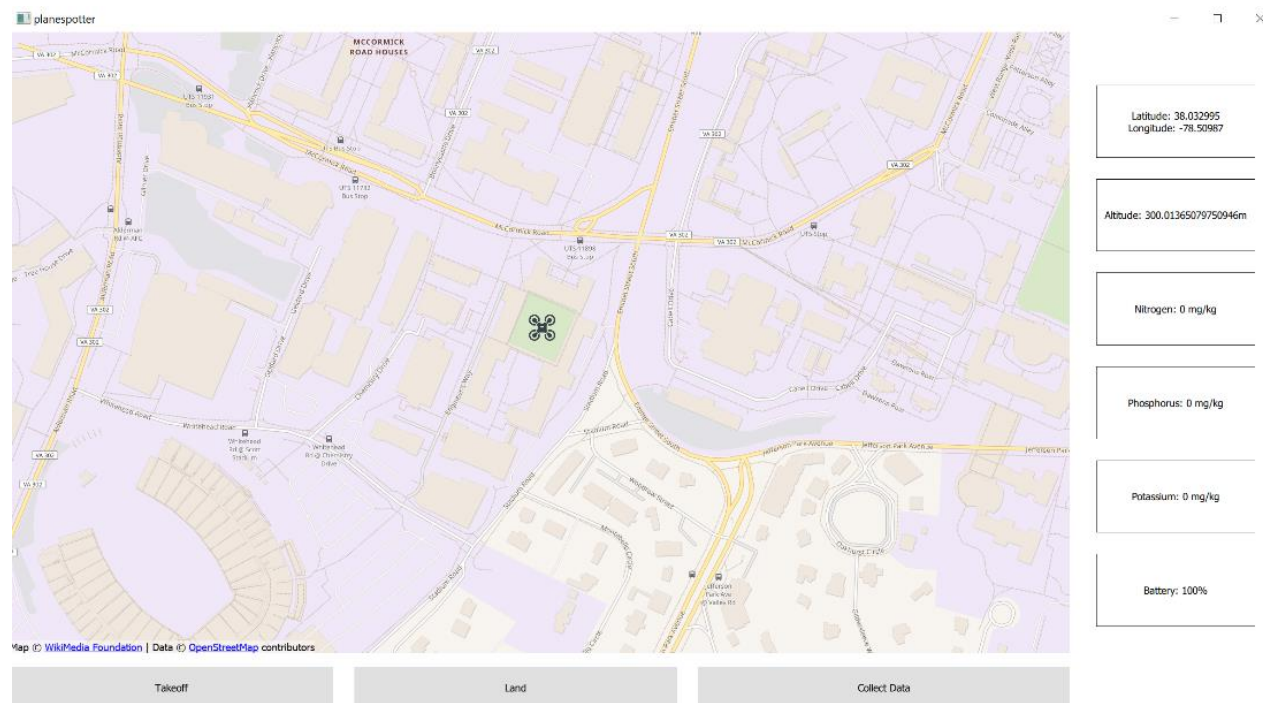


Figure 37: Drone Application GUI

The frontend user interface was developed using QML. It's a markup language unique to QT that enables the developer to implement the frontend using JSON like syntax. The main goal for implementing the frontend was to display the current position, battery, altitude, and soil nutrition data to the user from the DroneController object. Development of the map was split into three sections: map interface, control buttons, and display panels. The map interface was implemented

using the QML Map class and the OpenStreetMap plugin. The drone icon on the map represents the current location of the drone, which is set to be the middle of the Darden Court in Thornton. The panel of buttons displayed at the bottom are the takeoff, land, and collect data buttons. These buttons are pressed when the user wants to perform these actions. At the moment, the takeoff and land buttons are used as placeholders as the time required to integrate this functionality into the drone eluded the team. However, the collect data button can be clicked on to retrieve the current nitrogen, potassium, and phosphorus measurements from the DroneController instance and displayed on the panels to the right of the above figure. Additionally, the latitude, longitude, altitude, and battery percentage values are continuously updated in the display panels shown on the right which is accomplished by using the QML Timer class. This class enables the program to perform actions specified in its onTriggered field at a certain time interval specified by the interval field. Currently, the timer is implemented such that the drone's coordinate on the map, latitude, longitude, altitude, and battery life displays are updated every 1000 ms or 1 s. By using the drone controller object that was passed as the context object from the frontend, a GUI application that continuously displays the current values of the drone's position, altitude, and battery and the current NPK measurements when clicking the collect data button can be simulated.

Wi-Fi Drone Interface Library

In order to communicate with the drone, both a standard of communication and software to communicate with the drone had to be developed. Due to power consumption concerns with unnecessary TCP acknowledge packets, the connection-less user datagram protocol (UDP) served communication information between the GUI and the drone. To facilitate this connection, a library known as the Wi-Fi Drone Interface was written to manage this communication. In addition, the simple drone communication protocol (SDCP) created a “connection-light” protocol, one where unnecessary communication never occurs and connection checking kept the drone knowledgeable of the controller's presence. Finally, helper programs work with the library to configure devices and test functionality.

Interface

Written in C++, the communication library was designed to be written in only the C++ standard library in addition to POSIX sockets and POSIX peripherals. As a result, this code does not work immediately in a Windows or Mac environment, but the code can be changed in limited areas to do so.

The library presents its API through the “DroneComm” class, a wrapper for the entire communication between the drone and the GUI. Users can call methods like connect, disconnect, and drop to change the connection status to a drone, in addition to setting the drone's mode and callback functions. Every function that communicates with the drone also includes detailed error codes for debugging. To handle data returning to the GUI, callback functions are called whenever an event occurs, including connecting to a drone, blocked from connecting, connection dropped, and receiving telemetry. Transmission of the joystick as well as the heartbeat defined in the SDCP protocol are handled internally with multiple threads.

SDCP

The simple drone communication protocol (SDCP) defines how a controller communicates to the drone and how the drone responds. This protocol intends to create a “loose” connection with the drone and to determine when either the drone or the controller has stopped responding. It further defines the communication ports for the drone and controller (port 3960 for controller-to-drone and 3961 for drone-to-controller). **Error! Reference source not found.**³⁸ and Figure 393939 outline the protocol.

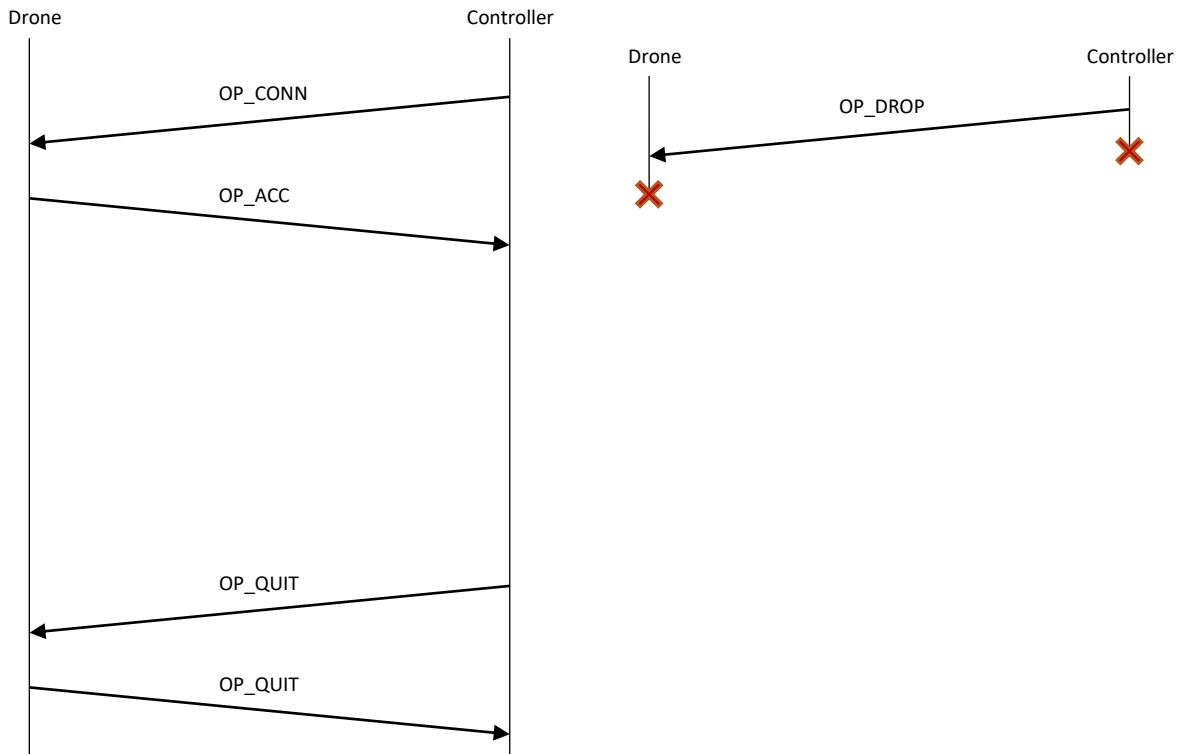


Figure 38: SDCP Communication Overview

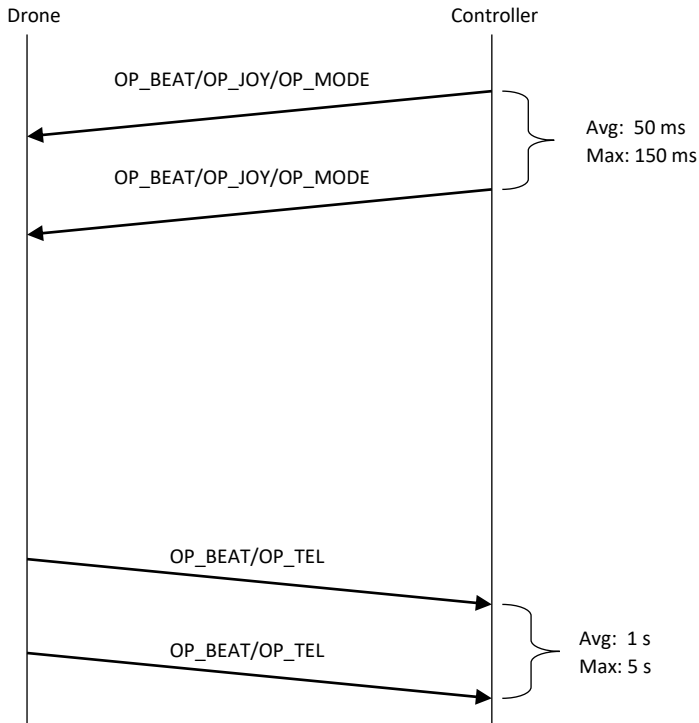


Figure 3939: SDCP Communication Overview

Communication begins when the controller sends a connection request to the drone (OP_CONN). If the drone is not being controlled, it will send OP_ACC, representing an established connection. If the drone is being controlled already, it will send OP_DENY. When the controller seeks to end communication, it sends OP_QUIT, which the drone will reply with to acknowledge the quit request. At any time and for any reason, the controller can send OP_DROP to drop the connection without waiting for a response. Similarly, the drone can send OP_DROP to the controller if it thinks the controller is no longer responding.

When sending joystick data to the drone, OP_JOY is send followed by payload bits defined in Table 9:Table 9. Similarly, OP_MODE can be sent followed by a 1-byte payload to enable explicit mode switching of the drone. Drone modes are not specified by the protocol, but modes used in AgroFlight are landed, sampling, flight, and hover. Lastly, the drone can send back telemetry data with OP_TEL followed a telemetry packet. For the application, a telemetry packet was defined in Table 10.

Table 9: OP_JOY Packet

Name	Header	Pitch	Roll	Yaw	Throttle	Button Bits
Data Type	uint8	int16	int16	int16	int16	uint16

Table 10: OP_TEL Packet

Name	Data Type
Header	uint8
Pitch	int16
Roll	int16
Heading	int16
Airspeed	uint8
Status	uint8
latitude	char8[8]
longitude	char8[8]
error	char8[6]
Nitrogen	char8
Phosphorus	char8
Potassium	char8

Both the drone and the controller must communicate with one another periodically in order to ensure communication is working properly. For the drone, it must receive communication every 150 ms or else it will drop the connection. The controller is more lenient, only needing communication every 5 seconds. Note, however, that communication should be faster than this limit. For this project, the computer will communicate every 50 ms and the drone will communicate every second. OP_JOY, OP_MODE, and OP_TEL will all keep this communication alive. If a user does not want to send data, they can send the OP_BEAT code to save bandwidth and power.

Helper Programs

The library includes 6 executables for testing library functionality and configuring controllers. The “testio” program and “stream” programs both test controller outputs. “State” improves this by include configuration data saved in “sequence” files. “Assign”, on the other hand, allows users to manipulate a controller, detect an input, and name the input for use in the program. This will then output a sequence file that a user can use to control the drone. Finally, the “net-comm” and “fake-drone” program provide testing suites for the communication library. “Net-comm” is a terminal interface for the library, allowing for testing of communication with the drone without the need for the GUI. “Fake-drone”, on the other hand, is a drone emulator that implements SDGP. With this program, one can test the library’s communication with the drone and view debugging information about the state of the drone.

The Drone Flight Controller

The drone flight controller is a small launchpad designed to interface with and control the motors in any remote-controlled flying toys. Besides the fact that it has a microprocessor in it, it has features specifically tailored towards the market. The headers used to interface different parts of the toy are there by default and no modifications have to be made. There are multiple different

sensors like gyroscope, magnetometer, accelerometer, barometer, etc. that are also built into this versatile device that resembles a devkit for a microprocessor like the launchpad. The control commands sent over from the ground station are processed by the controller and turned into physical motion by varying the speed of the motors.

