ECE 445

Fall 2024

October 3, 2024

Design Document - Early Response Drone for First Responders

Kevin Gerard, Lohit Muralidharan, Aditya Patel

Team #11

TA: Manvi Jha

Table of Contents

1 Introduction

1.1 Problem

Every week, UIUC students receive emails from the Illini-Alert system regarding crimes that are committed, fires that are occuring, and other dangerous situations to be aware of. With the latest reported median response time of first responders to a 911 call being over 6 minutes in Champaign County [1], the situation to which emergency personnel are responding can drastically change from the initial details that were provided. This problem is even worse in rural areas where, for example, the average response time for Emergency Medical Services (EMS) is over 14 minutes [2]. To best be able to manage the event, first responders need as much accurate information as they can possibly receive. This way, the situation can be handled in a timely manner and the safety of everyone involved is prioritized. Thus, having eyes on the area before arriving can provide emergency response personnel with valuable information about potential hazards, individuals involved, and the severity of the event.

Over the past decade, the use of drones by first responders has significantly increased. For example, the city of Fremont, California's police and fire departments primarily use drones for reconnaissance, documenting crime scenes, and helping with search and rescue operations [12]. However, the use of drones as an early response technology, and in a sense acting as the first responder, is not yet a widely explored concept. Theoretically, this type of drone will reduce response times, improve the safety of all involved individuals, increase efficiency and prioritization, and massively aid understaffed departments [15].

1.2 Solution

Our solution is to construct a cost-effective drone that first responders can deploy and immediately fly to the location of an emergency event. While en route, they could use the drone's on board camera and computer vision capabilities to assess the situation at hand. There are multiple scenarios in which this drone could be particularly beneficial, such as:

- Police: monitor crime scenes and track suspicious individuals; provide aerial surveillance for events with a high density of people (such as sports games, concerts, or protests) to ensure everyone's safety
- Fire: monitor the spread of fire at the location; obtain information on what kind of fire it is (electrical, chemical) and any potential hazards
- Medical: assess the type and number of injuries suffered, and locations of patients

Our drone system consists of 4 different elements: a cloud storage, a backend, a frontend, and the drone itself. In order to create a baseline early response drone, we need to be able to control the drone as well as receive information from the drone such as capture frames, altitude, roll, pitch, and yaw. The capture frames and data will be visually displayed in the frontend. However, this data bundle will first be stashed onto a cloud storage, and when the backend is ready to receive the data, it will retrieve it. If time permits, we want to perform machine learning processing using object tracking and detection models on the backend software. The other data transmission that occurs is the sending of command signals from the frontend to the drone itself. In other words, whenever there is a keyboard click, we can visually see the key click which is uploaded to the cloud storage.

Note: while we do currently plan to use cloud storage via Firebase, as referenced above, we are currently researching possible ways to implement our design without the need for cloud storage. In this design, the drone's cellular chip will directly communicate with our C++ backend for data transmission.

1.3 Visual Aid

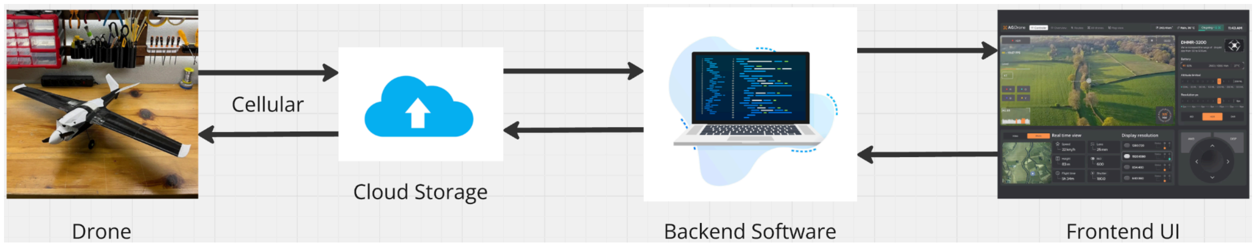


Figure 1: Primary components of the design

The fundamental components of our drone system are pictured above in Figure 1. The drone itself will either be constructed from foam board and will house the core electronics, such as the Printed Circuit Board (PCB), camera, cellular chip, servos, battery, and motor. After powering on the drone, the user will be able to connect to it by interacting with the intuitive User Interface (UI) run on the frontend software on their computer. The backend software will establish a connection between the frontend and embedded software running on the drone and transmit data between two through the cloud storage. Once this is completed, the user will be able to view the feed from the drone's onboard camera, various sensor data, and other critical flight information on the UI. Flight controls, such as yaw, pitch, roll, and throttle, can be inputted via keystrokes on the computer, and this data will then be transmitted through the network nodes to the drone over 4G cellular connectivity. Upon receiving the data, the drone's primary control loop will process it and act accordingly.

1.4 High-level Requirements

1. Maintain and Facilitate Cellular Data Transfers: Our drone will have the ability to receive commands from the front end. In terms of transmission, we will be able to transmit sensor data at a rate of one update per second, whereas the frames we hope to send 2-3 per second but set the minimum to be 1 frame per second.
2. Nodal Software System: Our design must comprise 3-4 primary nodes that efficiently intercommunicate to properly transmit and receive various data. These nodes include a C++ backend, a TypeScript frontend, the embedded drone software, and (optional) Firebase cloud storage as a medium between the drone and backend.
3. Preemptive Warning: For the safety of the drone operator and nearby individuals, the drone will monitor crucial parameters such as altitude, velocity, and battery levels to provide real-time alerts to the user via the frontend UI.
4. Minimal Hardware Response Time: The electronics and other hardware components on the drone must be able to appropriately respond to a given command from the user within 3 seconds with ideal cellular connectivity.
5. Efficient and Stable Power Distribution: The drone's power distribution system will convert the 12V battery supply to regulated 5V and 3.3V outputs for all components, ensuring that motors, sensors, servos, microcontroller, and camera operate within ±5% of required voltage. It will prevent voltage drops under high load (up to 5A for motors) and maintain stability during critical operations, including data transmission and flight control, with surge protection to safeguard sensitive electronics.

2 Design

2.1 Functional Overview & Block Diagram Requirements

2.1.1 Block Diagram

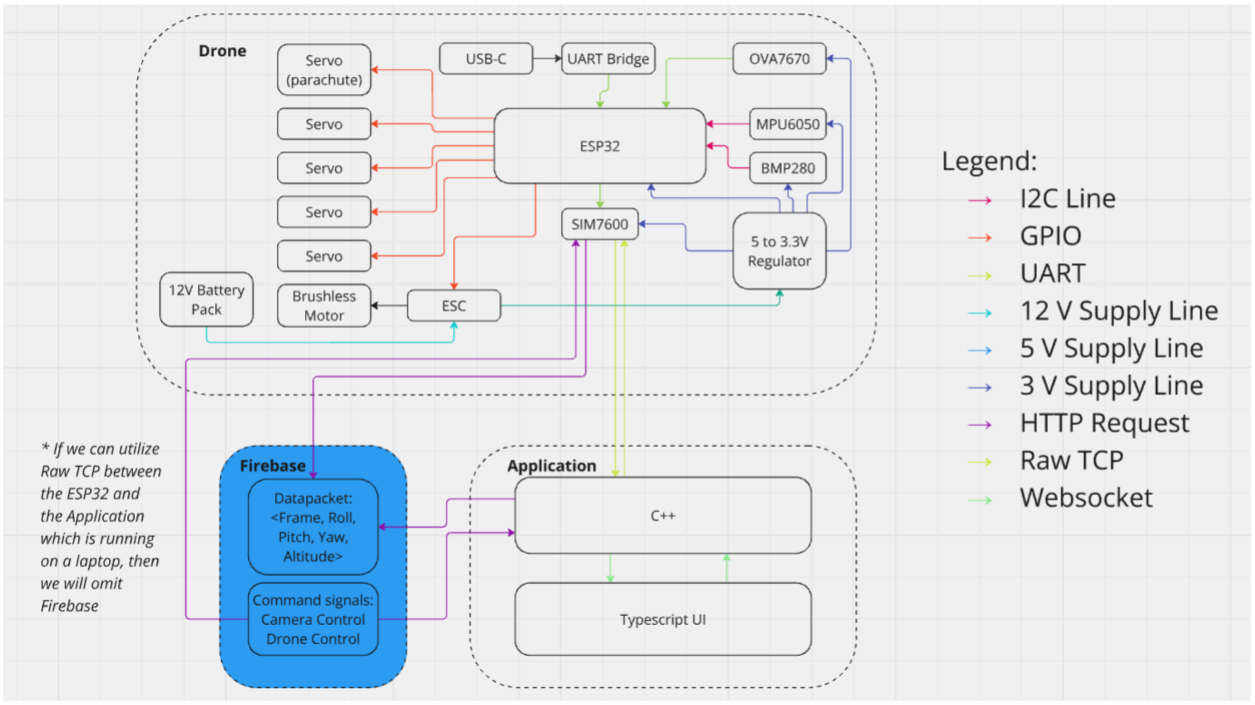


Figure 2: High-level block diagram for the overall drone system

2.1.2 System State Machine

The primary embedded control software for the drone will follow a simple state machine, as can be seen in the figure below. The state machine will encompass 5 states: START; Ready to Fly (RTF); FLIGHT; ERROR; SHUTDOWN. These states will all accomplish unique tasks to ensure safe operation of the drone, reliable uptime of data transmission, and proper interaction with various hardware components. Primary inputs to the state machine include data from the drone's onboard sensors and flight control inputs from the user, which will then be used to determine the appropriate state.

