CSE211s

Final Course Project

Report on Digital Timer and Voltmeter Using MultiFunction Shield with STM32 (NUCLEO-F401RE)

Mariam Ashraf Nabil	2200421	EPM
Sohailla Ahmed Fathy	2000354	EPM
Malak Mohamed Mostafa	2200462	EPM

Task 1:

1. Purpose:

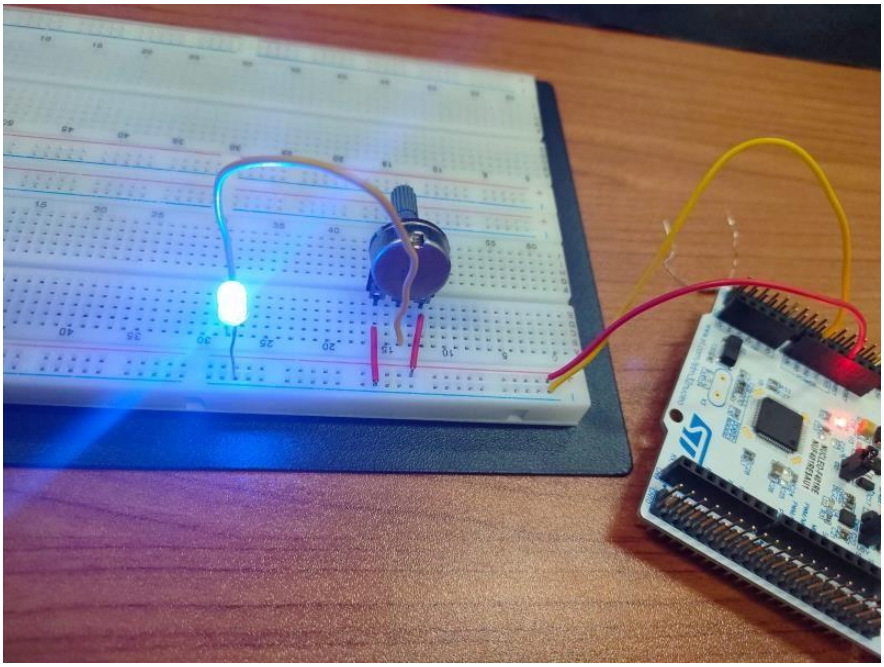
- **Learn how to read sensors and control lighting.**
  This code shows how to read a knob (potentiometer) and use its value to control the brightness of a light (LED).
- **Ensure that your configuration works properly.**
  Running this easy test assures that your board and equipment are ready to undertake a larger project.
- **Understand the board's main components.**
  You will learn how to use the input pin to read the knob and the output pin to control the LED. This is important for all projects.
- **Lay the groundwork for large-scale efforts.**
  Later, when your project grows more complicated, you will use the same ideas but with more elements.
- **Improve timing and control.**
  This example demonstrates how to govern things step by step, which will help you handle larger projects in the future.

2.Code:

```
#include "mbed.h"

// Potentiometer connected to A0
AnalogIn pot(A0);

// LED connected to D3 (supports PWM)
PwmOut led(D3);

int     main()     {
while (true) {
      float brightness = pot.read(); // Read from potentiometer (0.0 to 1.0)
      led.write(brightness);      // Set LED brightness      wait_us(10000);
// Small delay (10 ms)
   }
}
```

---

3. Circuit picture:



Task 2:

1. Introduction

This microcontroller-based system project uses a STM32 NUCLEO-F401RE microcontroller and a Multi-Function Shield to provide a **dual-mode digital display**:

- **Timer Mode**: Displays time as MM:SS.
- **Voltage Mode**: Displays analog input voltage as X.XXX.

The system controls a 4-digit 7-segment display with 74HC595 shift registers, an analog input potentiometer, and push buttons for resetting or changing modes.

---

2. Code Structure

2.1 Startup and Configuration

Code begins with establishing pin assignments between the NUCLEO board and the MultiFunction Shield:

- **DigitalOut**: to control shift registers (latch, clock, and data).
- **DigitalIn**: to read buttons.
- **AnalogIn**: to read voltage from a potentiometer.

