AIN SHAMS UNIVERSITY FACULTY OF ENGINEERING

CSE211s: Introduction to Embedded Systems

Spring 2025

Submitted to: Dr. Mohamed Hassan Elshafey

Eng. Ayman Bahria

Eng. Abdallah Awad

Name & ID	Omar Mohammed Mehany 2201058 Ammar Ahmed Mostafa 2200262 Tarek Hazem Salah 2200680

Introduction:

This report describes the implementation of a project for the CSE211s Introduction to Embed ded Systems course, Spring 2025. The project utilizes a NUCLEO-F401RE board interfaced with an Arduino Multifunction Shield to implement a Real-Time Clock (RTC) and an analog voltage display. The system displays elapsed time in minutes and seconds on a 7-segment dis play, with the ability to reset the clock using switch S1. When switch S3 is pressed, the display shows the voltage from an on-board potentiometer, measured via the on-chip ADC. The report details the code structure, startup code, main function, and Interrupt Service Routines (ISRs) of the provided C program.

Code Structure:

The program is a single C source file developed using the mbed framework, which abstracts low-level hardware details for the NUCLEO-F401RE.

The code is organized into sections for hardware configuration, helper functions, and the main program loop, ensuring modularity and clarity. The structure includes:

- **Header Inclusion:** The mbed.h header provides APIs for GPIO (DigitalOut, DigitalIn), analog input (AnalogIn), and timing functions (wait_us, get_ms_count)
- **Pin Assignments and Global Variables:**

– Shift Register Pins: shiftDataPin (D8), shiftClockPin (D7), and latchPin (D4) control the 7-segment display via a shift register.

– Input Pins: voltagePin (A0) reads the potentiometer voltage, resetButton (A1) is switch S1, and voltageButton (A3) is switch S3.

– Display Constants: segmentDigits[] defines 7-segment patterns for digits 0–9 (com mon anode, active LOW). digitSelectionMask[] specifies masks for selecting digits D0 D3.

- **Helper Functions:**

– shiftDataToRegister(): Sends 16-bit data (8-bit segment pattern + 8-bit digit mask) to the shift register to update the display.

– showVoltage(): Displays the potentiometer voltage in centivolts (e.g., 3.45V as 345) with a decimal point.

– readStableVoltage(): Averages 50 ADC readings to provide a stable voltage measure ment (0–5V).

– showTime(): Displays time in MM:SS format (e.g., 12:34).

- **Main Function:** Implements the core logic for timekeeping, voltage measurement, and display switching based on button inputs.

The code follows a simple, layered architecture:

- **Hardware Layer**: Managed by mbed, handling GPIO, ADC, and timer operations.
- **Application Layer:** Implements RTC and voltage display logic, meeting the project re quirements

Startup Code:

The startup code is not explicitly included in the source file, as it is provided by the mbed framework for the NUCLEO-F401RE. The startup process, typically defined in a file like startup_stm32f401xe.s, performs the following:

- **Interrupt Vector Table:** Initializes the vector table in FLASH memory, including the reset handler and default ISR addresses.
- **Stack and Heap Setup:** Sets the Main Stack Pointer (MSP) to the top of RAM and configures the heap, as defined in the mbed linker script.
- **Data Initialization:** Copies the .data section from FLASH to RAM and zeroes the .bss section.
- **System Configuration:** Initializes the system clock (e.g., 84 MHz for STM32F401RE) and enables peripherals via mbeds runtime.
- **Transition to main():** Calls the users main() function after setup.

In this project, the mbed framework ensures that the ADC, GPIO pins (D8, D7, D4, A0, A1, A3), and system timer are initialized before main() executes. The get_ms_count() function, critical for RTC, relies on mbeds timer setup.

Main Function:

The main() function serves as the entry point for the application logic, running in an infinite loop to meet the projects RTC and voltage display requirements. Its key tasks are:

- **Initialization:** Sets lastUpdateTime to the current system time using get_ms_count() to start the RTC from zero.
- **Main Loop:**

– **Timekeeping:**

∗ Reads the current time (currentTime) via get_ms_count().
∗ Increments secondsCounter every 1000 ms (1 second).

∗ Resets secondsCounter to 0 and increments minutesCounter when secondsCounter reaches 60.

– **Display Control:**

∗ If voltageButton (S3, A3) is pressed (active LOW, voltageButton == 0):

· Calls readStableVoltage() to measure the potentiometer voltage (0–5V).
· Convertsthevoltage to centivolts (e.g., 3.45V to 345) and displays it using showVoltage().

∗ Otherwise, displays the RTC time (MM:SS) using showTime(minutesCounter, secondsCounter).

– **Reset Logic:** Resets minutesCounter and secondsCounter to 0 if:

∗ resetButton (S1, A1) is pressed (active LOW, resetButton == 0).
∗ minutesCounter reaches 100 (additional feature beyond project requirements)

Project Requirements:

- **RTC:** The RTC starts from 00:00 after reset and displays minutes (D3, D2) and seconds (D1, D0), as required.
- **S1 Reset:** Pressing S1 (resetButton) resets the RTC to 00:00 at any time.
- **Voltage Display**: Pressing S3 (voltageButton) displays the potentiometer voltage (0–5V, scaled from ADCs 0.0–1.0 range) in volts with a decimal point (e.g., 3.45V). Releasing S3 resumes RTC display without stopping the clock.
- **Potentiometer Voltage**: The on-board potentiometer provides 0V (minimum) to 5V (max imum), as the ADC reference is tied to the 5V supply

Interrupt Service Routines (ISRs):

The provided code does not define any user-implemented ISRs, as the projects functionality is achieved through polling. The mbed framework handles interrupts internally for:

- **Timer Interrupts:** The get_ms_count() function relies on a system timer (e.g., SysTick), with ISRs managed by mbed.

- **ADCInterrupts:** The AnalogIn object (voltagePin) may use ADC interrupts for conver sions, abstracted by mbed.

- **GPIO Interrupts:** Switches S1 (resetButton) and S3 (voltageButton) are polled in main(), not interrupt-driven.

Polling is sufficient for this project due to the low frequency of button presses and the sim plicity of the display updates. If interrupts were needed (e.g., for debouncing S1/S3), mbeds InterruptIn class could be used, with an ISR like:

```
void button_isr() {
    // Handle button press (e.g., set a flag)
}
```

Conclusion:

The project successfully implements an RTC and voltage display using the NUCLEO-F401RE and Arduino Multifunction Shield. The code is structured as a single C file with mbed framework dependencies, featuring:

- **Code Structure:** Clear organization with pin definitions, helper functions, and a main loop.
- **Startup Code:** Handled by mbed, initializing hardware and calling main().
- **Main Function:** Manages RTC, voltage measurement, and display switching via polling.
- **ISRs:** None user-defined**;** mbed handles timer and ADC interrupts.