UNIVERSITY OF WATERLOO

Department of Mechanical and Mechatronics Engineering

Project KITE.

A Report Prepared For:
The University of Waterloo
MTE 482 – Capstone Project

Prepared By:
MTE482 Group 42 – Team Kite.
Lucas Arzoumanian
Joshua Cai
Neil Fernandes
Samuel Sun
Xiren Wang

April 8, 2024

200 University Ave W,
Waterloo, Ontario N2L 3G1
April 8, 2024

Prof Michael Mayer
Professor, MME
University of Waterloo
200 University Ave W.
Waterloo, Ontario, N2L 3G1

Prof Stephen Smith
Professor, ECE, Waterloo AI Institute Director
University of Waterloo
200 University Ave W.
Waterloo, Ontario, N2L 3G1

Dear Professor Mayer and Smith,

This report, titled "Project KITE." (formerly CaelVision), was written to fulfil the Winter 2024 MTE482 capstone design project requirements. The intent for this report is to continue documenting all the progress made on the design project from the end of the winter MTE481 report to present.

Over the past eight months, Team KITE. had been dedicated to the development of an autonomous VTOL system tailored for inspecting high-voltage transmission line insulators. Through rigorous effort, we had achieved the creation of a fully functional VTOL unit comprising both physical flight hardware and sophisticated onboard Computer Vision systems. These systems play a crucial role in guiding, inspecting, and ensuring safe landing onto designated docks. The project had served as a remarkable opportunity for our team of five undergraduate students to apply the technical skills acquired through classroom instruction and co-op experiences. Moreover, it had fostered effective teamwork, enabling us to seamlessly integrate feedback from instructors and stakeholders alike, thus refining our product iteratively.

This report was written entirely by us and has not received any previous academic credit at this or any other institution. We would like to thank professors Mahesh Pandey (CIVE) and Shesha Jayaram (ECE) for their invaluable advice and expertise. We would also like to thank the Norman Esch Foundation for giving us the opportunity to compete at the Esch Entrepreneurial competition for capstone groups and for providing us with an additional $5000 funding to our capstone project.

And finally, we would like to thank you for supporting us through the entire capstone project process, and for guiding us through our final school term at the University of Waterloo.

Warm regards,

Team 42 Team KITE.
Mechatronics Engineering BASc '24
University of Waterloo

Lucas Arzoumanian [redacted]

Josh Cai [redacted]

Samuel Sun [redacted]

Neil Fernandes [redacted]

Xiren Wang [redacted]

Executive Summary

The mechanical failure of electrical insulators in High Voltage Transmission Lines (HVTL) is the single highest individual source of power grid failure [1]. The inspection of these insulators is critical to having a robust electrical infrastructure; knowing the state of each one, and whether or not they require replacement, is key. Current solutions for inspection involve technicians leaning out of helicopters and manually checking each one. This is not only very costly, but dangerous.

Project KITE.'s proposed solution is to completely automate this process using a VTOL glider aircraft to inspect the conditions of the insulators using cameras, as well as a series of docks to recharge and store the aircraft. Designed to be completely autonomous, the VTOL glid in a sawtooth pattern along the power line using its ability to hover to closely inspect each insulator as needed. Using deep learning and trained computer vision models, the system will automatically detect insulators and alert operators once repair is required.

The design of this system was comprised of both off-the-shelf components and parts designed in-house, with reliability being the guiding principle of every design decision.

The final budget of the system ended up being $1,726.86, a cost that Project KITE. was able to recuperate thanks to the $5000 prize won at the Norman Esch competition.

The problem was identified in the F23 offering of MTE481, and a high-level architecture design of both hardware and software systems was designed. In W24, the prototype was build and completed, as well as all required accompanying software.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

High-voltage transmission line insulators, shown in Figure 1, are a vital component of the power grid. However, they are also the most prone to failure. Prolonged exposure to weather and outdoor conditions (including birds) can lead to physical damage like chipping, which can further result in failures such as flashover damage and self-explosion. Just one failure of an insulator can cause cascading failures in neighbouring towers and eventually take down that part of the grid. In fact, 35% of system failures are attributed to insulators [2].

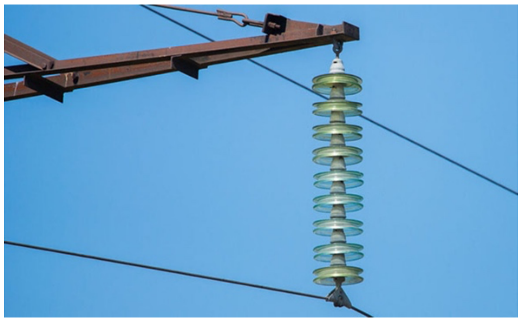


Figure 1: High-voltage insulator [3]

As a result, the inspection of these insulators plays a crucial part in ensuring the safe operation of transmission lines. Current inspection methods rely on sending workers out to the field for manual inspection of hardware conditions. This usually involves driving to sites or flying in helicopters for aerial inspection. Recently, there has been a significant increase in the use of unmanned aerial vehicles (UAVs) as tools for this purpose. This mainly involves tele-operated drones equipped with camera systems and deep-learning algorithms, allowing the system to perform visual inspections of towers.

1.2 Needs Assessment

There are issues with current insulator inspection methods. Traditional manual methods like helicopters can be effective but are labour-intensive and time-consuming, resulting in low efficiency and high risk to personnel. This also ends up being very expensive for the larger transmission lines, where most segments traverse through rural and unpopulated areas. In 2015, Hydro One alone spent US$16,000,000 on inspection and preventative maintenance operations [4]. This is just one company in Ontario.

For inspections involving tele-operated drones, the biggest concern is distance – quadcopters have very limited operational ranges that stem from battery limitations, power consumption, and effective remote-control range [5]. Based on all this, there is a need for a system that can inspect high-voltage insulators while being low-risk, cost-effective, power-efficient, and reliable.

1.3 Problem Formulation

This project aims to develop a system that addresses the issues with current methods of insulator inspection.

First, the system should be able to reliably identify the status of every insulator on multiple transmission towers. This will require some means of inspection that can detect the many types of defects like cracks, chips, and missing insulators.

Next, the system should send alerts to repair crews if an insulator is found to be damaged or missing. These alerts should include relevant information (e.g., location, type, picture), which allows crews to more easily identify and repair the broken insulator.

The system should also be easily deployable and usable. Given the remote and rugged environments that transmission lines traverse through, the hardware for this system should be easily moveable by humans. Any software should be intuitive and easy to set up for extended operation.

Finally, the system should be deployable for long periods of time. This will require high power efficiency and potentially additional resiliency to the environment.

1.4 Constraints

The following constraints were determined following extensive consultation with course advisors, faculty, and industry professionals. These constraints determine the main characteristics of the final solution and have not changed since the MTE481 final report [5].

<u>Have the capability to operate autonomously.</u>
The system must be able to traverse the transmission line autonomously without external human control. Normal operation should be constrained to automatic navigation, although operators may remotely access the system for data acquisition.

<u>Reliably detect insulator failure.</u>
The system must be able to detect damages and defects on insulators while alerting operators that a failure was found. False positives and especially false negatives should be minimized.

<u>Seamlessly move between towers.</u>
The system should only rely on its onboard sensors to sense the environment and automatically respond to external forces like gusts and rain. Potential sensors include GPS, IMUs, and onboard cameras.

<u>Always be safe to use.</u>
The system must have safeguards meant to protect humans and objects in the vicinity. This may include e-stops, software fail-safes, and accompanying safety procedures/checks that describe safe operation.

<u>Safe storage when not in use.</u>
The system must have the capability to remain outdoors for extended periods without constant maintenance. Operating costs can be significantly reduced if the system can remain in the wild unattended on account of the terrain that transmission lines traverse.

1.5 Criteria

System criteria are used to judge how effective a solution is at solving the outlined problem. All potential solutions are judged then compared to determine the best solution. Like the constraints, these have not changed since the MTE481 final report [5].

Speed:
Speed refers to how much time it takes to inspect a transmission line, which is influenced by several factors:

The most obvious is the movement speed of the system, which refers to how quickly the system can travel a given distance. Transmission lines are often hundreds of kilometers longs, so a faster movement speed will result in more frequent inspection readings and speedier maintenance responses.

Information processing time, referring to how quickly the system can identify the status of an insulator, will determine how quickly operators are updated on the conditions of the line. A system that can pre-process information and send only the most important insights to the operator will enable faster response times.

Finally, the reset time of the system is also a factor. This can refer to the recharge time of the system.

Accuracy:
Accuracy refers to the confidence level of system readings, which is the relation between false positives and false negatives. Both metrics should be as low as possible.

Efficacy:
Efficacy refers to how useful the system is to the end user. A useful system is one that can send clear and actionable data to operators. This may include the existence of defects, the location found, and visual media for further review.

Manufacturability:
Manufacturability refers to the ease with which the system can be built. Since the system will be constructed entirely by the team for the purposes of this project, the design does not need to be thoroughly tested for design for manufacturing (DFM) but should still be well thought out. This is for the purposes of quality of life and ease of modification/repair.

Deployment Time:
This refers to the time required for the system to be set up for use. This can be greatly affected by physical characteristics like size, weight, and ergonomics. Fewer deployable components is also preferable. Shorter deployment times and less labour required will be more economical for the operator.

Reliability:
Reliability refers to how long the system can remain operational in the field between maintenance. The system should be ruggedized for long-term operation, accounting for factors like water ingress and temperature.

1.6 Design Review

This section will evaluate the selected design as of the end of MTE481 against the above requirements, constraints, and criteria.

The overall architecture of a VTOL drone combined with deployable docks is sufficient to satisfy all requirements. A drone with the maneuverability of a quadcopter can easily access and inspect all insulators on a given transmission tower. Onboard cameras and computer vision software give it the capability to perform visual inspections and identify common types of defects. A stationary base serves as a flexible platform that can allow the drone to recharge and be protected from inclement weather.