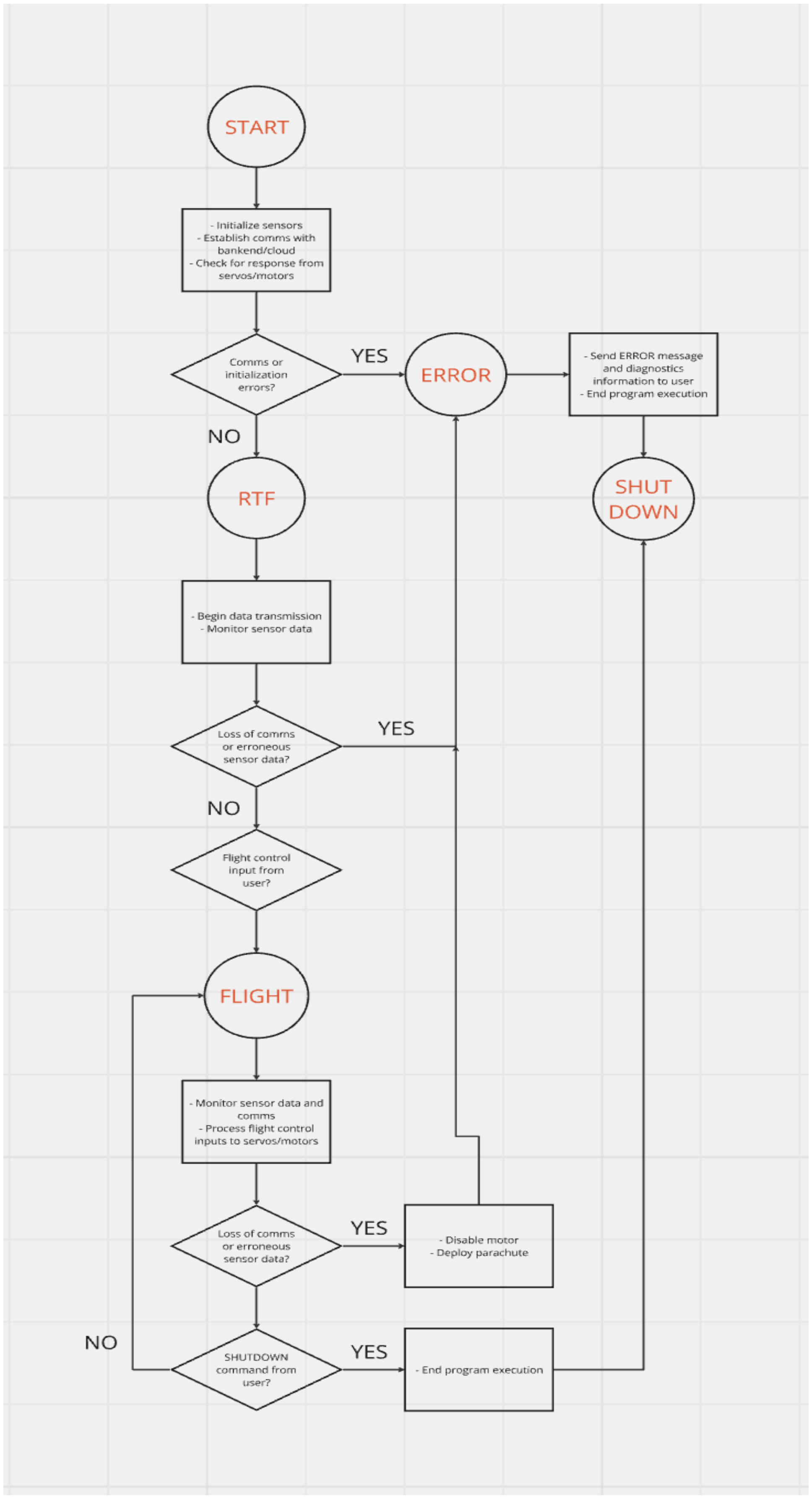

START
- Initialize sensors
- Establish comms with bankend/cloud
- Check for response from servos/motors
Comms or initialization errors?
YES
ERROR
- Send ERROR message and diagnostics information to user
- End program execution
NO
RTF
SHUT DOWN
- Begin data transmission
- Monitor sensor data
Loss of comms or erroneous sensor data?
YES
NO
Flight control input from user?
FLIGHT
- Monitor sensor data and comms
- Process flight control inputs to servos/motors
Loss of comms or erroneous sensor data?
YES
- Disable motor
- Deploy parachute
NO
SHUTDOWN command from user?
YES
- End program execution

Figure 3: State machine for the embedded control loop

A breakdown of the states is as follows:

- START: Upon receiving power from the batteries and running the control software, the program will begin in the START state. This state is used to do preliminary checks on the hardware components and also open connectivity with the cloud/backend software. First, the drone's onboard sensors, such as the Inertial Measurement Unit (IMU), barometer, and camera, are initialized and verified to be responding. Communication between the drone and the cloud/backend using a cellular connection is then established and checked for stability. Basic write commands to the servos are sent to confirm functionality. If any of these steps should fail, the program transitions to the ERROR state. Otherwise, it will transition to Ready to Fly (RTF).
- RTF: The RTF state serves as an idle state for the program to wait in until the user wants to fly the drone. In this state, the program will begin to transmit the sensor data and camera feed to the user via cellular connectivity. Additionally, the readings from the sensors are checked for validity, ensuring the modules are operating correctly. If any of these steps should fail, or if user inputs cannot be retrieved, the program transitions to the ERROR state. Otherwise, the program checks if the user has inputted a keystroke for flight controls. If an input is available, the program transitions to the FLIGHT state. If not, it will loop back to RTF.
- FLIGHT: The FLIGHT state is the primary state the drone will be in during takeoff, flight, and landing. The program will continue to monitor sensor data to ensure hardware validity and communication stability, and transmit sensor data to the user. Furthermore, command inputs from the user are retrieved and processed to control the servos and motors. If there is a loss of communication between the drone and cloud/backend, or if the sensor data is erroneous, then the drone will disable its motor, deploy a parachute, and transition to the ERROR state. Otherwise, the program checks to see if the user has sent a command to shut down the drone, in which case the program transitions to the SHUTDOWN state and ends its execution. If not, the program loops back to FLIGHT.
- ERROR: The ERROR state is designed to safely, but quickly, end the operation of the drone if there is a software or hardware malfunction. In this state, the program will attempt to notify the user of the error and send diagnostics information. It will then transition to the SHUTDOWN state and end its execution.
- SHUTDOWN: In the SHUTDOWN state, the program will end its execution. This, in turn, will half the operation of all hardware components.

2.1.3 Drone Control Subsystem Requirements

The drone has various control subsystems such as the servo/motor controls and sensor/cellular modules. There are specific scenarios where we believe that the drone should have a fail-safe which is described in the table below using these modules. These fail-safe procedures mainly occur when there are certain connection issues that could possibly occur.

Requirements	Verification
If the drone fails to send a GET Request to the Firebase Storage for requesting Commands, the drone deploys the parachute and stops motor if midair	1. Drone will send Get Request to a non-existent application 2. The Get Request will fail due to not being able to connect to this non-existent application 3. Motor should stop rotating and servo should deploy parachute
If application disconnects, the drone deploys the parachute and stops motor if midair	1. Application disconnects and shuts down by the user 2. Firebase sends message to drone saying the application disconnected 3. Drone stops motor from running and parachute deploys

Table 1: R-V table for the Drone Control Subsystem

2.1.4 Drone Sensing Subsystem Requirements

The Drone Sensing Subsystem encompasses the primary group of sensors on the drone. This includes the MPU-6050 IMU used for obtaining accelerometric and gyroscopic data, BMP280 barometric sensor used for estimating the drone's altitude from pressure readings, and OV7670 image sensor for capturing video frames. The outputs of these sensors will primarily be used to provide the drone operator with critical drone information during flight. Refer to section 2.4.2 for more information on the interaction with these sensors.

Requirements	Verification
MPU-6050 IMU 1. The MPU-6050 must be able to read gyroscope data in 3-axes (X, Y, Z) with an accuracy of 10%. 2. The sensor must provide acceleration data for velocity calculations with a full-scale range of ±2g and accuracy of 10%.	MPU-6050 IMU 1. Ensure the sensor is initialized to a reference point (X, Y, Z = 0, 0, 0). Execute a test program to read data from the IMU while moving the drone to predetermined locations. Compare the IMU readings to real displacement measured with a ruler.

3. The sensor should output angular velocity in 3-axes (X, Y, Z) at a range of ±360°/s with an accuracy of 10%. 4. The sensor must provide data at an output data rate (ODR) of at least 1 kHz for both accelerometer and gyroscope.	2. Ensure the sensor is initialized to a reference point (0 acceleration, 0 velocity). Execute a test program to read data from the IMU while placing a drone in a moving car at a constant speed. Compare the velocity values from the IMU calculations to the speed of the car. 3. Ensure the sensor is initialized to a reference point (angular velocity of X, Y, Z = 0, 0, 0). Execute a test program to read sensor data from the IMU while tilting the drone to a certain predetermined position in a fixed amount of time. Compare the angular velocity values from the IMU to the real values measured with angles from a protractor 4. Execute a test program to record data from the IMU over a set period of 10 seconds. Measure the ODR by dividing the total number of samples by the measurement time.
BMP280 Barometer 1. The BMP280 must measure atmospheric pressure with an accuracy of 10% for altitude readings. 2. The sensor should accurately measure temperature with a resolution of 0.01°C and accuracy of 10%. 3. The BMP280 must provide an ODR of at least 1 Hz.	BMP280 1. Execute a test program to measure the pressure readings from the BMP280 while the drone is at ground level and atop a tall apartment building. Use a reference barometer to measure atmospheric pressure and compare it to the BMP280 output. 2. Place the BMP280 in a temperature-controlled chamber, such as a fridge, then in a room-temperature room. Execute a test program to log temperature readings from the BMP280. Use a calibrated thermometer as a reference and compare the data.

	3. Execute a test program to record data from the BMP280 over a set period of 10 seconds. Measure the ODR by dividing the total number of samples by the measurement time.
OV7670 Image Sensor 1. The OV7670 must support a resolution of 640x480 Video Graphics Array (VGA) pixels and full RGB. 2. The camera should provide a frame rate of at least 6 frames per second (FPS).	OV7670 Image Sensor 1. Configure the OV7670 for VGA resolution. Execute a test program to capture a single image frame of nearby surroundings. Verify the pixel dimensions in the output image using a third-party software, ensure color quality is satisfactory, and image clarity is acceptable. 2. Ensure the camera is properly calibrated to capture steady image frames for video. Execute a test program to capture a video for a fixed amount of time. Count the number of frames received and calculate the FPS.

Table 2: R-V table for the Drone Sensing Subsystem

2.1.5 Network Stack Requirements

The network stack comprises mainly of the SIM7600's ability to transfer and retrieve data. This also includes the OV7670's frame generation and transfer capabilities including the frame rate maintenance. This system is primarily used for communication between the end user and the drone, and will check for any anomalies in this communication to respond correspondingly.

Requirements	Verification
The drone must be able to connect to the database and retrieve data.	1. Ensure the drone is powered on and connected to the internet. Verify the internet connection by pinging a known server and checking for a response. 2. Send a query to the database to retrieve a specific data entry. Record

	the response received from the database and confirm it matches the expected data. 3. Modify the data entry in the database and send another query to retrieve the updated data. Record the response and confirm it reflects the updated data. 4. Disconnect the drone from the internet and attempt to send a query to the database. Confirm that the query fails and an appropriate error message is received.
The drone must maintain a stable and reliable internet connection, and deploy parachute in case network connection is disrupted	1. Test the drone's internet connection in various locations. Measure the signal strength and quality using network diagnostic tools. 2. Monitor the connection stability over an extended period. Record any instances of connection drops or interruptions. 3. Perform a speed test to measure upload and download speeds. Confirm that the speeds meet the required thresholds for the application. 4. To check for the disrupted connection, make a method that is called in loop and checks the connection, and if not detected, trigger the servo for deployment.
The drone must stream video and sensor data in real-time without significant latency	1. Set up the drone to stream video and sensor data to a server. Monitor the video stream for latency or buffering issues. 2. Measure the end-to-end latency of the video stream. Confirm that the latency is within acceptable limits for real-time streaming. 3. Compare the transmitted data with the

	actual sensor readings. Confirm that the data matches the sensor readings within an acceptable margin of error. 4. Test the video stream under different network conditions. Record the performance and identify any issues that arise under varying conditions.

Table 3: R-V table for the Network Stack

2.1.6 Application Level Requirements

The application consists of a frontend and a backend. The backend is what is used to interface with a firebase storage. This is a key detail to ensure connection between the drone and the frontend portion of the application. This is due to the fact that whatever the data the backend retrieves will be processed and sent to the frontend, which is the sensor data. This is the same vice versa where the data the backend sends originated from the frontend which are the drone commands.

Requirements	Verification
The application must be able to retrieve the sensor data to view in the application	1. First view the sensor data through logging in the arduino terminal 2. View whether Firebase retrieves the sensor data or not 3. Check if a byte was read from the database in the C++ backend 4. Ensure the data is viewable within the frontend application
The drone must be able to retrieve commands from the application	1. First type a command key while the frontend application is running 2. View the log message in the C++ backend if 1 byte was read 3. View if the byte was received in the Firebase database 4. Check if the drone receives and clears the command

Table 4: R-V table for the Application Level

2.2 Physical Design

2.2.1 Physical Design Overview

Our design is going to be a V-tail winged drone with a camera mount below the drone. This will require a total of 2 servos for the wings, 2 servos for the V-tail, 2 servos for the camera, 1 servo for parachute, and 1 motor for creating thrust for the drone. In terms of the wing, we plan to create a larger wing area than most drones for easier controllability for users during flight. It is also important to note that we will be utilizing foam board for the entirety of the drone. Foam board is extremely lightweight and replaceable compared to its alternatives such as LW-PLA 3d printing material and carbon fiber. This is mainly due to the fact that foam board is extremely easy to cut with tools like box cutters compared to carbon fiber and plastic. In the next couple sections, we will describe how to control the drone and how to generate lift.

2.2.2 COG vs. COL

A key detail for flying winged drones is the concept of center of gravity vs center of lift. Center of gravity is the balancing point of the total weight of the aircraft, and the center of lift is the center point of the lifting forces acting upon the wing of the aircraft. An image below depicts the 2 centers in play where the orange center represents the center of gravity and the blue center represents center of lift.

