ECE 445
Spring 2025

Design Document
Project #1: Glove controlled quad-copter

Team Members: Atsi Gupta (atsig2), Aneesh Nagalkar (aneeshn3), Zach Greening (zg29)
TA: Wenjing Song
Professor: Cunjiang Yu

1. Introduction

1.1 Problem

Controlling drones typically requires handheld remote controllers or smartphones, which can feel unintuitive and demand significant practice to master. This steep learning curve limits accessibility for new users and prevents drones from being seamlessly integrated into areas like training, entertainment, or assistive technology. Existing remote-control methods also provide little user feedback and lack robust safety mechanisms, increasing the risk of crashes or improper handling.

1.2 Solution

Our project proposes a wearable gesture-control glove that enables intuitive, ergonomic drone operation. The glove will incorporate IMU and gyroscope sensors to capture the orientation and motion of the user's hand, translating gestures into commands such as forward, backward, strafe, yaw, and stop. An ESP32 microcontroller embedded in the glove will transmit these commands wirelessly to a drone equipped with an ESP32-based flight controller.

To improve upon previous iterations of gesture-control systems, our design will:

- Replace less precise flex sensors with IMUs for more accurate gesture tracking.
- Include a gesture-based emergency shutoff for safety.
- Optionally integrate haptic feedback to communicate drone status to the user (e.g., low battery, weak signal).
- Optionally integrate an ESP32-CAM on the drone to provide basic video feedback, enhancing user situational awareness.

The camera is not essential to the core glove-to-drone control system, but serves as a scalable feature that can expand the project's applications if successfully integrated.

1.3 Visual Aid

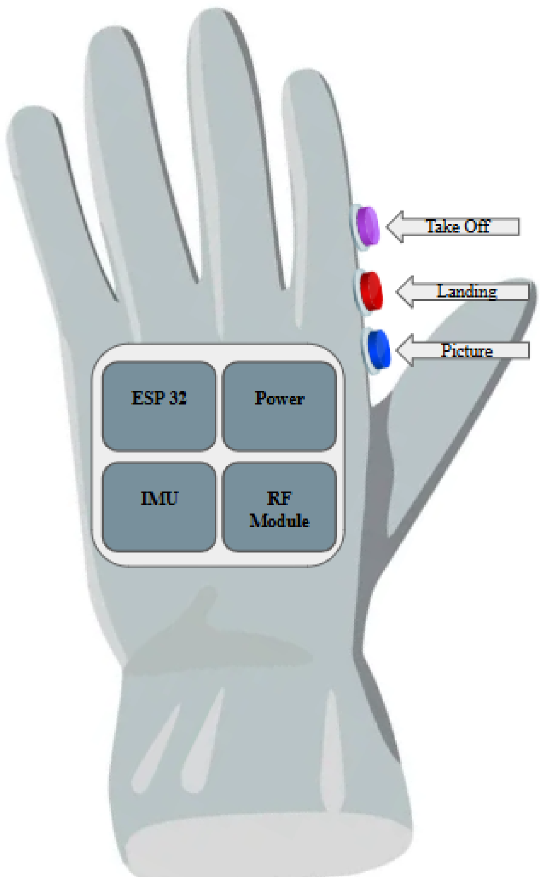

Figure 1: General layout of glove.

1.4 High-Level Requirements List

To demonstrate success, our project must meet the following measurable requirements:

1. The drone responds in real time to glove commands with minimal delay.
2. The buttons make the drone hover or land within 15 seconds of being pressed.
3. Directional commands (forward, back, left, right, up, down) work 80% of the time over 20 trials.
4. *(Stretch goal)* If the camera is integrated, the system should be able to store low-resolution images to the sd card.
5. *(Stretch goal)* Haptic feedback provides clear communication of system status to the user.

2. Design

Block Diagram:

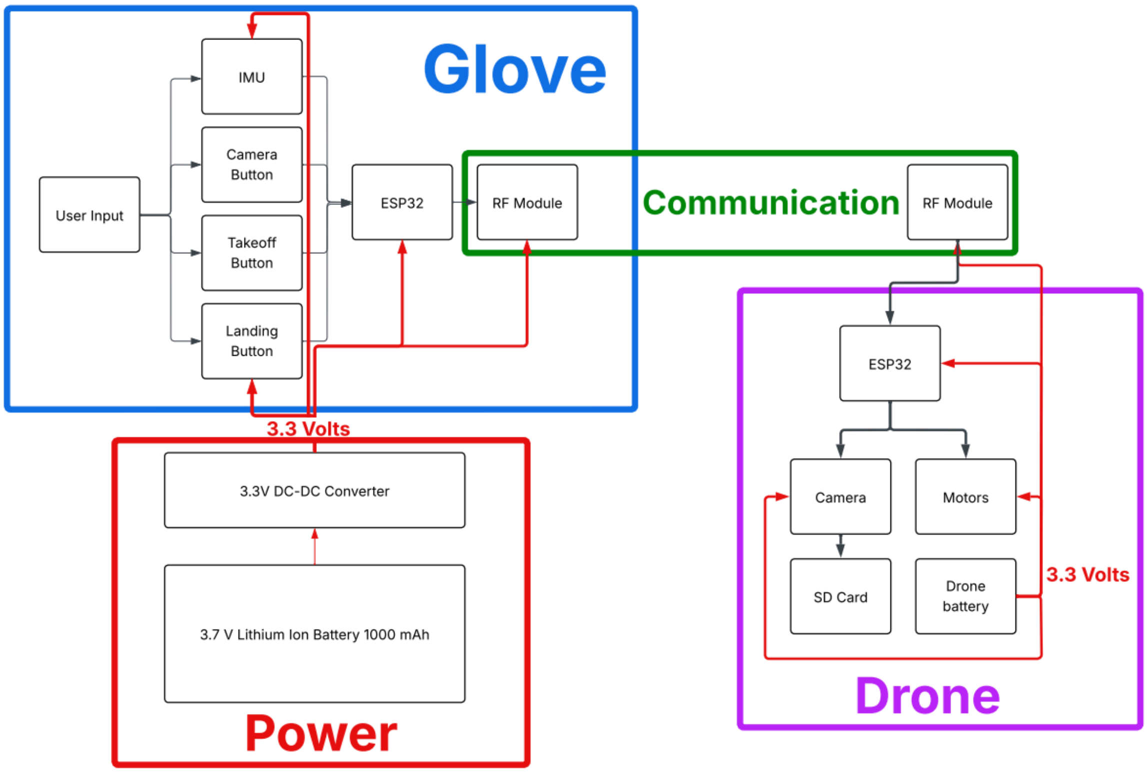

Figure 2: High level Block Diagram

Subsystem 1: Control Directions

Inertial Measurement Unit (IMU)

<u>Model purchased:</u> [MPU-6050 6-DoF Accel]

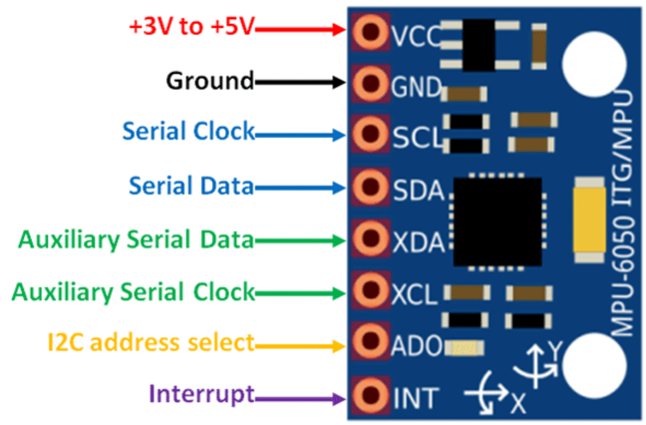

Figure 4: MPU 6050 Physical Layout

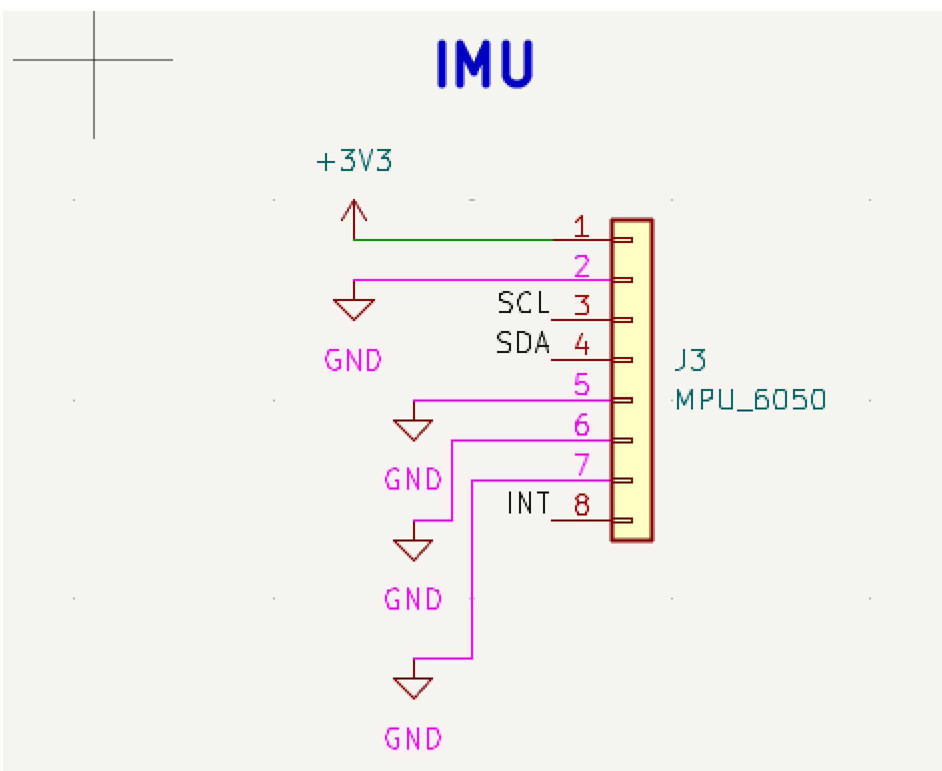

Figure 5: IMU Pinout in Circuit

Features

- Measures six independent axes: X, Y, Z acceleration and X, Y, Z angular velocity
- Sampling rate: up to 1 kHz for both accel and gyro
- Power: ~3.3V