Software Suite for the Controller

The controller hardware is simply a devkit that packs quite a few sensors on a small board. Without an adequate software to interface with and make use of these sensors, the device is useless. This is where controller operating systems come into play. These are real-time operating systems that operate on the flight controller board. The conversion of commands received from the ground station is handled by this software. It's responsible for managing all the different communication ports on the flight controller like UART, I2C, etc. Another important part of this software is safety. They come equipped with safety features that detect and prevent the drone from being a danger to itself and others around it. The controller software is also capable of detecting unwanted motion, for example a gust of wind pushing it to a side when the command is to move straight forward and correcting it. This is a very important part, and much attention needs to be given on selecting the right software for the use.

Betaflight

Investigation into the world of drone flight controllers immediately yielded some popular options. Betaflight was one of the first ones to be found. It's a flight controller suite that was first published to GitHub in 2015. It provided very well-written documentation about the different parts of the software, the features it has, and the configuration options it provides us with. The most important of these features is its support of multiple different control schemes. Betaflight can take in commands in common formats like SBUS, PWM, SerialRX, FrSKY, CRSF, XBUS, FLYSKY, etc. which is a healthy number of options to play around with. A deeper look into the documentation revealed that some of these schemes used inverted UART communication channels and had to be avoided due to the associated complexity. SerialRX was subsequently chosen and Betaflight was configured for it. Another nice feature provided by Betaflight is its configurator app. The application is a simple JavaScript web app that can be cross compiled to work on almost anything; they even have it on the chrome webstore as an extension and allows users to change an impressive amount of configuration options Betaflight has. It communicates with the hardware over UART, and also allows a terminal to directly communicate with the device. Everything about Betaflight is amazing – this massive open-source project that the community has been developing and maintaining over more than the last five years was critical for the team's success.

Other flight controller suites were considered, including Ardupilot. This software is similar in feature to Betaflight but has a rather nifty niche, GPS automation. A drone running Ardupilot can be given GPS coordinates and it will fly over to the exact location and hover awaiting further commands. It can also do path tracing using GPS. With more time, this would have been an interesting feature to implement into the project. Another popular option was called Cleanflight. This used to be the obvious choice just some years ago, but was replaced by its successor, Betaflight, given that it lacked many different advanced features needed by modern day pilots.

The Drone Hardware

The goal in selecting hardware was to ensure flight stability while remaining within budget. The xcopterCalc [80] calculator was used extensively to verify compatibility and to find expected flight times. Figure 40 shows the calculator configured for the project's hardware. The motor and propeller were chosen to be able to lift between two and four times as much of the expected mass. They have a unique relationship where a slower motor with a larger propeller and flatter pitch can carry more weight but flies slower, while a faster motor with smaller propellers and steeper pitch allows very zippy flight with quick acceleration [81]. AgroFlight needed to fly a relatively large payload compared to normal drone accessories like cameras. Subsequently, a 10" diameter 4.7" pitch propeller and 1100 KV motor were chosen. KV refers to the number of rotations per second per volt supplied, times 1000 [82]. Due to the lack of drone related experience of the team, the long shaft version of the motor was bought with the assumption that the propeller would be attached to that shaft. Instead, it turned out that the entire body of the motor spun, and the shaft was for encoder purposes that went unused. Additionally, the motors came pre-attached to 3.5mm bullet connectors, so compatible connectors were needed to properly connect to them.

General

Model Weight: 1600 g incl. Drive 56.4 oz

of Rotors: 4 flat

Frame Size: 500 mm 19.69 inch

FCU Tilt Limit: no limit

Field Elevation: 500 m ASL 1640 ft ASL

Air Temperature: 25 °C 77 °F

Pressure (QNH): 1013 hPa 29.91 inHg

Battery Cell

Type (Cont. / max. C) - charge state: LiPo 4000mAh - 80/120C - normal

Configuration: 3 S 1 P

Cell Capacity: 4000 mAh 4000 mAh total

max. discharge: 85%

Resistance: 0.0033 Ohm

Voltage: 3.7 V

C-Rate: 80 C cont. 114 g 120 C max 4 oz

Controller

Type: max 40A

Current: 40 A cont. 40 A max

Resistance: 0.006 Ohm

Weight: 50 g 1.8 oz

Accessories

Current drain: 0 A

Weight: 0 g 0 oz

Motor

Manufacturer - Type (Kv) - Cooling: SunnySky - X2216-1100-III (1100) good search...

KV (w/o torque): 1100 rpm/V

no-load Current: 0.8 A @ 10 V

Limit (up to 15s): 480 W

Resistance: 0.054 Ohm

Case Length: 34 mm 1.34 inch

mag. Poles: 14

Weight: 71 g 2.5 oz

Propeller

Type - yoke twist: select... 0°

Diameter: 10 inch 254 mm

Pitch: 4.7 inch 119 mm

Blades: 2

PConst / TConst: 1.2 / 1.0

Gear Ratio: 1 : 1

Results

Load: 22.1

Hover Flight Time: 9.3

electric Power: 223

est. Temperature: 38

Thrust-Weight: 2.4

specific Thrust: 6.75

Configuration

Remarks:

Battery	Motor @ Optimum Efficiency	Motor @ Maximum	Motor @ Hover	Total Drive	Multicopter
Load: 22.06 C	Current: 11.92 A	Current: 22.06 A	Current: 5.46 A	Drive Weight: 909 g	All-up Weight: 1600 g
Voltage: 10.23 V	Voltage: 10.56 V	Voltage: 10.09 V	Voltage: 10.85 V		56.4 oz
Rated Voltage: 11.10 V	Revolutions*: 10843 rpm	Revolutions*: 9680 rpm	Revolutions*: 5552 rpm	Thrust-Weight: 2.4 : 1	add. Payload: 1688 g
Energy: 44.4 Wh	electric Power: 125.9 W	electric Power: 222.7 W	Throttle (log): 39 %	Current @ Hover: 21.85 A	59.5 oz
Total Capacity: 4000 mAh	mech. Power: 109.7 W	mech. Power: 187.7 W	Throttle (linear): 52 %	P(in) @ Hover: 242.5 W	max Tilt: 61 °
Used Capacity: 3400 mAh	Efficiency: 87.2 %	Power-Weight: 556.7 W/kg	electric Power: 59.3 W	P(out) @ Hover: 199.8 W	max. Speed: 70 km/h
min. Flight Time: 2.3 min		Power-Weight: 252.5 W/lb	mech. Power: 49.9 W	Efficiency @ Hover: 82.4 %	43.5 mph
Mixed Flight Time: 6.8 min		Efficiency: 84.3 %	Power-Weight: 151.6 W/kg	Current @ max: 88.25 A	est. Range: - m
Hover Flight Time: 9.3 min		est. Temperature: 38 °C	68.8 W/lb	P(in) @ max: 979.5 W	- mi
Weight: 342 g		100 °F	Efficiency: 84.3 %	P(out) @ max: 750.9 W	est. rate of climb: 8.8 m/s
12.1 oz			est. Temperature: 29 °C	Efficiency @ max: 76.7 %	1732 ft/min
			84 °F		Total Disc Area: 20.27 dm²
			specific Thrust: 6.75 g/W		314.19 in²
			0.24 oz/W		with Rotor fail:

share

add to >> Download .csv (0) << clear

Range Estimator

Figure 4040: Xcopter Calculator for Drone Hardware

The battery's only requirement was to be able to supply more current than the burst draw of the four motors combined. Since batteries are rated by a "C" value that refers to their discharge rate,

this needed to be multiplied by the amount of energy the battery is able to store [83]. A 4000mAh 50C battery was selected since it could provide 200A instantly, which is more than the 4x 34A burst current of the motors. Since this would provide close to only 10 minutes of flight time, a second 5000mAh 50C battery was purchased that could deliver 250A. The ESC needed to be able to handle the motors' burst current as well as be compatible with bidirectional flight firmware. BLHeli [84] is the most common ESC firmware, and its latest BLHeli_32 [85] version supports bidirectional flight, so it was the main metric used in the search. A four-in-one ESC was found that supported BLHeli_32, could supply 35A to each motor continuously, and was compatible with the motor speed control signals from the flight controller.

Finally, the soil sensor needed to be selected. The team was looking to collect data on NPK in soil, and initially were stumped after most research pointed to formal laboratory analyses, chemistry test kits, or infrared spectrometers that cost thousands of dollars. Eventually, the sensor used in the project was chosen. Technical documentation was received directly from the vendor and confirmed compatibility with supply voltages and project complexity.

Project Timeline

The project timeline was originally proposed in the proposal at the start of the semester. Other than project planning, this preliminary timeline helped Up in Frames determine the scope of AgroFlight. Figure 41 shows the Gantt chart proposed at the start of the project.

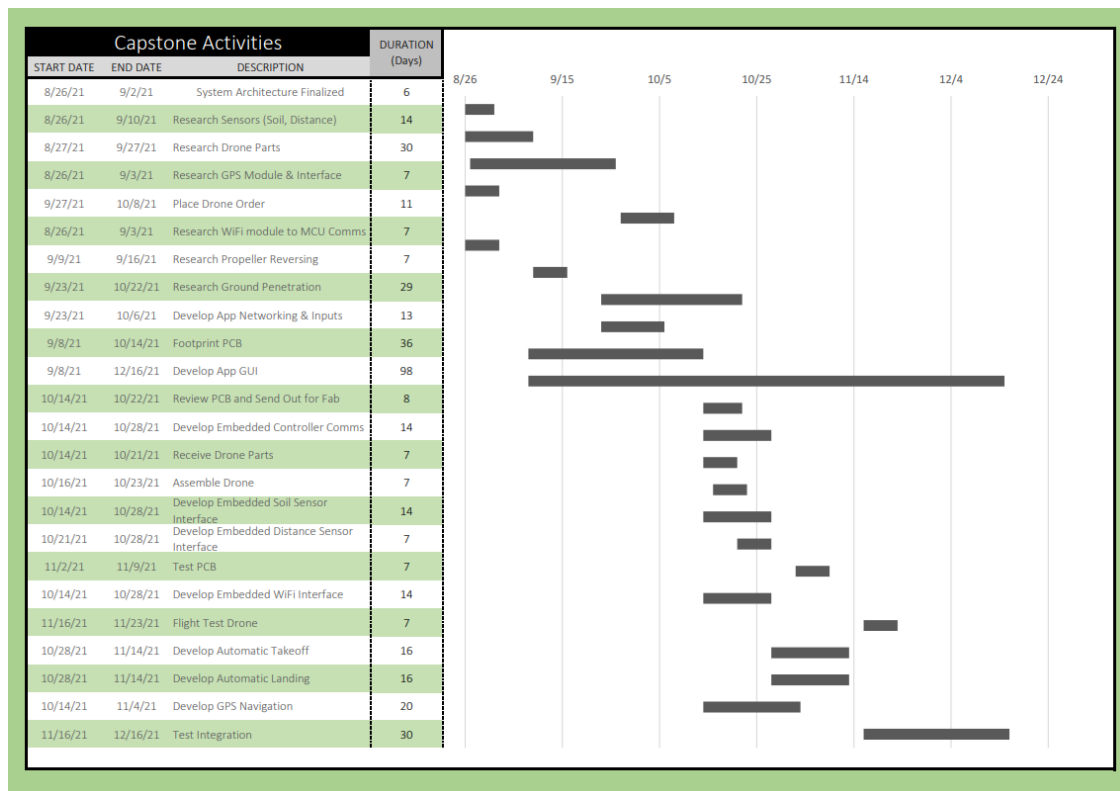


Figure 4141: Preliminary Gantt Chart

The project timeline changed drastically throughout the course of the project. Figure 42 shows the finalized Gantt chart that resembles the timeline workflow by Up in Frames.

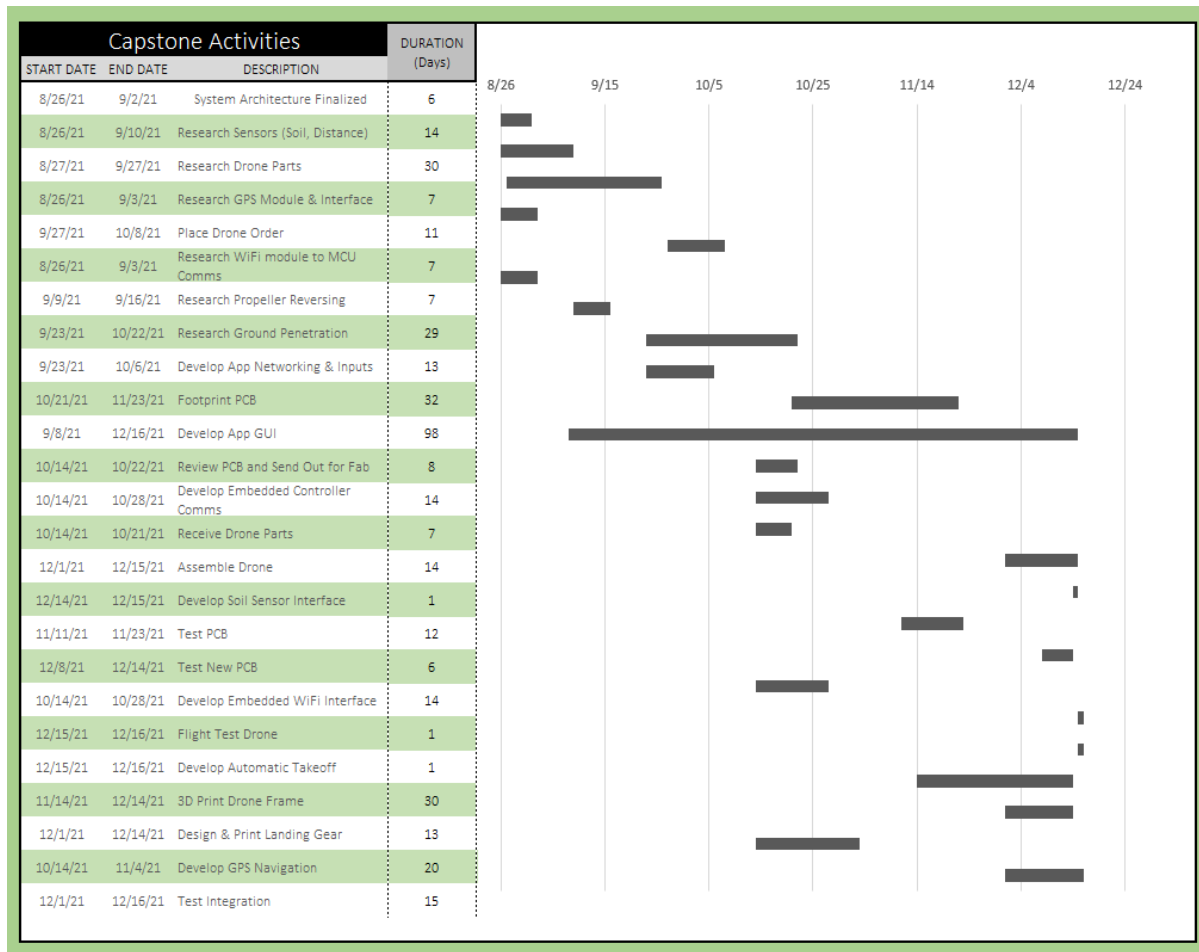


Figure 4242: Finalized Gantt Chart

The timeline reveals several setbacks. Firstly, the length of design and layout of the PCB was underestimated. It was assumed that the first iteration of the PCB would be perfectly functional, however, design flaws meant an updated board would have to be sent out. The schematic redesign and process of completing another footprint doubled the time allocated to the PCB design. The initial timeline also did not allocate any time for designing and 3D printing the landing gear. The landing gear added an additional 14 days of design work which took time away from testing and integration.

