Republic of the Philippines
PHILIPPINE STATE COLLEGE OF AERONAUTICS
Institute of Engineering and Technology
Piccio Garden, Villamor, Pasay City

Development & Progress Report:

ESP32 Aerial Drone: A Custom Flight Controller

Group 1

March 28, 2025

TABLE OF CONTENTS

I. INTRODUCTION

Drones have grown in popularity in recent years due to their applicability to a variety of applications, including surveillance, delivery services, agriculture, and research. Advances in microcontroller technology, combined with the open-source community's collaborative efforts, have made it possible to create affordable, customizable drones. The goal is to investigate the potential of adopting low-cost hardware for autonomous flight while retaining performance levels comparable to traditional flight controllers.

The drone will have a camera for live video transmission and a GPS module for navigation and positioning. The drone's frame and structure will be created with 3D modeling software and built with 3D printing technology, resulting in lightweight, customizable, and durable components. This document will offer an overview of the hardware components, software setup, system architecture, and step-by-step implementation of the ESP32-powered drone.

As a starting point, we reviewed Pratik Phadte's ESP32 Flight Controller project on GitHub to explore its potential as the basis for our drone system. This project served as a helpful starting point, offering valuable insights into early DIY drone development within the community. However, we found that the implementation was somewhat outdated, with limited ongoing developer support. It relied on the Arduino IDE rather than Visual C++ paired with a software configurator, which would have allowed easier tuning and modification of the ESP32 flight controller settings. Despite this, the raw code from the IDE provided a useful reference for understanding the system's functionality. Additionally, the project utilized obsolete hardware components that, although still available, were not recommended for modern applications.

Despite not proceeding with this approach, it offered significant learning opportunities. The similarities between its design principles and the more advanced system we later adopted helped us strengthen our project's foundation. This experience deepened our knowledge of flight control algorithms, hardware integration, and the unique challenges of using an ESP32-based drone.

After carefully assessing different options, we opted for the ESP32 Flight Controller project from rtlopez's Github repository. This project was chosen for its recent development, active maintenance, and comprehensive documentation, making it easier to implement than older alternatives. It supports configuration through Microsoft Visual C++ and Betaflight Configurator, a widely used and trusted tool in the drone community, allowing straightforward flight parameter adjustments and fine-tuning to achieve stable flight performance.

One of the primary reasons for choosing rtlopez's works was its support for advanced, high-performance components. The project promotes the use of modern electronic speed controllers, motors, and sensors, ensuring better efficiency, reliability, and seamless integration. Unlike the initial flight controller we evaluated, ESP-FC is built on up-to-date hardware, making it a more future-ready option for our drone.

An essential benefit of ESP-FC is its active developer involvement. We were able to reach out directly to the developer, who provided insightful recommendations on component selection, setup configurations, and troubleshooting techniques. The developer's prompt and helpful assistance enabled us to resolve issues faster, minimizing the learning curve and boosting our confidence in the system.

Moreover, ESP-FC incorporates enhanced flight control algorithms, optimized PID tuning, and broader compatibility with communication modules like NRF24 and others. These upgrades make it a highly flexible solution for both manual and autonomous drone applications. With its combination of modern technology, straightforward configuration, and consistent developer support, ESP-FC became the most practical and dependable choice for our project.

MIDTERM PERIOD

With the development of microcontroller technology and the open source community, low-cost, customizable drones have become feasible. Our project set out to explore the potential of utilizing hardware, focusing on building a fully functional powered by an ESP32 microcontroller, guided by innovation, adaptability, and hands-on learning.

The goal was to design a drone With the advent of microcontroller technology and the open-source community, low-cost, customizable drones have become feasible, and performance parity with off-the-shelf flight controllers is now a reality. Our project set out to investigate the potential of utilizing low-cost hardware, focusing on building a fully functional quadcopter powered by an ESP32 microcontroller, guided by the principles of innovation, adaptability, and hands-on learning.

The goal was to design a drone featuring real-time video transmission through an onboard camera and autonomous navigation via a GPS module, supported by a lightweight, durable, and 3D-printed frame. With the help of 3D modelling we created a frame that maximized weight distribution and carried structural strengths, and minimized material costs. Motor and communication trials also had its own issues; Although we were able to get motors running through direct control, we faced communication stability issues

between the transmitter (ESP32) and the receiver (NRF24L01) as we had signal dropouts and oppressive latency after integration with BetaFlight, which involved a lot of debugging. We also implemented and tested two remote control systems: NRF24-based Arduino controller and Wi-Fi enabled ESP8266 controller. Some advancements were achieved, but many problems were still encountered, especially in our quest to achieve good signal stability and less latency in wireless transfer. We constructed a prototype landing pad to facilitate the testing of our drone outdoors to ensure the size compatibility with our drone's dimensions. Throughout the midterm period, continuous system integration, calibration fine-tuning, partial flight tests, and persistent troubleshooting marked our journey.

Further, we kept build logs, developed circuit diagrams and wrote test results ito ensure any future modifications, enhancements or open source contributions could be easily added. Each technical refinement, such as PID adjustments or flight parameters tuning, have all helped hem in a more stable and flexible flight character. This paper provides a comprehensive overview of our hardware selection, software configurations, flight system architecture, development milestones, and the detailed implementation process behind the ESP32-powered drone.

II. OBJECTIVES

1. Design and build a quadcopter drone with the ESP32 microcontroller using ESP-FC project, guaranteeing optimal aerodynamics, weight distribution, and structural strength.
2. Incorporate GPS for autonomous navigation and position hold to enhance flight accuracy.
3. Install and configure a camera module for real-time video transmission and aerial monitoring.
4. Fine-tune the PID settings, motor response, and sensor calibration to achieve a balanced and efficient flight experience.
5. Construct a lightweight yet durable drone frame using 3D printing or carbon fiber materials for improved aerodynamics and strength.
6. Build a custom remote controller using an ESP32-based transmitter-receiver setup to ensure low-latency communication with the drone.
7. Leverage Betaflight or INAV Configurator for easy firmware updates, flight mode adjustments, and motor calibrations.
8. Perform multiple flight tests, troubleshoot hardware/software issues, and refine configurations for improved reliability.
9. Maintain detailed build logs, circuit diagrams, and flight test results for potential open-source contributions and future enhancements

III. MATERIALS AND COSTING

PRELIMINARY PERIOD

Components	Quantity	Cost
100nf Ω Capacitor	2	₱ 23.00
10µF/50V	2	₱ 2.00
2.2k Ω Resistor	1	₱ 1.00
22k Ω Resistor	1	₱ 1.00
2k Ω Resistor	1	₱ 1.00
3.3k Ω Resistor	1	₱ 1.00
47uf Ω Capacitor	1	₱ 1.00
5V ACTIVE BUZZER	2	₱ 22.00
ArduinoNano	2	₱ 300.00
B10K Single Pot	1	₱ 12.00
BC9V	1	₱ 5.00
Cap 223pf	5	₱ 2.50
DYS AM32 65A 2-85 4in1 ESC - 30x30mm	1	₱ 2935.00
ESP32 DevKitC	1	₱ 349.00
EZ1084CT 3.3V	1	₱ 55.00

FR4-5x7 White	2	₱ 52.00
GY-BMP280-3.3	1	₱ 40.00
HQProp 7X3.5X3 Light Gray (2CW+2CCW)-Poly Carbonate	1	₱ 220.00
LGR5MM	5	₱ 10.00
Metal Jacket Transistor 2N2219A	1	₱ 140.00
Metal Jacket Transistor 2N2222A	1	₱ 140.00
MPU-6050	1	₱ 92.00
MTG21SW	2	₱ 118.00
NRF24 PA+LRA Antenna	2	₱ 360.00
Paste	1	₱ 19.00
PNP transistor 2N905A	1	₱ 1.00
Resistor 1/4wt	30	₱ 15.00
SkyRC B5 Neo 200w DC Smart Charger - no Power Supply	1	₱ 1,825.00
SoloGood 2807 1300KV FPV Brushless Motor 4~6S	4	₱ 2,720.00
Stranded Wire 18AWG	–	₱ 150.00
SYB-120	1	₱ 108.00
Tattu R-Line Version 3.0 1800mAh 4s 120c Lipo Battery	1	₱ 1700.00

Thumb Joystick	2	₱ 96.00
Total:		**₱ 11,516.50**

MIDTERM PERIOD

Components	**Quantity**	**Cost**
2.4 GHz NRF24L01 Module with PA LNA SMA Antenna 1KM	2	₱
B50K Single Potentiometer	1	₱ 13.00
BC557 PNP Gen	2	₱ 6.00
Cable Ties (White)	1	₱ 30.00
Illustration board	1	₱ 15.00
BMP 180 Barometer	2	₱ 84.00
HMC5883L	1	₱ 154.00
DC-DC Converter	1	₱ 49.00
Female Header	2	₱ 76.00
Logic Level Converter Bidirectional	1	₱ 175.00
SW2 Slide Switch	2	₱ 16.00
Triple Axis Accelerometer and Gyro Breakout - MPU6050	1	₱ 139.00
YIHUA-08B Rosin Flux Soldering Iron Tip	1	₱ 90.00

IV. PROJECT DEVELOPMENT

PRELIMINARY PERIOD

ESP32 FLIGHT CONTROLLER

CODING

Project Milestones

- Testing of ESP32 Board Versions for problems during installation of firmware
- Application of Gyroscope

Progress

- Installation of complete code/firmware in ESP32 boards (WROOM32, S2, D1 Mini) was completed and successful in doing so.
- Gyroscope functioning properly. Awaiting electronics in order to further development of coding

Challenges

- Development of code for the different versions of the ESP32 boards in use for the drone and the controller
- The bought Magnetometer(HMC5883C) and Barometer(BMP280) were not responding accordingly. Through further research, no solution was found to troubleshoot the issue, and might not incorporate both since it does not contribute to drone flight stabilization. **(Further troubleshooting recommended)**

ELECTRONICS

Project Milestones

- Flight Controller Size Optimization
- Stacked PCB Design

Progress

- In the process of making a 35x35mm form factor
- Solution to size constraints by implementing a stacked PCB.

