Custom Flight Controller for FPV Drone

ECE 445 Design Document - Spring 2025

Project #42

Hulya Goodwin, Muhammad Rabbani, Jaelynn Abdullah
Professor: Viktor Gruev
TA: Jason Jung

Contents

1. Introduction

1.1. The Problem

First Person View Drones, or FPV drones, first were invented in 1999, but the term FPV was coined officially in 2002/2003 by an online forum user “Cyber-Flyer” [1]. Quickly after, these drones went from something engineers were building themselves to being commercially available. As FPV drones became more accessible for hobbyists and the technology was getting better, these drones became even more popular. It’s common to even see one flying across the quad on a warm Spring day. The problem with this popularity is that the average person who wants to pick up this hobby can be intimidated by all of the expensive drones that aren’t beginner friendly. In the FPV Reddit thread FPV drone users have said to have crashed at least 8 drones before they finally got the hang of how to use them safely without damage. [2] The cost of building an FPV drone is estimated to be between $400-1800, which can be a steep price for beginners who will most likely crash their first drones. [3] This makes getting into the hobby of FPV drone flying daunting for people who may not have an income that would allow them to spend-at minimum estimation- $400*6 = $2400 just to get the hang of flying drones without crashing them. When looking at the current drone market, there are categories of ‘quadcopters’, ‘GPS’, ‘FPV’, ‘Mini”, etc. There is a gap in the current market for beginner friendly drones that are specialized to be cheap, durable, and an easy way to ease into the FPV drone hobby.

1.2. The Solution

To address this gap in the current market for FPV, we will create our own custom flight controller and put together a drone to display its functionality. This custom flight controller (FC) will have all the necessary components to control the FPV drone ie. camera, IMU, radio controller/transmitter. On top of these basic components, we will add a humidity sensor to the FC, to let the user know if it’s going to rain while they’re flying. The FC will be compatible with the most common Open Source software for FPV drones, which is Betaflight, so that new users can get accustomed to the software they will most likely be using for their more advanced future FPV drones as well. If the user is content with our FC, they can easily continue to use it for their future drones, since they can customize their frames/ESCs because our FC will be compatible with a range of Electronic Speed Controllers (ESCs) and drone frames.

1.3. Visual Aid

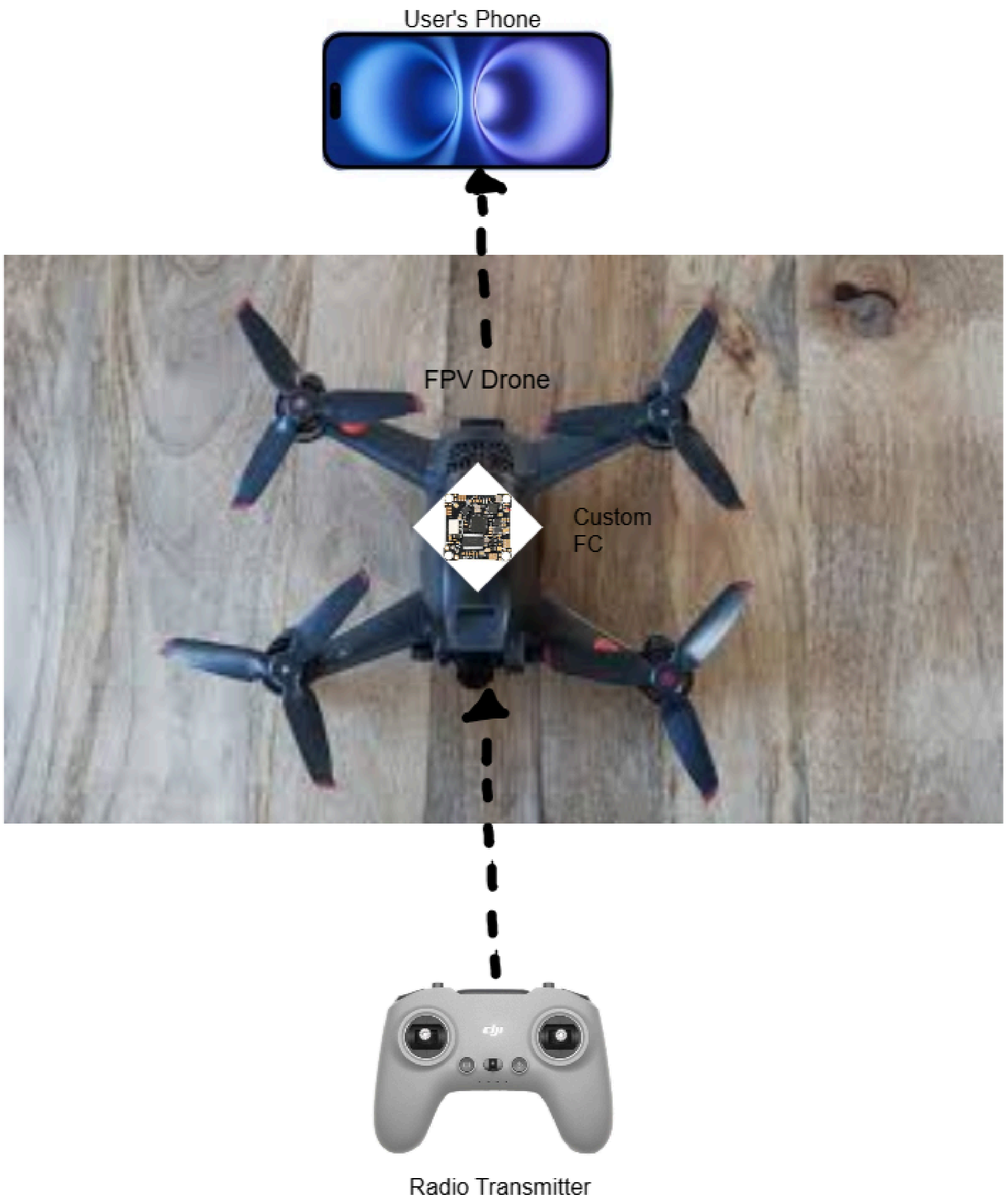


Figure 1, High Level Visual Aid of Solution

As seen in *Figure 1,* the overall delivered project will be a functional drone that runs based off of the custom FC. The drone frame will be purchased, along with the motors, propellers, electronic speed controllers (ESCs), and the radio transmitter. The FC will be mounted onto the drone frame as seen in *Figure 1*.

1.4. High Level Requirements

To determine that we have successfully made the custom FC, our project must complete the following requirements:

1. Demonstrate a functional flight controller that controls the motors to make balanced motions in the entire 360° plane. Motors can be powered and controlled for up to a full minute.
2. Demonstrates that the flight controller receives/sends accurate data from/to the microcontroller and integrated sensors. Video can stream 20+ FPS and sensors receive accurate live data within 5% error. Latency is below 100ms.
3. Drone has a functional system that turns on an LED and beeps when humidity sensors sense 90% air humidity or any raindrops. Humidity alert is audible and visible to users up to 10 feet away.