The final setback was not being able to get BetaFlight to arm the drone. Once the communication was confirmed to be established, an unidentified issue was preventing BetaFlight from arming.

Up in Frames was able to benefit from the greatly parallelized nature of Agroflight's design. The GUI was designed independently of the hardware. Testing with sample data was done to verify the GUI functionality, but design and development could be done with no overlap.

The embedded software design could also be done independently, though it did require testing with hardware once completed. For this reason, physically testing portions of the embedded code pertaining to the Wi-Fi module, GPS module, soil sensor, distance sensor, and solenoid were contingent on the circuit board being complete. Testing had to be delayed since a second board had to be redesigned.

The frame and landing gear design could be done almost completely in parallel of the rest of the project, though it was held off until more important work was completed. Though the frame was dependent on the selected drone parts, the frame design was able to proceed independently since part selection was one of the first tasks to be completed. The only design consideration was the need to mount various boards on the frame. Once the dimensions were known, a multi-tiered frame was chosen to mount the flight computer, flight controller, PDB, and ESC.

The highly serialized tasks revolved around integration and testing of various systems. In order for a first flight to happen, the PCB, network communication, embedded operating system running the state machines, and frame all had to be completed.

Figure 43 below illustrates the primary and secondary responsibilities of each teammate, as well as the correspondence for shared responsibilities.

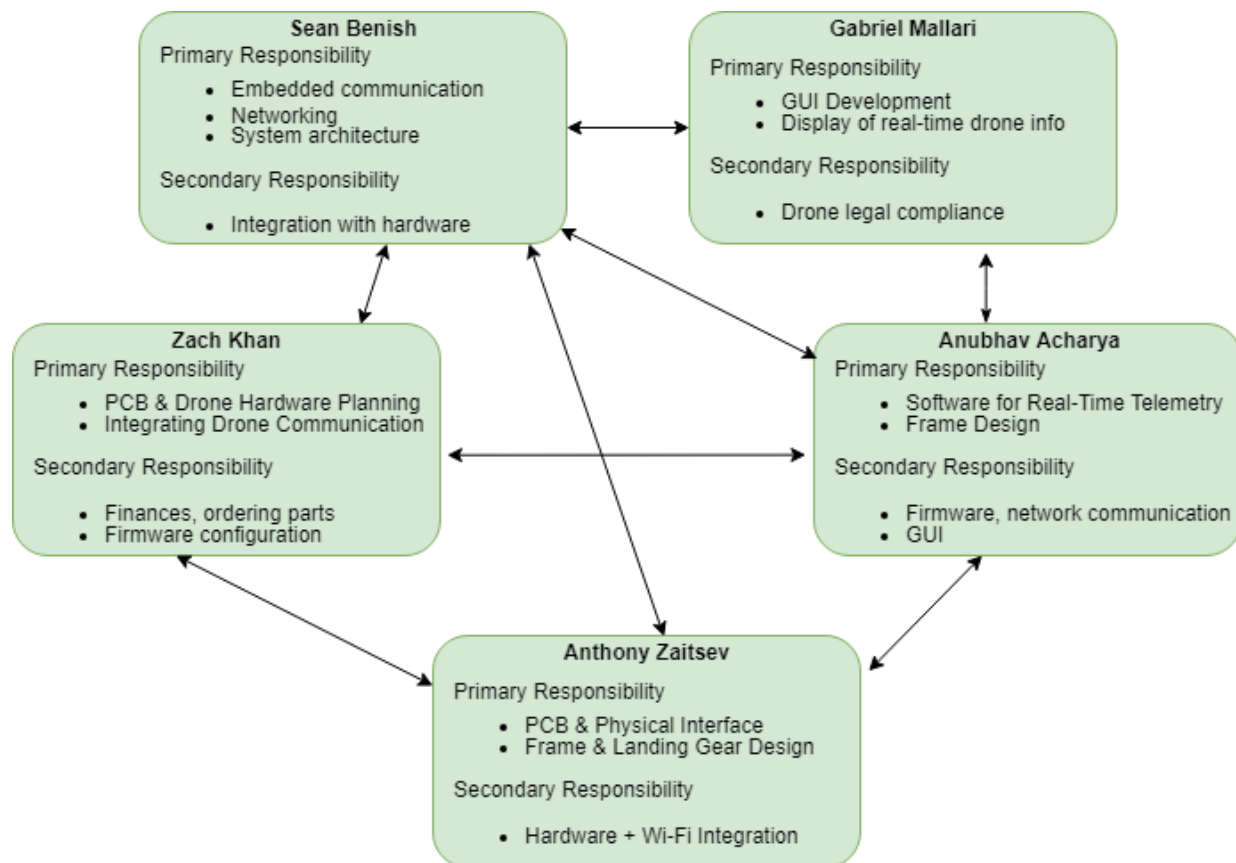


Figure 43: Team Member Responsibilities

Test Plan

Hardware

A test plan was created in order to verify that the expansion header's on-board hardware and off-board connections all worked. Appendix 3 shows the original test plan for the first revision of the board, including test results and changes to be made. Each IC, connector, and pin used on the MCU were tested (although there was much overlap across these three categories). Some IC connections had test points, but sporadic usage of test points redirected testing to be conducted by probing the IC leads. All connectors, including the headers to the MCU, had male pins which allowed them to be connected to the logic analyzer in VirtualBench.

While performing conductivity testing on the first revision board, the ground and 3.3V traces were observed to be shorted. After a few hours of probing, the problem was revealed to have been a misunderstanding of the pins on the original buck converter diode. It had three pins, with two pins for the anode and one for the cathode. An assumption was made that the cathode pin was for heat dissipation and that one of the anode pins was the cathode. This was an important error to catch prior to powering the board; otherwise, the buck converter would have been shorted, drawn current well above its limit, and could quite possibly have damaged many of the other more expensive components.

The buck converter circuit also contained an incorrect schematic for the regulation feedback, caught before being powered and causing damage thanks to the team's faculty advisor. First, the board was being powered by the unregulated 3.3V pin, which could have produced voltages beyond the recommended tolerances of the other components. Second, the catch diode was connected to the wrong side of the inductor and would have been useless to reroute the flyback voltage. Given these errors, the buck converter was removed entirely, and the board was powered by a DC power supply from the VirtualBench through the 3.3V test point.

Software

In order to ensure that all the different modules of the project work in harmony, six test plans for the embedded software were developed. Because each test is long, the test plan has been reduced to a dependency chain—a sequence of events that should occur in no error occurs—with the paragraphs describing possible fixes. Note that some fixes are self-evident and require no further explanation.

Figure 44 shows the dependency chain for the Wi-Fi module. In this module, the ultimate error is a lack of connection to the access point. The BGAPI UART port should be configured to UART 1 on the microcontroller with an 8N1 format at 115200 baud. Furthermore, model control must be enabled, which can be configured in the "EK_TM4C123GXL.c" file's "uartTivaHWAttrs" configuration structure. Furthermore, every command send should also receive a response. If this does not occur, check that the data is on the line, the receive thread is processing it, and "wifi_cmd_response_semaphore" is being posted. If a response is being received, reference the WF 121 datasheet for payload outputs and possible error codes [66]. In addition, the access point to be connected to should have its SSID and password set to the SSID

and password in “wifi.hh”. Finally, the drone utilizes DHCP to get its IP address, so ensure the DHCP server is operating in the access point.

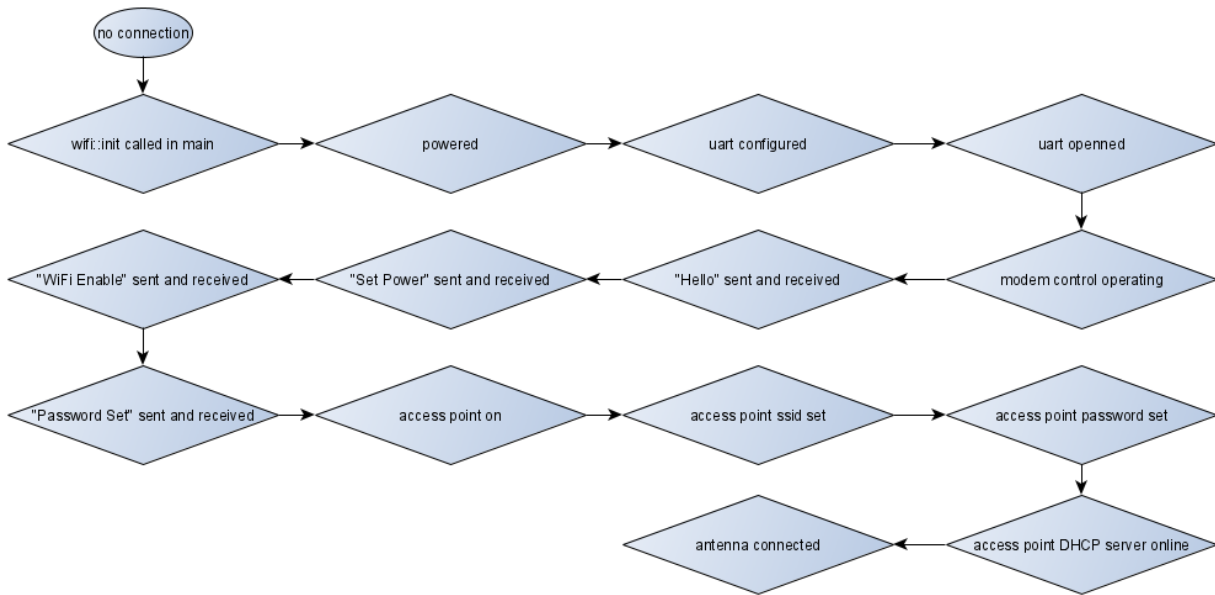


Figure 4444: Test Plan – Wi-Fi Module

Next, Figure 454545 defines the dependency chain for the simple drone communication protocol. An oscilloscope works best with this module, as it had tight timing constraints that will cause errors if not met. Therefore, reading inputs on the BGAPI UART line can help one understand the inputs to the system and debug from there. For example, both the mailbox *and* the semaphore need to be posted for the device to process data to send out.

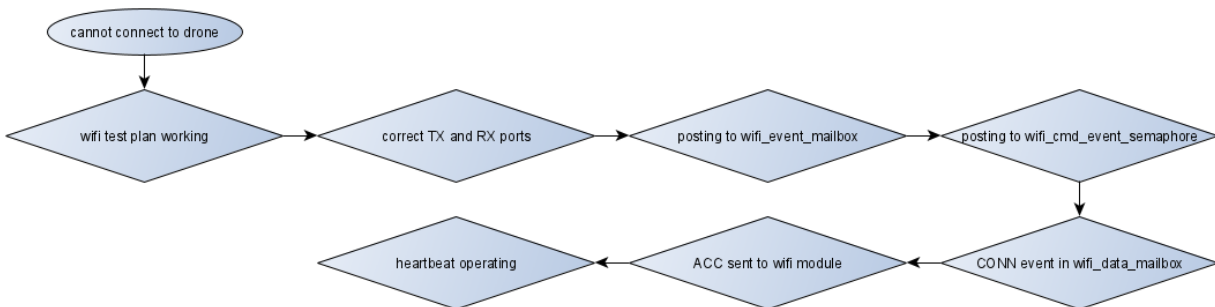


Figure 4545: Test Plan – SDCP

Figure 464646 then describes the dependency chain for the flight controller. Unlike the previous chains, this may require one to configure the flight controller in the Betaflight configurator. This will be needed to verify and/or reconfigure the flight controller to use the SUMD protocol, UART 3 is being used for Serial RX, and the receive format (AETR1234). To test for

new_joystick_data, the build option "" in "Build.h" will create a thread that continuously sets this value, allowing one to debug the flight controller module without any external module.