Challenges

- Designing mentioned form factor (35x35mm) whilst maintaining functionality
- Ensuring signal integrity and efficient component placement

NRF24 ARDUINO REMOTE CONTROLLER

CODING

Project Milestones

- Complete Remote Controller Code
- Joystick Pin Connections Calibration

Progress

- Controller wirings are checked and verified
- Controller Code is implemented on Arduino

Challenges

- Difficulty in establishing proper pin assignments for the joysticks but was resolved.

ELECTRONICS

Project Milestones

- Optimizing the size of the receiver
- Overall remote development
- Designing a custom transmitter circuit

Progress

- Developing a compact design to achieve a smaller form factor
- Remote controller still undergoing overall development
- Successfully tested and verified wiring connections

Challenges

- The PCB layout exceeds the available space on the designated circuit board within the drone frame
- Finalizing integrations and making hardware adjustments
- Achieving precise alignment and optimal orientation of components to enhance layout efficiency

STRUCTURE

Project Milestones

- Design Software Selection: The drone frame design will be created using 3D modeling software, such as Autodesk Fusion 360, to prepare for 3D printing.
- Defining Size Constraints: A border will be added to define the size limits of the frame, ensuring it stays within the required 1ft x 1ft dimensions.
- Propeller Clearance Planning: The layout of the 5ft propellers will be incorporated into the design to prevent any contact with other components.
- Reference Frame Selection: The initial design will be inspired by an existing 250mm FPV drone frame to serve as a reference for structure and layout.

- Arm Structure Development: The arms of the drone will be designed, extending from the propeller locations to the center of the frame.
- Battery Measurement Integration: The rough measurements of the Li-Po battery will be taken to ensure proper fitting within the frame.
- Main Plate Sizing: The center plate of the drone will be designed to be slightly larger than the battery dimensions for secure placement.
- Structural Reinforcement: The thickness of the arms will be adjusted and integrated with the center plate to improve structural integrity.
- Sketch Refinement: Unnecessary lines and elements will be removed to clean up the sketch and reveal the refined shape of the drone.
- 3D Model Preview: The initial frame design will be extruded to create a 3D preview of its overall shape and structure.

Progress

- The initial design of the drone frame was based on an existing Carbon Fiber 250mm FPV drone frame.
- A preliminary sketch has been started, but final adjustments will be made once all components are received.
- The target design aims to combine a traditional DIY drone frame with a modern shell for protection and aerodynamics.
- Major drone parts, such as the main plate and arms, will be professionally 3D printed for durability.
- Minor drone parts, like mounts and casings, will be 3D printed using the school's available resources.
- The frame design will be reviewed by a professional to verify its durability and suitability for flight conditions.
- The review process aims to prevent errors in 3D printing and minimize material waste to reduce costs.

Challenges

- The final frame design cannot be completed until all necessary drone components and other hardware are available. Accurate measurements of these components are essential to ensure they fit securely within the frame without compromising the overall structure or aerodynamics.
- Ensuring that the drone remains within the required 1ft x 1ft size constraint while accommodating a 5ft propeller layout is a significant design challenge. The frame must be structured in a way that allows for proper propeller clearance without interfering with other components. Precise positioning of each element is necessary to avoid inefficiencies and potential mid-flight instability.
- The decision to professionally 3D print major drone components, such as the main plate and arms, comes with added costs. To minimize expenses, precise planning is required to ensure that the design is finalized and error-free before printing. Any design mistakes that require reprinting could lead to increased material costs and extended project timelines.

MIDTERM PERIOD

ESP32 FLIGHT CONTROLLER

CODING

Project Milestones

- Installing Airtag using OpenHaystack

Progress

- Successfully installed the OpenHaystack for tracking using iCloud/FindMyAccount

Challenges

- Troubleshooting and flashing the firmware

ELECTRONICS

Project Milestones

- Testing the connections between the ESP and ESC
- Running motor tests through the ESP and ESC
- Adjusting the Buck Converter output from 16V down to 5V
- Integrating a logic level shifter into the system

Progress

- Established a working connection between ESP and ESC as part of system integration.
- Successful motor operation was achieved through signal transmission from the ESP, with regulation by the ESC.
- Completed the modification of the Buck converter to achieve a 5V output from a 16V input

Challenges

- Ongoing testing efforts have improved but not fully stabilized the wireless communication between the transmitter and receiver.

- While the motors functioned successfully, they became unresponsive when interfaced with the ESP during Beta Flight testing.

NRF24 ARDUINO REMOTE CONTROLLER

CODING

No changes were needed to be made

ELECTRONICS

Project Milestones

- Finalizing compact designs for the ESP32 flight controller and the Arduino-based NRF24 receiver
- Updating and revising the program codes

Progress

- Achieved a finalized, space-optimized design for the ESP32 flight controller and receiver module.
- Conducting code tests for both the transmitter and receiver

Challenges

- Ongoing troubleshooting of wireless data transmission between the transmitter and receiver
- Test codes were implemented to diagnose the issue; partial success was achieved but full functionality is still lacking
- Observed response delays on the receiver side when signals are sent via the potentiometer

ESP Wi-fi REMOTE CONTROLLER

CODING

Project Milestones

- Proper calibration of the ESP8266

- *Ensure connection through wi-fi is stable*

Progress

- The ESP8266 has been flashed and is in standby but the code and Betaflight requires a review

Challenges

- Betaflight Controller does not receive the inputs from the cellphone controller

STRUCTURE

Project Milestones

- All components have been measured.
- Finalized drone frame design.
- Printing of the final design of the drone frame.

Progress

- All the components to be installed on the drone frame have been measured, and 3D models were created to preview their fit.
- The initial drone frame design was refined to accommodate all components, including adding mounting holes where needed.
- A sample frame was 3D printed to check the actual fit of the components and ensure the screw holes were the correct size.
- The required hardware and fasteners have been ordered.
- The final version of the drone frame has been successfully 3D printed.

Challenges

- Some components do not have dedicated mounting holes, so we will use zip ties, Velcro straps, or double-sided tape to secure them.
- The initial cost of the drone frame exceeded our budget, so we had to adjust the design to reduce both weight and material cost.

- Although the frame has been printed, we cannot complete the drone assembly yet due to delays in the shipment of the fasteners.

LANDING PAD STRUCTURE

Project Milestones

- All materials have been measured, divided, and assembled for polishing.
- Base has been built.

Progress

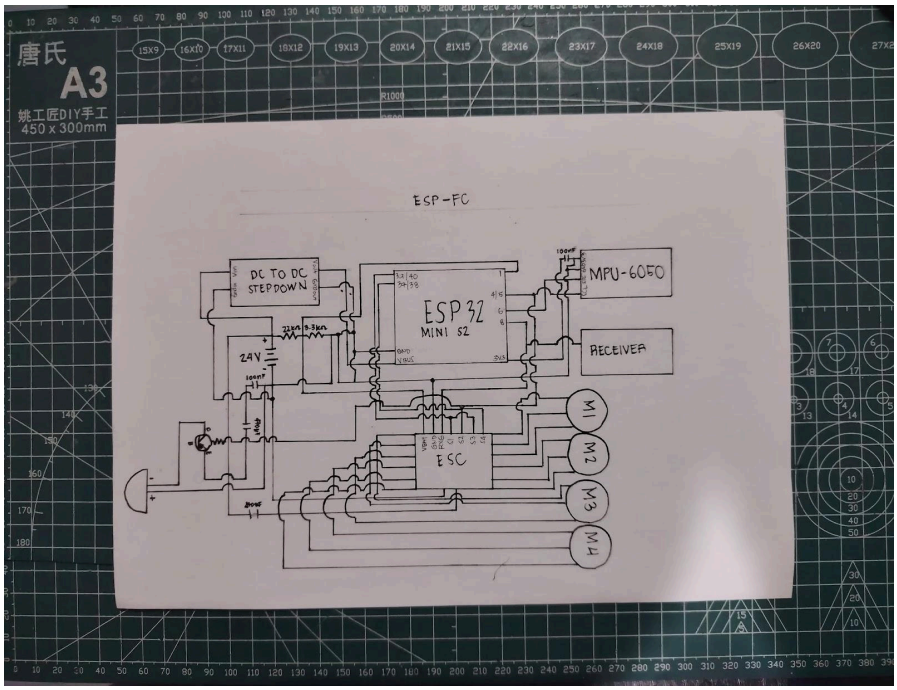
- The design of the landing pad was based on a circular shape to fit the drone's landing needs.
- Lightweight and durable materials were chosen for easy handling.
- The size of the landing pad was decided based on the drone's dimensions.
- The first prototype was created using sketches and 3D models.
- Construction of the prototype began using a sketch and some manual assembly.

