[LAB#2] GPIO Digital InOut

Date: Sep.26.2023

Author: 21900416 Gyeonheal An

Github: https://github.com/AnGyeonheal/Embedded_Control

Demo Video: https://www.youtube.com/shorts/HLNItlFfu7U

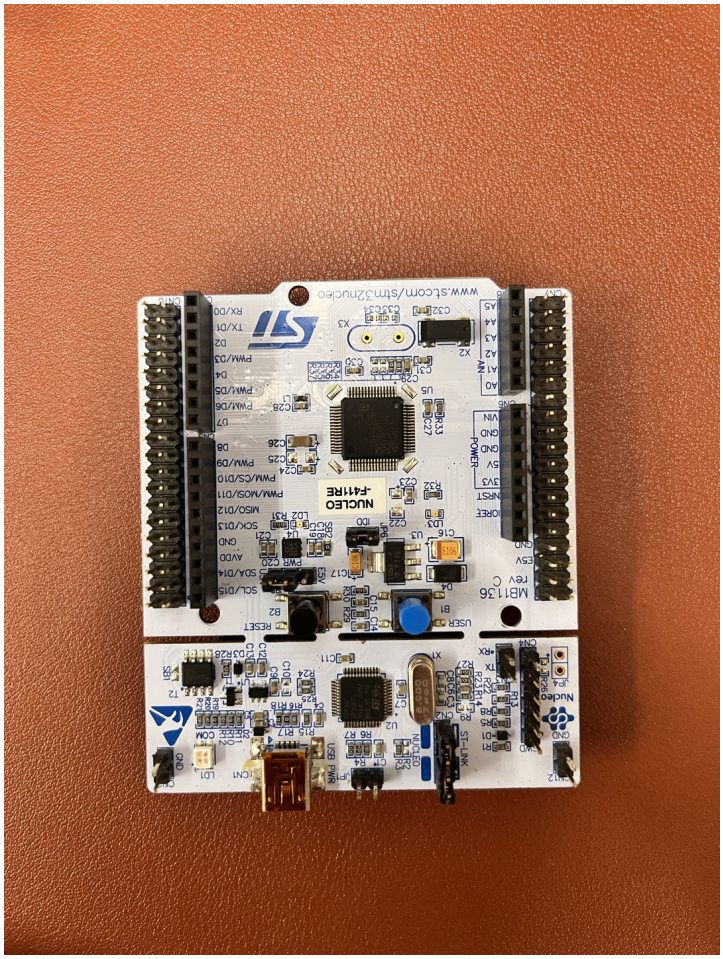
I. Introduction

In this lab, we are required to create a simple program that toggle multiple LEDs with a push-button input. Create HAL drivers for GPIO digital in and out control and use your library.

II. Requirement

i. Hardware

- **MCU**



  - NUCLEO-F411RE
- **Actuator/Sensor/Others**
  - LEDs x 4
  - Resistor 330 ohm x 3, breadboard

ii. Software

- Keil uVision, CMSIS, EC_HAL library

III. Problem 1: Create EC_HAL library

Procedure

Create the library directory `\repos\EC\lib\`.

Create your own library for Digital_In and Out : `ecGPIO.h, ecGPIO.c`

ecRCC.h

```
/*----------------------------------------------------------------\
@ Embedded Controller by Young-Keun Kim - Handong Global University
Author           : Gyeonheal An
Created          : 05-03-2021
Modified         : 09-25-2023
Language/ver     : C++ in Keil uVision

Description      : Distributed to Students for LAB_GPIO
/----------------------------------------------------------------*/

#ifndef __EC_RCC_H
#define __EC_RCC_H

#ifdef __cplusplus
 extern "C" {
#endif /* __cplusplus */

//#include "stm32f411xe.h"

void RCC_HSI_init(void);
void RCC_PLL_init(void);
void RCC_GPIOA_enable(void);
void RCC_GPIOB_enable(void);
void RCC_GPIOC_enable(void);
void RCC_GPIOD_enable(void);

// void RCC_GPIO_enable(GPIO_TypeDef * GPIOx);

extern int EC_SYSCL;

#ifdef __cplusplus
}
#endif /* __cplusplus */

#endif
```