The design follows constraints while performing well in most of the criteria. The choice of using the open-source autopilot software ArduPilot gives the drone the capability of fully autonomous flight. Onboard attitude and heading reference systems (AHRS) that make use of IMUs, magnetometers, and a GPS allow it to easily follow waypoints and navigate from tower to tower. The computer vision software tests that had been done during MTE481 showed promising results for the reliability of the defect detection systems. Comprehensive safety procedures and countermeasures were created to ensure safe operation of the drone and those around it.

The greatest challenge with the design presented at the end of MTE481 was its manufacturability. The shape and structure of the proposed VTOL frame was contoured and complex, with multiple complex flight control surfaces, and suboptimal placement of the tilt motors at the wingtips. These factors combined result in a frame that was difficult and time-consuming to construct, which was an issue for a prototype that was expected to crash frequently during testing.

2 Final Design

The design philosophy for this project revolved around two main principles: ease of manufacturing, and reliability of the design. This would enable rapid design iterations between test sessions, and free up time to focus on new feature development.

2.1 Final Design Details

The system contains three main development areas: A VTOL drone, dock, and software.

2.1.1 VTOL Drone

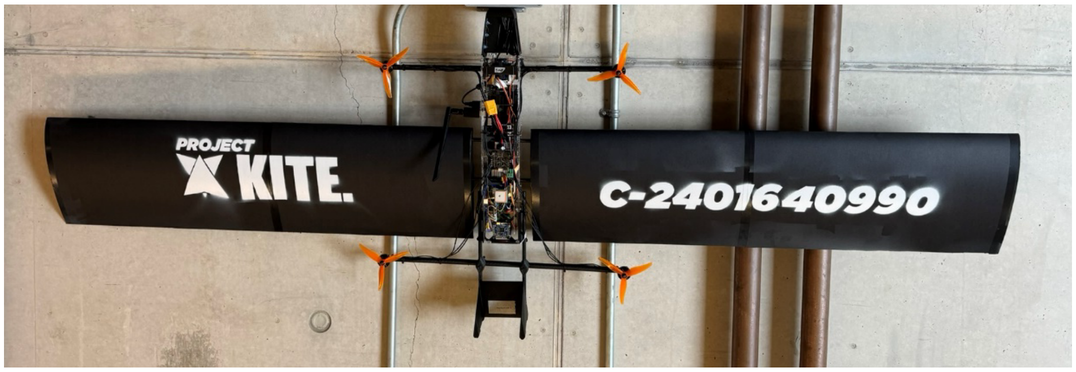


Figure 2: Final VTOL drone.

The VTOL drone is meant to serve as a mobile aerial platform for the computer vision hardware to operate from during inspection flights. The drone is meant to fly along sections of transmission line, periodically taking optical measurements of the insulator conditions.

A transmission line is composed of towers, which contain the insulator stacks of interest, as well as long stretches of cables that are not of interest. The flight path of the drone will hence consist of repeated cycles of stationary hovering to inspect and process data from transmission towers, as well as stretches of forward translation. A VTOL drone design combines the best of both worlds of rotor and fixed wing aircraft design, combining the agility of a quadcopter frame, as well as the range efficiency of a fixed wing.

The mechanical design of the VTOL drone was organized into the major subassemblies:

Table 1: Mechanical systems and associated components.

System	***Components***
Chassis	Main body frame Outboard mounting points Electronics mounting
Airfoils	Wing assembly Tail assembly Control surface electronics
Propulsion	Rotor assemblies Tilt shaft assemblies

CAD was organized in OnShape. The cloud-based nature of OnShape allowed for multiple collaborators to work on designs concurrently, as well as automated cloud-backup and revision management. This allowed for quick revision updates based on feedback from test flights.

2.1.1.1 Design Highlights

Chassis: The chassis was made from sheet material that is easily laser cuttable. This was done under the understanding that the drone would crash and be damaged often, so an easily and quickly repairable design would be preferable. The panels integrated interlocking tabs for easy alignment and positive engagement, as well as screw holes with which the panels could be screwed together.

The chassis also integrated mounting brackets and panels for the associated electronics, as well as adjustable camera mounts that could accurately position camera points of view.

Rotors: The two front rotors were individually tiltable 90deg each, from an “upright” to “forwards” position. This allowed for the VTOL to transition from vertical to horizontal flight, as well as pitch and roll control.

This was achieved by mounting each motor on a bearing-mounted carbon fibre shaft, each of which was actuated with a small servo motor through a linkage assembly. This was all mounted in a monolithic bearing block for stiffness purposes.

Wings: The wings provide most of the lift for the drone during flight operations and differentiates the drone from current market inspection drone offerings.

In the first design iteration, the airfoil of the wings was chosen to maximise the lift-to-drag ratio of the drone. However, due to the nature of the sawtooth-shaped drone flight trajectory, and thrust capacity of the motors, it was quickly realised that drag is not a significant factor. In fact, to keep the VTOL hovering under standard condition, only 20% of motor throttle is required. This means that while the drone is in cruise mode, if more airspeed is desired, the motors can simply by turned up to compensate for drag. Maximising lift was therefore the only consideration that needed to be made when it came to picking an airfoil. To this end, the NACA6412 was chosen due to its capacity to achieve high coefficients of lift. Figure 3, shown below, is a plot of what coefficient of lift is achieved based on the angle of attack of the wing.

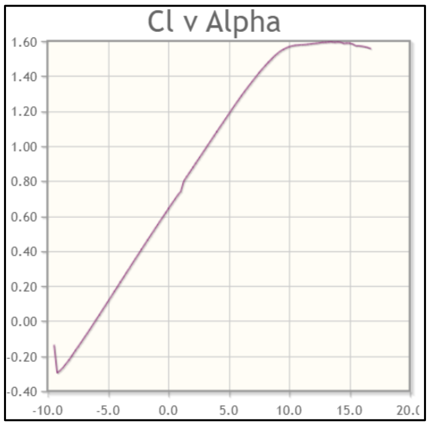


Figure 3: Plot of coefficient of lift vs angle of attack.

A profile of the airfoil can be seen in **Error! Reference source not found.**:

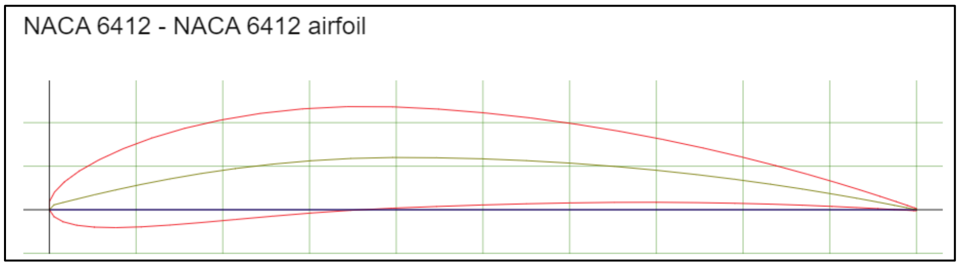


Figure 4: NACA6412 airfoil profile.

Since high lift is desired, the angle of incidence will be set to 8 degrees. Note the steep drop off in the coefficient of lift at around 10 degrees; this is the point at which the aircraft "stalls", when lift is no longer created. Since there is no situation where the drone is required to pitch upward, having the angle of incidence near the stall angle is justified. Also, in the event that stall is in fact reached, the propellors are more than capable of compensating.

Using the fact that the final mass of the electronics and housing came in at approximately 2 kilograms, and assuming a velocity of 5m/s, Equation (1) calculates the lift equation used to determine the final dimensions of the wing. Note that the entire gravity force should not need to be canceled by the wings due to the oscillatory flight path.

$$F_g = mg = 2 * 9.81 = 19.62N$$
$$F_l = C_l * \frac{\rho v^2}{2} * A_l = 1.5 * \frac{1.225 * 5^2}{2} * A \tag{1}$$
$$if: A = 0.35 * 2$$
$$F_l = 16.08N$$

The final airfoil design has a relatively large wing area of 2x0.35m. In cruise mode, the drone creates enough lift to have the drone glide long distances.

2.1.1.2 Electrical

As originally planned, as much off-the-shelf hardware was used as possible. The electrical system is based off of a known-good configuration of a hobbyist quadcopter FPV drone. Knowing that the configuration is proven to work allowed for focus to be diverted to developing the rest of the VTOL drone. The system can be split into three component types: Power, Controllers, and Sensors. A schematic of the entire electrical system can be seen in

Appendix C: SanBan Schematic.

Power: A 6S LiPo battery is used to remotely power all of electronics on the VTOL drone during flight. It has a nominal voltage of 22.2V, at which the main rotor motors are run at. This voltage was chosen based on the efficiency curves for the main rotor motors. At maximum throttle, the motors are expected to draw up to 3.8kW. The rest of the control electronics run at 5.0V, which is supplied by a separate stepped-down rail. The 5.0V rail is generated by a custom external circuitry, the SanBan.

Controllers: The controllers are responsible for collecting sensor data, and forming actionable reactions based on them. A list of the controllers onboard the VTOL are:

Table 2: Controller suite on VTOL drone.

Controller	Model Number	Purpose
Electronic Speed Controller (ESC)	Skystars KO45 ESC	Throttle/speed control of main rotors
Flight Controller (FC)	F4 V3S Plus	VTOL localization and flight control surface command calculations
FC Co-processor	Arduino Nano	PWM device signal generation and control
CV Processor	Nvidia Jetson Nano	Inspection computer vision computation

Sensors: The sensors are responsible for observing environmental conditions and relaying their readings to the controllers. The majority of the sensors are for localization purposes and are used to determine the VTOL's position and heading in space. The full list of sensors include:

Table 3: Sensor suite on VTOL drone.

Sensor	Model Number	Purpose
GPS	NEO-M9N GPS	Coarse localization
IMU	TDK InvenSense ICM-20948	Acceleration EKF
ToF	TF-Mini-S	Altitude sensing
Magnetometer	AltIMU-10 v6	Compass heading
Camera 1	NexiGo N60	Optical sensing of insulators
Camera 2	Rapberry Pi Camera Module 2	Precision landing

Power: A 6S LiPo battery is used to remotely power all of electronics on the VTOL drone during flight. It has a nominal voltage of 22.2V, at which the main rotor motors are run at. This voltage was chosen based on the efficiency curves for the main rotor motors. At maximum throttle, the motors are expected to draw up to 3.8kW. The rest of the control electronics run at 5.0V, which is supplied by a separate stepped-down rail. The 5.0V rail is generated by a custom external circuitry, the SanBan.