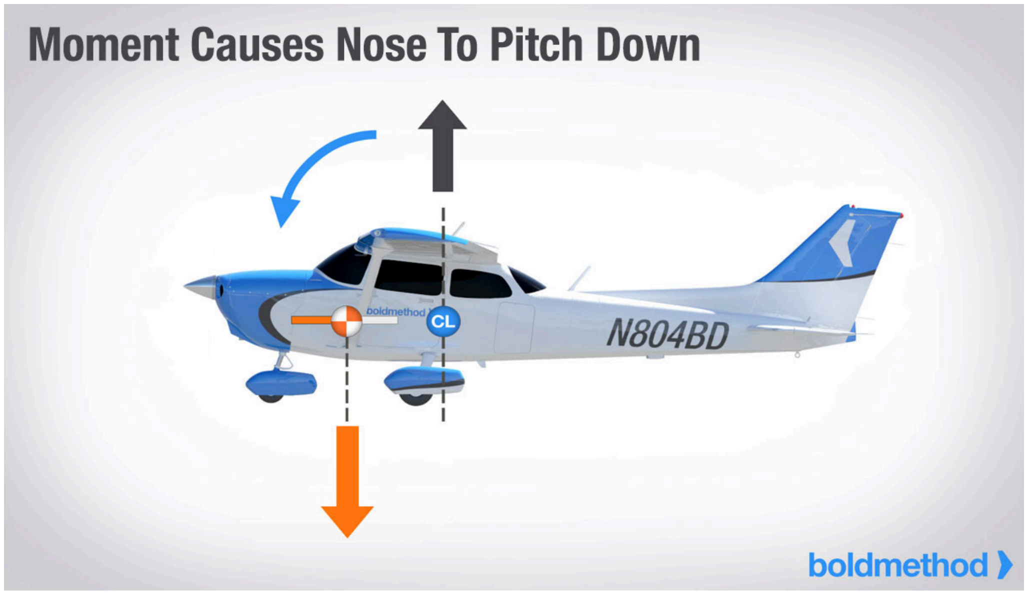


Figure 4: Effects of moments on flight behaviors

In general, for typical aircrafts, one needs to make their center of gravity slightly forward to the center of lift. If the center of gravity is too close to the center of lift, the aircraft may be neutrally or negatively stable. This will make the drone much harder to control. To change the center of gravity, we will modify the placement of our 12V Lipo drone battery pack. This will give us full control of where the center of gravity is located because we cannot control where the center of lift is located without modifying the wing structure overall.

2.2.3 Physical Control

For steering the direction of the drone, we utilize flaps and aileron. As mentioned earlier in excerpt 2.2.1, our drone is a V-tail drone. Therefore, we have a total of 2 flaps and 2 ailerons. Each wing has a single aileron and each tail in the V-tail has a single flap. These control the roll, pitch, and yaw of the drone mid flight due to knockbacks caused by airflow. For example, if the left aileron is angled at -45 degrees and the right aileron is angled at 45 degrees, the plane will roll clockwise. If the left aileron is angled at 45 degrees and the right aileron is angled at -45 degrees, the plane will roll counter clockwise. Similarly, the tails control yaw and pitch. When both flaps are up, the plane will pitch downwards. If both flaps are down, the plane will pitch upwards. If the flaps are opposite of one another, this will alter the yaw in a similar fashion to the ailerons with roll.

In order to control the flaps and aileron of the drone, we utilize servos to control the flaps on the drone. Below shows an image that depicts this idea:

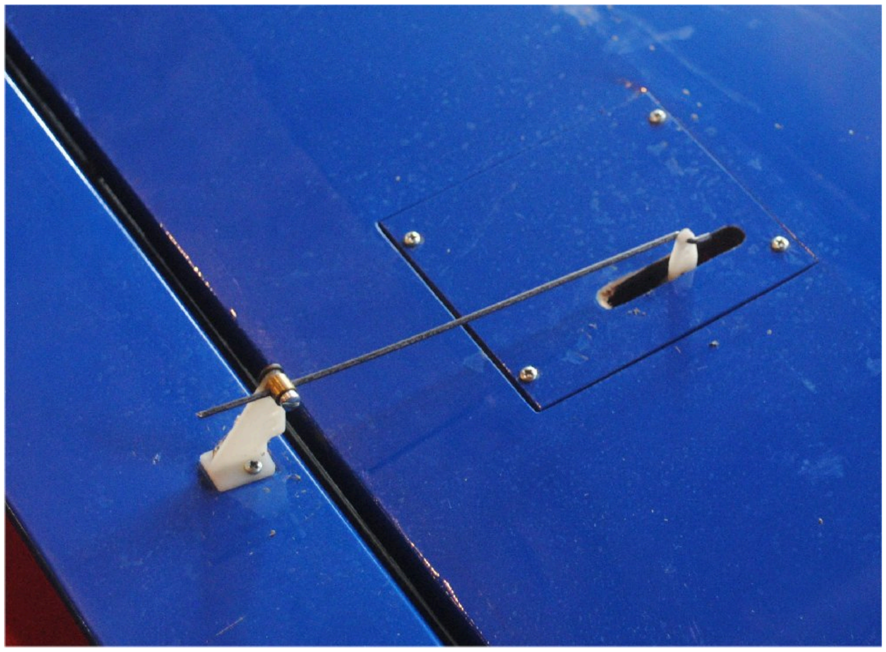


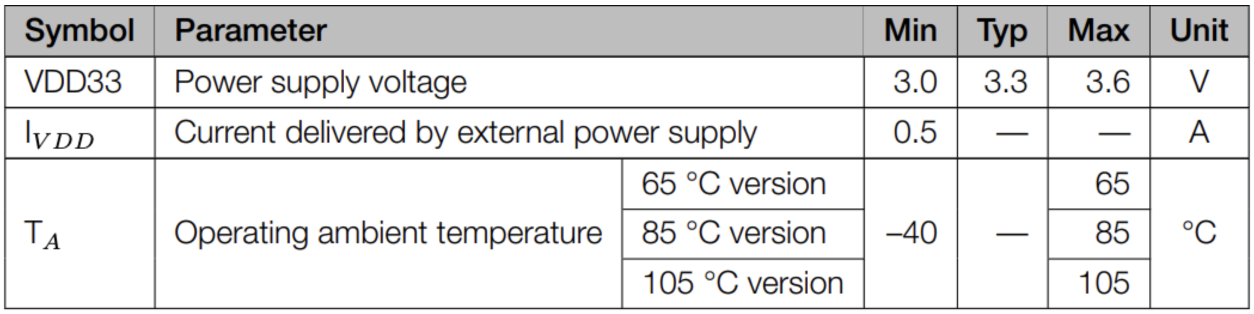
Figure 5: Example implementation of an aileron with a servo

Essentially, we utilize a servo that is connected to a rod, and the rod is connected to a piece connected to the aileron/flap. It is important to mention that the servos will be mounted directly on the wing for controlling the ailerons, and mounted within the fuselage of the drone for the flaps.

2.3 Hardware Analysis

2.3.1 Operating Voltage & Regulation

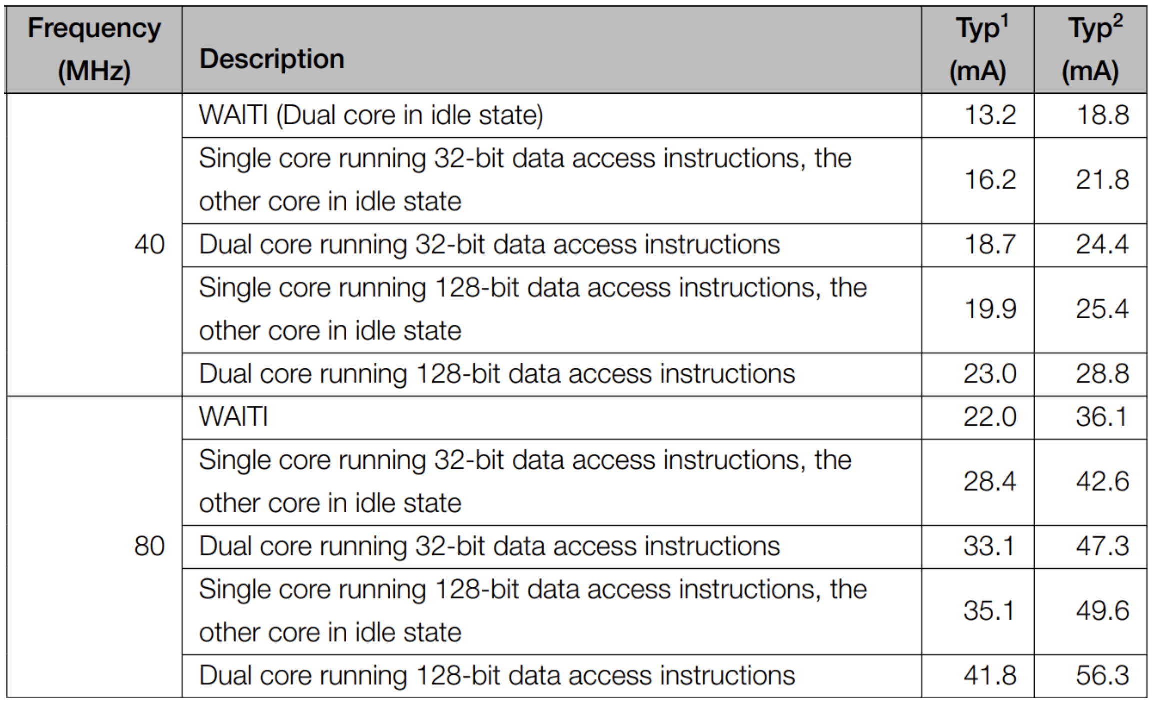
We need to provide specific voltages to the different components of the design. According to the ESP-32 S3 WROOM datasheet, the recommended operating conditions are as follows:



Symbol	Parameter		Min	Typ	Max	Unit
VDD33	Power supply voltage		3.0	3.3	3.6	V
I_{VDD}	Current delivered by external power supply		0.5	—	—	A
T_A	Operating ambient temperature	65 °C version	–40	—	65	°C
		85 °C version			85	
		105 °C version			105	

Figure 6: ESP-32 S3 WROOM power statistics

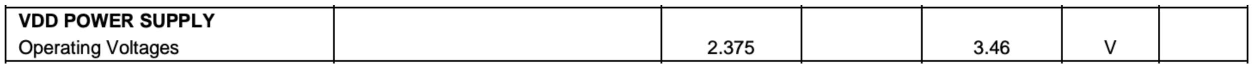
Here, the Power Supply voltage is set to a min of 3V, whereas the max is 3.6V. By default, the Serial Peripheral Interface (SPI) flash on the module operates at a maximum clock frequency of 80 MHz and does not support the auto suspend feature. Now, the current requirements for the maximum frequency we can attain with the current configuration is as follows:



Frequency (MHz)	Description	Typ[1] (mA)	Typ[2] (mA)
40	WAITI (Dual core in idle state)	13.2	18.8
	Single core running 32-bit data access instructions, the other core in idle state	16.2	21.8
	Dual core running 32-bit data access instructions	18.7	24.4
	Single core running 128-bit data access instructions, the other core in idle state	19.9	25.4
	Dual core running 128-bit data access instructions	23.0	28.8
80	WAITI	22.0	36.1
	Single core running 32-bit data access instructions, the other core in idle state	28.4	42.6
	Dual core running 32-bit data access instructions	33.1	47.3
	Single core running 128-bit data access instructions, the other core in idle state	35.1	49.6
	Dual core running 128-bit data access instructions	41.8	56.3