For the Glove:

- ±2g acceleration → very sensitive to small tilts and shifts in hand position (good for mapping tilt to drone motion)
- ±250 °/s gyroscope → very precise at slow/medium rotations (like turning your wrist to steer)

Mapping strategy:

- Definition: Think of the IMU as sitting flat on the back of your hand, with axes (also depicted in Figure 5):
    - X-axis → points forward (toward your fingers)
    - Y-axis → points left (toward your thumb if right hand)
    - Z-axis → points upward (out of the back of your hand)
- For the purposes of our project, we will only be rotating around the the **X and Y axes**

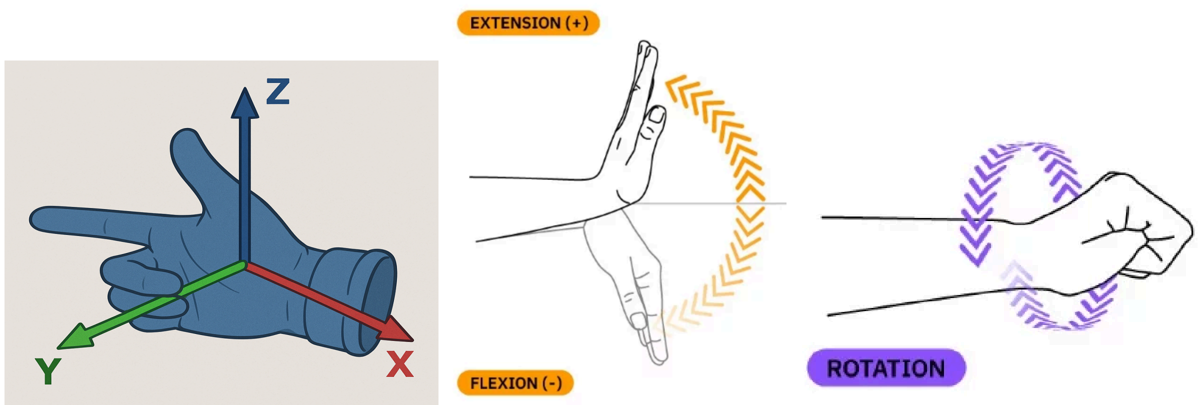

Figure 6: Visual of hand gesture orientation

- This project will be limited to translating the drone along the **Pitch** and **Roll** axes (as shown in Figure 6)
- Takeoff and landing (Thrust) will be handled by the buttons

