1. Introduction

The purpose of this lab is to explore the design of firmware to drive a matrix keypad and general purpose timers. Both blocking and non-blocking algorithms will need to be implemented to detect presses of the keypad. A timer API will be created and used track the amount of time in MS to identify random Hexadecimal patterns.

2. Prerequisites

- StmCubeIDE is installed on your machine.
- Your MSOE Development Board with the following components assembled:
    - STM32 Microcontroller
    - LED Bar Array (with drivers)
    - LCD Display (with drivers)
    - SysTick Counter (with drivers)
    - Wired and functional 4x4 Keypad

3. Activities

3.1 Create Lab4Keypad Project

- Copy your working Lab3LcdMaze Project within CubeIDE in the ProjectExplorer tab by copy-and-pasting.
- Name your project Lab4HexKeys
- Start a fresh main.c file

3.3 Keypad APIs

Reference the schematic and use your gpio.h structs to implement a keypad api with the following functions

- void KeypadInit() - Sets up pins for the keypad.
- uint8_t KeypadStatus() - non-blocking method which returns 0 if no key is pressed or the first row indices(1 2 3 4), second row indices(5 6 7 8), ...
- uint8_t KeypadGetKey() - blocking implementation which returns indices 1-16 and handles contact bounce.

Hints:

- Don't reinvent the wheel. You can use KeypadStatus() in the implementation of KeypadGetKey after you verify it is functional.
- If using Algorithm #2 from class, you may need a small delay(5us) after changing the direction of the pins before reading the idr.

3.3 Timer API

- Create a *tim.h* header file
  - Create a struct TIMx_t to access the control registers.
  - Include **#define**s for any necessary control bits or flags.

3.4 Hex game

Use your Keypad API to implement a hex decoding game with the following mechanics.

1. Start a Timer running.
2. Print your game title on the first line.
3. Prompt the user to “Hit any key” on the second line. (The Value of the Timer at this time seeds the random number generator.)
4. The game will be played in 5 rounds. Each round should consist of the following.
   i. Print "Round N" on the first line
   ii. Print 1.. and wait 1/2s on the second line
   iii. Print 2.. and wait 1/2s on the second line
   iv. Print 3.. and wait 1/2s on the second line
   v. Clear the screen and Print "GO!" in the middle of the top line.
   vi. Generate two random 4 bit patterns.
   vii. Place one pattern on LEDS[9..6] and the other on LEDS[3..0]
   viii. Mover the cursor to the middle of the second line and print "0x"
   ix. Start TIM2 Running
   x. Capture the next two keypresses for the hex equivilent of the binary patterns (*-E, #-F)
   xi. Stop TIM2 and use the value to calculate elapse tim.
   xii. Check the that first key matches the left pattern and the second key matches the right pattern.
      a. If they match print "Correct!" on the second line.
      b. If they don't match print the "Wrong! Ans:0x**??**" where the **??** are correct Hexadecimal values. (Wait 3 Seconds)
   xiii. Wait 2 Seconds
   xiv. Move to the next round
5. After 5 rounds print the following:
   - "**?**/5 Correct!" where **?** is the number of rounds correct on the first line.
   - The Average time (in ms) to calc for correct answers (incorrect responses don't count)
   - The Fastest Round (Correct) time (in ms)
6. Print "Rst 2 Play Again." on the second line and stall the program.

Hints:

- You may need multiple screens to show all of these results
- A const array can be used to store the char or value associated with each button.
- This approach is often faster than using a large switch statement.

4. Deliverables

- All code should include a header block with:
  - Your name
  - Course number and section
  - Assignment name
  - File name
  - List of any dependencies.
  - Description
- Print out you document code in the following order: (*italics implies optional*)
  - tim.h
  - *tim.c*
  - keypad.h
  - keypad.c
  - *game.h*
  - *game.c*
  - main.c
- Print in Light mode with Line numbers, filenames, and time and date using CubeIDE or Notepad++
- Staple together packet in the top left, in order, with rubric cover sheet.
- Phase one target – Working Game w/ keypad – (No timers) (Week8Lab)
- Phase two target - full application with timers added. (Week9Lab)
- Final packet and demonstration due at the beginning of Week9 Lab.