Challenges

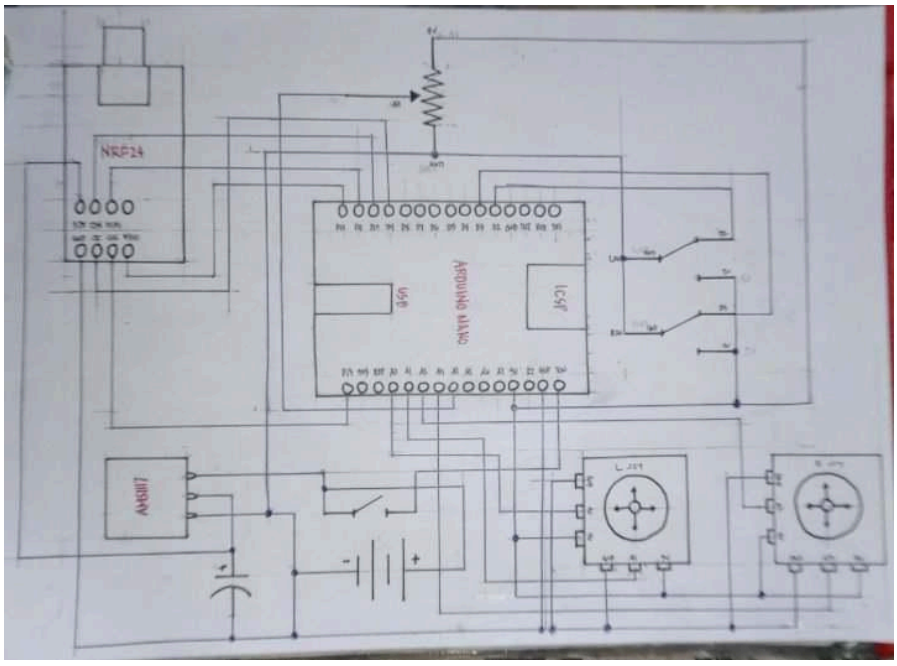
- Ensuring the chosen materials are durable enough to withstand repeated use and weather conditions (if outdoor testing is involved).
- Managing costs, especially if adjustments to the design are needed to improve stability or durability.
- No sufficient testing has commenced to ensure the pad is stable and can support the drone's weight during landing.

V. SCHEMATIC DIAGRAM

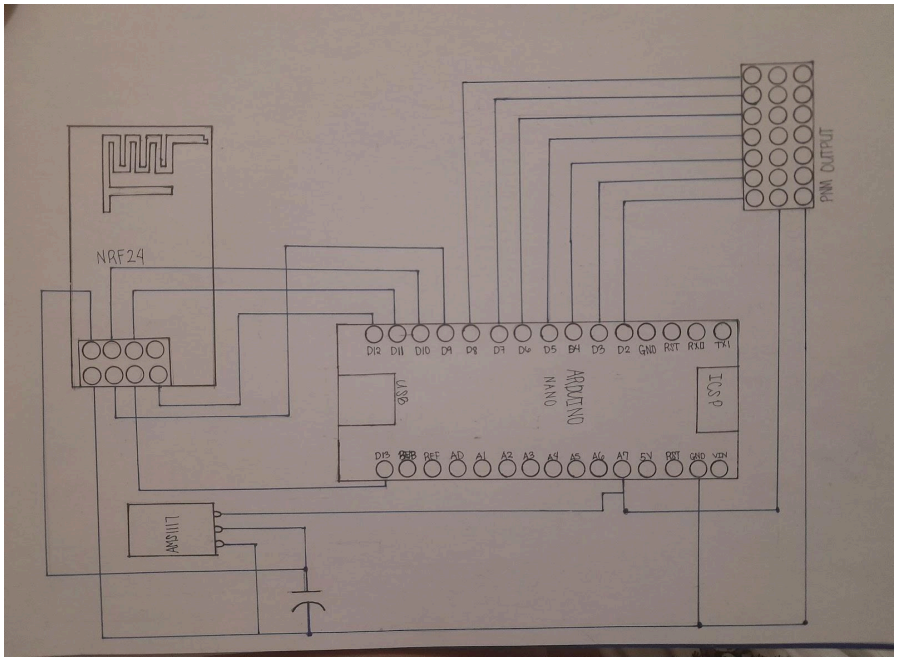
ESP32 S2 MINI FC



ARDUINO NRF24 RC TRANSMITTER



ARDUINO NRF24 RC RECEIVER



VI. STEP-BY-STEP PROGRESS

PRELIMINARY PERIOD

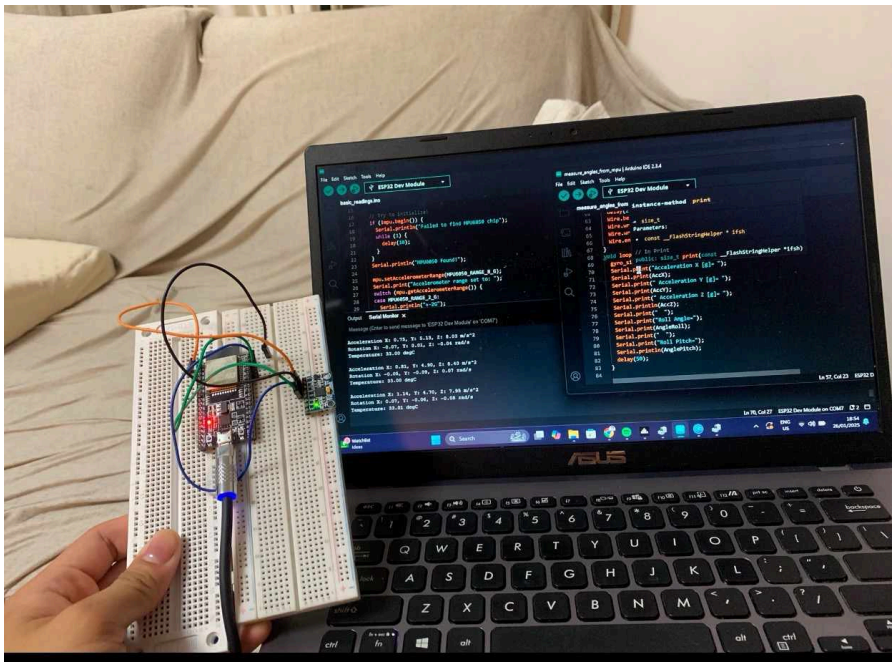
CODING

1. Install the Development Environment

Install Microsoft Visual Studio Code and add the PlatformIO extension. Download the latest firmware from the repository's Releases page.

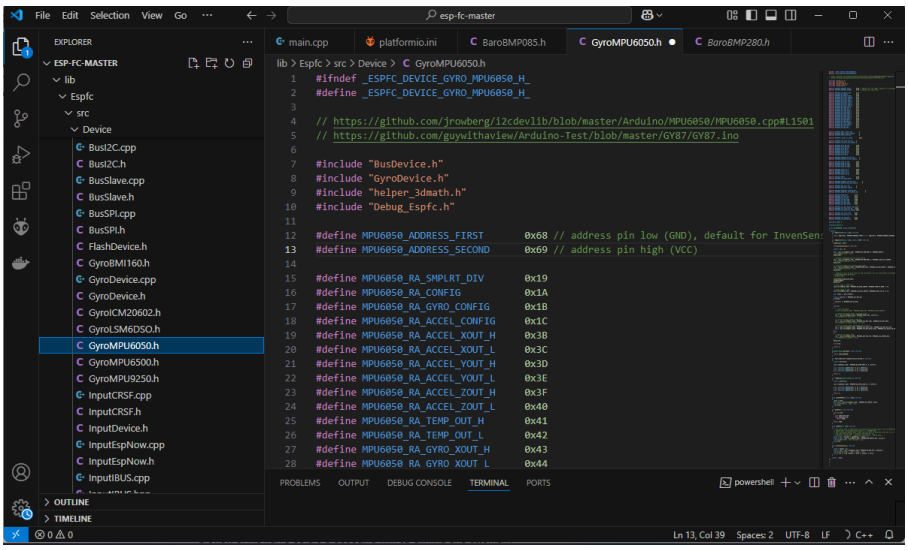
2. Verify Compatibility of ESP32 boards

Ensure your ESP32 model (e.g., ESP32-WROOM, ESP32-S2, ESP32-D1 Mini) is supported.