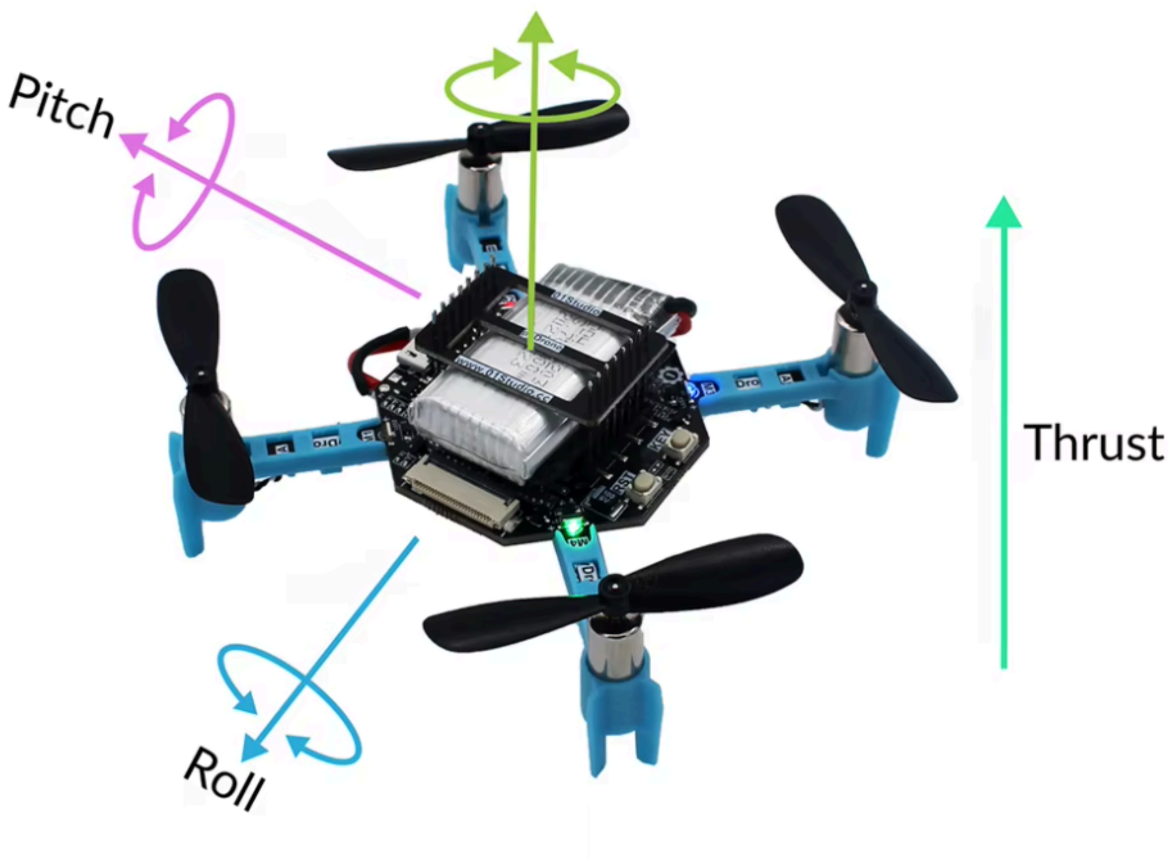

Figure 7: Visual of drone axes and movements

- The mapping of hand gestures to drone movements will be as follows. Refer to Figure 5 and 6 for reference

Table 1: Direction Mapping for Glove

Glove Movement	**Drone Movement**
Clockwise rotation around the glove's Y-axis (flexion)	Drone moves in the positive direction of the Roll axis (Y-axis)
Counterclockwise rotation around the glove's Y-axis (extension)	Drone moves in negative direction of the Roll axis (Y-axis)
Clockwise rotation around the glove's X-axis	Drone moves in the positive direction of the Pitch axis (X-axis)
Counterclockwise rotation around the glove's X-axis	Drone moves in the negative direction of the Pitch axis (X-axis)

ESP32

Model purchased:
https://www.digikey.com/en/products/detail/espressif-systems/ESP32-S3-WROOM-1-N16/16162647

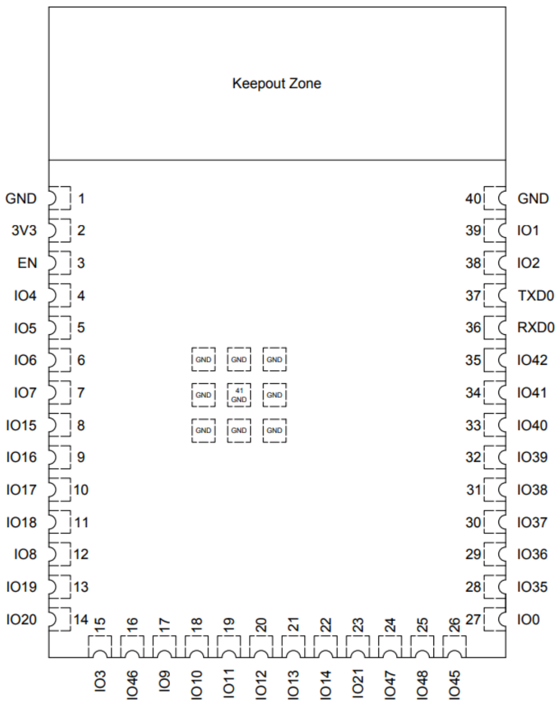

Figure 8: ESP32 pinout

Figure 9: ESP32 Physical Layout

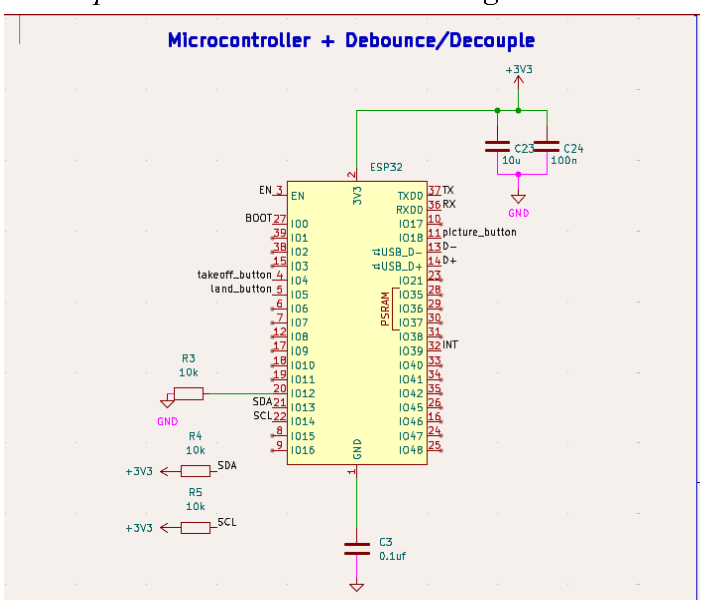

Figure 10: ESP32 Pinout in Circuit

Features

- Brain of the glove system to coordinate all operations between sensors and drone
- Collects real-time motion data from the IMU over the I^2C interface
- Sends data to over WIFI to ESP32 on the drone
- Programmed micropython through Thonny
- Multiple GPIO pins allow integration of buttons for user interaction

Milestones

- I2C scan: Confirm ESP32 detects MPU-6050 at address 0x68
- Raw Data Read: Use library MPU6050_tockn to print IMU values to Serial Monitor
- Calibration: determine gyro offsets while sensor is still
- Gesture Mapping: Map pitch and roll into commands
- Communication to Drone: Send packets wirelessly via UDP

2.2.3 Buttons

Model purchased:
https://www.digikey.com/en/products/detail/te-connectivity-alcoswitch-switches/1825910-6/1632536

