# Rider Telemetry System (ESP32 + GPS + MPU6050 + Flutter App)

## Overview

The Rider Telemetry System monitors a rider's movement in real time using sensors on an ESP32 board. It detects speed, acceleration, lean angle, and GPS location, and transmits this data via Bluetooth to a custom Flutter mobile app. The app shows live telemetry and automatically saves all data locally every 5 seconds.

## Hardware Used

| Component | Function |
|---|---|
| ESP32 | Main controller with Bluetooth |
| MPU6050 | Accelerometer + Gyroscope (motion & lean detection) |
| NEO-6M GPS | Location and speed tracking |
| Android Phone | Runs Flutter telemetry app |

## Connections

| ESP32 Pin | GPS | MPU6050 |
|---|---|---|
| 3.3V | VCC | VCC |
| GND | GND | GND |
| GPIO4 | TX | — |
| GPIO2 | RX | — |
| GPIO21 | — | SDA |
| GPIO22 | — | SCL |

## Firmware

File: rider_telemetry_v5_3_lite.ino
Platform: Arduino IDE

Key Features:
• Reads real-time GPS + MPU6050 data
• Calculates speed, acceleration, and lean angle
• Detects crash or harsh braking
• Sends clean formatted data via Bluetooth

Example Output:

```
[GPS] Lat:12.934567 Lon:77.600231 Spd:27.53 km/h Sats:7 Sig:Strong
[MPU] X:0.02 Y:-0.03 Z:0.98 | Lean:3.8 deg
[MODE] Moving | Speed:27.53 km/h | Signal:Strong
```

## Mobile App

Framework: Flutter (Kotlin backend)
Project Name: ascend

Main Features:

- Auto-connects to Bluetooth device 'RiderTelemetry'
- Displays: State, Activity, Speed, Lean, Acceleration, GPS & Signal
- Saves telemetry every 5 seconds in local memory
- History view accessible from top-right corner

## Setup Instructions

1. Install Flutter SDK → Extract to C:\src\flutter and run 'flutter doctor'
2. Get Project Dependencies:
```
flutter clean
flutter pub get
```
3. Connect Device:
   - Pair ESP32 via Bluetooth (named RiderTelemetry)
   - Enable Developer Mode & USB Debugging on Android
4. Run the App:
```
flutter run
```

## App Interface

Dashboard Shows:

- State (Idle/Moving)
- Activity (Standing/Walking/Scooter)
- Speed (km/h)
- Lean Angle (°)
- Acceleration (X, Y, Z)
- GPS Coordinates
- Signal Strength

History Tab:

- Stores last 500 readings locally in plain text
- Updates every 5 seconds automatically

## Technologies Used

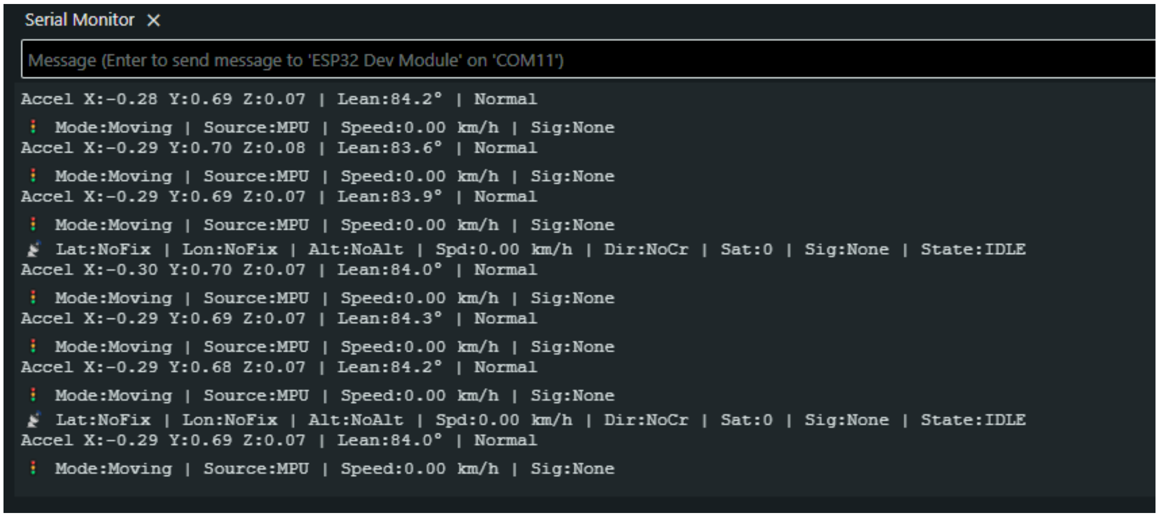
- C++ / Arduino (ESP32 firmware)
- Flutter + Dart (Mobile UI)