ecGPIO.h

```
/*----------------------------------------------------------------\
@ Embedded Controller by Young-Keun Kim - Handong Global University
Author           : Gyeonheal An
Created          : 05-03-2021
Modified         : 09-25-2023
```

```
Language/ver     : C++ in Keil uVision

Description      : Distributed to Students for LAB_GPIO
/----------------------------------------------------------------*/

#include "stm32f411xe.h"
#include "ecRCC.h"

#ifndef __ECGPIO_H
#define __ECGPIO_H

#define INPUT  0x00
#define OUTPUT 0x01
#define AF     0x02
#define ANALOG 0x03

#define HIGH 1
#define LOW  0

#define EC_NONE 0
#define EC_PU 1
#define EC_PD 2

#define EC_PUSH_PULL 0
#define EC_OPEN_DRAIN 1

#define EC_LOW 0
#define EC_MEDIUM 1
#define EC_FAST 2
#define EC_HIGH 3

#define LED_PIN     5
#define BUTTON_PIN 13

#ifdef __cplusplus
 extern "C" {
#endif /* __cplusplus */

void GPIO_init(GPIO_TypeDef *Port, int pin, unsigned int mode);
void GPIO_write(GPIO_TypeDef *Port, int pin, unsigned int Output);
int  GPIO_read(GPIO_TypeDef *Port, int pin);
void GPIO_mode(GPIO_TypeDef* Port, int pin, unsigned int mode);
void GPIO_ospeed(GPIO_TypeDef* Port, int pin, unsigned int speed);
void GPIO_otype(GPIO_TypeDef* Port, int pin, unsigned int type);
void GPIO_pupd(GPIO_TypeDef *Port, int pin, unsigned int pupd);

#ifdef __cplusplus
}
#endif /* __cplusplus */

#endif
```

- ecGPIO.c code is uploaded on Github

IV. Problem 2: Toggle LED with Button

i. Procedure

1. Create a new project under the directory `\repos\EC\LAB\`
    - Name the source file as "**LAB_GPIO_DIO_LED.c**"

2. Include your library **ecGPIO.h, ecGPIO.c** in `\repos\EC\lib\`.

3. Toggle the LED by pushing the button.
    - Push button (LED ON), Push Button (LED OFF) and repeat

ii. Configuration

Button (B1)	LED
Digital In	Digital Out
GPIOC, Pin 13	GPIOA, Pin 5
PULL-UP	Open-Drain, Pull-up, Medium Speed

iii. Code

```
/*----------------------------------------------------------------\
@ Embedded Controller by Young-Keun Kim - Handong Global University
Author           : Gyeonheal An
Created          : 05-03-2021
Modified         : 09-25-2023
Language/ver     : C++ in Keil uVision

Description      : Distributed to Students for LAB_GPIO
/----------------------------------------------------------------*/

#define LED_PIN     5
#define BUTTON_PIN 13
#include "stm32f4xx.h"
#include "ecRCC.h"
#include "ecGPIO.h"

void setup(void);

int main(void) {
    unsigned int OUT = 1;
    unsigned int count = 0;
    // Initialiization
    setup();

    // Inifinite Loop
    while(1){
       GPIO_write(GPIOA, LED_PIN, OUT);
```

```
        if(!GPIO_read(GPIOC, BUTTON_PIN) && (count > 50000)){
           OUT ^= 1UL;
           count = 0;
        }

        count++;
    }
}

// Initialiization
void setup(void)
{
    RCC_HSI_init();
    GPIO_init(GPIOC, BUTTON_PIN, INPUT);  // calls RCC_GPIOC_enable()
    GPIO_init(GPIOA, LED_PIN, OUTPUT);    // calls RCC_GPIOA_enable()
    GPIO_otype(GPIOA, LED_PIN, EC_OPEN_DRAIN);
    GPIO_ospeed(GPIOA, LED_PIN, EC_MEDIUM);
    GPIO_pupd(GPIOA, LED_PIN, EC_PU);
    GPIO_pupd(GPIOC, BUTTON_PIN, EC_PU);
}
```

Variable Description

- OUT: This is a variable related to the output of the LED. This is toggled within the while() function so that the LED can be turned on and off.
- count: This is used to solve the delay problem that occurs on IDE. This increases within the while function and resets to 0 when the button is pressed.

iv. Discussion

- **Find out a typical solution for software debouncing and hardware debouncing.**

  The schmitt trigger was used to solve debounging problems in hardware. This solves the debounceing problem by buffering the way the current flows only when the threshold voltage is exceeded.

