

Design, Testing, and Integration of a Modular Multi-Sensor System Using a Deterministic Serial Data-Frame Formation Protocol for Industrial IoT Edge Nodes

Prof. Rohini Mogarkar¹, Sneha Damke², Shraddha Zade³, Simran Gajbhiye⁴, Snehal Bagde⁵
^{1,2,3,4,5}*Industrial IoT, Priyadarshini College of Engineering, Nagpur, India*

Abstract—Industrial Internet of Things (IIoT) edge nodes frequently require the use of different sensor configurations and control algorithms. Traditionally, in an embedded design environment, changing these configurations would require modifying the firmware and re-programming the device. This paper presents the design of a modular multi-sensor system based on the Microchip PIC18F47Q10 8-bit microcontroller. The architecture of the proposed system consists of a panelized Modular Sensor Kit, a multi-port Controller Board, and a Desktop Software Application. A human-readable serial data-frame formation protocol is defined for transmitting configuration data, commands, acknowledgments, and telemetry. The embedded design employs an input–logic–output rule model for evaluating sensor conditions and controlling connected actuators. Temporal qualification, hysteresis, output priority arbitration, and a global STOP command are included in the planned firmware behavior. A bounded receive-frame length of 128 bytes is specified for memory-aware parsing. Additionally, the paper defines a reproducible test plan for frame validation, rule execution, conflict resolution, telemetry, and safe-stop behavior. Some features and performance metrics are subject to bench validation; thus, the paper distinguishes design specifications from experimentally verified results. The proposed framework aims to simplify the configuration, monitoring, and extension of small embedded edge nodes for selected automation applications.

Index Terms—Industrial Internet of Things (IIoT), PIC18F47Q10, modular sensors, serial data-frame formation protocol, embedded rule engine, edge computing, telemetry, safe-stop control.

I. INTRODUCTION

The Industrial Internet of Things connects sensors, embedded controllers, software tools, and actuators so that equipment and environmental conditions can be monitored and controlled. Applications include

greenhouse monitoring, cold-chain supervision, factory-floor gas detection, and equipment-condition monitoring. These applications do not always use the same sensor combination or the same control conditions.

In many conventional embedded systems, sensor assignments, threshold values, sampling routines, and interlocks are defined directly in firmware. When a configuration changes, a developer may need to edit, compile, and flash the firmware again. Repeating this process can slow down commissioning and maintenance, particularly when a system is used for multiple demonstrations or operating scenarios.

A modular architecture can separate the physical sensor interface from the rules that determine how the system responds. In such a design, the controller runs a general rule engine while a host application sends validated configuration data over a serial connection. This approach does not remove the need for firmware updates when hardware drivers or protocol behavior change, but it can reduce the need to recompile firmware for routine changes to supported rules and parameters.

This work describes the design and planned validation of a modular multi-sensor system built around the PIC18F47Q10 microcontroller. The system includes a Modular Sensor Kit, a central Controller Board, and a Desktop Software Application. Its primary focus is a bounded, human-readable serial data-frame formation protocol and an embedded input–logic–output processing model.

II. LITERATURE REVIEW

IoT research has investigated architectures that connect devices, communication networks, and cloud services. Cloud platforms are beneficial for large-scale

storage and analysis; however, a control loop that relies solely on remote processing can be negatively impacted by network latency or loss of connectivity. Edge computing moves selected processing closer to the physical equipment and can enable local decisions when the network is unavailable. Industrial automation typically utilizes Programmable Logic Controllers (PLCs) and engineering tools that allow control logic to be configured. PLCs are well-established; however, their hardware, software, and licensing requirements may not be appropriate for all low-cost or educational applications. Microcontrollers provide an alternative for smaller systems, provided that their electrical, timing, memory, and safety limitations are taken into account.