Figure 11: Button Physical Layout

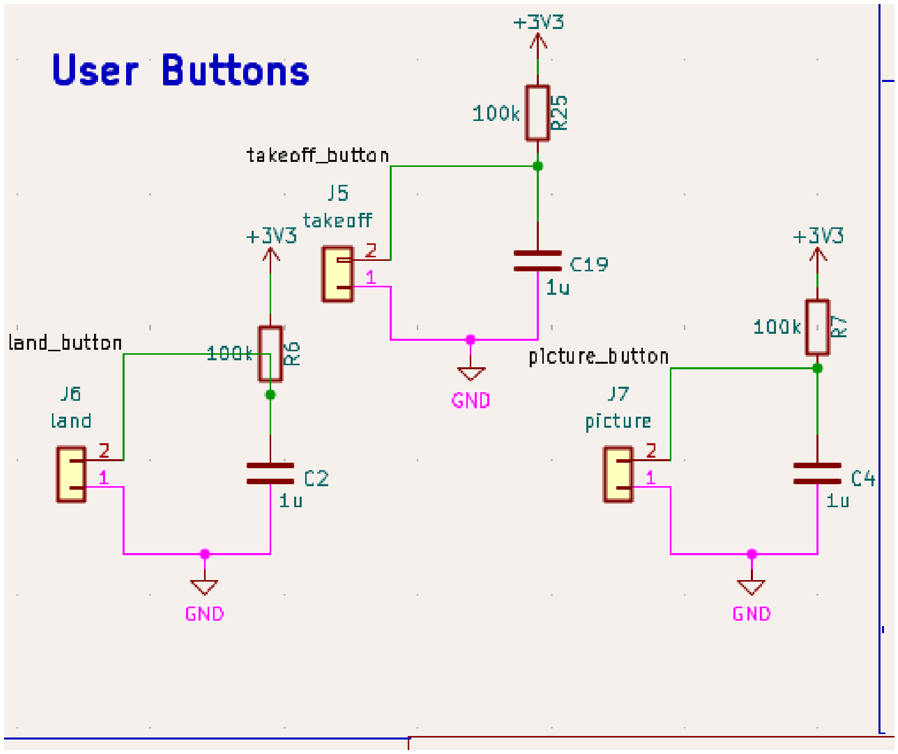

Figure 12: Debounced Buttons in Circuit

<u>Features</u>

- Simple pushdown button switch

<u>Purpose</u>

- Total of 3 buttons required for attaching onto the glove
    - 1 button for takeoff of the drone to make it hover 5 feet into the air
    - 1 button for landing of the drone to make it slowly descend and turn off the rotors
    - 1 button for taking pictures

<u>Wiring</u>

- These buttons will be wired to the GPIO pins of the ESP32
- Configure the GPIO as INPUT_PULLUP
- The ESP32 holds the pin at 3.3 V internally.
- Pressing the button shorts the pin to GND, pulling it LOW and creating a clean digital signal.

Table 2: Requirements and Verification for Control Directions

Requirements	**Verification**
Press registers a single event with debounce ≥10 ms	Capture button waveform + firmware logs for 20 different presses per button

Buttons use INPUT_PULLUP; idle = HIGH (~3.3 V), pressed = LOW (≤0.4 V)	Measure GPIO response using ESP32 output to confirm functionality
Ensure it can tolerate multiple button presses simultaneously in case of accident	Press multiple buttons at the same time and observe behavior (should do nothing)

Subsystem 2: Drone + Flight Controller

Model purchased:
https://www.aliexpress.us/item/3256808919448022.html?gatewayAdapt=glo2usa4itemAdapt

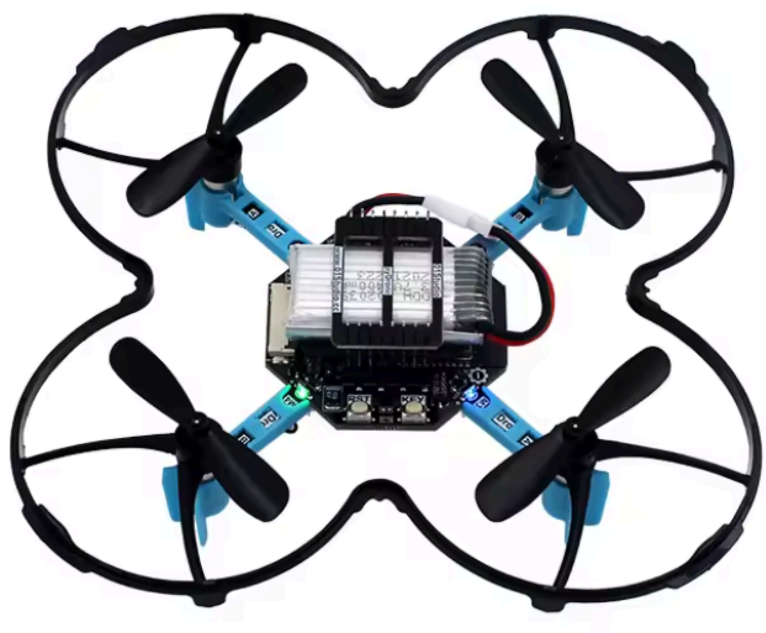

Figure 13: Pydrone Physical Layout

<u>Features:</u>

- Built on the ESP32-S3 platform, making it natively compatible with our glove's ESP32 transmitter.
- Exposed pins allow potential wiring of an ESP32-CAM module or other sensors.
- Actively maintained open-source codebase, which should simplify firmware edits for custom control schemes.

