

***Applied Industrial IoT Project***

**Accident Detection System**

*GitHub Link:* <https://github.com/23781A0421/Accident-Detection-System>

**Project Members:**

23781A0421 – B.Revanth

23781A0422 – B.Sumanth

23781A0423 – B. Venkataramana Reddy

23781A0424 – B.Karthik Reddy

23781A0425 – B.Kishore

**Problem Statement:**

Design and develop an accident detection system using the ESP8266 to provide real-time monitoring, immediate alerts and remote accessibility for enhancing emergency response to road accidents.

**Scope of solution:**

This project aims to develop a real-time Accident Detection System using the ESP8266 microcontroller and MPU6050 sensor to monitor sudden impacts or abnormal acceleration patterns. The system is designed to detect vehicle accidents and immediately trigger alerts via internet-based communication services such as Discord Webhooks and Thingspeak dashboards.

The system has the following features:

* Real-time monitoring of accelerometer and gyroscopic data using the MPU6050 sensor.
* Detecting abnormal impact magnitudes beyond a predefined threshold.
* Sending emergency alerts within 30 seconds of crash detection to Discord Webhook.
* Discord Webhook for alert notification.
* Thingspeak dashboard for data logging and visualization.
* Providing a manual cancel option via push button to avoid false alarms.

This version is primarily focused on simulation and possible prototype. The system does not currently include GPS tracking, but these features can be added in future iterations to enhance real-world usability.

**(Please follow the project files through GitHub link provided under the project title)**

**Required components and tools:**

***Note:*** *Due to unavailability of the components (ESP8266), we are using Wokwi with ESP32 to simulate exact functionality of the system. Both simulation for ESP32 and real implementation using ESP8266 codes and files are included in GitHub repository.*

**For simulation:**

**Hardware:**

* ESP32-DevKitC V4
* Buzzer
* MCU6065
* LED

**Softwares:**

* Discord
* Thingspeak
* Wokwi (Website)

**For real implementation:**

**Hardware:**

* ESP8266
* Buzzer
* MCU6065
* LED (With 220ohm resistor)

**Softwares:**

* Discord
* Thingspeak
* Fritzing (PCB Design)
* Arduino IDE

**Setup:**

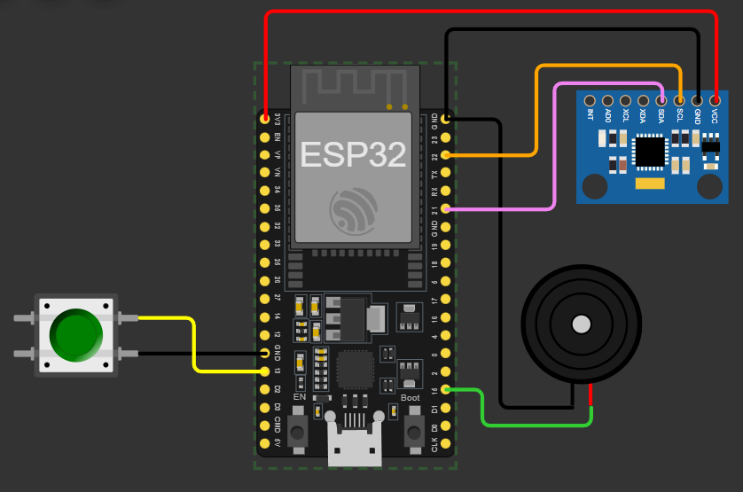
* **Discord (Webhook):**
  + Create an own server
  + Create a channel (with name for example: accident\_detection)
  + Go to “server settings”
  + Under “Apps”, go to “Integrations”
  + Click on Webhooks and click on “New Webhook”
  + Expand the newly created webhook and change the name accordingly(“Accident Detection Alerts” here) and profile picture as fit.
  + Choose the webhook channel to “#channel\_name” (“#accident\_detection” here)
  + We have successfully added webhook in discord.
  + Click on “Copy Webhook URL” and paste it in the main code file to send alerts directly to discord.
* **Thingspeak (Dashboard):**
  + Go to <https://thingspeak.mathworks.com/> and create an account
  + Under “My Channels”, click on “New Channel”
  + Give it a name and description and “Save Channel”
  + After creating, go to the channel and go to “API Keys” and copy the write API key.
  + Paste the API key in the main code to receive data directly to dashboard.
* **Wokwi (Simulation):**
  + Go to <https://wokwi.com/> and click on “ESP32”
  + Scroll down and click on “ESP32” under “Starter Templates”
  + Paste the main code and diagram code in the Wokwi editor
  + Setup as per circuit diagram and code is written or paste from the “main.py” file in “Accident Detection(Wokwi using ESP32-DevKitC V4)” folder in the Wokwi code editor.
  + Go to library manager and add “ArduinoJSON” and “MPU6050” libraries.
  + In simulation area add the push button, buzzer and MPU6050.
  + Connect the pins as per below details:
    - Push button
      * “2.l” to Pin 13
      * “1.l” to GND
    - Buzzer
      * Pin 2 to Pin 16
      * Pin 1 to GND
    - MPU6050
      * “SDA” to Pin 21
      * “SCL” to Pin 22
      * “VCC” to 3V3 Pin
      * “GND” to GND

