Project Report

Course Name: IoT and Embedded Systems

Team Members: Nishant Bhavsar

Project Title: FuelGuard

1. Introduction

This project, FuelGuard, is designed to prevent fuel theft and monitor fuel usage by location. It helps track how much fuel is used and where it is being consumed. FuelGuard can also detect fuel leaks, send alerts if fuel is stolen, and provide real-time reports on fuel levels.

2. Problem Statement (Empathize & Define)

Fuel theft is a big problem for people and businesses that use fuel for their vehicles and machines. Some people may steal fuel or use it without permission, causing money loss. Without a good system to track fuel, it is hard to know where and how it is being used. This makes it easy for theft to happen without being noticed.

Another problem is fuel misuse. Some drivers or workers may use fuel for personal reasons instead of work. This makes fuel records wrong and increases costs. Fuel leaks are also a common issue. If a leak is not found quickly, it can waste a lot of fuel, increase expenses, and even harm the environment.

Many businesses still use manual logs to track fuel, but these can be wrong or changed. Without real-time tracking, it is hard to know how much fuel is being used. Some people may also buy extra fuel without permission and change records to hide it.

3. Ideation & Proposed Solution

Fuel theft, misuse, and waste cause money loss and problems for businesses. Keeping fuel records by hand is not always correct and can be changed. Fuel leaks can also happen without being noticed, wasting fuel and harming the environment. A smart system is needed to track fuel, stop theft, and find leaks quickly.

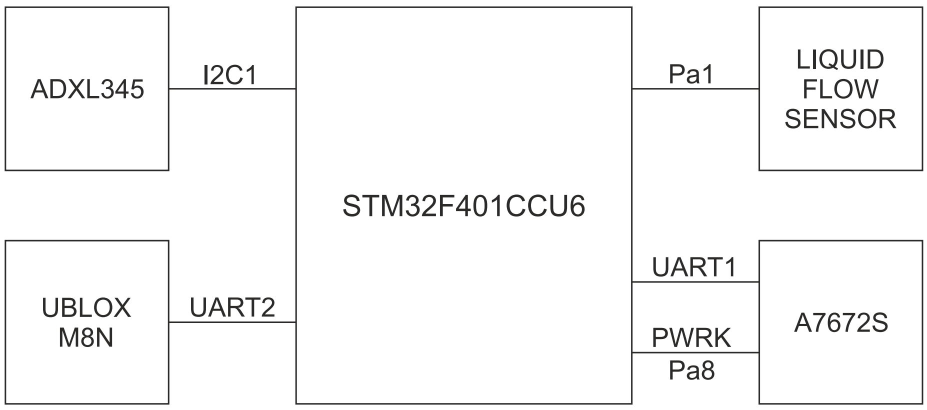
FuelGuard tracks fuel in real-time, sends alerts, and makes reports. It stops theft and makes sure fuel is used correctly. It also finds leaks early to save fuel and protect the environment. With FuelGuard, businesses can save money, use fuel better, and manage it more easily.

This solution uses the **STM32F401CCU6** microcontroller for fast processing and control. The **ADXL345** acceleration sensor checks movement and fuel use to find any misuse or tampering. The **YF-S401** flow sensor measures fuel flow and is placed between the fuel tank and engine to track fuel consumption. The **Ublox M8N GPS** module gives real-time location data, so users can see where fuel is used and find any unauthorized trips. The **Simcom A7672S** module sends real-time alerts anywhere in the world, warning users about fuel theft, misuse, or low fuel, so they can act quickly.

- A humidity or pressure change adjusts automated vents or humidifiers.
- Published events can control automated doors or window blinds, creating a smart home or industrial setup.

This solution leverages the ESP32-S3's BLE for low-power wireless communication and MQTT for scalable and reliable message handling, ensuring seamless integration of sensors, actuators, and control systems. The use of MQTT as a lightweight protocol allows for efficient real-time communication, while BLE ensures compatibility with portable devices, enhancing system flexibility and automation potential.

4. **Design & Architecture (Prototype & Testing)**



Hardware Used
STM32F401CCU6 (Black Pill F401 Microcontroller)
ADXL345
YF-S401 Liquid Flow Sensor
Ublox Neo M8N
Simcom A7672S
Breadboard
Breadboard Power Module

Software Stack
Visual Studio Code
PlatformIO

5. Implementation Details

Here is the pin definition for each sensor which should be connected to **Black Pill(STM32F401CCU6)**.