  In the software, delay was created to solve debounceing problems. There is a section in the loop where the computer calculates and does not input, and at this time, a condition was given to solve it.

- **What method of debouncing did this NUCLEO board use for the push-button(B1)?**

  Schmitt trigger method was used .

V. Problem 3: Toggle LED with Button

i. Procedure

1. Create a new project under the directory `\repos\EC\LAB\`
   - Name the source file as "**LAB_GPIO_DIO_multiLED.c"**

2. Include your library **ecGPIO.h, ecGPIO.c** in `\repos\lib\`.

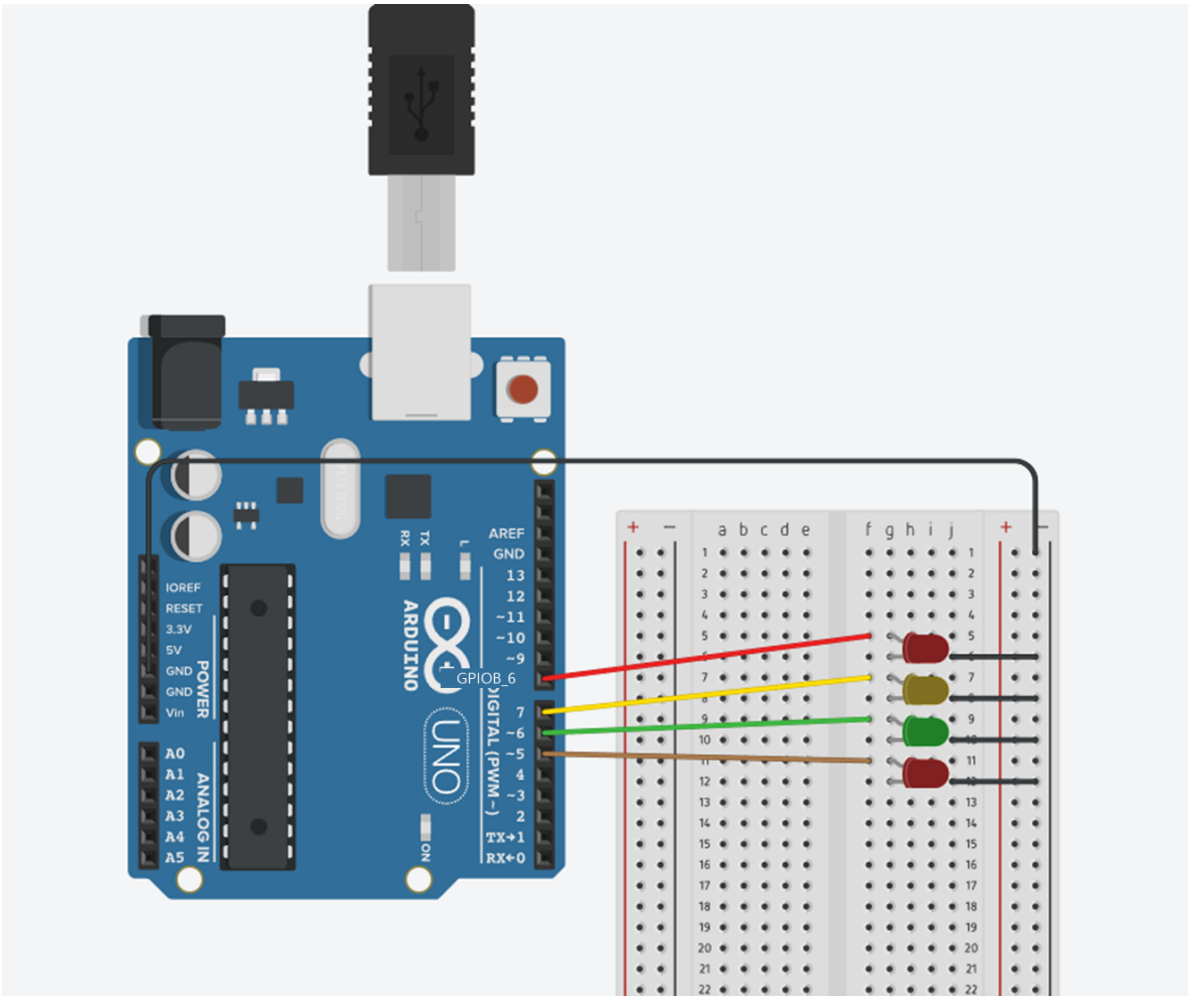
3. Connect 4 LEDs externally with necessary load resistors.