***Note:*** *You can use copy and paste code from “diagram.json” directly instead of manually connecting*

*everything.*

* + Modify at the start of the code as per your Discord webhook URL, Thingspeak API Key and use “Wokwi-GUEST” as WIFI SSID and “” as WIFI password.
  + Click on Run to run the simulation and in serial monitor we can see what the ESP32 is initializing.

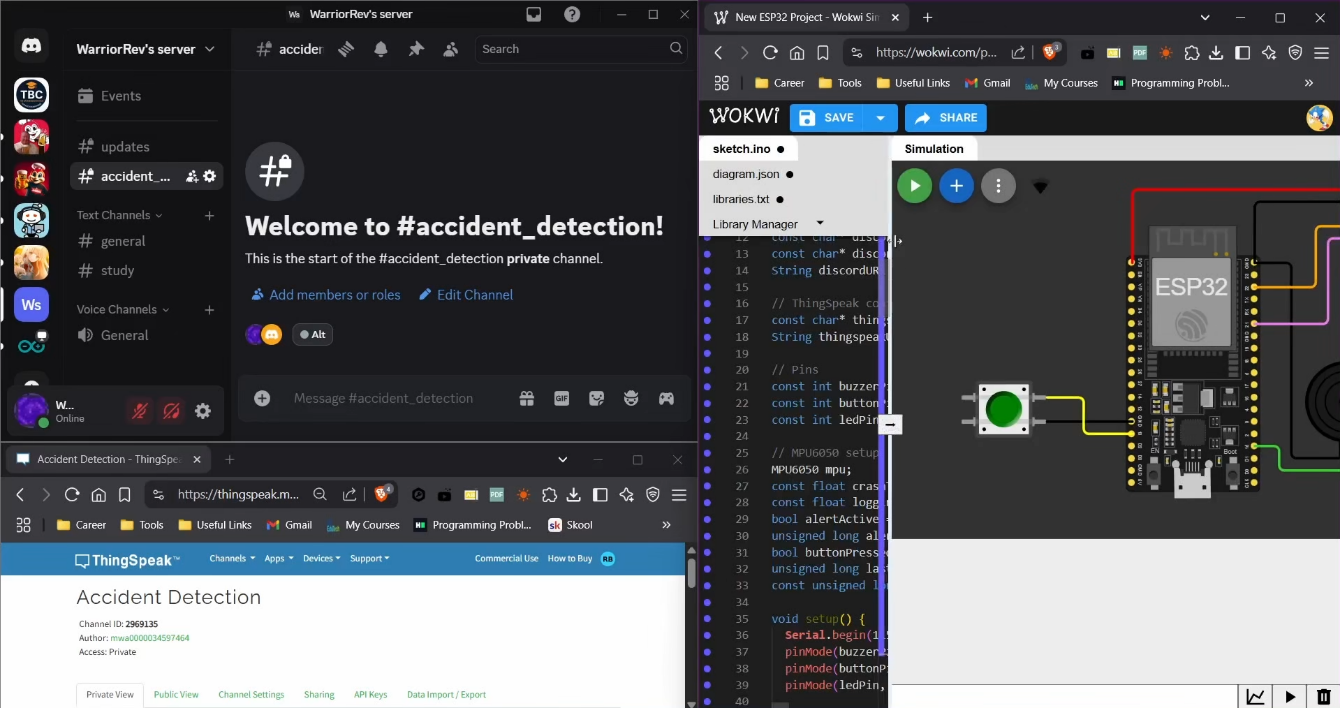
**Circuit Diagram (Wokwi using ESP32):**