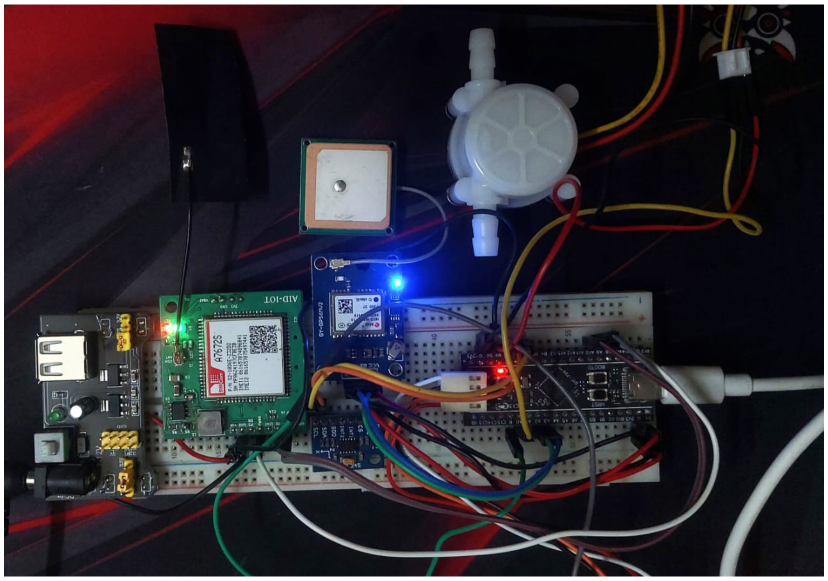
ADXL345 (I2C1)
PB6 (I2C1_SCL) - ADXL345 SCL
PB7 (I2C1_SDA) - ADXL345 SDA

Simcom A7672S (UART1)
PA9 (UART1_TX) - A7672S RX
PA10 (UART1_RX) - A7672S TX

Ublox Neo M8N (UART2)
PA2 (UART2_TX) - Neo M8N RX
PA3 (UART2_RX) - Neo M8N TX

YF-S401 Liquid Flow Sensor
Pa1 (GPIO1) - Sensor Data Pin

6. **Testing & Validation**



testing

For this setup, I have written a code that ensures responses are sent only to a predefined phone number specified in the **platform.ini** file. If a message is received from the defined PHONE_NUMBER, the system processes it and replies exclusively to that number.

Below are the commands that can be sent to retrieve specific data:

status - to check if system is responding or not.

location - to get system's latitude and longitude from gps module.

speed - to get speed of system from gps module.

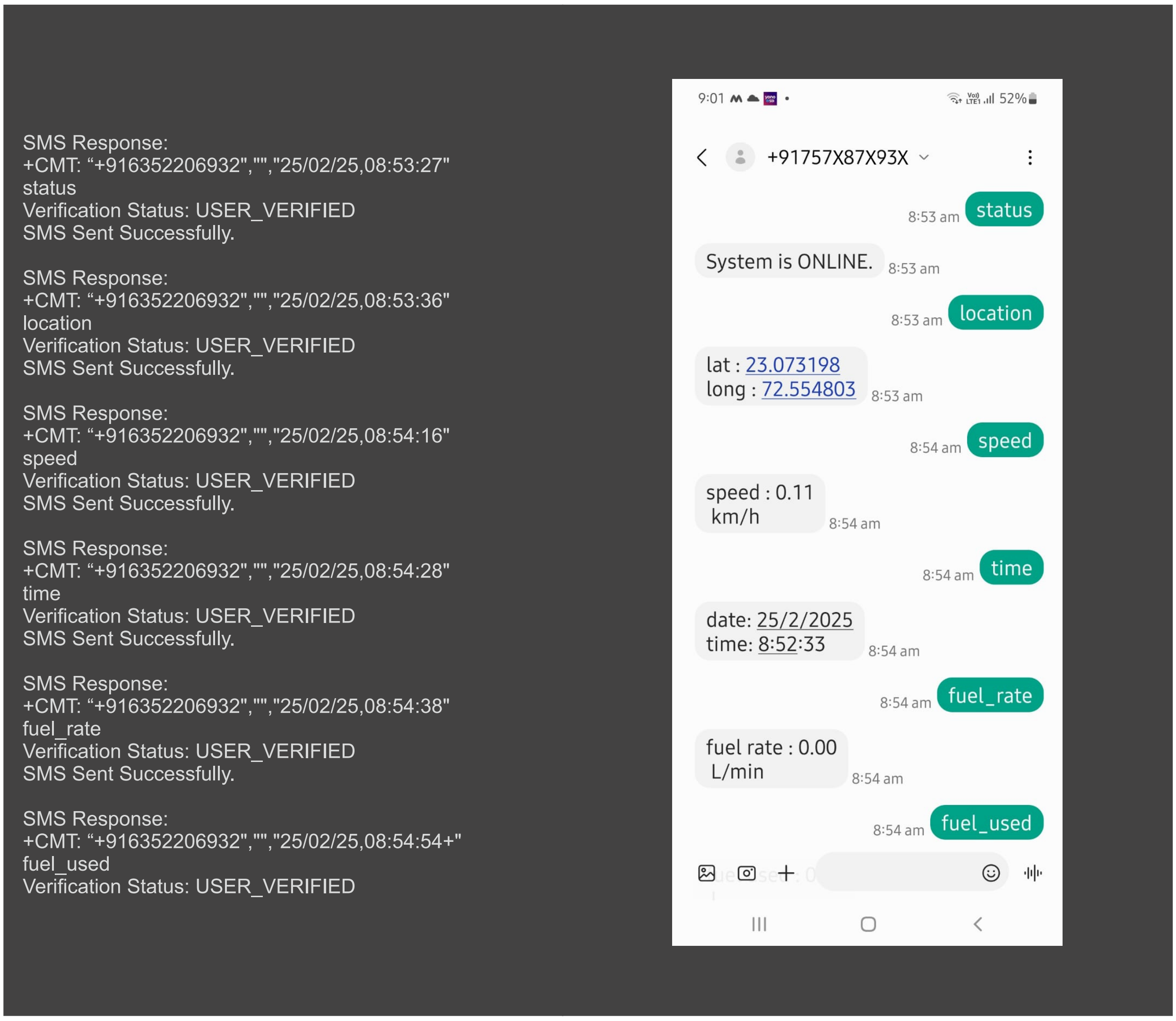
time - to get speed of system from gps module.

