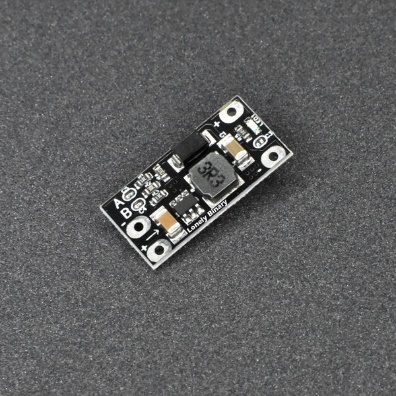
**Project Report: Frequency Detection Using ESP32 and INMP Microphone**

**1. Project Overview**

The project involves the development of a frequency detection system using an ESP32 microcontroller, an INMP microphone, and supporting components. The system detects sound frequencies in the range of 0 to 10 kHz, computes their Fast Fourier Transform (FFT), and identifies specific frequencies in real-time. This has potential applications in sound-based alarms and monitoring systems.

**2. Hardware Components**

1. **ESP32 Microcontroller**:
   * Provides computational capabilities and Wi-Fi connectivity.
   * Operates at 3.3V but requires a 5V power supply, supported by a boost converter.
2. **INMP Microphone**:
   * Captures audio signals in the target frequency range.
3. **3.7V Battery**:
   * Powers the system with a charging module for recharging.
4. **Boost Converter**:
   * Steps up the 3.3V battery output to 5V to power the ESP32.
5. **Push Button with Pull-up Resistor (10kΩ)**:
   * Triggers FFT analysis when pressed.
6. **Additional Components**:
   * GPIO-controlled LED for visual feedback during processing.

**3. Software Implementation**

**3.1 Libraries Used**

* **Arduino.h**: For general Arduino programming functions.
* **WiFi.h**: For Wi-Fi connectivity.
* **ArduinoJson.h**: For handling JSON data.
* **ESPAsyncWebServer.h**: To set up an HTTP server for data access.
* **arduinoFFT.h**: For performing FFT analysis.
* **driver/i2s.h**: For interfacing with the I2S microphone.

**3.2 Functional Blocks**

**3.2.1 FFT Analysis**

* **Purpose**: To analyze audio signals and extract their frequency components.
* **Frequency Ranges**:
  + Short FFT: Processes 1024 samples for quick 1-second analysis.
  + Long FFT: Processes 2048 samples for more detailed 4-second analysis.
* **Implementation**: Uses Hamming windowing to reduce spectral leakage.

**3.2.2 Interrupts and Tasks**

* **Button Interrupt**: Triggers both short and long FFT computations.
* **FreeRTOS Tasks**:
  + processShortFFT: Computes short FFT every 1 second.
  + processLongFFT: Computes long FFT every 4 seconds.

**3.2.3 HTTP Server**

* Hosts a web server to display detected frequency data in JSON format.

**3.3 Key Functions**

**3.3.1 I2S Setup**

Configures the I2S interface to capture audio data from the INMP microphone with the following parameters:

* Sampling frequency: 40 kHz.
* DMA buffer settings: 4 buffers, 512 samples each.

**3.3.2 Frequency Detection**

* Uses the majorPeak() method to identify the dominant frequency in the audio signal.
* Conditions:
  + For a specific frequency range (3000-3500 Hz), triggers a "Fire Alarm" alert.

**3.3.3 HTTP Server Setup**

* Responds with JSON data containing the most recent short and long FFT frequency detections.