Figure 7: ESP-32 S3 WROOM operating frequency and current statistics

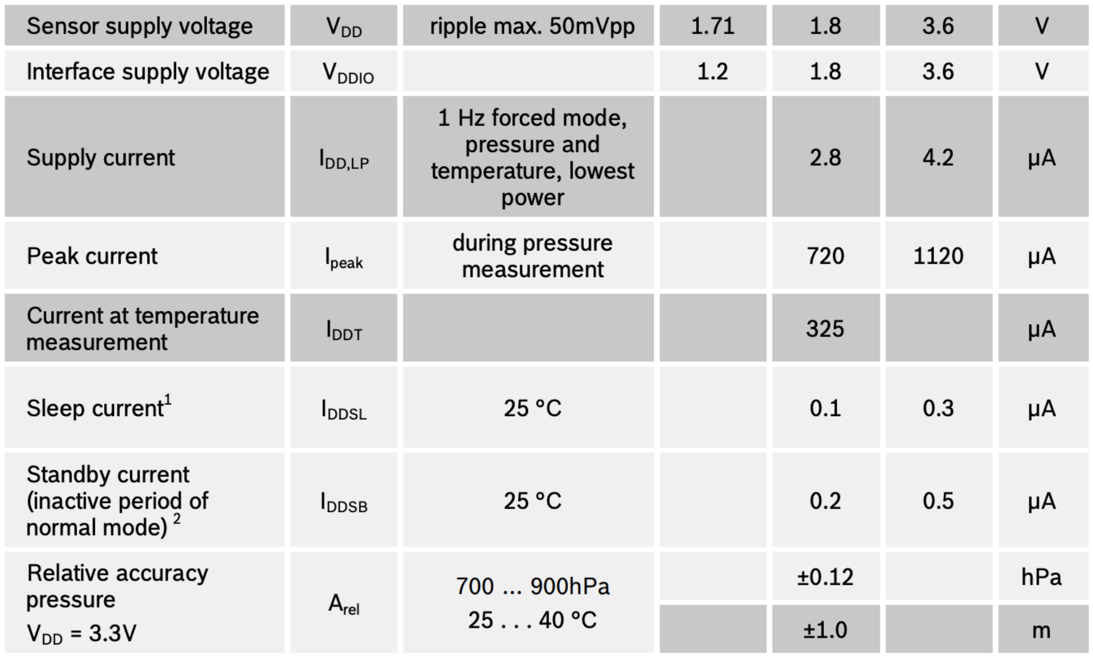
Now, we will be using a 3.3V for the microcontroller because it is a standard voltage level for many components and peripherals, making it easier to interface with the peripherals. The 3.3V is downgraded from a 5V supply (Output of the Electronic Speed Controller (ESC)) through a voltage regulator. Next, the MPU6050 has specific power requirements that are as such according to the datasheet:



VDD POWER SUPPLY Operating Voltages		2.375		3.46	V	

Figure 8: Operating voltage of the MPU-6050

The operating voltage we will be using for this sensor is 3.3, which is derived from the same input channel to the ESP. The next component is BMP280, which is utilized for pressure sensing accepts the following voltages:



Sensor supply voltage	V_{DD}	ripple max. 50mVpp	1.71	1.8	3.6	V
Interface supply voltage	V_{DDIO}		1.2	1.8	3.6	V
Supply current	$I_{DD,LP}$	1 Hz forced mode, pressure and temperature, lowest power		2.8	4.2	µA
Peak current	I_{peak}	during pressure measurement		720	1120	µA
Current at temperature measurement	I_{DDT}			325		µA
Sleep current[1]	I_{DDSL}	25 °C		0.1	0.3	µA
Standby current (inactive period of normal mode) [2]	I_{DDSB}	25 °C		0.2	0.5	µA
Relative accuracy pressure V_{DD} = 3.3V	A_{rel}	700 ... 900hPa 25 . . . 40 °C		±0.12		hPa
				±1.0		m

Figure 9: Power statistics for the BMP280

Here, we are once again using 3.3V for the input to this sensor due to the nature of reusability of the voltage regulator. Next, the ESC requires a 12V supply to power the motor and control the speed of the drone, which then translates into the 5V used by the regulator. The OV7670 camera module is utilized to capture VGA frames to send through the SIM7600. The camera requires 3.3V as the operating voltage, which can be supplied through the regulator. Now there are multiple iterations of this camera module, where some require 3V whereas others require 3.3V. Servos can generally work with a huge range of voltages, and for our specific use case, we will be using 5V servos that can directly draw power from the ESC. Lastly, the SIM7600A requires the standard 3.3V power supply, which can be derived from the regulator.

2.3.1 Drone Power Subsystem

For our entire subsystem, we will be utilizing a 12V lipo battery. However, various components only support up to 3.3V and 5V as mentioned above in the excerpt 2.3.1. Therefore, we need methods to drop the voltage down to these values such that we can power the rest of the system. In order to do so, we first step the voltage from 12V to 5V and 5V to 3.3V. In terms of the 12V to 5V conversion, we will be utilizing an ESC to produce this voltage. A 12V battery will connect from one end, and we can produce a 5 volt output. This is neat because we will also require the ESC for controlling brushless motor speeds. This 5V supply generated by the ESC is also good enough to power our pcb board. Given the 5V supply, we can also step it down to 3.3V using a voltage regulator. The voltage regulator creates and maintains a constant voltage.

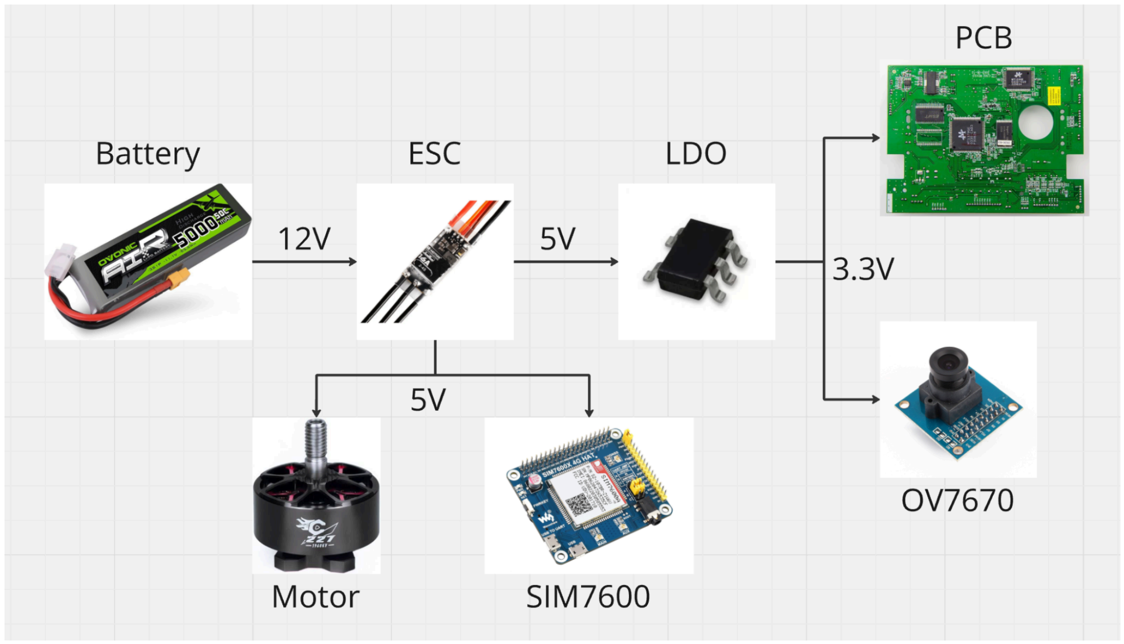


Figure 10: Flowchart of drone power subsystem

2.4 Software Analysis

2.4.1 SIM7600 Cellular Network Communication

The SIM7600 module is a versatile and powerful cellular module that supports 4G LTE connectivity, making it an ideal choice for applications requiring reliable and high-speed internet access. This module is capable of achieving upload speeds of up to 5 Mbps, which is sufficient for streaming video and transmitting sensor data in real-time. The SIM7600 supports various communication protocols, including HTTP, which allows it to send data to a server over the internet. It is commonly used in IoT devices, drones, and other applications where stable and fast internet connectivity is crucial.

To integrate the SIM7600 module with the ESP32 S3 WROOM, one must first establish a connection between the two devices. This involves connecting the appropriate pins on the SIM7600 to the corresponding pins on the ESP32. The SIM7600 module typically communicates with the microcontroller via UART, so the TX and RX pins of the SIM7600 should be connected to the RX and TX pins of the ESP32, respectively. Additionally, the module requires a stable power supply, which can be provided through the VCC and GND pins. Once the hardware connections are established, the next step is to configure the SIM7600 module to connect to the internet and transmit data using the Hypertext Transfer Protocol (HTTP) protocol. This is achieved by sending a series of AT commands to the module. The AT commands are used to configure the network settings, establish a connection to the cellular network, and send HTTP requests. Here is a general outline of the AT commands used in this process:

1. Initialize the module: `AT`
2. Set the APN (Access Point Name): `AT+CGDCONT=1,"IP","tmobile_apn"`
3. Enable the network registration: `AT+CREG?`
4. Check the signal quality: `AT+CSQ`
5. Activate the PDP context: `AT+CGACT=1,1`
6. Start the HTTP service: `AT+HTTPINIT`
7. Set the HTTP parameters: `AT+HTTPPARA="URL","http://your_server_address"`
8. Send the HTTP POST request: `AT+HTTPACTION=1`
9. Read the HTTP response: `AT+HTTPREAD`
10. Terminate the HTTP service: `AT+HTTPTERM`

Alternatively, the HTTP params can be set as such for the firebase database:

```
sendATCommand(AT+HTTPPARA=\"URL\",\"https://your-database.firebaseio.com/path/to/data.json\"");
sendATCommand("AT+HTTPPARA=\"CONTENT\",\"application/json\"");

String jsonData = "{\"name\":\"value\"}";
sendATCommand("AT+HTTPDATA=" + String(jsonData.length()) + ",10000");
sendATCommand(jsonData);
```

In the context of the drone project, the SIM7600 module is used to transmit camera and sensor data to a server. The camera captures video at a resolution of 640x480 pixels with a frame rate of 5 frames per second. The data is compressed using Joint Photographic Experts Group (JPEG) compression, resulting in a manageable data rate that is well within the upload capacity of the SIM7600 module. Additionally, sensor data is transmitted alongside the video stream, contributing a small amount to the overall data rate.

The total data rate for the camera and sensor data, including protocol overhead, is calculated to be 3.87 Mbps. This is comfortably within the 5 Mbps upload limit of the SIM7600 on an LTE network, ensuring that the system can operate without hitting bandwidth limitations. This setup allows the drone to stream video and transmit sensor data in real-time. The last thing we add to this is the Global Positioning System (GPS) location retrieval, which is retrievable to the backend through a POST:

1. Enable GPS: `AT+CGPS=1,1`
2. Check GPS status: `AT+CGPSSTATUS?`
3. Retrieve GPS location: `AT+CGPSINFO`

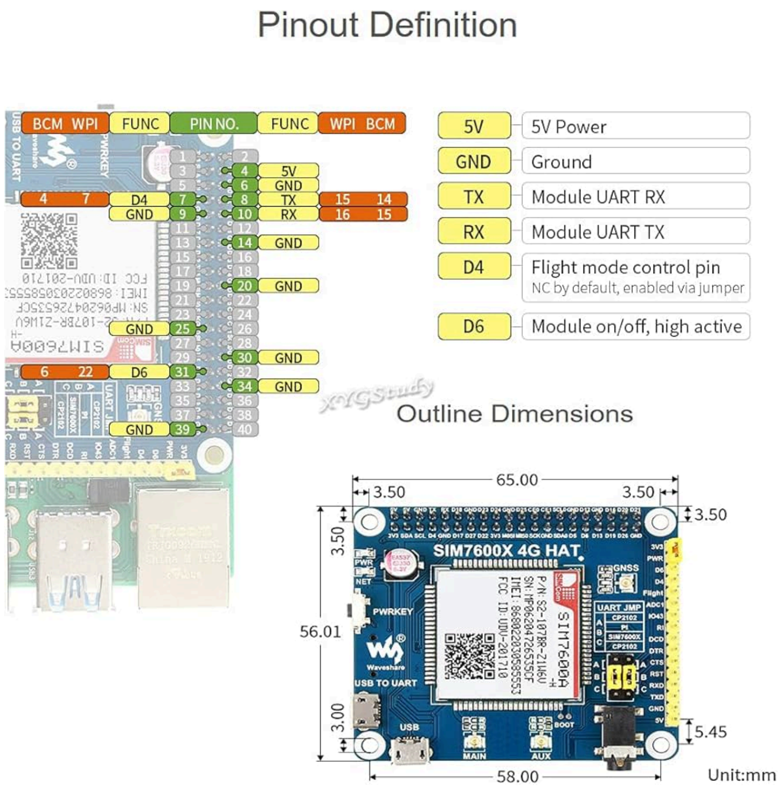


Figure 11. Example pinout of SIM7600A-H

2.4.2 Sensors Processing

<u>MPU-6050 IMU</u>:
The MPU-6050 is an extremely popular IMU used in embedded applications. It is capable of tracking up to six degrees of movement through a 3-axis gyroscope, measuring 'X,' Y,' and 'Z' positions, and a 3-axis accelerometer, measuring 'X,' Y,' and 'Z' angular velocities. The MPU-6050's low cost, small form factor, and easy-to-use interface makes it a widely used choice in drones, robotics, and other mobile devices.