SanBan:
One potential improvement identified in the MTE481 report was to replace the soldered connections on the flight controller stackup with disconnectable connectors [5].

The solution to this was the creation of a custom PCB, named the SanBan, whose role is to break out the connections of the existing off-the-shelf flight controller to disconnectable JST connectors. This was used to connect the various localization sensors, such as GPS, ToF, compass, and Bluetooth. With the SanBan, the sensors are able to be plugged and unplugged, improving drone serviceability.

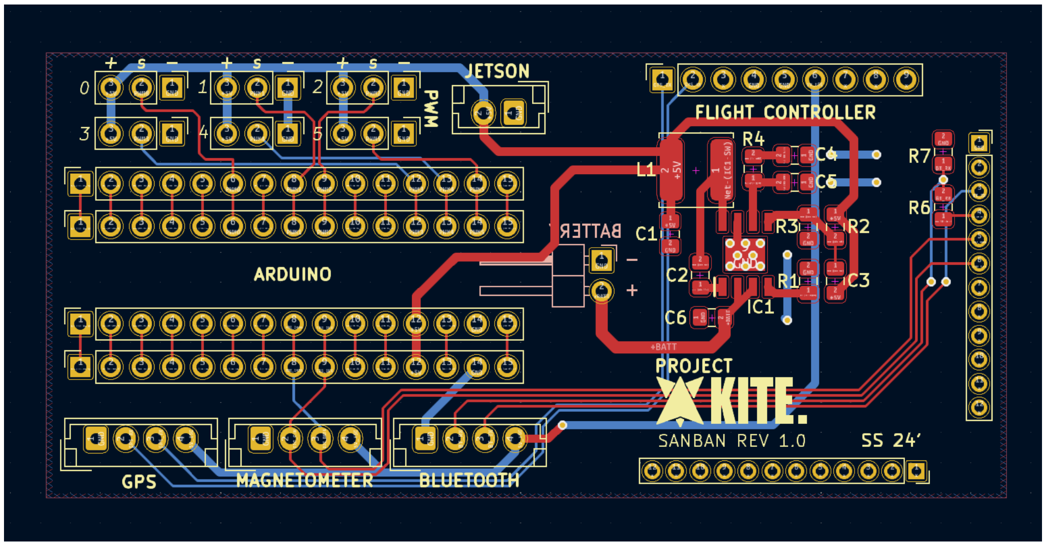


Figure 5: SanBan breakout PCB for FC stack, in KiCad.

Additionally, the SanBan provides ports from which PWM-driven devices can be controlled. This includes servo motor control for the control surfaces, as well as LED control. The flight controller has a limited number of timer channels in its MCU, so an Arduino Nano was added to the SanBan to act as a co-processor. By communicating through the flight controller's I^{2}C bus, the system is able to send commands to the Arduino, which in turn sends appropriate PWM signals to the external devices.

The entire flight stackup and external devices are powered by a 5.0V DC rail, which is generated by a standalone circuit on the SanBan. By connecting the board to the 22.2V DC LiPo battery, a switching power supply circuit steps down the voltage to 5.0V.

Dock

The purpose of the dock is to maintain the autonomy of the system. Every time the drone runs out of battery, it would land on one of the docks placed along the power line to recharge.

For the scope of this capstone, the dock was designed to open and close based on its proximity to, and whether it was within the field of view of, the drone. Once the drone sees the ArUco marker at a certain altitude, the dock opens up. After waiting for the time it would take for the drone to charge, the dock opens up to release it. All communications between the drone and dock are made through Bluetooth.

2.1.1.3 Mechanical

Figure shows the dock in its intended open state and closed state:

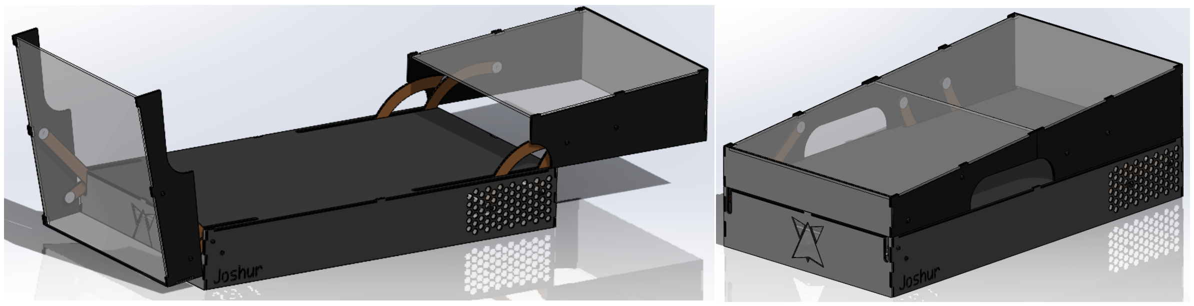


Figure 6: CAD render of VTOL dock – open and closed position

The top of the dock is made of acrylic to have the ArUco marker be visible from the perspective of the drone. The mechanism used for folding the dock open and close involves two separate four-bar linkages. When closed, the two halves come together to form a protective cover around the drone. There is a slot in the side walls for the wings. Under the dock, two separate gear trains are used to move the hatch. Since the back half of the hatch weighs a lot more than the front, the power train was designed to distribute the torque evenly across all links. The front hatch is lighter, and only involves 1 gear pair.

Torque is applied to the axle via the MG996R Servo. All the gears were designed to be helical to reduce backlash. Although the servos are theoretically capable of providing enough torque to move the hatch, a ratio of 3:1 was applied to be conservative. Figure 7 below shows the underside of the dock, including the Arduino mounting and a holder for the Wago connectors that were used as the power bus.

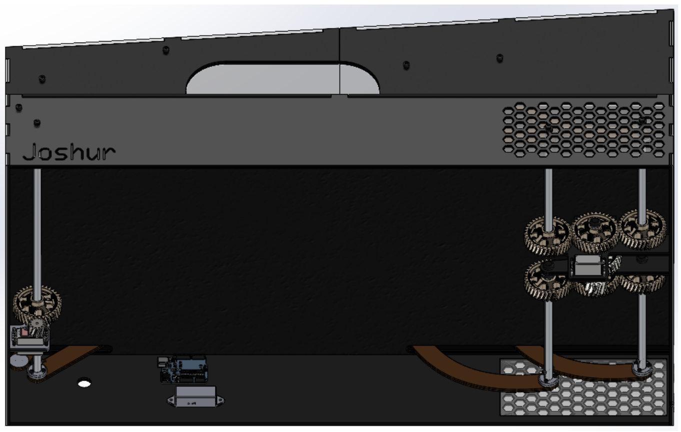


Figure 7: Underside of VTOL dock, with the two gear trains visible on either side, and Arduino in bottom middle.

2.1.1.4 Electrical

The electrical schematic of the dock is shown in Figure 8.

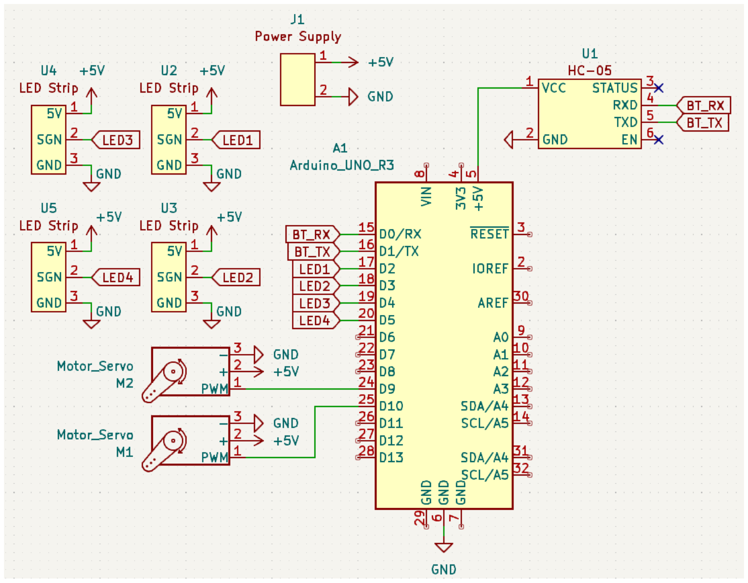


Figure 8: Electrical schematic of the dock hardware electronics.

An Arduino was used as the microcontroller. Since the maximum current draw of the servos was 2.5A, and the 60 LED, LED strip needs around 20mA per LED, a power supply was used to power everything, except for the Bluetooth module that was powered by the Arduino itself. This is a simulation of how the dock would be powered in the field, where each dock would be powered externally.

2.1.1.5 Software

The final configuration uses the same known-good electrical configuration, with the addition of additional sensor and processing hardware. The dock software consists of a simple Arduino program that receives the Bluetooth message from the drone when its close enough. It also controls the LED configuration, letting the user know when the dock is open or closed, as well the current battery level of the drone as it is recharging.

2.1.2 Software

The software for this project is roughly split into two categories: embedded software found in the flight controller and Arduino, and software in the Jetson. The embedded software is responsible for the fundamental operation of the drone while the Jetson is responsible for computer vision and mission control.

2.1.2.1 Embedded Controller Software

The flight controller runs a custom version of ArduPilot, an open-source autopilot software [6]. ArduPilot can be used for a multitude of vehicles; VTOLs fall under the ArduPlane codebase. Due to the sheer complexity of an autopilot, the team decided that a reliable and popular open-source platform was the way to go. A completely custom autopilot is not feasible in the timeframe given for this project. ArduPilot has all the basic functionality needed to get a drone flying. It facilitates communication with a ground station or companion computer, runs sensor fusion algorithms for attitude estimation, responds to movement commands, and runs control loops to calculate and output motor/actuator commands. A high-level diagram of the software architecture can be seen in Figure 9.