2. Design

2.1. Physical Design of Drone

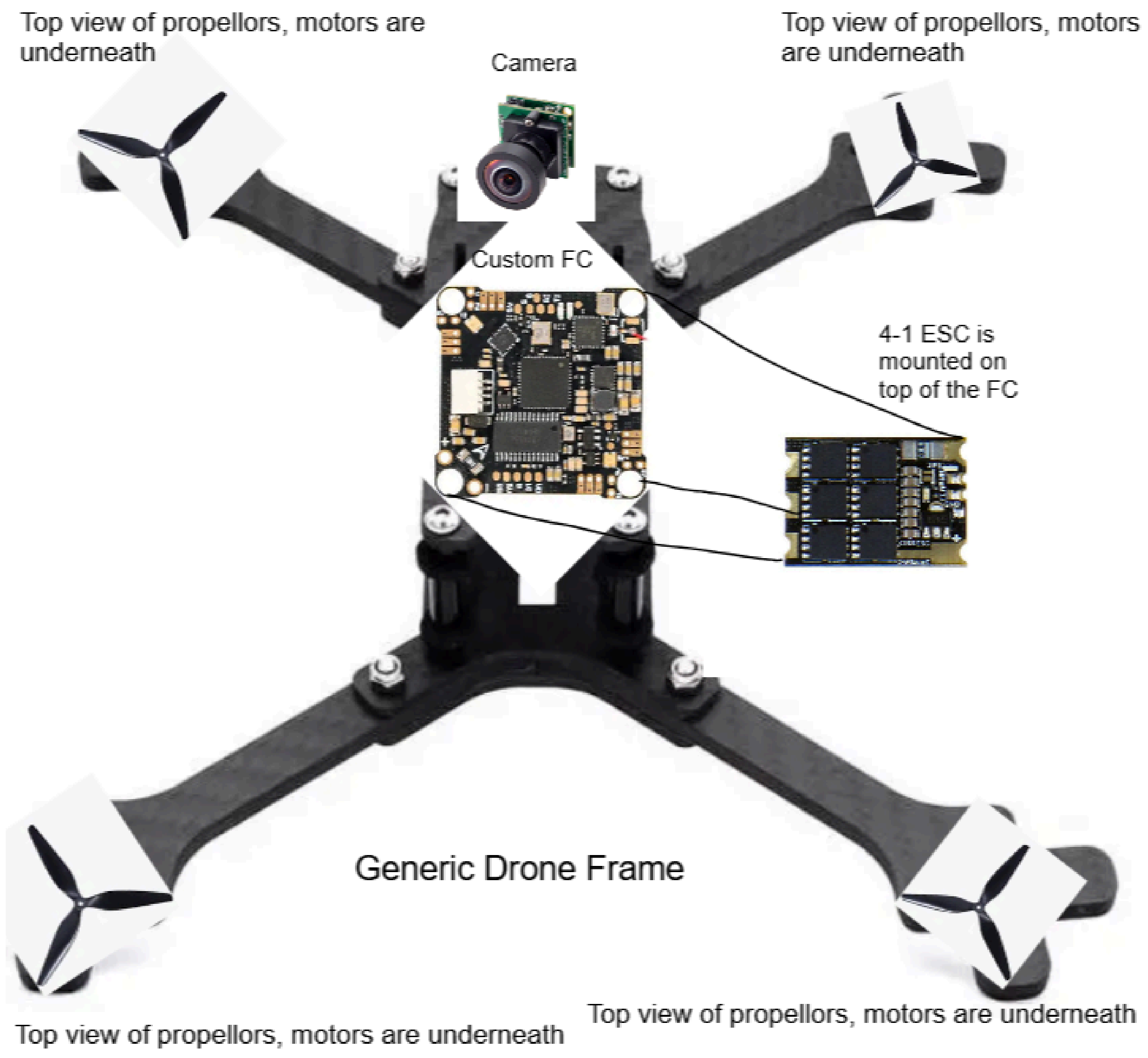

Top view of propellors, motors are underneath
Camera
Top view of propellors, motors are underneath
Custom FC
4-1 ESC is mounted on top of the FC
Generic Drone Frame
Top view of propellors, motors are underneath
Top view of propellors, motors are underneath

Figure 2. Physical Design of Drone

As seen in Figure 2, the physical design of the delivered project will be an X shaped frame with 4 motors connected to propellers in their respective corners. Each motor will have its own respective ESC, and all 4 ESCs are compacted into one chip in the center that is mounted above the FC. The center of the drone frame will have the FC mounted onto it, as well as the camera.

2.2. Block Diagram

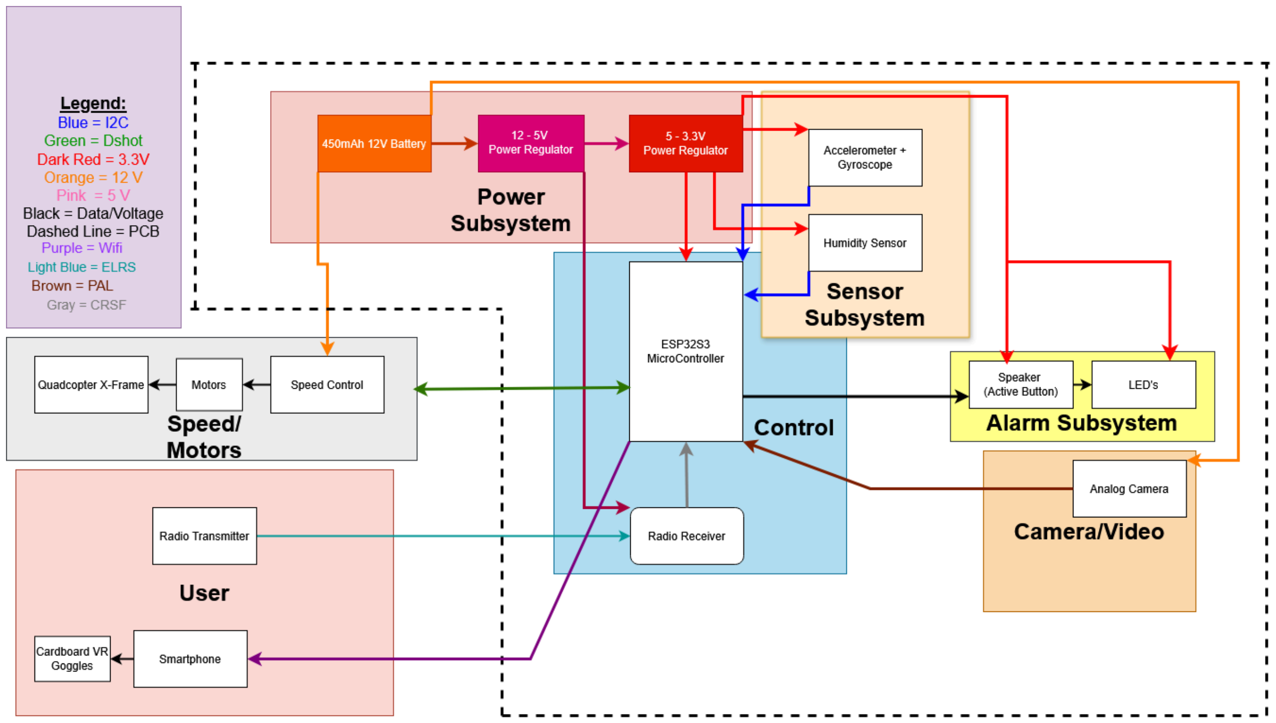


Figure 2. Block Design of Proposed Project

2.3. Functional Overview and Block Diagram Requirements

2.3.1. Control Subsystem

For the main control unit, we are opting for an ESP32S3 microcontroller to provide communication between all the subsystems. By modifying the open-source Betaflight software to add additional capabilities, it will be responsible for controlling the ESC to control the speeds of the brushless motors and receiving data back on the motor speed, triggering the alarm system in high humidity conditions, transmitting video data to the user’s device, receiving input from the remote controller to change the direction of flight through the radio receiver, and recording IMU data. The specific requirements and verifications of these requirements can be found below in Figure 3.