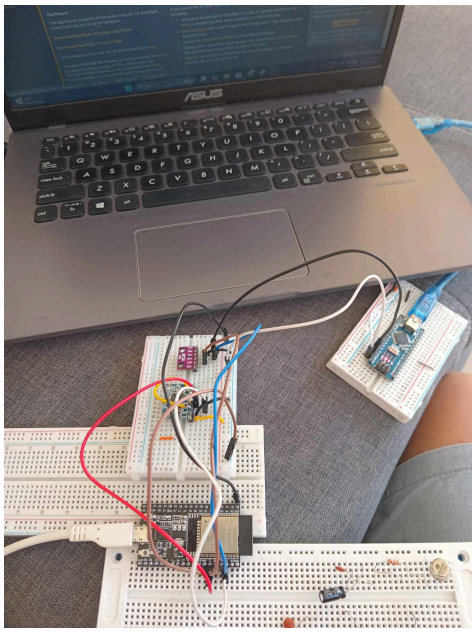
3. Check Sensor Compatibility & Configuration

Confirm sensor compatibility (MPU6050, barometer, magnetometer, etc.). Verify and configure correct I2C addresses in the firmware, modifying code if needed.



4. Flash Firmware to ESP32

Connect the ESP32 via USB, open ESPTool or PlatformIO, select the correct COM Port, and upload firmware at 115200 or 921600 baud rate. Ensure a successful upload and proper boot.



5. Verify Installation

Power cycle the board (connect and disconnect), open a serial monitor, and check system logs to confirm proper operation. If issues arise, verify port settings, baud rate, and connections.

ELECTRONICS

1. Selection of Flight Controller Platform

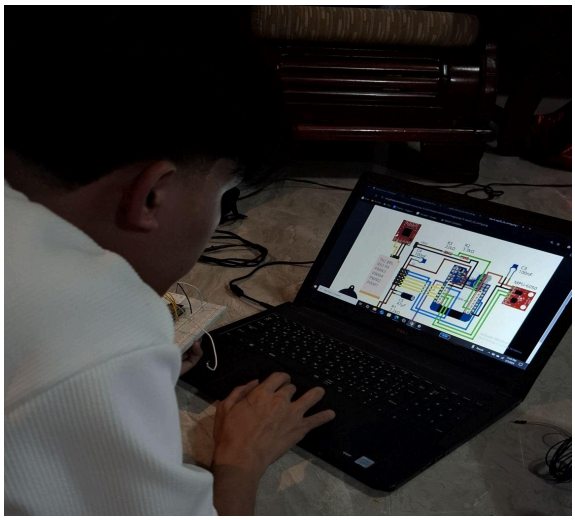
The initial stage in building the drone is to choose an acceptable flight controller platform. Extensive research is being undertaken on the numerous ESP32-based flight controller platforms that are currently available in the market. After considering other choices, the ESP-FC platform was chosen because of its continuous development, active community support, and compatibility with Betaflight Configurator. This option assures that the drone receives the most recent software upgrades and fine-tuning capabilities for flight performance.

2. Component Inventory and Selection

After determining the appropriate flight controller platform, a complete inventory of components is created, which includes critical parts such as sensors, motors, speed controllers, GPS modules, and communication modules. Each component is carefully chosen based on its compatibility with the ESP32 microcontroller and the drone's performance requirements. In addition to technical concerns, the cost of each component is estimated to ensure that the project is affordable without sacrificing quality

3. Circuit Design and Wiring Preparation

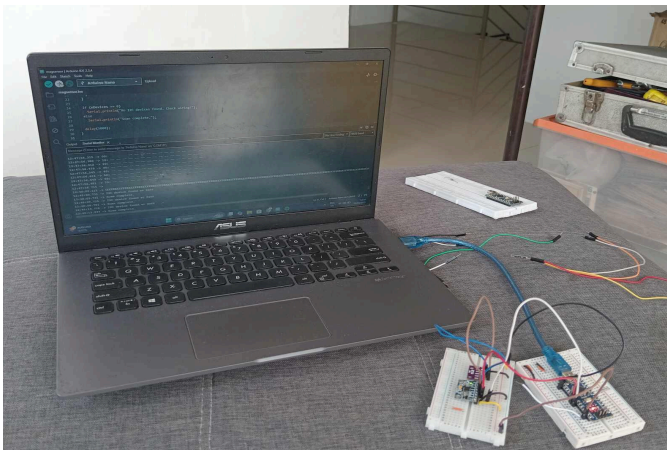
We will need to create diagrams to identify the links between the components we prepared. The circuit design process entails developing a thorough schematic diagram that depicts the interconnections of all electronic components. This schematic shows the power distribution network, signal transmission channels, and sensor interfaces. Circuit design software is used to create realistic representations, ensuring a thorough understanding of the wiring arrangement before starting with hardware construction.



4. Firmware Programming for Microcontroller

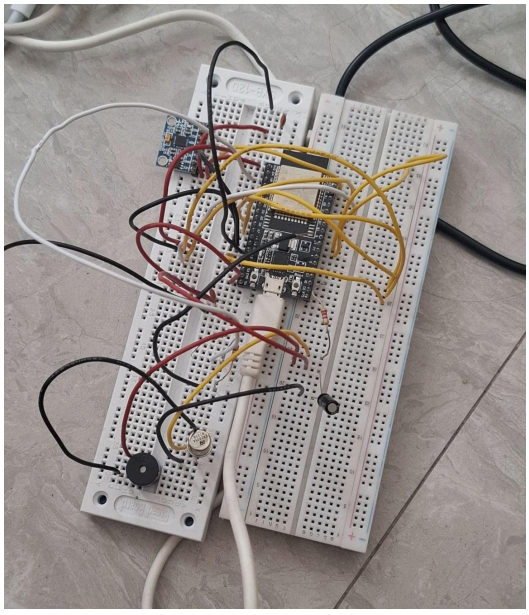
Programming the ESP32 microcontroller is an important stage in the project. The firmware is written in Arduino IDE and Visual C++ to control flying parameters, interpret sensor data, and manage wireless connection. The programming process entails writing, testing, and debugging code to ensure that the microcontroller responds to user commands and performs autonomous operations properly. And sensors attached to the ESP32 microcontroller via I2C communication include the MPU-6050 for gyroscopic and accelerometer data, as well as the BMP280 for altitude measurement. The NRF24 communication

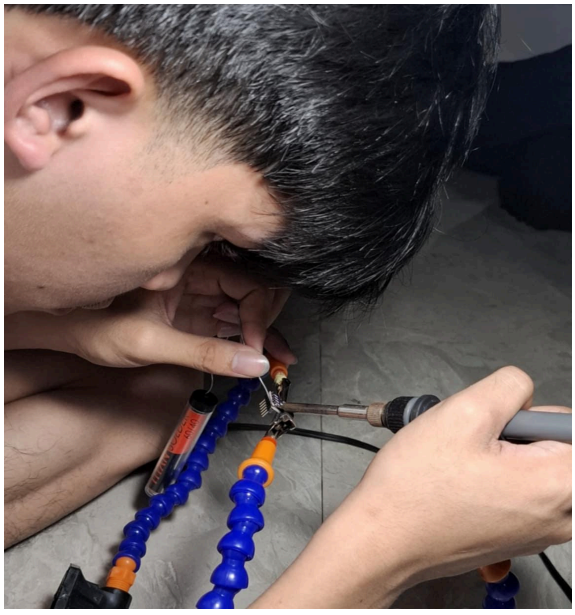
module is used to enable wireless communication between the drone and the remote controller. Sensor calibration and accuracy testing are carried out to ensure trustworthy data capture



5. Electronics Assembly and Initial Motor Integration

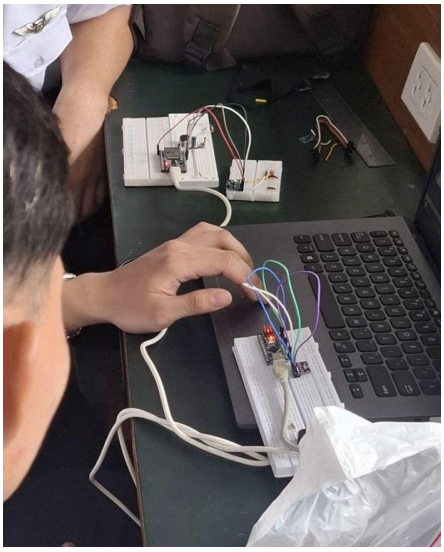
The drone's internal electronics are built by assembling electronic components on a PCB or breadboard. Speed controllers, motors, and power distribution boards, as well as the controller's receiver and transmitter, are firmly installed, and wiring connections are double-checked for alignment. Precision is required throughout the assembling phase to avoid electrical shorts and maintain signal integrity.

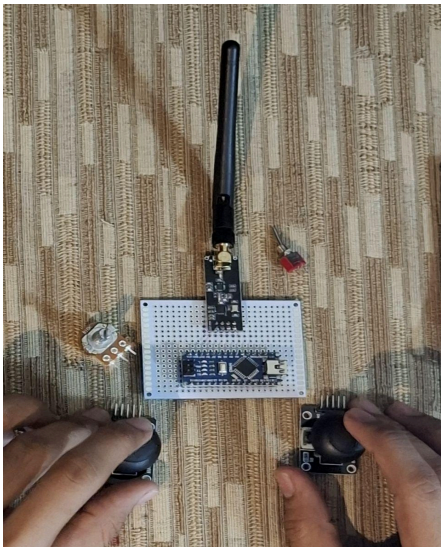




6. Component Testing and Motor Function Verification

Individual components are tested separately to ensure their functionality prior to final integration. Common problems such as wiring faults, voltage incompatibilities, and software defects are detected and fixed. This stage is critical to ensuring that each component works as intended before moving to full assembly.