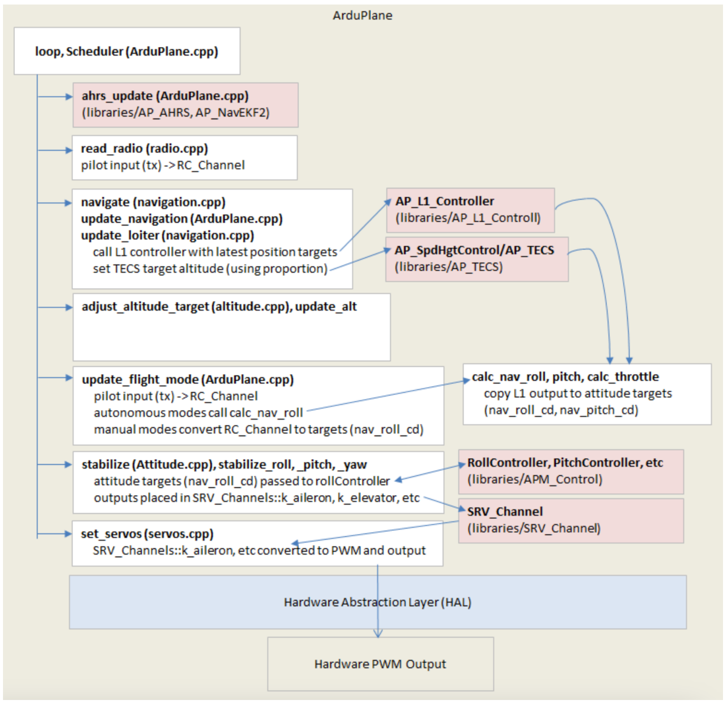

ArduPlane
loop, Scheduler (ArduPlane.cpp)
ahrs_update (ArduPlane.cpp)
(libraries/AP_AHRS, AP_NavEKF2)
read_radio (radio.cpp)
pilot input (tx) -> RC_Channel
navigate (navigation.cpp)
update_navigation (ArduPlane.cpp)
update_loiter (navigation.cpp)
call L1 controller with latest position targets
set TECS target altitude (using proportion)
AP_L1_Controller
(libraries/AP_L1_Controll)
AP_SpdHgtControl/AP_TECS
(libraries/AP_TECS)
adjust_altitude_target (altitude.cpp), update_alt
update_flight_mode (ArduPlane.cpp)
pilot input (tx) -> RC_Channel
autonomous modes call calc_nav_roll
manual modes convert RC_Channel to targets (nav_roll_cd)
calc_nav_roll, pitch, calc_throttle
copy L1 output to attitude targets
(nav_roll_cd, nav_pitch_cd)
stabilize (Attitude.cpp), stabilize_roll, _pitch, _yaw
attitude targets (nav_roll_cd) passed to rollController
outputs placed in SRV_Channels::k_aileron, k_elevator, etc
RollController, PitchController, etc
(libraries/APM_Control)
SRV_Channel
(libraries/SRV_Channel)
set_servos (servos.cpp)
SRV_Channels::k_aileron, etc converted to PWM and output
Hardware Abstraction Layer (HAL)
Hardware PWM Output

Figure 9: High-level flow chart of ArduPlane architecture [7].

The custom build of ArduPlane that this project uses contains custom code for communicating with the on-board Arduino. The flight controller is restricted to having only six output ports for actuation. With four motors and three servos, the drone requires more outputs than the flight controller is capable of supplying. The solution to this problem was to offload servo control to the Arduino. Rather than try to output servo PWM commands via hardware, the flight controller instead sends the calculated PWM values to the Arduino as I^2C messages.

The implementation uses the existing I^2C device manager class in ArduPlane. The Arduino is added as a new device with address 0x8 (akin to a sensor), allowing the flight controller to send data through that class in code. Packets are then sent right after new PWM values are calculated (50Hz). Figure 10 shows a snippet of this implementation.

```
if (function.get() == k_tiltMotorLeft ||
    function.get() == k_tiltMotorRight ||
    function.get() == k_rudder) {

    uint8_t* buf = SRV_Channels::arduino_servo_i2c_buf;
    buf[0] = (uint8_t)function.get();
    buf[1] = output_pwm & 0xff;
    buf[2] = (output_pwm >> 8) & 0xff;
    buf[3] = 0;

    SRV_Channels::arduino_servo_i2c_dev->transfer(buf, sizeof(buf), nullptr, 0);
}
```

Figure 10: Code snippet for sending data to Arduino over I^2C

On the Arduino, the implementation uses the in-built Wire and Servo libraries. Wire connects the Arduino to the I^2C bus with address 0x8 while Servo manages the PWM signals sent to the servos. On a loop, the Arduino continuously sends the saved PWM values to the servos. On an I^2C receive event, the corresponding saved servo PWM is updated to the received value. Figure 11 shows a snippet of this implementation:

```
// Callback for receiving PWM values from FC
void receiveEvent(int howMany)
{
  uint8_t buf[3];
  while(Wire.available()) // loop through all
  {
    Wire.readBytes(buf, 3); // Read function and PWM
    Wire.read();            // Read padding byte

    uint8_t func = buf[0];
    uint16_t pwm = buf[1]+(buf[2]<<8);

    switch(func){
      case RUDDER:
        rudder_pwm = pwm;
        break;
      case LEFT_TILT:
        left_tilt_pwm = pwm;
        break;
      case RIGHT_TILT:
        right_tilt_pwm = pwm;
        break;
      default:
        break;
}}}
```

Figure 11: Arduino I^2C callback function.

Tailoring the ArduPlane flight software to this specific project required the customization of many ArduPlane configuration parameters [7]. By default, ArduPlane assumes the drone is fixed-wing and connected via radio, with no support for VTOL transitions and quadcopter flight. These are only two of the main differences between VTOL and the default configuration that needed to be changed. Other major functionality changes include battery settings and monitoring, frame settings and motor ordering, ground station and radio fail safes, and additional sensor integration. Table 4 shows a non-exhaustive list of the most important parameter changes.

Table 4: List of major ArduPane parameter modifications [7]

Parameter	**Value**	**Description**
AHRS_ORIENTATION	6	6: Yaw270. The default orientation of the flight controller needs to match that of the UAV frame. On VTOL, the FC is mounted with a -90deg yaw offset, so this parameter adjusts for that.
BATT_MONITOR	4	4: Analog Voltage and Current. Enables the on-board current and voltage sensors for the battery, used to monitor battery health and trigger fail safes. Additional board-specific parameters also need to be set (e.g., BATT_VOLT_PIN and BATT_CURR_PIN).
PLND_ENABLED	1	1: Enabled. Enables precision landing flight logic.
PLND_TYPE	1	1: Companion Computer. Specifies that precision landing commands will be sent over MAVLink by a companion computer (Jetson).
Q_ENABLE	1	1: Enabled. For QuadPlane functionality, enables quadcopter & VTOL capabilities.

Q_FRAME_TYPE	18	18: BetaFlightXReversed. Specifies the configuration of motors in quadcopter flight, needed for proper motor mixing.
Q_TILT_ENABLE	1	1: Enabled. Specifies that the VTOL has a tilt-rotor setup, unlocking tilt transition capabilities.
Q_TILT_MASK	10	10: Motors 2 and 4. Specifies which motors are tiltable. Motors 2 and 4 correspond to the two front motors.
Q_TILT_TYPE	2	2: Vectored Yaw. Specifies that the motors can be tilted independently of each other, giving the VTOL more control over yaw and roll during flight to make up for the lack of control surfaces.
Q_M_PWM_TYPE	4	4: DShot150. Specifies the communication protocol between the flight controller and ESC. DShot150 is supported by the chosen ESC and a lower transfer rate results in more robustness.
Q_GUIDED_MODE	1	1: Enabled. Enables the use of VTOL in guided mode, allowing the UAV to be flown with automatic transitions from a ground station.
RNGFND1_TYPE	25	25: BenewakeTFminiPlus-I2C. Enables the use of a downward-facing ToF and specifies the model used.
SERVOx_FUNCTION	Varies	By default, output channels 1-4 are used for fixed-wing control (i.e., rudder, ailerons, elevator, throttle) while 5-8 are for motors. To allow the FC to control the motors directly, channels 1-4 were remapped to motors 1-4. Ailerons, elevator, and throttle were removed while adding servos for the left and right tilt motors.

2.1.2.2 Insulator Health Classification

The software running onboard the Jetson Nano classifies the condition of insulators captured within the field of view of the attached camera. It is imperative that the system is capable at detecting the presence of insulators and subsequently classifying their states. The algorithm must be robust enough to discern nuanced variations in the insulator conditions. Figure shows examples of the various anticipated insulator conditions.

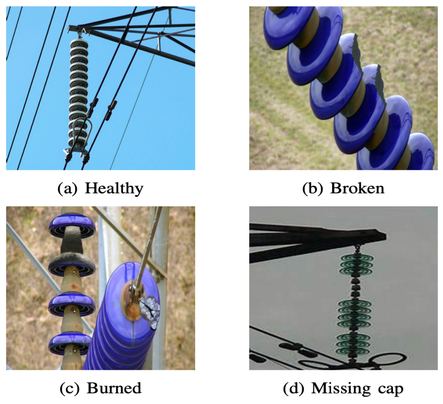


Figure 12: Failure modes of insulators present on transmissions line [1]

Usually, deep learning models exhibit improved classification performance when the primary subject of interest fills the frame; essentially, the input should be cropped to focus solely on the subject. This introduces two distinct tasks for the algorithm: identifying the subject's location through segmentation and subsequently classifying it. Segmentation and classification can be integrated into a single model or executed as separate stages. There are single-stage models (e.g., YOLO) which are notably more compact compared to two-stage approaches involving multiple deep learning models. They are thus, more efficient in handling both tasks as they necessitate fewer computational steps, less memory usage, and reduced time [8]. Hence, they are well-suited for deployment on edge devices like the Jetson Nano, where resources are limited.

Figure 13 shows the architecture that was designed in MTE 481, but due to limited hardware and timeline issues, the project did not contain the feature where the image of the faulty insulator that was detected is sent to the asset management team along with geo-location information [5]. The other difference from the figure is the fact that a USB webcam was used instead of the Raspberry Pi cam, which was repurposed for providing visual odometry for landing (more information on that in the next section).

