EE381 Project: Automated Drunk Driving Detection and Alert System

Sandeep Kumar (220958)

April 2025

Q1. What problem are you trying to solve and why is it important/interesting?

Answer: The problem we are addressing is drunk driving, a major cause of road accidents leading to severe injuries and fatalities. Many drivers fail to recognize their intoxication levels, endangering themselves and others. While breathalyzer tests exist, they require manual operation and law enforcement intervention, which is often impractical for real-time prevention. This project is important to us because we have observed the persistent risk of drunk driving in our community, and an automated solution could significantly reduce accidents and save lives. Our system prevents the vehicle from starting if alcohol is detected and sends real-time location alerts to a predefined contact via a Telegram chatbot, ensuring timely assistance and enhancing safety.

Q2. What are the existing solutions? Describe a few of them and list any shortcomings in them. Is your solution approach unique in some way?

Answer: Several existing solutions attempt to address drunk driving, but each has limitations:

1. **Traditional Breathalyzer Tests**
    - These require manual operation by the driver or law enforcement.
    - *Shortcomings:* Can be bypassed, requires human intervention, and is not suitable for real-time prevention without police presence.
2. **Ignition Interlock Devices (IID)**
    - Used in legal systems where convicted DUI offenders must pass a breath test to start their car.
    - *Shortcomings:* Expensive, requires professional installation, and is limited to convicted drivers, missing proactive prevention.
3. **Mobile Apps for Alcohol Detection**

- Some apps use cognitive tests or Bluetooth breathalyzers to detect impairment.
- *Shortcomings:* Inaccurate, easy to bypass, requires user participation, and lacks integration with vehicle controls.

Unique Aspects of This Project:

- Uses an MQ-135 gas sensor to detect alcohol in the driver's breath.
- Prevents vehicle ignition using an SG90 servo motor to block the ignition switch.
- Sends real-time location alerts via an ESP32 Wi-Fi module to a Telegram chatbot.
- Allows remote ignition unlocking via Telegram commands.
- Fully automated, non-bypassable, and cost-effective.

Q3. What resources do you require to complete the project? Give a breakdown of tasks that you need to accomplish week by week to complete the project.

Hardware Components Required

- MQ-135 Gas Sensor (for alcohol detection)
- Arduino Microcontroller
- ESP32 Wi-Fi Module (for sending alerts)
- Ublox NEO-6M GPS Module (for location tracking)
- SG90 Servo Motor (to disable ignition)
- Power Supply & Connecting Wires
- Breadboard

Software Requirements

- Arduino IDE
- Telegram Bot API
- Google Maps API

Timeline and Task Breakdown

- **Week 1: Component Familiarization and Basic Setup**
  - Tested MQ-135 gas sensor, ESP32, and servo motor individually.
  - Calibrated the gas sensor threshold using alcohol vapor.
  - Implemented basic ignition lock logic with the servo motor.
- **Week 2: GPS and Wi-Fi Integration**
  - Connected and tested the NEO-6M GPS module for location tracking.
  - Configured ESP32 Wi-Fi to connect to the network and interface with Telegram.
  - Sent initial location data to the Telegram chatbot.
- **Week 3: System Integration and Testing**
  - Integrated all components: MQ-135, servo, GPS, and ESP32.
  - Tested end-to-end functionality, including alcohol detection and alert sending.
  - Implemented Telegram command for remote ignition unlocking.
- **Week 4: Final Refinement and Documentation**
  - Conducted real-world testing with alcohol simulation.
  - Debugged and optimized the system for reliability.
  - Documented the project, including code and images.

Implementation Details

System Architecture

1. **Sensing Node (ESP32):**
   - Connected to the MQ-135 gas sensor, NEO-6M GPS module, and SG90 servo motor.
   - Processes gas sensor data, retrieves GPS location, and controls the servo.
   - Sends alerts and receives commands via the Telegram Bot API.
2. **Telegram Interface:**
   - Receives real-time location alerts with Google Maps links.
   - Allows remote ignition unlocking with the "On" command.

Circuit Design

- **Sensing Node:** MQ-135 sensor on analog pin 34, servo on pin 14, GPS on serial pins (RX=16, TX=17), ESP32 powered via USB.

Working Principle

1. The MQ-135 sensor detects alcohol in the driver's breath.
2. If the gas value exceeds 1200, the ESP32 locks the ignition by rotating the servo to 180°.
3. The GPS module retrieves latitude and longitude, and the ESP32 sends an alert to the Telegram chatbot.
4. The driver or contact can unlock the ignition by sending "On" via Telegram.