fuel_rate - to get the actual fuel passing through liquid flow sensor when message will send.

fuel_used - to get the value, how much fuel used since the system started.

avg_gps - to get average of used fuel and distance traveled according to gps since the system started.

avg_static - to get average of used fuel and distance traveled according to imu since the system started.



```
SMS Response:
+CMT: "+916352206932","","25/02/25,08:53:27"
status
Verification Status: USER_VERIFIED
SMS Sent Successfully.

SMS Response:
+CMT: "+916352206932","","25/02/25,08:53:36"
location
Verification Status: USER_VERIFIED
SMS Sent Successfully.

SMS Response:
+CMT: "+916352206932","","25/02/25,08:54:16"
speed
Verification Status: USER_VERIFIED
SMS Sent Successfully.

SMS Response:
+CMT: "+916352206932","","25/02/25,08:54:28"
time
Verification Status: USER_VERIFIED
SMS Sent Successfully.

SMS Response:
+CMT: "+916352206932","","25/02/25,08:54:38"
fuel_rate
Verification Status: USER_VERIFIED
SMS Sent Successfully.

SMS Response:
+CMT: "+916352206932","","25/02/25,08:54:54+"
fuel_used
Verification Status: USER_VERIFIED
```

9:01 52%

+91757X87X93X

8:53 am status