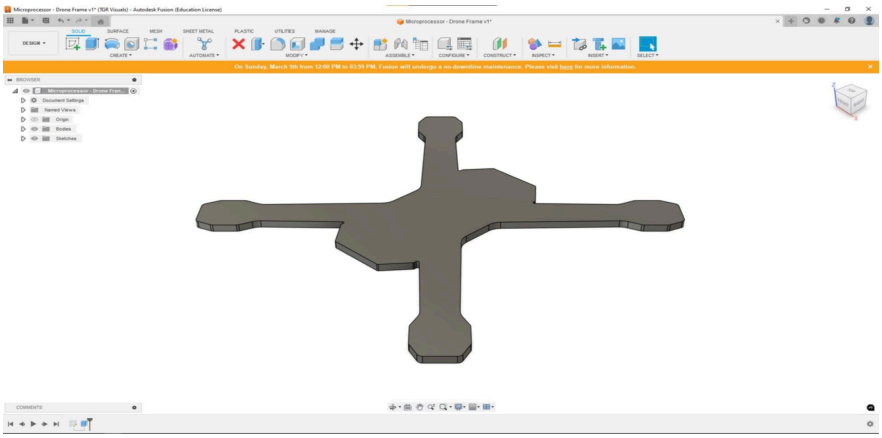




STRUCTURE

1. 3D Modelling Software Selection

Begin by choosing an appropriate 3D modeling software, which is the Autodesk Fusion 360, which includes extensive capabilities for generating complex and precise designs. This software will be the major foundation for designing the drone's structural components.



2. Border Creation for Frame Dimensions

Establish a perimeter outline to define the maximum size of the drone frame, adhering to the specified 1ft x 1ft dimension.

3. Propeller Layout Design

Integrate the propeller arrangement into the design to ensure that they do not conflict with other components. This procedure is critical to preventing collisions

4. Design Reference

Refer to an existing 250mm FPV drone frame as a foundation for the initial design concept. This reference serves as a guide to align the structure with industry standards.

5. Drone Arms Design

Design the drone arms, which extend from the center frame to the motor mounts. These arms must be strong enough to sustain the motors and propellers while avoiding needless weight.

6. Battery Measurement

Measure the dimensions of the Li-Po battery to ensure the frame provides adequate space to securely hold the battery without compromising the drone's balance.

7. Main Plate Design

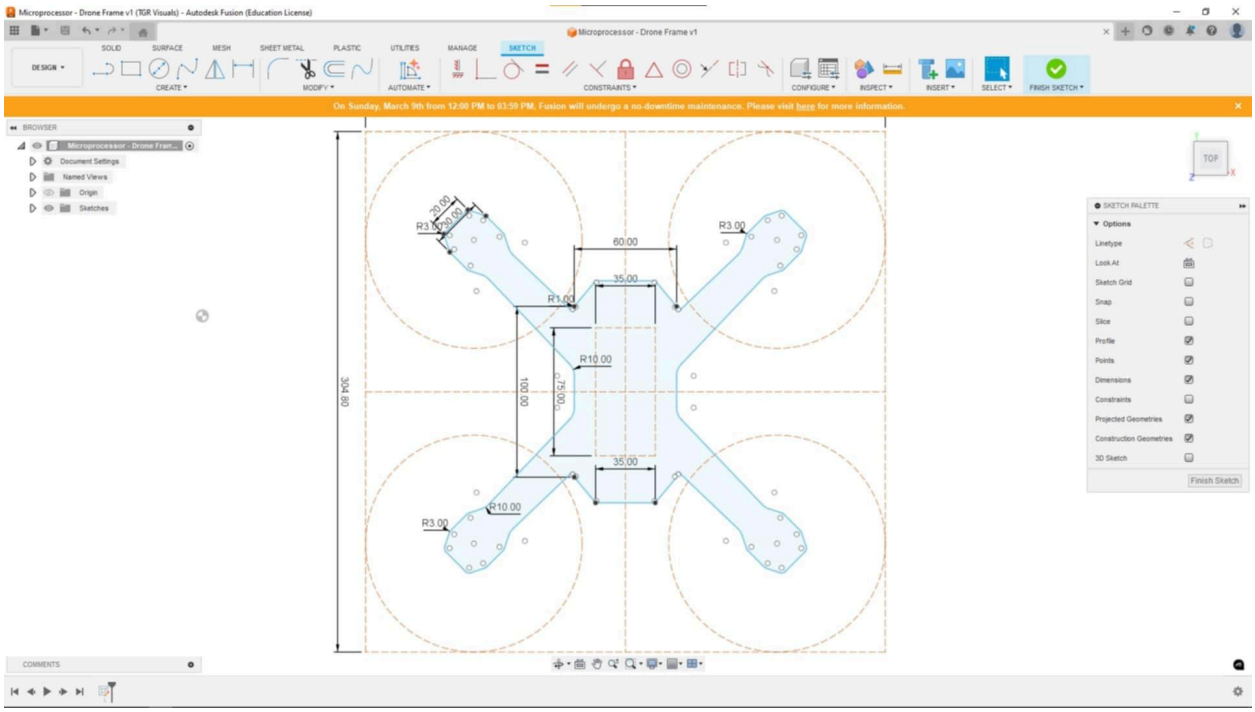
Develop the central main plate of the frame, making it slightly larger than the battery dimensions to offer enough room for mounting additional components such as the flight controller and power distribution board.

8. Arm Thickness Adjustment

Adjust the thickness of the drone arms and merge them with the central plate. This adjustment enhances the frame's durability while maintaining an optimal weight-to-strength ratio.

9. Sketch Refinement

Clean up the initial sketch by removing superfluous lines and refining the design. A simplified layout helps in visualizing the final shape and structure of the frame.



10. 3D Model Conversion

Convert the final sketch into a 3D model by extruding the design. This step provides a realistic preview of the drone frame, allowing for further modifications before 3D printing.

MIDTERM PERIOD

CODING

NRF24 ARDUINO REMOTE CONTROLLER

1. **Install the Development Environment**
    - Install Arduino IDE and ensure board support for ESP32 modules is added.
    - Connect the transmitter and receiver devices (both using ESP32) to your computer via USB.
2. **Upload the Transmitter Code**
    - Open the transmitter code sketch in Arduino IDE.
    - Verify that the code is correctly configured for the transmitter ESP32 model (e.g., ESP32 Dev Module).
    - Select the correct COM port, compile, and upload the transmitter code.
3. **Upload the Receiver Code**
    - Open the receiver code sketch in a new Arduino IDE window.
    - Ensure that the code matches the receiver ESP32 model settings.
    - Select the correct COM port, compile, and upload the receiver code.
4. **Test the Connection Using a Code Tester**
    - After flashing both devices, open the Serial Monitor for both the transmitter and receiver.
    - Input a test signal or use the predefined code tester built into the sketch to send test data from the transmitter.
    - Monitor the receiver's Serial output to verify that it correctly receives and responds to the transmitted test data.

5. **Confirm Successful Pairing**
    - Observe that the receiver successfully acknowledges and outputs the received test signals.
    - If the data appears correctly and in sync between the transmitter and receiver, the connection is verified and functional.

ESP Wi-fi REMOTE CONTROLLER

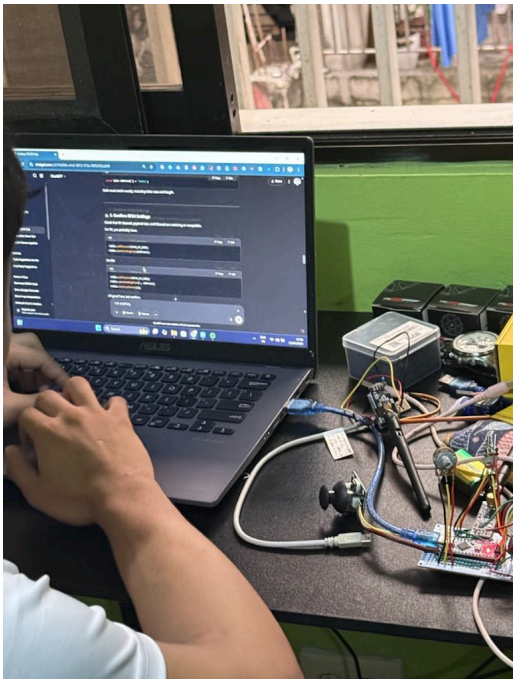
1. **Install the Development Environment**
    - Install Arduino IDE and add the ESP8266 board support URL under File > Preferences.
    - Open Boards Manager, search for ESP8266, and install.
    - Then, install the WebSockets library ("WebSockets by Markus Sattler") from the Library Manager.
2. **Prepare the WiFiPPM Firmware**
    - Download the WiFiPPM sketch from the original Instructables or GitHub source.
    - Open the sketch in Arduino IDE.
    - Check that it uses #include <ESP8266WiFi.h> for Wi-Fi and includes the WebSocket server code.
3. **Verify ESP8266 Board and Configure Settings**
    - Select the correct ESP8266 board under Tools > Board (e.g., NodeMCU 1.0, ESP-12E Module, etc.).
    - Ensure Upload Speed is 115200 baud, Flash Size is set to "4MB (no SPIFFS)", and CPU Frequency is 80 MHz.
4. **Flash Firmware to ESP8266**
    - Connect the ESP8266 module via a 3.3V USB-to-Serial adapter.
    - Select the correct COM port in Arduino IDE.
    - Put the ESP8266 into flash mode (GPIO0 grounded) before uploading.