Additionally, two crucial arrays are initialized:

- SEGMENT_MAP: Converts (0–9) to 7-segment binary encoding.
- SEGMENT_SELECT: Allows you to choose each position of the digit independently. 2.2

Global State and Tickers

The system uses:

- Ticker interrupts for periodic events (1-second timer tick and 2ms display refresh).
- Shared variables (suh, su_disp, su_digit) utilized for time, display modifications, and digit updates.

---

3. Interrupt Service Routines (ISRs)

3.1 suhaila_tick()

It is called every 1 second using second_tick. It increases the global seconds counter and wraps at 99:59 (6000 seconds).

3.2 suhaila_refresh()

It is called every 2 milliseconds via refresh_tick. Sets a flag (su_disp = true) to allow the main loop to refresh one digit of the display.

---

4. Main Function

4.1 Initialization

- Buttons are configured with internal pull-ups.
- Ticker ISRs are attached.

4.2 Main Loop

Runs indefinitely, performing:

- **Button Check**:
  - **S1**: Resets the timer when pressed.
  - **S3**: Switches display mode to voltage reading while held down.
- **Display Refresh**:
  - Displays either **MM:SS** or **X.XXX volts**, depending on mode. ○ Refreshes one digit per iteration to minimize flicker (multiplexing). ○ Uses suhaila_out() to send data to shift registers.

---

5. Full Code with Comments

```
#include "mbed.h"

// Define shield connections (NUCLEO-F401RE pins)
DigitalOut latch(PB_5);    // Latch Pin (D4)
DigitalOut clk(PA_8);      // Clock Pin (D7)
DigitalOut data(PA_9);     // Data Pin (D8)
DigitalIn  button1(PA_1);  // Reset button Pin (S1)
DigitalIn  button3(PB_0);  // Mode button Pin (S3)
AnalogIn   pot(PA_0);      // Potentiometer Pin (A0)

// 7-segment encoding (Common Anode – active low) const
uint8_t SEGMENT_MAP[10] =
{
   0xC0, 0xF9, 0xA4, 0xB0, 0x99,
   0x92, 0x82, 0xF8, 0x80, 0x90
};

// Digit selector (active low, 4-digit)
const uint8_t SEGMENT_SELECT[4] = { 0xF1, 0xF2, 0xF4, 0xF8 };

// Shared state
volatile int suh = 0;                  // Counter for total seconds volatile
bool su_disp = true;        // Display update flag
Ticker  second_tick;  Ticker
refresh_tick;
volatile int su_digit = 0;           // Current digit to refresh

// ISR: Increment seconds counter every 1 sec
void suhaila_tick() {    suh++;
   if (suh >= 6000) suh = 0;  // Wrap around at 99:59
}

// ISR: Trigger display refresh flag every 2 ms void
suhaila_refresh() {
   su_disp = true;
}

//    Send    data    to    74HC595    shift    registers     void
suhaila_out(uint8_t segments, uint8_t digitSelect) {    latch
= 0;    for (int i = 7; i >= 0; --i) {        data = (segments >> i) &
0x1;        clk = 0; clk = 1;
   }
```

```
#include "mbed.h"

// Define shield connections (NUCLEO-F401RE pins)
DigitalOut latch(PB_5);     // Latch Pin (D4)
DigitalOut clk(PA_8);       // Clock Pin (D7)
DigitalOut data(PA_9);      // Data Pin (D8)
DigitalIn  button1(PA_1);   // Reset button Pin (S1)
DigitalIn  button3(PB_0);   // Mode button Pin (S3)
AnalogIn   pot(PA_0);       // Potentiometer Pin (A0)

// 7-segment encoding (Common Anode - active low)
const uint8_t SEGMENT_MAP[10] = {
    0xC0, 0xF9, 0xA4, 0xB0, 0x99,
    0x92, 0x82, 0xF8, 0x80, 0x90
};

// Digit selector (active low, 4-digit)
const uint8_t SEGMENT_SELECT[4] = { 0xF1, 0xF2, 0xF4, 0xF8 };

// Shared state
volatile int suh = 0;                   // Counter for total seconds
volatile bool su_disp = true;           // Display update flag
Ticker second_tick;
Ticker refresh_tick;
volatile int su_digit = 0;              // Current digit to refresh

// ISR: Increment seconds counter every 1 sec
void suhaila_tick() {
    suh++;
    if (suh >= 6000) suh = 0;  // Wrap around at 99:59
}

// ISR: Trigger display refresh flag every 2 ms
void suhaila_refresh() {
    su_disp = true;
}

// Send data to 74HC595 shift registers
void suhaila_out(uint8_t segments, uint8_t digitSelect) {
    latch = 0;
    for (int i = 7; i >= 0; --i) {
        data = (segments >> i) & 0x1;
        clk = 0; clk = 1;
    }
```