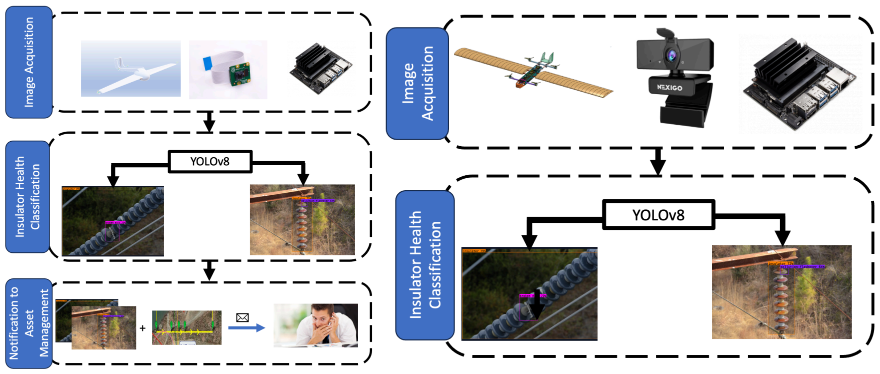


Image Acquisition
YOLOv8
Insulator Health Classification
Notification to Asset Management
Image Acquisition
NEXIGO
YOLOv8
Insulator Health Classification
Figure 13: Old Inspection architecture (left); New Inspection architecture [5] (right)

The YOLO-v8 model was pretrained on existing datasets, and the ONNX model is saved and used for the system. The model containes several labels pertaining to the different ways an insulator can be detected as defective. These labels can be seen in Figure 14. The model is executed using the ONNX runtime with execution providers implemented for NVIDIA CUDA and TensorRT, which take advantage of acceleration hardware. Through some latency test between normal CPU, CUDA, and TensorRT execution providers, the CUDA execution provider was selected due to lower resource usage and fast initialization time.

```
Glassdirty
Glassloss
Polymer
Polymerdirty
Two glass
broken disc
insulator
pollution-flashover
snow
```

Figure 14: Label names used for the machine learning model

The image is pre-processed to grab the image data in an acceptable format for the ONNX model and was then fed into model via Gstreamer. Gstreamer allows video to flow into the machine learning model. Model class output that exceed a confidence threshold is then logged to disk.

2.1.2.3 Mission Planning/Control

The flight controller, implemented by ArduPilot, is capable of navigating through a scripted series of waypoints, along with other directives/commands like takeoff, loiter, landing, and mode transition. To make the mission routes flexible and handle unexpected events, the onboard Jetson monitors and issues directives to the lower-level embedded flight controller during flight. The firmware Ardupilot implements MAVLink, a serial protocol that facilitates all the transactions between the flight controller and any companion computer. The protocol itself is also open-sourced, and a package abstracts the low level encoding/decoding functions and provides an easy-to-use API so that mission control could be scripted in Python. The script handles OS signals and handles expected exceptions.

The contractors add the GPS coordinate of every transmission tower so that VTOL can traverse through the list in order. This mission can be paused if VTOL's battery is low, in which case the nearest recorded dock can be found and docking proceeds.

To test that the mode transitions and that the script handles the OS signals robustly, Software-in-the-loop (SITL) simulation for Ardupilot is used.

2.1.2.4 Precision Landing

The landing dock has a limited amount of space for the drone to land. To ensure that the VTOL can control its position within tight margins, additional Computer Vision algorithms are integrated to track the dock position as the VTOL descends, thereby improving the attitude estimation accuracy compared to just GPS reference. This also makes dock deployment easier since their locations do not need to be exact, and the error can be rectified with this active correction.

A visual marker is installed inside the dock landing pad. There are mature and robust fiducial marker detection algorithms implemented in open-sourced projects, like OpenCV. Rather than creating a marker from scratch, ArUco tags are tried-and-true industry-standard markers that have been heavily used in the automotive industry. OpenCV implements functions for marker corner detection of this tags, and transformation matrix formulation from those corners to provide an estimation of where the tag is [9].

Before the corners can be detected, intrinsic parameters of the camera must be estimated to correct any distortion introduced by the lenses. In a compact camera with a high Field of View (FOV), like the Raspberry Pi 2 camera used on the VTOL, the pixels around the edges of the image frame no longer map to the 3D scene it points at with straight lines. In other words, parallel lines in the real world look curved and hyperbolic in the image. Thus, it's necessary to infer parameters to undo that distortion, so that the detected corners map to the right.

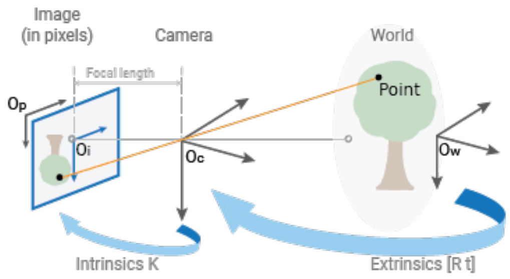


Figure 15: Intrinsic parameters describe how light projects onto the image frame, and the extrinsic parameters describe where the camera in the global reference frame [10]

First, images of the calibration chess board are taken and fed into the calibration pipeline, wherein a Harris Corner detection algorithm, with an additional refinement of the corners detected are done on the chess board image. The corners are then drawn onto the image and the points are saved. These points are then fed into another function within OpenCV called *calibrateCamera()* that finds the camera intrinsic and extrinsic parameters from several views of a calibration pattern [9]. If for any reason there is a no-good image that needs to be undistorted, the *getOptimalNewCameraMatrix()* function is used that outputs a new camera calibration matrix that is based on a free scaling parameter (it passes the distortion coefficients given by *calibrateCamera()* and helps undistort the image) [9].

An undistorted transformation map is also created using the *initUndistortedRectifyMap()* function to compute the joint undistortion and rectification transformation and represents the result in the form of maps, which later gets passed on to a function called *remap()* that applies a generic geometrical transformation to an image. The undistorted image looks like original, as if it is captured with a camera using the new camera matrix with zero distortion present. Once this is done, the image is then saved and the calibrated matrix that was created is saved.

With the camera calibrated for its intrinsic parameters, the next step was generating an ArUco tag with a specific tag ID. Figure 16 shows the ArUco tag used with an ID of 950 and has a size of 20 cm. The tag was also generated form the Original ArUco dictionary from [11]. This tag was attached onto a flat plate to easily detect the marker from the camera (and for portability reasons to not crumple the paper).

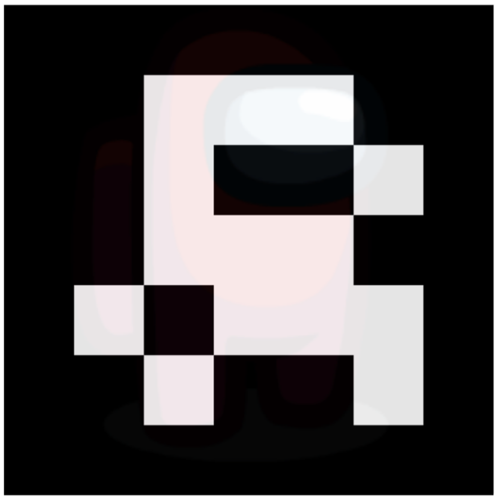


Figure 16: ArUco tag generated with an ID 950 [9]

aruco.py is a custom ArUco detection algorithm created by the team to extract metrics such as the X and Y translation, as well as Z altitude of the camera from the tag itself. The script was adapted from Tiziano Fiorenzani's youtube tutorial [12]. The algorithmic structure can be seen in Table

Table 5: ArUco Marker Pose Estimation algorithm

ARUCO POSE ESTIMATION	
1	***Define tag ID***
2	***Setup camera calibration matrix:*** #Load in the camera calibration matrix saved from the calibration step
3	***Setup object matrix***
4	***Define ArUco dictionary*** #Here is where the detector is setup that will be used later for marker detection inside frame
5	***Capture video camera and use camera calibration matrix*** #Capture video feed and undistort incoming camera feed using *getOptimalNewCameraMatrix()*
6	***While True:*** pertains to script running forever and be broken whenever necessary

```
7           Undistort each frame from incoming feed. Convert image into grayscale
8           Detect marker inside frame #Using detectMarkers() from the detector that was setup
9           If ID is detected
10              Use solvePnP() to find object pose on a 3D-2D point correspondence # Returns the rotational and
                translational vector of our axis based on the ArUco position
11              Draw detected markers and frame axes #Using drawDetectedMarkers() and drawFrameAxes()
12              Print tag position in camera frame
13              Obtain rotation matrix of the tag to the camera. Also obtain position and attitude of camera w.r.t.
                marker
14          Display frame
15      Break
```

The *solvePnP()* algorithm (PnP refers to the Perspective-n-Point (PnP)) helps in estimating the object pose, given a set of object points, their corresponding image projections and the camera's intrinsic parameters (along with the distortion coefficients) that were created at the camera calibration step (see Figure 17 below).

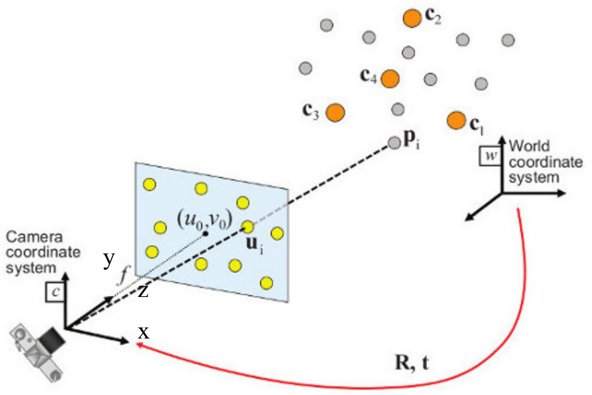


Figure 17: solvePnP() Diagram [13]

The general idea behind this strategy is to solve for the transformation matrix that maps a set of 3D points onto their corresponding 2D projections onto an image plane. This transformation matrix consists of rotation and translation parameters that describe the object's pose relative to the camera. The points that are in the world reference frame are projected into the image plane using the perspective projection model and the camera intrinsic model that was found. The estimated pose is given by (1), where T_w^c is the transformation (mentioned earlier) from the world to camera coordinate system. Also, in (1), we can observe the rotation and translation vectors contained within the transformation matrix.

