**LAB NOTE**

**Subject: Hardware/Software Interfacing**

**Lab 2: GPIO**

**Student: Minh Quan Tran**

**Sep 19th, 2024**

Table of Contents

[1. Objectives 4](#_Toc177634049)

[2. Problems and Solutions 5](#_Toc177634050)

[2.1 Problems 5](#_Toc177634051)

[2.2 Solutions 5](#_Toc177634052)

[3. Software Design 6](#_Toc177634053)

[3.1 List of function 6](#_Toc177634054)

[3.2 While loop 7](#_Toc177634055)

[4. Result 7](#_Toc177634056)

**TABLE OF FIGURES**

[Figure 4‑1: Lab2's result 7](#_Toc177634150)

# Objectives

* Code’s requirement:
  + Uses 3 GPIO for inputs and makes use of Interrupt Handler to detect state changes
  + Uses 3 GPIO for outputs and displays the state in an interesting way (not just on/off or 1/0)
  + Obtains input from the user to change the state of the GPIO outputs but does not wait forever for user input
  + Displays the current state of the GPIO pins when not interrupted to change GPIO  
    outputs

# Problems and Solutions

## Problems

* No problem.

## Solutions

* No problem.

# Software Design

## List of function

This call back is to start the timer for the deboune time for all GPIO Input

**void** **HAL\_GPIO\_EXTI\_Callback**(uint16\_t GPIO\_Pin)

{

btnPressed = GPIO\_Pin;

// Start a timer to handle debounce

HAL\_TIM\_Base\_Start\_IT(&htim2);

}

When Timer callback if button still pressed then it will change the btnState or else it will be counted as bounce and not updated the btnState.

**void** **HAL\_TIM\_PeriodElapsedCallback**(TIM\_HandleTypeDef \*htim)

{

**if** (htim->Instance == TIM2)

{

// Stop the timer

HAL\_TIM\_Base\_Stop\_IT(&htim2);

// Check the button state after 20ms

**if** ((btnPressed == *LEFT\_PIN*) && (HAL\_GPIO\_ReadPin(GPIOC,*LEFT\_PIN*) == *GPIO\_PIN\_RESET*))

{

btnState = *LEFT*;

}

**else** **if** ((btnPressed == *RIGHT\_PIN*) && (HAL\_GPIO\_ReadPin(GPIOC,*RIGHT\_PIN*) == *GPIO\_PIN\_RESET*))

{

btnState = *RIGHT*;

}

**else** **if** ((btnPressed == *TOGGLE\_PIN*) && (HAL\_GPIO\_ReadPin(GPIOC,*TOGGLE\_PIN*) == *GPIO\_PIN\_RESET*))

{

btnState = *TOGGLE*;

}

}

}

setLed is used to set Led base on the parameter and used that function to create a sequence of how the LED will blink like shift left LED or shift right LED.

**void** **setLed**(bool led1Status,bool led2Status, bool led3Status)

{

HAL\_GPIO\_WritePin(GPIOA,GPIO\_PIN\_5,led1Status);

HAL\_GPIO\_WritePin(GPIOA,GPIO\_PIN\_6,led2Status);

HAL\_GPIO\_WritePin(GPIOA,GPIO\_PIN\_7,led3Status);

}

**void** **shiftLeftLed**(**void**)

{

setLed(0,0,1);

HAL\_Delay(100);

setLed(0,1,0);

HAL\_Delay(100);

setLed(1,0,0);

HAL\_Delay(100);

setLed(0,0,0);

HAL\_Delay(100);

}

**void** **shiftRightLed**(**void**)

{

setLed(1,0,0);

HAL\_Delay(100);

setLed(0,1,0);

HAL\_Delay(100);

setLed(0,0,1);

HAL\_Delay(100);

setLed(0,0,0);

HAL\_Delay(100);

}

## While loop

**while** (1)

{

**switch**(controlState)

{

**case** *BTN*:

**switch**(btnState)

{

**case** *LEFT*:

shiftLeftLed();

**break**;

**case** *RIGHT*:

shiftRightLed();

**break**;

**case** *TOGGLE*:

controlState = *PROMPT*;

btnState = *NONE*;

**break**;

**default**:

**break**;

}

**break**;

**case** *PROMPT*:

// Toggle a button again to enter how the LED lit

**if** (btnState == *TOGGLE*)

{

**printf**("Insert L/R to control LED\r\n");

**switch**(getCharFromUart2())

{

**case** 'L':

rcvChar = 'L';

**printf**("Shift Left\r\n");

**break**;

**case** 'R':

rcvChar = 'R';

**printf**("Shift Right\r\n");

**break**;

**case** UART\_TIMEOUT\_ERROR:

**printf**("Prompt Timeout\r\n");

**printf**("Switching to BTN control state\r\n");

controlState = *BTN*;

rcvChar = ' ';

**default**:

**break**;

}

btnState = *NONE*;

}

**switch**(rcvChar)

{

**case** 'L':

shiftLeftLed();

**break**;

**case** 'R':

shiftRightLed();

**break**;

**default**:

**break**;

}

}

# Result

A black screen with white text

Description automatically generated

Figure 4‑1: Lab2's result

**REFERENCES**