- As Button B1 is Pressed, light one LED at a time, in sequence.
- Example: LED0--> LED1-->...LED3-->...LED0....

ii. Configuration

Button	LED
Digital In	Digital Out
GPIOC, Pin 13	PA5, PA6, PA7, PB6
PULL-UP	Push-Pull, Pull-up, Medium Speed

iii. Circuit Diagram



iv. Code

```
/*-----------------------------------------------------------------\
@ Embedded Controller by Young-Keun Kim - Handong Global University
Author           : Gyeonheal An
Created          : 05-03-2021
Modified         : 09-25-2023
Language/ver     : C++ in Keil uVision

Description      : Distributed to Students for LAB_GPIO
/-----------------------------------------------------------------*/
```

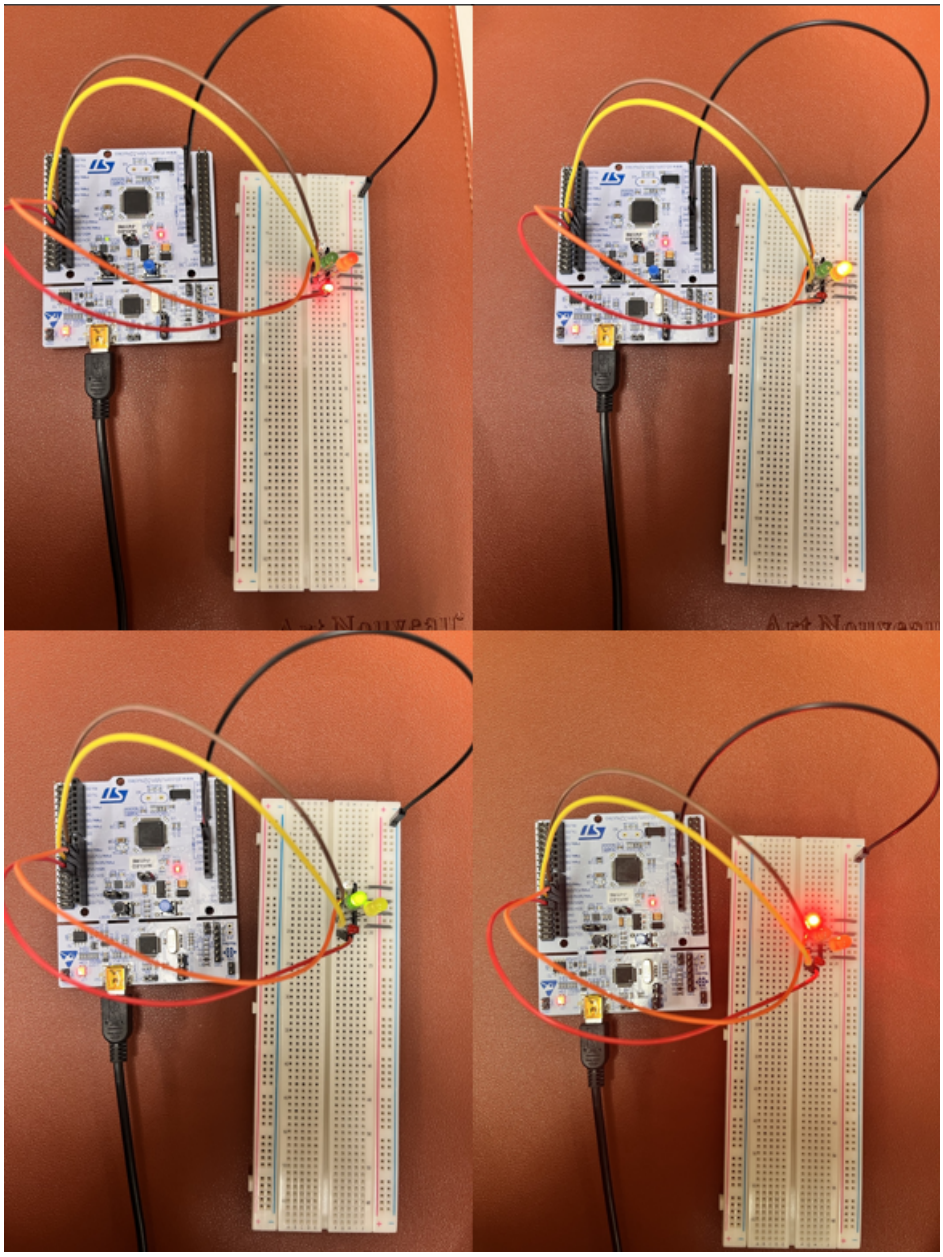
```c
#include "stm32f4xx.h"
#include "ecRCC.h"
#include "ecGPIO.h"

#define LED_PIN     5
#define BUTTON_PIN 13

void setup(void);

int main(void) {

    int delay;
    int cnt=4;

    // Initialiization --------------------------------------------------------
    setup();

    // Inifinite Loop ---------------------------------------------------------
    while(1){

        if((GPIO_read(GPIOC, BUTTON_PIN)==0) && (delay > 50000) && (cnt % 4 ==
0)){
            GPIO_write(GPIOA, LED_PIN, HIGH);
            GPIO_write(GPIOA, 6, LOW);
            GPIO_write(GPIOA, 7, LOW);
            GPIO_write(GPIOB, 6, LOW);
            delay = 0;
            cnt++;
        }

        else if((GPIO_read(GPIOC, BUTTON_PIN)==0) && (delay > 50000) && (cnt % 4
== 1)){
            GPIO_write(GPIOA, 6, HIGH);
            GPIO_write(GPIOA, LED_PIN, LOW);
            GPIO_write(GPIOA, 7, LOW);