Lightweight interpreters and bytecode engines can provide runtime flexibility; however, they also require careful management of program representation and execution resources. Text-based protocols offer a different compromise: they are easier for a developer to view in a serial terminal, but parsing must be designed to reject malformed and excessively large frames. The proposed design employs explicit delimiters and key-value fields to make messages readable while limiting the maximum frame size.

III. RESEARCH GAP

Sample paragraph, the entire document should be in cambria font. Type 3 fonts must not be used. Other font types may be used if needed for special purposes. The entire document should be in cambria font. Type

3 fonts must not be used. Other font types may be used if needed for special purposes.

- Hardware modularization: connect multiple transducers to a centralized PIC18F47Q10 Controller Board through modular breakouts and defined ports.
- Protocol formulation: define a serial ASCII frame grammar with explicit boundaries and a maximum frame length of 128 bytes.
- Rule processing: implement or specify an input–logic–output rule pipeline with timing qualification, hysteresis, and output arbitration.
- Host integration: provide a Desktop Software Application for rule configuration, serial command dispatch, acknowledgments, telemetry, and logging.
- Validation: define repeatable tests for parser behavior, rule response, priority handling, and safe-stop operation.

IV. HARDWARE DESING AND COMPONENTS

PIC18F47Q10 Microcontroller- The PIC18F47Q10 is the proposed central controller. The original project specification identifies an 8-bit PIC architecture, an internal oscillator up to 64 MHz, on-chip program and data memory, ADC capability, EUSART modules, MSSP serial interfaces, timers, and PWM-related peripherals. The exact specifications should be verified against the official datasheet and the selected package prior to final publication.

Table

Parameter	Specification stated in project notes	Relevance
CPU	8-bit PIC architecture	Compact local processing
Clock	Up to 64 MHz internal oscillator	Parser and rule scheduling
Supply range	1.8 V to 5.5 V (verify by device datasheet and conditions)	Power and interface planning
Program memory	128 KB stated in source draft; verify exact device notation	Firmware storage
Data SRAM	3,615 bytes stated in source draft; verify against official datasheet	Bounded buffers and rule table
EEPROM	1,024 bytes stated in source draft; verify device-specific details	Potential persistent configuration
I/O	Up to 35 user I/O in listed packages (verify package pinout)	Sensor/actuator interfaces
ADC	10-bit ADC with computation; channel count to be verified	Analog sensor acquisition
Serial	2 EUSART modules stated	Host link and auxiliary serial devices
Other peripherals	MSSP, timers, PWM-related modules (verify exact count)	Display, timing, and actuation

V. IMPLEMENTATION AND EXPERIMENTAL METHODOLOGY

5.1 Frame Parsing and Rule Compilation

The proposed parser receives serial bytes into a bounded buffer, identifies a complete frame, validates the envelope, and checks the field syntax. The configuration layer then checks sensor aliases, rule identifiers, operators, and parameter ranges. Only a valid configuration should be loaded into the active rule table.

The design notes describe converting logical expressions into an ordered token representation, such as reverse Polish notation (postfix form), and scaling selected decimal readings to signed integers. These are implementation options that should be verified against the actual source code. The periodic rule evaluation interval is specified as 10 ms in the project design; measured timing should be reported separately.

5.2 Output Conflict Arbitration

The proposed design specifies the following five-tier conflict policy:

- Tier 1 — Global authority: STOP or a defined FAULT condition overrides ordinary rule requests and applies the safe output policy.
- Tier 2 — Explicit safety override: rules designated with the highest safety classification, referred to as P9 in the source specification, override general routines.
- Tier 3 — Priority rank: a higher numerical priority from P1 to P9 prevails when rules request conflicting states.
- Tier 4 — Fail-safe tie-break: for equal-priority conflicting ON/OFF requests, OFF takes precedence.
- Tier 5 — Temporal recency: if priority and action type are equal, the most recent edge transition is intended to prevail.

This hierarchy is a proposed software arbitration policy. It should be implemented as an explicit and testable function. The exact definition of P9 versus a separate safety override must be clarified so that the priority rules do not conflict with one another.