Arduino Code Details

This code is for setting up components and implementing basic ignition lock

```
#include <ESP32Servo.h>

// Servo setup
Servo myServo;
const int servoPin = 14;
const int gasSensorPin = 34;
const int gasThreshold = 1200;

int lock = 0;

void setup() {
  Serial.begin(115200);
  myServo.attach(servoPin);
  myServo.write(90);  // Start at 0 degrees
}

void loop() {
  int gasValue = analogRead(gasSensorPin);
  Serial.print("Gas Sensor Value: ");
  Serial.println(gasValue);

  if (gasValue > gasThreshold) {
    Serial.println("Alcohol Detected!");
    if (lock == 0) {
      myServo.write(180);
      Serial.println("Ignition Locked");
      delay(300);
      myServo.write(90);
      lock = 1;
    }
  }

  delay(1000);
}
```

Listing 1: Code for Component Setup and Ignition Lock

This code is for integrating GPS and Wi-Fi with Telegram messaging

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <TinyGPS++.h>
#include <HardwareSerial.h>

// WiFi credentials
const char* ssid = "kaka";
const char* password = "12345678";

// Telegram Bot Token and Chat ID
const char* botToken = "8092574247:
   AAHcP1n0ytysOOJF2uq2BgyT5aioQpOsUXs";
const char* chatID = "1085014882";

// GPS setup
TinyGPSPlus gps;
HardwareSerial GPSserial(1);  // RX=16, TX=17
double latitude = 0.0;
double longitude = 0.0;

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("WiFi␣Connected!");
  GPSserial.begin(9600, SERIAL_8N1, 16, 17);
  Serial.println("GPS␣Module␣Initialized.");
}

void loop() {
  while (GPSserial.available() > 0) {
    gps.encode(GPSserial.read());
  }

  if (gps.location.isValid()) {
    latitude = gps.location.lat();
    longitude = gps.location.lng();
    Serial.print("Latitude:␣"); Serial.println(latitude, 6);
    Serial.print("Longitude:␣"); Serial.println(longitude, 6);
```

```
    String message = "Location: https://www.google.com/maps/
       search/?api=1&query=";
    message += String(latitude, 6) + "," + String(longitude, 6);
    sendTelegramMessage(message);
  } else {
    Serial.println("Waiting for GPS fix...");
  }

  delay(5000);
}

void sendTelegramMessage(String message) {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    message.replace(" ", "%20");
    message.replace("\n", "%0A");
    String url = "https://api.telegram.org/bot" + String(botToken
       )
                 + "/sendMessage?chat_id=" + String(chatID)
                 + "&text=" + message;
    http.begin(url);
    int httpResponseCode = http.GET();
    if (httpResponseCode > 0) {
      Serial.println("Telegram message sent!");
    } else {
      Serial.print("Error sending message. HTTP code: ");
      Serial.println(httpResponseCode);
    }
    http.end();
  } else {
    Serial.println("WiFi not connected.");
  }
}
```

Listing 2: Code for GPS and Telegram Integration

This code is for system integration and remote ignition unlocking

```
#include <ESP32Servo.h>

// Servo and variables
Servo myServo;
const int servoPin = 14;
int lock = 0;
int unlock = 0;
String lastReceivedMessage = "";
long lastUpdateId = 0;

void setup() {
  myServo.attach(servoPin);
  myServo.write(90);  // Start at 0 degrees
```

```
}

void loop() {
  if (lock == 1 && unlock == 1) {
    Serial.println("Ignition␣will␣Remain␣Unlocked␣for␣2␣hours");
    delay(1 * 60 * 1000); // 1 minute delay for testing
    lock = 0;
    unlock = 0;
  }

  getTelegramUpdates();
  if (lastReceivedMessage == "On" && unlock == 0) {
    Serial.println("Command␣Received:␣Unlocking␣the␣ignition...")
       ;
    myServo.write(0);
    delay(500);
    myServo.write(90);
    unlock = 1;
    sendTelegramMessage("Ignition␣Unlocked");
    lastReceivedMessage = "";
  }

  delay(1000);
}

void sendTelegramMessage(String message) {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    message.replace("␣", "%20");
    message.replace("\n", "%0A");
    String url = "https://api.telegram.org/bot" + String(botToken
       )
                 + "/sendMessage?chat_id=" + String(chatID)
                 + "&text=" + message;
    http.begin(url);
    int httpResponseCode = http.GET();
    if (httpResponseCode > 0) {
      Serial.println("Telegram␣message␣sent!");
    }
    http.end();
  }
}

void getTelegramUpdates() {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    String url = "https://api.telegram.org/bot" + String(botToken
       )
                 + "/getUpdates?offset=" + String(lastUpdateId +
                    1);
    http.begin(url);
```