- SharedPreferences (Local storage)
- BluetoothSerial (Wireless data transfer)

## Results

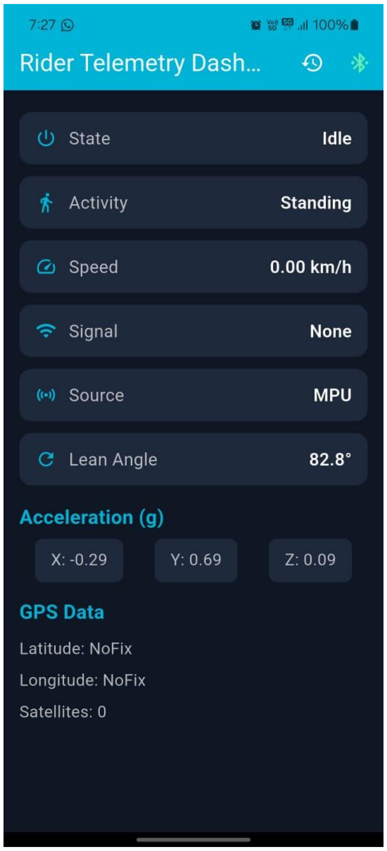
- Successfully displayed live telemetry from rider sensors.
- Reliable Bluetooth communication up to ~10 meters.
- Automatic local data logging for post-ride analysis.

Output in the Serial Monitor looks like this:


Serial Monitor ✕
Message (Enter to send message to 'ESP32 Dev Module' on 'COM11')

```
Accel X:-0.28 Y:0.69 Z:0.07 | Lean:84.2° | Normal
🚦 Mode:Moving | Source:MPU | Speed:0.00 km/h | Sig:None
Accel X:-0.29 Y:0.70 Z:0.08 | Lean:83.6° | Normal
🚦 Mode:Moving | Source:MPU | Speed:0.00 km/h | Sig:None
Accel X:-0.29 Y:0.69 Z:0.07 | Lean:83.9° | Normal
🚦 Mode:Moving | Source:MPU | Speed:0.00 km/h | Sig:None
📡 Lat:NoFix | Lon:NoFix | Alt:NoAlt | Spd:0.00 km/h | Dir:NoCr | Sat:0 | Sig:None | State:IDLE
Accel X:-0.30 Y:0.70 Z:0.07 | Lean:84.0° | Normal
🚦 Mode:Moving | Source:MPU | Speed:0.00 km/h | Sig:None
Accel X:-0.29 Y:0.69 Z:0.07 | Lean:84.3° | Normal
🚦 Mode:Moving | Source:MPU | Speed:0.00 km/h | Sig:None
Accel X:-0.29 Y:0.68 Z:0.07 | Lean:84.2° | Normal
🚦 Mode:Moving | Source:MPU | Speed:0.00 km/h | Sig:None
📡 Lat:NoFix | Lon:NoFix | Alt:NoAlt | Spd:0.00 km/h | Dir:NoCr | Sat:0 | Sig:None | State:IDLE
Accel X:-0.29 Y:0.69 Z:0.07 | Lean:84.0° | Normal
🚦 Mode:Moving | Source:MPU | Speed:0.00 km/h | Sig:None
```

Output on the mobile app we designed looks like this:



Rider Telemetry Dash...

| | |
|---|---|
| State | Idle |
| Activity | Standing |
| Speed | 0.00 km/h |
| Signal | None |
| Source | MPU |
| Lean Angle | 82.8° |

Acceleration (g)