Integrating the MPU-6050 with the ESP32-S3 WROOM-1 microcontroller is quite straightforward, as there are Arduino-compatible libraries for directly interfacing with the module. For communication, the IMU uses an Inter-Integrated Circuit (I2C) protocol to transmit and receive data from the microcontroller. The pseudocode for basic MPU-6050 drivers are as follows:

```
1. Include the MPU6050 library header file
2. Initialize I2C communication between the MPU-6050 and the ESP32 (SCL and SCA pins)
3. Initialize the MPU6050 and ensure its memory address range is allocated
```

4. Call a single function to real data from the gyroscope and accelerometer. The six return values are stored as 16-bit integers
5. Convert the raw sensor data into readable values with correct units
6. Add a small time delay and loop from step 4

For the purposes of our drone, the six points of data returned from the IMU will be useful in providing the operator with flight critical information. From the accelerometer data, we can obtain tri-directional linear velocities that will convey the direction the drone is flying. The gyroscopic data can be converted into roll, pitch, and yaw measurements for increased positional awareness.

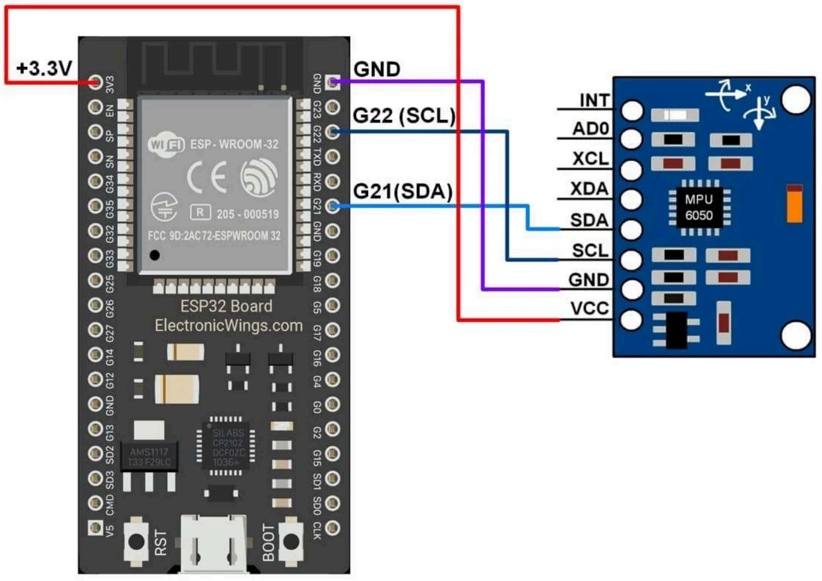


Figure 12: Example pinout of MPU-6050 with ESP32 Devkit

<u>BMP280 Barometer</u>:
The BMP280 is a high-precision and extremely power efficient barometric pressure sensor commonly used in various devices such as drones, GPS systems, and outdoor equipment. The module is able to supply the user with barometric pressure and temperature sensor data. Since it is designed to supply very accurate sensor readings, its usage in the Early Response Drone will largely be for estimating altitude from pressure.

Similar to the MPU-6050, the BMP280's wide usage amongst embedded developers means that there are numerous code libraries available for interacting with the module. Furthermore, the BMP280 communicates with the ESP32 microcontroller via an I2C protocol for data transmission. The pseudocode for interfacing with the BMP280 is found below:

1. Include the BMP280 header file

2. Initialize I2C communication between the BMP280 and the ESP32 (SCL and SCA pins)
3. Initialize the BMP280 and ensure its memory address range is allocated
4. Set sensor sampling rate and filter coefficient
5. Call function to retrieve temperature data in Celsius and pressure data in Pascals.
6. Use pressure reading and call function to estimate altitude from sea level
7. Add a small time delay and loop from step 5

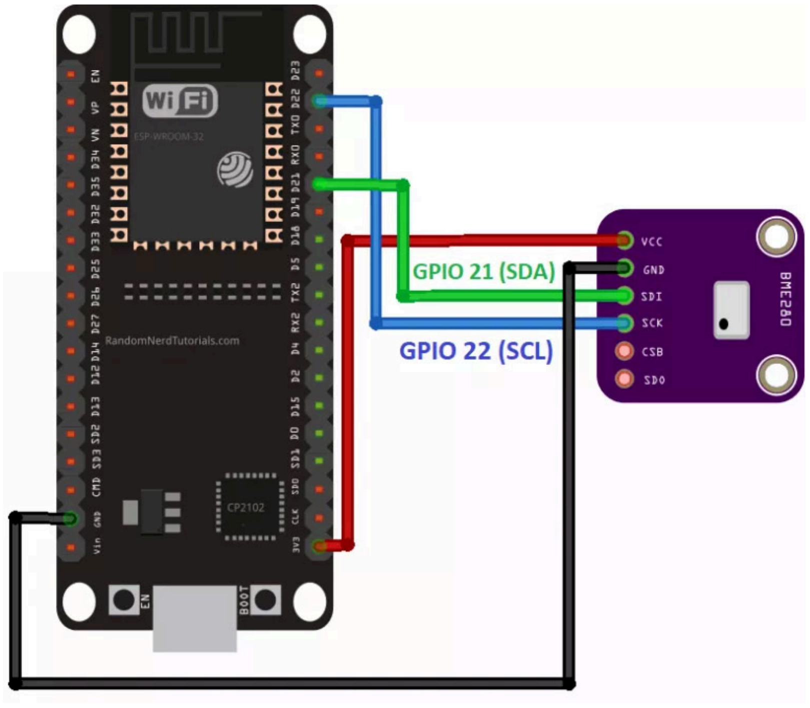


Figure 13: Example pinout of BME280 with ESP32 Devkit

<u>OV7670:</u>
The OV7670 is a low voltage Complementary Metal-Oxide-Semiconductor (CMOS) image sensor that provides the full functionality of a single-chip VGA camera and image processor in a small footprint package. It provides full-frame, sub-sampled or windowed 8-bit images in a wide range of formats. All required image processing functions, including exposure control, gamma, white balance, color saturation, hue control and more, are also programmable. The additional step we took is compression of the JPEG through the JPEG encoder.

```
Initialize camera with SIOC_PIN, SIOD_PIN, VSYNC_PIN, HREF_PIN,
PCLK_PIN
Initialize JPEG encoder with quality level 75

Setup:
  Begin Serial communication
  Begin I2C communication with SIOC_PIN, SIOD_PIN
  Initialize camera
  Set camera resolution to VGA
  Set camera color space to RGB

Loop:
  Capture image from camera
  If image capture successful:
    Compress image using JPEG encoder
    If compression successful:
      Print "Image compressed successfully"
      // Add code to transmit or store the compressed image
    Else:
      Print "Failed to compress image"
  Else:
    Print "Failed to capture image"
  Delay 1 second
```

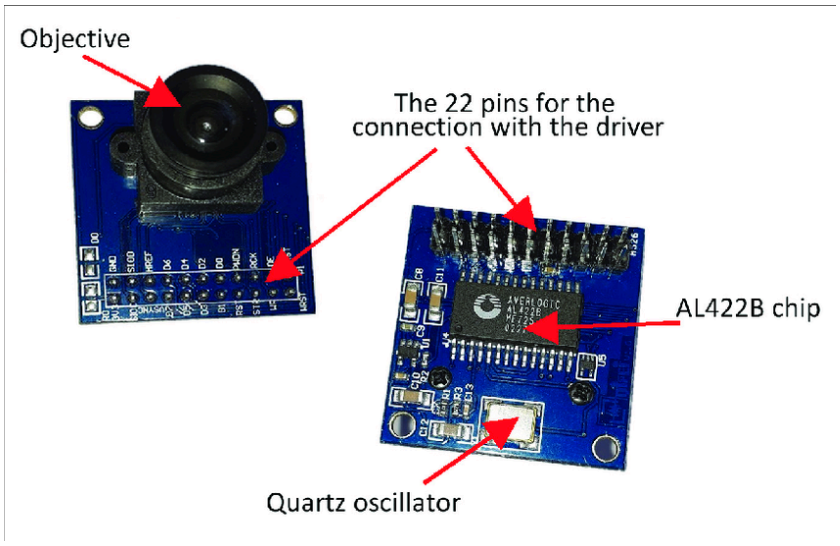


Figure 14: Camera module configuration

2.4.3 Motion Control

Servos play a key role in the flight control system of the drone, providing precise control and stable flight performance for the drone. The drone's roll, pitch, and yaw attitudes are all controlled and adjusted by the servo. Encoder and feedback mechanism integrated in for accurate control of position and angle. The servo's quick reaction time enables the drone to fly steadily and turn precisely. The steps to define and use servos are as follows:

1. Include the Library: `#include <ESP32Servo.h>`
2. Initialize Servos:

```
Servo servo1;
Servo servo2;

void setup() {
    servo1.attach(18); // Attach servo1 to GPIO 18
    servo2.attach(19); // Attach servo2 to GPIO 19
}
```

3. Control Logic:

```
void setServoAngle(Servo &servo, int angle) {
    servo.write(angle); // Set servo to the specified angle
}

void loop() {
    // Example: Set servos to specific angles
    setServoAngle(servo1, 90); // Set servo1 to 90 degrees
    setServoAngle(servo2, 45); // Set servo2 to 45 degrees
    delay(1000); // Wait for 1 second
}
```

Now, for our use case, the easiest way to control all the servos is to retrieve the HTTP data sent by the server, and change the angles to achieve desired flight control (for example a right turn command would indicate a certain configuration of wing pattern whereas a left would require the exact opposite).

