Faculty of Engineering - Ain Shams University

Department of Computer Science and Engineering

Embedded Systems Course Project

Duel-mode Digital Display System Using STM32 NUCLEO-F401RE

Saif Khaled Saleh	**2200693**	**EPM**
Ahmed Ashraf Helmy	**2200680**	**EPM**

Task 1: Sensor Input and Output Control

1. Purpose

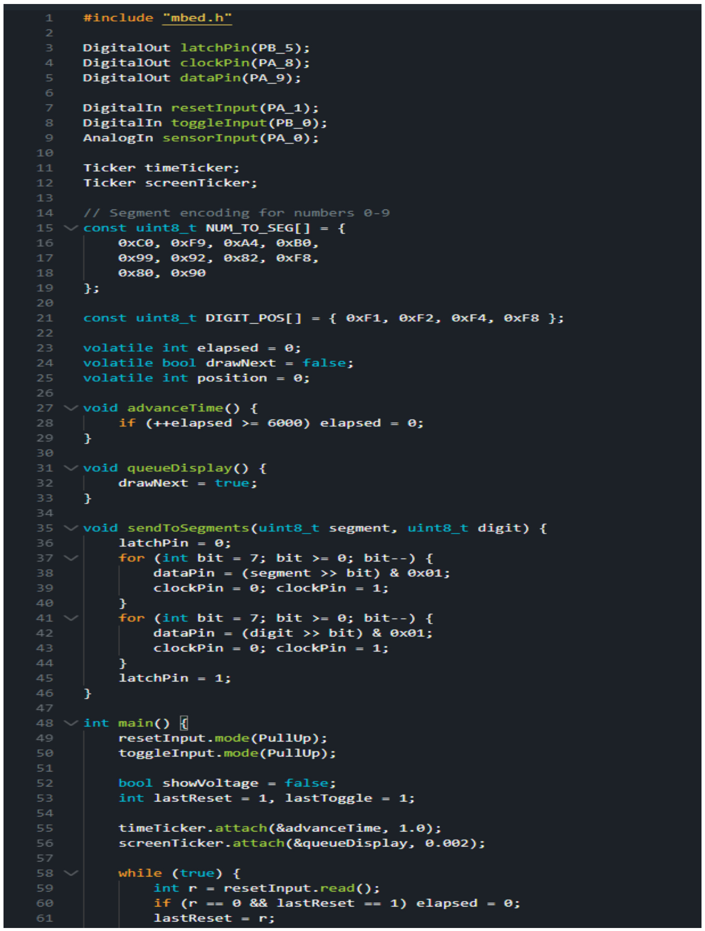
- Learn sensor interfacing and display control.
  This code demonstrates reading analog input from a potentiometer and dynamically updating a 7-segment display.

- Validate hardware configuration.
  Ensures proper communication between the microcontroller, shift registers, and peripherals.

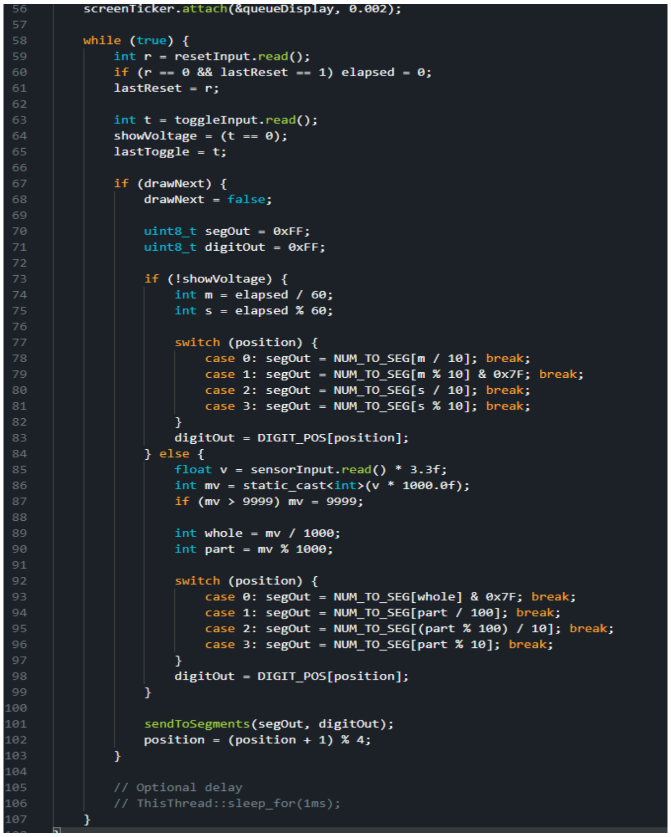
- Master core components.
  Familiarize with GPIO configuration, PWM-free brightness control via multiplexing, and shift register communication.

- Foundation for complex systems.
  Establishes principles applicable to advanced projects involving sensors and multi-element displays.