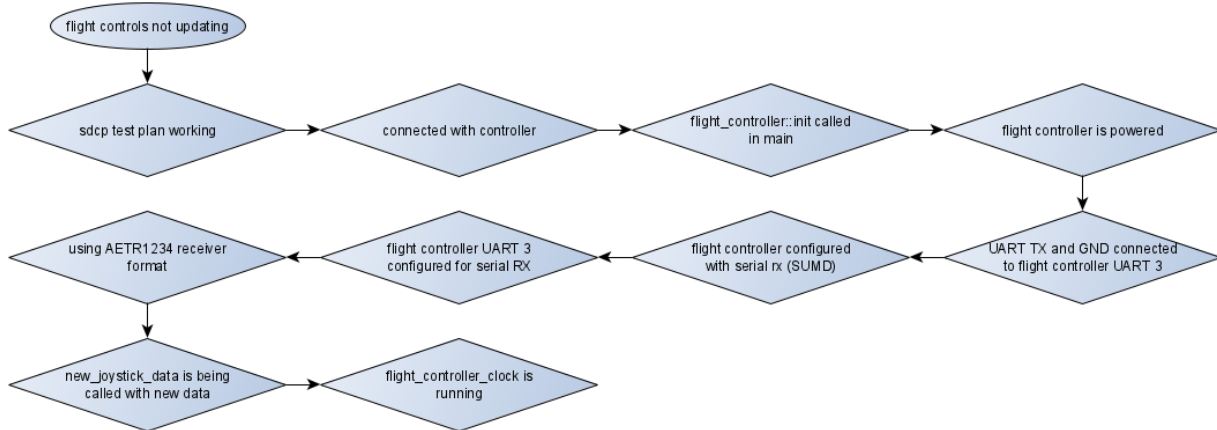


Figure 4646: Test Plan – Flight Controller

Figure 474747 moves on to testing the telemetry. As telemetry comes from the flight controller, the ports on the flight controller must be configured to output telemetry. Furthermore, the port must output the correct telemetry protocol (LTM). If telemetry data is going out and the UART is configured properly, then looking at the synchronization objects serves as the next step.

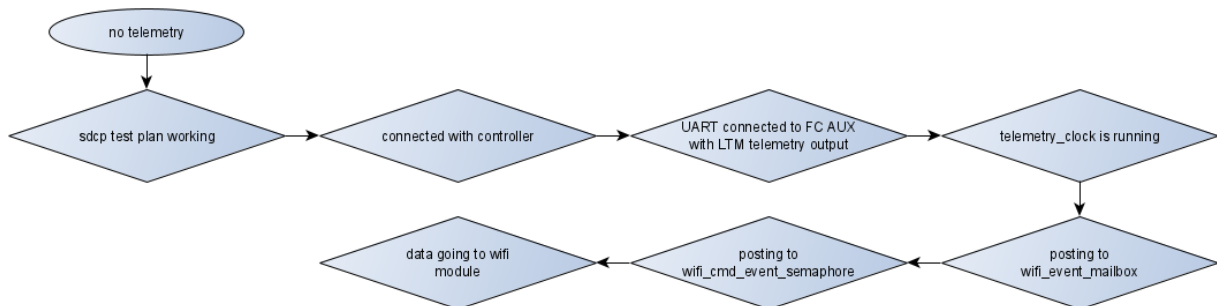


Figure 4747: Test Plan – Telemetry

Figure 48 describes the testing procedure for the GPS module if it were to fail. The GPS Module is a very simple system. It has a boot-up sequence of 39 NMEA sentences after which it starts periodically sending in 8 sentences per second. These sentences start with a '\$' character and end with '\r\n' sequence of bytes. There are threads that run to take this buffer of 8 sentences from the GPS Module, which is connected to UART 2 in the launchpad, and extract useful information from these sentences to a data structure that is much easy to read than the raw sentences. If at any point, the data would stop being streamed into the structure, the team would descend down the software stack from the top to the bottom to figure out what the problem is. The first test the team would do is to check if the GPS Module is receiving power or not. After that has been verified, the thread that is taking the raw sentences would be checked for incoming data and saving that data. Then, after that, the thread responsible for extraction would be checked. Both would have to be checked for different things like the stack size, the UART port,

etc. If both these threads were found to be functional, the team would then connect an oscilloscope to UART 2 and see if data is being sent from the Module.

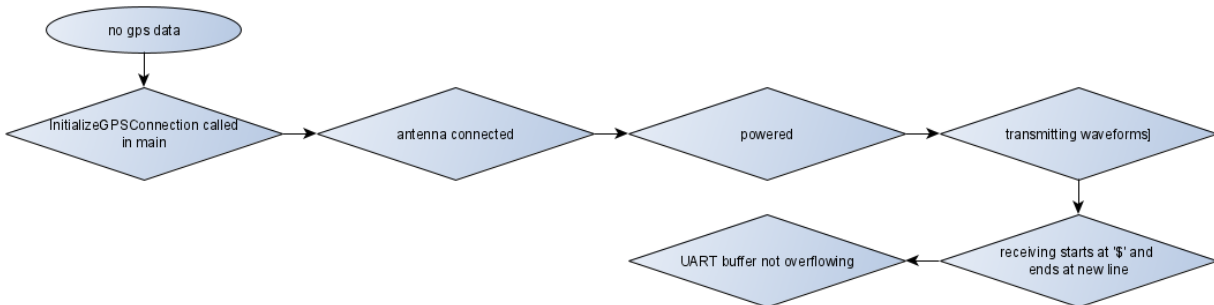


Figure 4848: Test Plan – GPS Module

Finally, Figure 494949 gives an overview for the dependency chain of the soil sensor. Unlike other modules, the soil sensor can operate between 5 V and 30 V, so the soil sensor operates purely on battery voltage. Note that although this is a RTU Modbus protocol, it communicates with an 8N1 format [86] instead of either 8N2 or 811 that the RTU format specified [68]. In addition, the soil sensor’s data sheet also specifies a factory-default baud rate of 4800 [86]. **This value is incorrect; the default baud rate is 9600!** Moreover, note that the RTU format requires 3.5-character times (about 4 milliseconds at 9600 baud) of no data transmission between transmission frames [68]. Finally, note that DE/RE GPIO pin should be pulled high just before UART data is sent, kept high while sending, and set low to receive data. This allows for the UART signal to be converted into an RS-485 signal and then for the RS-485 signal to be converted to UART for the response.

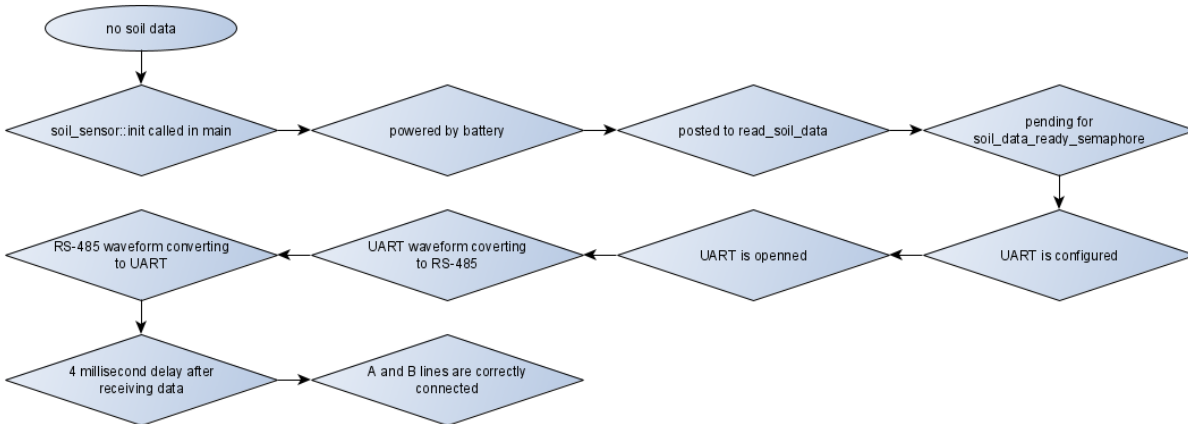


Figure 4949: Test Plan – Soil Sensor

Final Results

At the beginning of the semester, the team set out goals in key areas it expected to accomplish. However, the lack of experience in each area meant the steep learning curves were going to be a challenge to deal with alongside solving unforeseeable problems that take place during a research and development project. The team also had no familiarity with drones and did not expect the degree of experimentation needed to supplement the sparse datasheets. As a result, the areas that the team originally measured success by were wildly different than aspects of the project that took the most time to develop. Going strictly off of the metrics defined by the project proposal, the team did not meet the criteria for high success in drone flight, drone navigation, and soil sampling, although it did achieve the goals for drone interface. Much of the focus was around a stable autonomous flight, which was beyond the scope of a one semester class.

As of the last day of class, the drone was able to achieve tethered flight and could communicate with the soil sensor. The entire stack from frame to hardware to firmware to GUI were implemented to high degrees of success. Starting with the frame, the drone was able to transition back and forth between contracted (soil landing) and extended (normal landing) modes as well as support incorporate traditional design elements for standard drone hardware. It also securely housed all electronics with a custom design to fit objective specific components, including the solenoids and flight computer. For the flight computer PCB, the components that worked were the buck converter, RS485 transceiver, GPS module, and Wi-Fi module. Each was able to properly communicate with the microcontroller (power the MCU in the case of the buck converter). The MCU itself was able to correctly take the external data, process it in real time, and reroute necessary information. It established a solid serial connection with the flight computer, allowing control signals from the Wi-Fi module to be received by the flight controller. The flight controller was properly configured by Betaflight to convert the incoming signals into speed control signals for the ESC. The ESC was also properly configured by BLHeli_32 to spin the motors at the throttle and direction specified by the user's flight stick. The MCU also properly communicated with the Wi-Fi module to create a network for a user to connect to and send flight stick throttle and direction instructions through. Data sent back over the network from the flight computer included soil sensor data and GPS coordinates. Once received by the user's ground station, the data was displayed on the GUI in map view and numerical view.

In terms of a product that can be visually demonstrated without signal analysis, a user can first connect to the drone's network. By using an analog controller connected by USB [101], they can send roll, yaw, pitch, and throttle information wirelessly to the drone. The drone can then spin the propellers, with the speed of each of the four motors reflecting the commanded operation (e.g. reducing left motors' speed while maintaining right motors' speed to roll left). On the user's GUI, they can see NPK information sent by the soil sensor as well as the drone's GPS coordinates. The drone is able to stay in the normal extended position on its own, as seen in Figure 50, and transition to the contracted soil collecting position with assistance.

Given that communication from the user to the motor was established 30 minutes before the end of the final demonstration session, achieving flight was highly desirable but understood to be extremely unsafe. This was due to untested throttle responses, previous unexplained spontaneous

rotation from the motors, an untuned controller input, untrained pilots in suboptimal flying conditions (nighttime flight on a slightly inclined field within 200 ft of people, trees, and buildings), and no way to safely recover the drone should it lose connection while throttled to lifting speeds. However, the team compromised with a tethered flight where the drone's structural shafts were tied with paracord to stakes in the ground, with a team member in snowboarding protective gear (helmet, goggles, heavy duty jacket) ready to disconnect the battery should connection be lost. This resulted in a one second flight where the drone lifted up approximately 3-6" above the ground before tilting to a side and flipping over. Given the success of the drone lifting off of Earth by its own power, the team tried a second flight with maximum throttle to try to quickly fly up to the limits of the tethers to effectively remove any time the drone might lose stability before sustaining flight. This second flight yielded the same results as the first flight, only quicker.

