SEH500 Course Assignment Report: The Pain Level Communicator (PLC)

Group Members: Dhruv Chotalia, Navdisha Bhakri
Platform: NXP FRDM-K66F (ARM Cortex-M4)

Date: November 30, 2025

---

1. Problem Statement

Pain management is a critical component of patient care, yet it relies heavily on the patient's ability to communicate verbally. For patients with limited mobility, neurological impairments (such as stroke or ALS), or those who are intubated in critical care, expressing pain levels is physically difficult and slow. Traditional methods, such as the "**Wong-Baker FACES**" paper chart, require a nurse to be physically present to interpret the patient's distress. (Morales-Brown, 2022) This latency in communication can lead to delayed analgesic administration, increased patient suffering, and poorer recovery outcomes.

The problem this project addresses is the lack of an **immediate, digital, and remote** interface for non-verbal patients to quantify their pain and receive instant confirmation that help is on the way.

---

2. Background Research

The *Federal Disability Report* states that over 13% of Canadian adults face daily activity limitations due to disability. (Government of Canada, 2011) In clinical settings, assistive technology is shifting from mechanical aids to microcontroller-based "smart" systems.

Existing solutions fall into two categories: simple nurse-call buttons (binary ON/OFF) which lack severity detail, or complex bedside tablets requiring fine motor control often unavailable to the target demographic. Our research identifies a gap for a device combining the simplicity of a single-button interface with the data granularity of a digital system. By utilizing the Cortex-M4 processor, we can bridge this gap with real-time processing and low-latency hardware control [2]. (Yiu, 2013)

---

3. Proposed Solution

The **Pain Level Communicator (PLC)** is an embedded system designed on the NXP FRDM-K66F platform. It serves as a bridge between the patient and the nursing station.

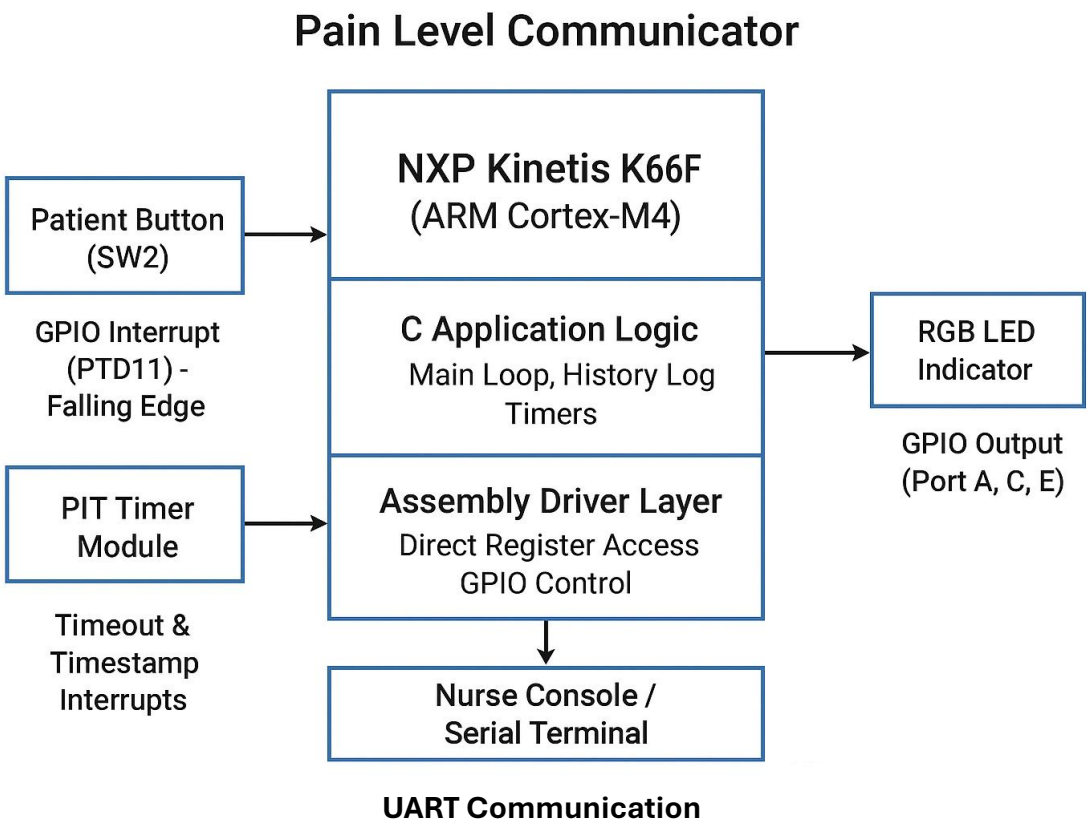
The solution operates on a "Single-Input, Multi-Output" architecture:

1. **Input:** A single button (SW2) allows the patient to increment their pain level from 1 (Mild) to 5 (Severe).
2. **Feedback Loop:** An onboard RGB LED provides immediate visual confirmation to the patient using a color-coded spectrum (Green to Red).
3. **Communication:** The system transmits the data via UART to a Nurse Console, logging the event.
4. **Two-Way Interaction:** Crucially, the system allows the nurse to send feedback commands ('A' for Acknowledge, 'M' for Meds, 'D' for Doctor) which triggers specific lighting states on the patient's board, closing the communication loop.

The system uses **mixed-language programming**, utilizing C for high-level logic and data structures, and **GNU Assembly** for direct, low-level hardware driver control.

---

4. System Block Diagram & Module Explanation



4.1 Hardware Abstraction Layer (Assembly Module)

To meet the rigorous timing and control requirements of real-time embedded systems, the LED control logic was written entirely in GNU Assembly **(led_logic.s).** This module acts as a custom driver, bypassing the standard SDK libraries to interact directly with the microcontroller's Memory-Mapped I/O registers.

The assembly module operates through a precise "Read-Modify-Write" sequence to ensure atomic control over the hardware:

1. **Direct Register Addressing:** Utilizes .equ directives for absolute physical addresses like SIM_SCGC5 (0x40048038) and PDOR.
2. **Bitwise Logic & Active-Low Control:** The FRDM-K66F LEDs are wired in an active-low configuration (Common Anode), meaning a logic '0' activates the light. To handle this, our assembly logic uses the BIC (Bit Clear) instruction to turn specific color channels ON and the ORR (Logical OR) instruction to turn them OFF. This ensures we manipulate only the specific bits for Pins A11, C9, and E6 without disturbing other peripherals on the same port.
3. **Nurse Acknowledgment Routine:** A specialized subroutine, asm_nurse_ack, was developed to perform a multi-port write operation. It simultaneously clears the bits for Red, Green, and Blue across three different GPIO ports, driving the RGB LED to a steady White state to signal medical acknowledgment.
4. **AAPCS Compliance:** Stack pointers are managed via PUSH {LR} and POP {PC} to ensure seamless mixed-language linking.

4.2 Application Logic (C Module)

The Project.c file acts as the system supervisor. It utilizes:

- **Interrupt Service Routines (ISRs):** A GPIO interrupt handles the button presses to ensure no inputs are missed, while a Periodic Interrupt Timer (PIT) handles the "input timeout," determining when the patient has finished entering their level.
- **Data Structures:** A circular buffer (patient_history array) logs the last 10 events.

5. Concept Validation & Improvements

Our solution validates the concept of **"Responsive Assistive Tech."** Unlike a standard call bell, the PLC validates the patient's input twice: once locally (via color change) and once remotely (via the Nurse Acknowledgment light).