Requirements	Verifications
The ESP32 must be able to take in data from the sensor subsystem i.e. the speed/orientation of the drone and the humidity sensor and send the alert signal to the alarm subsystem if humidity is too high.	● Use a multimeter to ensure that the pin on the microcontroller will be an input to the alert subsystem and send a voltage>1 V to be accepted as a logical 1. ● Connect the micontroller's sensor ports as outputs of the microcontroller to the computer's terminal and send the IMU data to the terminal through UART. See the data changes as we move the IMU's speed/orientation.
The ESP32 must be able to take in the RF data from the Radio Receiver on how to move the drone through UART.	● Set up a UART port on the ESP32 that prints incoming CRSF data to the computer's monitor using Arduino Framework. ● Check on BetaFlight's Ports Tab in Betaflight Configurator and ensure the UART connected to ESP32 has Serial RX enabled and set the receiver protocol to CRSF under the "Receiver" tab.
The ESP32 must be able to convert the analog camera's data to a digital signal.	● Connect the video output of the camera to Channel 1 of the oscilloscope and connect the oscilloscope ground to the camera ground. Adjust the oscilloscope's

	time base to 50 µs/div (to see horizontal sync pulses), voltage scale to 0.2V - 1V/div (PAL signal is about 1V peak-to-peak), and the trigger to negative edge (to lock onto sync pulses). ● Make sure that there are sync pulses **(** A PAL composite signal has regular dips at 0V), that the image data varies in voltage (representing brightness levels), that the horizontal sync occurs every 64 µs, and that the vertical sync occurs **e**very 20 ms to match PAL timing on the oscilloscope. ● Write Aruduino code that sets up the A2D converter and prints the digital outputs to the serial monitor, we will know that they're right if they're in the 12 bit range of 0-4095.
The ESP32 must be able to receive data from the ESCs on the speed of the motors.	● In Betaflight Configurator: Go to Configuration Tab → ESC/Motor Features and enable Bi-directional DShot. In the Ports Tab, enable ESC Telemetry (RX) on the appropriate UART that's set to receive the data from the ESCs. Go to the Motors Tab in Betaflight and verify that ESC telemetry values (RPM, voltage, temperature) update when motors are running.
The ESP32 must be able to send the digitized camera data to the user's phone through Wifi.	● In the code to connect the ESP32 to Wifi, have print statements that print out if Wifi is connected and print the IP Address that the ESP is connected to.

	● Ping the IP address on a nearby computer. ● Open a web browser on a phone and type in [http://*IP_address/camera_data*](http://IP_address/camera_data) to stream the data on the phone.
The radio receiver must be able to get the data from the radio transmitter (determines direction of movement) within the range of a 500Hz- 1000Hz receive refresh range from the user subsystem.	● Check the LED on the radio receiver and ensure it's a solid light, indicating a stable connection to the radio transmitter. ● In Betaflight Configurator, go to the Receiver tab to check the receiver's status and make sure that the receiver is enabled and properly bound to the transmitter and that the correct ExpressLRS protocol is selected under the Ports tab in Betaflight. ● Go to the Receiver tab in BetaFlight and observe the real time receiver's input on the screen. Test by moving the radio transmitter's stick and seeing how quickly that data appears on screen at ranges of 1fr, 4ft, and 10ft away.
The radio receiver must be able to send the received RF data into the ESP32 through UART.	Verify the same way that we verify the ESP32 is inputting that data.

Figure 3. R and V table for Control Subsystem

2.3.2. Power Subsystem

The power subsystem is responsible for powering the components on board. The 12V LiPo battery is lightweight and a commonly used component for FPV drones. The total calculated current draw is around 33A with a predicted flight time of around 1.5 minutes. The regulators will step down the voltage to 5V and 3.3V to power the different subsystems. All specific requirements and verification of the requirements can be seen in Figure 4.

Requirements	Verifications
LiPo 14.8V 450 mAh Battery must provide continuous current of 33A and deliver 14.8V to power the flight controller with a maximum amount of 32A to Motors and 1A for remaining sensors and camera subsystem.	● Connect an ammeter to the power ports of the Motors, camera and sensors to ensure that the expected current is provided. ● Put a multimeter on the Power pin of the ESP32 to ensure it gets 3.3V.
Voltage Regulator 12V to 5V must provide clean 5V to power receiver and camera subsystem (+/- .3V)	● Connect a multimeter to the input pin of the 12-5 regulator and make sure the input is 12V +/- .3V. ● Connect a multimeter to the output pin of the 12-5 regulator and make sure the output is 5V +/- .3V.
Voltage Regulator 5V to 3.3V must provide clean 3.3V to power microcontroller and sensor subsystem (+/- .1V)	● Connect a multimeter to the input pin of the 5-3.3 regulator and make sure the input is 5V +/- .1V. ● Connect a multimeter to the output pin of the 5-3.3 regulator and make sure the output is 3.3V +/- .1V.

Figure 4. R and V table for the Power Subsystem

2.3.3. Sensor Subsystem

The IMU will measure the drone's acceleration and angle as it is controlled by the user. Using this data, we can detect at what speed the drone is under maximum load, what angle the drone is at to allow the user to correct the flight path, and the position of the drone.

The humidity detector will be a crucial component as it will be connected to the alarm subsystem that alerts the user of a high humidity level and alerts the user to bring the drone back to them and power it off.

The exact requirements and their respective verification methods can be seen below in Figure 5.

Requirements	Verifications
Humidity Detector must be able to accurately represent the relative humidity(RH) percentage in the air around the PCB by taking 6 measurements/second of the RH and saving this value into a 16-bit register.	● Connect the humidity sensor to the ESP32 through I2C protocol: Sensor's Vdd is connected to 3.3V, Sensor's GND is connected to ESP32's GND, Sensor's SDA is connected to ESP32's SDA (Data Line) and the sensor's SCL is connected to the ESP32's SCL (Clock Line) ● In Arduino IDE, download the SHT3x library (our sensor's library) and write code to receive the temperature and humidity measurements from the sensor that's uploaded to the ESP32. Then check the Arduino Serial Monitor to see these measurements and verify their accuracy in given environments.
The humidity sensor must be able to communicate relative humidity% to the Control sub-system and send the ALERT	● Connect the ALERT pin of the sensor to a GPIO pin on the ESP32 and configure this GPIO pin as an interrupt input to detect when the ALERT pin is triggered in Arduino

interrupt to the ESP32 if RH is above 90%.	IDE. ● Simulate a high humidity environment and ensure that the ALERT interrupt goes off when RH is 90%.
The IMU's Accelerometer + Gyroscope must be able to accurately represent the acceleration of the physical drone by taking measurements in the X,Y,Z axis with maximum measurable acceleration before saturation set to +16g (g equals about 9.81 m/s^2)	● Write Arduino code to configure the accelerometer to +16g and print out the measured X,Y,Z values in terms of gravity. Move the IMU around at different forces to simulate different g's and verify that the raw data should be in the range of -32768 to 32767 for +16g. ● Write Arduino code to configure the gyroscope to 2000m/s and print its measurements to the serial monitor. Ensure when it's flat on a surface the X,Y,Z values are 0. Rotate the IMU on only the x-axis and ensure only X values change, etc.
Must be able to accurately represent the pitch, yaw, and roll of the physical drone throughout the full scale 360° range within an error of ±15% to the Microcontroller through I2C protocol at 400 kHz.	● Betaflight provides a real-time view of the pitch, yaw, and roll of the drone in the "Flight Data" tab, under the "Angle" indicator. To verify this data: gently move the IMU in all directions (pitch up, pitch down, roll left, roll right, yaw left, yaw right) and observe the changes in the 3D model or the angle indicator in the Betaflight Configurator and verify that the pitch, roll, and yaw values match the expected orientation based on our physical movements. ● To test error margin, rotate the drone to known angles (e.g., 0°, 90°, 180°, 270° for each axis), and compare the Betaflight displayed values to the expected angles.

	Ensure the displayed angles stay within the required ±15% margin of error.

Figure 5. R and V table for the Sensor Subsystem

2.3.4. Camera Subsystem

The camera subsystem is what will define 'FPV' for our drone. This allows new users to view what the drone sees. The Caddx Ant Lite 1200TVL FPV Camera will be responsible for measuring analog video data and sending it to the Control Subsystem to be digitized. The exact requirement(s) and their respective verification methods can be seen below in Figure 6.

