Ain Shams University

Department of Mechatronics Engineering

Into into Embeded systems[CSE211]

Submitted to:

Dr. **Mohamed Hassan El-Shafey**

Adham Sabry	**2100703**
Kholoud Ahmed	**2100712**
Ahmed Ismail	**2100722**

Table of Contents

Introduction

This embedded systems project integrates a real-time clock (RTC) and analog voltage display using the NUCLEO-F401RE development board and an Arduino Multifunction Shield. The goal is to familiarize students with real-time counters, ADCs, digital displays, and push-button inputs. This hands-on experience highlights the practical application of hardware interfacing, shift registers, and MBED OS-based system design.

1. Objective

This project utilizes the NUCLEO-F401RE microcontroller board in conjunction with the Arduino Multifunction Shield to build a real-time clock (RTC) system. The system continuously displays elapsed time on a 7-segment display in minutes and seconds, and allows the user to view analog input voltage from the on-board potentiometer upon pressing a button.

2. Hardware Components

- NUCLEO-F401RE microcontroller board
- Arduino Multifunction Shield
- USB cable (for programming and power)

The Arduino Multifunction Shield is directly mounted onto the NUCLEO board, ensuring all digital and analog pins are properly aligned. No additional wiring is required.

3. Software and Development Tools

- MBED Studio IDE
- MBED OS 6.x
- NUCLEO-F401RE board configuration

4. System Features

4.1 Real-Time Clock (RTC)

- Starts counting from 00:00 after reset.

- Displays time in MM:SS format on the 7-segment display.
- Time increments every second.
- S1 (connected to pin A1) resets the clock to 00:00.

4.2 Analog Voltage Display

- Reads analog voltage from the on-board potentiometer via ADC input (A0).
- When S3 (connected to pin A3) is pressed, displays the voltage in volts.
- Once S3 is released, the system reverts to displaying the time.
- The clock continues running in the background during voltage display.

5. Code Structure

5.1 Startup Code

This section includes the necessary hardware initializations:

- DigitalOut pins for the 74HC595 shift register connected to the 7-segment display (D4, D7, D8).
- DigitalIn inputs for the push buttons S1 (A1) and S3 (A3), configured with internal pull-ups.
- AnalogIn for reading the analog voltage from the potentiometer (A0).
- A Ticker object is configured to call the updateTime() function every 1 second.

```
s1.mode(PullUp);
s3.mode(PullUp);
timerTicker.attach(&updateTime, 1.0f);
```

5.2 Main Loop

The while (1) loop performs the following:

- Monitors button S1 to reset time to 00:00.
- Reads and scales analog voltage from the potentiometer.
- Updates min and max voltage levels.
- Displays either time or voltage based on the state of button S3:

- If S3 is pressed: shows scaled voltage (e.g., 2.45V → 245) with a decimal point.
- If S3 is not pressed: shows current time in MMSS format.

```
if (!s3) {
  int voltageScaled = static_cast<int>(voltage * 100);
  displayNumber(voltageScaled, true, 1);
} else {
  int timeDisplay = minutes * 100 + seconds;
  displayNumber(timeDisplay);
}
```

5.3 ISR: Ticker Function

This function runs every 1 second using the Ticker object. It:

- Increments the seconds counter.
- Rolls over to the next minute after 60 seconds.
- Resets to 00:00 after reaching 99:59 due to 2-digit minute limitation.

```
void updateTime() {
  seconds++;
  if (seconds >= 60) {
    seconds = 0;
    minutes = (minutes + 1) % 100;
  }
}
```

6. Functionality Demonstration

- Upon power-up, the display starts from 00:00.

- Each second, the display updates.
- Pressing S1 at any point resets the time.
- Pressing and holding S3 shows the voltage from the potentiometer.
- Releasing S3 reverts the display to time, which resumes without disruption.

7. Conclusion

This project demonstrates the implementation of a basic embedded system using MBED OS, interrupts, real-time counters, analog signal acquisition through ADC, and multiplexed 7-segment display control via a shift register. The system may be extended for more complex functionalities like time setting, alarm triggering, or wireless communication.

8. Appendix

8.1 Minimum and Maximum Potentiometer Voltage

- **Minimum Voltage:** ~0.0 V
- **Maximum Voltage:** ~3.3 V (dependent on board VCC)

8.2 Deliverables

- **Video:** Demonstrates the real-time clock and voltage display in action.
- **Report:** This document.
- **Code:**5.3 ISR: Ticker Function

This function runs every 1 second using the Ticker object. It:

  - Increments the seconds counter.
  - Rolls over to the next minute after 60 seconds.
  - Resets to 00:00 after reaching 99:59 due to 2-digit minute limitation.

```
void updateTime() {
  seconds++;
  if (seconds >= 60) {
    seconds = 0;
    minutes = (minutes + 1) % 100;
  }
```

}

6. Functionality Demonstration

- Upon power-up, the display starts from 00:00.
- Each second, the display updates.
- Pressing S1 at any point resets the time.
- Pressing and holding S3 shows the voltage from the potentiometer.
- Releasing S3 reverts the display to time, which resumes without disruption.