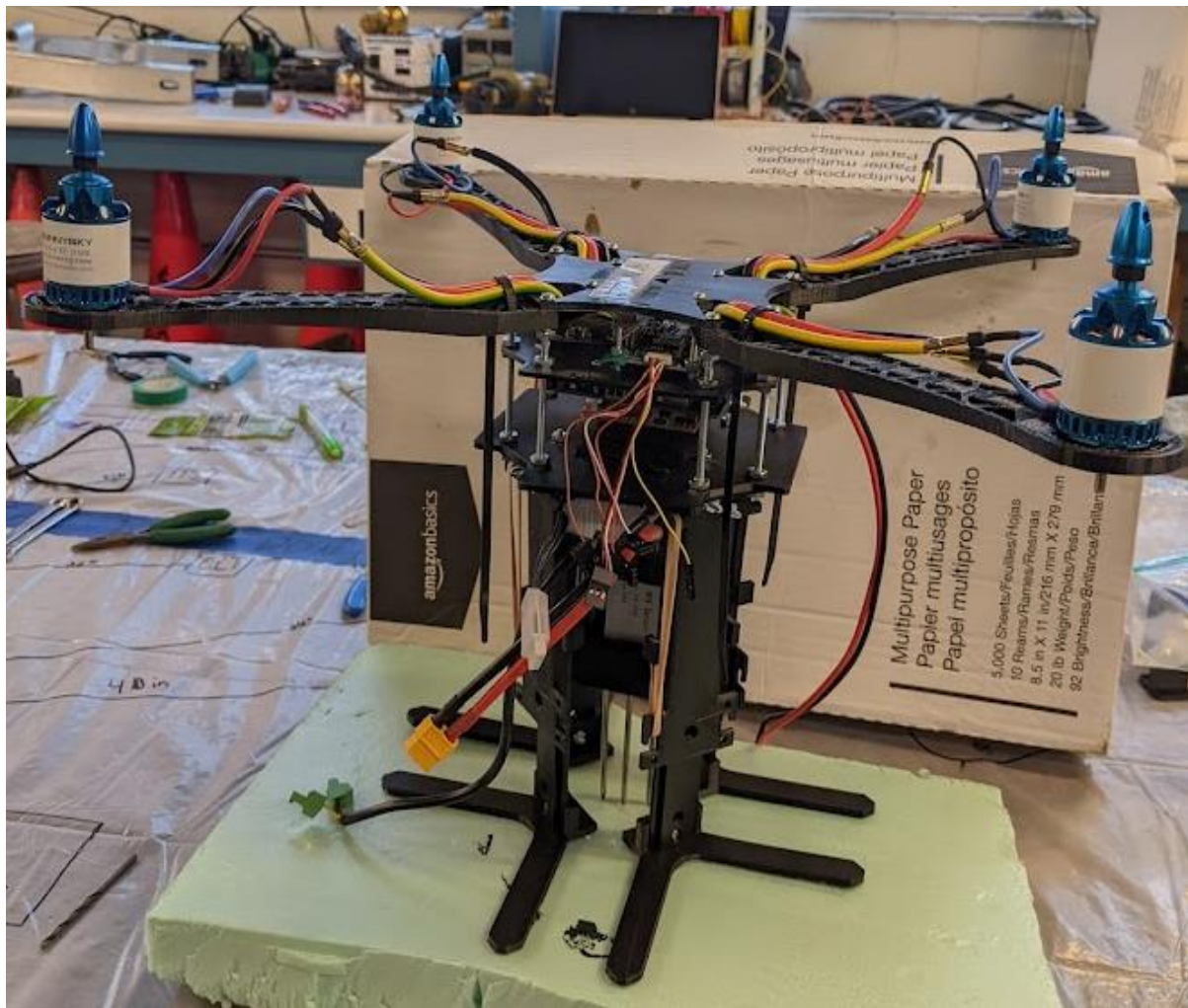


Figure 50: Final Drone Assembly

Costs

The budget for the capstone project was \$500; however, a misunderstanding of the manufacturing process and an additional board revision caused the team to significantly exceed this budget. The research and development of the drone cost \$764.94 as shown in Appendix 1, not including the 3D filament personally provided from team members. In order to cover the costs that exceeded the budget, the university was willing to cover \$53.56 more than it initially allotted, and the team funded the rest of the \$211.38. A majority of the budget went towards the drone hardware and high-powered wireless communication modules. The drone needed motors powerful enough to lift the standard weight plus the additional electronics, soil sensor, and actuating landing gear. A large enough battery was also needed to provide a useful flight time while providing a safety buffer of being able to supply more than the burst current of the combined motors. Furthermore, the university being a public institution placed a requirement of buying parts from American distributors. This both increased costs and limited the parts available to easily obtain.

One of the major budgeting mistakes came from the novelty of PCB manufacturing. While this error seems laughable in hindsight, the team expected that the assembly team at 3W would have common passive components in stock which was included in their quote. This was clearly not the case, and two iterations of the board saw over \$130 in passive components alone. The team also only budgeted for one board, and this naïve mistake would have to be doubled after all of the mistakes found on the first board. While the total cost of experimentation was \$764.94, the cost to build one new AgroFlight unit is \$580.61 as detailed in Appendix 2.

Appendix 2 also shows the price breakdown for building 10,000 units of AgroFlight. The reduced cost per unit is \$490.70, with much of the reduction coming from electronic components from the flight computer. However, there are several inaccuracies with this figure. The drone hardware price is not changed since it was bought from a hobbyist distributor that expects to sell just a unit or two to a customer. Buying these in bulk would likely result in significantly more savings, likely at least under \$400 if the prices are half of the single unit prices like the electronic components. Additionally, the price reflects replacing the MCU development board with its discrete chip. This transition would require significantly more engineering work and more electronic components to make it work properly. The price also doesn't consider the process of mass scale manufacturing. PCB production, 3D printing, and assembly costs would likely decrease, and although that information available freely online, estimates can be made. Advanced Circuits sold their PCBs at \$33 per single unit. JLCPCB sold theirs at \$2 per pack of five boards. The bulk order assumed Advance's original price, although it should be able to come down to under \$10 just to stay competitive. 3D printing costs also go significantly down from \$32 per unit to \$10 per unit at 10,000 units [87]. However, this is at the outsourcing price; scaling up in-house operations would need investigation of a bulk equipment investment, which isn't publicly available.

Future Work

Out of all the components to this project, the soil sampling and drone communication protocol share a great potential for expansion and/or improvement. Controlling a drone through the common Wi-Fi protocol implies a connection between the device and the Internet. This implication could result in great improvements (remote data centers storing soil data, extremely remote piloting, etc.) but also carries a great risk. For example, the Simple Drone Communication Protocol (SDCP) was designed with some security aspects in mind, but not enough to be considered a secure method of communication. If communication packets were to be identified by some other method, then drones operating with this protocol would be protected against forgery attacks.

Furthermore, hardware changes and implementations could be improved that would create a more capable drone. First, endpoint streaming from the Wi-Fi module, the ability to stream data to and from a UDP/TCP connection on a different UART port than the BGAPI port [66], was a selling feature of the module, but its implementation was cut when time constraints became a factor. If this feature was implemented, and implemented well, it could reduce load on the CPU by removing the handshaking between the Wi-Fi handler thread and the SDCP handler thread. In addition, the flight controller supported GPS data, but would need a 5 V UART input to forward this information [88], a voltage level not possible for the TM4C. If an amplifier were included, then a UART connection from the microcontroller could forward this data to the flight controller, enabling GPS-related navigation. Lastly, the flight controller used, the F3 Acro, became outdated after Betaflight 4.0 [89].

With any project, reflection reveals issues that became important, ideals too steep to achieve, common patterns in development, and schedules that would have worked better. Many of these suggestions will be discussed, but any group or individual wanting to build a drone *should get the flight controller as soon as possible*. Unlike integrate circuits, flight controllers do not have well-documented data sheets and easily-defined protocols. Therefore, to understand what you can do, you must first understand if the device you have can support it. Thus, get this device as soon as one can, less you promise something that the device could no longer achieve.

When developing embedded code, testing techniques and programming patterns proved vital for completing features in a reasonable time. First, data streams need to know when they have “lost track”, when a data frame has been shifted out of position. This error was most prevalent in the Wi-Fi feature and was commonly because breakpoints in a receive loop would cause data to be dropped from the internal UART FIFO. Therefore, debugging responses from a data stream should be outside of that data stream’s receive thread, and the receive thread should be able to reject data until it returns back on track. In addition to breakpoints, understand how signals are sent over the metal, as reading these signals not only preserves timing constraints, but can be used to understand the state of the device.

In the design phase of any project, a solution that looks simple then could become a nightmare later on. However, if some else has either provided a library to use for your module or a narrative about developing the same or a similar feature, one can worry less of that aspect becoming a problem. In this project, the transmission method to the flight controller (SUMD)

came from an article written by Erich Styger [69] about his endeavors of connecting a flight controller to a microcontroller. With this article, SUMD proved to be simple and quick to implement and test.

Lastly, know what has to be done, what should be done, and how to work within a schedule. Aspects crucial to the operation (and grading) of one's project should be completed first and ensured to operate without fail. Aspects that should be done (either to make the system better or to maximize one's grade) needs to be developed during downtime or after crucial aspects become set in stone. If you go over the Gantt chart's timeframe, make crucial code ugly but functional and make additional code quick or non-existent. Software can drag one into hours of debugging when a simple, yet dumb solution could have worked as well and tested in much less time.

References

- [1] “Careers in Soil Science | NRCS Soils.” [Online]. Available: https://www.nrcs.usda.gov/wps/portal/nrcs/detail/soils/edu/?cid=nrcs142p2_054277. [Accessed: 15-Dec-2021]
- [2] “Sampling Soils for Fertilizer and Lime Recommendations and Frequency of Soil Sampling (E0498),” *MSU Extension*. [Online]. Available: https://www.canr.msu.edu/resources/farm_soil_sampling. [Accessed: 15-Dec-2021]
- [3] “Drone (UAV) Solution for Inspection,” *Airobotics*. [Online]. Available: <https://www.airoboticsdrones.com/applications/inspection/>. [Accessed: 15-Dec-2021]
- [4] “DJI Agras T30-A New Digital Flagship for Agriculture,” *DJI*. [Online]. Available: <https://www.dji.com/photo>. [Accessed: 15-Dec-2021]
- [5] “12 Companies Using AI In Drones | Built In.” [Online]. Available: <https://builtin.com/artificial-intelligence/drones-ai-companies>. [Accessed: 15-Dec-2021]
- [6] A. Meola, “Precision agriculture in 2021: The future of farming is using drones and sensors for efficient mapping and spraying,” *Business Insider*. [Online]. Available: <https://www.businessinsider.com/agricultural-drones-precision-mapping-spraying>. [Accessed: 15-Dec-2021]
- [7] Texas Instruments, “Tiva TM4C123GH6PM Microcontroller,” 2014. [Online]. Available: <https://www.ti.com/lit/ds/symlink/tm4c123gh6pm.pdf?ts=1639519619828>. [Accessed 14 12 2021].
- [8] Texas Instruments, “Tiva C Series TM4C123G LaunchPad Evaluation Board User's Guide,” April 2013. [Online]. Available: <https://www.ti.com/lit/ug/spmu296/spmu296.pdf?ts=1639540084196>. [Accessed 14 12 2021].
- [9] Texas Instruments, “TI-RTOS 2.16 for TivaC Getting Started Guide,” February 2016. [Online]. Available: <https://www.ti.com/lit/ug/spruhu5d/spruhu5d.pdf?ts=1639540353257>. [Accessed 14 12 2021].
- [10] “Product.” [Online]. Available: <https://www.ni.com/en-us/shop/software/products/multisim.html>. [Accessed: 15-Dec-2021]
- [11] “Product.” [Online]. Available: <https://www.ni.com/en-us/shop/software/products/ultiboard.html>. [Accessed: 15-Dec-2021]
- [12] “CCSTUDIO IDE, configuration, compiler or debugger | TI.com.” [Online]. Available: <https://www.ti.com/tool/CCSTUDIO>. [Accessed: 15-Dec-2021]
- [13] T. Q. Company, “Embedded Software Development Tools | Cross Platform IDE | Qt Creator.” [Online]. Available: <https://www.qt.io/product/development-tools>. [Accessed: 15-Dec-2021]
- [14] “Fusion 360 | 3D CAD, CAM, CAE & PCB Cloud-Based Software | Autodesk.” [Online]. Available: <https://www.autodesk.com/products/fusion-360/overview>. [Accessed: 15-Dec-2021]

- [15] “Ultimaker Cura: Powerful, easy-to-use 3D printing software,” *ultimaker.com*. [Online]. Available: <https://ultimaker.com/software/ultimaker-cura>. [Accessed: 15-Dec-2021]
- [16] “USA PCB Manufacturer & Assembly | Advanced Circuits.” [Online]. Available: <https://www.4pcb.com/>. [Accessed: 15-Dec-2021]
- [17] “PCB Prototype & PCB Fabrication Manufacturer - JLCPCB.” [Online]. Available: <https://jlcpcb.com/>. [Accessed: 15-Dec-2021]
- [18] “WWW Electronics Incorporated.” [Online]. Available: <https://3welec.com/>. [Accessed: 15-Dec-2021]
- [19] “Ender-3 Pro 3D Printer.” [Online]. Available: <https://www.creality.com/goods-detail/ender-3-pro-3d-printer>. [Accessed: 15-Dec-2021]
- [20] “Tevo Tarantula 3D Printer Kit with 2 Free Rolls of Filament,” *Tevo USA*. [Online]. Available: <https://www.tevousa.com/products/tevo-tarantula-3d-printer-kit-with-2-free-rolls-of-filament>. [Accessed: 15-Dec-2021]
- [21] “Why is there a chip shortage?,” *BBC News*, 26-Aug-2021 [Online]. Available: <https://www.bbc.com/news/business-58230388>. [Accessed: 15-Dec-2021]
- [22] “Agriculture and Farming Equipment | John Deere US.” [Online]. Available: <https://www.deere.com/en/agriculture/>. [Accessed: 15-Dec-2021]
- [23] “Attachments, Implements and Parts Search Lookup | John Deere US.” [Online]. Available: <https://www.deere.com/en/attachments-accessories-and-implements/attachments-search-tool/#/US/en/lawn-garden/gator-utility-vehicles?type=Ag%20Management%20Solutions&p=1&sort=price%3AASC>. [Accessed: 15-Dec-2021]
- [24] “Soil NPK Sensor For Soil Nutrients in Agriculture - Renke.” [Online]. Available: <https://www.renkeer.com/product/soil-npk-sensor/>. [Accessed: 15-Dec-2021]
- [25] “How to Register Your Drone.” [Online]. Available: https://www.faa.gov/uas/getting_started/register_drone/. [Accessed: 15-Dec-2021]
- [26] “Become a Drone Pilot.” [Online]. Available: https://www.faa.gov/uas/commercial_operators/become_a_drone_pilot/. [Accessed: 15-Dec-2021]
- [27] “eCFR :: 14 CFR Part 107 -- Small Unmanned Aircraft Systems.” [Online]. Available: <https://www.ecfr.gov/current/title-14/chapter-I/subchapter-F/part-107>. [Accessed: 15-Dec-2021]
- [28] “[Policy Directory.” [Online]. Available: <https://uvapolicy.virginia.edu/policy/SEC-040#Statement>. [Accessed: 15-Dec-2021]
- [29] “Operating a UAS at UVA | Research.” [Online]. Available: <https://research.virginia.edu/unmanned-aircraft-uas/operating-uas-uva>. [Accessed: 15-Dec-2021]
- [30] “What Is VirtualBench?” [Online]. Available: <https://www.ni.com/en-us/shop/electronic-test-instrumentation/virtualbench/what-is-virtualbench.html>. [Accessed: 15-Dec-2021]

