1. Introduction

The purpose of this lab is to write a simple program to explore the use of GPIO port configuration on an embedded system. We will introduce the control of memory mapped control registers accessed by pointer. We will look at enabling various GPIO ports for output to a led array and reading digital inputs from switches. The issue of contact bounce will be discussed, and we will look at an initial software driven debounce solution.

2. Prerequisites

- StmCubeIDE is installed on your machine.
- Real-term or another terminal program is installed on your machine.
- Your MSOE Development Board with the following components assembled:
    - STM32 Microcontroller
    - LED Array
    - Resistor Bars
    - Rotary Encoder
    - Analog Control Stick

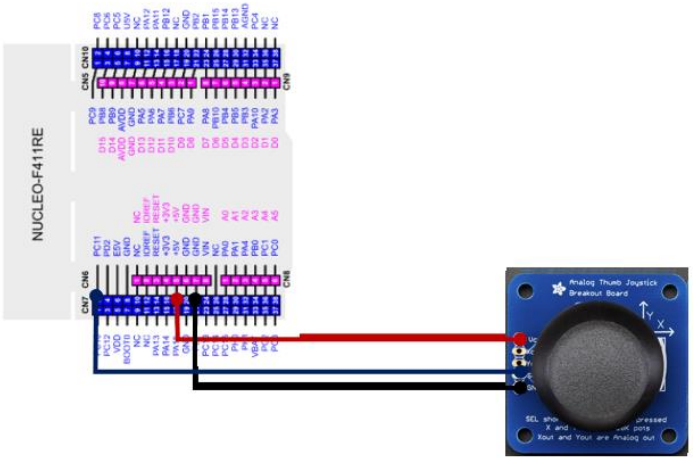
3. Activities

3.1 Create Lab2Match Project

- Copy the SerialConsoleTemplate Project within CubeIDE in the ProjectExplorer tab by copy-and-pasting.
- Name your project Lab2Match

3.2 Add starter APIs

- Add ledbar.h to the project
- Add pushbutton.h to the project
- Wire the Encoder button as shown below.



3.3 Create a LED API ()

- Complete the ledbar.h by finding the required information in the documentation
- **Do not modify the provided function definitions.**
- Feel free to add more for function call to the public API if you feel they would be helpful and not duplicate those already included.

- Write the ledbar.c to implement the functions in the ledbar.h file.
- Implement the API one method at a time.
- After writing each method construct temporarily add some code to a main driver to validate the function works correctly.
    - You do not need to save these or turn them in.
    - NEVER trust your code until you have seen it work.
    - I recommend the debugger here, but you can also use printf if you wish.

3.4 Create Switch API

- Complete the pushbutton.h by finding the required information in the documentation
- Write the pushbutton.c to implement the functions in pushbutton.h
- This file should include:
    - Any necessary defined constants
    - **Separate** function declarations to initialize the Encode, User, and Control buttons.
        - Encoder Button – PA12 (Requires a pullup resistor)
        - User Button – PC13
            - Has a built-in pullup resistor on the NUCLEO
        - Controller Button – PC11 (Requires a pullup resistor)
    - **Separate** function declarations to determine the status of the button.
        - Returning true if the button is pressed
        - Return false if it is not pressed.
- Note: All buttons are active low, producing a ‘0’ when pressed.
- You can assume only one button is press at a time.
- Limit/Eliminate global variables.
- Test as you go.

3.5 Write Pattern Match Application

- The goal is to make the pattern of LEDs on the LED Bar match the randomly generated pattern in as few steps as possible.
- Write a drive program that will use your API to implement a matching game with the following mechanics.
    - Prompt the player with the games name.
    - Ask the player to enter a random seed.
    - Generate a random 10-bit pattern and print it to the console in binary.
        - https://www.geeksforgeeks.org/generating-random-number-range-c/
        - Note: there is no way to print binary using printf and “%?”
        - You will have to write a file static helper to do this.

- Start with all lights on the LED Bar off.
- Prompt the user for their “move”:
  - User Button to toggle LED0
  - Control Button to rotate the ledbar left
  - Encoder Button to rotate the ledbar right
- If the patterns match
  - Print out a victory message including the number of moves it took.
  - Clear/”Sweep” the LED Bar
    - Turn on LED0, wait 1/8 sec
    - Turn on LED1, wait 1/8 sec
    - …
    - Until all lights are ON
    - Turn off LED9, wait 1/8 sec
    - Turn off LED8, wait 1/8 sec
    - …
    - Until all lights are Off
  - Get a new pattern and start over
- If the patterns don’t match
  - Prompt for the next move.

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:
  - main.c (driver/game mechanics)
  - ledbar.h
  - ledbar.c
  - pushbutton.h
  - pushbutton.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 – Win Animation to LED Bar (Week3 Lab)
- Phase two full application with switch control. (Week4 Lab)
- Final packet and demonstration due by 3PM, Friday Week4.