Feature	Standard Call Bell	Bedside Tablet	PLC (Our Solution)
Input Method	Single Button	Touchscreen	Single Button (Debounced)
Information	Binary (On/Off)	High Detail	Quantitative (1–5 Scale)

Feature	Standard Call Bell	Bedside Tablet	PLC (Our Solution)
Latency	Low	High (Boot/UI Load)	Ultra-Low (Interrupt Driven)
Feedback	None	Screen UI	RGB LED Color Code

By using direct register manipulation in Assembly, we demonstrated that the GPIO control has negligible latency compared to HAL-based libraries, ensuring the patient sees the LED change the instant the logic confirms the level.

6. Development Process & Obstacles

Developing the PLC presented several technical challenges that required debugging and architectural changes.

Obstacle 1: The Bus Fault (Precise Data Error)

Issue: Early in development, the application crashed immediately upon startup with a PRECISERR Hard Fault.
Root Cause: We attempted to write to the GPIO Output Registers in our Assembly driver before the generic C initialization code had enabled the clocks for Ports A and E. The memory-mapped addresses were "dark," causing the CPU to panic. (Yiu, 2013).
Solution: We implemented a "Manual Clock Force" routine in main(), explicitly calling CLOCK_EnableClock(kCLOCK_PortA/C/D/E) before invoking any Assembly setup routines. This ensured the hardware peripheral bus was active.

```
int main(void) {
    BOARD_InitBootPins();
    BOARD_InitBootClocks();
    BOARD_InitBootPeripherals();
    BOARD_InitDebugConsole();

    CLOCK_EnableClock(kCLOCK_PortA);
    CLOCK_EnableClock(kCLOCK_PortC);
    CLOCK_EnableClock(kCLOCK_PortD);
    CLOCK_EnableClock(kCLOCK_PortE);
```

Obstacle 2: Input Accumulation Logic

Issue: Pressing the button for "Level 1" followed by "Level 2" resulted in a reading of "Level 3." The system was accumulating presses infinitely.
Solution: We introduced a session-based logic. A global variable press_count tracks the current input sequence. Once the PIT timer fires (indicating the user has stopped pressing for 1.5 seconds), the value is committed to the stored_pain_level variable, and press_count is strictly reset to 0.

Obstacle 3: 2-Way Communication Feedback

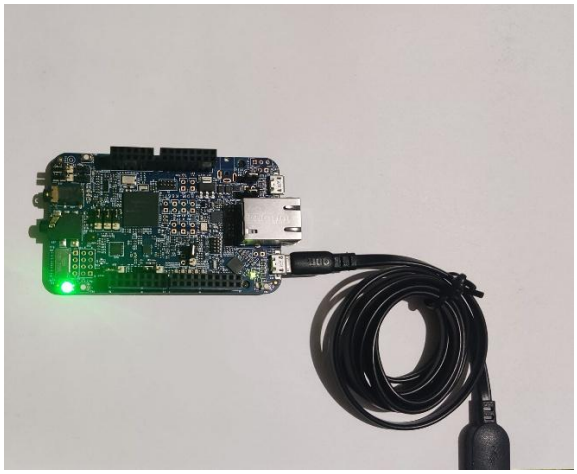
Issue: Initially, the nurse acknowledgment simply blinked a blue light. Testing revealed this might be confusing or alarming to a patient.
Solution: We upgraded the Assembly logic to support a "Medical Protocol." We mapped UART characters to distinct states: 'A' (Ack) triggers a steady White light (calming), while 'M' triggers Cyan. This required rewriting the asm_set_pain_led branching logic to handle command codes > 5.

7. Project Configuration & Reproducibility To ensure the successful replication of this project on the NXP FRDM-K66F platform, the following hardware and software configurations must be verified within the MCUXpresso IDE environment:

1. **Driver Dependencies:** Include GPIO, UART, PIT, and PORT drivers during creation.
2. **Clock Gating:** We implemented a "Manual Clock Force" routine in main() to explicitly enable clocks for Ports A, C, D, and E. This prevents PRECISERR Bus Faults when Assembly drivers access hardware registers.
3. **Pin Routing:**
    - **Input:** The User Button (SW2) must be routed to **PTD11** (GPIOD, Pin 11).
    - **Interrupts:** The internal Pull-Up Resistor and "Falling Edge" interrupt logic are configured programmatically in Project.c, removing the need for complex GUI configuration tools.
    - **Communication:** The Debug UART must be configured to **115200 Baud, 8N1** to ensure synchronization with the serial terminal.