- Upload the sketch and wait for the "Done Uploading" confirmation.

5. **Verify Setup and Connect**
   - After upload, disconnect GPIO0 from ground and reset the ESP8266.
   - Connect a smartphone or PC to the "WifiPPM" Wi-Fi network.
   - Open http://192.168.4.1 in a browser to access the joystick or touch control interface.
   - Test the PPM output signal by moving the controls.

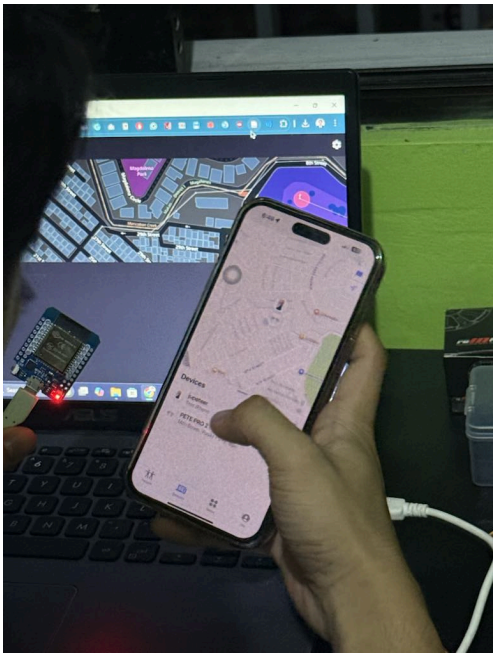
ESP32 AIRTAG

1. **Install the Development Environment**
   - Install Arduino IDE or Visual Studio Code with the PlatformIO extension.
   - Add ESP32 board support by including the ESP32 JSON URL in the board manager settings.
   - Download or clone the macless-haystack repository from GitHub.



2. Verify ESP32 D1 Mini Compatibility

- Ensure that you are using an ESP32 D1 Mini board.
- This board is fully compatible with macless-haystack and ideal due to its small size and built-in WiFi.



3. Configure the Firmware

- Open the macless-haystack project folder.
- In the src directory, check if any settings need adjusting for your ESP32 D1 Mini (such as WiFi channels, scan intervals, or Serial settings).
- Make sure the firmware does not expect extra peripherals unless you add them manually.

4. Flash Firmware to ESP32

- Connect the ESP32 D1 Mini to your computer via USB.
- In Arduino IDE or PlatformIO, select the board type as ESP32 Dev Module or manually configure for ESP32 D1 Mini settings.

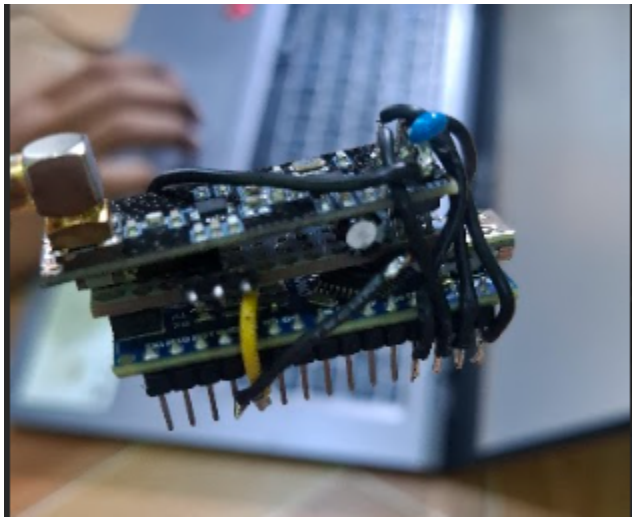
- Select the correct COM port, compile the project, and upload the firmware at 115200 baud or higher.

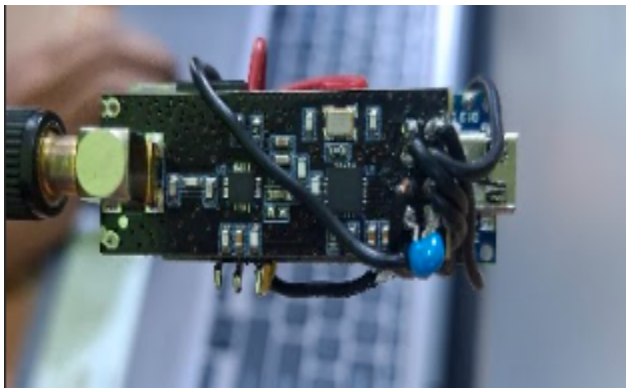
5. **Verify Installation and Functionality**
   - After flashing, open the Serial Monitor and set the baud rate to 115200.
   - Reset the ESP32 D1 Mini if needed.
   - The device will begin scanning nearby WiFi networks and display the results without exposing its own MAC address.

ELECTRONICS

1. Stacked PCB Design for Receiver and Space Optimization

To address the space constraints within the drone's compact frame, a stacked PCB design is created specifically for the receiver module. Instead of laying out the components side by side, the receiver's circuitry is arranged vertically in layers. This organization helps free up internal space for crucial components like ESCs and motor wiring. The stacked design ensures that despite the limited space, motor connections and other critical circuits remain accessible and protected.

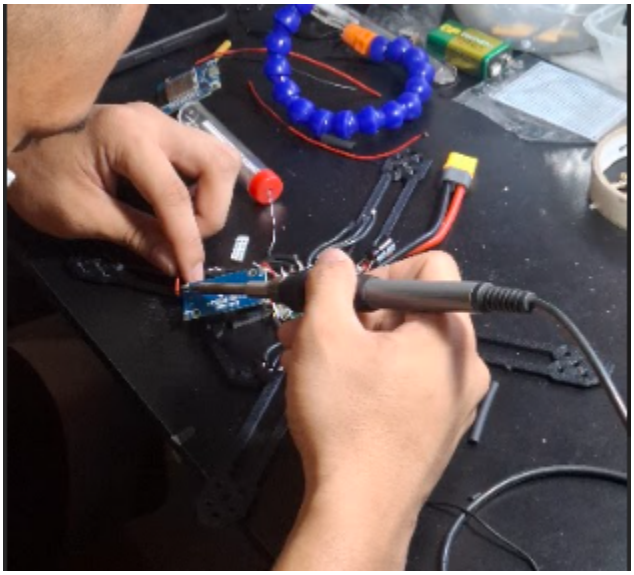


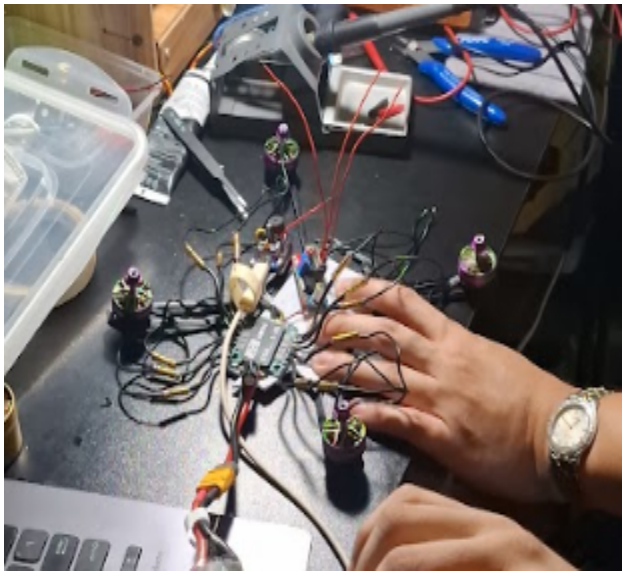


2. Testing of Connection Between ESP32 and ESCs

Once the customized receiver is assembled, it is connected directly to the ESP32 microcontroller. This link is critical because it enables the receiver to decode and relay wireless signals sent by the NRF24 communication module on the remote controller. The receiver forwards throttle, pitch, yaw, and roll inputs, which are interpreted by the ESP32 and translated into control signals for the motors via the ESCs.

Testing of the connection between the ESP32 and ESCs is conducted at this stage to ensure that motor control signals can be sent accurately. Correct wiring and communication protocols are verified to guarantee stable signal transmission and accurate motor behavior.





3. Modification of Buck Converter from 16V to 5V and Addition of Logic Shifter

During the integration phase, slight revisions are made to the transmitter design to accommodate essential components like the battery or power source. The layout is adjusted to maintain a balance between compactness

and functionality. The Buck converter is modified from 16V to 5V to provide a safe and regulated voltage for sensitive electronic parts like the ESP32 and the receiver module. Additionally, a logic level shifter is incorporated to manage communication between devices operating at different voltage levels—specifically 3.3V for ESP32 and 5V for modules like the NRF24. This step ensures reliable and accurate data transmission across components.

4. Connection of Power Supply to the Transmitter

Following the transmitter adjustments, the power supply is connected to the transmitter system. This ensures that the NRF24 module, joystick, and other transmitter electronics are properly powered. Special care is taken to maintain proper voltage levels and avoid overloading or damaging the transmitter circuits.