Requirements	Verifications
Analog camera must be able to capture 20+ FPS video feed from on top of the drone.	● Connect the analog data output of the camera to a monitor to watch it if we can find an old enough monitor that supports RCA cables. ● Connect the video output of the camera to Channel 1 of the oscilloscope and connect the oscilloscope ground to the camera ground. Adjust the oscilloscope's time base to 50 µs/div (to see horizontal sync pulses), voltage scale to 0.2V - 1V/div (PAL signal is about 1V peak-to-peak), and the trigger to negative edge (to lock onto sync pulses). ● Make sure that there are sync pulses (A PAL composite signal has regular dips at 0V), that the image data varies in voltage (representing brightness levels), that the horizontal sync occurs every 64 µs, and that the vertical sync occurs every 20 ms to match PAL timing on the oscilloscope.

Figure 6. R and V Table for Camera Subsystem

2.3.5. Alarm Subsystem

The alarm subsystem is what we will define as the LED and Active Button. This allows for both visual and audible feedback to the user for when humidity levels are dangerous/potentially damaging for the drone to fly in. A basic red LED and 5V Active Buzzer will be used for our Alarm Subsystem. The exact requirement(s) and their respective verification methods can be seen below in Figure 7.

Requirements	Verifications
5V Active Buzzer must be able to receive high DC input from ESP32 Alert in response to humidity sensor to create sound of 100 dB audible within 100 ft.	● Apply a 5V high signal directly to the buzzer, using a standalone multimeter and voltage source. ● Use a calibrated sound level meter to measure the buzzer's dB output to 5V at 3ft, 10ft, 100ft distances ● Measure voltage from designated GPIO pin during ESP32 Alert to ensure 3.3V-5V ● Ensure Alarm buzzes from ESP32 Alert once all connected
Red LED must be able to light up on high DC input from ESP32 Alert in response to humidity sensor	● Apply a 5V high signal directly to the LED, using a standalone multimeter and voltage source. ● View the LED output to 5V at 3ft, 10ft, 100ft distances ● Measure voltage from designated GPIO pin during ESP32 Alert to ensure 3.3V-5V ● Ensure LED lights from ESP32 Alert once all connected

Figure 7. R and V table for the Alarm Subsystem

2.3.6. User Subsystem

On the user end, the user will be responsible for controlling the drone via a remote controller that has a transmitter. The user can view the camera during operation while wearing the cardboard goggles that display their smartphone. The requirements and their verifications can be seen in Figure 8 below.

Requirements	Verifications
Radio Transmitter must be able to wirelessly send data to the Radio Receiver in accordance with physical joystick inputs under 100ms latency.	● In BetaFlight, under the Receiver section, set the Receiver Mode to Serial-based Receiver and choose CRSF for the receiver protocol and set the Serial Receiver Provider to CRSF. Also, set the Serial Baud Rate to 400,000. ● In the Ports Tab on BetaFlight, find the UART that the ELRS receiver is connected to and enable Serial RX for the appropriate UART port. In the Receiver tab in Betaflight Configurator while moving the joystick of the LiteRadio2 transmitter, observe the channels (e.g., Throttle, Roll, Pitch, Yaw) and make sure that the values change in real-time as we move the joysticks. This confirms that the receiver is receiving signals from the transmitter and correctly mapping them to the flight controller. ● Measure the seconds between the joystick movement and mapping onto the channel, ensuring it's consistently less than 100ms at different distances with range of the transmitter.
Smartphone must be able to receive live video stream from the ESP32's Wifi in 20+FPS quality and below 100ms latency.	● Make sure that the user's phone can connect to Wifi (any Wifi) and then specifically the Wifi of the ESP32. ● Open the video stream on our smartphone and start counting frames in a 10-second interval to

	ensure that the number of frames displayed in this interval equals or exceeds 200 frames (for 20 FPS). ● Setup an LED in front of the camera and flash the LED on and off every second. Start a stopwatch as soon as the LED flashes and observe the time it takes for the flash to appear on the smartphone screen. The latency is the time between the moment we initiate the flash and the moment we see it on the smartphone.
Cardboard VR Goggles must fit the smartphone and give the optical illusion of VR goggles and be sturdy enough to support the weight of the smartphone for multiple wears.	● Insert our smartphone into the compartment of the VR goggles and ensure that the smartphone fits snugly without excessive extra room or pressure on the screen based on its dimensions. ● Wear the goggles for 30 minute increments multiple days in a row to ensure that they won't break after multiple long wears.

Figure 8. R and V table for the User Subsystem

2.3.7. Speed/Motor Subsystem

The motors subsystem consists of the speed controller, the brushless motors, and quadcopter frame. Located in all 4 corners of the quadcopter frame will be the brushless motors that are controlled by the ESC to control the direction and speed of travel. The directions of travel will include up, down, left, right, straight, and backwards using the joysticks on the remote controller. Each motor draws up to 8A and must be continuous during flight. The requirements and their verifications can be found in Figure 9 below.

Requirements	Verifications
Electronic Speed Control must be able to receive control inputs from the ESP32 from the Radio Transmitter to send varying Dshot signals to speed up or slow	● In Betaflight Configurator, manually adjust the throttle slider in the Motors tab (in the Motors tab set Master Switch to do this) to test

down the motors/propellers.	each motor and verify that it is receiving and responding to the DShot signal. Make sure that the motor's speed increases or decreases as we vary the throttle signal in Betaflight, which will show the Dshot values should change accordingly. ● Disable Master Switch and use the radio transmitter to see how the motors will move and if the Dshot values are changing in the BetaFlight Configurator.
Quadcopter X-Frame must be able to house motors, propellers, PCB, and battery and must be able to move throughout the 3D plane according to motor controls.	● Take measurements of the motors, propellers, PCB, and battery to make sure that they will fit onto the frame. ● When moving the joysticks of the radio transmitter, make sure that the drone moves in a balanced manner in any X,Y,Z direction.

