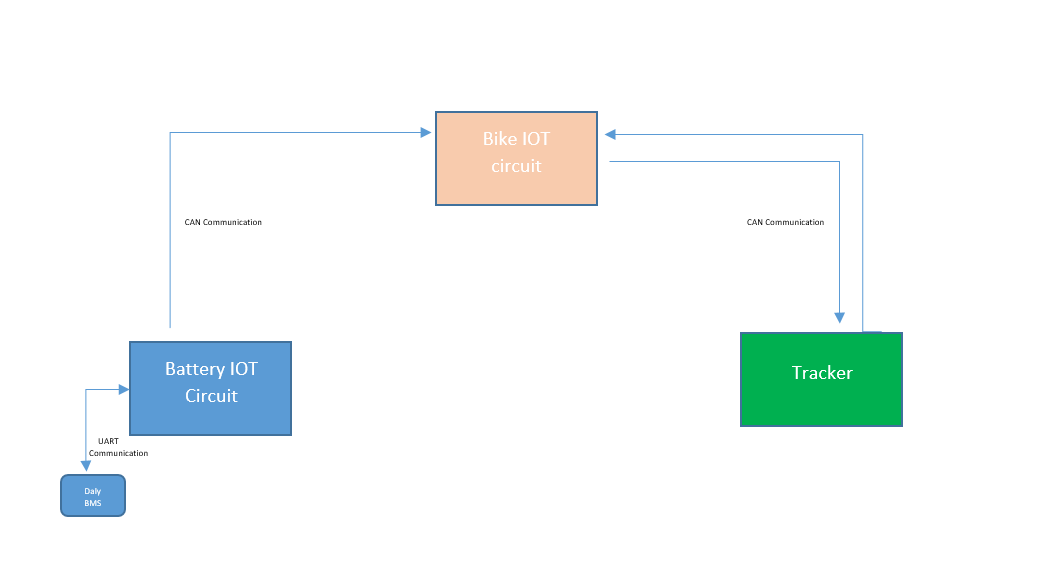
**TFT-100 tracker to IOT integration documentation**

* **Revision Number:**  
  Rev. 1.0
* **Date of Creation or Update:**  
  22/11/2024
* **Prepared By:**  
  M. Sulaiman Shahid (Design Engineer)  
  Muhammad Sabahat Akhtar(Design & firmware Engineer)
* **Reviewed By:**
* **Approved By:**

**Table of Contents**

1. Introduction
2. Working Flow Diagram
3. Battery IOT Working Details
4. Bike IOT Working Detail.
5. TFT-100 tracker working details
6. Project overview
7. Challenges
8. Correction approaches
9. Algorithm flow
10. Key outcomes
11. Troubleshooting Guide for TFT-100 Tracker Integration with IoT Systems

**1. Introduction**

This documentation outlines the integration process of the **TFT-100 Tracker** with IoT circuits, utilizing **CAN Bus** communication for data transmission and **UART communication** for fetching data from the Battery Management System (BMS). The integration aims to enable real-time monitoring and efficient communication between the tracker, the BMS, and the IoT system, ensuring seamless data flow and enhanced system functionality.

**2. Working Flow Diagram**

Figure1: Workflow chart of battery to tracker communication

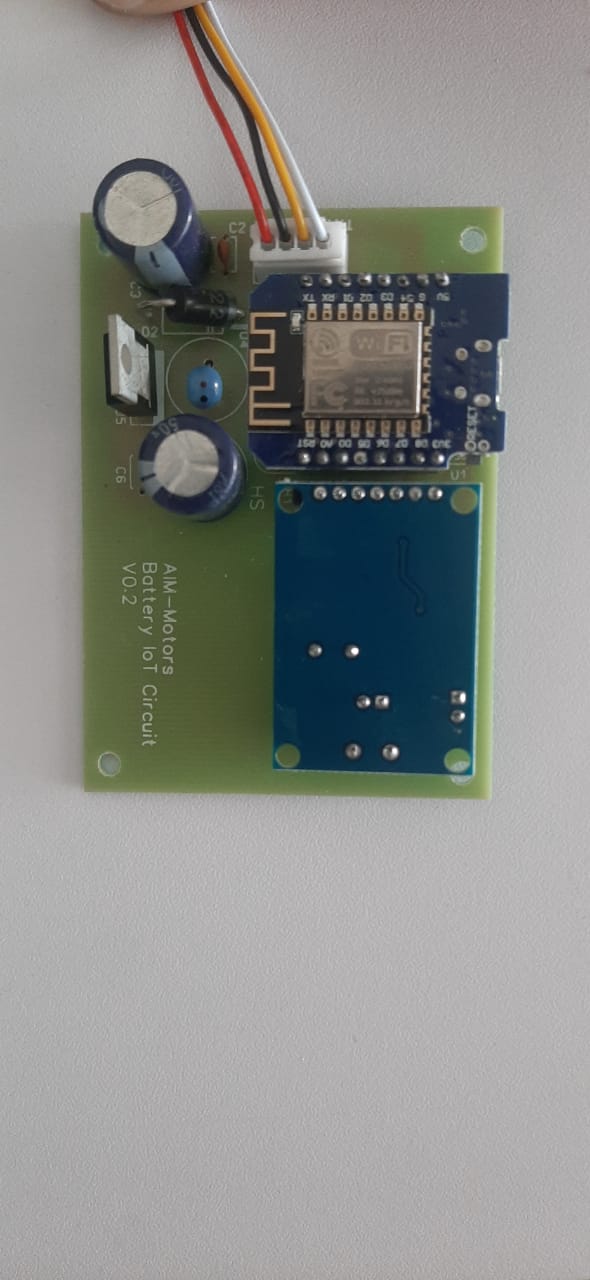
**.Battery IOT Working Details**   
  
**Battery IoT Circuit** that retrieves data from the Battery Management System (BMS) using UART communication, processes the data, and transmits it to the Bike IoT system via CAN communication.

Figure2: Battery IoT Circuit

 **4.Bike IOT Working Details** The **Bike IoT Circuit** is designed to retrieve and send data from both the **Battery IoT** system and the **TFT-100 Tracker** using **CAN communication**. This circuit facilitates seamless data exchange between the two systems, enabling real-time monitoring of battery parameters and tracking information. The data retrieved from the Battery IoT system includes vital metrics such as voltage, current, and state of charge, while the TFT-100 Tracker provides location and tracking data. Using the CAN communication protocol ensures reliable, high-speed data transmission for both systems, allowing for efficient integration and enhanced functionality within the bike’s IoT ecosystem.  
  
  
  
  
Figure3: Bike Iot Circuit

**5. TFT-100 tracker working details**The **TFT-100 Tracker** is a versatile tracking device designed for real-time location monitoring and data logging. It is widely used in applications such as vehicle tracking, fleet management, and IoT-based monitoring systems. Below are the working details of the TFT-100 tracker

Figure 4: TFT-100 Teltonika Tracker

### ****Core Functionality****

The TFT-100 Tracker integrates GPS and GSM/GPRS technologies to perform its tracking and data transmission functions. The **TFT-100 Tracker** also integrates with the Bike IoT system to retrieve parameters of Battery Management System (BMS) from Battery Iot by sending **CAN request IDs** from the tracker. This integration ensures seamless communication and enables the tracker to fetch vital battery data, such as **voltage, current, and state of charge, Individual Cell voltages, Battery Temperature, Battery ID** for efficient monitoring and management within the IoT ecosystem.

### ****Key Features****

* **Real-Time Location Monitoring**: Accurate positioning with GPS/GLONASS support.
* **Data Storage**: Built-in memory to log data when the GSM network is unavailable.
* **Alerts and Notifications**: Configurable alerts for geofencing, overspeed, low battery, Bike Ignition detection.
* **CAN Bus Integration**: Supports communication with other systems like Battery IoT or vehicle IoT circuits via the CAN protocol.

### ****Communication Protocols****

* **GPS Module**: Receives signals from satellites for position tracking.
* **GSM/GPRS Module**: Sends data to the server or IoT platform.
* **CAN Protocol**: Facilitates communication with onboard IoT systems like Battery IoT or Bike IoT and other Iot circuits which supports CAN protocol.

### Wiring Connection of Tracker with Bike Iot

### ****C:\Users\AIM MOTORS\AppData\Local\Microsoft\Windows\INetCache\Content.Word\tracker connector.jpeg 1. Power Input****

* **Wires**:
  + **Red (VCC):** Connect to the vehicle’s positive terminal (10V - 97V).
  + **Black (GND):** Connect to the vehicle’s ground or negative terminal.
* **Notes**:
  + Ensure the input voltage matches the tracker’s specifications (typically 9V–97V DC).

### Figure5: Tracker connector

### ****2. CAN Bus Wiring****

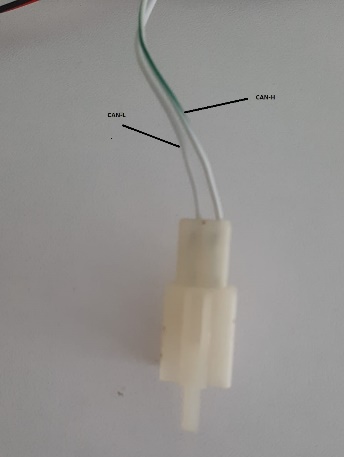
* **Wires**:
* **CAN-H (High):** Connect to the bike Iot CAN bus high line.
* **CAN-L (Low):** Connect to the bike Iot CAN bus low line
* **Functionality**:  
  To enable CAN communication, connect the high-line wire of the bike's IoT system to the tracker’s **white/green** wire (CAN-H). Similarly, connect the low-line wire of the bike's IoT system to the tracker’s **white wire** (CAN-L).
* **Notes**:
  + Verify the tracker’s CAN configuration matches the Bike Iot CAN bus or (baud rate 250k, ID, etc.).
  + The tracker sends and receives CAN messages to communicate with the IoT system and BMS.

Figure 6: CAN connect

### ****3. Ignition Input (ACC Wire)****

* **Wire**:  
   **(IGN):** For ignition detection of bike, connect Green wire (AINT/DINT) of tracker with 12-volt output of key fob or anti-theft system
* **Functionality**:
  + The ignition input detects when the vehicle is turned on or off.
  + It enables the tracker to switch between active and sleep modes based on the ignition status.
* **Notes**:
  + Use a diode or relay to prevent reverse current flow if necessary.
  + Ensure the ignition input voltage does not exceed the tracker’s specified limit (12V-25V)

### ****4. Digital/Analog Inputs****

* **Wires**:
  + **Input 1 (Digital/Analog):** Connect to a sensor or input device (e.g., door status, SOS button).
  + **Input 2 (Digital):** Can be used for additional monitoring (e.g., brake signal or auxiliary device).
* **Notes**:
  + Analog inputs can read voltage levels, while digital inputs detect on/off states.
  + Verify voltage compatibility for connected sensors or devices

### ****5. Digital Outputs****

* **Wires**:
  + **Output 1 (Relay Control):** Can be used to control external devices (e.g., fuel pump, starter relay).
* **Functionality**:
  + Outputs can drive relays or external circuits for vehicle control functions.
  + Typically open-drain outputs that require external power sources to drive loads

**6. Project Overview**

The goal is to integrate bike IoT functionality with the **TFT-100 Tracker** while fetching battery data from a **Daly BMS (R25T-HJ06)**. The project encountered challenges due to power leakage faults and communication inefficiencies. The solution evolved through several correction approaches

**7. Challenges**

1. **UART Port Faults**:  
   External power leakage through the bike Iot to the Daly BMS UART port caused recurring faults.
2. **CAN Limitations**:
   * Daly BMS’s CAN port does not provide individual cell voltages.
   * The existing CAN library in the ESP8266 microcontroller supports only data reception, not transmission to the tracker.
3. **ESP-NOW MAC Dependency**:
   * Initial fixes using ESP-NOW worked, but the MAC address-based pairing is unsuitable for large-scale deployment due to manual configuration requirements.

**8. Correction Approaches**

**1. Using the Daly CAN Interface (Initial Fix)**

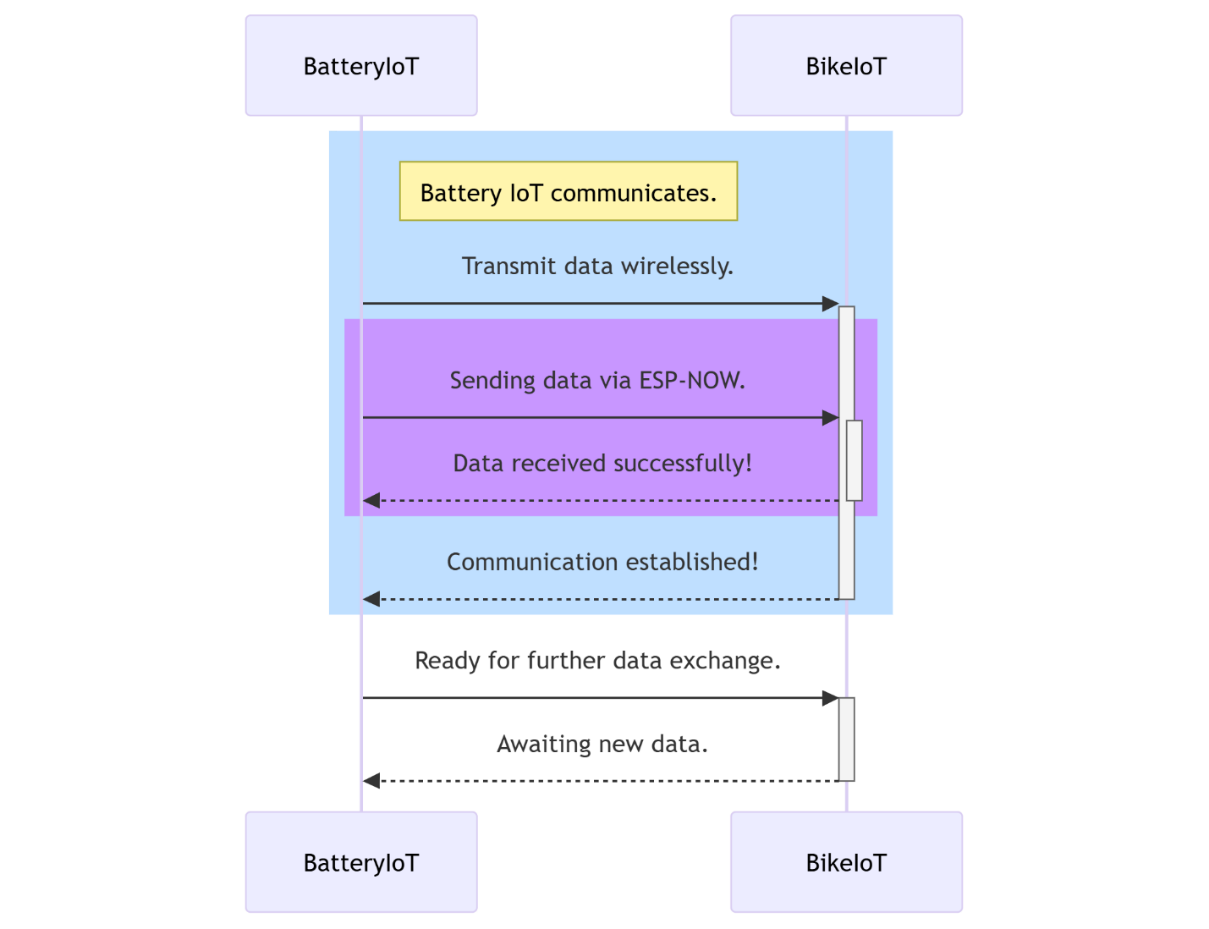
* **Problem**: Individual cell voltages were unavailable from the Daly BMS via CAN.
* **Outcome**: Limited functionality; didn’t meet the project requirements.

**2. Updating the CAN Library**

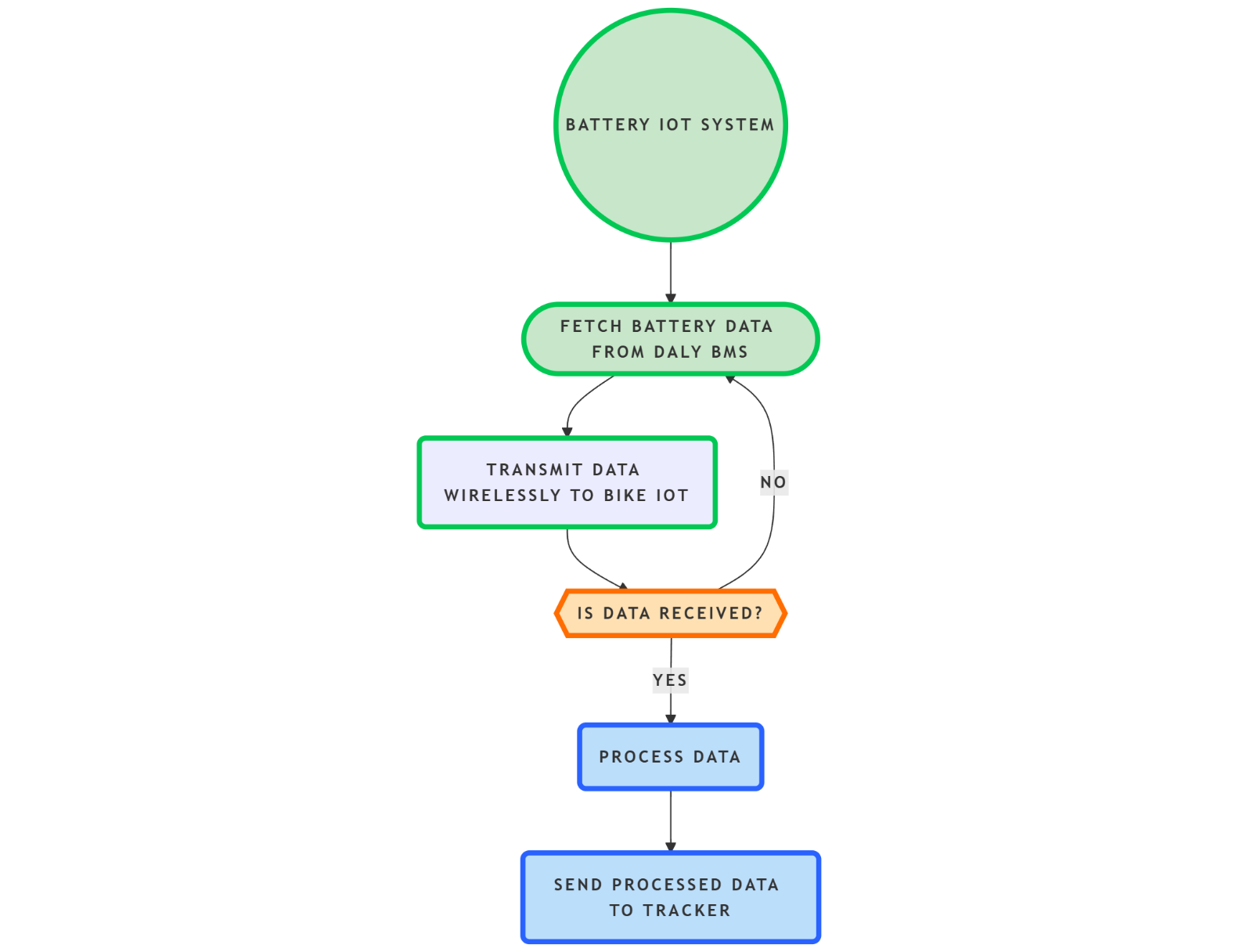
* Introduced a new CAN module library capable of both fetching data and transmitting requests via Request IDs.
* **Problem**: Firmware update and integration required significant development time, missing project deadlines.

**3. Implementing ESP-NOW Communication (Temporary Fix)**

* Utilized **ESP-NOW** to wirelessly communicate data between the **Battery IoT** (slave) and the **Bike IoT** (master).



* Data Flow:
  1. Battery IoT fetched data from Daly BMS via UART.
  2. Transmitted data wirelessly using ESP-NOW.
  3. Bike IoT sent data to the tracker via its CAN module.



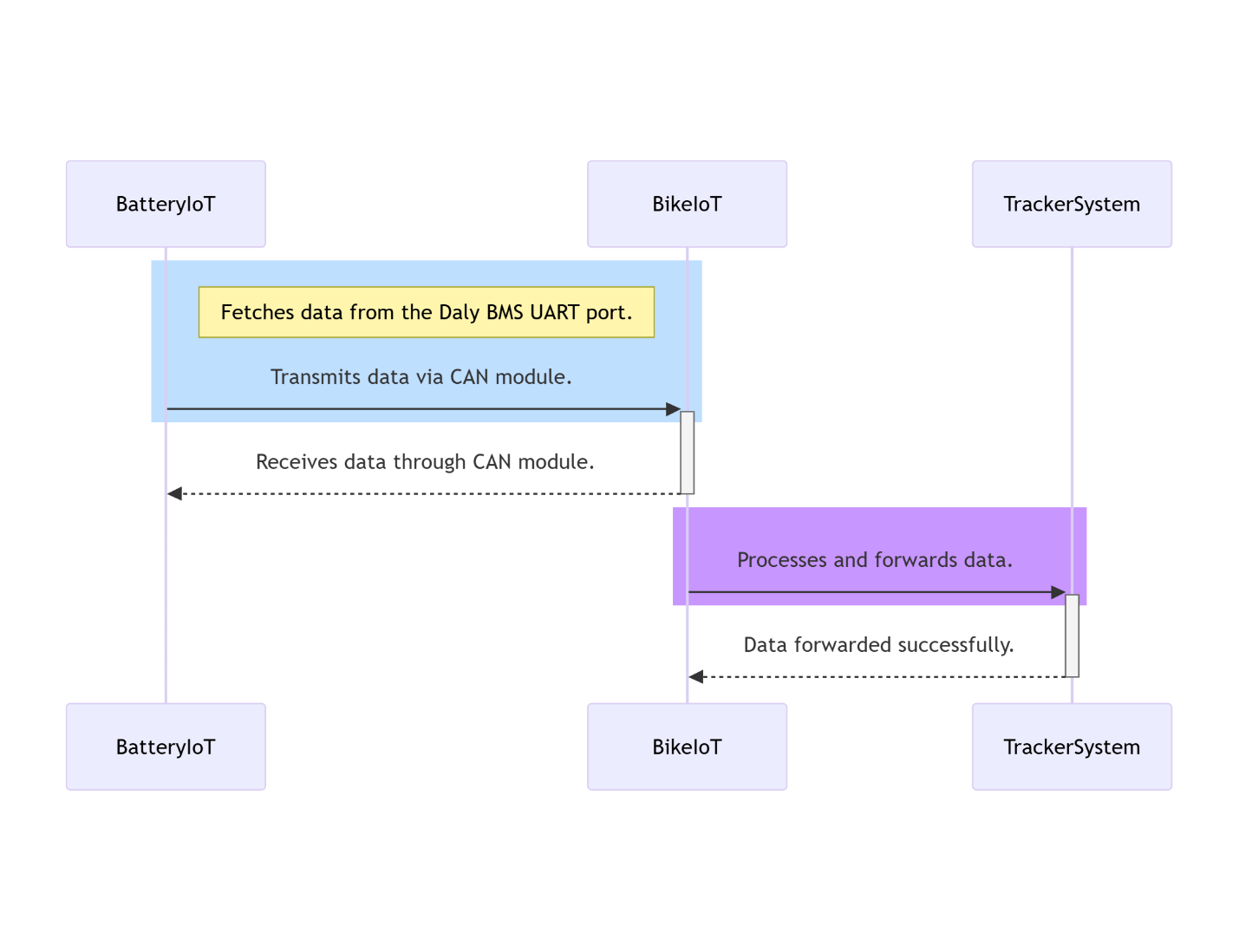
* **Drawback**: ESP-NOW relies on MAC address pairing, leading to scalability challenges in large-scale deployments.

**4. Final Solution: Dynamic Battery Data Retrieval with Updated CAN Library**

* Worked on the new CAN library to enable dynamic data fetching and transmission from the **Battery IoT**, eliminating MAC dependency.
* Designed an algorithm to manage dynamic communication, allowing seamless integration of Battery IoT and Bike IoT irrespective of their MAC addresses.

**9. Algorithm Flow**

1. **Battery IoT**:
   * Fetches data from the Daly BMS UART port.
   * Transmits the data via its CAN module to Bike IoT.
2. **Bike IoT**:
   * Receives data through its CAN module.
   * Processes and forwards data to the tracker system Via CAN.



**10. Key Outcomes**

* The final solution ensures:
  + Scalability for large-scale deployment.
  + Dynamic and seamless communication between IoT devices.
  + Elimination of manual MAC address configuration.
  + This algorithm serves as a robust framework for future projects involving battery data tracking and IoT integration.

### 11. Troubleshooting Guide for TFT-100 Tracker Integration with IoT Systems

This troubleshooting document provides a structured approach to identify, diagnose, and resolve issues in the integration of the TFT-100 Tracker with IoT systems, which includes Battery IoT and Bike IoT circuits.

|  |  |  |  |
| --- | --- | --- | --- |
| Issue | Possible Cause | Affected Component |  |
| UART communication failure | Power leakage through the bike to Daly BMS UART port. | Battery IoT |  |
| Data not received via CAN | Incorrect CAN configuration or mismatched baud rates. | Battery IoT, Bike IoT, Tracker |  |
| Missing battery cell data | Daly BMS CAN protocol does not transmit individual cell voltages. | Daly BMS, Battery IoT |  |
| ESP-NOW connection drops | MAC address dependency or interference in wireless communication. | Battery IoT, Bike IoT |  |
| Tracker not powering on | Incorrect wiring of power input or mismatched voltage specifications. | TFT-100 Tracker |  |
| CAN communication failure | Loose connections or incorrect wiring for CAN-H and CAN-L lines. | Battery IoT, Bike IoT, Tracker |  |
| Ignition detection failure | Incorrect connection of ignition input wire or faulty relay setup. | Tracker |  |
|  |  |  |  |

#### 2. Diagnostic Steps

**Step 1: Verifying Power Supply**

* **Action**: Check the voltage at the tracker’s power input terminals.
  + Ensure the voltage is within the range of 10V to 97V.
  + Inspect for loose connections or damaged wiring.
* **Resolution**: Fix wiring issues or replace damaged components.

**Step 2: Checking UART Communication**

* **Action**: Use a serial monitor to verify data transmission from the Daly BMS to the Battery IoT.
  + Confirm that the UART port on the Daly BMS is functional.
  + Check for power leakage causing port failures.
* **Resolution**: Add isolation circuitry to prevent power leakage.

**Step 3: Verifying CAN Communication**

* **Action**: Inspect CAN-H and CAN-L connections between all devices.
  + Use a CAN analyzer to check message transmission and reception.
  + Verify the baud rate and CAN IDs match across all devices.
* **Resolution**: Reconfigure baud rates and CAN IDs. Ensure proper termination resistors are in place.

**Step 4: Testing ESP-NOW Functionality**

* **Action**: Monitor ESP-NOW communication logs for successful pairing and data transmission.
  + Ensure both devices (Battery IoT and Bike IoT) have correct MAC addresses configured.
  + Check for wireless interference.
* **Resolution**: Update firmware to dynamically fetch MAC addresses or switch to a more scalable communication protocol.

**Step 5: Tracker Diagnostics**

* **Action**: Check tracker LEDs for status indications (power, GPS, GSM, and CAN).
  + Confirm GPS and GSM modules are operational.
  + Verify tracker CAN configurations match the Bike IoT settings.
* **Resolution**: Reconfigure tracker settings using its configuration tool. Replace damaged components if needed.

**Step 6: Ignition Detection**

* **Action**: Measure voltage at the ignition input wire when the bike is turned on.
  + Check diode or relay connections to prevent reverse currents.
* **Resolution**: Correct wiring issues or replace faulty relays.

**3. Wiring Inspection Checklist**

|  |  |  |
| --- | --- | --- |
| Connection | Inspection Point | Status |
| Battery IoT UART to Daly BMS | Confirm secure connection and proper baud rate configuration. |  |
| Battery IoT CAN to Bike IoT | Verify CAN-H and CAN-L lines are correctly connected. |  |
| Bike IoT CAN to Tracker | Check for termination resistors and proper wiring of CAN lines. |  |
| Tracker Power Input | Ensure voltage is within specified range (10V - 97V). |  |
| Ignition Input Wire | Confirm correct voltage detection and proper relay setup. |  |

|  |  |
| --- | --- |
| Error | Resolution |
| No data on UART | Check for power leakage, use isolated UART circuits, and verify Daly BMS settings. |
| Incomplete CAN messages | Reconfigure CAN IDs, ensure baud rates match, and verify termination resistors are in place. |
| ESP-NOW pairing failure | Implement dynamic MAC address retrieval or check for wireless interference. |
| Tracker not logging data | Inspect tracker configuration, ensure sufficient GSM signal, and verify memory functionality. |
| GPS location not updating | Relocate tracker for better satellite visibility and check GPS antenna connection. |
| Ignition not detected | Replace faulty relay or diode; ensure voltage from the ignition wire is within acceptable range. |

#### 4.Error Resolution Techniques

This troubleshooting guide should assist in resolving integration issues effectively, ensuring reliable operation of the IoT ecosystem.