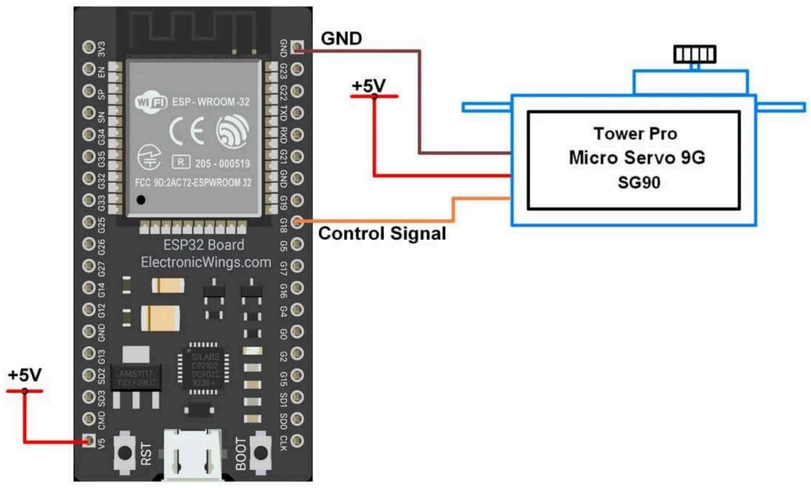
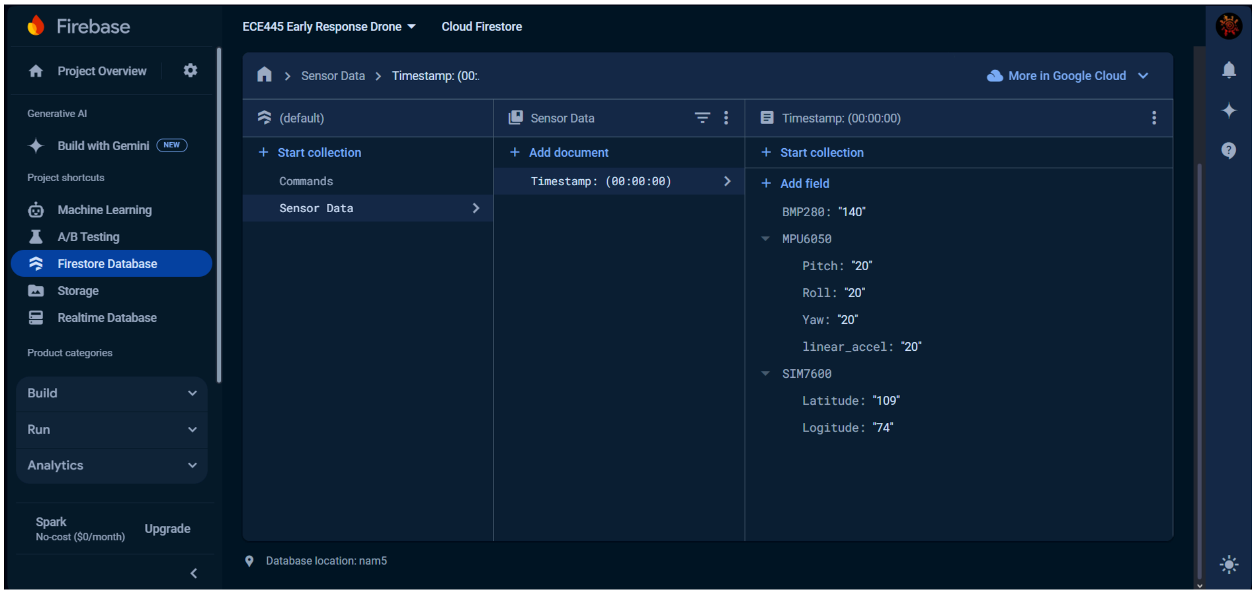


Figure 15. Example pinout of servo with ESP32 Devkit

2.4.4 Firebase Cloud Storage

Before diving into the usage of Firebase Cloud Storage, we will go into exactly what Firebase is. Firebase is a cloud computing service bought by Google which contains various toolchains such as Storage, specifically for files and images, Realtime Database for storing and syncing realtime, Firestore which not only allows for real time updates but also automatic scaling of database and powerful queries library to perform the updates. For our purposes, we will utilize Firestore and the regular Storage. As slightly mentioned within the SIM7600 and Application Level excerpt, we can modify our storage using the 4 types of HTTP requests: GET, POST, PUT, and DELETE.

Below shows an example of how Firestore looks internally. As we can see, we have a 3 column setting where we can either start a collection, add a document, and add a field.



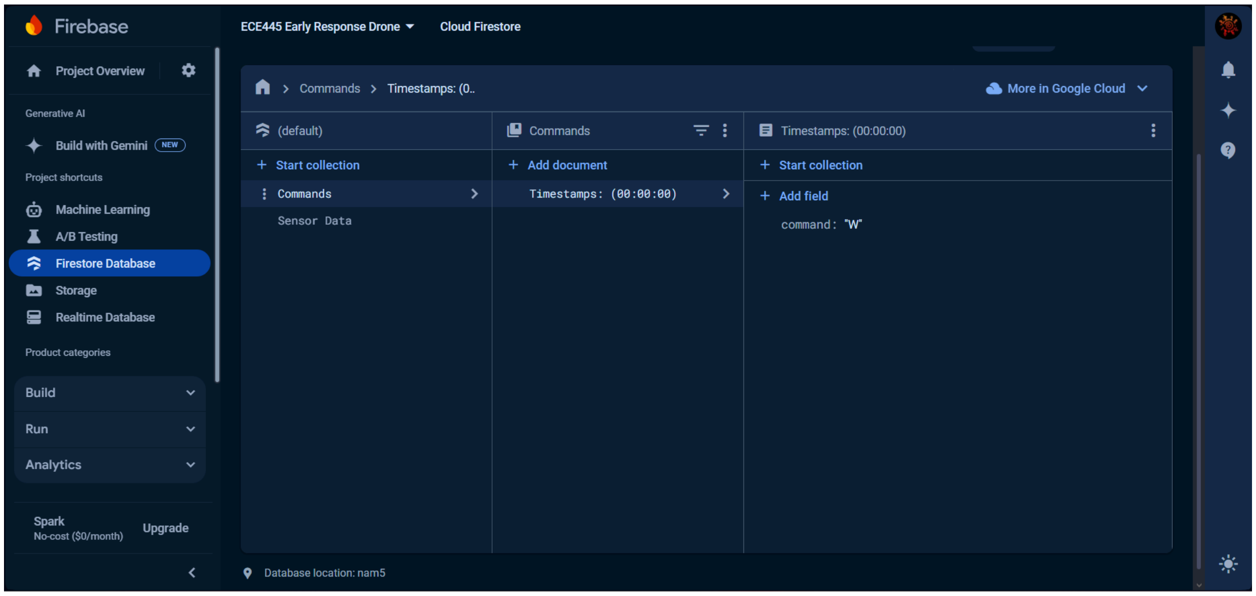
```
Commands
Sensor Data

Timestamp: (00:00:00)

BMP280: "140"
MPU6050
    Pitch: "20"
    Roll: "20"
    Yaw: "20"
    linear_accel: "20"
SIM7600
    Latitude: "109"
    Logitude: "74"
```

Figure 16: Firestore sensor data collection

For our purposes, we will have 2 collections: Sensor Data and Commands. Sensor Data will receive a continuous stream of "documents" from the drone which will be stored on Firestore. Each document will contain a JavaScript Object Notation (JSON)-like data which is a pressure measurement from the BMP280, Roll, Pitch, Yaw, and Linear Acceleration measurement from the MPU6050, and finally GPS location from SIM7600. This data will then be retrieved and cleared by an application. We will also be storing commands received from the application by Firebase which is shown in the example below:



```
Commands
Sensor Data

Commands
Timestamps: (00:00:00)

Timestamps: (00:00:00)
command: "W"
```

Figure 17: Firestore commands for data collection

Similar to the Sensor Data collection, the Command collection will be retrieved and deleted by the drone for processing. The main issue with Firestore is that storing images as Base-64 is compute intensive. This is a problem when sending frames from the drone to store onto Firebase. However, this is where storage comes in which is another service provided by Firebase.

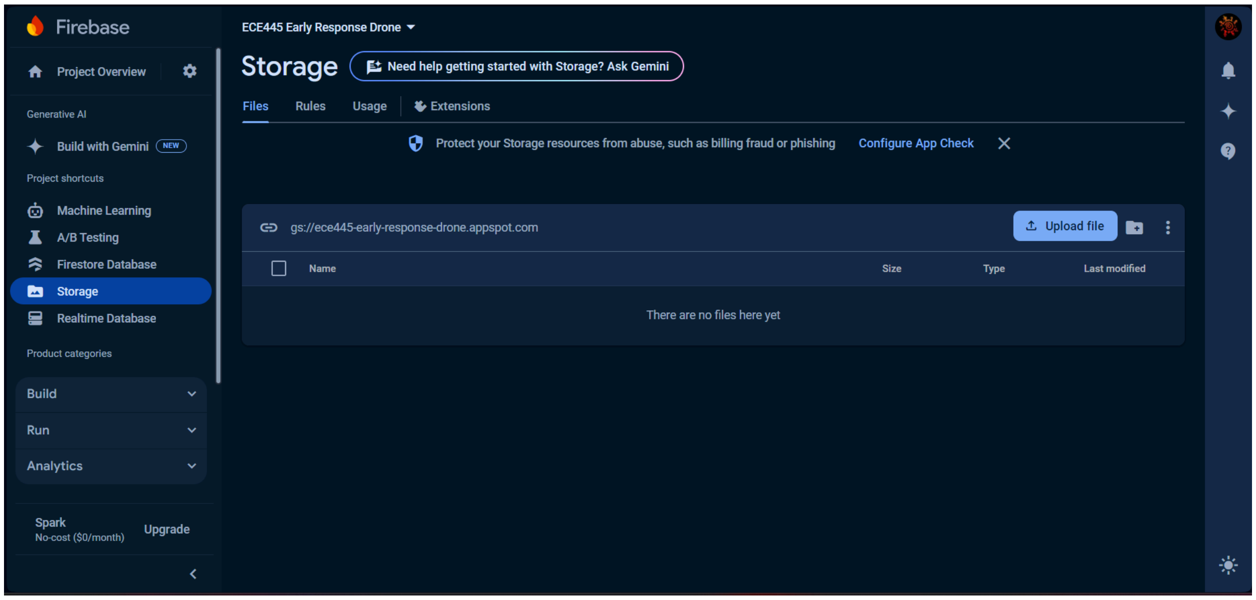


Figure 18: Firebase storage for frames

Storage, as shown in the image above, is specifically used for seamlessly storing files and providing an identification for these files as URLs. We can easily collect these frames one at a time for displaying on our application by also performing HTTP requests.

Note: We mentioned how the application explicitly connects with the Firebase, however in the next section, we will discuss what the application exactly is.

Note: Firebase is a medium that we are currently trying to remove. If we are able to remove Firebase, we will directly send the document data to the Application from the Drone without Firebase. This is not mentioned in the 2.4.5 Application Level.

2.4.5 Application Level

As mentioned in the previous section, we discussed that the application will retrieve Sensor Data and send Commands. The application comprises two subparts: Boost C++ Backend and TypeScript Frontend.

Boost C++ Backend:
Boost is an industry standard networking library that supports multiple protocols such as HTTP, TCP, and Websockets. We will be utilizing HTTP Requests to retrieve data from the Firestore and Storage. Retrieving the frames and the sensor data will open multiple doors to ML processing. For example, we can run object tracking algorithms on the frames we receive. Afterwards, we will send the data, unprocessed or processed, to our TypeScript Frontend.

TypeScript Frontend:
The TypeScript frontend acts as a command control for the drone. Essentially, the frontend will provide all of the necessary information/visuals to give an understanding of what the drone is actually doing to the pilot. For example, we will have unit-circle-like charts to display roll, pitch, and yaw, and a visual for the altitude reading. We will also provide a real-time video feed on the command control as well as a google map to show location of the drone.

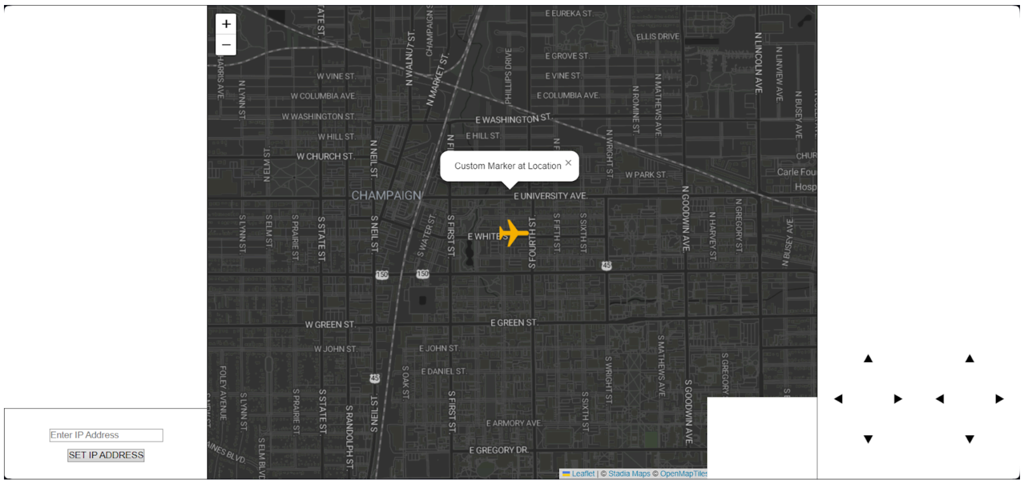


Figure 19: Current frontend application UI

2.5 Tolerance Analysis

Looking at our entire design, we anticipate that the drone's cellular capabilities will be the most difficult to implement and pose the greatest risk to our plan. The cellular module we plan to use, the SIM7600, theoretically should support 4G connectivity and up to 5 Mbps upload speeds. This would be enough to stream video at a frame rate higher than we expect to achieve; however, there is not much information available on the usage of this chip in a drone setting.