embedded system code is written for the NUCLEO-F401RE board using Mbed OS. It controls a 4-digit 7-segment display via 74HC595 shift registers. The display shows either elapsed time in MM:SS or the voltage from potentiometer, depending on which mode is selected using a button.

Connections:

latch, clk, data: Used to control the shift registers (74HC595).

button1 (S1): Resets the timer.

button3 (S3): Switches between time and voltage display modes.

pot: Reads analog voltage from a potentiometer.

Segment Display Control:

SEGMENT_MAP[10]: Contains the binary patterns to display digits 0–9 on a common anode 7-segment display.

SEGMENT_SELECT[4]: Used to choose which of the 4 digits is currently active (multiplexing).

Timing with Interrupts:

A Ticker calls suhaila_tick() every second to increase the counter suh, which tracks total time in seconds.

Another Ticker calls suhaila_refresh() every 2 ms to trigger a display update by setting a flag su_disp = true.

Display Output Function:

The function suhaila_out() shifts out two bytes:

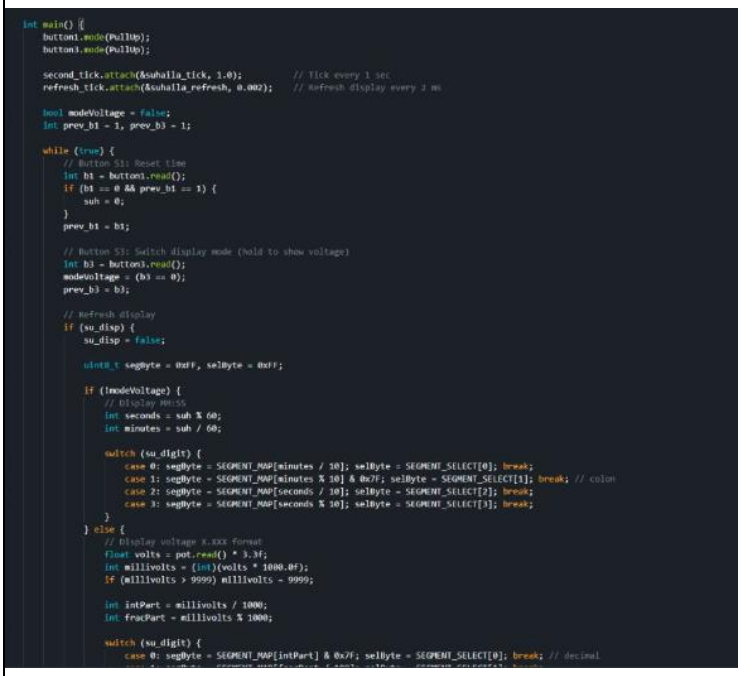
Segment data (which segments to light up)
Digit selection (which digit to activate)
It uses digital pins and toggles latch, clk, and data to communicate with the shift register.