System is ONLINE. 8:53 am

8:53 am location

lat : 23.073198
long : 72.554803 8:53 am

8:54 am speed

speed : 0.11
km/h 8:54 am

8:54 am time

date: 25/2/2025
time: 8:52:33 8:54 am

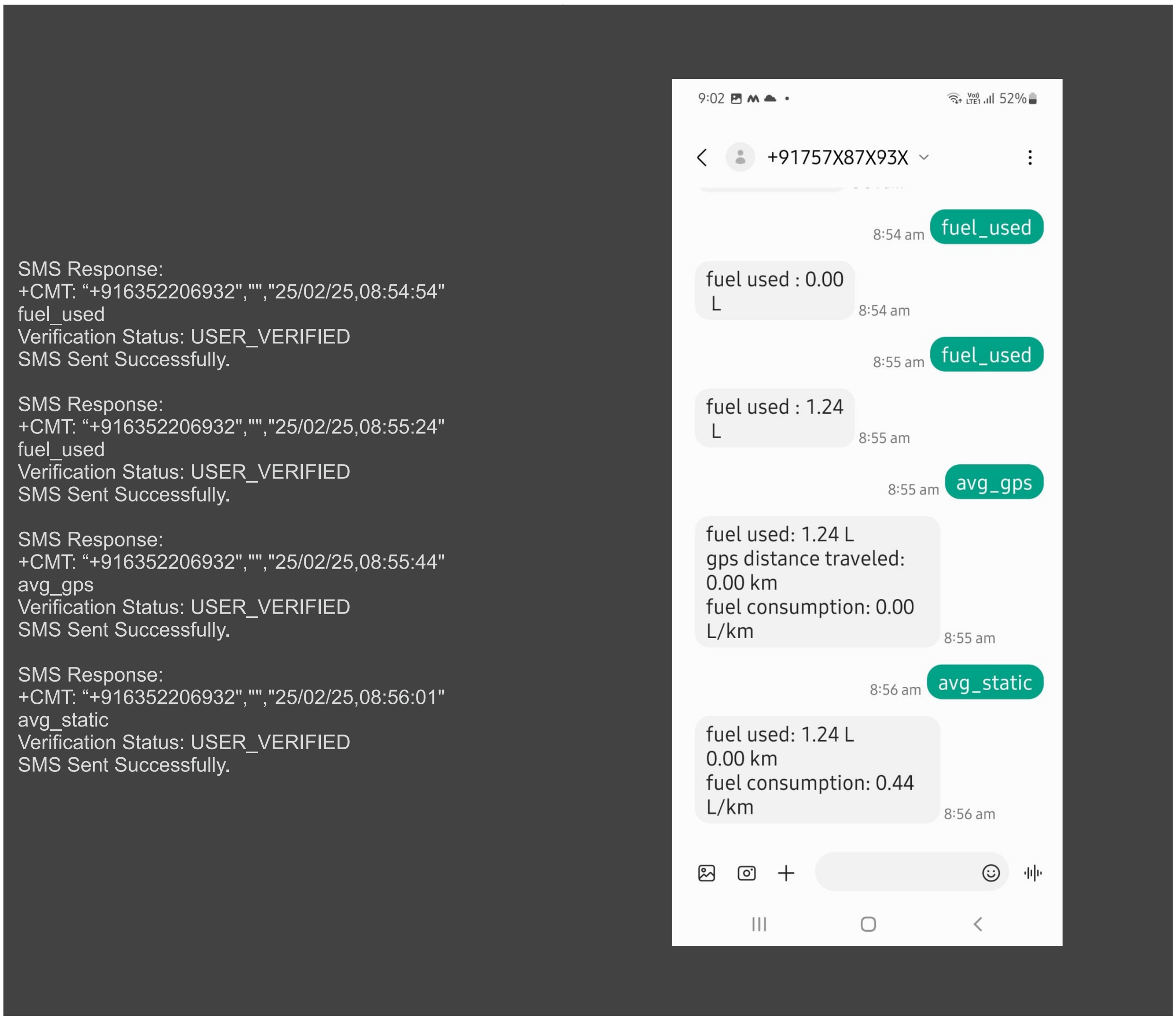
8:54 am fuel_rate

fuel rate : 0.00
L/min 8:54 am

8:54 am fuel_used

On the left side, you can see the copied text data from the **PuTTY** Serial Monitor on COM9 at 115200 baud (in my case).

This time, I sent a message to the system from the phone number specified in the **platformio.ini** file. As a result, the system successfully responded with a message containing each sensor's data, which is shown in the screenshot of my Android device.



```
SMS Response:
+CMT: "+916352206932","","25/02/25,08:54:54"
fuel_used
Verification Status: USER_VERIFIED
SMS Sent Successfully.

SMS Response:
+CMT: "+916352206932","","25/02/25,08:55:24"
fuel_used
Verification Status: USER_VERIFIED
SMS Sent Successfully.

SMS Response:
+CMT: "+916352206932","","25/02/25,08:55:44"
avg_gps
Verification Status: USER_VERIFIED
SMS Sent Successfully.

SMS Response:
+CMT: "+916352206932","","25/02/25,08:56:01"
avg_static
Verification Status: USER_VERIFIED
SMS Sent Successfully.
```