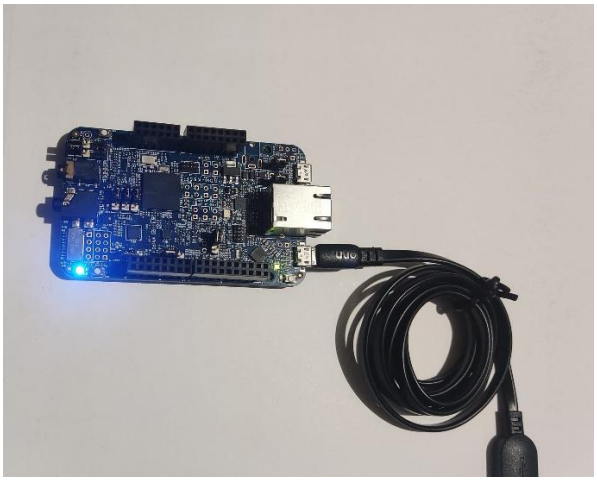
8. Demonstration of Functionality

```
COM3 ×

========================================
   Pain Level Communicator (PLC) v1.0
========================================
COMMANDS:
 [A] - Acknowledge (White Light)
 [M] - Meds Sent (Cyan Light)
 [D] - Doctor Summoned (White Light)
 [S] - Generate Analysis Sheet
 [R] - Reset System & Clear History
========================================
Button Pressed! Current Session Count: 1
Input Complete. Final Pain Level: 1
PAIN_LEVEL:1
```



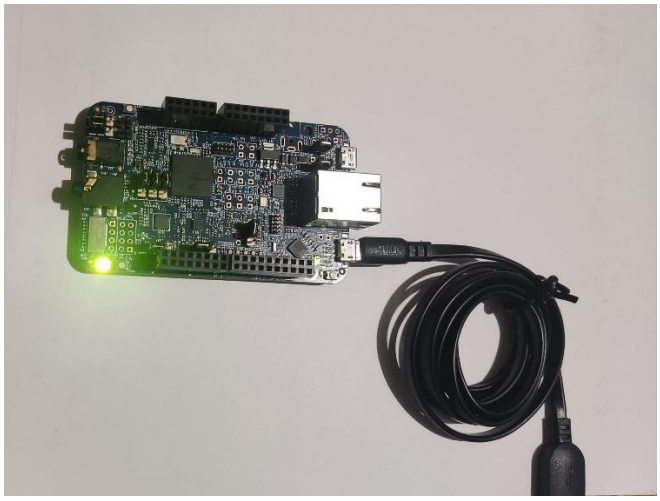
Caption: Figure 1: The FRDM-K66F board indicating "Level 1" via the Assembly-driven RGB LED.

```
COM3 ×

========================================
   Pain Level Communicator (PLC) v1.0
========================================
COMMANDS:
 [A] - Acknowledge (White Light)
 [M] - Meds Sent (Cyan Light)
 [D] - Doctor Summoned (White Light)
 [S] - Generate Analysis Sheet
 [R] - Reset System & Clear History
========================================
Button Pressed! Current Session Count: 1
Input Complete. Final Pain Level: 1
PAIN_LEVEL:1
Button Pressed! Current Session Count: 1
Button Pressed! Current Session Count: 2
Input Complete. Final Pain Level: 2
PAIN_LEVEL:2
```