5. Testing of Motors Using ESP32 and ESCs

In operation, as the user manipulates the thumb joystick on the transmitter, corresponding movement commands are wirelessly transmitted using the NRF24 communication module. These commands are received by the drone's receiver, passed to the ESP32, and then executed by sending signals to the ESCs.

Motor testing using the ESP32 and ESCs is performed to verify that motors spin correctly according to the joystick inputs. This ensures that motor speeds adjust in real-time, allowing smooth and responsive control of the drone's flight.

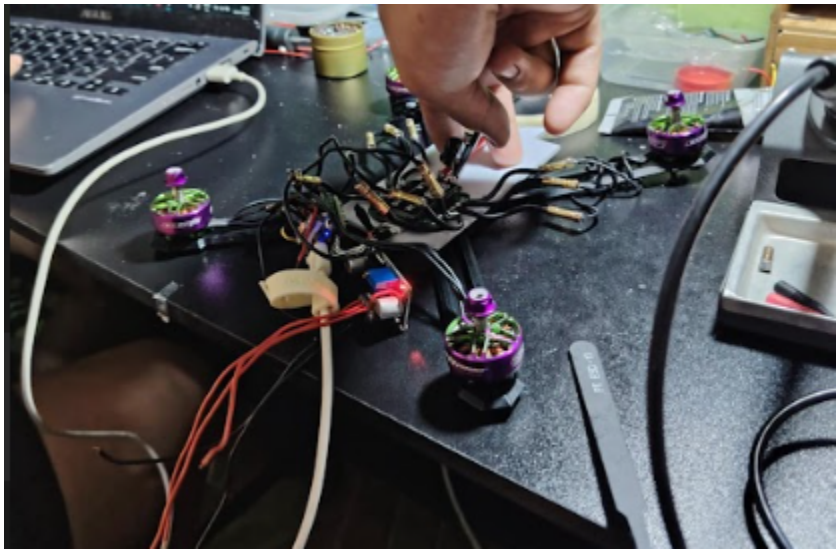
6. Testing of Codes for Both Transmitter and Receiver

At this stage, the software codes uploaded to both the transmitter and the receiver units are tested. This involves checking the accuracy of command transmission (e.g., throttle, pitch, yaw, and roll) and ensuring correct wireless communication between the two modules.

Any bugs, delays, or inconsistencies detected during this testing are debugged and corrected to ensure full synchronization of commands and motor responses.

7. Final Testing and Motor System Validation

Before full integration into the flight-ready drone, comprehensive testing of both the transmitter and receiver systems will be conducted. The motors' responses to joystick movements are carefully observed to ensure that throttle, yaw, pitch, and roll commands are correctly executed. Tests will also confirm stable wireless communication, minimal latency, and consistent motor behavior. Any inconsistencies will be diagnosed and resolved to ensure a stable, safe, and responsive system ready for field operation.



8. Completion of Compact ESP32 Flight Controller and Receiver Designs

Finally, the compact designs of both the ESP32-based flight controller and the Arduino-based NRF24 receiver are completed.

The stacked PCB designs and optimized layouts ensure that all components fit neatly within the drone's frame, maintaining both performance and durability while addressing space and weight constraints.

STRUCTURE

1. Progress started on Drone accessories

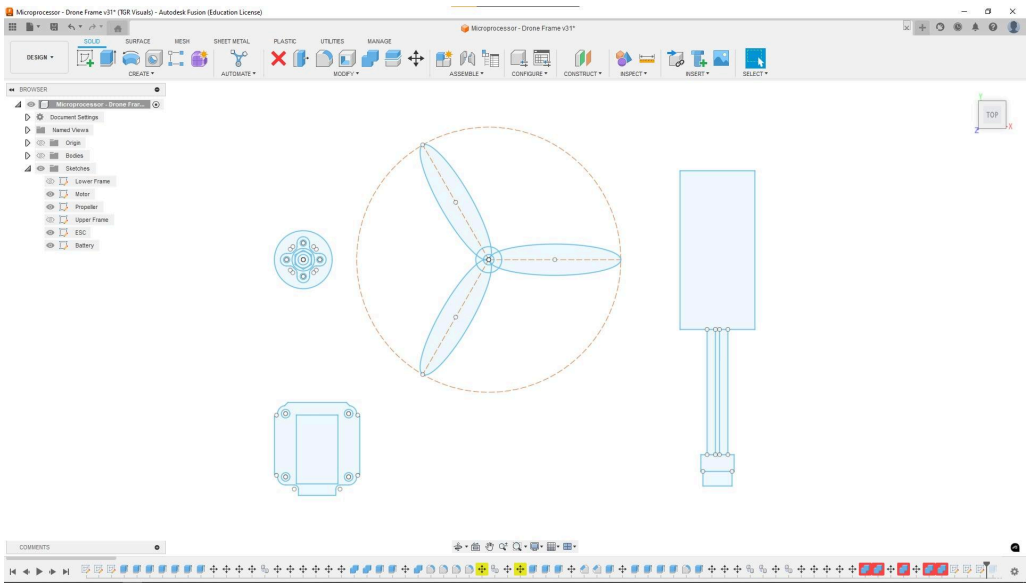
The progress on auxiliary drone accessories such as the helipad had been designed and had started assembly.

2. Measuring of Components

All of the components that will be installed on the frame are measured. A digital caliper was used, allowing for a precise and accurate measurement of the screw holes, LxWxH, and Clearances.

3. Creating a 2D Model of the Components

With the measurements gathered, 2D models of the main components and parts were sketched. This includes: Motors, ESC, Flight Controller, Screws, and Standoffs.



4. Positioning of Components

layout the 2D modeled components within the outline of the initial design of the drone frame and place them to the desired position. While making sure that they have enough clearance from the propellers.

5. Refinement of Drone Frame Design

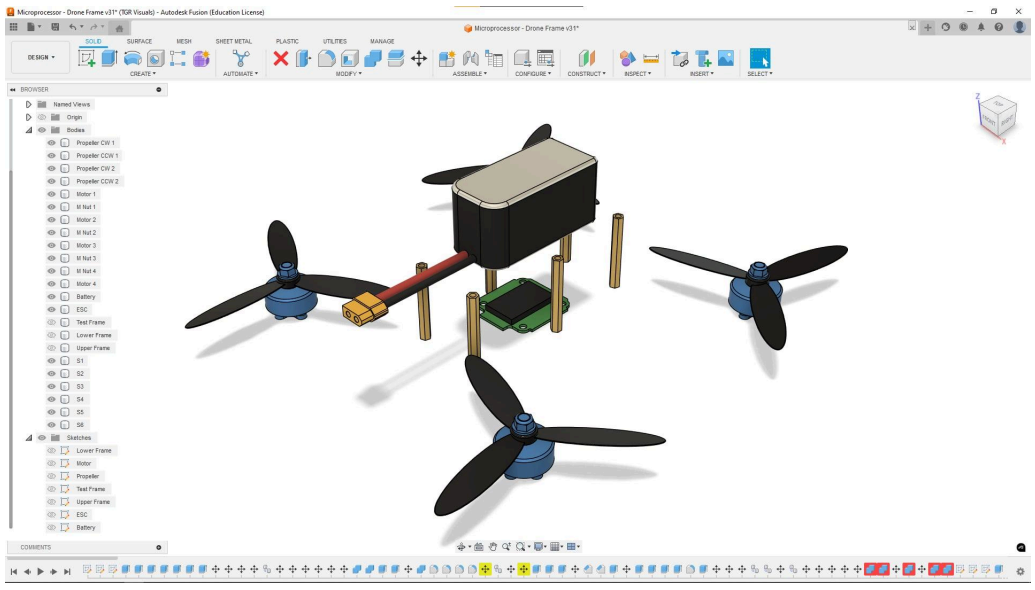
With the components positioned, refine the initial design by extending the main plate of the frame to fit all components into one area. Then adjusting its overall shape to make it more aesthetically pleasing.

6. Adding Mounting Holes and Slots

After finalizing the shape of the frame and positions of the components, add the necessary mounting holes using the measurements taken from before.

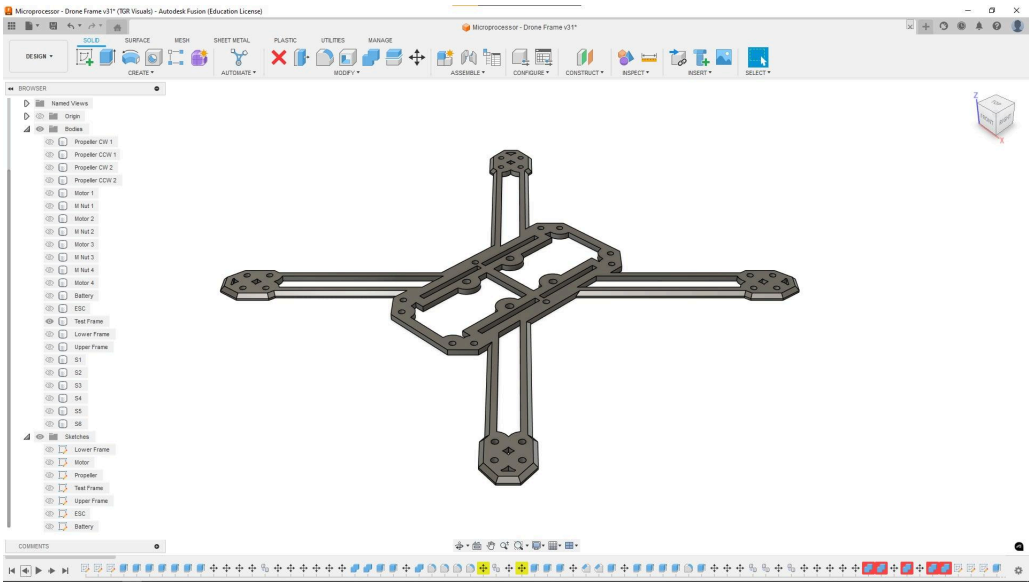
7. 2D to 3D Model Conversion of Components

Convert the 2D model of the components into 3D in order to get a better preview of their gap from each other, especially vertical clearances, and get an idea of how it will look.