2.Code:



```cpp
#include "mbed.h"

DigitalOut latchPin(PB_5);
DigitalOut clockPin(PA_8);
DigitalOut dataPin(PA_9);

DigitalIn resetInput(PA_1);
DigitalIn toggleInput(PB_0);
AnalogIn sensorInput(PA_0);

Ticker timeTicker;
Ticker screenTicker;

// Segment encoding for numbers 0-9
const uint8_t NUM_TO_SEG[] = {
    0xC0, 0xF9, 0xA4, 0xB0,
    0x99, 0x92, 0x82, 0xF8,
    0x80, 0x90
};

const uint8_t DIGIT_POS[] = { 0xF1, 0xF2, 0xF4, 0xF8 };

volatile int elapsed = 0;
volatile bool drawNext = false;
volatile int position = 0;

void advanceTime() {
    if (++elapsed >= 6000) elapsed = 0;
}

void queueDisplay() {
    drawNext = true;
}

void sendToSegments(uint8_t segment, uint8_t digit) {
    latchPin = 0;
    for (int bit = 7; bit >= 0; bit--) {
        dataPin = (segment >> bit) & 0x01;
        clockPin = 0; clockPin = 1;
    }
    for (int bit = 7; bit >= 0; bit--) {
        dataPin = (digit >> bit) & 0x01;
        clockPin = 0; clockPin = 1;
    }
    latchPin = 1;
}

int main() {
    resetInput.mode(PullUp);
    toggleInput.mode(PullUp);

    bool showVoltage = false;
    int lastReset = 1, lastToggle = 1;

    timeTicker.attach(&advanceTime, 1.0);
    screenTicker.attach(&queueDisplay, 0.002);

    while (true) {
        int r = resetInput.read();
        if (r == 0 && lastReset == 1) elapsed = 0;
        lastReset = r;
```



```cpp
    screenTicker.attach(&queueDisplay, 0.002);

    while (true) {
        int r = resetInput.read();
        if (r == 0 && lastReset == 1) elapsed = 0;
        lastReset = r;

        int t = toggleInput.read();
        showVoltage = (t == 0);
        lastToggle = t;

        if (drawNext) {
            drawNext = false;

            uint8_t segOut = 0xFF;
            uint8_t digitOut = 0xFF;

            if (!showVoltage) {
                int m = elapsed / 60;
                int s = elapsed % 60;

                switch (position) {
                    case 0: segOut = NUM_TO_SEG[m / 10]; break;
                    case 1: segOut = NUM_TO_SEG[m % 10] & 0x7F; break;
                    case 2: segOut = NUM_TO_SEG[s / 10]; break;
                    case 3: segOut = NUM_TO_SEG[s % 10]; break;
                }
                digitOut = DIGIT_POS[position];
            } else {
                float v = sensorInput.read() * 3.3f;
                int mv = static_cast<int>(v * 1000.0f);
                if (mv > 9999) mv = 9999;

                int whole = mv / 1000;
                int part = mv % 1000;

                switch (position) {
                    case 0: segOut = NUM_TO_SEG[whole] & 0x7F; break;
                    case 1: segOut = NUM_TO_SEG[part / 100]; break;
                    case 2: segOut = NUM_TO_SEG[(part % 100) / 10]; break;
                    case 3: segOut = NUM_TO_SEG[part % 10]; break;
                }
                digitOut = DIGIT_POS[position];
            }

            sendToSegments(segOut, digitOut);
            position = (position + 1) % 4;
        }

        // Optional delay
        // ThisThread::sleep_for(1ms);
    }
```

Task 2: System Design and Operation

1. Introduction

This project implements a dual-mode digital display using the STM32 NUCLEO-F401RE and a Multi-Function Shield. Key features include:

- **Timer Mode**: Displays elapsed time in MM:SS format.
- **Voltage Mode**: Shows analog input voltage in X.XXX V format.
  Components include 74HC595 shift registers, a 4-digit 7-segment display, push buttons, and a potentiometer.

2. Code Structure

2.1 Hardware Configuration

- **DigitalOut**: Controls shift registers (latchPin, clockPin, dataPin).
- **DigitalIn**: Reads button states (resetInput, toggleInput).
- **AnalogIn**: Reads potentiometer voltage (sensorInput).

2.2 Global Variables and Interrupts

- **NUM_TO_SEG**: Maps digits (0–9) to 7-segment patterns.
- **DIGIT_POS**: Selects active digit via multiplexing.
- **Tickers**:
  - timeTicker: Increments elapsed every second.

   - screenTicker: Triggers display refresh every 2ms.

3. Key Functions

3.1 advanceTime()

- Increments elapsed every second, wrapping at 6000 seconds (99:59).

3.2 sendToSegments()

- Transmits segment and digit data to shift registers using bit-shifting.

4. Main Loop Workflow

1. **Button Handling**:
   - **Reset**: Resets elapsed to 0 when resetInput is pressed.
   - **Mode Toggle**: Switches between timer and voltage modes using toggleInput.
2. **Display Update**:
   - **Timer Mode**: Splits elapsed into minutes and seconds.
   - **Voltage Mode**: Converts analog input to millivolts and formats as X.XXX.
3. **Multiplexing**: Cycles through digits rapidly to reduce flicker.

5. Demonstration video

https://drive.google.com/file/d/1jnRnxYZksTrob9_xqjG9Xt2BPM0aGScD/view?usp=drive_link