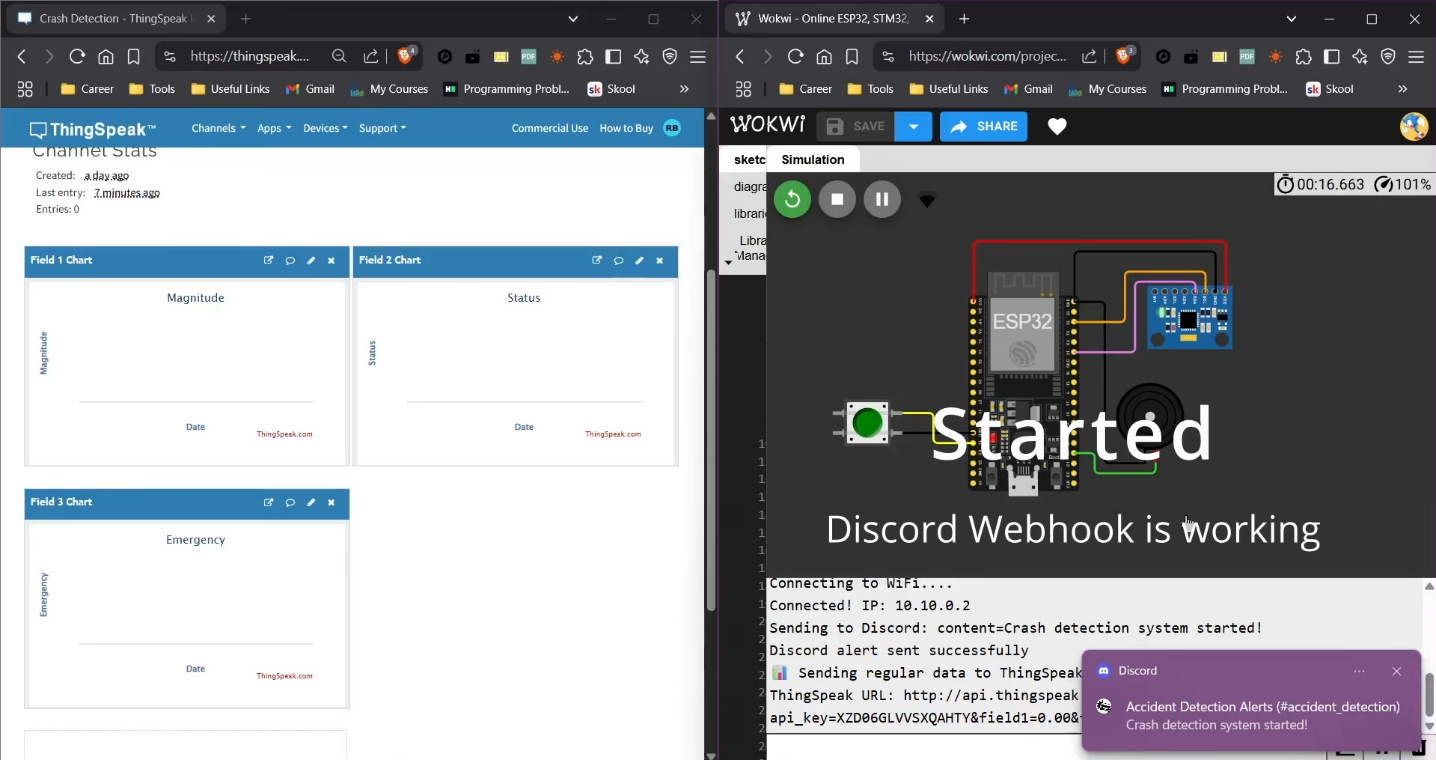
The above circuit diagram shows the connections to the ESP32 in Wokwi. Here consists of ESP32-DevKitC V4, push button, buzzer, built-in LED is used, MPU6050.