            GPIO_write(GPIOB, 6, LOW);
            delay = 0;
            cnt++;
        }

        else if((GPIO_read(GPIOC, BUTTON_PIN)==0) && (delay > 50000) && (cnt % 4
== 2)){
            GPIO_write(GPIOA, 7, HIGH);
            GPIO_write(GPIOA, LED_PIN, LOW);
            GPIO_write(GPIOA, 6, LOW);
            GPIO_write(GPIOB, 6, LOW);
            delay = 0;
            cnt++;
        }

        else if((GPIO_read(GPIOC, BUTTON_PIN)==0) && (delay > 50000) && (cnt %4
== 3)){
            GPIO_write(GPIOB, 6, HIGH);
            GPIO_write(GPIOA, LED_PIN, LOW);
            GPIO_write(GPIOA, 6, LOW);
            GPIO_write(GPIOA, 7, LOW);
            delay = 0;
            cnt++;
```

```
        }
        delay ++ ;
    }
}

// Initialiization
void setup(void)
{
    RCC_HSI_init();
    GPIO_init(GPIOC, BUTTON_PIN, INPUT);  // calls RCC_GPIOC_enable()
    GPIO_init(GPIOA, LED_PIN, OUTPUT);    // calls RCC_GPIOA_enable()
    GPIO_init(GPIOA, 6, OUTPUT);    // calls RCC_GPIOA_enable()
    GPIO_init(GPIOA, 7, OUTPUT);    // calls RCC_GPIOA_enable()
    GPIO_init(GPIOB, 6, OUTPUT);    // calls RCC_GPIOB_enable()

    GPIO_pupd(GPIOC, BUTTON_PIN, EC_PU);
    GPIO_ospeed(GPIOA, LED_PIN, EC_MEDIUM);
    GPIO_otype(GPIOA, LED_PIN, EC_PUSH_PULL);
    GPIO_pupd(GPIOA, LED_PIN, EC_PU);

    GPIO_ospeed(GPIOA, 6, EC_MEDIUM);
    GPIO_otype(GPIOA, 6, EC_PUSH_PULL);
    GPIO_pupd(GPIOA, 6, EC_PU);

    GPIO_ospeed(GPIOA, 7, EC_MEDIUM);
    GPIO_otype(GPIOA, 7, EC_PUSH_PULL);
    GPIO_pupd(GPIOA, 7, EC_PU);

    GPIO_ospeed(GPIOB, 6, EC_MEDIUM);
    GPIO_otype(GPIOB, 6, EC_PUSH_PULL);
    GPIO_pupd(GPIOB, 6, EC_PU);
}
```