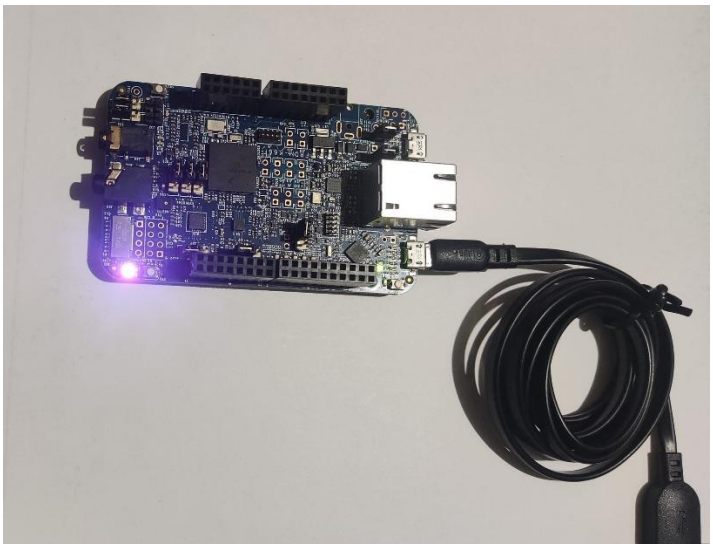
Caption: Figure 2: The FRDM-K66F board indicating "Level 2" via the Assembly-driven RGB LED.

```
COM3 ×

========================================
   Pain Level Communicator (PLC) v1.0
========================================
COMMANDS:
 [A] - Acknowledge (White Light)
 [M] - Meds Sent (Cyan Light)
 [D] - Doctor Summoned (White Light)
 [S] - Generate Analysis Sheet
 [R] - Reset System & Clear History
========================================
Button Pressed! Current Session Count: 1
Input Complete. Final Pain Level: 1
PAIN_LEVEL:1
Button Pressed! Current Session Count: 1
Button Pressed! Current Session Count: 2
Input Complete. Final Pain Level: 2
PAIN_LEVEL:2
Button Pressed! Current Session Count: 1
Button Pressed! Current Session Count: 2
Button Pressed! Current Session Count: 3
Input Complete. Final Pain Level: 3
PAIN_LEVEL:3
```



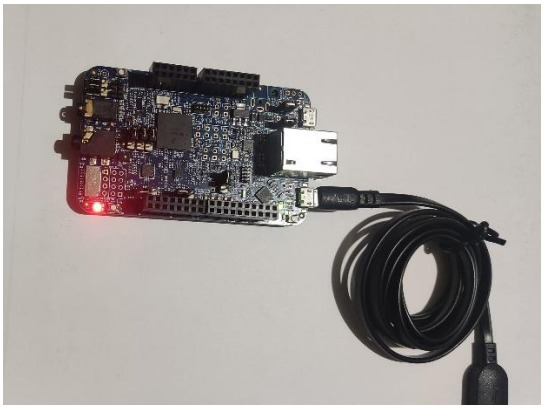
Caption: Figure 3: The FRDM-K66F board indicating "Level 3" via the Assembly-driven RGB LED.

```
COM3 ×

========================================
   Pain Level Communicator (PLC) v1.0
========================================
COMMANDS:
 [A] - Acknowledge (White Light)
 [M] - Meds Sent (Cyan Light)
 [D] - Doctor Summoned (White Light)
 [S] - Generate Analysis Sheet
 [R] - Reset System & Clear History
========================================
Button Pressed! Current Session Count: 1
Input Complete. Final Pain Level: 1
PAIN_LEVEL:1
Button Pressed! Current Session Count: 1
Button Pressed! Current Session Count: 2
Input Complete. Final Pain Level: 2
PAIN_LEVEL:2
Button Pressed! Current Session Count: 1
Button Pressed! Current Session Count: 2
Button Pressed! Current Session Count: 3
Input Complete. Final Pain Level: 3
PAIN_LEVEL:3
Button Pressed! Current Session Count: 1
Button Pressed! Current Session Count: 2
Button Pressed! Current Session Count: 3
Button Pressed! Current Session Count: 4
Input Complete. Final Pain Level: 4
PAIN_LEVEL:4
```



