Wiring & Power Considerations

1. Purpose of This Document

This document provides **guidance on wiring and power considerations** for the CAN-based embedded monitoring system. It explains how the MCP2515 CAN controller and CAN transceiver connect to the Arduino UNO Q, as well as best practices for power supply, grounding, and CAN bus termination.

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

---

2. Wiring Overview

2.1 MCU to MCP2515

• **Interface:** SPI (Serial Peripheral Interface)
• **Connections:**
• MOSI, MISO, SCK, CS lines between MCU and MCP2515
• Interrupt line from MCP2515 to MCU for frame reception
• **Notes:**
• Use short, direct connections to reduce noise.
• Pull-up resistors may be required on CS and interrupt lines depending on board design.

2.2 MCP2515 to CAN Transceiver

• **Connections:**
• TX and RX signals from MCP2515 to transceiver
• VCC and GND power supply lines
• **Notes:**
• Ensure correct voltage levels for the transceiver (typically 5V or 3.3V depending on the module)

2.3 CAN Bus Connection

• Connect transceiver CAN_H and CAN_L to the physical CAN bus.
• Include termination resistors at each end of the bus (typically 120Ω) to prevent reflections.
• Keep CAN lines twisted pair to reduce EMI.

2.4 Linux Processor Interface

• The MCU transfers buffered frames to the Linux MPU via the chosen IPC mechanism.
• No physical wiring is required beyond the MCU board connections.

---

3. Power Considerations

- **Arduino UNO Q**: Must be powered according to manufacturer specifications (typically 5V or USB power).
- **MCP2515 & Transceiver**:
- Ensure both VCC and GND are correctly connected.
- Power supply should provide stable voltage and sufficient current for both MCP2515 and transceiver.
- **Bus Termination Impact**:
- Proper termination affects signal quality and should be included in power planning.

3.1 Safety Notes

- Double-check polarity of power connections before applying voltage.
- Avoid powering the MCP2515 and transceiver from separate unstable sources.
- Verify voltage levels to prevent damage to MCU and Linux processor.

---

4. Best Practices

- Keep SPI and interrupt lines as short as possible.
- Use decoupling capacitors near power pins of MCP2515 and transceiver.
- Use twisted pair cables for CAN_H and CAN_L.
- Maintain consistent ground reference between MCU and CAN devices.
- Label connections clearly to avoid mistakes during assembly.

---

5. Summary

- The Arduino UNO Q connects via SPI to the MCP2515, which in turn connects to the CAN transceiver and physical CAN bus.
- Proper power supply and grounding are essential for stable operation.
- CAN bus termination and wiring practices ensure signal integrity.
- These wiring and power considerations complete the **hardware architecture phase**, making the system ready for software and IPC design.

---

6. Next Steps

With Phase 1 complete, the project moves to **Phase 2 – Software Architecture**, which will cover: - Firmware responsibilities on the MCU - Linux-based application design - Data flow between hardware and software - Preparing for IPC design and message format definition