- [31] “CCSTUDIO IDE, configuration, compiler or debugger | TI.com.” [Online]. Available: <https://www.ti.com/tool/CCSTUDIO>. [Accessed: 15-Dec-2021]
- [32] E. Staff, “Introduction to Rate Monotonic Scheduling,” *Embedded.com*, 28-Feb-2002. [Online]. Available: <https://www.embedded.com/introduction-to-rate-monotonic-scheduling/>. [Accessed: 15-Dec-2021]
- [33] Available: https://dev.ti.com/tirex/explore/node?node=ALw..F84P5oOmnCUM7zGVQ__ge-t7a__LATEST. [Accessed: 15-Dec-2021]
- [34] “cplusplus.com - The C++ Resources Network.” [Online]. Available: <https://www.cplusplus.com/>. [Accessed: 15-Dec-2021]
- [35] “About Qt - Qt Wiki.” [Online]. Available: https://wiki.qt.io/About_Qt. [Accessed: 15-Dec-2021]
- [36] Microsoft, “Explore Windows 11 OS, Computers, Apps, & More | Microsoft,” *Windows*. [Online]. Available: <https://www.microsoft.com/en-us/windows>. [Accessed: 15-Dec-2021]
- [37] “Linux.org,” *Linux.org*. [Online]. Available: <https://www.linux.org/>. [Accessed: 15-Dec-2021]
- [38] “QML Applications | Qt 5.15.” [Online]. Available: <https://doc.qt.io/qt-5/qmlapplications.html>. [Accessed: 15-Dec-2021]
- [39] “JSON.” [Online]. Available: <https://www.json.org/json-en.html>. [Accessed: 15-Dec-2021]
- [40] “JavaScript.com.” [Online]. Available: <https://www.javascript.com/>. [Accessed: 15-Dec-2021]
- [41] “Qt QML 5.15.7.” [Online]. Available: <https://doc.qt.io/qt-5/qtqml-index.html>. [Accessed: 15-Dec-2021]
- [42] D. H. P. Kang, M. Chen, and O. A. Ogunseitan, “Potential Environmental and Human Health Impacts of Rechargeable Lithium Batteries in Electronic Waste,” *Environ. Sci. Technol.*, vol. 47, no. 10, pp. 5495–5503, May 2013, doi: 10.1021/es400614y. [Online]. Available: <https://doi.org/10.1021/es400614y>. [Accessed: 15-Dec-2021]
- [43] O. US EPA, “Used Lithium-Ion Batteries,” 16-May-2019. [Online]. Available: <https://www.epa.gov/recycle/used-lithium-ion-batteries>. [Accessed: 15-Dec-2021]
- [44] “Welcome | USPS.” [Online]. Available: <https://www.usps.com/>. [Accessed: 15-Dec-2021]
- [45] “Department of Transportation.” [Online]. Available: <https://www.transportation.gov/>. [Accessed: 15-Dec-2021]
- [46] “Transporting Lithium Batteries | PHMSA.” [Online]. Available: <https://www.phmsa.dot.gov/lithiumbatteries>. [Accessed: 15-Dec-2021]
- [47] C. McGrady, “Brushless vs Brushed Motors.” [Online]. Available: <https://www.arrow.com/en/research-and-events/articles/which-dc-motor-is-best-for-your-application>. [Accessed: 15-Dec-2021]

- [48] N. Corporation, “Role of Brushless DC Motors in Environmental Conservation,” *Nidec Corporation*. [Online]. Available: <https://www.nidec.com/en/sustainability/environment/system/motor/brushless/>. [Accessed: 15-Dec-2021]
- [49] “ABS plastic recycling – everything you need know | 3DRIFIC.” [Online]. Available: <https://3drific.com/abs-plastic-recycling-everything-you-need-know/>. [Accessed: 15-Dec-2021]
- [50] “Recreational Flyers & Modeler Community-Based Organizations.” [Online]. Available: https://www.faa.gov/uas/recreational_fliers. [Accessed: 15-Dec-2021]
- [51] “Certified Remote Pilots including Commercial Operators.” [Online]. Available: https://www.faa.gov/uas/commercial_operators. [Accessed: 15-Dec-2021]
- [52] A. Meola, “Precision agriculture in 2021: The future of farming is using drones and sensors for efficient mapping and spraying,” *Business Insider*. [Online]. Available: <https://www.businessinsider.com/agricultural-drones-precision-mapping-spraying>. [Accessed: 15-Dec-2021]
- [53] “Free CAD Designs, Files & 3D Models | The GrabCAD Community Library.” [Online]. Available: <https://grabcad.com/library/s500-frame-1>. [Accessed: 15-Dec-2021]
- [54] “Betaflight.” [Online]. Available: <https://github.com/betaflight/betaflight>. [Accessed: 15-Dec-2021]
- [55] “IEEE 802.11, The Working Group Setting the Standards for Wireless LANs.” [Online]. Available: <https://www.ieee802.org/11/>. [Accessed: 15-Dec-2021]
- [56] “UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter | Analog Devices.” [Online]. Available: <https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html>. [Accessed: 15-Dec-2021]
- [57] “WF121 - Silicon Labs.” [Online]. Available: <https://www.silabs.com/wireless/wi-fi/bluegiga-legacy-modules/device.wf121>. [Accessed: 15-Dec-2021]
- [58] “What is User Datagram Protocol (UDP)? Definition from SearchNetworking,” *SearchNetworking*. [Online]. Available: <https://www.techtarget.com/searchnetworking/definition/UDP-User-Datagram-Protocol>. [Accessed: 15-Dec-2021]
- [59] “Multi-GNSS Receiver Module GN-87 | GPS/GNSS Modules | Products | FURUNO.” [Online]. Available: <https://www.furuno.com/en/products/gnss-module/GN-87>. [Accessed: 15-Dec-2021]
- [60] “NMEA-0183 Sentences for GPS Receivers.” [Online]. Available: <https://w3.cs.jmu.edu/bernstdh/web/common/help/nmea-sentences.php>. [Accessed: 15-Dec-2021]
- [61] “Full Guide to Serial Communication Protocol and Our RS-485 |.” [Online]. Available: <https://www.maximintegrated.com/en/design/technical-documents/app-notes/3/3884.html>. [Accessed: 15-Dec-2021]

- [62] “STL File Format: Everything You Need to Know.” [Online]. Available: <https://all3dp.com/1/stl-file-format-3d-printing/>. [Accessed: 15-Dec-2021]
- [63] “Lacy Hall – Student Experiential Center.” [Online]. Available: <http://lacy.seas.virginia.edu/>. [Accessed: 15-Dec-2021]
- [64] “STM32F070RB - Mainstream Arm Cortex-M0 Value line MCU with 128 Kbytes of Flash memory, 48 MHz CPU, USB - STMicroelectronics.” [Online]. Available: <https://www.st.com/en/microcontrollers-microprocessors/stm32f070rb.html>. [Accessed: 15-Dec-2021]
- [65] “EK-TM4C123GXL Evaluation board | TI.com.” [Online]. Available: <https://www.ti.com/tool/EK-TM4C123GXL>. [Accessed: 15-Dec-2021]
- [66] “LM2596 data sheet, product information and support | TI.com.” [Online]. Available: <https://www.ti.com/product/LM2596>. [Accessed: 15-Dec-2021]
- [67] “DO5022P-153 | Coilcraft.” [Online]. Available: <https://www.coilcraft.com/>. [Accessed: 15-Dec-2021]
- [68] “Gravity__VL53L0X_ToF_Laser_Range_Finder_SKU_SEN0245-DFRobot.” [Online]. Available: https://wiki.dfrobot.com/Gravity__VL53L0X_ToF_Laser_Range_Finder_SKU_SEN0245. [Accessed: 15-Dec-2021]
- [69] “VL53L0X - Time-of-Flight ranging sensor - STMicroelectronics.” [Online]. Available: <https://www.st.com/en/imaging-and-photonics-solutions/vl53l0x.html>. [Accessed: 15-Dec-2021]
- [70] “tkinter — Python interface to Tcl/Tk — Python 3.10.1 documentation.” [Online]. Available: <https://docs.python.org/3/library/tkinter.html>. [Accessed: 15-Dec-2021]
- [71] “Welcome to Python.org,” *Python.org*. [Online]. Available: <https://www.python.org/>. [Accessed: 15-Dec-2021]
- [72] “Dear ImGui.” [Online]. Available: <https://github.com/ocornut/imgui>. [Accessed: 15-Dec-2021]
- [73] “Plane Spotter (QML) | Qt Location 5.15.7.” [Online]. Available: <https://doc.qt.io/qt-5/qtlocation-planespotter-example.html>. [Accessed: 15-Dec-2021]
- [74] T. Q. Company, “Software Licensing | Open Source Community | Qt.” [Online]. Available: <https://www.qt.io/licensing/>. [Accessed: 15-Dec-2021]
- [75] “QThread Class | Qt Core 5.15.7.” [Online]. Available: <https://doc.qt.io/qt-5/qthread.html>. [Accessed: 15-Dec-2021]
- [76] “eCalc - xcopterCalc - the most reliable Multicopter Calculator on the Web.” [Online]. Available: <https://www.ecalc.ch/xcoptercalc.php>. [Accessed: 15-Dec-2021]
- [77] “All about Multirotor FPV Drone Propellers,” *GetFPV Learn*, 03-Feb-2018. [Online]. Available: <https://www.getfpv.com/learn/new-to-fpv/all-about-multirotor-fpv-drone-propellers/>. [Accessed: 15-Dec-2021]

- [78] J. Reid, "Understanding Kv Ratings," *RotorDrone*, 29-Nov-2016. [Online]. Available: <https://www.rotordronepro.com/understanding-kv-ratings/>. [Accessed: 15-Dec-2021]
- [79] J. Reid, "C Rating for Drone LiPo Battery Packs," *RotorDrone*, 22-Dec-2017. [Online]. Available: <https://www.rotordronepro.com/c-rating-drone-lipo-battery-packs/>. [Accessed: 15-Dec-2021]
- [80] "BLHeli." [Online]. Available: <https://github.com/bitdump/BLHeli>. [Accessed: 15-Dec-2021]
- [81] "BLHeli/BLHeli_32 ARM at master · bitdump/BLHeli," *GitHub*. [Online]. Available: <https://github.com/bitdump/BLHeli>. [Accessed: 15-Dec-2021]
- [82] Silicon Labs, "WF121 Wi-Fi Software," 30 September 2021. [Online]. Available: <https://www.silabs.com/documents/public/reference-manuals/WF121-WiFi-Software-4.0-API-RM.pdf>.
- [83] Shandong Renke Control Technology Co., *Soil nitrogen, phosphorus, potassium three-in-one fertility sensor (Type 485) user's Guide*.
- [84] Modbus.org, "MODBUS over Serial Line," 2 12 2002. [Online]. Available: https://www.modbus.org/docs/Modbus_over_serial_line_V1.pdf. [Accessed 2021].
- [85] "High Volume 3D Printing - Order Bulk 3D Printing - SD3D Printing." [Online]. Available: <https://www.sd3d.com/3d-printing/volume-manufacturing-quote/>. [Accessed: 15-Dec-2021]
- [86] D. Clifton, "Speriously Pro Racing F3 Flight Controller," 2015. [Online]. Available: <http://www.seriouslypro.com/files/SPRacingF3-Manual-latest.pdf>. [Accessed 2021].
- [87] Montis, "Betaflight: The end of F3 flight controllers," *Multirotor Guide*, 3 1 2019. [Online]. Available: <https://www.multirotorguide.com/news/betaflight-the-end-of-f3-flight-controllers/>. [Accessed 2021].
- [88] E. Styger, "Kinetis Drone: Remote Controller with SUMD," *DZone*, 3 November 2015. [Online]. Available: <https://dzone.com/articles/kinetis-drone-remote-controller-with-sumd>. [Accessed 2021].
- [89] "eCFR :: 47 CFR Part 95 -- Personal Radio Services." [Online]. Available: <https://www.ecfr.gov/current/title-47/chapter-I/subchapter-D/part-95>. [Accessed: 15-Dec-2021]
- [90] Available: <https://oehha.ca.gov/proposition-65/about-proposition-65>. [Accessed: 15-Dec-2021]
- [91] IPC, "Qualification and Performance Specification for Rigid Printed Boards." [Online]. Available: <https://www.ipc.org/TOC/IPC-6012D.pdf>. [Accessed: 15-Dec-2021]
- [92] L. Iantosca, "IPC-6012 or IPC-A-600: Which Standard Should You Use?," *Sierra Circuits*, 21-Oct-2020. [Online]. Available: <https://www.protoexpress.com/blog/ipc-6012-ipc-600-standard-use/>. [Accessed: 15-Dec-2021]