Figure 9. R and V table for the Speed/Motor Subsystem

2.4. Hardware Design

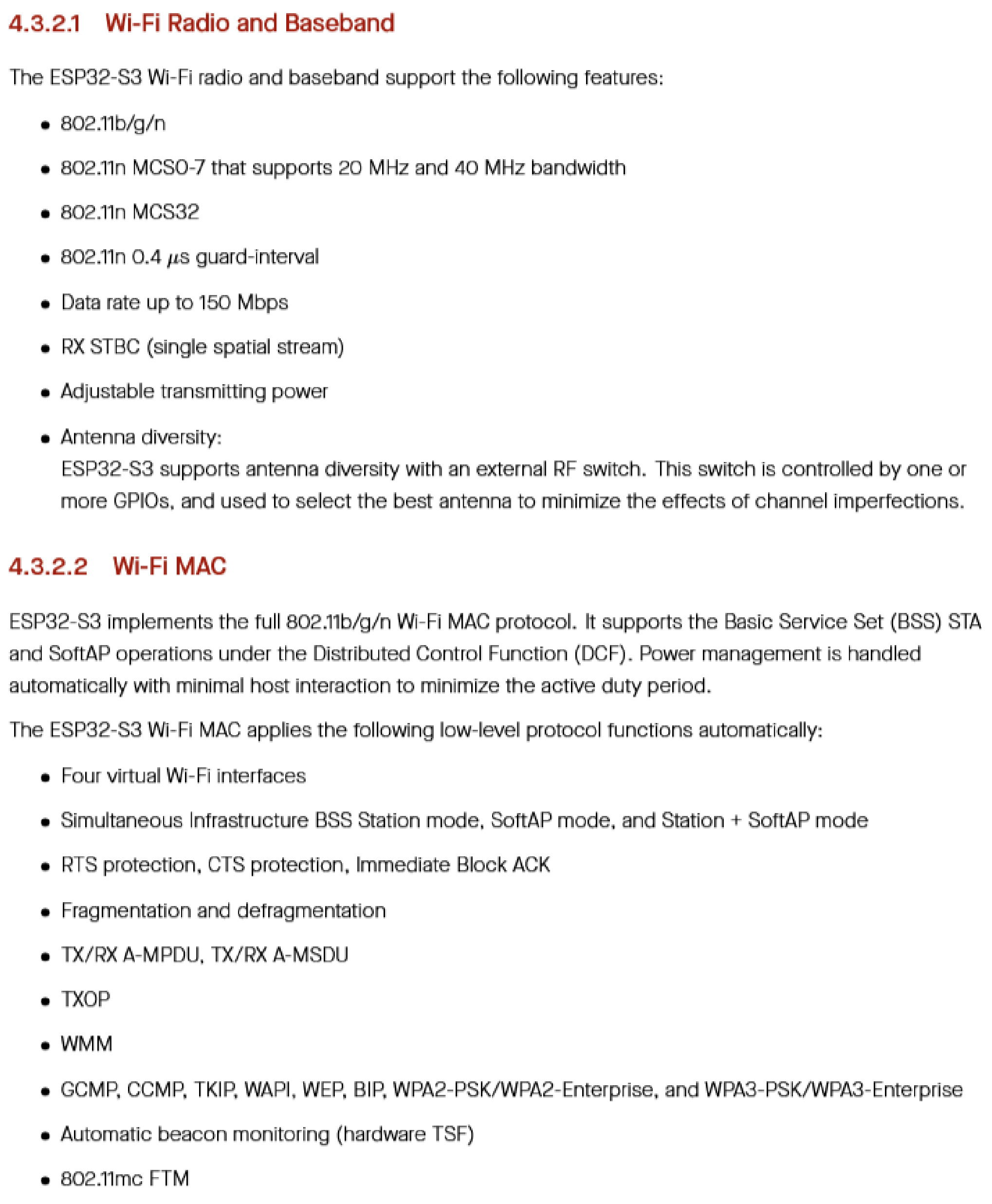
2.4.1. Signal Integrity and Protection

Since noise and there are critical parts needed to operate in order ensure the success of our project, voltage regulators as well as additional resistors and capacitors were used across power traces. From the battery, a 5V voltage regulator and a 3.3V voltage regulator. Our camera, receivers, and other peripherals need a constant 5V to operate while our microcontroller and sensors require a constant 3.3V. 100 microfarad decoupling capacitors are placed at the input and 10 nanofarad decoupling capacitors output terminals of the voltage regulators in order to prevent static-hazard glitches as well as send in cleaner signals.

For the EN and VDD pins, we are using 10 kOhm resistors as pull-up resistors to ensure we have a constant high during operation. Similar to the using I2C and UART lines, we will be using 10 kOhm to ensure the signals are clean at the input due to the importance of sensor data for our drone.

2.4.2. Microcontroller Choice

Our team originally planned on using an STM32F4 Microcontroller. However, that would require a VTX, VRX, and OSD in order to stream the video to the user. We decided to minimize parts in the interests of minimizing weight of the FC, to use an ESP32 Microcontroller. Seen in Figure 10, the ESP32S3 has Wifi capabilities that we will be



4.3.2.1 Wi-Fi Radio and Baseband

The ESP32-S3 Wi-Fi radio and baseband support the following features:

- 802.11b/g/n
- 802.11n MCS0-7 that supports 20 MHz and 40 MHz bandwidth
- 802.11n MCS32
- 802.11n 0.4 μs guard-interval
- Data rate up to 150 Mbps
- RX STBC (single spatial stream)
- Adjustable transmitting power
- Antenna diversity:
  ESP32-S3 supports antenna diversity with an external RF switch. This switch is controlled by one or more GPIOs, and used to select the best antenna to minimize the effects of channel imperfections.

4.3.2.2 Wi-Fi MAC

ESP32-S3 implements the full 802.11b/g/n Wi-Fi MAC protocol. It supports the Basic Service Set (BSS) STA and SoftAP operations under the Distributed Control Function (DCF). Power management is handled automatically with minimal host interaction to minimize the active duty period.

The ESP32-S3 Wi-Fi MAC applies the following low-level protocol functions automatically:

- Four virtual Wi-Fi interfaces
- Simultaneous Infrastructure BSS Station mode, SoftAP mode, and Station + SoftAP mode
- RTS protection, CTS protection, Immediate Block ACK
- Fragmentation and defragmentation
- TX/RX A-MPDU, TX/RX A-MSDU
- TXOP
- WMM
- GCMP, CCMP, TKIP, WAPI, WEP, BIP, WPA2-PSK/WPA2-Enterprise, and WPA3-PSK/WPA3-Enterprise
- Automatic beacon monitoring (hardware TSF)
- 802.11mc FTM

Figure 10. Wifi Capabilities of an ESP32S3 from it's Datasheet[3]

utilizing to stream the video to the user's phone. Rather than use a VTX, we can use the built-in A/D converter on the ESP32 to convert the analog video into a digital signal that the ESP32 can digest and send over Wifi. This reduces the number of components required by 2, which is important on a scale so small for the FC. The biggest hesitation on switching to ESP32 was if BetaFlight will be compatible with it, but we have made it work and changed our original idea.

2.4.3. Weight & Size Management

Our main limitations with our FPV drone to achieve lift under its total physical weight are our motor RPM, propellor size, and battery capacity. Most sensors and control subsystems are chosen beforehand to achieve functionality. Their total weight (PCB & ESC) comes out to around 42.5 grams. For ease and efficiency, we chose a standard 5" frame to house our components as all components laid out fit comfortably within such a standard frame. With 5" propellers, we can decide on an efficient motor paired with a compatible battery. A motor with high documentation weighing below 50 grams we found was the EMAX RS2205S 2300KV. Pairing this with the LiPo battery, Or 14.8V 4S 650mAh 80C, gives us a total compatible system of motor and battery weighing under 100 grams total. This gives us ample room in terms of weight for our 5" frame and additional components as this motor consistently provides 360+ grams of thrust at 50% PWM [7]. Details of weight management can be found in Figure 14.

2.5. Software Design

2.5.1. Humidity Sensor Addition to BetaFlight

BetaFlight is one of the most common FPV drone softwares that's available to the public. As it has lots of documentation, we decided to use it. In order to add the humidity sensor, which isn't already included in the software we will follow the steps below:

1. Clone the BetaFlight repository to our local repositories, then push to a shared repo for the 3 of us.
2. Since the SHT30 humidity sensor uses the I2C protocol, we need to ensure I2C support is enabled in Betaflight by opening the src/main/ folder and checking if I2C support is already enabled, which we should already have. But if it isn't enabled, in our configuration file, config.h, we can enable it by defining: #define I2C_SPEED and #define USE_I2C
3. Based on how BetaFlight handles other sensors, we can create a SHT30.h and SHT30.c driver file in the Drivers folder of BetaFlight. The SHT30.c file will need to include the SHT30.h and the i2c.h file as it will need i2c to get the data.
4. We can define a struct that will hold the 16 bits of humidity data and temperate from the humidity sensor to use in the driver file.

5. We can make the modified BetaFlight (configure it) and flash it to the flight controller.

2.6. Tolerance Analysis

2.6.1. Noise Impacting Video Transmission

The part of the design that's the biggest risk to our design is the video transmission from our analog camera to our smartphone. Between noise from ESC, we need to ensure we have a filter to limit the amount of noise from the circuit for the worst case scenario. With Betaflight, they do not recommend filtering lower than 70 Hz, however, most high frequencies that would affect our VTX occur higher than 150-200 Hz. In addition, we will simulate a noise that has an amplitude of 500 mV. To filter this noise out, we are using a LPF that has a 1k ohm resistor and 470 microFarad capacitor. This configuration will allow an attenuation of ~.5V. With this, we can ensure that the ESC has a clean DC power source, and in the best case scenario that there is no noise, there still will be around 14V being supplied to the ESC, which is within the operating regime.

