
CHAPTER 4 - PRELIMINARY RESULTS

This chapter presents the initial findings from the prototyping, integration, and preliminary field testing of the proposed smart vehicle telemetry and tamper detection system. The evaluation focuses on hardware integration, firmware stability, energy autonomy, and communication reliability.

4.1. Prototype Implementation and Hardware Integration

4.1.1. Mechanical Design and Switch Prototyping

Physical tamper detection mechanisms were validated through bench-testing prior to vehicle deployment. A custom circular mounting bracket (Figure 4.1) was fabricated to house a micro limit switch. When interfaced with the ESP32, this assembly successfully generated an instantaneous hardware interrupt upon actuation, simulating unauthorized fuel cap removal. Furthermore, a custom, water-resistant enclosure was designed to securely house the main microcontroller and communication modules while protecting them from environmental factors (Figure 4.2 and Figure 4.3). A micro limit switch was integrated directly into this enclosure to provide active monitoring; opening the main case instantly triggers a hardware interrupt. Additionally, contactless magnetic limit switches were integrated into the test circuit. Physical separation of the magnetic contacts successfully triggered immediate tamper alerts, confirming the hardware logic required to secure the rear cargo doors.

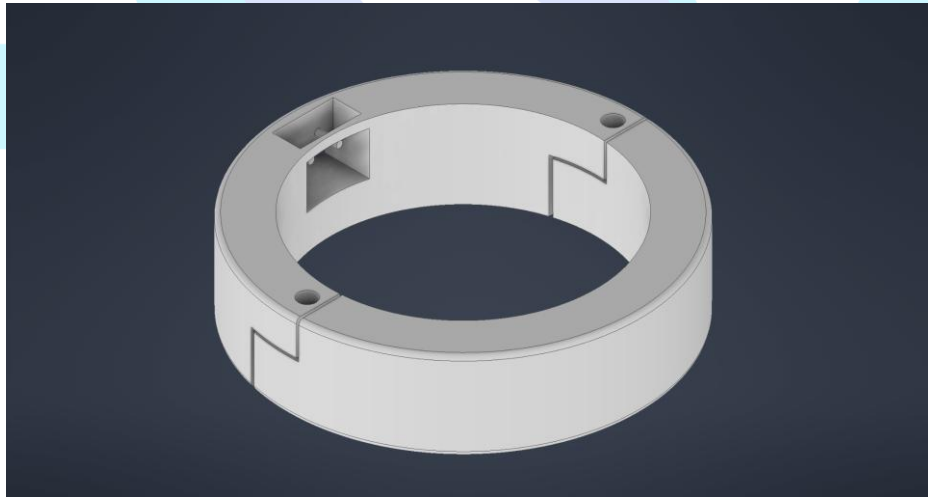


Figure 4.1: 3D CAD model of the custom fuel cap mounting bracket for the switch.

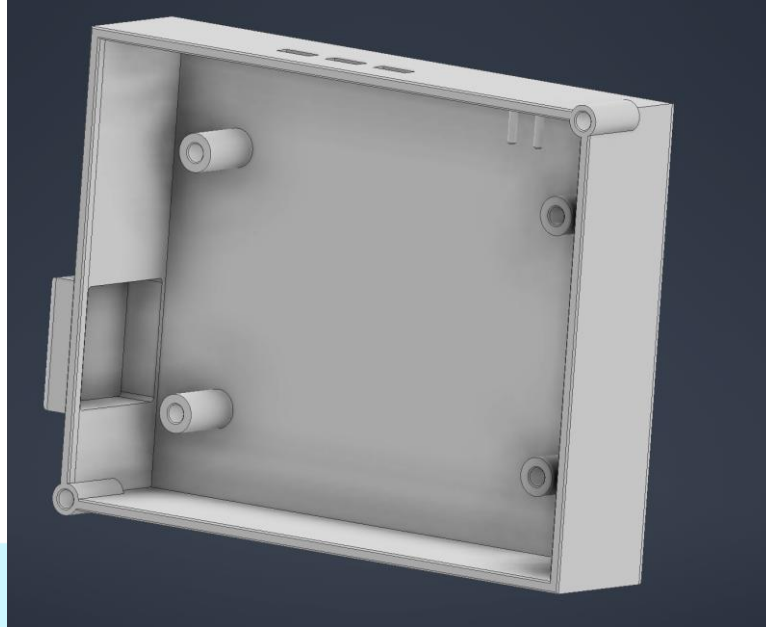


Figure 4.2: 3D CAD rendering of the base chassis, featuring integrated PCB standoffs, a micro limit switch mount for tamper detection, and cutouts for switch ports.

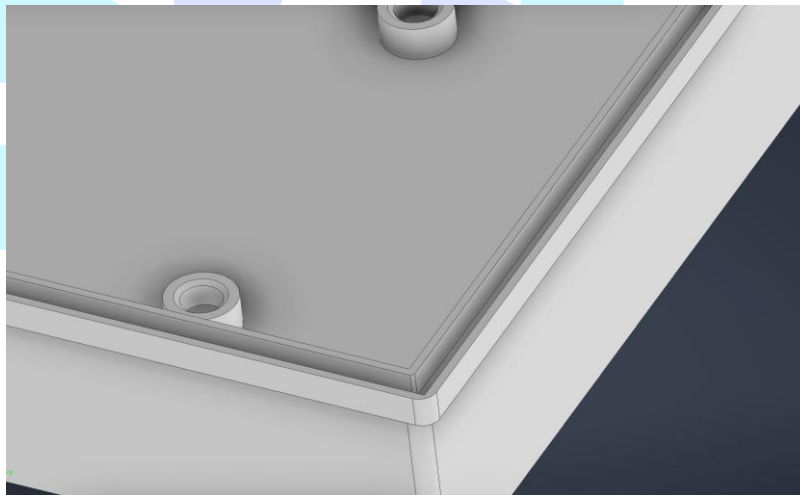
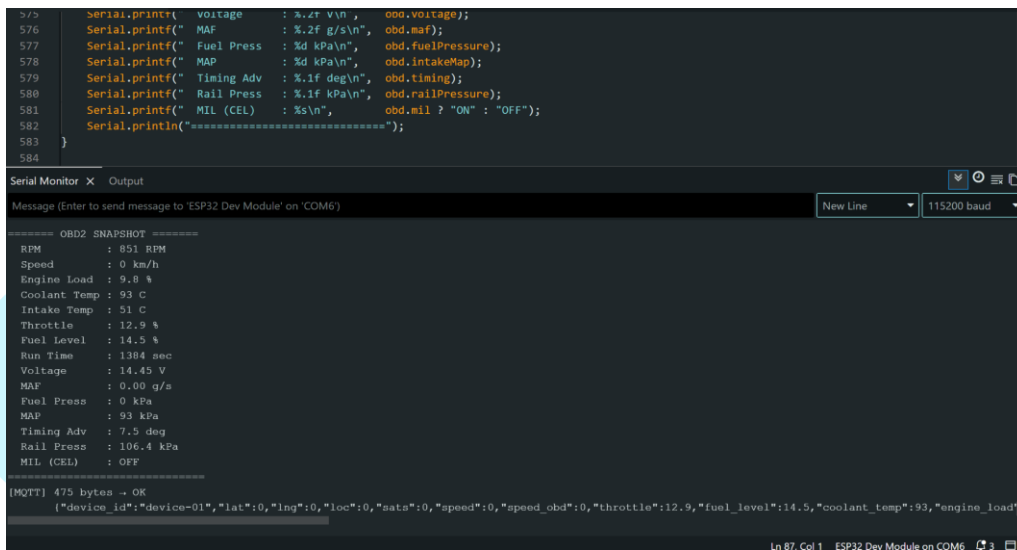


Figure 4.3: Close-up 3D CAD rendering of the enclosure's interlocking lip-and-groove mating edge, designed to provide a water-resistant barrier for internal electronics.

4.1.2. OBD-II Integration

The secondary ESP32 was successfully interfaced with the vehicle's CAN Bus utilizing the MCP2515 module. Preliminary bench and vehicle testing confirmed that the node accurately parses critical engine telemetry. To ensure low-latency and reliable local wireless

communication without the need for a traditional Wi-Fi network, the extracted data is transmitted from the secondary sensor node to the main security node utilizing the ESP-NOW protocol. As demonstrated in the serial monitor output (see Figure 4.4), the system successfully captures a comprehensive snapshot of real-time parameters. These metrics specifically include Engine RPM, Vehicle Speed, Engine Load, Coolant Temperature, Intake Temperature, Throttle Position, Fuel Level, and System Voltage. Furthermore, the output confirms that the secondary node accurately compiles these parameters into a structured JSON payload, successfully preparing them for aggregation and transmission by the main security node.



```

575 Serial.println( voltage      : %.2f V\n",    oob.voltage);
576 Serial.printf(" MAF          : %.2f g/s\n",    oob.maf);
577 Serial.printf(" Fuel Press   : %d kPa\n",    oob.fuelPressure);
578 Serial.printf(" MAP          : %d kPa\n",    oob.intakeMap);
579 Serial.printf(" Timing Adv   : %.1f deg\n",    oob.timing);
580 Serial.printf(" Rail Press   : %.1f kPa\n",    oob.railPressure);
581 Serial.printf(" MIL (CEL)    : %s\n",    oob.mil ? "ON" : "OFF");
582 Serial.println("=====");
583 }
584

```

Serial Monitor X Output

Message (Enter to send message to 'ESP32 Dev Module' on 'COM6')

New Line 115200 baud

```

===== OBD2 SNAPSHOT =====
RPM      : 851 RPM
Speed    : 0 km/h
Engine Load : 9.8 %
Coolant Temp : 93 C
Intake Temp : 51 C
Throttle  : 12.9 %
Fuel Level : 14.5 %
Run Time  : 1384 sec
Voltage   : 14.45 V
MAF       : 0.00 g/s
Fuel Press : 0 kPa
MAP       : 93 kPa
Timing Adv : 7.5 deg
Rail Press : 106.4 kPa
MIL (CEL) : OFF

[MQTT] 475 bytes -> OK
{"device_id":"device-01","lat":0,"lng":0,"loc":0,"sats":0,"speed":0,"speed_obd":0,"throttle":12.9,"fuel_level":14.5,"coolant_temp":93,"engine_load":

```

Ln 87, Col 1 ESP32 Dev Module on COM6

Figure 4.4: Serial debugging monitor output from the secondary microcontroller, confirming the successful extraction and parsing of real-time OBD-II vehicle metrics. The log also demonstrates the compiled data being successfully formatted into a structured JSON payload for wireless transmission to the primary node.