<u>Design Considerations:</u>

- The PyDrone only supports MicroPython code. Tests have been conducted to ensure that MicroPython can be successfully flashed onto the ESP32 module in the glove, allowing it to later communicate with the drone
- This drone is not meant to fly very far distances but rather respond to glove commands within a 50 meter radius

Table 3: Requirements and Verification for Drone + Flight Controller

Requirements	Verification
Must be configurable to receive data via WI-FI using onboard ESP32	Verify that the drone can receive data via WI-FI from the ESP32 via debug logs from Pydrone
Must support real-time command reception from the glove with latency <200 ms	Measure command latency during test flights either visually or via program
Must provide basic flight stabilization (hovering, pitch, roll, yaw, throttle)	Validate that the drone can hover for at least 5 minutes

Subsystem 3: Communication

The glove and drone will communicate directly over Wi-Fi, leveraging the built-in Wi-Fi functionality of the ESP32 modules used on both ends. This eliminates the need for an external RF transceiver and simplifies the hardware design.

Purpose

- Provides a dedicated Wi-Fi link between the glove's ESP32 and the drone's ESP32 flight controller.
- Control commands will be sent as UDP packets for low-latency transmission.
- Leaves GPIO pins available for additional peripherals such as buttons, haptic feedback, or the optional camera.

Design Considerations

- UDP will be chosen over TCP to minimize retransmission delays, though packet loss handling will be addressed in software.
- Wi-Fi traffic will be limited to control packets only if the optional ESP32-CAM is integrated, to ensure flight commands remain responsive.

- Both glove and drone ESP32 boards must be configured to connect on the same Wi-Fi network, or one can act as a soft access point (AP) while the other connects as a station.

Table 4: Requirements and Verification for Communication

Requirements	Verification
Frequency band: 2.4 GHz Wi-Fi (802.11 b/g/n)	Stress test by introducing background Wi-Fi traffic to ensure drone commands remain prioritized.
Typical throughput: up to 65 Mbps, but only a few kbps required for control packets	Verify, through a different receiver (windows machine), that the glove is correctly sending UDP packets to the correct socket, with the correct wifi network.
Range: 30–50 m indoors, up to 100 m line-of-sight outdoors (depending on antenna quality and environment)	Verify that we can reliably control the drone from 30-50 meters away.
Interface: MicroPython socket libraries	All of this must be done using libraries native to MicroPython, or lightweight programs that can be flashed to the ESP32.

Subsystem 4: Power + DC-DC Regulator

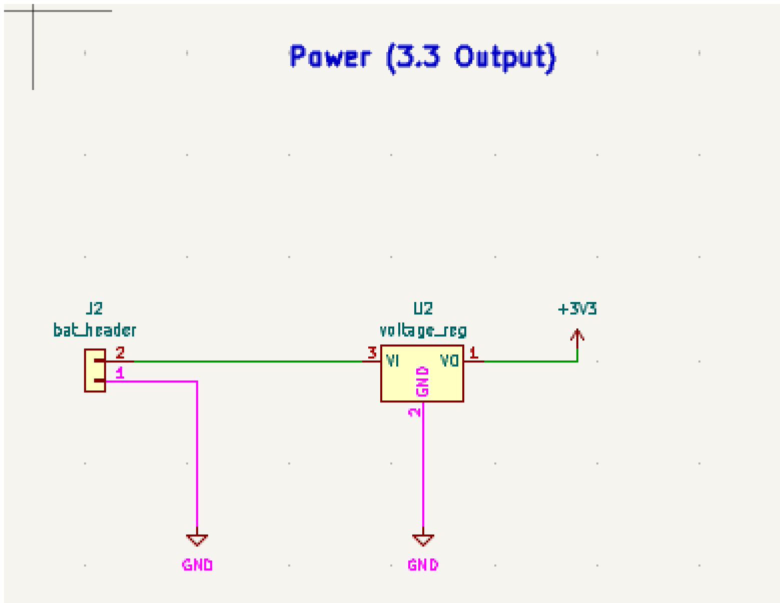

Figure 14: Power in Circuit

Features:
The glove needs a controlled 3.3 V supply to power the ESP32, IMU, and three buttons. We chose a single-cell 3.7 V, 1000 mAh Li-Po with a 3.7 V to 3.3 V converter (1 A).

Key Requirements:
Provide stable 3.3 V power; supply the expected continuous and burst currents; include Li-Po protection (overcharge/overdischarge/short).

Components:
Battery: 3.7 V, 1000 mAh Li-Po.
Regulator: 3.7 V to 3.3 V converter, rated at 1 A.