<u>Camera Data Calculation</u>:
Resolution: 640x480 pixels
Color Depth: 24 bits per pixel (3 bytes per pixel for RGB)
Frame Rate: 5 FPS (as specified)

<u>Raw Frame Size</u>:
Pixels per frame = 640×480 = 307,200 pixels
Raw size per frame = 307,200 × 3 (bytes per pixel) = 921,600 bytes
Using the 10:1 compression ratio (JPEG Compression (Lossy)), the compressed frame size is:
921,600 / 10 = 92,160 bytes per frame

At 5 frames per second, the camera data rate will be:
Data rate for camera = 92,160 bytes per frame×5 frames per second = 460,800 bytes per second = 460.8kB/s

Data rate for sensors = 50 bytes per second

The total data rate for the camera and sensor data is:

460,800 bytes per second+50 bytes per second = 460,850 bytes per second

The protocol overhead is:
Overhead = 460,850×0.05 = 23,042.5 bytes per second ≈ 23,043 bytes per second
So, the total data rate is 0.483893MB/s × 8 = 3.87Mbps

The SIM7600 module supports LTE upload speeds of up to 5 Mbps. With the total data rate calculated as 3.87 Mbps, we are well within the limits of the SIM7600's upload capacity on a 4G LTE network.

This is within the 5 Mbps upload limit of the SIM7600 on an Long Term Evolution (LTE) network, so the system should work under these conditions without hitting bandwidth limitations.

3 Testing and Demonstration

For the purposes of testing our project, we will spend a sizable amount of time unit testing each of the individual components involved in the drone’s physical hardware, as well as electrical and software subsystems. To the best of our ability, we will ensure reliability, stability, safety, and robustness of our drone before flying it in the air. When we are confident that our complete drone system is ready for in-air testing, we will first need to register our drone with the Federal Aviation Administration, since we anticipate our drone will weigh more than the unlicensed limit of 0.55 lbs. This process is straightforward, but requires an additional $5 registration fee.

Upon registering our drone, we will then be able to fly and test our drone in-air. However, many high-traffic outdoor areas on campus, such as quads, do not allow the usage of drones. We have identified some local fields that should serve as ideal locations for testing our drone, since there is little to no civilian presence; however, we must first confirm with the university or local government that these spaces are free for drone use. For the Final Demonstration, we will be able to show the drone’s bespoke software suite and basic electronics indoors, without flying the drone. We will then either supply a pre-recorded video of the drone’s in-flight performance, characteristics, and features, or perform an in-person exhibition of our drone flying in the air if a suitable location is close enough.

4 Cost and Schedule

4.1 Cost Analysis

PCB Part	Link	Amount	Cost per Part	Total Cost
2.2nF capacitor	https://item.szlcsc.com/15869.html	5	\$0.001	\$0.005
10uF capacitor	https://item.szlcsc.com/362304.html	8	\$0.017	\$0.136
100nF capacitor	https://item.szlcsc.com/362304.html	1	\$0.002	\$0.002
10nF capacitor	https://item.szlcsc.com/362304.html	1	\$0.002	\$0.002
LESD5D5.0CT1G	https://lcsc.com/product-detail/TVS_ESD5Z5V0C_C129211.html	3	\$0.02	\$0.06
1N5819HW-7-F	https://item.szlcsc.com/13409.html	1	\$0.028	\$0.028
CON22X1		2	\$0	\$0
XL-1608SURC-06	https://item.szlcsc.com/221679.html	1	\$0.003	\$0.003
L8050QLT1G	https://www.diodes.com/assets/Package-Files/SOT23.pdf	2	\$0.012	\$0.024
BSS138_C713688	https://www.diodes.com/assets/Package-Files/SOT23.pdf	2	\$0.023	\$0.046
10kΩ resistor	https://www.mouser.in/datasheet/2/447/PYu_RT_1_to_0_01_RoHS_L_11-1669912.pdf	7	\$0.001	\$0.007
1kΩ resistor	https://www.mouser.in/datasheet/2/447/	1	\$0.001	\$0.001

	PYu_RT_1_to_0_01_RoHS_L_11-1669912.pdf			
4.7kΩ resistor	https://www.mouser.in/datasheet/2/447/PYu_RT_1_to_0_01_RoHS_L_11-1669912.pdf	1	$0.001	$0.001
5.1kΩ resistor	https://item.szlcsc.com/323315.html	1	$0.001	$0.001
47.5kΩ resistor	https://item.szlcsc.com/323315.html	1	$0.001	$0.001
22.1kΩ resistor	https://item.szlcsc.com/323315.html	2	$0.001	$0.002
PTS645SH50SMTR92LFS	https://item.szlcsc.com/279067.html	2	$0.123	$0.246
SK6812MINI-HS	https://img.jlc.com/pdf/applyPasteComponent/2021-11-17/554769A/a77f6abd580f4801839c6cef98018a08/SK6812MINI-HS.pdf	1	$0.103	$0.103
SGM2212-3.3XKC3G/TR_C3294699	https://item.szlcsc.com/410724.html	1	$0.661	$0.661
CP2102N-A02-GQFN28	https://item.szlcsc.com/245064.html	1	$2.464	$2.464
BMP280	https://atta.szlcsc.com/upload/public/pdf/source/20200702/C83291_F39E84AB7DFC569A4C7554F8659640A1.pdf	1	$3.141	$3.141
MPU6050	https://www.lcsc.com/product-detail/Attitude-Sensors_TDK-InvenSense-MPU-6050_C24112.html	1	$7.28	$7.28

U-F-M5DS-W-3	https://item.szlcsc.com/157120.html	1	$0.087	$0.087
ESP32-S3-WROOM-1-N4R8	https://www.mouser.cn/datasheet/2/891/Espressif_ESP32_S3_WROOM_1_1U_Preliminary-2530171.pdf	1	$4.24	$4.24

Table 5: Bill of materials for components on the PCB

External Parts	Link	Amount	Cost per Part	Total Cost
SIM7600	https://www.aliexpress.us/item/3256806596464964.html?spm=a2g0o.productlist.main.59.44f0spbWspbW99&algo_pvid=11478075-80d6-459c-96f0-061d85809414&algo_exp_id=11478075-80d6-459c-96f0-061d85809414-29&pdp_npi=4%40dis%21USD%2163.00%2131.50%21%21%2163.00%2131.50%21%402101f93417280069407757053e8ead%2112000038282768379%21sea%21US%210%21ABX&curPageLogUid=r1tX8CwKKEGJ&utparam-url=scene%3Asearch%7Cquery_from%3A	1	$31.50	$31.50
Servo 4 pack	https://www.amazon.com/Micro-Servos-Helicopter-Airplane-Controls/dp/B07MLR1498/ref=sr_1_6?crid=PUEGKW6FVXXS&dib=eyJ2IjoiMSJ9.POZxW8ict	2	$7.99	$16

	f28-1c0EFTfUjj_MoyYfzzsuC5MMwjT6rbZ6G91lio_kdzjsWDqFwzZbVMuHW53QXTXvd5kfqbGZ_4d_9icku_pdEr98YMVePwnvhxa52B90b00LyTQ8TnL-DZklUO3KZNFFlB7HpCQR8_ca0jramEaUlm04ka3fsdFBRvF3x9cGyVNL1z3-DReahCVnicuvSO0enQeTQE23sQLzZxcF60amqaSQhCuLd5Yd-mbc3bFI3DGxrgHIPl17nmkRWNLWt1hY3yUiX9Mt_HGD_nZ_mDTluwG90UBf28.8t6nO3l5BkW67_XOUMF2VkWKubaXnJqess4rTP0Tey4&dib_tag=se&keywords=servos&qid=1728005772&sprefix=servo%2Caps%2C229&sr=8-6			
Brushless Motor/ESC Combo	https://www.amazon.com/abcGoodefg-Brushless-Propeller-Quadcopter-Helicopter/dp/B08DY19WF1/ref=pd_bxgy_d_sccl_2/145-9217241-3264016?pd_rd_w=iN3rT&content-id=amzn1.sym.3858a394-39a9-4946-90e6-86a3153d2546&pf_rd_p=3858a394-39a9-4946-90e6-86a3153d2546&pf_rd_r=BSCCQM55J3P06JD9JV76&pd_rd_wg=EgFSP&pd_rd_r=4d7fbff9-802b-43c1	1	$22.99	$22.99

	-9fb4-221ec2b34919&pd_rd_i=B08DY19WF1&psc=1			
OV7670	https://www.amazon.com/HiLetgo-OV7670-640x480-0-3Mega-Arduino/dp/B07S66Y3ZQ/ref=sr_1_1?crid=Y2KEXEYXO2R6&dib=eyJ2IjoiMSJ9.3d2Qqhhwm1oAQVdz5II2gBVP-XYfhrvZEC677kD3bbwEFy2YONZiRuwTUQjFNbl08UPuqGT5L5I6VsulczqWKGTnCUUwYj1o-3UVTVpYu_oEgGP37GiyEUx2_h5676mzxe0BoTvO4Es1pPm6jFQgEC4cJi1zVX9m_FhVsdx5mhvEApf25GQhW_-ph-m4RSPsyeb3d2ex9CuOV5v18oT8hBLQkfGqzL_ewcm856F55qU_2kFLugIlDzabTiDwMd60f6e0TKjxniAFFMLfSBwO8FnvJ_bqL6xdGy-YurQAuZk.PsfSB2VMsI9XMoLS-nyL2aNId5GToJBwaCgsDUwRiL4&dib_tag=se&keywords=OV7670&qid=1728006821&s=electronics&sprefix=ov7670%2Celectronics%2C95&sr=1-1#customerReviews	1	$8.99	$8.99
12V Lipo battery	https://www.amazon.com/Battery-Dean-Style-Connector-Parkzone-Airplane/dp/	1	$16.99	$16.99

	B077P73SDS/ref=sr_1_2?crid=2IJUMKOXHLQG6&dib=eyJ2IjoiMSJ9.WGM943GUBs6WFEqw1HOii7CS4E28O1YJSkVPXgPEn8fm5xPVVyad8bka-KOO1cMxqOvRRewsEiQDMceNtbGWaTfTedtHZ8G1zDv4CB7i-9U2VVVFRoSLHRTxaPwd1AtHH1faD_sLi7bN2EWwQ84X9Xqa7XuQ1wGLOngwwxJTVw4Rd6daXr94Vxofo7t2y8i0W96EmwW8MEYQ4guIuLLOmRrFr212QEVZmtmVOJQYuz-gMmzO_Yudu88bFlareW1LK5GDXT0Zb-lo0akWMAU0MBqcmvPoCGkBoK2bEHCuPYQ.oCDOLxATZB3dxKqLVCdIsYe89zpy7zKxcVxKfC8ag8w&dib_tag=se&keywords=12%2BV%2Blipo&qid=1728006567&s=toys-and-games&sprefix=12%2Bv%2Blipo%2Ctoys-and-games%2C99&sr=1-2&th=1			
12 V Lipo Charger	https://www.amazon.com/SUPULSE-Battery-Charger-7-4-11-1V-B3V2/dp/B099K8XFG6/ref=pd_bxgy_d_sccl_1/145-9217241-3264016?pd_rd_w=iN3rT&content-id=amzn1.sym.3858a394-39a9-4	1	$13.99	$13.99

	946-90e6-86a3153d2546&pf_rd_p=3858a394-39a9-4946-90e6-86a3153d2546&pf_rd_r=BSCCQM55J3P06JD9JV76&pd_rd_wg=EgFSP&pd_rd_r=4d7fbff9-802b-43c1-9fb4-221ec2b34919&pd_rd_i=B099K8XFG6&psc=1			
Foamboard	https://www.dollartree.com/black-foam-boards-20x30-in/25957	3	$1.25	$3.75

Table 6: Bill of materials for components not on the PCB

Name	Hourly Rate	Total Hours	Total (x2.5)
Aditya	$50	150	$18750
Kevin	$50	150	$18750
Lohit	$50	150	$18750
Total Labor Cost:			$56250

Table 7: Labor costs for all members

The total cost for our complete drone system, including the hardware components and labor, is approximately: $56368.086.