Variable Description

- delay: This is a variable related to the output of the LED. It gives delay in while() function.
- cnt: This is a variable that allows LEDs to light up in order. When this variable is divided by 4, the rest is set from 0 to 3, so that the next order of lights can be turned on.

v. Results



vi. Discussion

Find out a typical solution for software debouncing and hardware debouncing. What method of debouncing did this NUCLEO board use for the push-button(B1)?

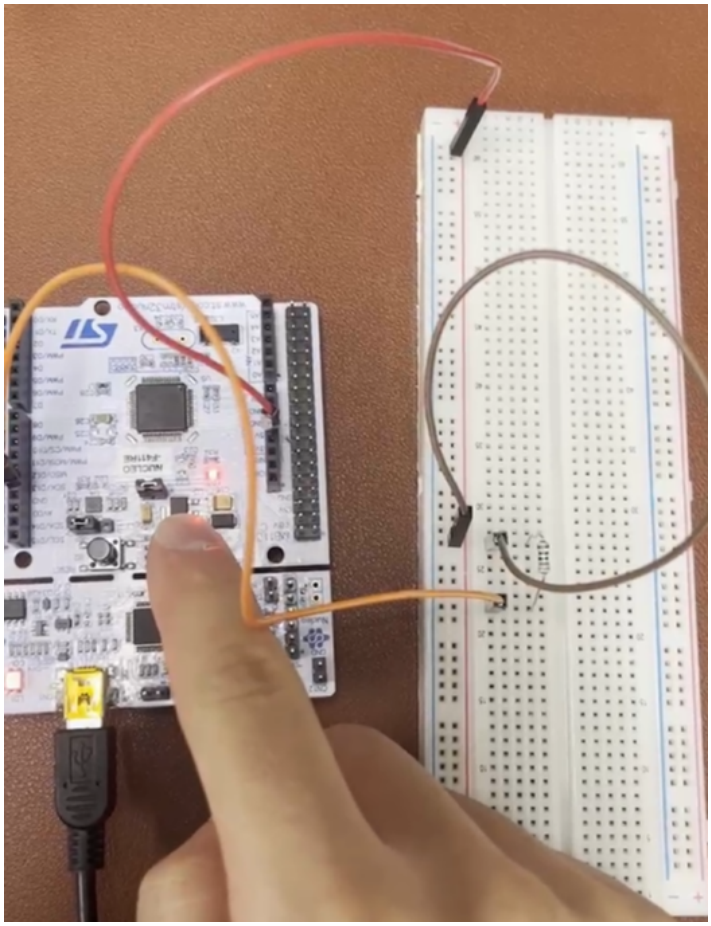
The software solved the bouncing problem by creating delay as in Problem 2. Nucleobord uses the schmitt trigger method and RC circuit to solve the bouncing problem in hardware. The Schmitt Trigger method reduces noise by creating a buffer using the voltage comparator. The RC circuit creates a low pass filter to block high-frequency noise.

VI. Reference

https://ykkim.gitbook.io/ec/ec-course/lab/lab-gpio-digital-inout

Zhu, Yifeng. *Embedded Systems with ARM Cortex-M Microcontrollers in Assembly Language and C*. Pearson, [2015].

VII. Troubleshooting



In the process of setting up GPIO_otype in problem 2, the LED light was very low when set to open-drain (1) than when set to push-pull (0).
Open drain is a circuit technology that enables two-way communication. When the transistor is turned on by attaching the button to the gate position using the MOSFET, the drain and source are connected to the ground. Therefore, when a High signal enters the input, a Low signal is output. Conversely, when a low signal is input, a high signal is output. At this time, the output becomes an open circuit and becomes a high impedance state with high resistance. Therefore, we added a pull-up resistor (3.3 kohm) to the outside to allow the LED to light up more brightly.