4.1.3. PCB Integration and Hardware Validation

Before final deployment, the integrated electrical schematic was rigorously validated and transitioned into a custom Printed Circuit Board (PCB). As illustrated in the PCB layout (see Figure 4.5), this custom design efficiently routes the ESP32-C3 microcontroller, the upgraded SIM7600G LTE modem, and dedicated terminal connections for the physical tamper switches and power inputs. This setup confirmed the final pin configurations and signal integrity while validating the system's ability to save data directly to the ESP32's internal flash memory, ensuring operational stability without the need for external storage modules.

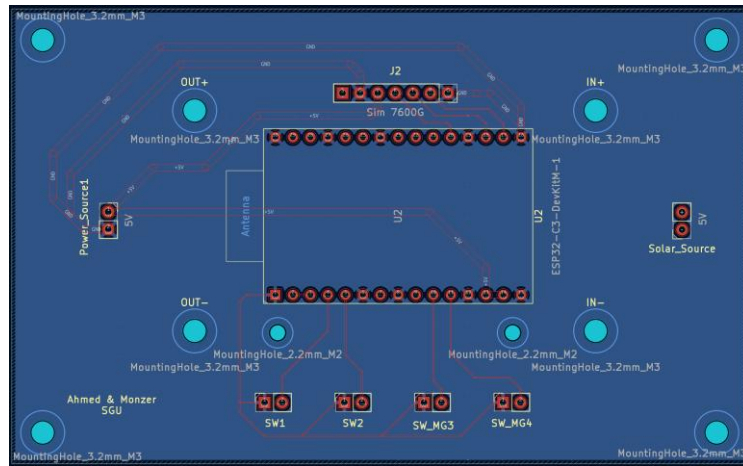


Figure 4.5: Custom PCB schematic detailing the ESP32-C3, SIM7600G integration, 4 switches, and buck boost holder in the back of the board.

The physical realization of this PCB was successfully populated and mounted within the custom water-resistant enclosure (see Figure 4.6). This integration demonstrates the secure internal wiring of the active tamper switches, power inputs, and communication antennas, alongside the integrated micro limit switch designed to detect enclosure tampering.

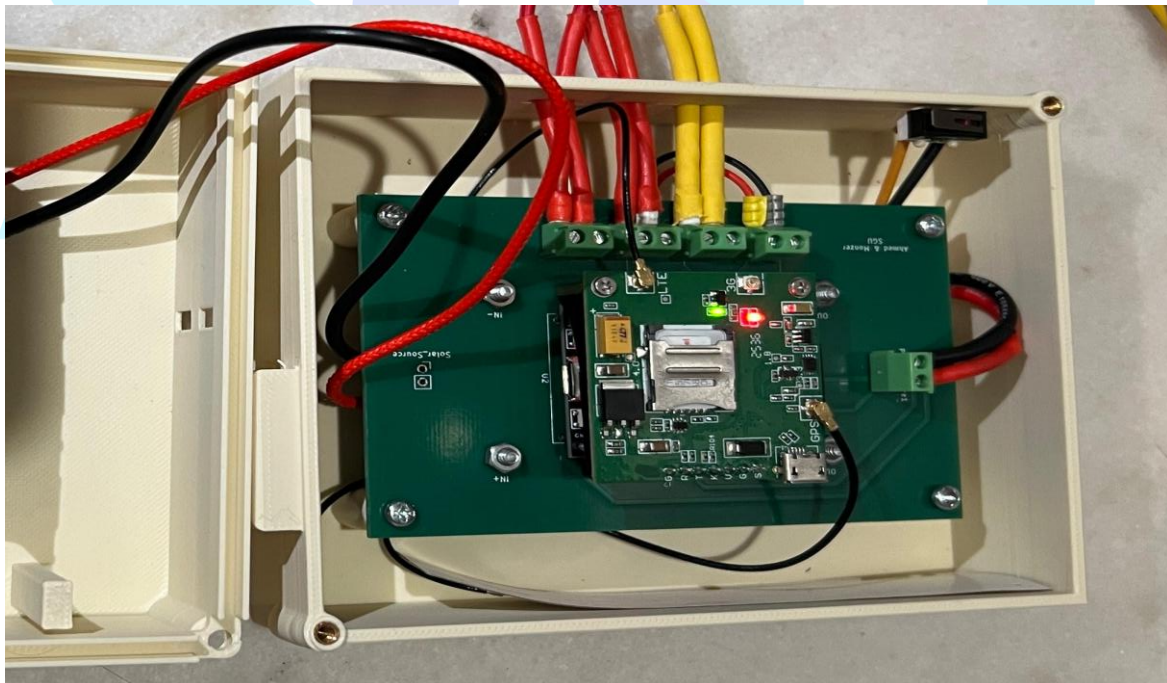


Figure 4.6: Populated PCB integrated within the custom 3D-printed enclosure

Finally, the complete hardware architecture was assembled to validate the independent power system. The final device configuration (see Figure 4.7) 1S3P Lithium-Ion battery pack, a

solar charge controller paired with the external solar panel, and an auxiliary charging module that draws power directly from the vehicle's OBD-II port. This dual-input charging architecture confirms the physical readiness of the system for extended autonomous field deployment.

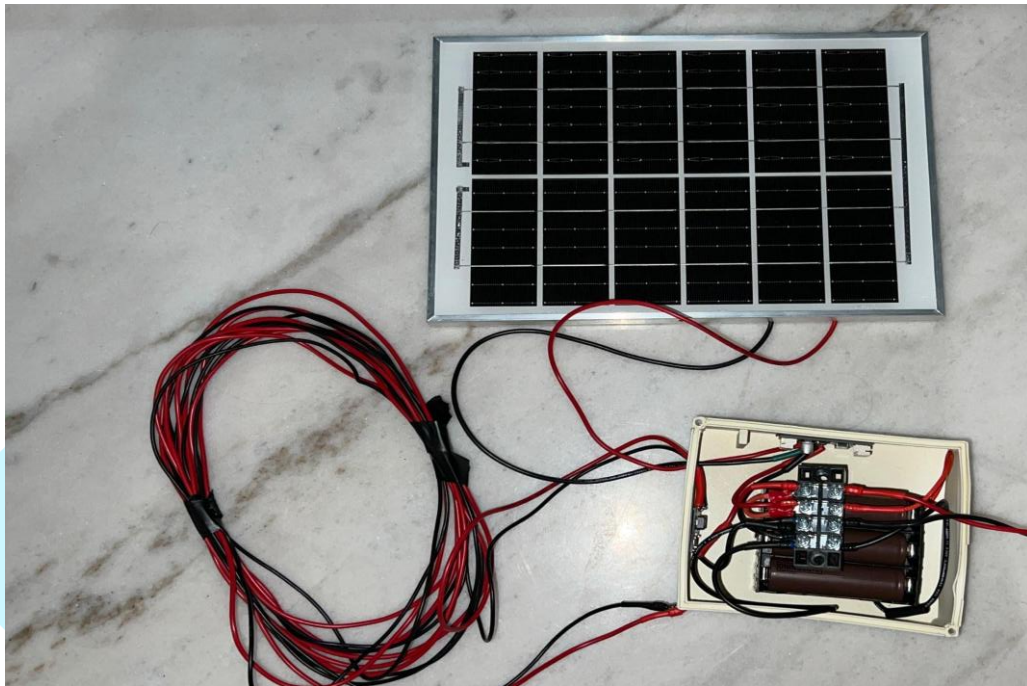


Figure 4.7: Complete smart telemetry device with solar panel and independent battery power supply

4.2. Firmware Optimization and System Stability

4.2.1. Power Optimization and Global Time Sync

Significant improvements were made to the ESP32-C3 firmware to reduce baseline power consumption. By implementing dynamic sleep logic, reducing the CPU frequency, and shutting down unnecessary peripherals (Wi-Fi and Bluetooth), the system's active current consumption was successfully optimized to just 100mA—330mA at 5V. To further extend battery autonomy, a low-power "heartbeat mode" was integrated into the system's architecture. If the OBD-II connection is lost or disconnected for five consecutive minutes, the device automatically transitions into this deep sleep state, significantly dialing back network activity to transmit location updates only once per hour. Crucially, the system remains actively responsive to hardware interrupts during this prolonged sleep period; if any physical tamper switch is triggered, the microcontroller wakes instantaneously to process and transmit the alarm. Additionally, the firmware dynamically adjusts the size of the telemetry data payload

based on the specific parameters being sent. By eliminating the transmission of empty data fields or null packets, the system actively minimizes the duration of cellular network transmissions, which directly translates to further reductions in overall power consumption. Furthermore, to minimize power-intensive network requests, the system was programmed to extract the global time directly from the GNSS/GPS module's initial satellite connection. By fetching this time data—which is acquired instantly before a full positional lock is established—just once upon system boot, the microcontroller subsequently relies on its internal real-time counter to accurately timestamp offline telemetry data without requiring continuous module polling.