8. Designing a Test Frame

To avoid mistakes for the final print of the final design of the drone frame, create a test frame with the mounting holes and slots included.

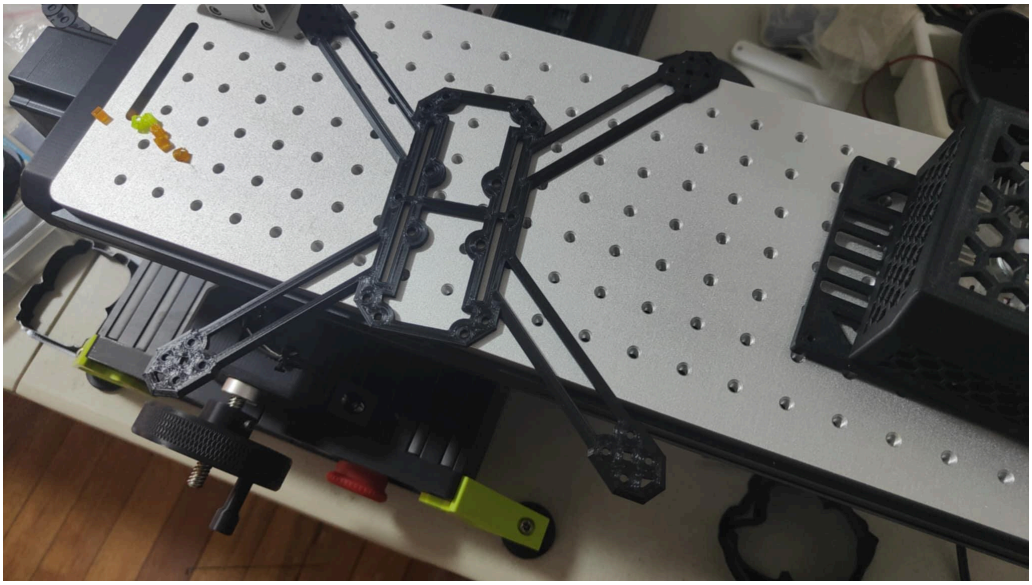


9. Adding Cost Saving Holes/Slots

Since it is just a test piece, durability of the structure is not important. Add holes or negative spaces to areas where nothing will be mounted or installed. This is to reduce the printing cost of the test piece.

10. 3D Printing and Test Fitting

Have the test frame printed using a cheaper filament, PETG, to further bring down the cost. Then once printed, test fit all components and fasteners to get an idea of any adjustments needed.

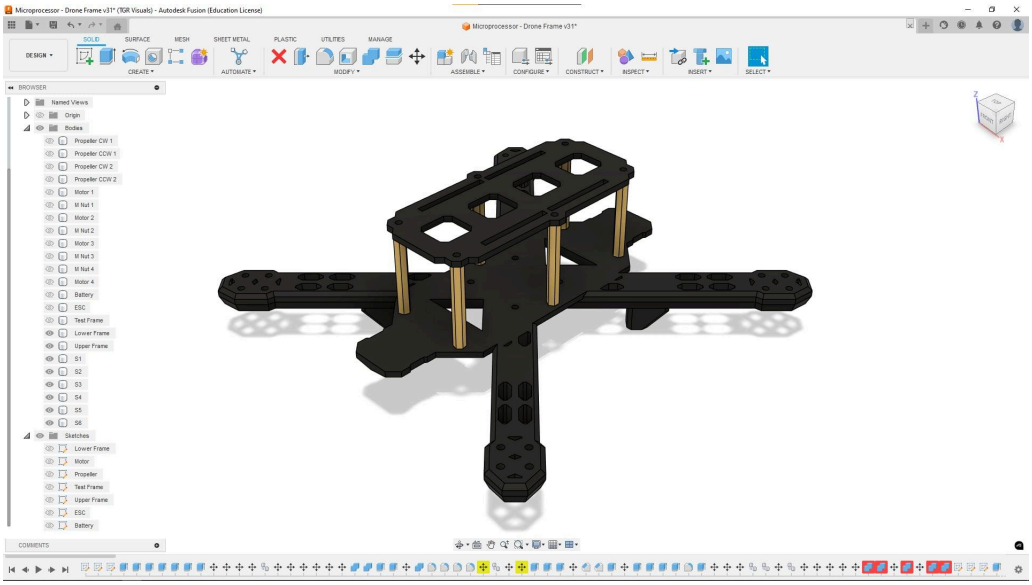


11. Final Adjustments

After gathering information from the test fit, do some final adjustments of measurements of the mounting holes, slots, thickness, etc. to make sure that all components will fit properly for the final design.

12. 2D to 3D Model Conversion of Final Frame Design

Convert the 2D model of the frame into a 3D model while keeping in mind the required thickness in certain areas to ensure durability and stability during flight.



13. Adding Cost and Weight Saving Holes/Slots

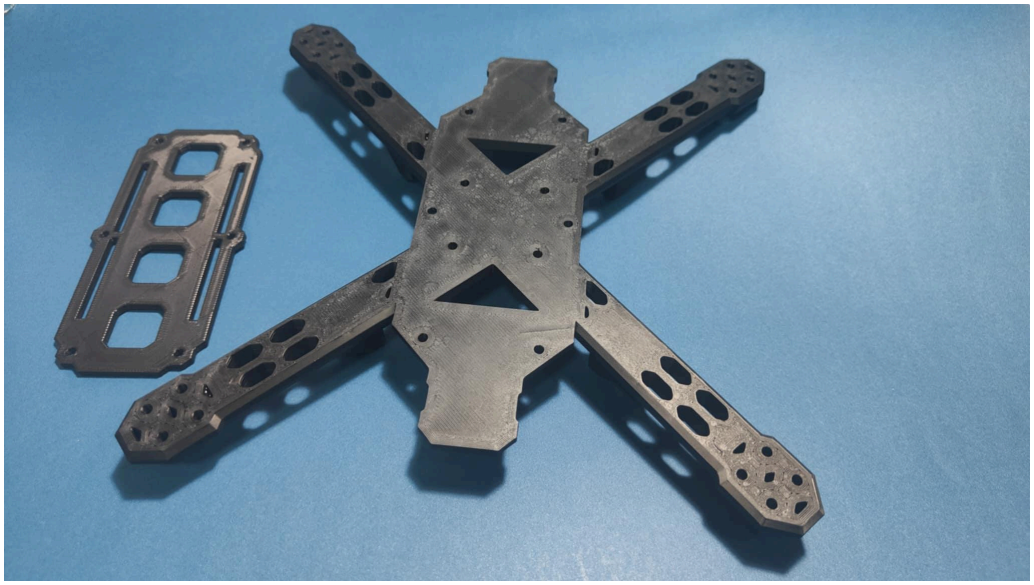
To further reduce the printing cost and the overall weight of the drone frame, place holes and slots to areas where the structural integrity of the entire drone won't be affected.

14. Design Inspection

Before printing, have the final frame design inspected by a professional to ensure that the frame is suitable for flight and is able to achieve all requirements.

15. 3D Printing

Have the final drone frame design 3D printed using ASA filament, which is durable enough for the drone's requirements while keeping the costs low.



VII. TAKEAWAYS AND INSIGHTS

Our first step in making our drone project was to clearly define the scope and objectives. What is the primary purpose of the drone? Will it be used for navigation, surveillance, delivery, or another application? Understanding the end goal will help us determine the necessary features, such as autonomous navigation, obstacle avoidance, or payload capacity. This clarity will guide our decisions regarding hardware, software, and overall design. Additionally, considering the environment in which the drone will operate indoor, outdoor, or both as this will influence sensor selection and navigation algorithms.

Thorough research is a must before starting building our drone project. We studied/reviewed existing drone projects, especially those with similar goals, to learn from their successes and challenges. It also included identifying the key components required, such as the flight controller, motors, propellers, sensors, and power supply and creating a detailed plan that outlines the steps involved, from assembling the hardware to programming the navigation system.

The team selected the ESP-FC platform for its modern support, active community, and compatibility with Betaflight Configurator. They sourced compatible components while managing costs and designed circuits for stable power, signal transmission, and sensor integration.

Programming the ESP32 using Arduino IDE and Visual C++ posed challenges due to incomplete documentation, but they overcame them through troubleshooting and research. This process emphasized the importance of well-supported platforms, careful planning, and persistence in problem-solving.