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = T_w^c \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} => \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{3\times3} & t_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{1}$$

These matrices are then used to project the 3D points detected in the world coordinate system into the 2D camera plane, as seen below. This results in (2). Note that A and Π relate to the intrinsic parameters of the camera and the perspective projection model respectively.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A\Pi T_w^c \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{2}$$

Where, $\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$ corresponds to the points on the 2D camera plane. In the end, using *solvePnP()* helps us project the corners of the ArUco marker onto the camera frame and make it detect it with the aforementioned detection methods. The complete script can be seen on our Github repository. With the script created, the complete software for precision landing was integrated into the mission planning script – *meishur.py*. The results of this implementation can be seen in the section - Testing and Performance

2.2 Modifications/Deviation from Original Design

The majority of the design changes between the initial design concept to the final symposium design revolved around Design for Manufacturability (DFM) considerations.

- Additional changes were added from lessons learned from test flights.
- The position of the rotors was moved inwards for stiffness purposes.
- Flight surfaces were removed due to redundancy. The quadcopter kinematics allowed for control in all 6DOF already.

Geometry was changed for DFM purposes. The curved surfaces were replaced with flat sections wherever possible and made with sheet stock. Wherever complex geometry was required, such as in the tail assembly and nosecone, designs were made to optimize printability using 3D printers.

2.3 Construction/Manufacturing

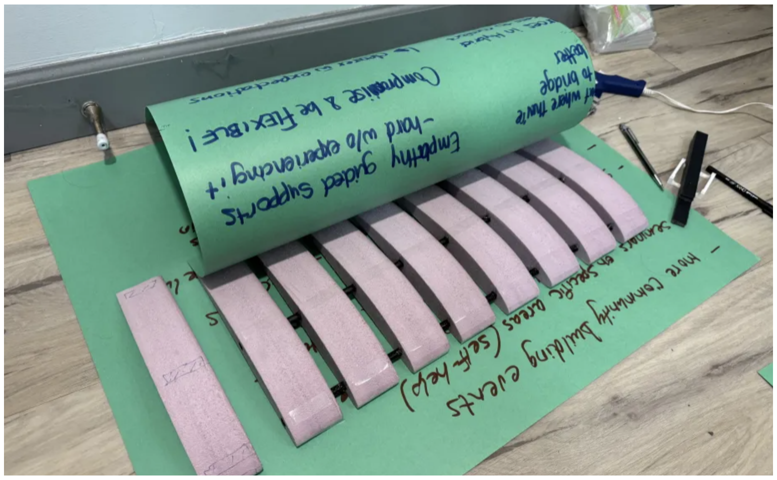
Due to DFM considerations, mechanical changes were implemented relatively quickly, often same-day as the design changes. Almost all of the hardware parts were made through laser cutting or 3D printing.

Sheet stock was either laser cut at the Rapid Prototyping Centre (RPC) in E5, or waterjet cut at the E5 Student Machine Shop (ESMS). All of the machining operations, for parts like the aluminum motor mounts and tilt shaft spacers, were done at the ESMS as well.

The E5 CNC router was also used to mill airfoil sections out of closed cell foam, courtesy of Sedra Student Design Centre manager Graeme Adair.

Personal 3D printers were used to create more complex mechanical parts, such as the VTOL bearing blocks and dock hatch geartrains. Available printers include the Creality Ender 3, Prusa Mini, and Voron 0.1. All of the printed VTOL parts were made from Carbon-fibre filled PETG (CF-PETG), which exhibits high stiffness and impact resistance, as well as UV resistance for outdoor exposure.

<u>Airfoils</u>: The airfoils were made to be as stiff and lightweight as possible. They are composed of a core of two carbon fibre tubes, with CNC routed foam wing spars hot glued on top to form the NACA 6412 airfoil profile. Ample double-sided tape was applied to the foam, and stiff card stock was draped on top of the foam to sheath the wing.


Empathy guided supports
-hard w/o experiencing it
Compromise & be flexible!
more community building events (self-help)
Figure 18: Exposed wing spars during assembly. Foam core is covered with double-sided tape, and card stock is pressed on.

The tail assembly was predominantly 3D printed, due to its complex geometry and relatively small size. Stiffeners made from carbon fibre rod were embedded in the 3D printed airfoil with epoxy.

<u>Chassis</u>: The chassis was designed to be as simple as rigid as possible. It was built predominantly from sheet stock, cheap plywood for the prototypes, and polycarbonate for the final version. Interspersed were 3D printed stiffeners, which gave rigidity to the chassis and provided mounting points for internal hardware.

The main panels were first screwed together by the stiffening brackets, and cosmetic panels would be hot glued on. This was a fast and reversible method of assembling the chassis, while maintaining access to the internal electronics.

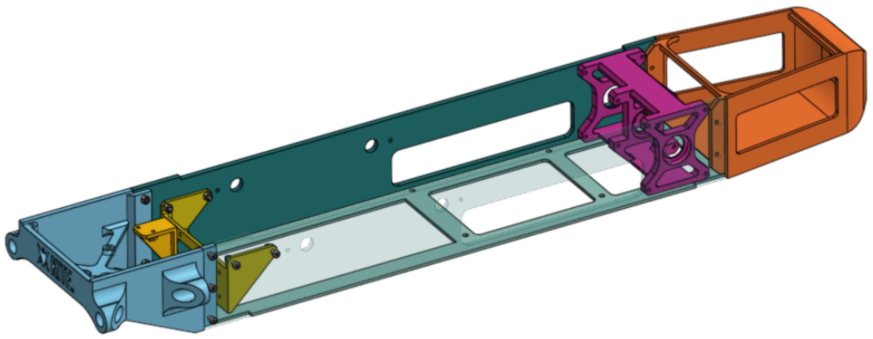


Figure 19: Empty chassis in OnShape CAD.

Propulsion: The rotor shaft assemblies were mostly 3D printed from CF-PETG, with some critical components, such as the motor mounts and retaining spacers, being machined out of 6061-T6 aluminum. The tilt struts were made from off-the-shelf carbon fibre tubes, with appropriate mounting holes milled.

To tilt the rotor shafts, small servo motors were connected to the rotor shaft assemblies via linkage assemblies. This allows for independent tilting of each of the front rotors.

2.4 Commissioning

Before test flights out in the field, several steps were always taken for system bring-up and evaluation. The first is the calibration of the following sensors: accelerometer, gyroscope, and barometer. This process involved saving expected IMU measurements at all orientations of the drone by placing it on the ground in all six directions. The next sensor to be calibrated is the compass. The magnetometer minimum and maximum values must be mapped to a real-world heading. This involves pointing every axis of the compass towards the Earth for a few seconds on top of random figure eight movements. The IMU calibration is verified by comparing estimated attitude with expected (e.g., the drone can detect a 45deg roll to the left). The compass calibration is verified by comparing the estimated heading of the drone with the compass on a smartphone. A less than 10deg error between the two was found to be acceptable. All these calibrations are implemented by ArduPilot.

Other pieces of hardware that need to be verified are motors and servos. Motors are spun at 5% throttle individually with propellors detached to ensure correct direction and motor numbering. Servos are verified by switching between MANUAL and QSTABILIZE modes in ArduPilot. The tilt motors should be tilted directly forward in MANUAL and directly up in QSTABILIZE. The rudder should move left when the drone rolls right and vice versa.

For flight maneuvers, the first step was to simulate using ArduPilot SITL whenever possible. This includes functions like basic hovering, point-to-point flight, precision landing, and VTOL transitions. Once the simulation confirms that the flight logic is in good working order, those same functions are slowly tested in sequence with the real drone.

2.5 Testing and Performance

Testing was conducted often, whenever changes were made to firmware or hardware design.

As per Transport Canada regulations, two FYDP team members obtained Drone Pilot Certifications, and the drone was filed under registration number C-2401640990.

Testing was conducted in an open field in North Campus, directly north of Columbia Lake. The test location was chosen due to its remoteness from any unrelated bystanders, as well as its flat area and far visibility.

There were always at least 3 people at each test flight:

1. Pilot: Responsible for remotely controlling all flight operations.
2. Spotter: Responsible for staying close to the VTOL before, during, and after flight.
3. Recorder: Responsible filming/recording each test flight for later review.

The precision landing of the VTOL was also tested on the same grounds, wherein the VTOL was commanded to go to a height of 5m. The detection results (of the ArUco tag) while testing the precision landing of the VTOL can be seen below in Figure 20. We can observe that the VTOL senses the tag at ~4.9m away. The altitude values did fluctuate between 4.5 and 5.6, but did have a decent altitude estimation. We can also observe that the camera calibrated in a satisfactory condition wherein the output video seems to be undistorted and flattened out fairly well.

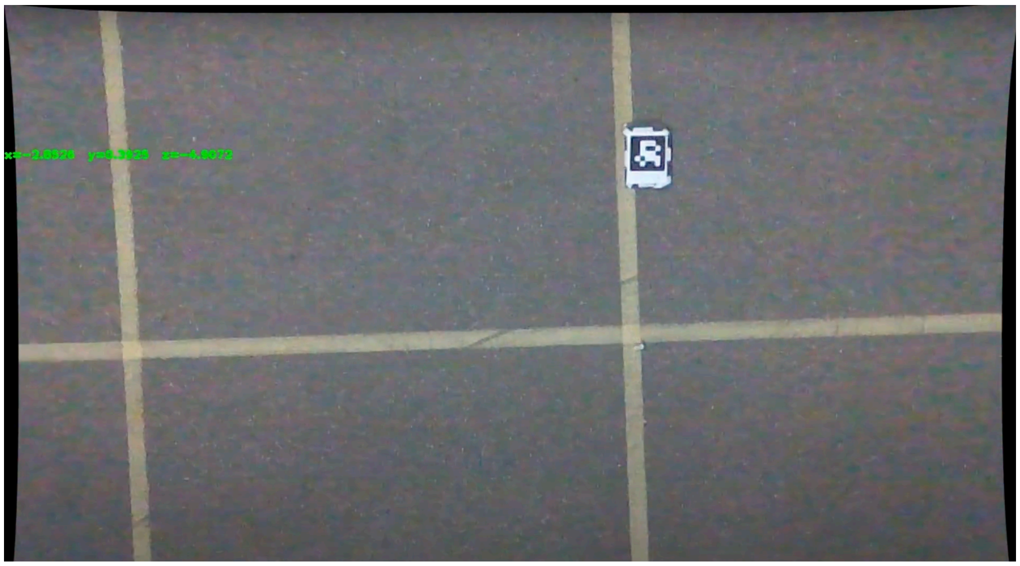


Figure 20: Detection results of VTOL in landing mode.