9:02 52%

+91757X87X93X

8:54 am fuel_used

fuel used : 0.00 L 8:54 am

8:55 am fuel_used

fuel used : 1.24 L 8:55 am

8:55 am avg_gps

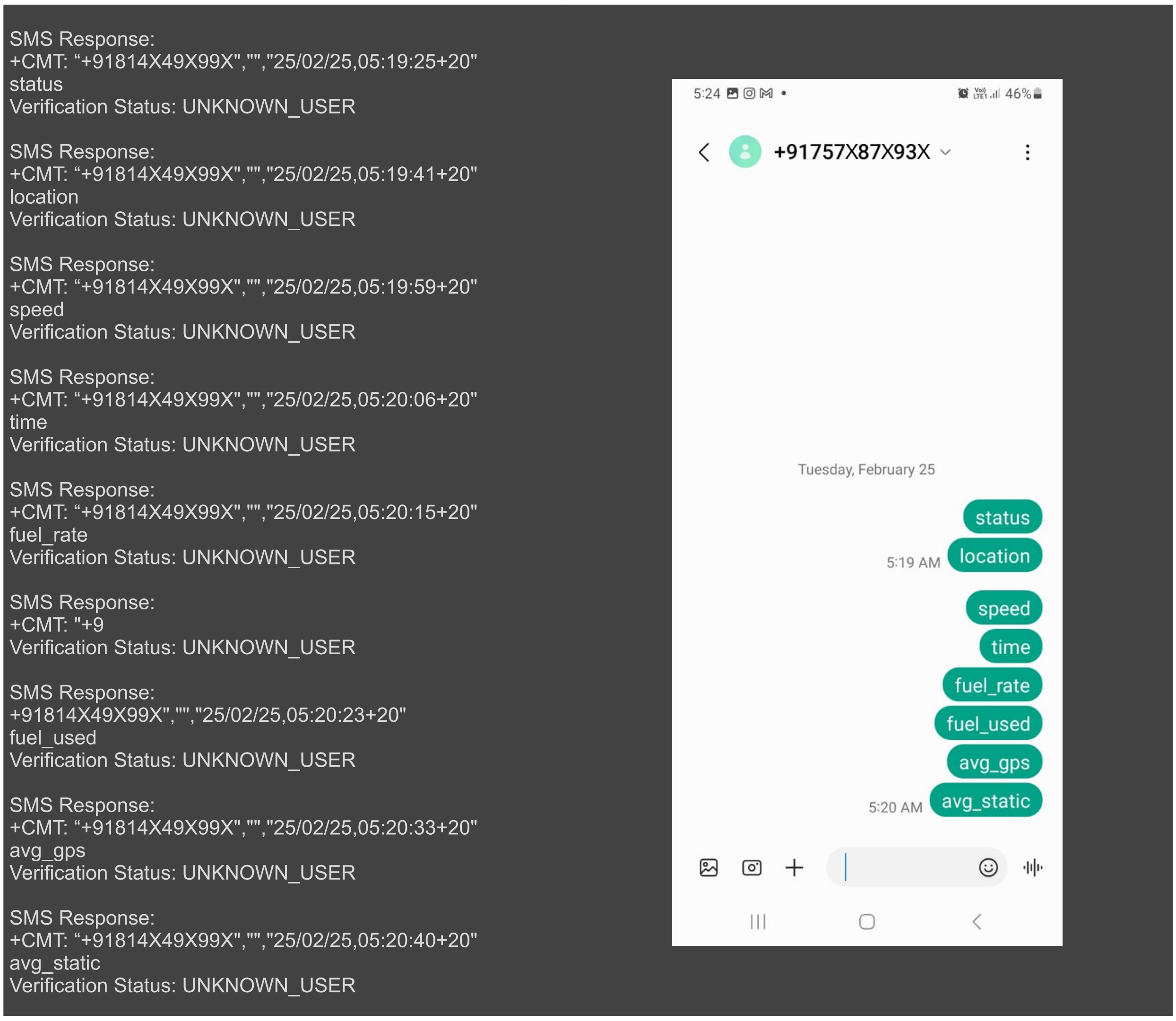
fuel used: 1.24 L
gps distance traveled: 0.00 km
fuel consumption: 0.00 L/km 8:55 am

8:56 am avg_static

fuel used: 1.24 L
0.00 km
fuel consumption: 0.44 L/km 8:56 am

On the left side, you can see the copied text data from the **PuTTY** Serial Monitor on COM9 at 115200 baud (in my case).