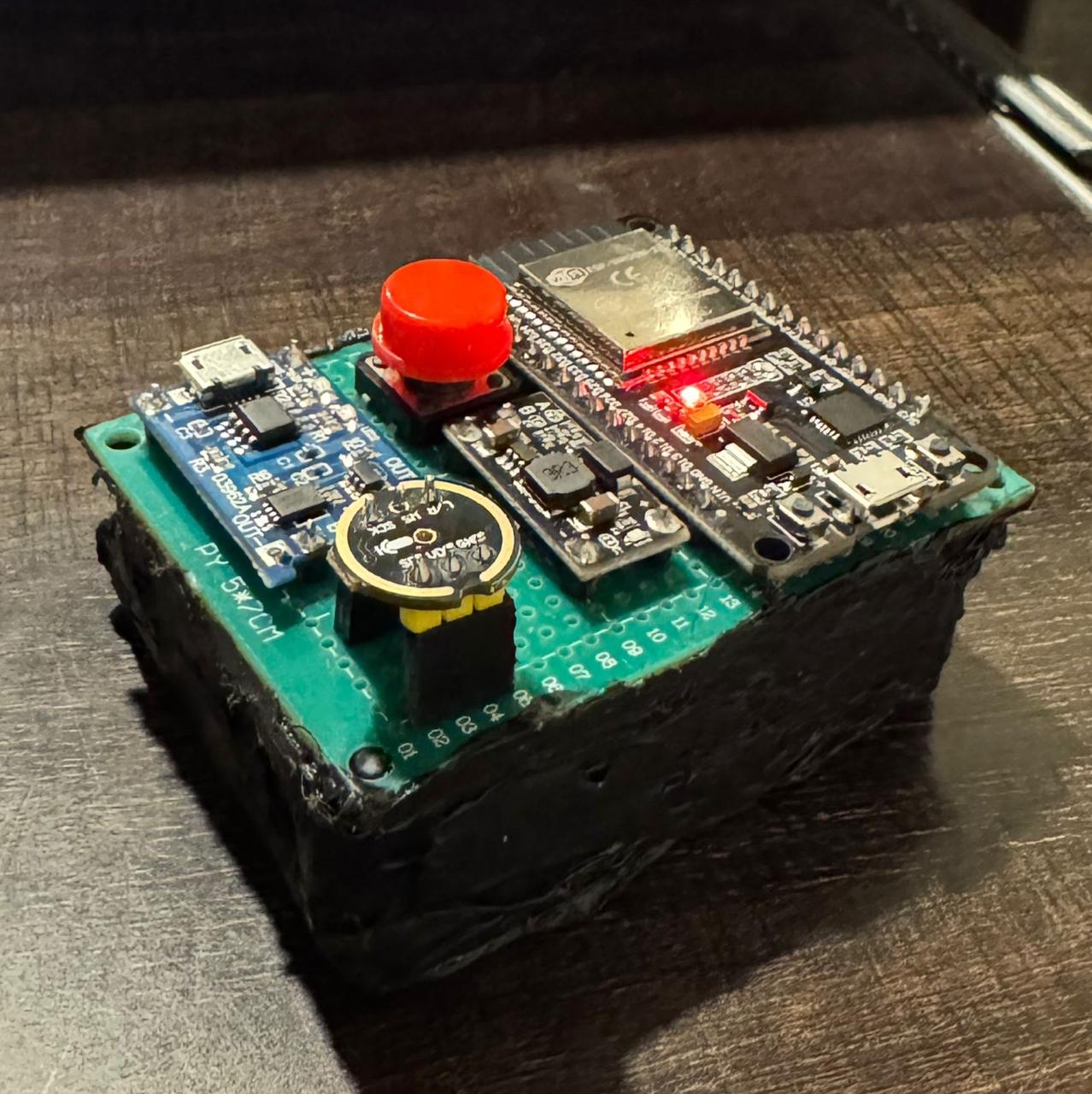
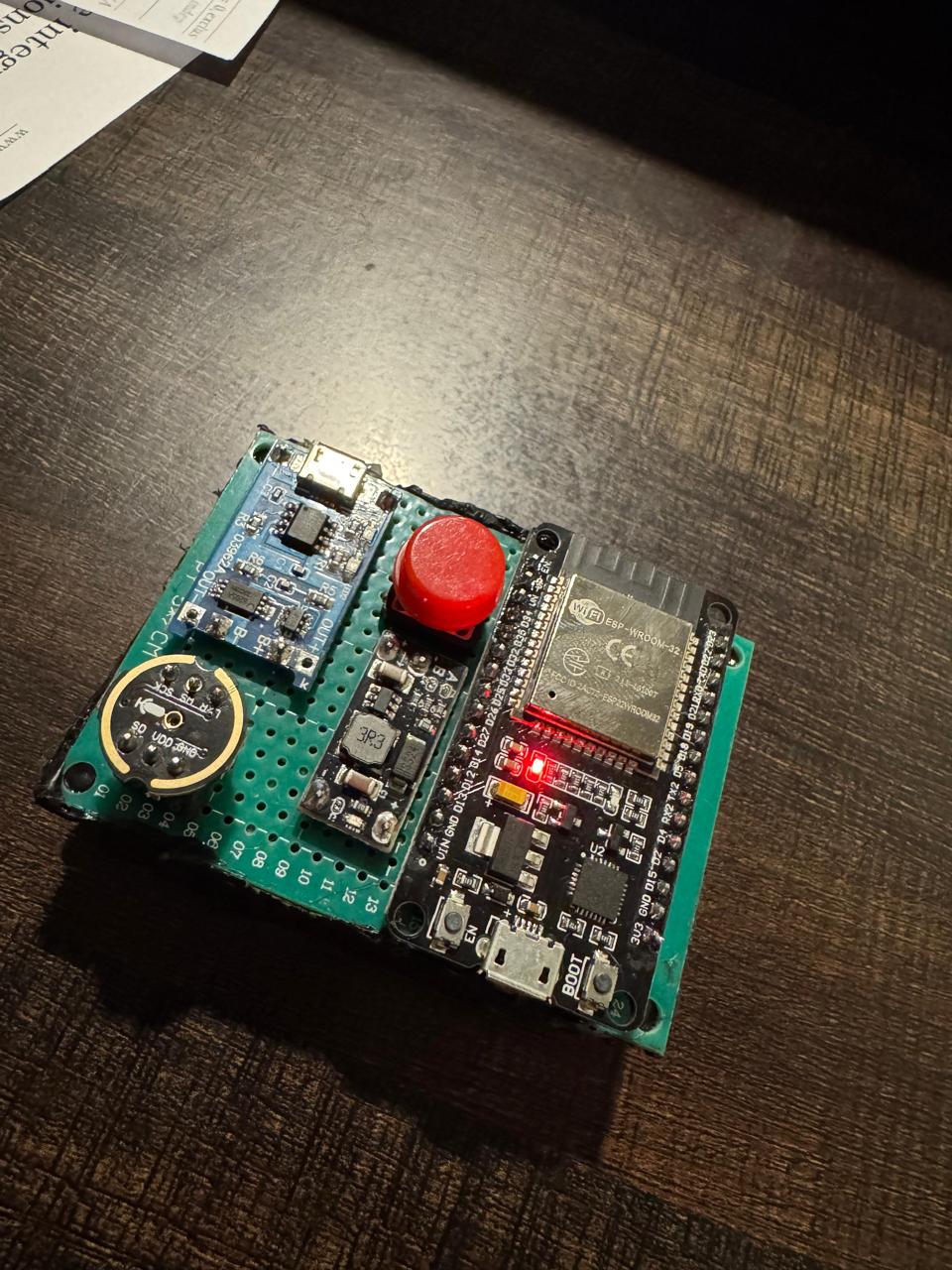
**4. Working Mechanism**