4.2.2. System Self-Healing Validation

During preliminary continuous-run tests, hardware initialization faults were simulated to evaluate the system's resilience. The implementation of the auto-reboot watchdog successfully stopped these issues. The logic actively monitored the LTE modem's hardware initialization status and successfully executed a power-cycle wake-up routine when simulated crashes were detected. Additionally, the programmed boot-up delay successfully filtered out electrical noise, eliminating the false rebooting that were initially observed during system power-on surges.

4.3. Energy System and Power Profiling

4.3.1. Battery Autonomy Validation

To ensure accurate energy profiling, an initial battery capacity test was conducted to verify the true energy storage capabilities of the 1S3P Lithium-Ion pack, rather than relying solely on the manufacturer's labeled specifications; the empirical results of this physical capacity validation are detailed in the appendix (Figure A.2).

Crucially, to guarantee that the power profiling process did not artificially inflate the energy consumption of the primary telemetry node, an isolated measurement architecture was employed. A secondary ESP32 microcontroller was dedicated strictly to interfacing with the INA226 precision current sensor, logging the primary system's battery discharge profile, and transmitting the data to a remote dashboard. Prior to continuous deployment, the INA226 sensor's accuracy was successfully validated against a digital multimeter, confirming exact measurement parity.

Following this physical verification, a continuous battery runtime test was performed to evaluate the primary system's performance operating strictly on battery power without external power input. The system was successfully operated and monitored for a continuous

10-hour period, providing empirical proof of the hardware's stability and sustained energy consumption over an extended duration (Figure 4.8). Furthermore, the firmware's safety logic was successfully validated by observing an automatic system cut-off when the battery voltage reached the 3.3V threshold, effectively protecting the cells from over-discharge.

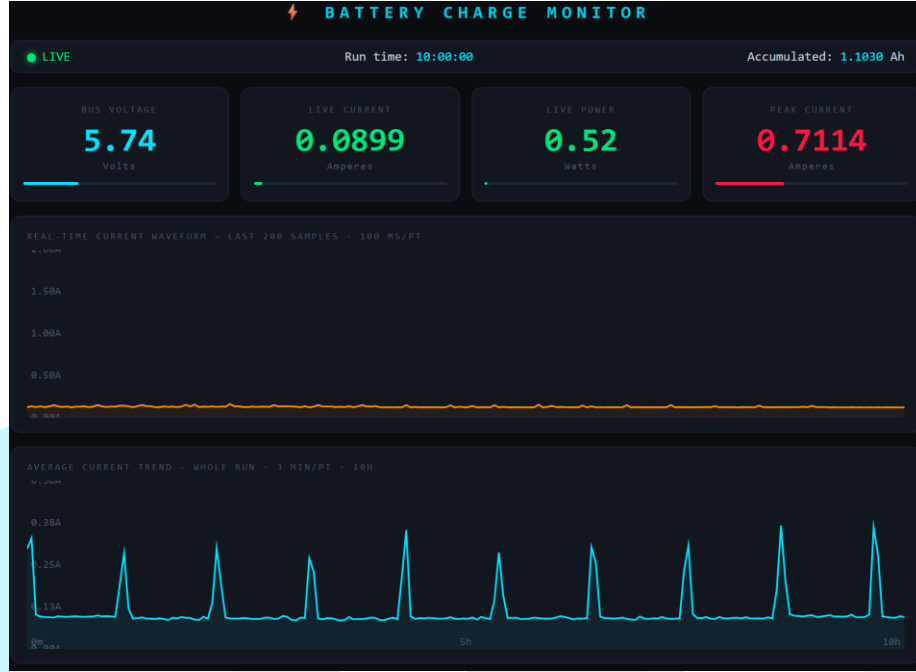


Figure 4.8: 10-hour continuous battery discharge and current consumption profile captured via the isolated INA226 precision current sensor

Based on the empirical data captured during this 10-hour test, the system consumed a total of 1.1 Ah (1100 mAh). Extrapolating this consumption rate against the fully charged 9000 mAh capacity of the 1S3P battery pack allows for the calculation of the total theoretical autonomy:

$$Autonomy_{hours} = \frac{9000mAh}{1100mAh} \times 10h \approx \mathbf{81.8 \text{ hours}} \quad (4.1)$$

Converting this value to standard operational days:

$$Autonomy_{days} = \frac{81.8h}{24h} \approx \mathbf{3.4 \text{ days}} \quad (4.2)$$

This calculation verifies that the system can sustain continuous functionality for approximately 3.4 days without any external power input, successfully exceeding the initial 48-hour design target.

To validate the system's power resilience in a real-world field environment, a MAX17048 battery fuel gauge IC was integrated into the primary architecture. This sensor enabled continuous, accurate monitoring of the battery voltage and State of Charge (SoC) percentage

while the device was mounted on the vehicle and actively interfacing with the solar panel. The system successfully operated under these dynamic charging and discharging conditions for an extended, continuous 50-hour period (Figure 4.9), confirming the long-term viability of the solar-assisted independent power supply.

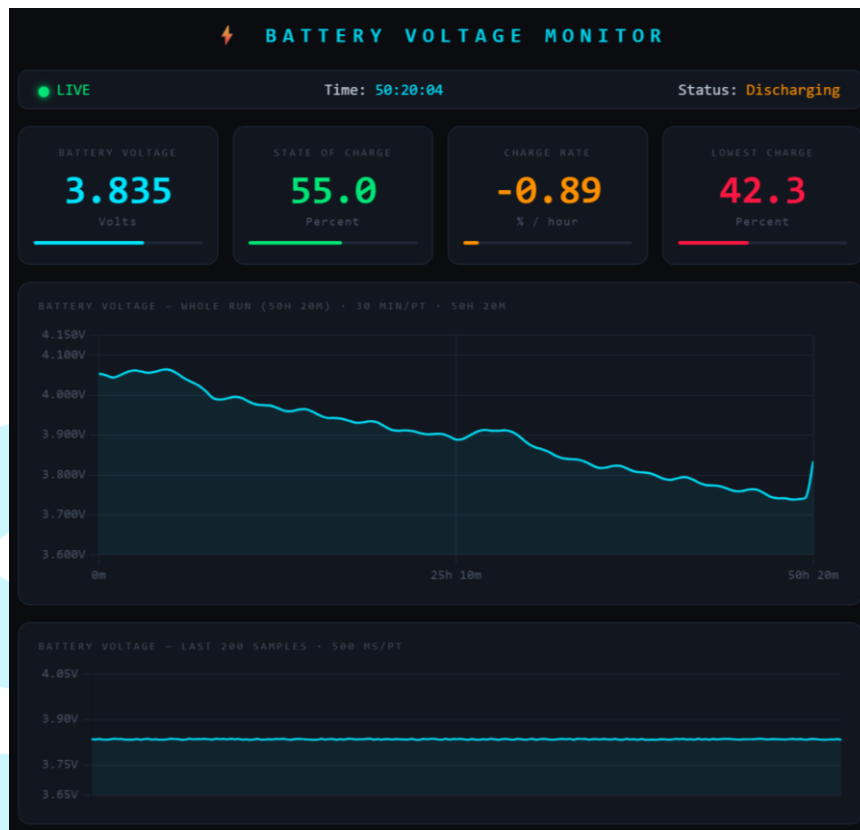


Figure 4.9: Dashboard interface displaying the 50-hour continuous field test results, capturing dynamic battery voltage trends, state of charge (SoC), and discharge rates via the MAX17048 fuel gauge IC

1. Daily Current and Power Consumption

The measured current draw during active transmission (I_{Active}) and deep sleep (I_{Sleep}) alongside their respective estimated daily durations ($T_{Active} = 16h$ and $T_{Sleep} = 8h$), the total daily current (I_{Total}) was calculated:

$$I_{Total} = (I_{Active} \times T_{Active}) + (I_{Sleep} \times T_{Sleep}) = (0.1103 \times 16) + (0.009 \times 8) = \mathbf{1.8368 \text{ Ah}} \quad (4.3)$$

The average continuous current (I_{avg}) over a standard 24-hour period is then derived:

$$I_{avg} = \frac{I_{Total}}{24h} = \frac{1.8368}{24} = \mathbf{0.0765\ A} \quad (4.4)$$