```
    int httpResponseCode = http.GET();
    if (httpResponseCode > 0) {
      String payload = http.getString();
      int updateIdIndex = payload.indexOf("\"update_id\":");
      if (updateIdIndex != -1) {
        int startId = updateIdIndex + 12;
        int endId = payload.indexOf(",", startId);
        long updateId = payload.substring(startId, endId).toInt()
            ;
        lastUpdateId = updateId;
        int messageIndex = payload.indexOf("\"text\":\"");
        if (messageIndex != -1) {
          int start = messageIndex + 8;
          int end = payload.indexOf("\"", start);
          String message = payload.substring(start, end);
          if (message == "On") {
            lastReceivedMessage = "On";
          }
        }
      }
    }
    http.end();
  }
}
```

Listing 3: Code for System Integration and Remote Unlocking

Photos

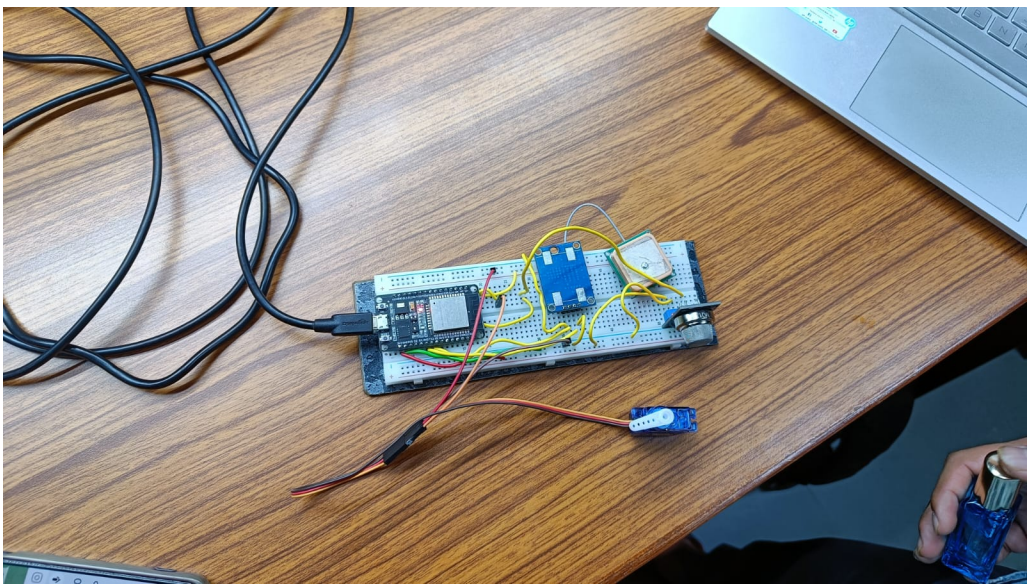

Figure 1: Photograph of the project prototype showing the ESP32, MQ-135 sensor, GPS module, and servo motor.

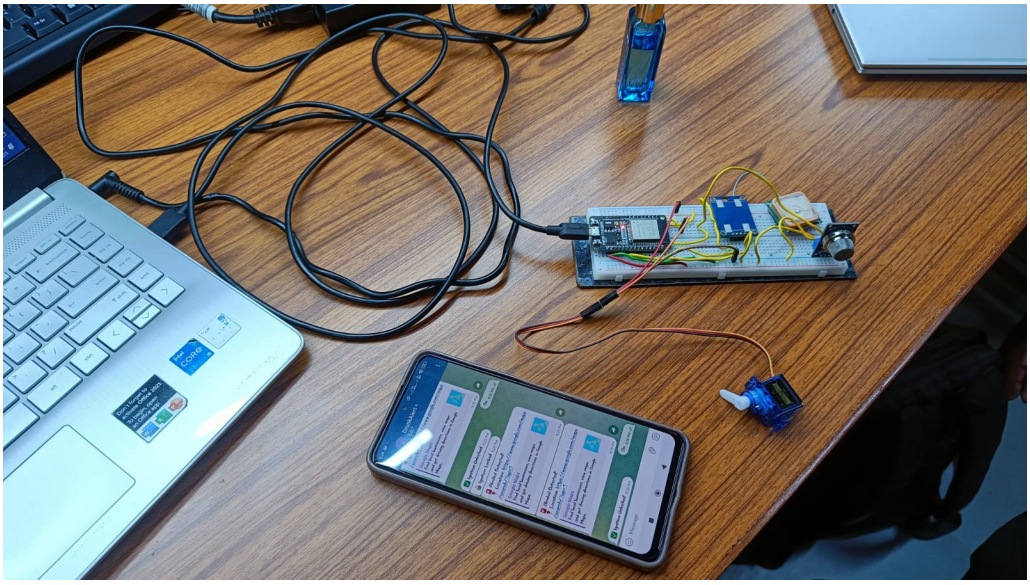

Figure 2: Screenshot of the Telegram alert with location link received on the mobile device.

Block Diagram

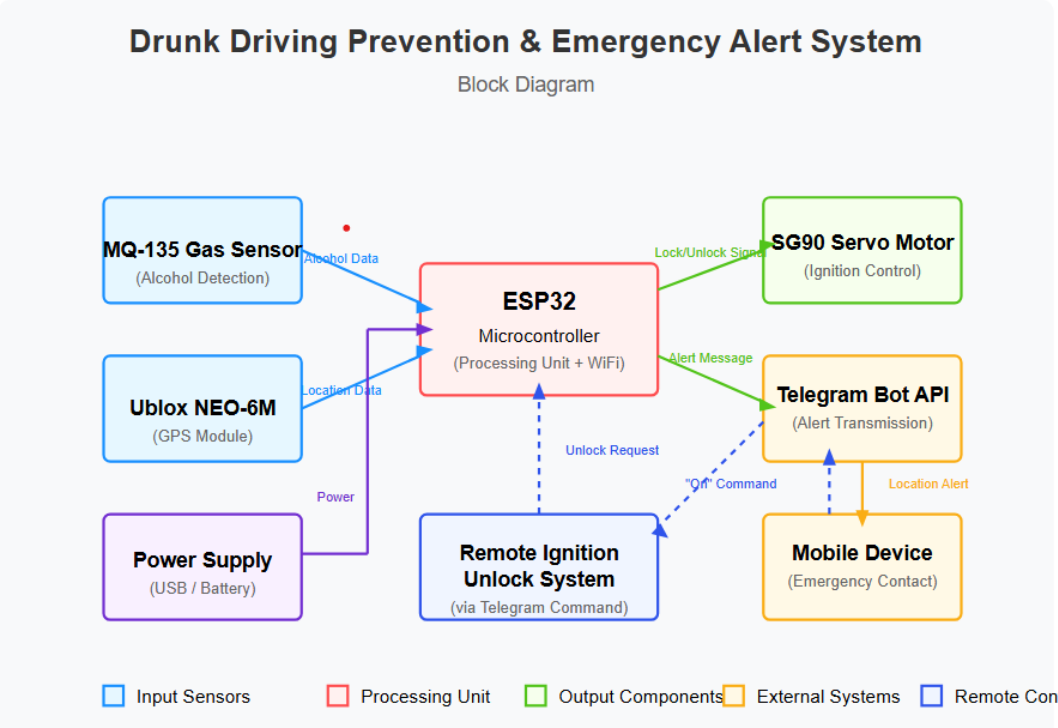

Figure 3: Simple block diagram of the drunk driving prevention system showing the ESP32 as the central controller.

Results

1. **Accurate Alcohol Detection:** The MQ-135 sensor reliably detected alcohol, with a response time of approximately 1-2 seconds during testing with simulated breath.
2. **Ignition Control:** The SG90 servo motor effectively locked the ignition when alcohol was detected, with the ESP32 maintaining control based on sensor input.