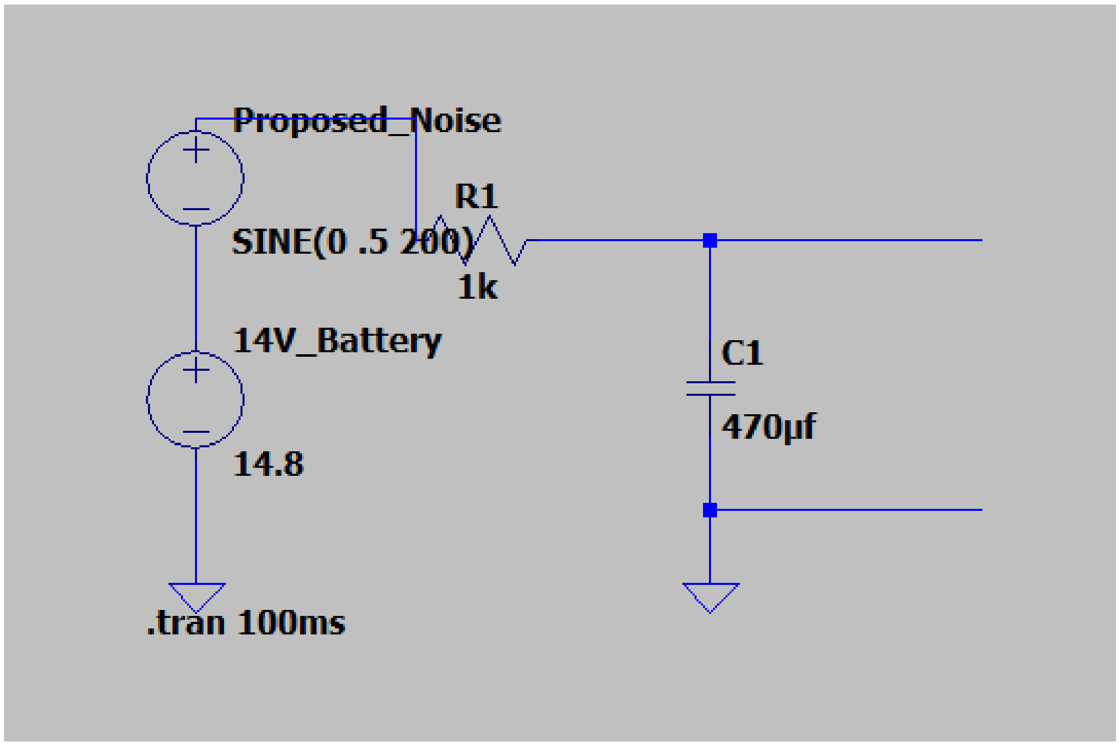


Figure 11, Schematic of LPF

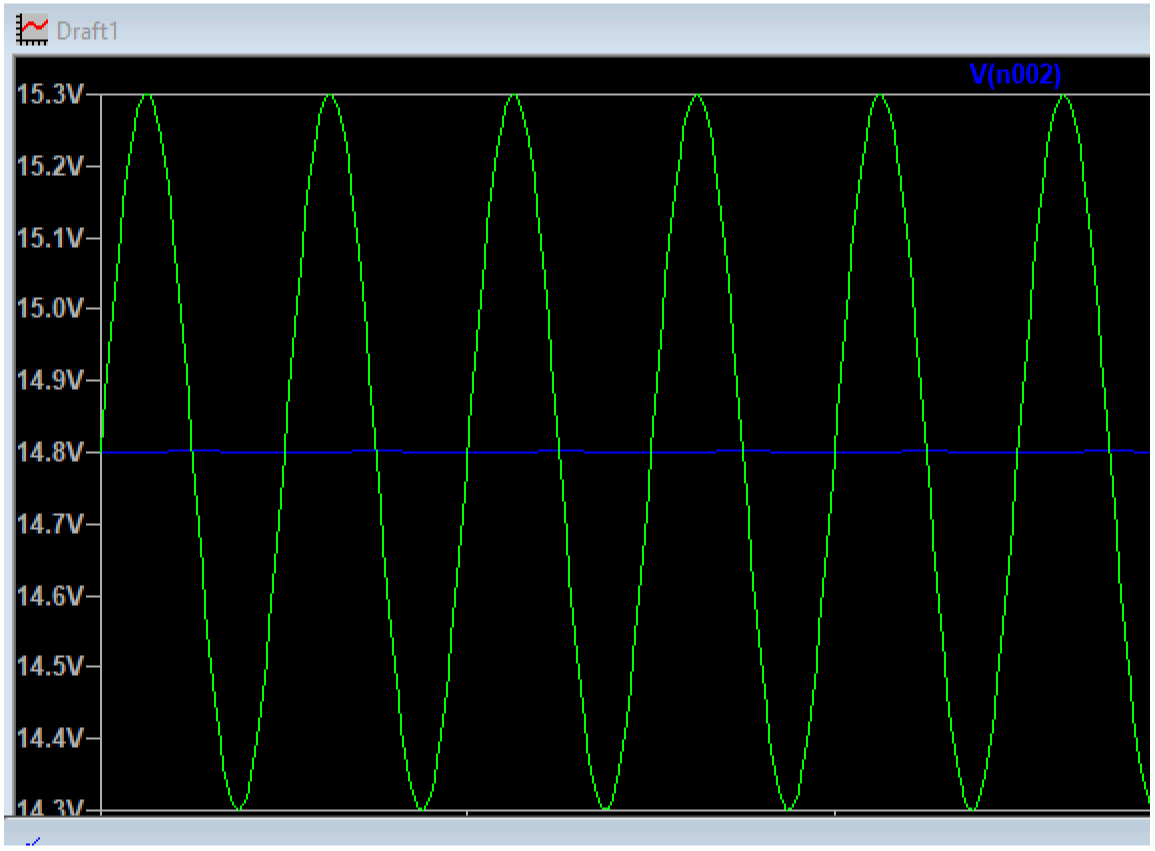


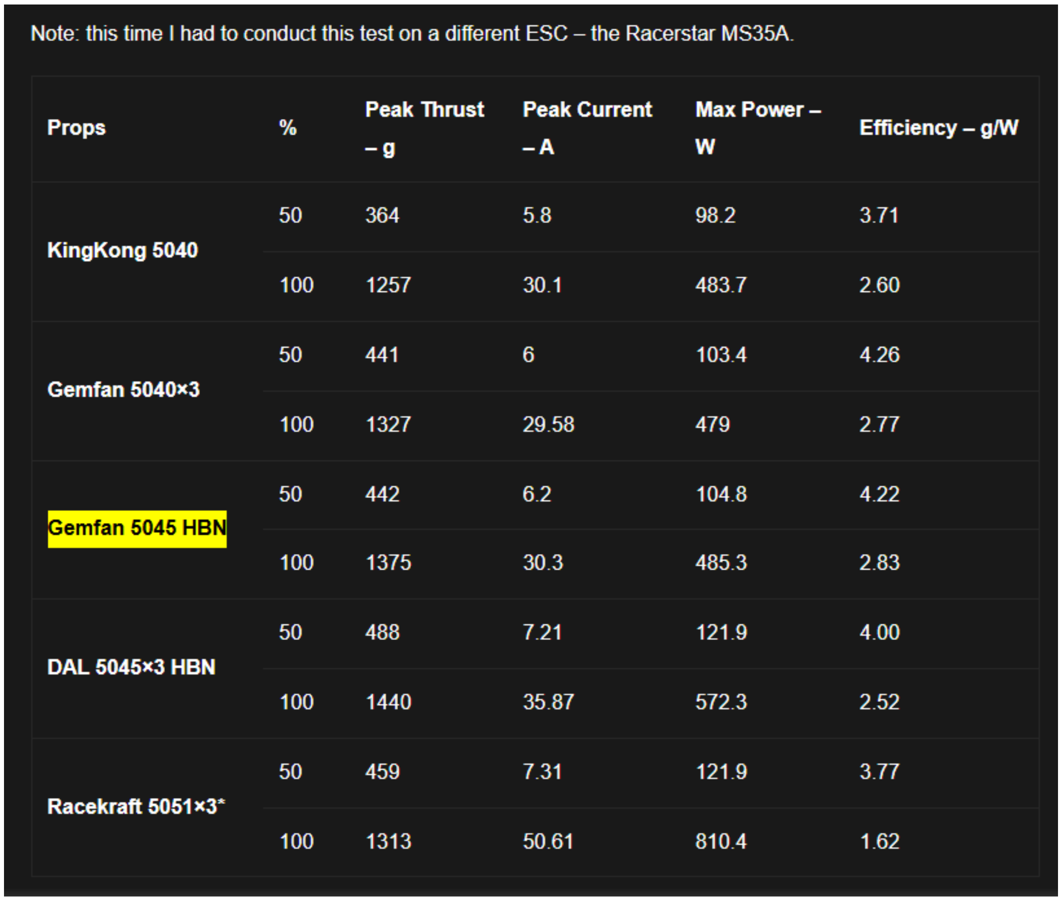
Figure 12, Waveform of voltage input with noise (green sine wave) and filtered output (blue line)

2.6.2. Motor Thrust vs. Weight

Another risk to our design is achieving lift despite the drone's weight. The table below shows our motor paired with a similar ESC and similar 5" propellors with thrust measurements at different percentage PWM's [7]. As seen below, all propellors provide 350+ [g] thrust at 50% PWM, achieving lift with our design specifications.