Using the measured system voltage (V) of 5V, the continuous average power consumption (P) is verified:

$$P = V \times I_{avg} = 5 \times 0.0765 = \mathbf{0.3825\ W} \quad (4.5)$$

2. Energy Autonomy and Battery Capacity

The empirical energy required to sustain the system for a full 24-hour cycle ($E_{1\ day}$) is:

$$E_{1\ day} = P \times 24h = 0.3825 \times 24 = \mathbf{9.18\ Wh / Day} \quad (4.6)$$

To achieve the targeted autonomy of 2 days ($Days = 2$), the total autonomy energy requirement ($E_{autonomy}$) is:

$$E_{autonomy} = E_{1\ day} \times Days = 9.18 \times 2 = \mathbf{18.36\ Wh / 2\ Days} \quad (4.7)$$

To protect the Lithium-Ion cells from over-discharge degradation, a safe Depth of Discharge (DoD) of 80% was applied. The true required battery energy ($E_{Battery}$) is:

$$E_{Battery} = \frac{E_{autonomy}}{DoD} = \frac{18.36}{0.8} = \mathbf{22.95\ Wh} \quad (4.8)$$

The minimum required battery capacity ($C_{Battery}$) to sustain this target is calculated:

$$C_{Battery} = \frac{E_{autonomy}}{V_{nominal}} = \frac{22.95}{3.7} = \mathbf{6.203\ Ah \approx 6200\ mAh} \quad (4.9)$$

The physical 1S3P battery pack provides a measured capacity of **9000 mAh**. Comparing the theoretical requirement against this physical capacity confirms that the hardware design successfully meets more than 48-hour autonomous operation targets under real-world, safety-rated conditions.

4.3.2. Oscilloscope Power Profiling

Real-time power states were visualized using a digital oscilloscope to evaluate the system's energy consumption across three distinct operational profiles: active standby (no transmission), routine GPS telemetry, and high-load continuous transmission.

First, the system was monitored in an active standby state, where the microcontroller and modules were powered and listened, but no data was actively being transmitted to the cloud. The resulting waveform (Figure 4.10) demonstrates a highly consistent, low-power square wave pattern indicative of efficient background polling and connection maintenance. Small current spikes reaching 327mA were observed as briefly cycled to maintain essential readiness, followed immediately by a drop to a highly stable 54.9mA baseline between these background cycles.

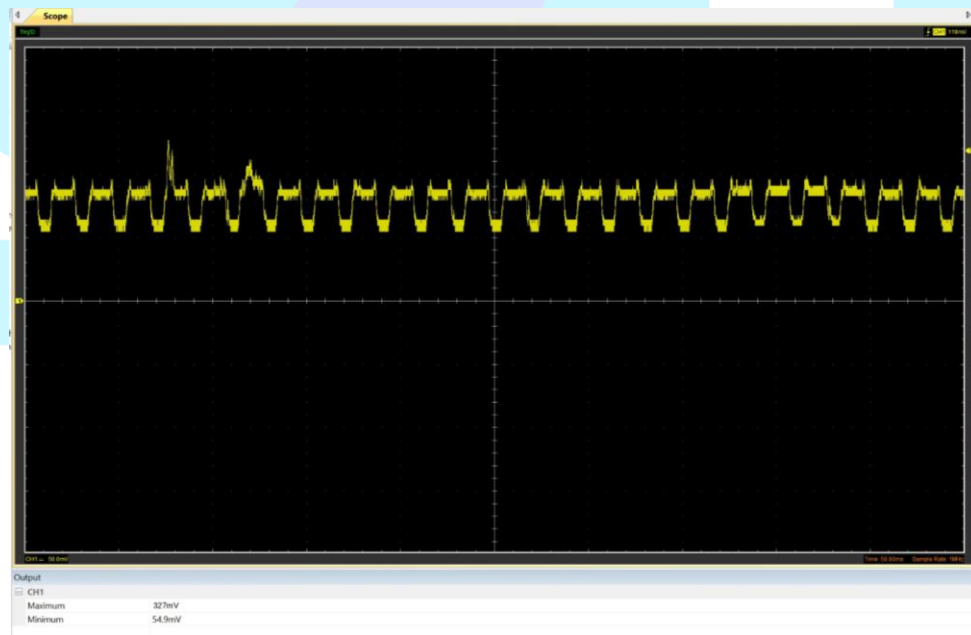


Figure 4.10: Oscilloscope waveform illustrating system power consumption during an active standby state without transmission.

Second, the power profile was captured during routine telemetry operations, specifically when the system actively acquired and transmitted standard GPS coordinates without triggering external alarms. As shown in Figure 4.11, the waveform displays consistent, repetitive functional behavior. Distinct current spikes reaching 380mA were

observed during the active network transmission phases, dropping back to a 91mA baseline when the system rested between transmission intervals.

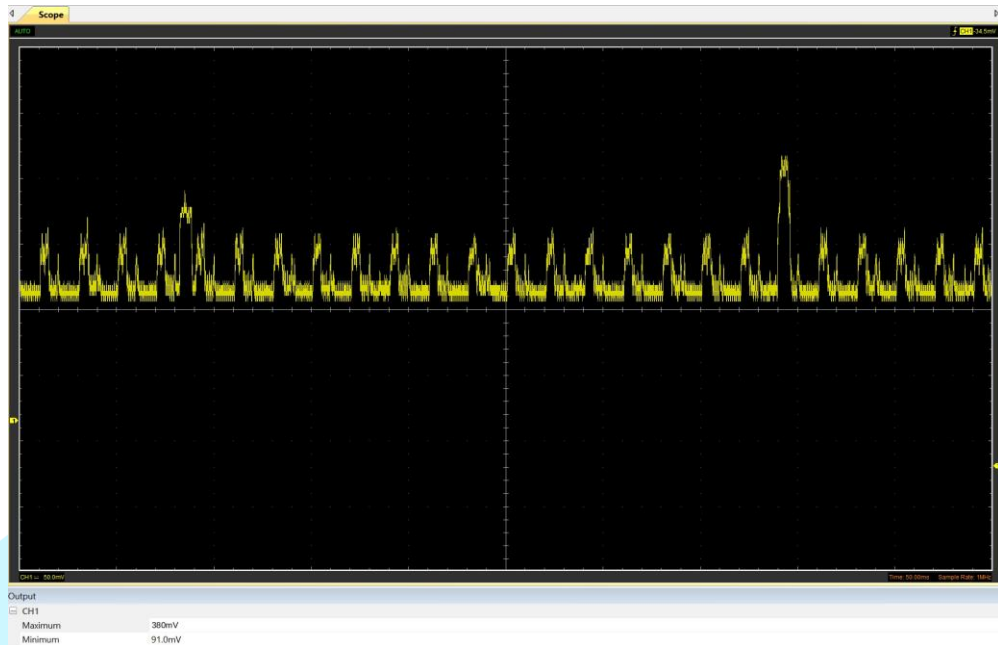


Figure 4.11: Oscilloscope waveform illustrating system power states during routine, periodic GPS telemetry transmission.

Finally a stress test was conducted to simulate a high-load continuous transmission event. Multiple hardware interrupts were triggered by rapidly and repeatedly actuating the physical tamper switches. This forced the ESP32 into a highly active state, constantly processing physical interrupts and pushing dense alarm payloads over the network. The waveform in Figure 4.12 clearly demonstrates this intensive operational state, showing continuous, broad current spikes peaking at 407mA during these heavy processing and transmission phases, followed by a drop to a 119mA baseline only when the system was allowed to recover and re-enter a resting state.

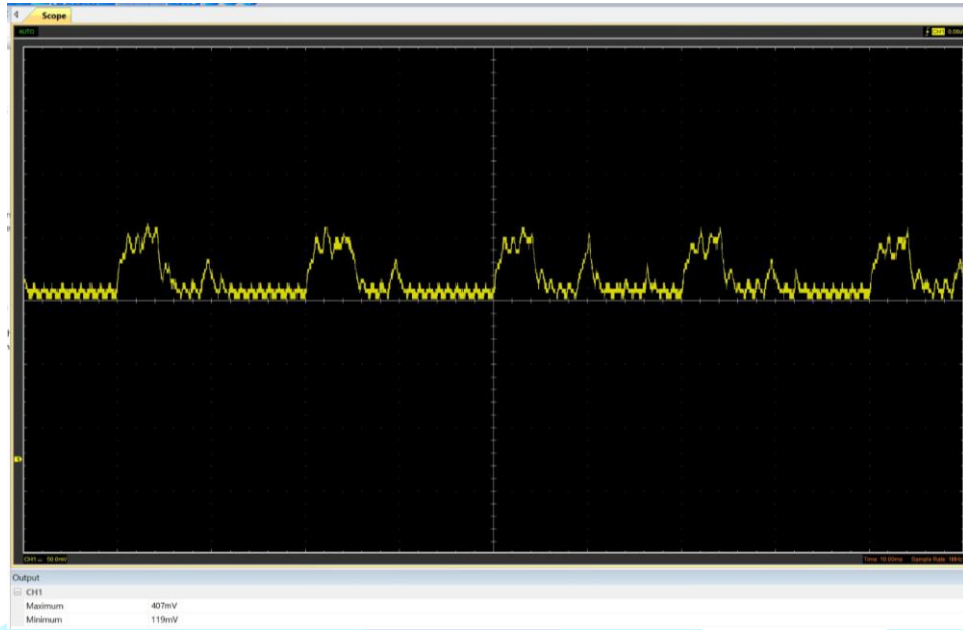


Figure 4.12: Oscilloscope waveform illustrating system power states during a high-load transmission test, triggered by continuous tamper switch actuations.