- [93] “IPC-A-600 Acceptability of Printed Boards Endorsement Program,” *IPC International, Inc.*, 03-Aug-2020. [Online]. Available: <https://www.ipc.org/ipc-600-acceptability-printed-boards-endorsement-program>. [Accessed: 15-Dec-2021]
- [94] “Serial Communication - learn.sparkfun.com.” [Online]. Available: <https://learn.sparkfun.com/tutorials/serial-communication/uarts>. [Accessed: 15-Dec-2021]
- [95] “2022 RoHS Compliance Guide: Regulations, 10 Substances, Exemptions.” [Online]. Available: <https://www.rohsguide.com/>. [Accessed: 15-Dec-2021]
- [96] J. Kelly, “RS-485 Serial Interface Explained,” *CUI Devices*. [Online]. Available: <https://www.cuidevices.com/blog/rs-485-serial-interface-explained>. [Accessed: 16-Dec-2021]
- [97] “Modicon Modbus Protocol Reference Guide,” *MODICON, Inc.* [Online]. Available: https://www.modbus.org/docs/PI_MBUS_300.pdf. [Accessed: 16-Dec-2021]
- [98] “U.FL Series,” *Hirose Electric Co., Ltd.* [Online]. Available: <https://www.hirose.com/product/series/U.FL>. [Accessed: 16-Dec-2021]
- [99] “MAX485,” *Maxim Integrated*. [Online]. Available: <https://www.maximintegrated.com/en/products/interface/transceivers/MAX485.html>. [Accessed: 16-Dec-2021]
- [100] “VN3205,” *Microchip*. [Online]. Available: <https://www.microchip.com/en-us/product/VN3205>. [Accessed: 16-Dec-2021]
- [101] “USB,” *USB*. [Online]. Available: <https://www.usb.org/>. [Accessed: 16-Dec-2021]
- [102] Y. Liu, “Drone for agriculture,” *United States Department of Commerce*. [Online]. Available: <https://patents.google.com/patent/US10364029B2/>. [Accessed: 16-Dec-2021]
- [103] V. Salnikov, A. Filin, H. Burema, “Hybrid airship-drone farm robot system for crop dusting, planting, fertilizing and other field jobs,” *United States Department of Commerce*. [Online]. Available: <https://patents.google.com/patent/US9852644B2/>. [Accessed: 16-Dec-2021]
- [104] J. Lee, “Soil sampling drone,” *United States Department of Commerce*. [Online]. Available: <https://patents.google.com/patent/KR101845395B1/en/>. [Accessed: 16-Dec-2021]
- [105] “Embedded C Coding Standard,” *Barr Group Software Experts*. [Online]. Available: <https://barrgroup.com/embedded-systems/books/embedded-c-coding-standard>. [Accessed: 16-Dec-2021]

Appendix

Appendix 1: Parts Ordered

Item	Unit Price	Quantity	Budget Remaining	Link
Initial Budget	\$500		\$500	
Microcontroller	\$15.59	1		https://www.digikey.com/en/products/detail/texas-instruments/EK-TM4C123GXL/3996736
MCU Order	\$15.59		\$484.41	
GPS Receiver	\$37.28	1		https://www.digikey.com/en/products/detail/kaga-electronics-usa/GN-8720/7491503
GPS Antenna	\$2.77	1		https://www.digikey.com/en/products/detail/molex/2065600100/9094612
GPS Antenna Connector	\$1.22	1		https://www.digikey.com/en/products/detail/hirose-electric-co-ltd/U-FL-R-SMT-1-40/4285474
RS485 Transceiver	\$3.68	1		https://www.digikey.com/en/products/detail/maxim-integrated/MAX485ESA/1495323
IR Distance Sensor	\$12.90	1		https://www.digikey.com/en/products/detail/dfrobot/SEN0245/8827827
Compass	\$1.49	1		https://www.digikey.com/en/products/detail/silicon-labs/SI7210-B-03-IV/7323050
Buck Converter	\$2.67	1		https://www.digikey.com/en/products/detail/texas-instruments/LM2596SX-3-3-NOPB/366832
Wire	\$6.25	1		https://www.digikey.com/en/products/detail/dfrobot/FIT0582/9559251
Digikey Order 1	\$68.26		\$416.15	
Propellor	\$6.88	2		https://www.buddyrc.com/collections/10-inch-prop/products/carbon-fiber-slow-flyer-10x4-7-inch-props-normal-and-reverse?variant=31974559187030
Motor	\$22.80	4		https://www.buddyrc.com/collections/sunnysky-x-v3-motors/products/sunnysky-x2216-v3-brushless-motors-long-shaft-version?variant=33002331144278
Battery	\$46.99	1		https://www.buddyrc.com/collections/3s-11-1v-1/products/gens-ace-11-1v-50c-3s-4000mah-lipo-battery-pack-with-xt60-plug?variant=37942307455164
ESC	\$32.95	1		https://www.buddyrc.com/collections/quad-esc/products/hakrc-35a-blheli_32-2-5s-4in1-brushless-esc
Flight Controller	\$9.99	1		https://www.buddyrc.com/collections/quad-fc/products/f3-acro-flight-controller
PDB	\$3.95	1		https://www.buddyrc.com/products/kensun-power-distribution-board-with-5v-12v-output?variant=30735600418902
Li-Po Charger	\$35	1		https://www.buddyrc.com/collections/all-battery-chargers/products/isdt-q6-nano-battgo-200w-8a-

				colorful-pocket-battery-balance-charger-for-1-6s-lipo-battery?variant=32636734079062
Charging Bag	\$4.98	1		https://www.buddyrc.com/collections/lipo-bag/products/fireproof-explosionproof-lipo-battery-safe-bag-lipo-battery-guard-185x75x60mm?variant=30735650062422
Battery Strap	\$0.99	2		https://www.buddyrc.com/collections/battery-straps/products/20cm-coated-glacier-battery-strap-1
Motor Connector	\$1.49	4		https://www.buddyrc.com/collections/connector/products/3-5mm-gold-plated-bullet-connector-3-pairs?variant=30735582888022
BuddyRC Order 1	\$246.76		\$169.39	
Battery Connector Splitter	\$8.99	1		https://www.amazon.com/dp/B07DKXQLKT
Battery Connector Leads	\$6.49	1		https://www.amazon.com/dp/B07QH249CR/
Soil Sensor	\$83.66	1		https://www.amazon.com/dp/B0836WYNJ1/
Amazon Order 1	\$99.14		\$70.25	
Wi-Fi Module	\$21.46	2		https://www.mouser.com/ProductDetail/Silicon-Labs/WF121-E-V2C?qs=8HqIXBaaO6yQS%2FkPjITxCw%3D%3D
Mouser Order 1	\$42.92		\$27.33	
Solenoid	\$13.86	2		https://www.digikey.com/en/products/detail/ledex-dormeyer-saia-a-division-of-johnson-electric/B17-L-155-B-3/9676166
MOSFET	\$1.37	2		https://www.digikey.com/en/products/detail/infineon-technologies/BSZ031NE2LS5ATMA1/5845204?
Diode	\$0.39	2		https://www.digikey.com/en/products/detail/nexperia-usa-inc/PMEG4005ET-215/1157707
Resistor	\$0.10	2		https://www.digikey.com/en/products/detail/yageo/RC1206FR-074K7L/728887
Connector	\$0.17	2		https://www.digikey.com/en/products/detail/harwin-inc/M20-1060200/3728191
Inductor	\$1.85	1		https://www.digikey.com/en/products/detail/pulse-electronics-power/P0250-153NL/5436370
Capacitor	\$1.64	2		https://www.digikey.com/en/products/detail/kyocera-avx/TPSC337K006R0080/3068585
Diode	\$0.68	1		https://www.digikey.com/en/products/detail/vishay-general-semiconductor-diodes-division/VS-30WQ03FNTR-M3/4933681
Connector	\$0.98	1		https://www.digikey.com/en/products/detail/phoenix-contact/1844210/349195
Connector	\$1.25	1		https://www.digikey.com/en/products/detail/on-shore-technology-inc/OSTTJ0211530/1588359
Resistor	\$0.10	4		https://www.digikey.com/en/products/detail/yageo/RC1206FR-074K7L/728887
Connector	\$0.18	2		https://www.digikey.com/en/products/detail/adam-tech/PH1-10-UA/9830653

Capacitor	\$0.22	1		https://www.digikey.com/en/products/detail/samsung-electro-mechanics/CL31A106MAHNNNE/3886839
Capacitor	\$0.22	1		https://www.digikey.com/en/products/detail/samsung-electro-mechanics/CL31A106MAHNNNE/3886839
Resistor	\$0.10	2		https://www.digikey.com/en/products/detail/panasonic-electronic-components/ERJ-8GEYJ393V/21309
Resistor	\$0.10	1		https://www.digikey.com/en/products/detail/stackpole-electronics-inc/RMCF1206JT51R0/1753854
Capacitor	\$4.45	1		https://www.digikey.com/en/products/detail/samsung-electro-mechanics/CL31A106MAHNNNE/3886839
Connector	\$0.22	1		https://www.digikey.com/en/products/detail/3m/N2520-6V0C-RB-WE/1125437
Connector	\$0.13	1		https://www.digikey.com/en/products/detail/metz-connect-usa-inc/PR20203VBDN/12342894
Resistor	\$0.10	3		https://www.digikey.com/en/products/detail/stackpole-electronics-inc/RNCP1206FTD20K0/2240380
Resistor	\$0.10	1		https://www.digikey.com/en/products/detail/yageo/AC1206FR-07120RL/5897228
Capacitor	\$0.22	1		https://www.digikey.com/en/products/detail/samsung-electro-mechanics/CL31A106MAHNNNE/3886839
Connector	\$0.98	3		https://www.digikey.com/en/products/detail/phoenix-contact/1844210/349195
Connector	\$1.25	3		https://www.digikey.com/en/products/detail/on-shore-technology-inc/OSTTJ0211530/1588359
Connector	\$0.49	2		https://www.digikey.com/en/products/detail/jst-sales-america-inc/S04B-PASK-2-LF-SN/926755
Connector	\$0.10	1		https://www.digikey.com/en/products/detail/adam-tech/PH1-04-UA/9829296
Connector	\$0.22	1		https://www.digikey.com/en/products/detail/harwin-inc/M20-1060400/3728193
Capacitor	\$0.22	1		https://www.digikey.com/en/products/detail/samsung-electro-mechanics/CL31A106MAHNNNE/3886839
Test Point	\$0.42	8		https://www.digikey.com/en/products/detail/keystone-electronics/5012/255334
Wi-Fi Antenna	\$2.36	1		https://www.digikey.com/en/products/detail/pulsarsen-antennas/W3921B0100/7667486
Connector	\$0.10	2		https://www.digikey.com/en/products/detail/amphe-nol-icc-fci/10129378-902002BLF/7915925
Digikey Order 2	\$60.87		-\$33.54	
Connector	\$0.254	3		https://www.newark.com/jst-japan-solderless-terminals/pap-04v-s/conn-housing-rcpt-4pos-2mm/dp/90R9077
Contact	\$0.103	12		https://www.newark.com/jst-japan-solderless-terminals/sphd-002t-p0-5/contact-socket-28-24awg-crimp/dp/90R9142
Contact	\$0.056	12		https://www.newark.com/harwin/m20-1180046/connector-contact-female-22awg/dp/96K4323

Newark Order	\$2.67		-\$36.21	
Wire	\$3.38	1		https://www.digikey.com/product-detail/en/FIT0584/1738-1434-ND/9559253
Wire	\$0.63	4		https://www.digikey.com/product-detail/en/44A0111-22-2-MX/44A0111-22-2-MX-DS-ND/239970
Wire	\$0.63	4		https://www.digikey.com/product-detail/en/44A0111-22-0-MX/44A0111-22-0-MX-DS-ND/2399708
Test Point	\$0.42	4		https://www.digikey.com/product-detail/en/5012/36-5012-ND/255334
Buck Converter	\$6.73	1		https://www.digikey.com/product-detail/en/SK32A-LTP/SK32A-LTPMSCT-ND/2642068
Diode	\$0.52	1		https://www.digikey.com/product-detail/en/SK32A-LTP/SK32A-LTPMSCT-ND/2642068
Digikey Order 3	\$17.35		-\$53.56	
Soil Test Kit	\$15.14	1		https://www.amazon.com/dp/B0000DI845/
Laptop Charger	\$25	1		https://www.amazon.com/dp/B077898C1R/
Battery	\$25.99	1		https://www.amazon.com/dp/B07JQ6NGN3/
Amazon Order 2	\$66.13		-\$119.69	
Connector	\$0.64	4		https://www.digikey.com/en/products/detail/molex/0395011002/1280602
Wire	\$6.25	1		https://www.digikey.com/en/products/detail/dfrobot/FIT0582/9559251
Wire	\$1.63	6		https://www.digikey.com/en/products/detail/lapp/4160600/12147881
Digikey Order 4	\$18.59		-\$138.28	
Capacitor	\$1.64	2		https://www.digikey.com/en/products/detail/kyocera-avx/TPSC337K006R0080/3068585
Capacitor	\$0.18	2		https://www.digikey.com/en/products/detail/samsung-electro-mechanics/CL31B106KQHNFNE/3888765
Capacitor	\$0.11	2		https://www.digikey.com/en/products/detail/w%C3%BCrth-elektronik/885012208087/5453982
Diode	\$0.39	2		https://www.digikey.com/en/products/detail/nexperia-usa-inc/PMEG4005ET-215/1157707
LED	\$0.22	1		https://www.digikey.com/en/products/detail/w%C3%BCrth-elektronik/150120GS75000/4489936
Connector	\$0.19	2		https://www.digikey.com/en/products/detail/harwin-inc/M20-1060200/3728191
Connector	\$1.22	1		https://www.digikey.com/en/products/detail/hirose-electric-co-ltd/U-FL-R-SMT-10/513011
Connector	\$0.49	2		https://www.digikey.com/en/products/detail/jst-sales-america-inc/S04B-PASK-2-LF-SN/926755
Connector	\$0.10	3		https://www.digikey.com/en/products/detail/adam-tech/PH1-04-UA/9829296