X: -0.29 Y: 0.69 Z: 0.09

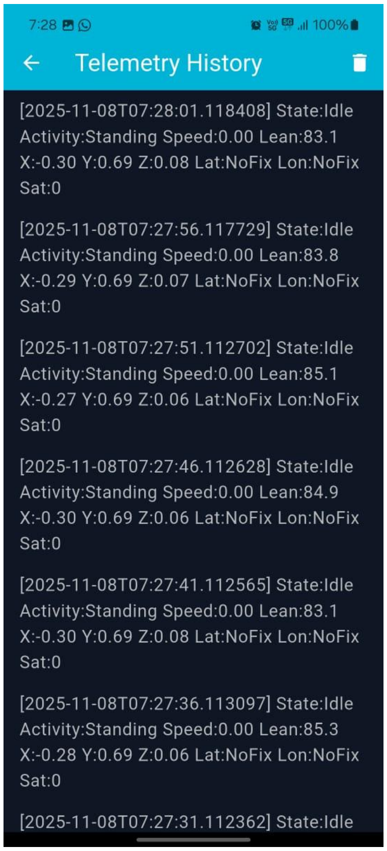
GPS Data

Latitude: NoFix

Longitude: NoFix

Satellites: 0

This is how the terminal history looks like:



```
[2025-11-08T07:28:01.118408] State:Idle
Activity:Standing Speed:0.00 Lean:83.1
X:-0.30 Y:0.69 Z:0.08 Lat:NoFix Lon:NoFix
Sat:0

[2025-11-08T07:27:56.117729] State:Idle
Activity:Standing Speed:0.00 Lean:83.8
X:-0.29 Y:0.69 Z:0.07 Lat:NoFix Lon:NoFix
Sat:0

[2025-11-08T07:27:51.112702] State:Idle
Activity:Standing Speed:0.00 Lean:85.1
X:-0.27 Y:0.69 Z:0.06 Lat:NoFix Lon:NoFix
Sat:0

[2025-11-08T07:27:46.112628] State:Idle
Activity:Standing Speed:0.00 Lean:84.9
X:-0.30 Y:0.69 Z:0.06 Lat:NoFix Lon:NoFix
Sat:0

[2025-11-08T07:27:41.112565] State:Idle
Activity:Standing Speed:0.00 Lean:83.1
X:-0.30 Y:0.69 Z:0.08 Lat:NoFix Lon:NoFix
Sat:0

[2025-11-08T07:27:36.113097] State:Idle
Activity:Standing Speed:0.00 Lean:85.3
X:-0.28 Y:0.69 Z:0.06 Lat:NoFix Lon:NoFix
Sat:0

[2025-11-08T07:27:31.112362] State:Idle
```

How to run:

## 1. Hardware Setup

- Use ESP32 + NEO-6M GPS + MPU6050 sensors.
- Wiring:
    - GPS → RX = 4, TX = 2
    - MPU6050 → SDA = 21, SCL = 22
    - Connect GND and 3.3V for both modules.
- Upload the Arduino sketch (e.g. RiderTelemetry_v5.3-Lite.ino) to the ESP32.
- Power the ESP32. It will broadcast Bluetooth with the name: RiderTelemetry.

---

## 2. Pairing with Phone

- On your Android phone, go to Bluetooth settings.
- Find and pair with **RiderTelemetry** (ESP32).
- Make sure Bluetooth and Location are both turned ON.

---

## 3. Flutter App Setup

- Install Flutter SDK from their official website.
- Open the Flutter project folder (e.g. ascend/) in VS Code or Android Studio.
- Run the following commands in the terminal:
    - flutter clean
    - flutter pub get
    - flutter run
- Connect your Android phone via USB (enable Developer mode + USB debugging).

---

## 4. Using the App

- The app will automatically try to connect to the ESP32 Bluetooth.

- Once connected, the dashboard will display:
    - State: Idle or Moving
    - Activity: Standing, Walking, or Scooter
    - Speed, Lean Angle, Acceleration (X/Y/Z), GPS data, and Signal strength
- Data updates in real time.
- Every 5 seconds, readings are automatically saved to local memory.

---

**5. Viewing History**

- Tap the **History** button (top-right corner in the app).
- You'll see timestamped sensor data stored as plain text logs.
- You can clear history from the same page.

---

Created by:
Team Ascend.ai