Code with screens and explanation

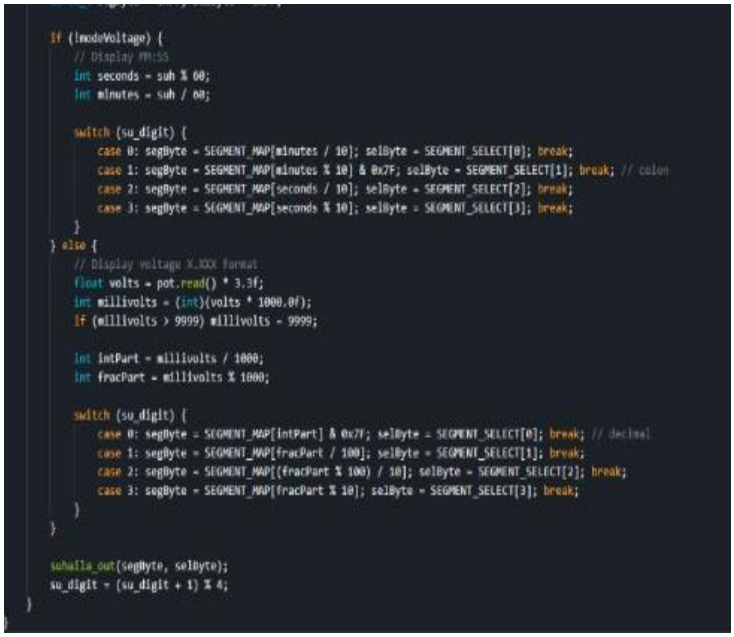
```
for (int i = 7; i >= 0; --i) {        data =
(digitSelect >> i) & 0x1;        clk = 0;
clk = 1;
   }
   latch = 1;
}

int main() {
   button1.mode(PullUp);    button3.mode(PullUp);

   second_tick.attach(&suhaila_tick, 1.0);         // Tick every 1 sec
   refresh_tick.attach(&suhaila_refresh, 0.002);    // Refresh display every 2 ms

   bool modeVoltage = false;
   int prev_b1 = 1, prev_b3 = 1;

   while (true) {
      // Button S1: Reset time
int b1 = button1.read();
      if (b1 == 0 && prev_b1 == 1) {
suh = 0;
      }
      prev_b1 = b1;

      // Button S3: Switch display mode (hold to show voltage)
int b3 = button3.read();      modeVoltage = (b3 == 0);      prev_b3
= b3;

      // Refresh display
if        (su_disp)        {
su_disp = false;

         uint8_t segByte = 0xFF, selByte = 0xFF;

         if       (!modeVoltage)       {
// Display MM:SS                    int
seconds = suh % 60;
            int minutes = suh / 60;
```

```
            switch (su_digit) {
              case 0: segByte = SEGMENT_MAP[minutes / 10]; selByte = SEGMENT_SELECT[0]; break;
case 1: segByte = SEGMENT_MAP[minutes % 10] & 0x7F; selByte = SEGMENT_SELECT[1]; break; // colon
              case 2: segByte = SEGMENT_MAP[seconds / 10]; selByte = SEGMENT_SELECT[2]; break;
case 3: segByte = SEGMENT_MAP[seconds % 10]; selByte = SEGMENT_SELECT[3]; break;
            }
          } else {
            // Display voltage X.XXX format
float volts = pot.read() * 3.3f;              int
millivolts = (int)(volts * 1000.0f);              if
(millivolts > 9999) millivolts = 9999;

            int intPart = millivolts / 1000;
int fracPart = millivolts % 1000;

            switch (su_digit) {
              case 0: segByte = SEGMENT_MAP[intPart] & 0x7F; selByte = SEGMENT_SELECT[0]; break;
// decimal
              case 1: segByte = SEGMENT_MAP[fracPart / 100]; selByte = SEGMENT_SELECT[1]; break;
case 2: segByte = SEGMENT_MAP[(fracPart % 100) / 10]; selByte = SEGMENT_SELECT[2]; break;
              case 3: segByte = SEGMENT_MAP[fracPart % 10]; selByte = SEGMENT_SELECT[3]; break;
            }
          }

          suhaila_out(segByte, selByte);
          su_digit = (su_digit + 1) % 4;
        }
      }
```