This time, I sent a message to the system from the phone number specified in the **platformio.ini** file. As a result, the system successfully responded with a message containing each sensor's data, which is shown in the screenshot of my Android device.



```
SMS Response:
+CMT: “+91814X49X99X","","25/02/25,05:19:25+20"
status
Verification Status: UNKNOWN_USER

SMS Response:
+CMT: “+91814X49X99X","","25/02/25,05:19:41+20"
location
Verification Status: UNKNOWN_USER

SMS Response:
+CMT: “+91814X49X99X","","25/02/25,05:19:59+20"
speed
Verification Status: UNKNOWN_USER

SMS Response:
+CMT: “+91814X49X99X","","25/02/25,05:20:06+20"
time
Verification Status: UNKNOWN_USER

SMS Response:
+CMT: “+91814X49X99X","","25/02/25,05:20:15+20"
fuel_rate
Verification Status: UNKNOWN_USER

SMS Response:
+CMT: "+9
Verification Status: UNKNOWN_USER

SMS Response:
+91814X49X99X","","25/02/25,05:20:23+20"
fuel_used
Verification Status: UNKNOWN_USER

SMS Response:
+CMT: “+91814X49X99X","","25/02/25,05:20:33+20"
avg_gps
Verification Status: UNKNOWN_USER

SMS Response:
+CMT: “+91814X49X99X","","25/02/25,05:20:40+20"
avg_static
Verification Status: UNKNOWN_USER
```

5:24 46%

+91757X87X93X

Tuesday, February 25

status

5:19 AM location

speed

time

fuel_rate

fuel_used

avg_gps

5:20 AM avg_static

On the left side, you can see the copied text data from the PuTTY Serial Monitor on COM9 at 115200 baud (in my case).

From this output, the system extracts the sender's phone number using the **+CMT:** command and only responds if the number matches the one predefined in the platformio.ini file.

However, as shown, the system consistently returns "Verification status: UNKNOWN USER", meaning it is ignoring messages from unverified numbers.

On the right side, I have attached a screenshot of my Android device, from which I sent all the commands to the system. Despite sending the messages, I did not receive any response in return.

7. **Results & Impact**

The prototype is working well and performing as expected, but there are some small issues with accuracy. The distance traveled is not calculated correctly because both the GPS and IMU have errors. The GPS shows slight movement even when the device is not moving, usually around 0.11 km/h, which is normal due to signal variations. The IMU has a lot of noise, making it hard to get stable readings. To improve accuracy, we need to add filters like the Kalman filter or other noise-reduction methods in the code. These changes will help make the measurements more reliable and accurate.

8. **Future Scope & Enhancements**

One important improvement is adding an SD card to store location data with timestamps. This will help when the system enters areas with low network coverage. Instead of losing data, the system can save all locations on the SD card and upload them later to an online spreadsheet when the network is available. This way, tracking remains accurate even in poor network areas.

Another useful upgrade is creating a mobile app for real-time location tracking. The app can show the system's movement on a map, making it easy to follow. It can also send alerts if the system goes off-route or enters a restricted area. This feature can help in security, logistics, and fleet management.

The system can also be improved to track fuel usage. By using **GPS** data, we can estimate how much fuel is used during travel. This can help drivers or companies optimize routes and save fuel costs. The app can show this data on the map, helping users find better and more efficient travel paths.

To make the data more useful, we can add graphs and charts to show speed, distance, fuel usage, and other important details. These graphs will help users understand travel patterns, find issues, and make better decisions based on real-time and past data.

Another important improvement is using sensor fusion, where **GPS** and **IMU** data are combined using filters like the Kalman filter. This will help in reducing errors and noise, giving more accurate tracking even in areas where the GPS signal is weak.

In the future, AI-based predictive analysis can be added. **AI** can analyze past travel data and suggest better routes or detect unusual movement patterns.

This feature will be helpful in smart transportation, logistics, and security tracking. By adding these features, the system will become more reliable, accurate, and useful for different applications. It will help improve tracking, save fuel, and provide better data analysis for decision-making.

9. **Conclusion**

The prototype has successfully demonstrated its ability to track location and respond to predefined commands. However, there are minor accuracy issues due to GPS drift and IMU noise, which can be improved with filtering techniques like the Kalman filter. Despite these limitations, the system works well and can be further enhanced with additional features.

By adding an SD card for offline data storage, the system can continue logging locations even in areas with poor network coverage and upload them later. A real-time tracking mobile app can improve monitoring, while fuel utilization tracking and data visualization with graphs can provide better insights.
Further improvements, like sensor fusion and AI-based predictive analysis, can make the system even more efficient and accurate.

With these enhancements, the system can become a powerful tool for tracking, monitoring, and analyzing movement data, making it useful for logistics, security, and transportation applications.

10. **References & Appendices**

Detailed schematic is available in **schematic.pdf**

Code explaination by different devices connected to mcu is in **code_explain.pdf**