Another risk to our design is achieving lift despite the drone's weight. The table below shows our motor paired with a similar ESC and similar 5" propellors with thrust measurements at different percentage PWM's [7]. As seen below, all propellors provide 350+ [g] thrust at 50% PWM, achieving lift with our design specifications.

Additionally, calculating the total weight of our drone and its parts, we find that our weight is well below 350g. This gives us plenty of weight room to prevent potential risks.



Note: this time I had to conduct this test on a different ESC – the Racerstar MS35A.

Props	%	Peak Thrust – g	Peak Current – A	Max Power – W	Efficiency – g/W
KingKong 5040	50	364	5.8	98.2	3.71
	100	1257	30.1	483.7	2.60
Gemfan 5040×3	50	441	6	103.4	4.26
	100	1327	29.58	479	2.77
Gemfan 5045 HBN	50	442	6.2	104.8	4.22
	100	1375	30.3	485.3	2.83
DAL 5045×3 HBN	50	488	7.21	121.9	4.00
	100	1440	35.87	572.3	2.52
Racekraft 5051×3*	50	459	7.31	121.9	3.77
	100	1313	50.61	810.4	1.62

Figure 13, Table of thrust tests and measurements for RS2205S Motor

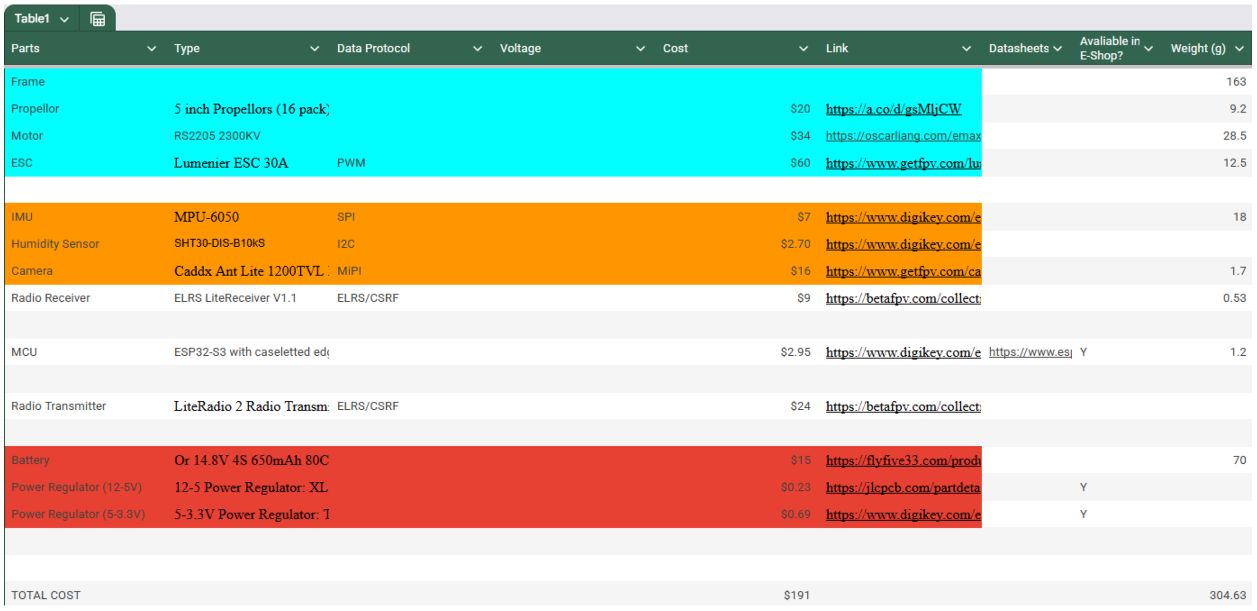


Table1

Parts	Type	Data Protocol	Voltage	Cost	Link	Datasheets	Avaliable in E-Shop?	Weight (g)
Frame								163
Propellor	5 inch Propellors (16 pack)			$20	https://a.co/d/gsMljCW			9.2
Motor	RS2205 2300KV			$34	https://oscarliang.com/emax			28.5
ESC	Lumenier ESC 30A	PWM		$60	https://www.getfpv.com/lu			12.5
IMU	MPU-6050	SPI		$7	https://www.digikey.com/e			18
Humidity Sensor	SHT30-DIS-B10kS	I2C		$2.70	https://www.digikey.com/e			
Camera	Caddx Ant Lite 1200TVL	MiPI		$16	https://www.getfpv.com/ca			1.7
Radio Receiver	ELRS LiteReceiver V1.1	ELRS/CSRF		$9	https://betafpv.com/collect			0.53
MCU	ESP32-S3 with caseletted ed			$2.95	https://www.digikey.com/e	https://www.es	Y	1.2
Radio Transmitter	LiteRadio 2 Radio Transm	ELRS/CSRF		$24	https://betafpv.com/collect			
Battery	Or 14.8V 4S 650mAh 80C			$15	https://flyfive33.com/prod			70
Power Regulator (12-5V)	12-5 Power Regulator: XL			$0.23	https://jlcpcb.com/partdeta		Y	
Power Regulator (5-3.3V)	5-3.3V Power Regulator: T			$0.69	https://www.digikey.com/e		Y	
TOTAL COST				$191				304.63

Figure 14, Table of all drone parts and average weights

3. Cost And Schedule

3.1. Cost Analysis

3.1.1. Labor

Our team is made out of two Computer Engineers and one Electrical Engineer. Looking at the UIUC report about starting pay of $109,176 and $87,769 respectfully, after taking the average of these two salaries and assuming 30 days per month as well as 8 hour work days, we can assume (if we are charging hourly and not on a salary) that each member of the team has an hourly rate of around $34. [5] Each week, members are expecting to work 6 hours for 12 weeks, giving us a total of **$7344** for labor.

3.1.2. Parts List

Item	Manufacturer	Cost	Quantity	Description
RS2205 2300KV Brushless Motors	EMax	$8.5	4	Brushless motors for drone
Lumenier Mini Razor Pro ESC 45A	GetFPV	$59.99	1	ESC for motors
ICM-20948	DigiKey	$7.11	1	9-axis IMU that has accelerometer, gyro, and uses I2C and SPI protocol
SHT30-DIS-B10kS Humidity Sensor	DigiKey	$2.70	1	Humidity Sensor for drone that uses I2C protocol
Caddx Ant Lite 1200TVL FPV Camera	GetFPV	$16	1	Mini camera for FPV camera
ELRS LiteReceiver V1.1	BetaFPV	$9	1	Radio Receiver

ESP32-S3-WROOM-U1	DigiKey	$2.95	4	Microcontroller
LiteRadio 2 Radio Transmission	BetaFPV	$24	1	Radio Transmission
LiPo 14.8V 4S 650 mAh 80C	Flyfive33	$15	1	14.8V battery
5 Inch Propellers (16 Pack)	Gemfan Hurricane	$13	1	Propellers for motors
5-Inch Frame	In-house	$13.47	1	Frame for Drone
Tax (11.5%)				
Shipping (5%)				
		Total: $241.26		

3.1.3. Total

The grand total for this project would be $7575.26. Considering that drone projects are a very expensive hobby, this was around the estimate we were expecting.