3 Schedule and Budgeting

3.1 Schedule

The schedule was carried over from MTE481 [5]. Despite a disastrous crash just before the final advisor meeting, VTOL development and testing was finished in time for symposium. Figure 21 shows the Gannt chart from the previous report.

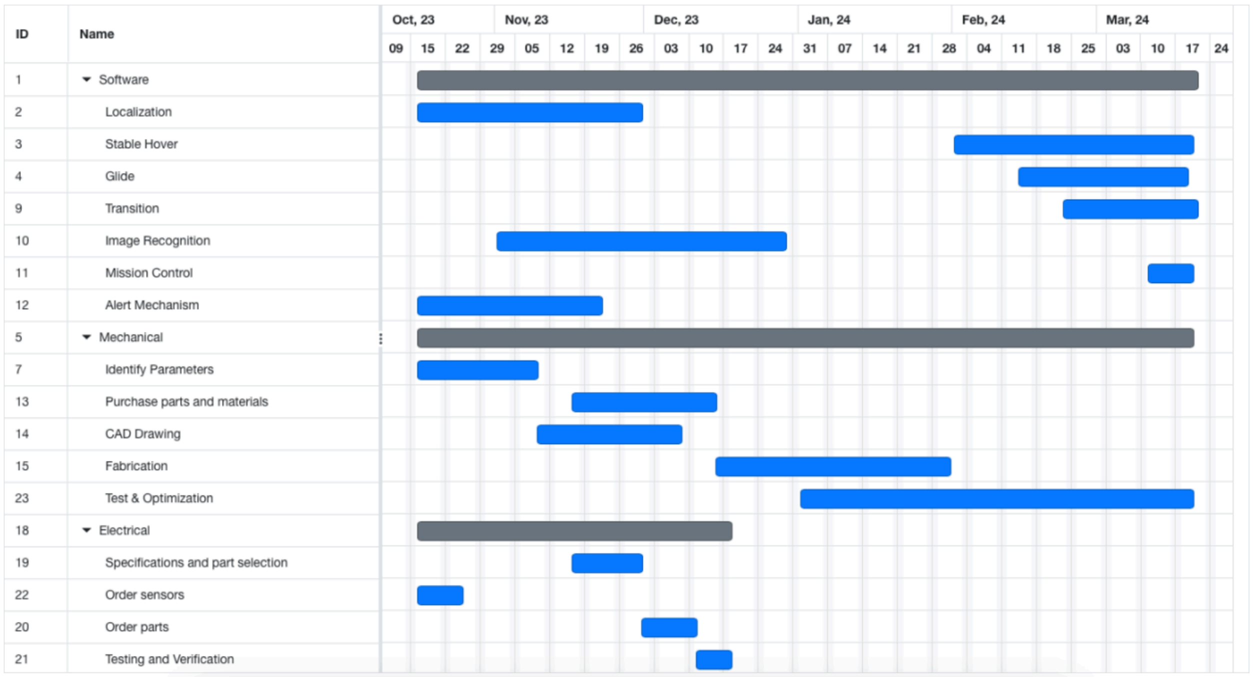


Figure 21: Gantt chart for project timeline scheduling [5]

3.2 Budget

The initial budget for the capstone came from two sources: the $750 provided by the MME department, as well as contributions from personal funds.

Additionally, external funding was sought. This came from two sources; industry and awards. Relevant industry stakeholders, such as existing inspection drone manufacturers, as well as transmission line operators were contacted. None of these contacts resulted in financial support though. Several capstone design awards were applied to as well. Team Kite had success with the Norman Esch Entrepreneurship Award, making it to the final round and winning $5000. The MME-provided funds plus the Norman Esch award winnings amounted to $5750, covering all capstone project-related development costs.

3.3 Projected Cost

The originally projected cost at the time of the MTE481 project report was $1156.20 [5]. This included the forecasted material shipping costs. The original breakdown can be found in Appendix A: Original Budget**.**

3.4 Actual Cost

An itemized list of the final project expenditures can be found in Appendix B: Final Expenditure Breakdown. The final total cost was $1726.86, and consists of two main costs – hardware, and associated shipping and duties costs.

Most of the costs came from hardware purchases. These either came from online sources, or local suppliers. Amazon.ca was used extensively for hardware, due to its quick shipping and large selection of specialized parts. Locally, the E3 EMS was used for all the sheet stock, like plywood and polycarbonate sheets, as well as fasteners.

4 Conclusions and Recommendations

The final VTOL and dock architecture is sufficient in meeting the major requirements of the proposed insulator inspection problem. The choice of sensors, actuators, flight control software, computer vision framework, and UAV frame have led to a drone fully capable of autonomous flight and insulator detection. The dock is also capable of protecting and housing the drone.

However, there are recommendations for the future of the project. Time constraints descoped the functionalities of dock charging and alert system. Much of the base support structure has already been laid for these and proper implementation would lead to the full realization of all requirements. In addition, the drone and dock materials can be upgraded. Given their prototype nature, the materials are relatively cheap, flimsy, and certainly not industry-grade. Reducing the use of plywood and 3D printed material will greatly increase the strength and durability of both the drone and the dock.

5 Teamwork Effort

The team still comprised of original five mechatronics students from MTE 481 [5], who possessed varying experiences from industry and academic courses. The team came together to with a common goal – to apply the great experiences they have had in their time at the University of Waterloo and make something great. From valuable input gained from the course advisors, the team put together a plan to create a novel VTOL inspection system – Project Kite. (formerly known as CaelVision).

Samuel spearheaded the drone's mechanical development, leveraging his extensive expertise in machine design and manufacturing to craft the various iterations of the VTOL design. Not only did he conceive these iterations, but he also brought them to fruition through his adept machining and additive manufacturing skills. In tandem, Stephen and Joshua were instrumental in crafting the control software pivotal to the realization of Project Kite. Their proficiency in embedded software facilitated the seamless integration of the sensor suite, a task in which they collaborated closely with Samuel. Together, they co-designed the SanBan PCB board, streamlining sensor connections for enhanced reliability and ease. On the Computer Vision front, Neil and Stephen delved into creating sophisticated systems such as the insulator classification and precision landing systems crucial to the VTOL's operations. Their efforts extended to seamlessly integrating these features using MAVlink and ArduPilot protocols. Meanwhile, Lucas concentrated on developing the landing dock, leveraging his expertise in manufacturing and mechanical design. Through meticulous planning and construction, he fashioned the dock, a cornerstone of the project's demonstration and symposium, with invaluable assistance from Joshua.

6 References

[1] A. Alahyari, A. Hinneck, R. Tariverdi and D. Pozo, "Segmentation and Defect Classification of the Power Line Insulators: A Deep Learning-based Approach," [Online]. Available: https://arxiv.org/abs/2009.10163. [Accessed 02 12 2023].

[2] S. Harjo, "Partial Discharge in High Voltage Insulating Materials," [Online]. Available: https://www.researchgate.net/publication/301535335_Partial_Discharge_in_High_Voltage_Insulating_Materials. [Accessed 8 April 2024].

[3] RAX, "Types of Insulators in Overhead Lines: The Ultimate Guide," [Online]. Available: https://www.hbjinyong.com/insulators-in-overhead-lines/. [Accessed 8 April 2024].

[4] Hydro One, "SUMMARY OF OM&A EXPENDITURES," [Online]. Available: https://www.hydroone.com/abouthydroone/RegulatoryInformation/txrates/202022_Tx_Rate_Application/HONI_App_Ex_F_20190321.pdf. [Accessed 8 April 2024].

[5] L. Arzoumanian, J. Cai, N. Fernandes, S. Sun and S. Wang, "Project MeishurVision," Waterloo, 2023.

[6] Github, "ArduPilot," [Online]. Available: https://github.com/ArduPilot/ardupilot. [Accessed 6 April 2024].

[7] ArduPilot, "Complete Parameter List," [Online]. Available: https://ardupilot.org/plane/docs/parameters.html. [Accessed 7 April 2024].

[8] Z. Li, Y. Wang, N. Zhang, Y. Zhang, Z. Zhao, D. Xu, G. Ben and Y. Gao, "Deep Learning-Based Object Detection Techniques for Remote Sensing Images: A Survey," *Remote Sensing,* vol. 14, no. 10, p. 2385, 2022.

[9] OpenCV, "Camera Calibration and 3D Reconstruction," [Online]. Available: https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html. [Accessed 02 04 2024].

[10] Mathworks, "What Is Camera Calibration?," [Online]. Available: https://www.mathworks.com/help/vision/ug/camera-calibration.html. [Accessed 02 04 2024].

[11] O. Kalachev, "ArUco markers generator!," [Online]. Available: https://chev.me/arucogen/. [Accessed 01 04 2024].

[12] T. Fiorenzani , *PRECISION LANDING | with OPENCV and ARUCO Markers,* Youtube, 2018.

[13] OpenCV, "Perspective-n-Point (PnP) pose computation," [Online]. Available: https://docs.opencv.org/4.x/d5/d1f/calib3d_solvePnP.html. [Accessed 02 04 2024].

[14] I. Archive, "The Wayback Machine," 2014. [Online]. Available: https://web.archive.org. [Accessed 25 10 2023].

[15] S. Anjum, S. Jayaram, A. El-Hag and A. N. Jahromi, "Detection and classification of defects in ceramic insulators using RF antenna," *IEEE Transactions on Dielectrics and Electrical Insulation,* vol. 24, no. 1, pp. 183-190, 2017.

[16] MAVlink, "MAVLink Developer Guide," [Online]. Available: https://mavlink.io/en/. [Accessed 01 04 2024].

[17] MathWorks, "What Is Camera Calibration?," [Online]. Available: https://www.mathworks.com/help/vision/ug/camera-calibration.html. [Accessed 7 Apr 2024].

