Seneca Polytechnic

School of Software Design & Data Science

SEH500 - Course Assignment - Fall 2025

Project Report: One-Hand Typing Trainer for Post-Stroke Rehab

Group 11: Krish Patel, Mit Patel

Problem Statement

Individuals recovering from stroke or experiencing unilateral motor impairments face significant challenges with rhythmic timing and consistent press/hold control, fundamental skills for typing and daily device interaction. Traditional metronome-based rehabilitation lacks immediate, quantitative feedback on timing accuracy and hold stability, making it difficult for patients and therapists to track progress objectively or adapt difficulty levels appropriately. There is a critical need for a simple, low-cost tool that provides rhythmic guidance, measures performance in real-time, and adapts to the user's evolving abilities.

Background Research

Existing Solutions

Current rehabilitation approaches for motor timing include:

1. **Traditional Metronomes**: Provide auditory beats but lack feedback mechanisms or performance tracking .
2. **Commercial Rehabilitation Software**: Applications like MusicGlove and Flint Rehab offer gamified exercises but require expensive tablets or specialized hardware ($200-$500) [1].
3. **Physical/Occupational Therapy**: Manual exercises supervised by therapists provide personalized feedback but are resource-intensive and lack quantitative metrics between sessions.
4. **Research Prototypes**: Academic studies have explored rhythm-based motor training using computers and sensors, but these remain in laboratory settings and are not accessible to patients at home.

Proposed Solution

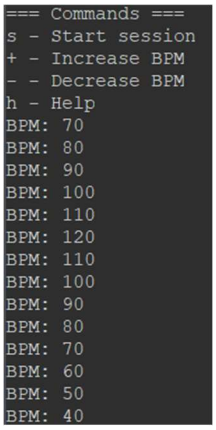
The proposed solution is an embedded rehabilitation device designed to improve motor timing and coordination for post-stroke patients. Built on the **NXP FRDM-K66F** development board, the system utilizes a hybrid software architecture combining high-level C logic for data processing with low-level Assembly drivers for precise GPIO control.

The system functions as a rhythmic trainer, using an interrupt-driven approach to measure user reaction times with millisecond precision.

Key System Features

- **Adjustable Visual Metronome**

Instead of audio, the system uses the Periodic Interrupt Timer (PIT) to generate a visual rhythm (Blue LED) suitable for hearing-impaired users or quiet environments. The tempo is fully adjustable via serial commands, ranging from 40 to 120 Beats Per Minute (BPM) to accommodate different rehabilitation stages.



- **Instant Biofeedback Loop**

The system provides immediate visual validation of the user's motor skills. By calculating the latency between the metronome beat and the user's input (SW2), the system delivers real-time feedback:

  - **Success (Green LED):** Input occurred within the strict $\pm$100ms tolerance window.
  - **Correction (Red LED):** Input was too early or too late, prompting the user to adjust their rhythm.

- **Quantitative Performance Analytics**

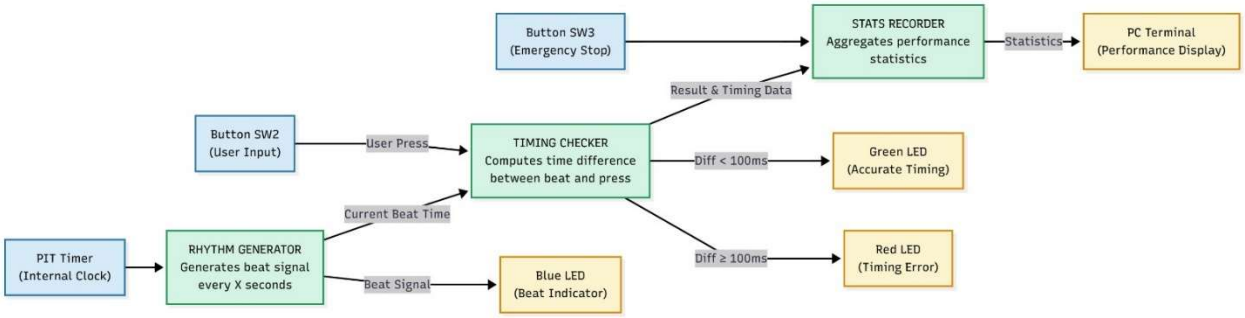
To track patient progress objectively, the system automatically aggregates session data. Upon session completion or interruption, a detailed report is transmitted via UART to a connected PC terminal, displaying:

- **Net Accuracy:** The percentage of successful hits.
- **Mean Error:** The average timing deviation in milliseconds.
- **Endurance:** Total number of attempted presses.

```
Good! Error: 61 ms
Good! Error: 53 ms
Good! Error: 26 ms
Good! Error: 67 ms
Good! Error: 52 ms
Good! Error: 41 ms
Good! Error: 28 ms
Good! Error: 58 ms
Off beat. Error: 124 ms
Good! Error: 56 ms
Good! Error: 29 ms
Good! Error: 19 ms

=== Session Complete ===
Total presses: 80
Accurate presses: 40
Accuracy: 50%
Mean error: 115 ms
Type 's' to start
```

Block Diagram and Implementation



1. Input & Timing Modules (Blue Blocks): These modules are responsible for gathering external signals and establishing the system's time base.

- **PIT Timer (Internal Clock):** This is the heartbeat of the system. It generates a precise, high-frequency signal (interrupt) that allows the software to measure time in milliseconds. It ensures that the rhythm remains steady regardless of what else the processor is doing.
- **Button SW2 (User Input):** This is the primary interaction point. It captures the exact moment the patient presses the button to match the rhythm.

- **Button SW3 (Emergency Stop):** This is a safety control. When pressed, it immediately interrupts the current session, stops the metronome, and triggers the final report generation.

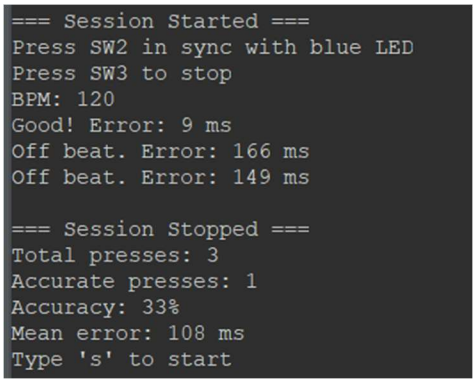
2. Logic & Processing Modules (Green Blocks): These represent the software "brain" that makes decisions based on the inputs.

- **Rhythm Generator:** Acting as a digital metronome, this module uses the time from the PIT Timer to decide when the next beat should occur. It triggers the visual cue (Blue LED) at the exact interval set by the user's BPM (Beats Per Minute).
- **Timing Checker:** This is the core analysis engine. When the user presses SW2, this module calculates the time difference (latency) between the *actual* press and the *expected* beat. It determines if the press was "Accurate" (within 100ms) or an "Error" (too early or too late).
- **Stats Recorder:** This module acts as a data aggregator. Throughout the session, it tallies the total number of presses, the number of accurate hits, and the average timing error. This data is stored in memory until the session ends.

3. Output & Feedback Modules (Yellow Blocks): These modules communicate the system status and results back to the user.

- **Blue LED (Beat Indicator):** This provides the visual stimulus. It flashes rhythmically to guide the user, replacing the sound of a traditional metronome.
- **Green LED (Accurate Timing):** This provides positive reinforcement. It lights up instantly when the Timing Checker determines the user's press was within the allowed tolerance (<100ms).
- **Red LED (Timing Error):** This provides corrective feedback. It lights up when the user's press is off-beat (≥100ms), signaling that they need to adjust their timing.
- **PC Terminal (Performance Display):** At the end of a session (or upon emergency stop), this interface displays the final report. It presents the statistics gathered by the Stats Recorder (e.g., "Accuracy: 85%"), allowing the therapist or patient to track progress over time.

Emergency Stop:



Implementation

Implemented in C Language (High-Level Logic) It handles the decision-making, math, and flow control.