4.2 Schedule

Week	Tasks	Responsibility
9/29	1) Complete Design Document and Proposal Regrade 2) Complete initial design of schematic and PCB	1) Everyone 2) Lohit
10/6	1) Prepare for Design Review	1) Everyone

	2) Review schematic with experienced individuals 3) Begin prototyping drone shell 4) Conduct testing on network stack (frontend/cloud/backend)	2) Aditya 3) Lohit and Kevin 4) Lohit and Aditya
10/13	1) Complete Team Evaluation I 2) Make revisions to PCB 3) Continue testing on network stack (SIM7600) 4) Construct drone shell	1) Everyone 2) Lohit 3) Aditya and Kevin 4) Lohit and Kevin
10/20	1) Make revisions to PCB 3) Unit test sensors 3) Unit test servos and rotors 4) Complete frontend UI	1) Lohit 2) Aditya 3) Kevin 4) Lohit and Aditya
10/27	1) Make revisions to PCB 2) Test and verify microcontroller 3) Code primary embedded control loop 4) Assemble and test complete drone hardware	1) Lohit 2) Aditya and Kevin 3) Kevin 4) Lohit
11/3	1) Complete Individual Progress Reports 2) Make revisions to PCB 3) Code sensor modules	1) Everyone 2) Lohit 3) Aditya and Kevin
11/10	1) Perform embedded software debugging 2) Perform test flights and refine flight characteristics	1) Aditya 2) Lohit and Kevin
11/17	1) Complete Team Contract Fulfillment 2) Organize Mock Demonstration materials 3) Conduct flight controls refinements 4) Conduct general debugging	1) Everyone 2) Kevin and Aditya 3) Lohit 4) Everyone
11/24 (Fall Break)	Fall Break	Fall Break
12/1	1) Prepare Final Paper 2) Prepare Final Presentation 3) Prepare Demonstration	1) Kevin 2) Aditya 3) Lohit

12/8	1) Complete Final Paper 3) Checkout lab materials	1) Everyone 3) Aditya

Table 8: Schedule and tasks for all members

5 Ethics and Safety

Ethical Issue: The drone's ability to provide real-time video surveillance and tracking in public or private spaces raises serious concerns about privacy. Since drones may operate in environments where individuals have a reasonable expectation of privacy, such as homes or sensitive locations, the deployment of the drone has to be properly considered.

- IEEE/ACM Code of Ethics: Both the ACM Code of Ethics (section 1.6) [3] and the IEEE Code of Ethics (section 1) [4] emphasize respecting the privacy and autonomy of individuals. The project must avoid infringing on people's right to privacy.
  Prevention Measures: To avoid privacy violations, the drone should have defined operational protocols, such as geographic boundaries or "no-fly" zones, to avoid areas where privacy could be compromised. Additionally, transparent communication with the public about the drone's use and purpose can help to mitigate privacy concerns.

Ethical Issue: Since the drone collects and transmits sensitive data, including video feeds and sensor data, there is the potential risk of data breaches or tampering.

- IEEE/ACM Code of Ethics: Section 2.9 of the ACM Code of Ethics [3] stresses the need to design systems that protect the privacy and security of data. The IEEE Code of Ethics also advocates for ensuring data security and avoiding harm (section 1) [4].
  Prevention Measures: To ensure data security and integrity, encryption protocols should be applied to the communication channels (e.g., TCP encryption methods). Additionally, multi-factor authentication (MFA) and strict access controls should be enforced, especially if cloud storage is used.

Ethical Issue: Drones can pose physical risks, such as collisions with buildings, people, or wildlife, or interference with other airborne vehicles. Ensuring the physical safety of the public is a critical consideration in both the design and operation of the drone.

- IEEE/ACM Code of Ethics: The IEEE Code of Ethics emphasizes prioritizing public safety (section 1) [4]. Similarly, the ACM Code of Ethics (section 1.2) stresses the responsibility to avoid harm [3].
  Prevention Measures: The drone should be equipped with collision avoidance systems (ECS shutdown on command) and programmed to adhere to established FAA drone regulations, including maintaining a safe altitude and distance from populated areas [5]. Geofencing technology can also ensure the drone does not operate in restricted or hazardous areas. Regular maintenance checks and firmware updates should be part of the operational protocol to avoid malfunctions.

Ethical Issue: Deploying drones in public spaces without the public's consent or knowledge may undermine trust and raise concerns about government or institutional overreach. Citizens may feel uncomfortable or surveilled if drones are present in their neighborhoods or public spaces without clear communication.

- IEEE/ACM Code of Ethics: The ACM Code of Ethics (section 1.7) emphasizes the need to honor confidentiality and avoid misleading the public [3]. The IEEE Code of Ethics underscores the importance of transparency and honesty (section 5) [4].
  Prevention Measures: To address this, it is important to work with local law enforcement and municipal agencies to ensure transparency regarding how, when, and where the drones are deployed. Regular public consultations and open communication channels will help maintain trust. Additionally, signage or notifications should be placed in areas where the drone is actively surveying.

Ethical Issue: The project must adhere to relevant federal, state, and local regulations governing drone usage. This includes obtaining necessary certifications from the Federal Aviation Administration (FAA) and following safety regulations.

- IEEE/ACM Code of Ethics: Both the ACM and IEEE Codes of Ethics (ACM section 1.5 [3] and IEEE section 7 [4]) stress the importance of abiding by applicable laws and regulations.
  Prevention Measures: In Champaign, Illinois, the drone system must comply with FAA Part 107 regulations governing the operation of unmanned aerial vehicles (UAVs) [5]. This includes rules about keeping the drone within visual line of sight, avoiding flying over people without waivers, and operating during daylight hours. Additionally, all operators should receive proper drone pilot certification. The system should include built-in compliance features, such as geo-fencing to prevent the drone from flying in restricted areas (e.g., near airports) [6].

Safety Concern: Mechanical or electronic failures in critical drone components, such as the propulsion system or servos, could lead to accidents, resulting in potential harm to people, property, or the drone itself.

- Prevention Measures: Regular maintenance and pre-flight inspections will be mandatory to ensure all components are functioning properly. The drone will undergo routine performance checks to detect wear and tear early. Redundant systems should be implemented for critical functions, and real-time diagnostics will monitor the drone’s systems during operation. This aligns with the IEEE Code of Ethics, which emphasizes prioritizing public safety (section 1), and the ACM Code of Ethics (section 1.2), which stresses the responsibility to avoid harm.

Safety Concern: Drone operation in sensitive environmental areas, such as wildlife preserves or nature reserves, could disrupt ecosystems or harm wildlife.

- Prevention Measures: The drone will be programmed with geofencing technology to avoid sensitive environmental areas. Specific operational protocols will be developed in collaboration with environmental agencies to ensure minimal disturbance. Flight altitude and duration will be adjusted in these areas to reduce any environmental impact. This complies with the IEEE Code of Ethics (section 1) and ACM Code of Ethics (section 1.2), which focus on avoiding harm.

Safety Concern: Loss of communication between the drone and the backend system could result in erratic or unsafe behavior, potentially causing accidents or loss of the drone.

- Prevention Measures: The communication system will incorporate failsafe protocols, such as retaining the current state, predefined location (home point) in case of a signal loss. Additionally, signal encryption and interference mitigation strategies (e.g., frequency hopping) will be used to ensure stable and secure communication. Regular testing in various conditions will be performed to assess the reliability of the communication link. This aligns with the ACM Code of Ethics (section 2.9) and IEEE Code of Ethics (section 1), which advocate for data security and reliability.

Safety Concern: The drone's batteries pose a fire hazard, especially in cases of overheating or physical damage.

- Prevention Measures: The drone will use certified, high-quality batteries with integrated thermal management systems to prevent overheating. The battery compartment will be reinforced to minimize the risk of damage in the event of a collision. Additionally, battery levels will be monitored continuously during flight, and pre-flight safety checks will include ensuring batteries are not overcharged or showing signs of wear. This supports the IEEE Code of Ethics (section 1) regarding public safety and the ACM Code of Ethics (section 1.2) regarding harm prevention.

6 Works Cited

[1] Illinois Department of Public Health, "EMS Median Response Times," *Illinois Department of Public Health*, 2019. [Online]. Available: https://dph.illinois.gov/topics-services/emergency-preparedness-response/ems/prehospital-data-program/emsresponsetimes.html. [Accessed: Sept. 12, 2024].

[2] H. K. Mell et. al., "Emergency Medical Services Response Times in Rural, Suburban, and Urban Areas," *National Library of Medicine*, Oct, 2017. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5831456/. [Accessed: Sept. 12, 2024].

[3] *ACM Code of Ethics and Professional Conduct*, Association for Computing Machinery, 2018. [Online]. Available: https://www.acm.org/code-of-ethics. [Accessed: Sept. 17, 2024].

[4] *IEEE Code of Ethics*, Institute of Electrical and Electronics Engineers, 2020. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: Sept. 17, 2024].

[5] *Small Unmanned Aircraft Systems (Part 107)*, Federal Aviation Administration, 2016. [Online]. Available: https://www.faa.gov/uas/commercial_operators/part_107. [Accessed: Sept. 17, 2024].

[6] *Campus Drone Policy*, University of Illinois Urbana-Champaign. [Online]. Available: https://cam.illinois.edu/policies/drone-policy/. [Accessed: Sept. 17, 2024].

[7] Espressif, "ESP32 Series Datasheet Including," 2024. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. [Accessed: Oct. 2, 2024].

[8] Free Tutorials, "MPU6050 breakout board details," *YouTube*, Oct. 08, 2019. https://www.youtube.com/watch?v=159I9QYUpNU. [Accessed: Oct. 2, 2024]..

[9] "Adafruit Learning System," *Adafruit.com*, 2024. https://learn.adafruit.com/assets/93017. [Accessed: Oct. 2, 2024].

[10] I. Garibi, "Center of Gravity, and Center of Lift," *PowerUp Toys*, 2021. https://poweruptoys.zendesk.com/hc/en-us/articles/204840965-Center-of-Gravity-and-Center-of-Lift. [Accessed: Oct. 2, 2024].

[11] “How to Make Custom ESP32 Board in 3 Hours | Full Tutorial,” *www.youtube.com*. https://www.youtube.com/watch?v=S_p0YV-JlfU. [Accessed: Oct. 3, 2024].

[12] “Drone as First Responder (DFR) | City of Fremont, CA Official Website,” *Fremont.gov*, 2023. https://www.fremont.gov/government/citywide-initiatives/public-safety-initiatives/drone-as-first-responders-dfr. [Accessed: Oct. 3, 2024].

[13] Steve King Ph.D, S. Major, and M. McCollum, “Drone as First Responder Programs: A New Paradigm in Policing,” *MITRE*, Aug. 11, 2023. https://www.mitre.org/news-insights/publication/drone-first-responder-programs-new-paradigm-policing. [Accessed: Oct. 3, 2024].

[14] “CMOS OV7670 Camera Module,” *Components101*. https://components101.com/modules/cmos-ov7670-camera-module-pinout-features-datasheet. [Accessed: Oct. 3, 2024].

[15] “Drone as First Responder (DFR) - Skydio Public Safety Solutions,” *Skydio.com*, 2024. https://www.skydio.com/solutions/public-safety/. [Accessed: Oct. 3, 2024].