4.3.3. Solar Charging Performance and Hardware Iterations

To ensure the renewable energy architecture can sustain the system's daily load and recharge the autonomy deficit despite these weather fluctuations, the required solar panel wattage (P_{Solar}) was calculated based on the models established in Section 3.3.1. Assuming an average of 4 Peak Sun Hours (PSH) for the deployment region and an estimated system loss factor (L) of 0.53 the requirement is:

$$P_{Solar} = \frac{E_{autonomy}}{PSH \times L} = \frac{18.36}{4 \times 0.53} = 8.66 \approx 9 \text{ W} \quad (4.10)$$

To validate the renewable energy architecture, solar charging performance tests were conducted under varying environmental conditions. A dedicated ESP32 monitoring node, interfaced with an INNA226 precision current sensor, was utilized to log the solar panel's current delivery to the system.

During optimal sunny conditions, the solar panel demonstrated a strong and stable charging profile, maintaining a live current generation of approximately 0.46A and reaching peak currents of 0.54A (Figure 4.13). Conversely, during cloudy conditions, the generation capacity significantly decreased, with the panel delivering a fluctuating average of 0.17A and peaking at just 0.30A (Figure 4.14).

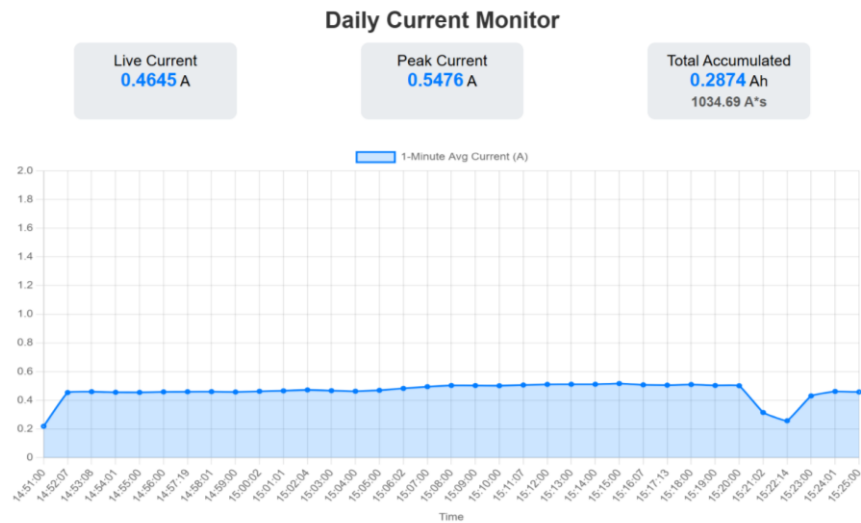


Figure 4.13: Solar Current Generation during Sunny Conditions

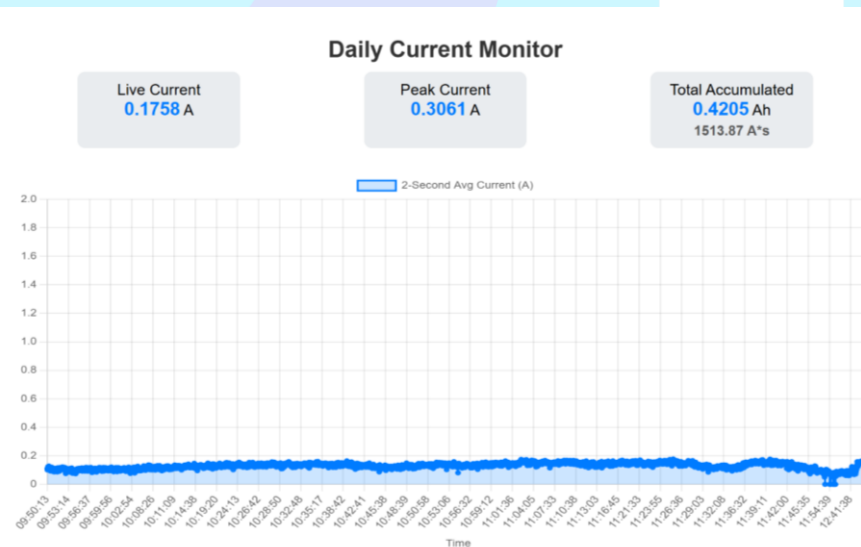


Figure 4.14: Solar Current Generation during Cloudy Conditions

Furthermore, to establish the absolute upper performance limit of the energy harvesting architecture, an additional peak-yield test was conducted under direct, unobstructed sunlight. During this concentrated observation period of approximately one hour, the system successfully recorded a maximum peak current of 0.68A. As illustrated in the performance dashboard (see Figure 4.15), this surge in generation significantly exceeds the standard stable averages. This demonstrates the solar panel's robust charging capability

when exposed to maximum solar irradiance, confirming its ability to rapidly replenish the battery reserve during brief windows of optimal weather conditions.



Figure 4.15: Dashboard interface illustrating the maximum solar current generation under direct sunlight, achieving a recorded peak of 0.68A

4.4. Telemetry and Communication Validation

Following the validation of the power architecture, the system's data acquisition and communication pipelines were evaluated. This preliminary validation focused on ensuring the hardware modules could accurately acquire physical data and that the microcontroller could successfully format these localized payloads for cloud ingestion.

4.4.1. GPS Accuracy and Field Testing

To validate the location-tracking capabilities, field tests were conducted to evaluate the GPS module's satellite acquisition time and dynamic coordinate accuracy. Under open-sky conditions, the module successfully achieved a stable satellite lock within standard operational parameters. To confirm spatial accuracy during real-world movement, the acquired latitude and longitude coordinates were continuously transmitted to a custom tracking dashboard. The dashboard interface successfully plotted the device's physical trajectory, verifying that the system accurately captures and reports live location data over time without significant geographic drift or signal loss (Figure 4.16).

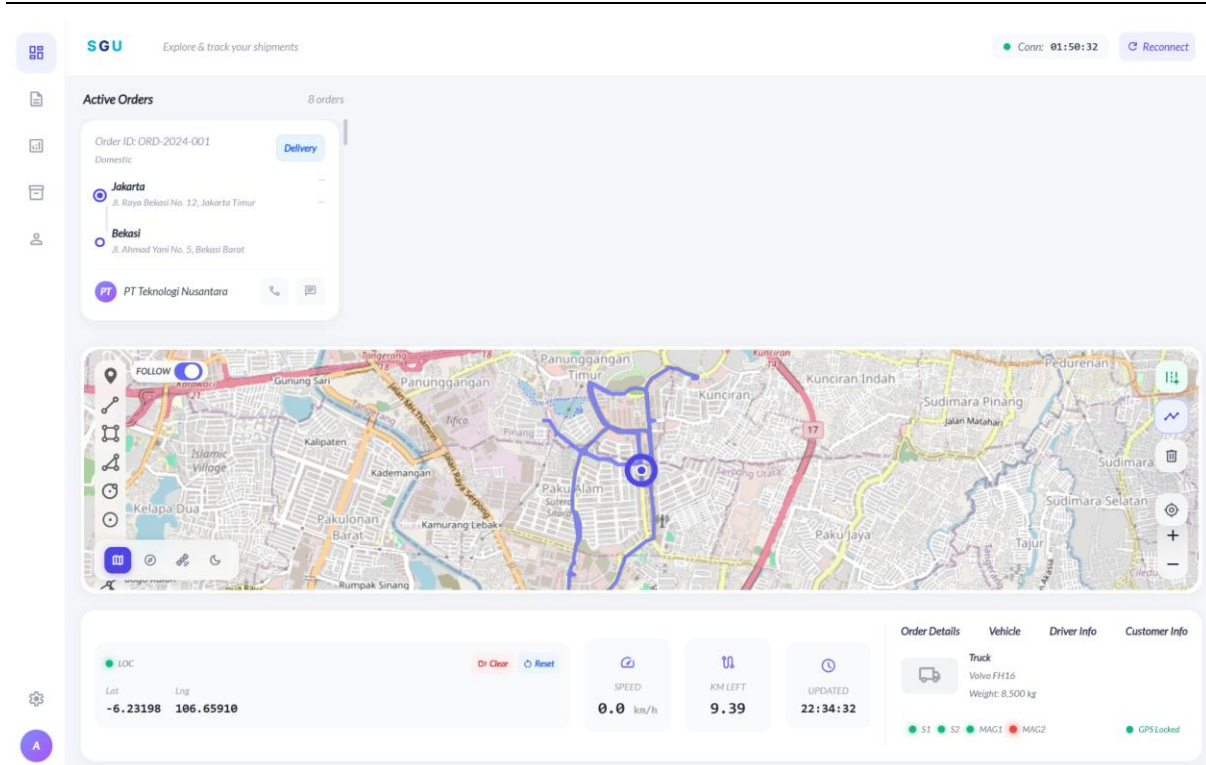


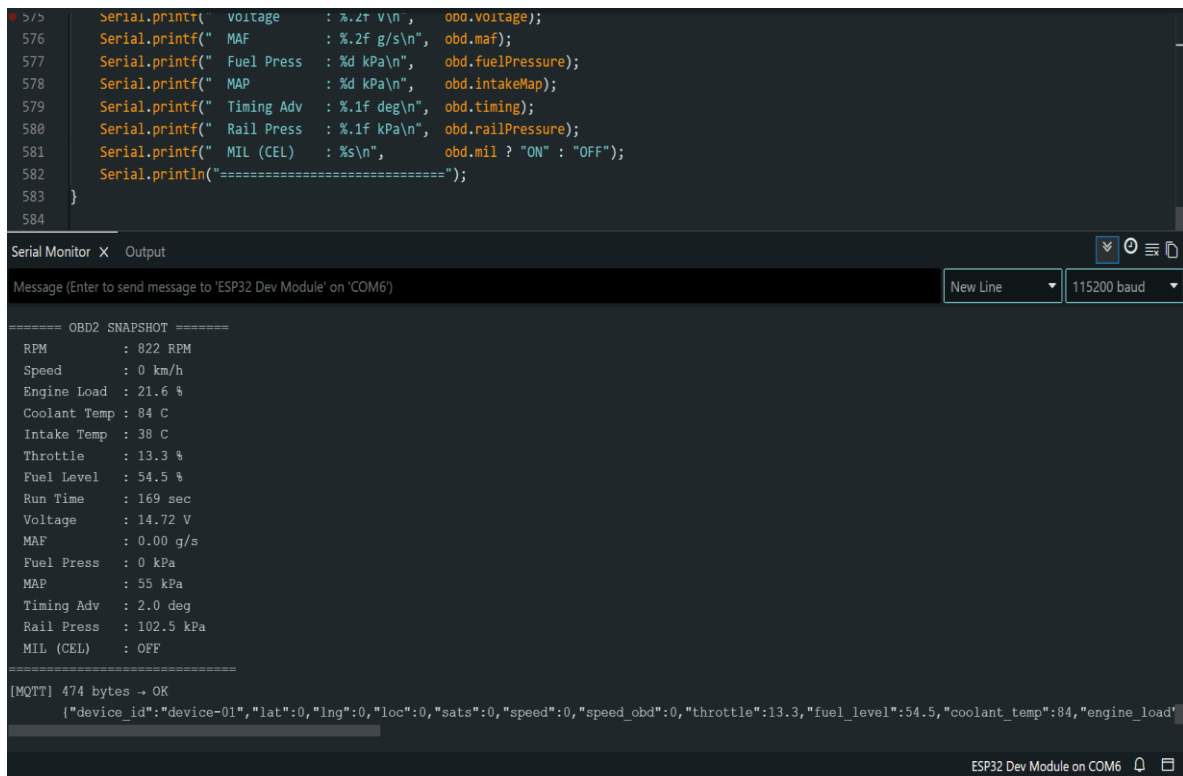
Figure 4.16: Dashboard interface visualizing the system's live GPS trajectory and spatial movement validation.

4.4.2. Data Pipeline and Payload Validation

The core function of the primary telemetry node (ESP32-C3) is to act as a central aggregator for disparate data streams. Utilizing a distributed processing architecture, the main ESP32-C3 receives pre-processed OBD-II vehicle telemetry from a dedicated secondary microcontroller, while directly acquiring spatial coordinates from the GPS module and monitoring physical tamper alerts via hardware interrupts. To validate this multi-node software pipeline, a real-world functional test was conducted on the vehicle.

During physical testing, the secondary microcontroller successfully established a communication protocol with the vehicle's ECU. It systematically polled the OBD-II interface and successfully extracted critical real-time vehicle metrics, specifically: Engine RPM, Vehicle Speed, Fuel Level, Engine Coolant Temperature, and Engine Load.

To verify the integrity of this initial extraction process, the raw metric data was outputted to the serial debugging interface of the secondary microcontroller. Inspecting the serial console confirmed that all requested PID (Parameter ID) responses from the ECU were accurately captured, parsed, and translated into readable numerical formats (Figure 4.17).



```
575 Serial.println(" Voltage      : %.2f V\n", obd.voltage);
576 Serial.printf(" MAF          : %.2f g/s\n", obd.maf);
577 Serial.printf(" Fuel Press   : %d kPa\n", obd.fuelPressure);
578 Serial.printf(" MAP           : %d kPa\n", obd.intakeMap);
579 Serial.printf(" Timing Adv   : %.1f deg\n", obd.timing);
580 Serial.printf(" Rail Press   : %.1f kPa\n", obd.railPressure);
581 Serial.printf(" MIL (CEL)    : %s\n", obd.mil ? "ON" : "OFF");
582 Serial.println("=====");
583 }
584
```

Serial Monitor X Output

Message (Enter to send message to 'ESP32 Dev Module' on 'COM6') New Line 115200 baud

```
===== OBD2 SNAPSHOT =====
RPM      : 822 RPM
Speed     : 0 km/h
Engine Load : 21.6 %
Coolant Temp : 84 C
Intake Temp : 38 C
Throttle   : 13.3 %
Fuel Level  : 54.5 %
Run Time   : 169 sec
Voltage     : 14.72 V
MAF         : 0.00 g/s
Fuel Press  : 0 kPa
MAP         : 55 kPa
Timing Adv  : 2.0 deg
Rail Press  : 102.5 kPa
MIL (CEL)   : OFF
=====
[MQTT] 474 bytes -> OK
{"device_id":"device-01","lat":0,"lng":0,"loc":0,"sats":0,"speed":0,"speed_obd":0,"throttle":13.3,"fuel_level":54.5,"coolant_temp":84,"engine_load":21.6,"mil":"OFF"}
```

ESP32 Dev Module on COM6

Figure 4.17: serial debugging output from the secondary microcontroller, verifying the successful extraction and parsing of live OBD-II vehicle metrics

Following this local validation, the system's end-to-end transmission capabilities were tested. The secondary node successfully transmitted the structured OBD-II payload to the primary ESP32-C3 utilizing the low-latency ESP-NOW wireless protocol. The primary node then aggregated this telemetry and successfully relayed it to the cloud server via the LTE network. The integrity of this complete pipeline was definitively proven by verifying the live vehicle metrics seamlessly updating and displaying accurately on the remote fleet dashboard (Figure 4.18).

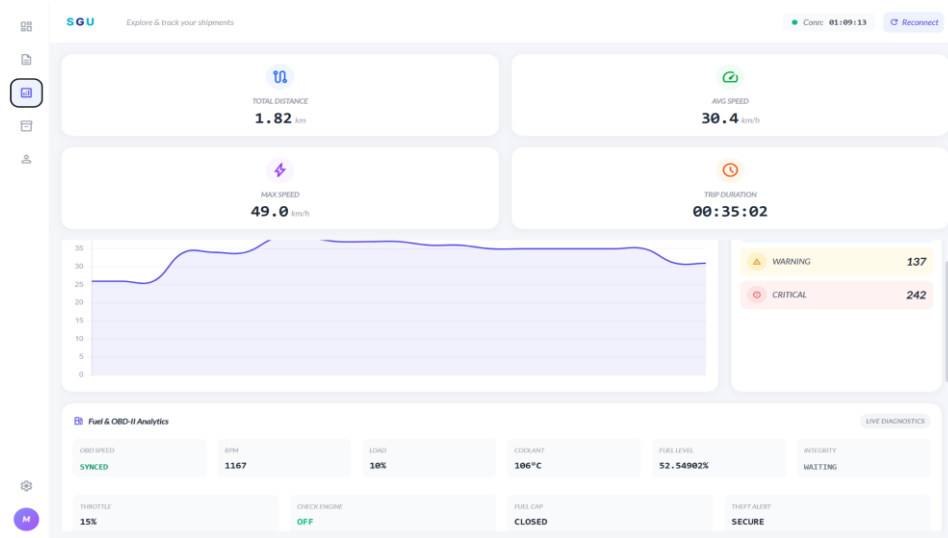


Figure 4.18: Cloud dashboard interface displaying live OBD-II telemetry metrics, verifying successful end-to-end data transmission via ESP-NOW and LTE aggregation

4.4.3. Packet Delivery Ratio (PDR) and Network Reliability

To mathematically quantify the reliability of the telemetry communication pipeline, the Packet Delivery Ratio (PDR) was evaluated. PDR is a critical performance metric for IoT architectures—particularly those handling security alerts and vehicle diagnostics—as it determines the percentage of successfully delivered data packets compared to the total number generated by the system. The PDR is calculated using the following equation:

$$PDR (\%) = \frac{N_{received}}{N_{generated}} \times 100 \quad (4.11)$$

Where:

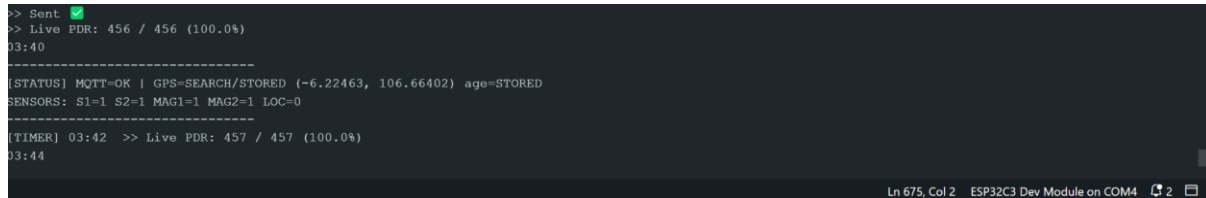
$N_{received}$: The total number of data packets successfully received and logged by the cloud.

$N_{generated}$: The total number of data packets compiled and transmitted by the primary ESP32-C3 node.

To empirically validate the system's network stability, a continuous transmission test was conducted over an 8-hour operational period. The system was programmed to consistently generate and transmit aggregated JSON telemetry payloads at predefined intervals. Upon conclusion of the test, the local transmission logs from the microcontroller were cross-referenced against the reception logs on the test dashboard.

The data confirmed that exactly zero packets were dropped or corrupted during transit. The system achieved a 100% Packet Delivery Ratio throughout the entire 8-hour duration (Figure

4.19). This flawless transmission rate demonstrates exceptional software stability, proving that the microcontroller's packet-handling and transmission logic is highly robust and fully prepared for deployment.

A screenshot of a serial monitor interface showing the output of an ESP32-C3 module. The background is dark with light-colored text. The output includes status messages like '[STATUS] MQTT=OK', GPS coordinates, sensor data, and a 'Live PDR' (Packet Delivery Ratio) of 100.0% at two different times (03:40 and 03:42). The bottom status bar indicates 'Ln 675, Col 2' and 'ESP32C3 Dev Module on COM4'.

```
>> Sent ✓
>> Live PDR: 456 / 456 (100.0%)
03:40
-----
[STATUS] MQTT=OK | GPS=SEARCH/STORED (-6.22463, 106.66402) age=STORED
SENSORS: S1=1 S2=1 MAG1=1 MAG2=1 LOC=0
-----
[TIMER] 03:42 >> Live PDR: 457 / 457 (100.0%)
03:44
```

Figure 4.19: Serial monitor output from the ESP32-C3 primary node during continuous field testing, empirically validating a 100% Packet Delivery Ratio (457/457 successfully transmitted payloads) alongside active sensor states and MQTT network stability.

4.4.4. Offline Data Logging and Recovery Validation

To guarantee data continuity in areas with poor cellular coverage, the system's offline logging and auto-recovery logic were empirically validated. A controlled network outage was simulated by temporarily disabling the LTE modem's connection while the system actively generated vehicle telemetry.

During this simulated outage, the primary ESP32-C3 successfully detected the network loss and automatically routed the aggregated JSON telemetry payloads to its internal flash memory. To verify the integrity of this local storage mechanism, the serial debugging interface was monitored. The console output confirmed that the system accurately formatted the data and successfully wrote it to the internal flash sector without encountering memory overflow or formatting errors (Figure 4.20).

```

Output  Serial Monitor X
[Message (Enter to send message to 'ESP32C3 Dev Module' on 'COM5')] New Line 115200 baud

--- OFFLINE DATA LOG ---
[Alarm Events]
[Telemetry / Sensor Events]
[DATA] ("seq":1,"ts_ms":17526,"device_id":"monztrack-01","event":"sensor_alert","gps_fix":false,"gps_has_fix":false,"gps_has_stored":true,"gps_age_ms":
[DATA] ("seq":2,"ts_ms":17803,"device_id":"monztrack-01","event":"sensor_alert","gps_fix":false,"gps_has_fix":false,"gps_has_stored":true,"gps_age_ms":
[DATA] ("seq":3,"ts_ms":18673,"device_id":"monztrack-01","event":"sensor_alert","gps_fix":false,"gps_has_fix":false,"gps_has_stored":true,"gps_age_ms":
[DATA] ("seq":4,"ts_ms":18934,"device_id":"monztrack-01","event":"sensor_alert","gps_fix":false,"gps_has_fix":false,"gps_has_stored":true,"gps_age_ms":
[DATA] ("seq":5,"ts_ms":28970,"device_id":"monztrack-01","event":"sensor_alert","gps_fix":false,"gps_has_fix":false,"gps_has_stored":true,"gps_age_ms":
[DATA] ("seq":6,"ts_ms":32145,"device_id":"monztrack-01","event":"sensor_alert","gps_fix":false,"gps_has_fix":false,"gps_has_stored":true,"gps_age_ms":
[DATA] ("seq":7,"ts_ms":41542,"device_id":"monztrack-01","event":"telemetry","gps_fix":true,"gps_has_fix":true,"gps_has_stored":true,"gps_age_ms":1421;
[DATA] ("seq":8,"ts_ms":68397,"device_id":"monztrack-01","event":"sensor_alert","gps_fix":true,"gps_has_fix":true,"gps_has_stored":true,"gps_age_ms":2;
[DATA] ("seq":9,"ts_ms":68649,"device_id":"monztrack-01","event":"sensor_alert","gps_fix":true,"gps_has_fix":true,"gps_has_stored":true,"gps_age_ms":2;
[DATA] ("seq":10,"ts_ms":69072,"device_id":"monztrack-01","event":"sensor_alert","gps_fix":true,"gps_has_fix":true,"gps_has_stored":true,"gps_age_ms":;
[DATA] ("seq":11,"ts_ms":69291,"device_id":"monztrack-01","event":"sensor_alert","gps_fix":true,"gps_has_fix":true,"gps_has_stored":true,"gps_age_ms":;
[DATA] ("seq":12,"ts_ms":69817,"device_id":"monztrack-01","event":"sensor_alert","gps_fix":true,"gps_has_fix":true,"gps_has_stored":true,"gps_age_ms":;
[DATA] ("seq":13,"ts_ms":70055,"device_id":"monztrack-01","event":"sensor_alert","gps_fix":true,"gps_has_fix":true,"gps_has_stored":true,"gps_age_ms":;
[DATA] ("seq":14,"ts_ms":75578,"device_id":"monztrack-01","event":"sensor_alert","gps_fix":true,"gps_has_fix":true,"gps_has_stored":true,"gps_age_ms":;
[DATA] ("seq":15,"ts_ms":76237,"device_id":"monztrack-01","event":"sensor_alert","gps_fix":true,"gps_has_fix":true,"gps_has_stored":true,"gps_age_ms":;
[DATA] ("seq":16,"ts_ms":76676,"device_id":"monztrack-01","event":"sensor_alert","gps_fix":true,"gps_has_fix":true,"gps_has_stored":true,"gps_age_ms":;
[DATA] ("seq":17,"ts_ms":78527,"device_id":"monztrack-01","event":"sensor_alert","gps_fix":true,"gps_has_fix":true,"gps_has_stored":true,"gps_age_ms":;
[DATA] ("seq":18,"ts_ms":78847,"device_id":"monztrack-01","event":"sensor_alert","gps_fix":true,"gps_has_fix":true,"gps_has_stored":true,"gps_age_ms":;
[DATA] ("seq":19,"ts_ms":79170,"device_id":"monztrack-01","event":"sensor_alert","gps_fix":true,"gps_has_fix":true,"gps_has_stored":true,"gps_age_ms":;
[DATA] ("seq":20,"ts_ms":79170,"device_id":"monztrack-01","event":"sensor_alert","gps_fix":true,"gps_has_fix":true,"gps_has_stored":true,"gps_age_ms":;
[DATA] ("seq":21,"ts_ms":79284,"device_id":"monztrack-01","event":"sensor_alert","gps_fix":true,"gps_has_fix":true,"gps_has_stored":true,"gps_age_ms":;
Ln 932, Col 1 ESP32C3 Dev Module on COM5

```

Figure 4.20: Serial debugging output demonstrating live telemetry data being successfully routed and saved to the ESP32 internal flash memory during a simulated network outage

Following the offline logging validation, the LTE connection was restored to evaluate the system's self-healing and data recovery pipeline. Upon re-establishing a stable cellular link, the firmware successfully executed its recovery protocol. The ESP32-C3 systematically retrieved the stored telemetry logs from the internal flash memory and transmitted them to the remote server chronologically. The success of this recovery phase was confirmed by cross-referencing the uploaded data against the fleet dashboard, which accurately displayed the historical data points generated during the offline period (Figure 4.21). This test definitively proves that the internal flash memory architecture provides a highly reliable fail-safe, ensuring complete data preservation during temporary network disruptions.

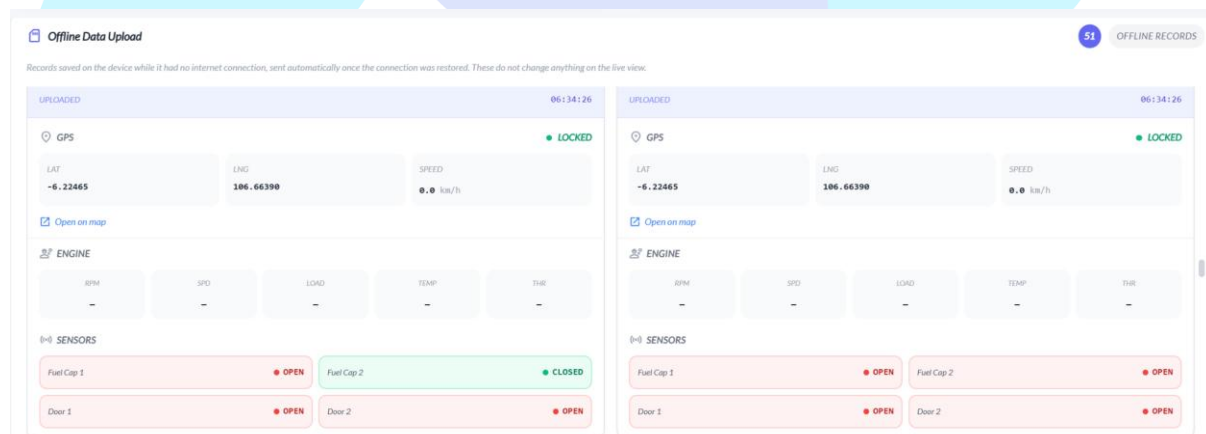


Figure 4.21: Cloud dashboard interface displaying the recovered historical telemetry data, confirming the successful synchronization of logs stored in the internal flash memory

4.4.5. Non-Volatile Storage Capacity and Flash Endurance Analysis

To mathematically validate the long-term viability of the ESP32's internal memory as a fail-safe storage medium, a capacity and endurance analysis was conducted. The system utilizes the LittleFS file system, which provides precisely 1400 KB of allocated non-volatile storage space for data persistence. Crucially, LittleFS inherently employs wear-leveling algorithms, ensuring that sequential data logging does not continuously degrade isolated memory blocks.

Based on the aggregated JSON telemetry architecture, each generated payload (containing timestamps, switch states, GPS coordinates, and OBD-II metrics) comprises approximately 250 bytes. The total theoretical record capacity ($C_{Records}$) of the allocated partition is calculated as:

$$C_{Records} = \frac{1400 \text{ KB} \times 1024 \text{ bytes/KB}}{250 \text{ bytes / records}} = 5734 \text{ records} \quad (4.12)$$

Operating at the standard active transmission interval of one payload every 30 seconds, the maximum continuous offline logging duration (T_{Fill}) before reaching memory capacity is:

$$T_{Fill} = \frac{5734 \text{ records} \times 30 \text{ s / Records}}{3600 \text{ s / h}} = 47.8 \text{ hours} \quad (4.13)$$

The physical flash memory is hardware-rated for a standard lifecycle of 100,000 erase/write cycles. Due to the wear-leveling architecture, filling the entire 1400 KB drive once constitutes approximately one uniform wear cycle across the memory blocks. The theoretical maximum offline logging lifespan ($L_{Offline}$) is therefore:

$$L_{Offline} = 100,000 \text{ cycle} \times 47.8 \text{ h/cycle} = 4,780,000 \text{ hours} \quad (4.14)$$

Converting this value to years confirms a theoretical endurance of over **545 years** of continuous offline writing.

4.5. Multi-Vehicle Field Validation and Compatibility Testing

To definitively validate the system's "plug-and-play" versatility and its ability to interface with varying Engine Control Unit (ECU) architectures, a multi-vehicle deployment test was conducted. The ESP32-based telemetry node was temporarily installed in three distinct vehicle models: a BMW 320i, a Mercedes CLA, and a BMW Z4.

4.5.1. Cross-Platform OBD-II Metric Extraction

The primary objective of this phase was to verify that the secondary microcontroller's firmware could successfully negotiate communication protocols and parse Parameter IDs

(PIDs) across different manufacturer standards. During testing, the system successfully extracted the complete suite of targeted telemetry metrics from all three vehicles.

Crucially, the system successfully acquired accurate, real-time Fuel Level percentage data directly from the OBD-II interface of all three test vehicles. This empirical success validates the system's non-intrusive design philosophy, proving that for modern, compatible fleet vehicles, accurate fuel monitoring can be achieved entirely via software polling without requiring expensive and invasive physical modifications to the vehicle's fuel tank. Comprehensive photographic evidence of the physical deployment and successful metric extraction is provided in Appendix A; specifically, the BMW Z4 results are documented in Figure A.4, Figure A.5, and Figure A.6, while the corresponding Mercedes CLA results are presented in Figure A.7, Figure A.8, and Figure A.9.

4.5.2. Extended Operational Test: BMW 320i

Following the cross-compatibility validation, an extended continuous field test was initiated using the BMW 320i to evaluate the system's sustained performance during long-duration driving cycles.

During this extended deployment, the system continuously acquired live ECU data, synchronized it with dynamic GPS spatial coordinates, and seamlessly transmitted the aggregated JSON payloads to the cloud dashboard via the LTE network. The telemetry node effectively handled fluctuating vehicle speeds, variable RPM loads, and continuous fuel consumption tracking without experiencing software crashes, memory buffer overflows, or ECU polling timeouts.

The successful completion of this extended trial on BMW 320i provides definitive, real-world proof of the hardware architecture's stability, demonstrating its full readiness for enterprise-level fleet monitoring applications.

4.6. Comparative Analysis with Existing Telematics Solutions

To evaluate the effectiveness of the proposed ESP32-based telemetry and tamper-detection system, a comparative analysis was conducted against commercially available fleet-monitoring platforms including McEasy and the Geotab. The comparison focuses on telemetry capability, tamper detection, operational reliability, hardware flexibility, and deployment cost.

Feature comparison data was obtained from official product documentation, product descriptions, and publicly accessible service references from McEasy and Geotab.

Table 4.1: Comparative Analysis of the Proposed System Against Commercial Telematics Solutions

Feature / Device	Proposed ESP32 Prototype	McEasy	Geotab
Hardware Cost	Rp 1.4M	Rp 1.4–3.0M	Rp 3.5M+
Monthly Subscription	Rp 0	Rp 100k–150k	Rp 150k–250k
4G LTE Connectivity	✓ Yes	Yes	Yes
Live GPS Tracking	✓ Yes	Yes	Yes
POWER & AUTONOMY			
Primary Power Source	Solar + OBD	Vehicle Battery	Vehicle Battery
Solar Panel Charging	✓ Yes (~8W)	✗ No	✗ No
Independent Battery Pack	✓ Yes (1S3P)	✗ No	✗ No
Battery Autonomy (Offline)	3.4 Days	<8 hours	<2 hours
TAMPER DETECTION			
Case/Enclosure Tamper Alarm	✓ Dedicated Switch	✗ No	⚠ Tilt Detection
Fuel Cap Tamper Alarm	✓ Yes (Micro Limit)	✗ No	✗ No
Cargo Door Monitoring	✓ Magnetic Switches	⚠ External Probe	✗ No
DATA & DIAGNOSTICS			
OBD-II Data Access	✓ Native OBD-II	✗ No	✓ Native OBD-II
Local Storage (Offline Backup)	✓ Yes	✗ No	✗ No
Anomaly Detection	✓ Yes	✗ No	✗ No
KEY ADVANTAGE	Independent Power + Tamper	Local Fleet Presence	Premium Enterprise Ecosystem

**Commercial telematics features and pricing references were obtained from publicly available product documentation and service information from McEasy and Geotab (McEasy, 2024; Geotab, 2024).*

4.6.1. Feature-Based Evaluation

The proposed ESP32-based system delivers significant advantages over commercial platforms like McEasy and Geotab GO9 in three key areas: tamper detection, power independence, and operational reliability (McEasy, 2024; Geotab, 2024).

Power Independence and Extended Autonomy

Unlike conventional telematics systems that depend entirely on vehicle power, the proposed prototype incorporates a solar-assisted charging system and independent 1S3P battery pack. This hybrid approach enables continuous telemetry transmission and tamper monitoring even during vehicle power loss or intentional OBD-II disconnection capability unavailable in competing commercial systems.

Comprehensive Tamper Detection

The system includes dedicated hardware switches for real-time threat detection:

- One enclosure tamper switch
- One fuel-cap limit switch
- Two magnetic cargo-door switches

This multi-point approach detects unauthorized physical access far more effectively than conventional commercial devices, which typically lack such specialized monitoring.

Enhanced Data Resilience

The prototype implements onboard TinyFlash memory logging, which stores telemetry and tamper events during network outages. Data automatically synchronizes to the monitoring dashboard when connectivity is restored. This feature ensures data continuity in unstable network environments - a gap in most commercial platforms.

Performance Verification

Experimental testing confirmed robust system performance: internal event processing occurs within milliseconds, while dashboard alerts transmit in approximately one second under normal LTE conditions. The finalized PCB design also improved electrical stability and reduced hardware complexity compared to breadboard prototypes.

4.6.2. Cost and Operational Analysis

Competitive Hardware Cost with Advanced Features

The finalized prototype costs approximately Rp 1.4 million using off-the-shelf components, yet integrates enterprise-grade capabilities (McEasy, 2024; Geotab, 2024):

- Tamper detection (enclosure, fuel cap, cargo doors)
- Independent battery operation with solar charging
- Offline data logging via TinyFlash
- LTE telemetry and anomaly detection

This positions the system as a feature-rich alternative to subscription-dependent platforms like McEasy and Geotab GO9.

Elimination of Subscription Dependency

Unlike commercial solutions that require ongoing monthly fees (Rp 100k–250k), the proposed system operates indefinitely at zero subscription cost after initial deployment. Over a multi-year fleet lifecycle, this represents substantial operational savings.

Scalability and Cost Reduction Potential

Because the prototype operates on a validated PCB architecture, large-scale manufacturing can significantly reduce production costs. Industry estimates suggest that bulk component procurement, PCB panelization, and assembly optimization could reduce per-unit production costs by 40–60%.

Conclusion

The proposed telemetry and tamper-detection system demonstrates that a low-cost embedded architecture can deliver advanced operational-security and resilience capabilities while maintaining lower deployment costs and greater hardware flexibility than proprietary enterprise platforms.