Appendix

Appendix A: Original Budget

The following budget was estimated in the MTE481 project report [5]:

Table 6: Originally estimated bill of materials.

Qty	Component	Cost	Existing?
1	560uF Capacitor	$0.61	Y
2	22uF Capacitor	$0.66	Y
4	XING2 2207 + Props	$100.35	N
3	D-625MW	$119.97	N
1	1k	$0.14	Y
1	2k	$0.14	Y
2	ALTIMU-10_V6	$59.90	N
1	HC-05	$18.90	Y
1	GY-GPSV3-M9N	$37.24	N
2	F4 V3S Plus	$105.28	Y
1	KO45	$70.05	Y
1	Jetson Nano	$327.81	Y
1	Raspberry Pi Camera	$61.64	Y
1	Wi-Fi Adapter	$28.24	Y
1	LM2596S-5	$10.46	Y
1	Work F450	$33.89	N
1	LiPo	$60.69	N
1	Oracover	$29.98	N
1	Carbon Fiber Tube Set	$36.95	N
1	Polulu Duties	$53.30	N
Total:	**$1,156.20**		
Paid-to-Date:	**$532.27**		

Appendix B: Final Expenditure Breakdown

The final project spend is broken down as follows:

Table 7: Final itemized expenditures.

TYPE	ITEM	QTY	COST	SUPPLIER
Raw Materials	2PCS CARBON FIBRE TUBE 12X10X500MM	1	$9.13	ALIEXPRESS
	2PCS CARBON FIBRE TUBE 8X7X500MM	3	$17.67	ALIEXPRESS
	ERYONE CF PETG 1KG BLACK	1	$28.80	AMAZON
	3/8" ACRYLIC RODS, 1/8" BIRCH PLY	1	$3.58	EMS
	3/8" ACRYLIC RODS	1	$0.45	EMS
	JAYO PLA 1KG WHITE	1	$23.72	AMAZON
	1/8" PLYWOOD	1	$4.37	EMS
	1/2" AL ROUND	1	$3.06	EMS
	1/2" AL ROUND	1	$3.27	EMS
	1/16" ACRYLIC	1	$1.51	EMS
	1/4" AL ROUND	1	$2.57	EMS
	INSULATION FOAM	1	$56.13	HOME DEPOT
	1/8" PLY, 1/16" ACRYLIC	1	$10.17	EMS
	ALU ROUND, ACRYLIC	1	$12.31	EMS
	DOCK AXLE ROD	1	$18.06	HOME DEPOT
	DOCK ACRYLIC	1	$56.43	EMS
	DOCK PLYWOOD	1	$99.78	EMS
	DOCK ACRYLIC	1	$46.56	EMS
	CF-PETG, WHITE GLUE	1	$46.32	AMAZON
	BLACK POSTER BOARD	5	$8.36	WALMART
	1/8" LEXAN	1	$6.12	EMS
	1/8' PLYWOOD, 1/16" ACRYLIC	1	$14.52	EMS
Hardware	51466 PROPELLERS	1	$7.21	ALIEXPRESS
	QUADCOPTER FRAME F450	1	$33.89	AMAZON
	5/8" AL ROUND, M3 SCREWS	1	$6.16	EMS
	QTY100 M2 LOCKNUTS	1	$12.95	AMAZON
	M3 SCREWS	1	$0.51	EMS
	QTY20 2RS608 BEARINGS	1	$14.41	AMAZON
	EPOXY, FISHING LINE	1	$19.63	AMAZON
	FASTENERS MISC	1	$13.68	SPAENAUR
	WOOD GLUE	1	$24.65	HOME DEPOT
	FASTENERS MISC	1	$11.19	SPAENAUR
	L BRACKETS	1	$52.03	HOME HARDWARE
	DOCK ACRYLIC, HARDWARE	1	$30.00	EMS

Electronics	XING2 2207 1855KV BLDC	1	$94.14	ALIEXPRESS
	NEO-M9N GPS RECIEVER	1	$37.24	ALIEXPRESS
	ZEEE 6S LIPO BATTERY	1	$60.69	AMAZON
	EC5 MALE BANANA CONNECTOR	1	$15.81	AMAZON
	SAMSUNG 128GB MICROSD CARD	1	$22.59	AMAZON
	TTKK BENCHTOP DC POWER SUPPLY	1	$53.10	AMAZON
	BOLSEN DIGITAL LOAD CELL	1	$16.94	AMAZON
	STM32F405 FC & 60A ESC	1	$82.45	ALIEXPRESS
	5PCS LM2596 DC TO DC CONVERTER	1	$4.68	ALIEXPRESS
	5PCS SG90 9G 360-DEGREE SERVO	1	$7.42	ALIEXPRESS
	WS2812B DC5V LED STRIP	1	$6.68	ALIEXPRESS
	SG90 SERVO	1	$5.42	RIGIDWARE
	SANBAN REV 1.0 COMPONENTS	1	$16.50	DIGIKEY
	10UF SMD CAPS	1	$0.27	EMS
	SANBAN REV 1.0	1	$38.15	JLCPCB
	SOLDER PASTE	1	$16.60	AMAZON
	TFMINI-S LIDAR RANGE FINDER	1	$67.69	AMAZON
	HRB 6S LIPO BATTERY	1	$110.16	AMAZON
	BESOMI HC-05 BT MODULE	1	$21.46	AMAZON
	BINIFUB NEO-8M GPS MODULE	1	$19.48	AMAZON
	ROTOR MOTOR	1	$111.55	ROTOR VILLAGE
	32GB MICROSD	1	$16.02	AMAZON
	SG90 9G SERVO	1	$19.20	AMAZON
	XT90 CONNECTORS	1	$16.94	AMAZON
Misc.	HELPING HANDS	1	$29.30	ALIEXPRESS
	HARDENED STEEL NOZZLE	1	$11.29	AMAZON
	BLACK VINYL WRAP	1	$18.07	AMAZON
	BLACK SPRAY PAINT	2	$33.83	HOME DEPOT
	BLACK ACRYLIC PAINT	2	$4.52	DOLLARAMA
	SYMPOSIUM POSTER	1	$69.47	STAPLES
Total:			$1,726.86	

Appendix C: SanBan Schematic

The electrical schematic of the SanBan is shown below.

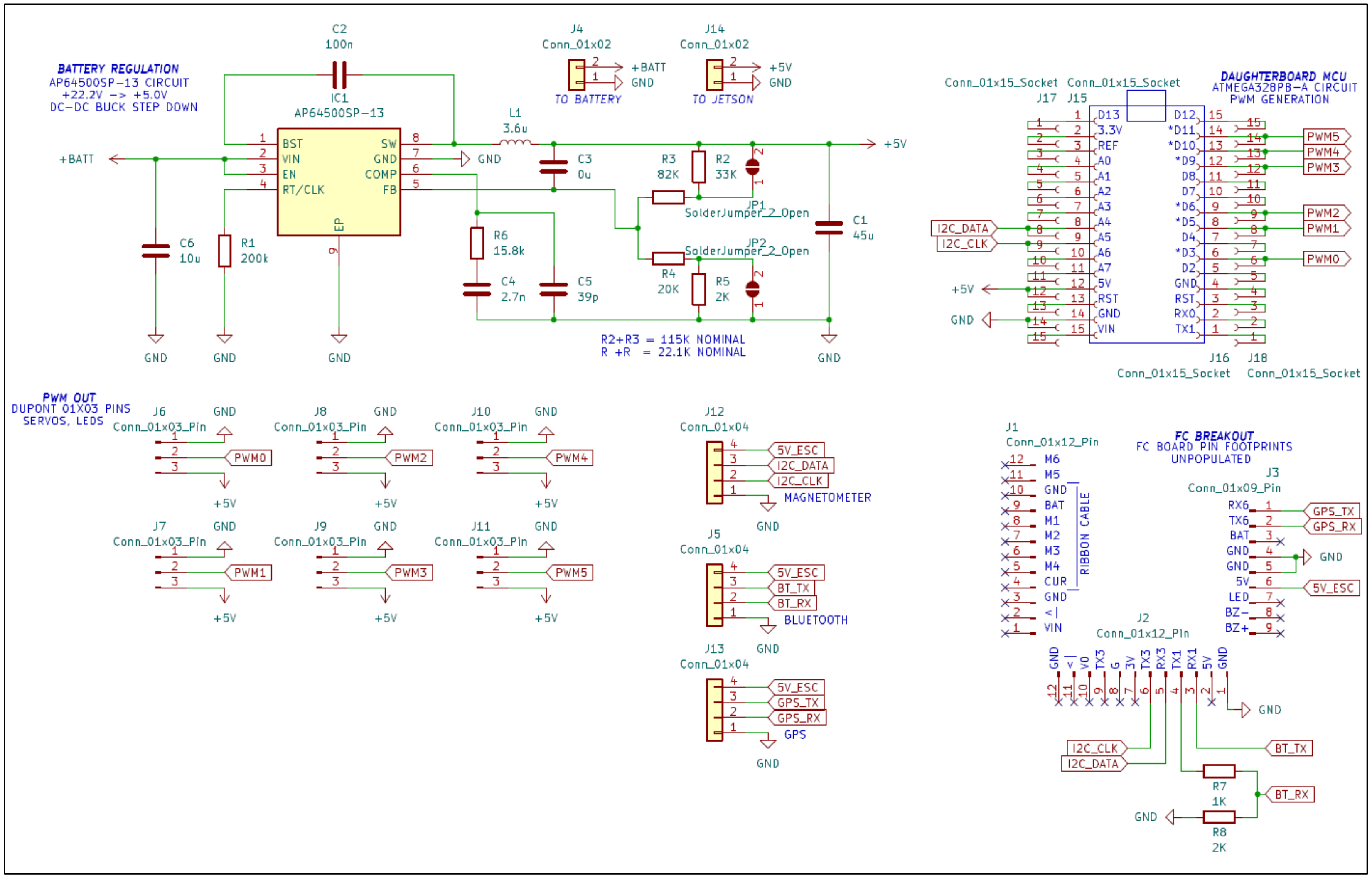


Figure 22: SanBan electrical schematic.