**Demo (Using Wokwi) Explanation:**

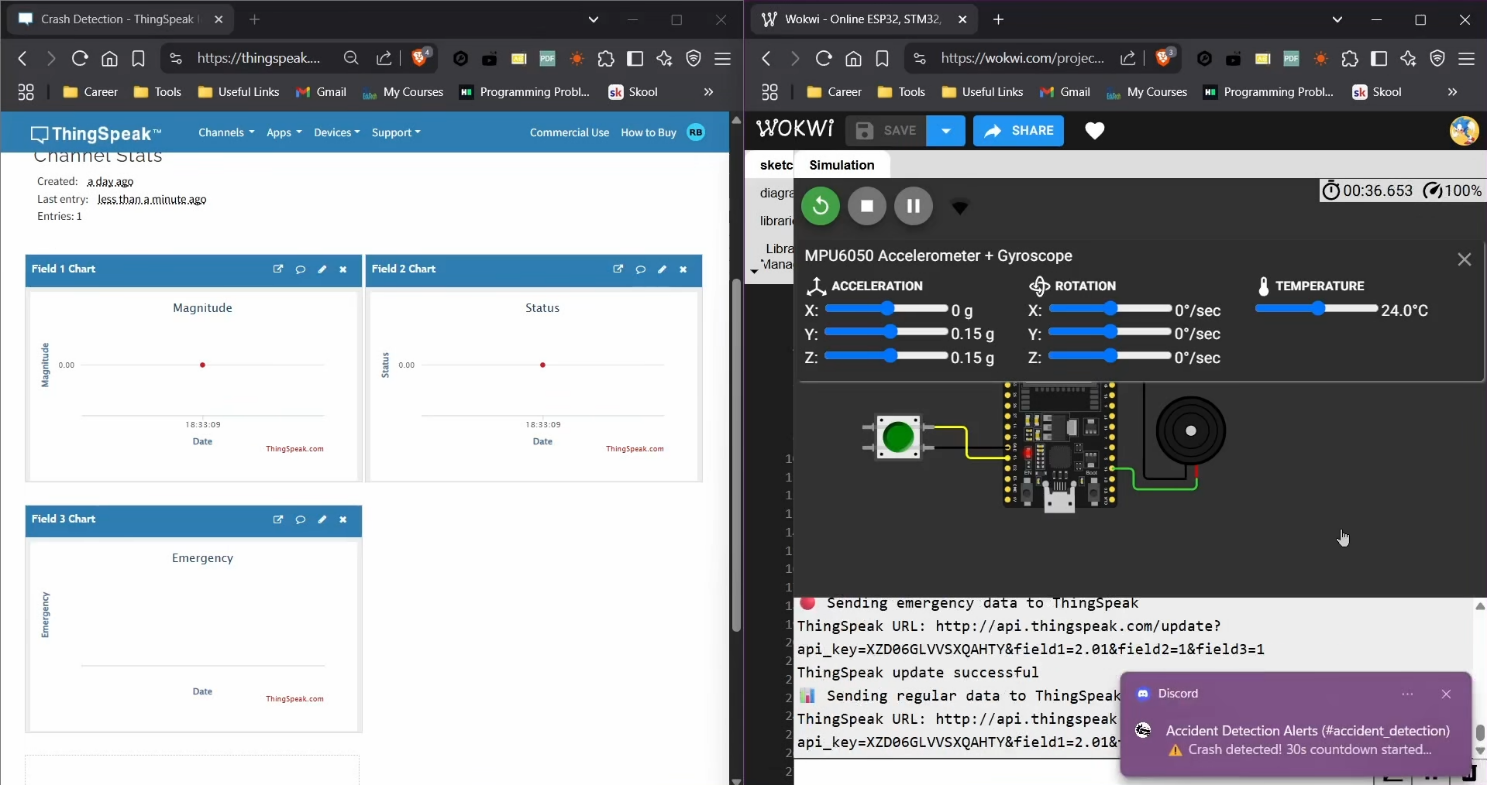
* Open the Wokwi setup before along with Thingspeak and Discord channel.