7. Conclusion

This project demonstrates the implementation of a basic embedded system using MBED OS, interrupts, real-time counters, analog signal acquisition through ADC, and multiplexed 7-segment display control via a shift register. The system may be extended for more complex functionalities like time setting, alarm triggering, or wireless communication.

8. Appendix

8.1 Minimum and Maximum Potentiometer Voltage

- **Minimum Voltage:** ~0.0 V
- **Maximum Voltage:** ~3.3 V (dependent on board VCC)

8.2 Deliverables

- **Code**

```
#include "mbed.h"


// Shift register pins (common anode)
DigitalOut latchPin(D4);
DigitalOut clockPin(D7);
DigitalOut dataPin(D8);
```

```
// Buttons (S1 for reset, S3 for voltage display)
DigitalIn s1(A1), s3(A3);

// ADC for potentiometer (A0)
AnalogIn pot(A0);

// Common anode segment patterns (0-9 with decimal point)
const uint8_t digitPattern[10] = {
  static_cast<uint8_t>(~0x3F), // 0
  static_cast<uint8_t>(~0x06), // 1
  static_cast<uint8_t>(~0x5B), // 2
  static_cast<uint8_t>(~0x4F), // 3
  static_cast<uint8_t>(~0x66), // 4
  static_cast<uint8_t>(~0x6D), // 5
  static_cast<uint8_t>(~0x7D), // 6
  static_cast<uint8_t>(~0x07), // 7
  static_cast<uint8_t>(~0x7F), // 8
  static_cast<uint8_t>(~0x6F)  // 9
};

// Digit positions (left to right)
const uint8_t digitPos[4] = {0x01, 0x02, 0x04, 0x08};

// Timer and voltage variables
volatile int seconds = 0, minutes = 0;
```

```
volatile float minVoltage = 3.3f, maxVoltage = 0.0f;
Ticker timerTicker;

// Function prototypes
void updateTime();
void shiftOutMSBFirst(uint8_t value);
void writeToShiftRegister(uint8_t segments, uint8_t digit);
void displayNumber(int number, bool showDecimal = false, int decimalPos = -1);

// ISR for timer updates
void updateTime() {
  seconds++;
  if (seconds >= 60) {
    seconds = 0;
    minutes = (minutes + 1) % 100;
  }
}

// Shift register driver
void shiftOutMSBFirst(uint8_t value) {
  for (int i = 7; i >= 0; i--) {
    dataPin = (value & (1 << i)) ? 1 : 0;
    clockPin = 1;
    clockPin = 0;
  }
}
```

```
// Write data to shift register
void writeToShiftRegister(uint8_t segments, uint8_t digit) {
  latchPin = 0;
  shiftOutMSBFirst(segments);
  shiftOutMSBFirst(digit);
  latchPin = 1;
}

// Display number with optional decimal point
void displayNumber(int number, bool showDecimal, int decimalPos) {
  int digits[4] = {
    (number / 1000) % 10,
    (number / 100) % 10,
    (number / 10) % 10,
    number % 10
  };

  for (int i = 0; i < 4; i++) {
    uint8_t pattern = digitPattern[digits[i]];
    if (showDecimal && i == decimalPos) pattern &= ~0x80; // Enable DP
    writeToShiftRegister(pattern, digitPos[i]);
    ThisThread::sleep_for(2ms);
  }
}
```

```
int main() {
  // Initialize hardware
  s1.mode(PullUp);
  s3.mode(PullUp);
  timerTicker.attach(&updateTime, 1.0f); // 1-second updates

  while (1) {
    // Reset timer on S1 press
    if (!s1) {
      seconds = 0;
      minutes = 0;
      ThisThread::sleep_for(200ms); // Debounce
    }

    // Read potentiometer voltage
    float voltage = pot.read() * 3.3f; // Convert to volts (0-3.3V)

    // Update min/max voltages
    if (voltage < minVoltage) minVoltage = voltage;
    if (voltage > maxVoltage) maxVoltage = voltage;

    // Display voltage if S3 pressed, else display time
    if (!s3) {
      int voltageScaled = static_cast<int>(voltage * 100); // e.g., 2.45V → 245
      displayNumber(voltageScaled, true, 1); // Show as X.XX
    } else {
```

```
        int timeDisplay = minutes * 100 + seconds; // MMSS format
        displayNumber(timeDisplay);
      }
    }
}
```

- **Video:** Demonstrates the real-time clock and voltage display in action.

  https://drive.google.com/drive/folders/1E5YHmWzo0RAoc6Xj658yZC7uGXRHUVWC?usp=sharing