1. **Power Up**:
   * The ESP32 is powered by a 3.3V battery, stepped up to 5V using a boost converter.
2. **Audio Signal Capture**:
   * The INMP microphone captures sound signals and sends data to the ESP32 via the I2S protocol.
3. **FFT Computation**:
   * FFT tasks process the audio data for short (1 second) and long (4 seconds) intervals.
   * Frequency analysis identifies the dominant frequencies.
4. **Wi-Fi Connectivity**:
   * The ESP32 connects to a Wi-Fi network and hosts an HTTP server.
5. **User Interaction**:
   * The push button triggers FFT computations.
   * The user accesses detected frequency data through the web server.

**5. Results and Observations**

* The system successfully detects and computes the dominant frequencies in the 0-10 kHz range.
* The "Fire Alarm" feature reliably identifies frequencies between 3000 and 3500 Hz.
* Real-time frequency data is accessible through the HTTP server in JSON format.

**Hardware Images**



**6. Challenges and Solutions**

1. **Spectral Leakage**:
   * Solution: Applied Hamming windowing to improve frequency resolution.
2. **Memory Constraints**:
   * Solution: Reduced DMA buffer size and FFT sample count.
3. **Noise in Audio Signal**:
   * Solution: Used a high sampling frequency (40 kHz) for improved signal clarity.

**7. Conclusion and Future Scope**

This project demonstrates a robust method for frequency detection using an ESP32 and an INMP microphone. Potential future improvements include:

* Enhancing frequency resolution with higher FFT sample sizes.
* Implementing noise filtering for more accurate detections.
* Expanding frequency range or incorporating machine learning for sound classification.

**8. References**

* Arduino and ESP32 documentation.
* ArduinoFFT library reference.
* ESPAsyncWebServer and Wi-Fi library documentation.