- **Interrupt Service Routines (ISRs):** The PIT0_IRQHandler (Timer), PORTD_IRQHandler (Button SW2), and PORTA_IRQHandler (Button SW3) are written in C to handle events logically.
- **Timing Analysis:** The calculation of millisecond differences (tick_count - last_beat_time) and statistical averages is performed in C.
- **Command Parsing:** The processing of serial inputs (changing BPM, starting/stopping sessions) is handled by the C main loop.

Implemented in Assembly Language (Hardware Drivers) It handles the direct manipulation of the microcontroller's hardware registers.

- **LED Initialization (asm_led_init):** The code that directly accesses the System Integration Module (SIM) to enable clocks and configures the Port Control Registers (PCR) for GPIO mode is written in Assembly.
- **LED State Control (asm_led_set_...):** The functions that physically toggle the voltage on the pins (Red, Green, and Blue) using the GPIO Data Output Registers (PDOR) are optimized Assembly instructions. This isolates the hardware layer from the C logic.

Implementation Challenges & Solutions

During the development phase, we encountered specific hardware and logic integration issues. Below are the primary challenges and the engineering solutions applied.

Peripheral Configuration Discrepancy

- **The Problem:** Despite configuring the Interrupt Service Routines (ISRs) for the Timer (PIT) and GPIO buttons using the development environment's GUI configuration tools, the hardware remained unresponsive. The blue LED failed to blink, and button presses were not registered, indicating that the interrupts were not triggering the CPU.
- **The Solution:** We bypassed the auto-generated configuration code and manually enabled the specific interrupt in the C code by explicitly calling the CMSIS EnableIRQ() functions for PIT0, PORTA, and PORTD (suggested by Gemini).

Unidirectional Timing Logic

- **The Problem:** Our initial timing algorithm only calculated the time elapsed since the last beat. This created a logic flaw: if a user pressed the button slightly early for the upcoming beat (e.g., 50ms before the flash), the system interpreted it as being extremely late for the previous beat (e.g., 950ms late), resulting in incorrect "Red LED" error feedback.

- **The Solution:** We implemented a bidirectional timing check. The logic now determines if the press is closer to the previous beat or the next beat. If the time elapsed is greater than 50% of the beat interval, the system calculates the difference from the upcoming beat, ensuring early presses are correctly identified as "Accurate" or "Early" rather than "Late."

Advantages Over Existing Solutions

Our One-Hand Typing Trainer offers several key benefits compared to what's currently available. Unlike basic metronomes, which only provide a beat without telling you how well you're doing, our device gives you **immediate, clear feedback** on your timing and accuracy. While some commercial rehabilitation software offers similar feedback, it often comes with a high price tag (hundreds of dollars) and relies on specific tablets. Our solution, built on a low-cost microcontroller, is **much more affordable** and just as **portable** as a traditional metronome. It also provides **real-time measurements** of your performance and allows you to **easily adjust the difficulty** (BPM) to match your progress. Finally, because our project is **open-source**, it can be freely used, modified, and improved by others. In essence, our trainer proves that powerful, clinical-grade rehabilitation tools with detailed performance tracking can be made accessible and affordable using simple embedded technology.

Future Work

1. **Audio Integration**: Add PWM-based buzzer for auditory metronome and feedback tones
2. **Adaptive BPM**: Implement automatic difficulty adjustment based on rolling accuracy metrics
3. **Multi-Button Training**: Support 2-3 button alternating patterns for finger independence

Conclusion

We successfully developed a functional rhythm-based motor training device that provides quantitative feedback for rehabilitation. The system demonstrates accurate beat generation, low-latency input capture, and real-time performance metrics. By leveraging embedded systems and assembly-level programming, we created an affordable, portable solution that addresses gaps in current rehabilitation tools.

References

[1] Flint Rehab, "MusicGlove Hand Therapy," [Online]. Available: https://www.flintrehab.com/musicglove/

[2] NXP Semiconductors, "K66 Sub-Family Reference Manual," Rev. 4, 2020.