Calculations:
Component current estimates (conservative):

ESP32: active = 0.200 A (datasheet), peak 0.300 A
IMU: 0.004 A
Buttons: 0.001 A.
Typical total current: 0.200 + 0.004 + 0.001 = **0.205 A**.
Worst continuous estimate (ESP32 peak): 0.300 + 0.004 + 0.001 = **0.315 A**.
Output power at 3.3 V: 3.3 V × 0.205 A = **0.6765 W**.
P_worst = 3.3 V × 0.315 A = **1.0395 W**.

Battery energy: 3.7 V × 1.000 Ah = **3.7 Wh**.
Efficiency = 0.90 (estimate)
Typical battery draw: 0.6765 W / (3.7 V × 0.90) = 0.6765 / 3.33 = **0.2031 A**.
Worst battery draw: 1.4025 W / 3.33 = **0.312 A**.
Typical runtime = 1.000 / 0.2031 = **4.92 h**.
Worst runtime = 1.000 / 0.312 = **3.20 h**.
Apply conservative usable capacity (80%): typical = **3.93 h**, worst = **2.56 h**. These figures justify the 1000 mAh choice. Even under heavy continuous load the glove can still operate for almost two hours which will more than suffice.

Table 5: Requirements and Verification for Power + DC-DC Regulator

Requirements	Verification
The power system must provide a stable, 3.3V output to the entire circuit.	Verify 3.3 V under idle, typical, and peak loads using oscilloscope
The power system must operate for more than an hour before needing to recharge.	Measure real-world runtime to ensure >1 hour per charge

3. Tolerance Analysis

The most critical and challenging component of our project is the communication subsystem between the ESP32 on the glove and the ESP32 on the drone. The system's success relies heavily on the reliability and latency of this connection. If the communication link fails or experiences excessive delay, the glove's control inputs will not translate accurately to the drone's movements, rendering the design ineffective. To address this, we have proposed two possible forms of communication: (1) configuring the glove's ESP32 as a Wi-Fi hotspot that the drone connects to directly, or (2) using an external hotspot that both devices connect through. The first option is simpler and requires no additional hardware, but the range and reliability may be limited by the ESP32's onboard antenna. The second option offers potentially greater range and stability but adds complexity and more potential failure points due to the extra hardware.

To determine whether either design can meet performance requirements, we analyzed the communication link mathematically. Using a free-space path loss (FSPL) model for 2.4 GHz Wi-Fi, the loss in dB is given by:

$$FSPL(dB) = 20log10(dkm) + 20log10(fMHz) + 32.44\text{FSPL(dB)} = 20\log_{10}(d_{\text{km}}) + 20\log_{10}(f_{\text{MHz}}) + 32.44FSPL(dB) = 20log10(dkm) + 20log10(fMHz) + 32.44$$

where dkm is the distance in kilometers and fMHz is the operating frequency in megahertz [1]. For 2.4 GHz, this yields path losses of approximately 60 dB at 10 m, 74 dB at 50 m, and 80 dB at 100 m. According to the datasheet, the ESP32 transmits at 20 dBm with 0 dBi antenna gain. The received power levels would be approximately -42 dBm at 10 m, -56 dBm at 50 m, and -62 dBm at 100 m. Since most Wi-Fi receivers on ESP32 modules can reliably decode signals down to around -72 dBm [2], the system should maintain a stable link out to roughly 150–200 meters in free space. In real-world conditions with body interference, drone orientation changes, and multipath reflections, we expect a practical range closer to 50–100 meters. Adding a 10–20 dB fade margin ensures reliable operation under realistic conditions [3].

Our analysis indicates that using the glove's ESP32 as a hotspot will likely be sufficient for short-range, line-of-sight operation. However, the human body and environmental obstacles can significantly change the signal, particularly when the glove is turned away from the drone. Because of the level of uncertainty we have with the transmission of this signal, we have chosen it to be the subject of our tolerance analysis.

4. Cost and Schedule

4.1 Bill Of Materials

Item / Description	**Qty**	**Part # (or Model)**	**Key Specs**	**Vendor**	**Link**
Programmable ESP32-S3 Drone (PyDrone)	1	PyDrone	ESP32-S3 FC, exposed GPIO, open-source	rcdrone.top	https://rcdrone.top/products/pydrone-python-programming-drone?_pos=1&_psq=pydr&_ss=e&_v=1.0
9 V Alkaline Battery	1	–	9 V primary, bench-only	Amazon	link

ESP32 Module (for PCB)	1	ESP32-WROOM-32E-N4	Dual-core, 4 MB Flash, 2.4 GHz Wi-Fi	ECE Supply Center	–
ESP32 Dev Board (DevKitC/HiLetgo)	1	DevKitC / ESP-WROOM-32 Dev	USB-UART onboard, 3.3 V	Amazon	https://www.amazon.com/AITRIP-ESP-WROOM-32-Development-Microcontroller-Integrated/dp/B0CR5Y2JVD
IMU	1	GY-521 (MPU-6050)	6-DoF, I²C, 3.3 V	Amazon	https://www.amazon.com/HiLetgo-MPU-6050-Accelerometer-Gyroscope-Converter/dp/B078SS8NQV
Momentary Pushbutton	3 (+2 spare)			ECE Supply Center	–
3.3 V Regulator	1	LP2950	3.3 V @ 1 A, Vin 4–11 V	ECE Supply Center	–
USB-to-UART Adapter	1		3.3 V logic, 6-pin header	ECE Supply Center	–
Debounce RC + Series + etc.	per button	10 kΩ, 0.1 µF, 100 Ω	Active-LOW pull-up + RC	ECE Supply Center	–

4.2 Schedule

Week of 9/23 Tasks

- Zach: Finish KiCAD schematic of PCB.
- Atsi: Order remaining breadboard + PCB components.
- Aneesh: Set up initial ESP32 + IMU test code on breadboard.