5.3 Measurement Procedure

For each test, record the hardware revision, firmware version, host application version, sensor configuration, serial baud rate, supply conditions, test

duration, and number of repeated trials. Record transmitted frames, received frames, error codes, output states, and timestamps. Response time should be measured from a defined event (for example, the last byte of a command received) to a defined outcome (for example, all required outputs achieving the target state).

At 115,200 baud, serial traffic capacity depends on the framing format and protocol overhead. Therefore, continuous LIVE and DASH messages should be rate-limited or scheduled so that telemetry does not impede the processing of control commands.

VI. CONCLUSIONS

This paper has presented the design of a modular multi-sensor IIoT edge node based on the PIC18F47Q10 microcontroller. The architecture integrates a Modular Sensor Kit, a central Controller Board, a Desktop Software Application, and a structured serial data-frame formation protocol. The intended rule engine employs an input–logic–output model to enable configurable sensor conditions and actuator actions without requiring firmware recompilation for each supported rule change.

The proposed design also addresses bounded frame reception, error handling, telemetry, temporal qualification, output conflict resolution, and a global STOP policy. Its practical utility will depend on consistent protocol implementation, careful memory management, proper electrical integration, and repeatable bench testing. Additional experimental measurements are required to determine response time, communication reliability, maximum rule capacity, and suitability for specific deployment scenarios.

VII. PROBLEM STATEMENT AND OBJECTIVES

7.1 Problem Statement

Multi-sensor embedded systems can be difficult to maintain when sensor combinations, thresholds, delays, and actuator actions are tightly coupled to the firmware. Modifying these settings may require multiple programming cycles and increases the likelihood of configuration errors. Therefore, a practical system requires a defined method for transmitting, validating, storing, and executing supported configuration rules while rejecting invalid

input and predictably resolving conflicting output requests.

7.2 Research Objectives

- Hardware modularization: connect multiple transducers to a centralized PIC18F47Q10 Controller Board through modular breakouts and defined ports.
- Protocol formulation: define a serial ASCII frame grammar with explicit boundaries and a maximum frame length of 128 bytes.
- Rule processing: implement or specify an input–logic–output rule pipeline with timing qualification, hysteresis, and output arbitration.
- Host integration: provide a Desktop Software Application for rule configuration, serial command dispatch, acknowledgments, telemetry, and logging.

ACKNOWLEDGEMENT

The authors acknowledge the guidance and laboratory support provided by the Department of Industrial IoT Engineering and the industry project team during the system design, hardware integration, and protocol development activities. The final version should name the actual individuals or organizations only after their permission and contribution have been confirmed.

REFERENCES

- [1] Microchip Technology Inc., “PIC18F27/47Q10 28/40/44-Pin, Low-Power, High-Performance Microcontrollers Data Sheet,” document DS40002043E, Rev. E, November 2020. Official datasheet (PDF).
- [2] Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M., “Internet of Things (IoT): A vision, architectural elements, and future directions,” *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013, doi: 10.1016/j.future.2013.01.010.
- [3] Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., and Ayyash, M., “Internet of Things: A survey on enabling technologies, protocols, and applications,” *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015, doi: 10.1109/COMST.2015.2444095.
- [4] Sisinni, E., Saifullah, A., Han, S., Jennehag, U., and Gidlund, M., “Industrial Internet of Things: Challenges, opportunities, and directions,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, pp. 4724–4734, 2018, doi: 10.1109/TII.2018.2852491.
- [5] Xu, L. D., He, W., and Li, S., “Internet of Things in industries: A survey,” *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233–2243, 2014, doi: 10.1109/TII.2014.2300753.
- [6] Project Technical Working Group, Master Technical Specification for Multi-Sensor Configuration and Serial Rule Engine Protocol, technical report MTS-V1.0-2026, Embedded Systems Integration Lab, 2026. (Include only if this is an actual internal document available to the authors.)