3.2. Schedule

Week	Tasks	Person
February 23 - March 1st	• Finalize and order parts • Flash Betaflight onto ESP32 using GitHub port • Begin schematic design	• Jaelynn • Jaelynn & Muhammad • Hulya
March 2nd - March 8th	• Begin breadboard demo for placement • Update schematic design and create PCB	• All • Hulya & Jaelynn

March 9th - March 15th	● Second round of PCB Design ● Have transmitter work with ESC and motors ● Have IMU and humidity sensor integrated to Betaflight using I2C and SPI ● Demo breadboard	● All ● Jaelynn & Muhammad ● Muhammad ● All
March 16th - March 22nd	● SPRING BREAK	
March 23rd - March 29th	● Debug PCB ● Redesign PCB ● Have LED and Alarm system integrated to Betaflight using GPIO ● Have frame designed	● Hulya ● Hulya & Jaelynn ● Muhammad & Jaelynn ● Muhammad
March 30th - April 5th	● Correct weight balancing ● Have camera integrated to Betaflight using GPIO ● Debug PCB ● Redesign PCB	● Muhammad ● Muhammad ● Hulya ● Hulya & Jaelynn
April 6th - April 12th	● Order second round of PCB ● Have Humidity Sensor calibrated to activate LED and Alarm system at desired humidity environment ● Ensure Motors are calibrated to Radio Transmitter	● Jaelynn & Hulya ● Muhammad & Hulya ● Muhammad & Jaelynn
April 13th - April 19th	● Test flights ● Solder new PCB ● Debug PCB ● Ensure live-streaming of camera to Wifi (standalone)	● All ● Hulya & Jaelynn ● Hulya & Jaelynn ● Muhammad

April 20th - April 26th	● Test flights ● Final debugging ● Ensure entire system achieves high level requirements	● All ● All ● All
April 27th - May 3rd	Mock Demo	All
May 4th - May 10th	Final Presentation Final Paper	All

4. Ethics And Safety

4.1.1. Ethics

With open-source hardware and software, we have a responsibility to ensure that our final product should be available for the public to continue the development of beginner friendly drones in accordance with IEEE 7.8.I.2 [6]. Moreover, with our criteria, we want to ensure that the user has honest performance reports with the appropriate tolerancing to upkeep safety and the integrity of our project.

Additionally, we have a responsibility to address ethical concerns regarding military applications and privacy violations. While our drone is meant for civilian recreational use, we acknowledge the potential misuse of our FPV drone in illegal and unauthorized aerial reconnaissance and surveillance. To prevent such abuse, we provide strict guidelines to any users and implement altitude restrictions and geofencing capabilities through our intended drone flight capability range. The limited range of our drone will allow us to mitigate any risk of potential military or surveillance abuse. Through this, we uphold IEEE 7.8.I.1 by prioritizing public welfare to ensure responsible technological use [6].

4.1.2. Safety

With our project, there are important considerations to keep in mind to ensure that the drone targeted towards new users protects the user and the environment in which the drone will take flight. Through our design, we considered thermal generation, safe voltage delivery within the flight controller, and battery management to prevent fire hazards. With the total load from the battery reaching around 33A, ensuring that users do not experience electrical shock while operating the drone is the highest priority. The battery will be housed and the right copper path size will be used to prevent shorting. Any wired connections will be bundled and twisted as needed to prevent loose connections. These and other safety considerations are based around 7.8.I.1 in IEEE's Code of Ethics to ensure the safety and use ethical design practices [6]. Moreover, by recognizing that the users are newer, in accordance with IEEE 7.8.I.6 [6], considerations were made into the design to limit the amount of training experience required to fly the drone and diagnose issues that occur. The alarm system and the built-in protection for the parts we want to order heavily contribute to our goal of protecting our users.

4.2. Safety Procedure

Ensuring the safety of users and the environment is a top priority when operating our FPV drone. The following safety procedures align with IEEE Code of Ethics guidelines and should be followed strictly to prevent accidents, injuries, or damage to property.

1. Pre-Flight Safety Checks

Before operating the drone, ensure the following:

- **Battery Inspection**: Check for any physical damage, swelling, or loose connections before inserting the battery. Only use manufacturer-approved batteries.
- **Wiring and Connections**: Inspect all power and signal wires for loose connections or exposed conductors. Secure any loose wiring to prevent in-flight disconnections.
- **Propeller Check**: Ensure that propellers are properly mounted and free from cracks or defects. Tighten them securely before flight..

2. Safe Flight Operation

- **Authorized Flight Areas**: Always check and follow local regulations regarding drone usage. Do not fly in restricted areas (e.g., near airports, military zones, or crowded public spaces).
- **Weather Conditions**: Avoid flying in strong winds, rain, or low-visibility conditions that can lead to loss of control.
- **Emergency Protocols**: Familiarize yourself with emergency shutdown procedures in case of malfunction. If the drone becomes unresponsive, safely disarm it.

- **Visual Line of Sight**: Maintain a clear view of the drone at all times unless operating in designated FPV (First Person View) zones with a trained spotter.
- **Altitude and Range Limits**: Keep the drone within legal altitude limits (e.g., 400 feet in the U.S.) and avoid exceeding the controller's signal range.

3. Electrical and Fire Safety

- **Battery Handling**:
  - Never short-circuit or puncture the battery.
  - Do not charge batteries unattended or near flammable materials.
  - Store batteries in fireproof containers when not in use.
- **Voltage and Current Protection**:
  - Ensure proper insulation of electrical components.
  - Use recommended voltage levels to prevent overheating.
  - Keep electronic components cool to avoid thermal runaway.

4. Post-Flight Procedures

- **Battery Disconnection**: Remove the battery after landing to prevent accidental activation.
- **Component Inspection**: Check for signs of wear, overheating, or damage after each flight. Replace faulty components immediately.
- **Data Logging and Diagnostics**: Review flight logs for irregularities and adjust settings accordingly.

5. Ethical and Regulatory Compliance

- **FCC and Aviation Regulations**: Follow all communication and frequency transmission regulations to prevent signal interference.
- **Privacy Considerations**: Avoid recording or flying over private property without consent.
- **Safe Community Use**: Respect local drone-flying communities and avoid reckless behavior that may endanger others.

By following these safety procedures, users can ensure the safe and responsible operation of the FPV drone while protecting themselves and the environment.

References

[1] CurryKitten. “The History of FPV.” CurryKitten. https://www.currykitten.co.uk/the-history-of-fpv/ (accessed Mar. 5, 2025).

[2] “How many drones have you crashed since you started flying?” Reddit. https://www.reddit.com/r/fpv/comments/12qhckv/how_many_drones_have_you_crashed_since_you (accessed Mar. 5, 2025).

[3] Le, S. Supporting, I. 11, G. Wi-Fi, and Le), “ESP32-S3 Series Datasheet 2.4 GHz Wi-Fi + Bluetooth Including.” Available: https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf

[4] O. Liang. "How to get started with an FPV drone – the ultimate beginner’s guide." https://oscarliang.com/fpv-drone-guide/#How-Much-Does-an-FPV-Drone-Cost (accessed Mar. 5, 2025).

[5] ECE Illinois. “Salary Averages.” University of Illinois Urbana-Champaign. https://ece.illinois.edu/admissions/why-ece/salary-averages (accessed Mar. 5, 2025).

[6] IEEE. “IEEE Policies: Section 7 – Statement of Policy.” IEEE. https://www.ieee.org/about/corporate/governance/p7-8.html (accessed Mar. 5, 2025).

[7] Max, et al. “Review - EMAX RS2205S 2300KV Motors.” *Oscar Liang*, 24 Apr. 2017, https://oscarliang.com/emax-rs2205s-2300kv-motors/