3. **Real-time Alerts:** The ESP32 successfully sent Telegram alerts with Google Maps links within 5 seconds of detection, as shown in Figure 2.
4. **Remote Unlocking:** The system responded to the "On" command, unlocking the ignition for a 2-hour period (simulated with a 1-minute delay for testing).

Limitations and Future Improvements

While the system performed well, several limitations were identified:

1. **Detection Accuracy:** The MQ-135 sensor may be affected by environmental factors (e.g., humidity). Future improvements could include:
   - Adding a secondary sensor for confirmation.
   - Implementing machine learning to refine detection thresholds.
2. **Power Management:** The current prototype relies on USB power. Future versions could incorporate:
   - Battery backup with low-power modes.
   - Solar charging for sustainable operation.
3. **Scalability:** Designed for a single vehicle, expansion to multiple vehicles would require:
   - A networked system with a central server.
   - Enhanced GPS tracking for fleet management.
4. **Additional Features:** Potential enhancements include:
   - Integration with smart vehicle systems.
   - Voice command support via Telegram.
   - Predictive analytics for driver behavior.

References

1. Arduino Official Documentation - `https://www.arduino.cc/reference/en/`
2. ESP32 Documentation - `https://docs.espressif.com/projects/esp-idf/en/latest/esp32/`
3. TinyGPS++ Library - `https://github.com/mikalhart/TinyGPSPlus`

4. Telegram Bot API - `https://core.telegram.org/bots/api`

5. Road Safety Statistics - World Health Organization