****

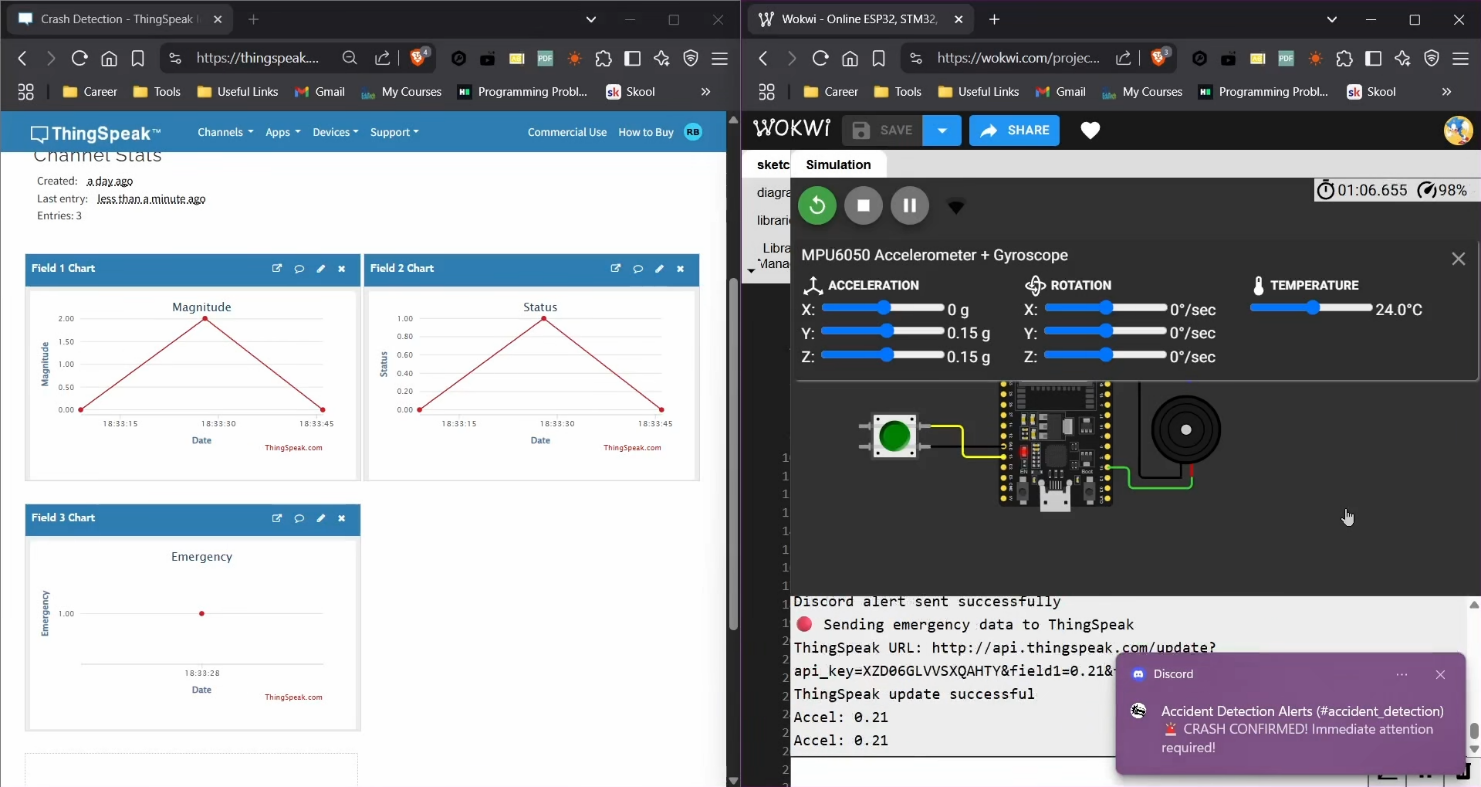
* Click on “Run” button and after few seconds (as you can see in the serial monitor) it will connect.
* You will get “Crash detection system has started!” message in the discord channel which indicates that it is successfully sending messages to discord channel.

****

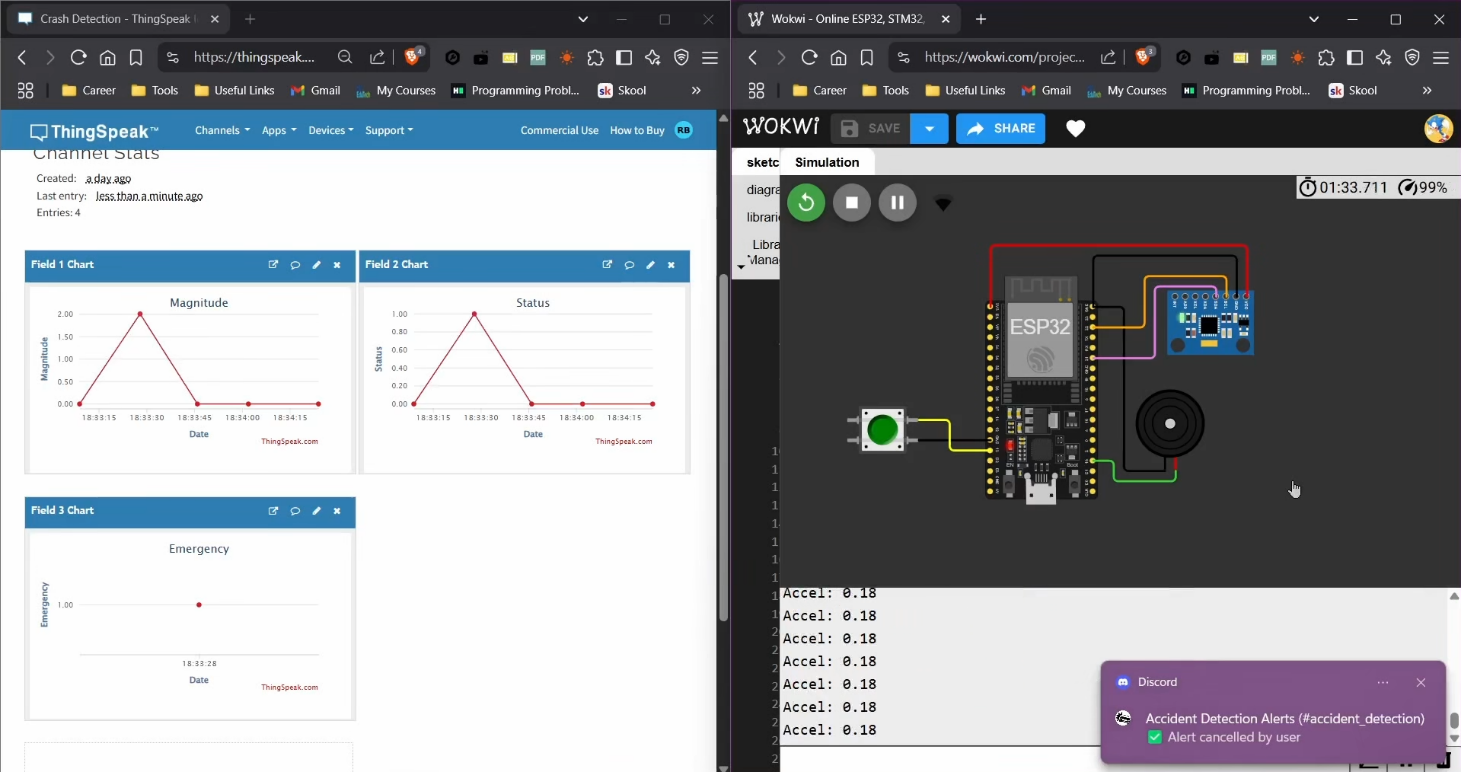
* To simulate a crash, click on MPU6050 and manually change the acceleration values. It will automatically send a crash detection message “Crash detected! 30s countdown started...” to discord channel.

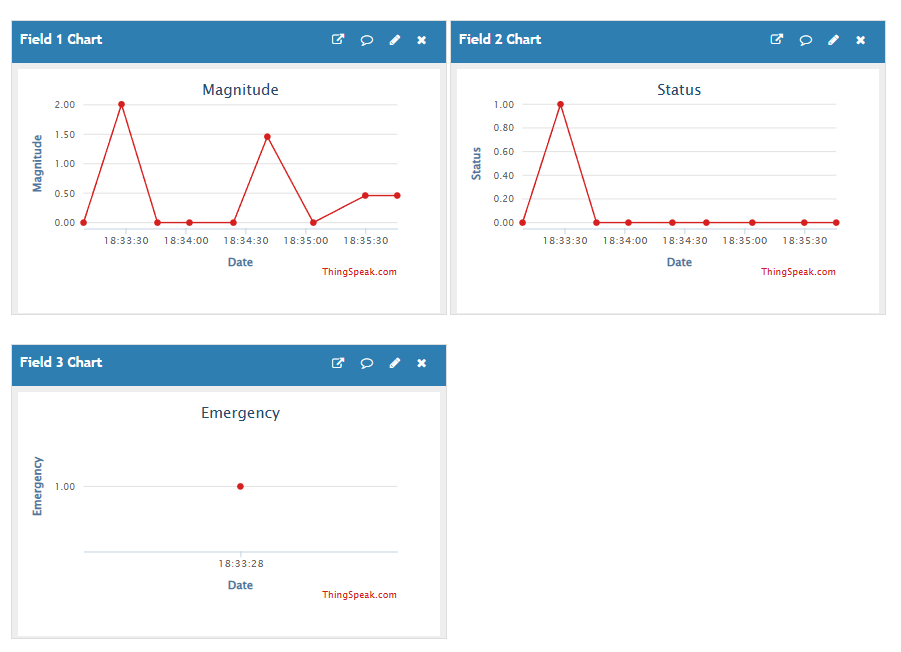


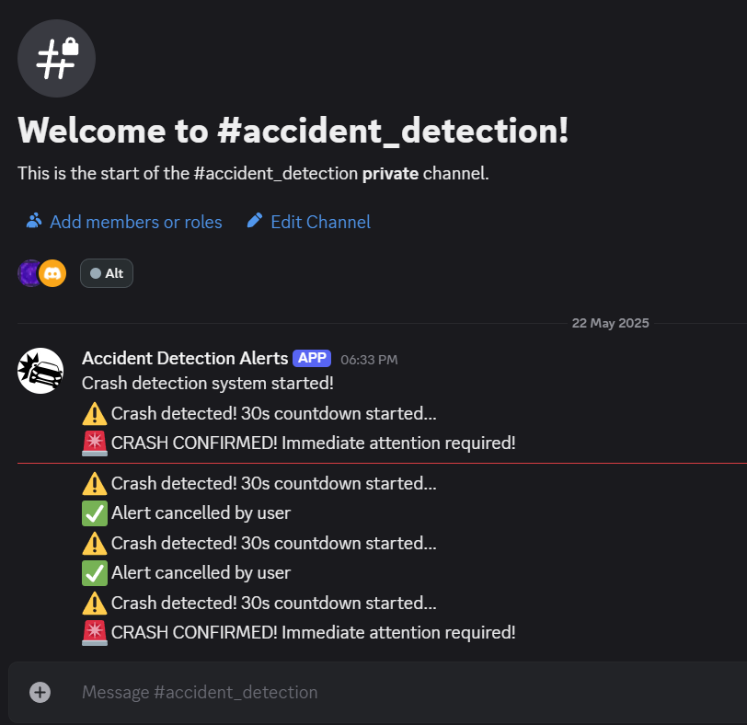
* After 30 seconds without clicking the button, it will send a crash confirmation message “CRASH CONFIRMED! Immediate attention required!” to notify that the crash notification is not a false trigger.



* If we click the button, it will send a “Alert cancelled by user” to indicate that it was a false trigger.



* Periodically every few seconds, Thingspeak will automatically receive the crash detection data.



* These are the messages coming to discord through webhook.

**Code Explanation:**

* **WiFi Credentials**

const char\* ssid = "Wokwi-GUEST";

const char\* password = "";

* Connects ESP32 to Wokwi's built-in Wi-Fi using "Wokwi-GUEST".
* **Discord Webhook Configuration**

const char\* discord\_id = "...";

const char\* discord\_token = "...";

String discordURL = "https://discord.com/api/webhooks/" + String(discord\_id) + "/" +

String(discord\_token);

* + Prepares a custom Discord Webhook URL to post alerts into a specific channel.
* **ThingSpeak Configuration**

const char\* thingspeakAPIKey = "...";

String thingspeakURL = "http://api.thingspeak.com/update";

* + Configures the ThingSpeak channel API key to log crash and motion data.
* **Pin Definitions**

const int buzzerPin = 15;

const int buttonPin = 13;

const int ledPin = 2; // Onboard LED

* + Sets pins for buzzer, button, and LED on ESP32.
* **MPU6050 Initialization**

MPU6050 mpu;

const float crashThreshold = 1.5;

const float loggingThreshold = 0.3;

* + Initializes MPU6050 sensor and sets two thresholds:
  + crashThreshold: Above 1.5g = crash
  + loggingThreshold: Above 0.3g = regular movement to log
* **setup() Function**
  + Starts serial communication and initializes the pins
  + Starts I2C communication on SDA (21) and SCL (22) for MPU6050
  + Connects to WiFi and confirms connectionSends "System Started" message to Discord and sends initial dummy data to ThingSpeak
* **loop() Function**

Performs the core logic repeatedly:

1. Reads Acceleration from MPU6050
2. Converts to G-force from raw values
3. Calculates magnitude using formula: √(ax² + ay² + az²)
4. Checks push button: If pressed and alert is active, cancel the alert
5. Crash Detection: If magnitude > 1.5g, trigger crash alert
6. Regular Data Logging: Every 15 seconds, sends motion data if magnitude > 0.3g, otherwise sends heartbeat (zero)
7. Alert Timeout: If 30 seconds pass and alert still active, sends final alert and turns off buzzer

* **Alert & Logging Functions:**

**triggerAlert(float magnitude)**

* Activates buzzer and LED
* Sets alertActive = true
* Sends Discord alert: "WARNING: Crash detected!"
* Sends ThingSpeak data with emergency flag

**cancelAlert()**

* Turns off buzzer and LED
* Sends Discord message: "Alert cancelled by user"
* Logs status 0 to ThingSpeak

**sendAlerts(float magnitude)**

* Sends final confirmation alert: "CRASH CONFIRMED!"
* Logs status 2 to ThingSpeak
* Turns buzzer off (LED remains on for visual warning)

***sendDiscordAlert(const char message)*\***

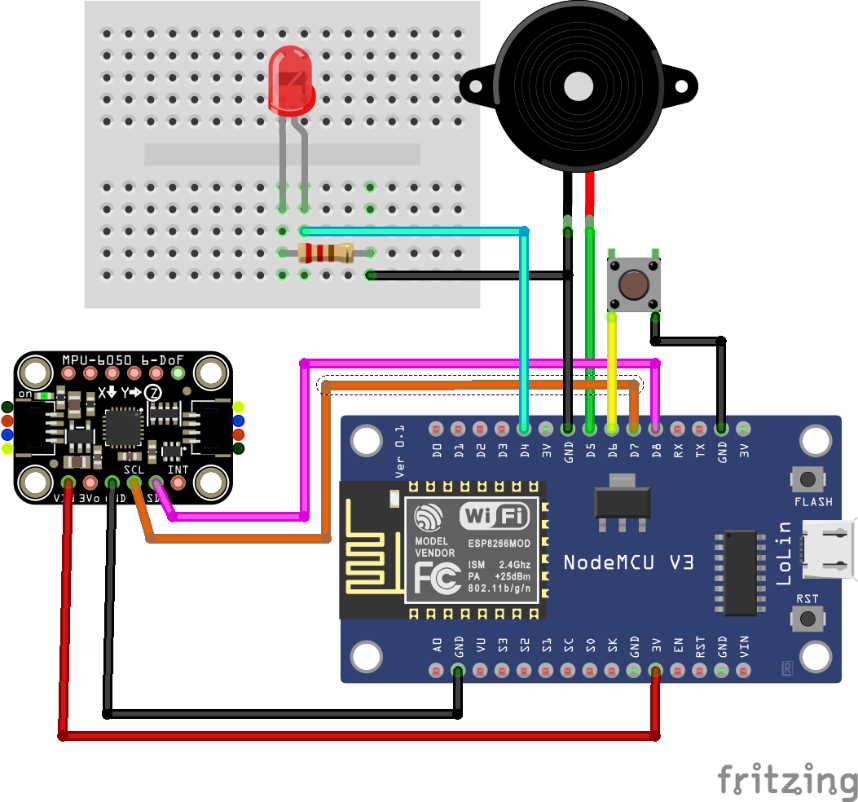
* Sends a message to Discord using Webhook
* Uses HTTP POST with content-type: application/x-www-form-urlencoded

**sendThingSpeakData(float magnitude, int status, bool isEmergency)**

* Sends data to ThingSpeak using HTTP GET
* Fields:
  + field1: Acceleration magnitude
  + field2: Status code (0 = idle, 1 = pending crash, 2 = confirmed crash)
  + field3: Emergency flag (1 if crash)

**Implementation using ESP8266:**

By using ESP8266 which is cheaper version, we can implement the same system but with minor changes. The files are provided in “Accident Detection(Using ESP8266)” folder. It includes code file, PCB Gerber file for direct usage and circuit diagram for reference (Friting file is also included to modify PCB design if required).



* Connect the pins as per below details for ESP8266:
  + Push button
    - One to Pin D6
    - Another on same side to GND
  + Buzzer
    - Positive to Pin D5
    - Negative (Ground) to GND
  + LED
    - Anode to Pin D4
    - Cathode to GND
  + MPU6050
    - “SDA” to Pin D8
    - “SCL” to Pin D7
    - “VCC” to 3V Pin
    - “GND” to GND
* Follow the same instructions to add Thingspeak API key, Discord Webhook URL and WiFi Credentials in the “accident\_detection\_esp8266\_code.ino” file.

**Expansion and addition of features to improve further:**

We can use GSM module to further improve the system capabilities. It will add location tracking which will be very helpful and also immediate alerts. Even without GSM, we can add location tracking feature through this [GitHub repository](https://github.com/gmag11/WifiLocation). Its description is as follows:

“*When location information is needed in an electronic project, we normally think about a GPS module. But we know that mobile phones can get approximate location listening WiFi signals, when GPS is disabled or not usable because we are inside a building*.”

It is very cheap and cost effective to implement. This system is further can be improved by using multiple MPU6050 sensors placed at different locations (for example corners) of the vehicle to more accurately detect the crash due to better spatial data collection and helping distinguish between minor impacts and severe collisions.

**Reference links:**

ArduinoJSON Library: <https://github.com/bblanchon/ArduinoJson>

MPU6050 Library: <https://github.com/electroniccats/mpu6050>

Wokwi Simulation Project: <https://wokwi.com/projects/431580179335709697>

WifiLocation Library: <https://github.com/gmag11/WifiLocation>

Thingspeak Documentation: <https://www.mathworks.com/help/thingspeak/getting-started-with-thingspeak.html>

Wokwi Documentation: <https://docs.wokwi.com/>

Accident Detection Algorithm: <https://koreascience.kr/article/JAKO202032265178628.page?volissCtrlNo=v5n1s14&pubDt=2006&lang=en>

**Used Tools:**

* Microsoft Word (Documentation)
* OBS Studio (Demo recording)
* Davinci Resolve (Cut edit video)
* Fritzing (PCB and Circuit Designing)
* Wokwi (Simulation)
* Arduino IDE (Programming)