Week of 9/30 Tasks

- All: Attend 10/3 PCB Review and log feedback.

- Zach + Aneesh: Add buttons to breadboard and program a visual output
- Atsi: Draft of Design Document sections

Week of 10/7 Tasks

- All: Submit teamwork evaluation + finalize PCB edits.
- All: Oversee PCB order submission (10/6).
- All: Prepare and run Breadboard Demo 1.
- All: Complete Design Document

Week of 10/14 Tasks

- Zach: Integrate PCB edits into KiCAD for Round 2.
- Atsi: Assemble and test the drone
- Aneesh: Begin communication test between breadboard and drone.

Week of 10/21 Tasks

- Zach: Document results of drone communication tests.
- Atsi + Aneesh: Update breadboard wiring based on Demo 1 feedback.
- All: Prepare for Breadboard Demo 2 (10/28).

Week of 10/28 Tasks

- All: Start writing Final Paper introduction.
- All: Begin creating Final Presentation slides.
- All: Conduct Breadboard Demo 2 and log results.

Week of 11/4 Tasks

- All: Order PCB Round 3 if necessary after edits.
- All: Continue debugging
- All: Document debugging progress in lab notebook.

Week of 11/11 Tasks

- All: Order PCB Round 4 if necessary and check for design stability.
- Atsi: Validate drone + glove communication on updated hardware.
- All: Update final paper notes.

Week of 11/18 Tasks

- All: Run Mock Demo (11/18) and collect TA feedback.
- All: Edit Final Presentation slides with demo results.

- All: Continue expanding Final Paper draft.

Week of 12/2 Tasks

- All: Run Final Demo (12/1), Deliver Final Presentation (12/8), Submit Final Paper (12/10).
- All: Submit Lab Notebook (12/11)

5. Ethics and Safety

Our project follows the IEEE and ACM Codes of Ethics, prioritizing safety, honesty, and responsible design. Drones raise ethical concerns related to misuse, privacy, and airspace regulations. To address this, we will limit our system to hobbyist-level drones, comply with FAA rules (flying under 400 ft in uncontrolled airspace), and obtain approval before campus test flights.

Electrical Safety: All glove-mounted circuits will be insulated and tested to prevent shorts. Li-Po batteries will follow IEEE battery safety standards, using proper charging, protection circuitry, and enclosures to reduce risks of overheating or puncture.

Mechanical Safety: Propellers will be guarded, and flights limited to controlled test areas. A gesture-based emergency shutoff ensures immediate motor disablement in unsafe conditions.

Wireless Safety: ESP32 Wi-Fi communication will be tested for reliability. A fail-safe mode will cut motors if signals are lost.

Lab Safety: Work will follow UIUC lab policies and OSHA guidelines, including PPE use, safe soldering practices, and risk assessments during flight tests.

6. References

[1] T. S. Rappaport, *Wireless Communications: Principles and Practice*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2002.
[2] Espressif Systems, “ESP32 Series Datasheet,” Version 3.9, 2024. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
[3] M. A. Khan and S. W. Kim, “Performance analysis of 2.4 GHz wireless communication under varying environmental conditions,” *IEEE Access*, vol. 9, pp. 14201–14210, 2021.
[4] IEEE Std 802.11-2020, *IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements*, IEEE, 2020.

[5] D. Halperin, B. Greenstein, A. Sheth, and D. Wetherall, “Demystifying 802.11n power consumption,” in *Proc. USENIX HotPower Workshop*, Berkeley, CA, USA, 2010.
[6] Espressif Systems, “ESP-NOW: Low Power Wireless Communication Protocol,” Technical Documentation, 2023. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_now.html
[7] IEEE. "IEEE Code of Ethics." 2023. https://www.ieee.org/about/corporate/governance/p7-8.html
[8] ACM. "ACM Code of Ethics and Professional Conduct." 2023. https://www.acm.org/code-of-ethics
[9] OSHA. "Occupational Safety and Health Administration Regulations." 2023. https://www.osha.gov/
[10] NFPA. "National Fire Protection Association Electrical Safety Standards." 2023. https://www.nfpa.org/ University of Illinois.
[11] Division of Research Safety." 2023. https://www.drs.illinois.edu/
[12] “ESP32-S3-WROOM-1 ESP32-S3-WROOM-1U Datasheet 2.4 GHz Wi-Fi (802.11 b/g/n) and Bluetooth ® 5 (LE) module Built around ESP32-S3 series of SoCs, Xtensa ® dual-core 32-bit LX7 microprocessor Flash up to 16 MB, PSRAM up to 8 MB 36 GPIOs, rich set of peripherals On-board PCB antenna.” Available: https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf