Hardware Block Diagram

1. Purpose of This Document

This document presents a **high-level hardware block diagram** for the CAN-based embedded monitoring system. It shows how major hardware components interact, without delving into pin-level wiring or detailed electrical schematics.

The goal is to provide a **visual understanding** of the system, connecting the Arduino UNO Q, MCP2515 CAN controller, and the CAN bus, while supporting the layered architecture introduced in previous PDFs.

This is the third PDF in Phase 1 – Hardware Architecture.

---

2. Major Hardware Components

The system consists of the following high-level components:

1. **Arduino UNO Q Development Board**
2. MCU (real-time tasks)
3. Linux-capable MPU (non-real-time tasks)
4. **External MCP2515 CAN Controller**
5. Interfaces with MCU via SPI
6. Provides CAN frame buffering and protocol handling
7. **CAN Transceiver**
8. Connects MCP2515 to the physical CAN bus
9. Provides proper voltage levels and termination support
10. **Physical CAN Bus**
11. The network where CAN frames are transmitted and received
12. **External CAN Frame Generator (Software)**
13. Outside the system boundary
14. Sends frames to the CAN bus for testing and monitoring purposes

---

3. High-Level Block Diagram (Textual Representation)

```
        +----------------------+
        | External CAN         |
        | Frame Generator SW   |
        +----------+-----------+
                   |
           CAN Frames on Bus
                   |
        +----------v-----------+
        | Physical CAN Bus      |
        +----------+-----------+
                   |
        +----------v-----------+
        | CAN Transceiver       |
        +----------+-----------+
                   |
        +----------v-----------+
        | MCP2515 CAN Controller|
        +----------+-----------+
                   |
        +----------v-----------+
        | Arduino UNO Q Board   |
        |  +----------------+  |
        |  | MCU (RT tasks) |  |
        |  +----------------+  |
        |  | MPU (Linux)    |  |
        |  +----------------+  |
        +----------------------+
```

Notes: - The MCU handles real-time CAN reception, buffering, and timestamping. - The Linux MPU handles decoding, storage, and monitoring algorithms. - The MCP2515 isolates CAN protocol handling from the MCU, simplifying learning. - The external CAN frame generator is **outside the system boundary**.

---

4. Benefits of This Representation

- Visualizes **data flow from CAN bus to software layers**
- Clarifies **hardware responsibilities** and system boundaries
- Supports understanding of **SPI + MCP2515 trade-offs** discussed in PDF 1.2
- Serves as a teaching aid for beginners without introducing low-level wiring complexity

---

5. Next Steps

After understanding the hardware blocks, the next document will address:

- **PDF 1.4 – Wiring & Power Considerations**
- How to safely connect the MCP2515 and CAN transceiver to the Arduino UNO Q
- Power supply considerations
- Bus termination and signal integrity

This completes the high-level overview of Phase 1 – Hardware Architecture.