Connector	\$0.24	3		https://www.digikey.com/en/products/detail/harwin-inc/M20-1060400/3728193
Connector	\$0.13	1		https://www.digikey.com/en/products/detail/metz-connect-usa-inc/PR20203VBDN/12342894
Connector	\$0.98	2		https://www.digikey.com/en/products/detail/phoenix-contact/1844210/349195
MOSFET	\$1.57	3		https://www.digikey.com/en/products/detail/microchip-technology/VN3205N3-G/4902407
Resistor	\$0.10	1		https://www.digikey.com/en/products/detail/stackpole-electronics-inc/RMCF1206JT51R0/1753854
Resistor	\$0.10	2		https://www.digikey.com/en/products/detail/stackpole-electronics-inc/RMCF1206JT20K0/1757361
Resistor	\$0.10	4		https://www.digikey.com/en/products/detail/yageo/RC1206JR-074K7L/729295
Resistor	\$0.10	2		https://www.digikey.com/en/products/detail/te-connectivity-passive-product/CRGCQ1206F120R/8576401
Resistor	\$0.10	3		https://www.digikey.com/en/products/detail/stackpole-electronics-inc/RNCP1206FTD10K0/2240370
Resistor	\$0.10	2		https://www.digikey.com/en/products/detail/yageo/RC1206FR-0739KL/728855
Transceiver	\$5.25	1		https://www.digikey.com/en/products/detail/maxim-integrated/MAX485CPA/948026
Buck Converter	\$6.73	1		https://www.digikey.com/en/products/detail/texas-instruments/LM2596SX-3-3-NOPB/366832
Connector	\$4.07	2		https://www.digikey.com/en/products/detail/samtec-inc/ESQ-110-14-T-D/6678046
Connector	\$1.57	2		https://www.digikey.com/en/products/detail/amphe-nol-icc-fci/65039-036LF/1002657
Contact	\$0.05	24		https://www.digikey.com/en/products/detail/cvilux-usa/CI33T021PE0/13175839
Connector	\$0.24	2		https://www.digikey.com/en/products/detail/hirose-electric-co-ltd/DF3A-4P-2DS/560461
Connector	\$0.14	2		https://www.digikey.com/en/products/detail/hirose-electric-co-ltd/DF3-4S-2C-10/4282726
Contact	\$0.11	10		https://www.digikey.com/en/products/detail/hirose-electric-co-ltd/DF3-22SC/371509
Socket	\$0.19	1		https://www.digikey.com/en/products/detail/assmann-ww-components/A-08-LC-TT/821740
Digikey Order 5	\$73.17		-\$175.55	
Inductor	\$2.51	1		https://www.mouser.com/ProductDetail/Coilcraft/D05022P-153MLD?qs=zCSbvcPd3pYrsuHvv2R2WQ%3D%3D
Mouser Order 2	\$2.51		-\$178.06	
Connector	\$0.95	2		https://www.pololu.com/product/4762
Pololu Order	\$1.90		-\$179.96	
PCBs	\$0.40	5		https://jlcpcb.com/
JLC Order	\$2		-\$181.96	

Flight Controllers	\$9.99	2		https://www.buddyrc.com/products/f3-acro-flight-controller?variant=30735600517206
BuddyRC Order 2	\$19.98		-\$201.94	
PCBs	\$33	1		https://www.4pcb.com
Advanced Circuits Order	\$33		-\$234.94	
PCBs	\$5	1		https://www.3welec.com
Component Population	\$0.05	40		
3W Order 1	\$7		-\$241.94	
PCBs	\$5	1		https://www.3welec.com
Component Population	\$0.05	60		
3W Order 2	\$8		-\$249.94	
Assorted Hardware	\$15	1		https://www.lowes.com
Lowes Order	\$15		-\$264.94	

Appendix 2: Drone Parts Unit and Bulk Cost

Item	Quantity	Unit Price	Bulk Price	Link
Assorted Hardware	1	\$15	\$15	Lowes
Battery	1	\$25.99	\$25.99	https://www.amazon.com/dp/B07JQ6NGN3/
Battery Straps	2	\$0.99	\$0.99	https://www.buddyrc.com/collections/battery-straps/products/20cm-coated-glacier-battery-strap-1
Buck Converter	1	\$6.73	\$4.36	https://www.digikey.com/en/products/detail/texas-instruments/LM2596SX-3-3-NOPB/366832
Bullet Connectors	4	\$1.49	\$1.49	https://www.buddyrc.com/collections/connector/products/3-5mm-gold-plated-bullet-connector-3-pairs?variant=30735582888022
Capacitor	2	\$1.64	\$0.84	https://www.digikey.com/en/products/detail/kyocera-avx/TPSC337K006R0080/3068585
Capacitor	2	\$0.18	\$0.03	https://www.digikey.com/en/products/detail/samsung-electro-mechanics/CL31B106KQHNFE/3888765
Capacitor	2	\$0.11	\$0.06	https://www.digikey.com/en/products/detail/w%C3%BCrth-elektronik/885012208087/5453982
Charging Bag	1	\$4.98	\$4.98	https://www.buddyrc.com/collections/lipo-bag/products/fireproof-explosionproof-lipo-battery-safe-bag-lipo-battery-guard-185x75x60mm?variant=30735650062422
Connector	2	\$0.19	\$0.07	https://www.digikey.com/en/products/detail/harwin-inc/M20-1060200/3728191
Connector	1	\$1.22	\$0.23	https://www.digikey.com/en/products/detail/hiroseelectric-co-ltd/U-FL-R-SMT-10/513011
Connector	1	\$0.1	\$0.03	https://www.digikey.com/en/products/detail/adam-tech/PH1-04-UA/9829296
Connector	2	\$0.24	\$0.1	https://www.digikey.com/en/products/detail/harwin-inc/M20-1060400/3728193
Connector	1	\$0.13	\$0.05	https://www.digikey.com/en/products/detail/metz-connect-usa-inc/PR20203VBDN/12342894
Connector	4	\$0.98	\$0.63	https://www.digikey.com/en/products/detail/phoenix-contact/1844210/349195
Connector	2	\$4.07	\$2.36	https://www.digikey.com/en/products/detail/samtec-inc/ESQ-110-14-T-D/6678046
Connector	1	\$0.24	\$0.09	https://www.digikey.com/en/products/detail/hiroseelectric-co-ltd/DF3A-4P-2DS/560461
Connector	1	\$0.14	\$0.05	https://www.digikey.com/en/products/detail/hiroseelectric-co-ltd/DF3-4S-2C-10/4282726
Connector	1	\$0.95	\$0.80	https://www.pololu.com/product/4762
Contact	4	\$0.11	\$0.05	https://www.digikey.com/en/products/detail/hiroseelectric-co-ltd/DF3-22SC/371509
Diode	1	\$0.52	\$0.10	https://www.digikey.com/product-detail/en/SK32A-LTP/SK32A-LTPMSCT-ND/2642068
Diode	2	\$0.39	\$0.10	https://www.digikey.com/en/products/detail/nexperia-usa-inc/PMEG4005ET-215/1157707

ESC	1	\$32.95	\$32.95	https://www.buddycrc.com/collections/quad-esc/products/hakrc-35a-blheli_32-2-5s-4in1-brushless-esc
Flight Controller	1	\$9.99	\$9.99	https://www.buddycrc.com/collections/quad-fc/products/f3-acro-flight-controller
GPS Antenna	1	\$2.77	\$1.57	https://www.digikey.com/en/products/detail/molex/2065600100/9094612
GPS Antenna Connector	1	\$1.22	\$0.57	https://www.digikey.com/en/products/detail/hirose-electric-co-ltd/U-FL-R-SMT-1-40/4285474
GPS Receiver	1	\$37.28	\$30.23	https://www.digikey.com/en/products/detail/kaga-electronics-usa/GN-8720/7491503
Inductor	1	\$2.51	\$1.28	https://www.mouser.com/ProductDetail/Coilcraft/DO5022P-153MLD?qs=zCSbvcPd3pYrsuHvv2R2WQ%3D%3D
IR Distance Sensor	1	\$12.90	\$12.90	https://www.digikey.com/en/products/detail/dfrobot/SEN0245/8827827
Laptop Charger	1	\$25	\$25	https://www.amazon.com/dp/B077898C1R/
LED	1	\$0.22	\$0.15	https://www.digikey.com/en/products/detail/w%C3%BCrth-elektronik/150120GS75000/4489936
Li-Po Charger	1	\$35	\$35	https://www.buddycrc.com/collections/all-battery-chargers/products/isdt-q6-nano-battgo-200w-8a-colorful-pocket-battery-balance-charger-for-1-6s-lipo-battery?variant=32636734079062
Microcontroller	1	\$15.59	\$6.30	https://www.digikey.com/en/products/detail/texas-instruments/EK-TM4C123GXL/3996736
MOSFET	3	\$1.57	\$1.19	https://www.digikey.com/en/products/detail/microchip-technology/VN3205N3-G/4902407
Motor	4	\$22.80	\$22.80	https://www.buddycrc.com/collections/sunnysky-x-v3-motors/products/sunnysky-x2216-v3-brushless-motors-long-shaft-version?variant=33002331144278
PCB Fabrication	1	33	\$33	Advanced
PCB Population Base Cost	1	\$5	\$5	3W
PDB	1	\$3.95	\$3.95	https://www.buddycrc.com/products/kensun-power-distribution-board-with-5v-12v-output?variant=30735600418902
Population	60	\$0.05	\$0.05	3W
Propellor	2	\$6.88	\$6.88	https://www.buddycrc.com/collections/10-inch-prop/products/carbon-fiber-slow-flyer-10x4-7-inch-props-normal-and-reverse?variant=31974559187030
Resistor	1	\$0.10	\$0.01	https://www.digikey.com/en/products/detail/stackpole-electronics-inc/RMCF1206JT51R0/1753854
Resistor	2	\$0.10	\$0.01	https://www.digikey.com/en/products/detail/stackpole-electronics-inc/RMCF1206JT20K0/1757361
Resistor	4	\$0.10	\$0.01	https://www.digikey.com/en/products/detail/yageo/RC1206JR-074K7L/729295
Resistor	2	\$0.10	\$0.01	https://www.digikey.com/en/products/detail/te-connectivity-passive-product/CRGCQ1206F120R/8576401

Resistor	3	\$0.10	\$0.01	https://www.digikey.com/en/products/detail/stackpole-electronics-inc/RNCP1206FTD10K0/2240370
Resistor	2	\$0.10	\$0.01	https://www.digikey.com/en/products/detail/yageo/RC1206FR-0739KL/728855
Socket	1	\$0.19	\$0.07	https://www.digikey.com/en/products/detail/assmann-wsw-components/A-08-LC-TT/821740
Soil Sensor	1	\$83.66	\$83.66	https://www.amazon.com/dp/B0836WYNJ1/
Transceiver	1	\$5.25	\$2.97	https://www.digikey.com/en/products/detail/maxim-integrated/MAX485CPA/948026
Wi-Fi Module	1	\$37.56	\$28.9	https://www.mouser.com/ProductDetail/Silicon-Labs/WF121-E-V2C?qs=8HqIXBaaO6yQS%2FkPjITxCw%3D%3D
Wire	2	\$6.25	\$6.25	https://www.digikey.com/en/products/detail/dfrobot/FIT0582/9559251
Wire	4	\$0.70	\$0.70	https://www.digikey.com/product-detail/en/44A0111-22-0-MX/44A0111-22-0-MX-DS-ND/2399708
Wire	6	\$1.63	\$1.05	https://www.digikey.com/en/products/detail/lapp/4160600/12147881
XT60 Leads	1	\$6.49	\$6.49	https://www.amazon.com/dp/B07QH249CR/
XT60 Splitter	1	\$8.99	\$8.99	https://www.amazon.com/dp/B07DKXQLKT

Appendix 3: PCB Test Plan

Component	Continuity, and then Power with no signals	Signals/Inputs Injected to Test	Notes/Results
		Expected Outcome	
U1 – GPS	U1.23 – 3.3V U1.24 – GND	Condition: move module around NI Lab	Passed
		Out: read changing output from MCU	
U2 – Wi-Fi	U2.9 – 3.3V U2.1 – GND	Condition: move module around NI Lab	Passed
		Out: sustained connection	
U3 – RS485	U3.8 – 3.3V U3.5 – GND	Condition: put sensor in soil and request to read N parameter	5V powered but 3.3V supplied
		Out: read N conditions of soil	
U6 – Buck Converter	U6.1 – 5V U6.3 – GND	Condition: 5V applied	Regulated and unregulated 3.3V were switched
		Out – U6.2 3.3V	
Q1 – MOSFET	Q1.2 – 11.1V	Condition: send GPIO signal from MCU	Circuit moved from source to drain since NMOS
		Out: Q1.3 11.1V	
Q2 – MOSFET	Q2.2 – 11.1V	Condition: send GPIO signal from MCU	Circuit moved from source to drain since NMOS
		Out – Q2.3 11.1V	
J3 – Distance Sensor	J3.1 – GND	Condition: send I2C signal from MCU	Power confirmed, data not tested due to time
		Out: read distance data	
J7 – Flight Controller Serial	J7.2 – 3.3V J7.1 – GND	Condition: send UART sequence	Passed, switched to 5V line
		Out: J7.3 read telemetry data	
J13 – Flight Controller General	J13.2 – 5V J13.1 – GND	Condition: send UART sequence	Unused

Appendix 4: Drone Certificate of Registration

Small UAS Certificate of Registration

Registered Owner: University of Virginia

UAS Manufacturer: UVA SEAS ECE

UAS Model: Agroflight Capstone

Serial Number: N/A

Registration Number: FA37TMKAWW

Issued: 12/13/2021

Expires: 12/13/2024



This Small UAS Certificate of Registration is not an authorization to conduct flight operations with an unmanned aircraft. Operations must be conducted in accordance with applicable FAA requirements. The operator of the aircraft is responsible for knowing and understanding what those requirements are. For more information on flying requirements, please visit the FAA website at www.faa.gov/uas.

For U.S. citizens, permanent residents, and certain non-citizen U.S. corporations, this document constitutes a Certificate of Registration. For all others, this document represents a recognition of ownership.

Operators of unmanned aircraft must ensure they comply with the appropriate safety authority from the FAA and economic authority from the DOT.