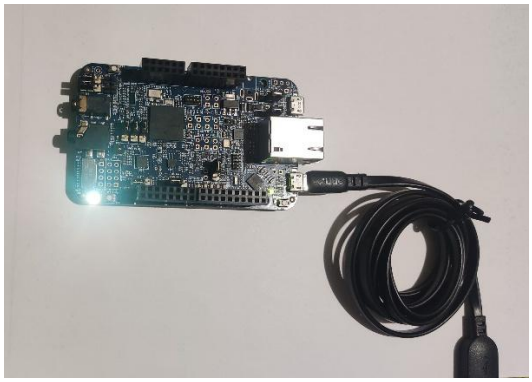
Caption: Figure 4: The FRDM-K66F board indicating "Level 4" via the Assembly-driven RGB LED.

```
PAIN_LEVEL:4
Button Pressed! Current Session Count: 1
Button Pressed! Current Session Count: 2
Button Pressed! Current Session Count: 3
Button Pressed! Current Session Count: 4
Button Pressed! Current Session Count: 5
Input Complete. Final Pain Level: 5
PAIN_LEVEL:5
```



Caption: Figure 5: The FRDM-K66F board indicating "Level 5" via the Assembly-driven RGB LED.

```
PAIN_LEVEL:4
Button Pressed! Current Session Count: 1
Button Pressed! Current Session Count: 2
Button Pressed! Current Session Count: 3
Button Pressed! Current Session Count: 4
Button Pressed! Current Session Count: 5
Input Complete. Final Pain Level: 5
PAIN_LEVEL:5
[Nurse]: DOCTOR SUMMONED.
```



Caption: Figure 6: The "Doctor" state. The White light indicates the doctor has received the message.

```
--- PATIENT HISTORY ANALYSIS ---
Event #1: Pain Level 2
Event #2: Pain Level 1
Event #3: Pain Level 3
Event #4: Pain Level 3
Event #5: Pain Level 4
Event #6: Pain Level 5
------------------------------
Total Events Recorded: 6
------------------------------
```

Caption: Figure 7: Serial Terminal showing the Patient History Log with Timestamps and Nurse Interaction.

8. Future Work

While the current prototype is fully functional, future iterations could enhance usability:

1. **IoT Integration:** Replacing the UART cable with an ESP8266 Wi-Fi module to transmit pain levels to a web dashboard.
2. **PWM Brightness:** Using Pulse Width Modulation (PWM) in Assembly for smoother color transitions (fading) rather than binary switching.
3. **Non-Volatile Storage:** Writing the patient_history array to the board's Flash memory so data persists even after power loss.

9. Conclusion

The Pain Level Communicator successfully demonstrates the power of the ARM Cortex-M4 in medical applications. By integrating 100+ lines of custom GNU Assembly with high-level C logic, we created a device that is fast, reliable, and user-centric. The project meets all requirements, including two-way serial communication, interrupt-driven inputs, and hardware timer logging, providing a measurable improvement over traditional non-verbal pain assessment tools.

10. References

[1] Government of Canada, "Federal Disability Report," *Publications.gc.ca*, 2011. [Online]. Available: https://publications.gc.ca/site/eng/9.505762/publication.html. [2]

J. Yiu, *The Definitive Guide to ARM® Cortex®-M3 and Cortex®-M4 Processors*, 3rd ed. Oxford: Newnes, 2013. [3] NXP Semiconductors, "FRDM-K66F User's Guide," Rev. 2, 2016.

Morales-Brown, L. (2022, December 23). *What to know about the Wong-Baker pain scale*. **Medical News Today**. Medically reviewed by Lauren Castiello, MS, AGNP-C. https://www.medicalnewstoday.com/articles/wong-baker-pain-scale