}



```
int main() {
    button1.mode(PullUp);
    button3.mode(PullUp);

    second_tick.attach(&suhaila_tick, 1.0);         // Tick every 1 sec
    refresh_tick.attach(&suhaila_refresh, 0.002);   // Refresh display every 2 ms

    bool modeVoltage = false;
    int prev_b1 = 1, prev_b3 = 1;

    while (true) {
        // Button S1: Reset time
        int b1 = button1.read();
        if (b1 == 0 && prev_b1 == 1) {
            suh = 0;
        }
        prev_b1 = b1;

        // Button S3: Switch display mode (hold to show voltage)
        int b3 = button3.read();
        modeVoltage = (b3 == 0);
        prev_b3 = b3;

        // Refresh display
        if (su_disp) {
            su_disp = false;

            uint8_t segByte = 0xFF, selByte = 0xFF;

            if (!modeVoltage) {
                // Display MM:SS
                int seconds = suh % 60;
                int minutes = suh / 60;

                switch (su_digit) {
                    case 0: segByte = SEGMENT_MAP[minutes / 10]; selByte = SEGMENT_SELECT[0]; break;
                    case 1: segByte = SEGMENT_MAP[minutes % 10] & 0x7F; selByte = SEGMENT_SELECT[1]; break; // colon
                    case 2: segByte = SEGMENT_MAP[seconds / 10]; selByte = SEGMENT_SELECT[2]; break;
                    case 3: segByte = SEGMENT_MAP[seconds % 10]; selByte = SEGMENT_SELECT[3]; break;
                }
            } else {
                // Display voltage X.XXX format
                float volts = pot.read() * 3.3f;
                int millivolts = (int)(volts * 1000.0f);
                if (millivolts > 9999) millivolts = 9999;

                int intPart = millivolts / 1000;
                int fracPart = millivolts % 1000;

                switch (su_digit) {
                    case 0: segByte = SEGMENT_MAP[intPart] & 0x7F; selByte = SEGMENT_SELECT[0]; break; // decimal
```



```
    if (!modeVoltage) {
        // Display MM:SS
        int seconds = suh % 60;
        int minutes = suh / 60;

        switch (su_digit) {
            case 0: segByte = SEGMENT_MAP[minutes / 10]; selByte = SEGMENT_SELECT[0]; break;
            case 1: segByte = SEGMENT_MAP[minutes % 10] & 0x7F; selByte = SEGMENT_SELECT[1]; break; // colon
            case 2: segByte = SEGMENT_MAP[seconds / 10]; selByte = SEGMENT_SELECT[2]; break;
            case 3: segByte = SEGMENT_MAP[seconds % 10]; selByte = SEGMENT_SELECT[3]; break;
        }
    } else {
        // Display voltage X.XXX format
        float volts = pot.read() * 3.3f;
        int millivolts = (int)(volts * 1000.0f);
        if (millivolts > 9999) millivolts = 9999;

        int intPart = millivolts / 1000;
        int fracPart = millivolts % 1000;

        switch (su_digit) {
            case 0: segByte = SEGMENT_MAP[intPart] & 0x7F; selByte = SEGMENT_SELECT[0]; break; // decimal
            case 1: segByte = SEGMENT_MAP[fracPart / 100]; selByte = SEGMENT_SELECT[1]; break;
            case 2: segByte = SEGMENT_MAP[(fracPart % 100) / 10]; selByte = SEGMENT_SELECT[2]; break;
            case 3: segByte = SEGMENT_MAP[fracPart % 10]; selByte = SEGMENT_SELECT[3]; break;
        }
    }

    suhaila_out(segByte, selByte);
    su_digit = (su_digit + 1) % 4;
}
```

This program controls a 4-digit 7-segment display connected to the NUCLEO-F401RE board using the Mbed OS library. The display is driven by two 74HC595 shift registers connected via three pins: latch, clk, and data.

At the start, two buttons are set up:

- button1 (reset button)
- button3 (mode switch button)
  Both buttons use pull-up mode, so they read 1 when not pressed and 0 when pressed.

Two timers (Tickers) are started:

- second_tick calls the function suhaila_tick every 1 second to increase the seconds counter suh.
- refresh_tick calls the function suhaila_refresh every 2 milliseconds to signal when to update the display.

Inside the main loop:

- The program reads the buttons' states.
- If the reset button (button1) is pressed, the seconds counter is reset to zero.

- If the mode button (button3) is pressed, the display mode switches between showing elapsed time or the potentiometer voltage.

When the display needs updating (flag su_disp is true):

- The program calculates which digit to show on the current 7-segment display position (su_digit), based on the selected mode (time or voltage).
- In time mode: the total seconds suh are converted into minutes and seconds and displayed in MM:SS format.
- In voltage mode: the analog voltage from the potentiometer is read, converted to a decimal format (X.XXX volts), and displayed.
- The function suhaila_out is called to send the segment data (segByte) and digit select data (selByte) to the shift registers, which control the LEDs on the display.
- The digit counter su_digit is incremented to cycle through all 4 digits rapidly (multiplexing), so all digits appear lit continuously.

---

6. Video link during operation:

https://drive.google.com/file/d/1-UAKupa_B3oNdbzofihhfN3bNHHTJC6Z/view?usp=drive_link