LAB: GPIO Digital InOut

Embedded Controller Lab Report

LAB: GPIO Digital InOut

Date: 2023.09.24

Author: Hanmin Kim

Github:https://github.com/ansterdaz/HKim/tree/main

Demo video:https://youtube.com/shorts/-IGCFM2Ks4g?si=UJdjGTAwtymDmU3f

Introduction

In this lab, a program will be written to output an LED through button input. Additionally, a HAL driver will be created and the library for GPIO digital input and output control will be utilized.

Requirement

Hardware

MCU

- NUCLEO-F411RE

Actuator/Sensor/Others:

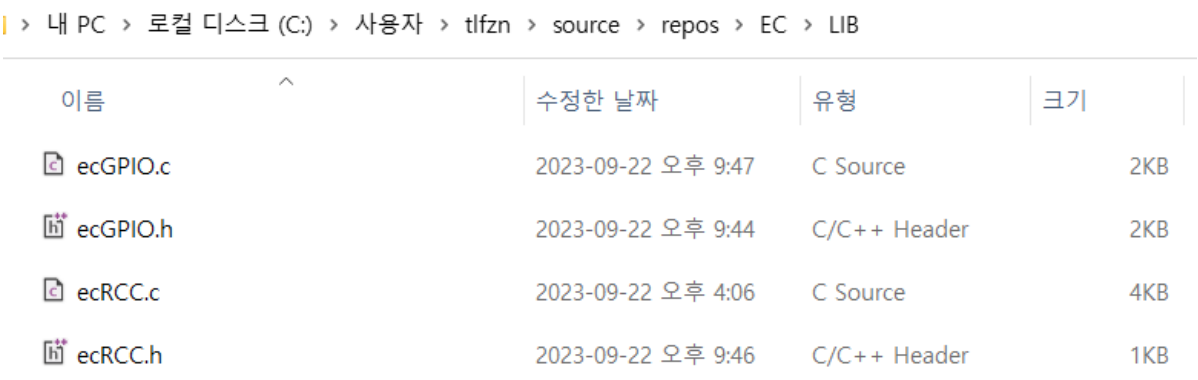
- LEDs x 4
- Resistor 330 ohm x 4, breadboard

Software

Keil uVision, CMSIS, EC_HAL library

Problem 1: Create EC_HAL library

Procedure



ecRCC.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_GPIOE_enable(void);
```

ecGPIO.h

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

Example code

ecGPIO.c

```
// GPIO Mode          : Input(00), Output(01), AlterFunc(10), Analog(11)
void GPIO_mode(GPIO_TypeDef *Port, int pin, int mode){
   Port->MODER &= ~(3<<(2*pin));
   Port->MODER |= mode<<(2*pin);
}


// GPIO Speed          : Low speed (00), Medium speed (01), Fast speed (10), High speed (11)
void GPIO_ospeed(GPIO_TypeDef *Port, int pin, int speed){
   Port->OSPEEDR &= ~(3<<(2*pin));
   Port->OSPEEDR |=  speed <<(2*pin);
}

// GPIO Output Type: Output push-pull (0, reset), Output open drain (1)
void GPIO_otype(GPIO_TypeDef *Port, int pin, int type){
    Port->OTYPER &= ~(1<<pin);
    Port->OTYPER &= ~(type<<pin);
}

// GPIO Push-Pull    : No pull-up, pull-down (00), Pull-up (01), Pull-down (10), Reserved (11)
void GPIO_pupd(GPIO_TypeDef *Port, int pin, int pupd){
    Port-> PUPDR &= ~(3<<(2*pin));
    Port-> PUPDR |= pupd<<(2*pin);
}

// GPIO Push-Pull    : No pull-up, pull-down (00), Pull-up (01), Pull-down (10), Reserved (11)
void GPIO_pupd(GPIO_TypeDef *Port, int pin, int pupd){
    Port-> PUPDR &= ~(3<<(2*pin));
    Port-> PUPDR |= pupd<<(2*pin);
}

int GPIO_read(GPIO_TypeDef *Port, int pin){
   unsigned int btVal= (Port->IDR >> pin) &1;

  return btVal;
  //[TO-DO] YOUR CODE GOES HERE
    //[TO-DO] YOUR CODE GOES HERE
}

void GPIO_write(GPIO_TypeDef *Port, int pin, int Output){
    Port-> ODR &= ~(1<<(pin));
    Port-> ODR |= Output<<pin;
}
```

GPIO_read and GPIO_write were additionally created.

problem 2: Toggle LED with Button

Procedure

1. Create a new project under the directory `\repos\EC\LAB\`

   The project name is "**LAB_GPIO_DIO_LED".**

   Name the source file as "**LAB_GPIO_DIO_LED.c"**

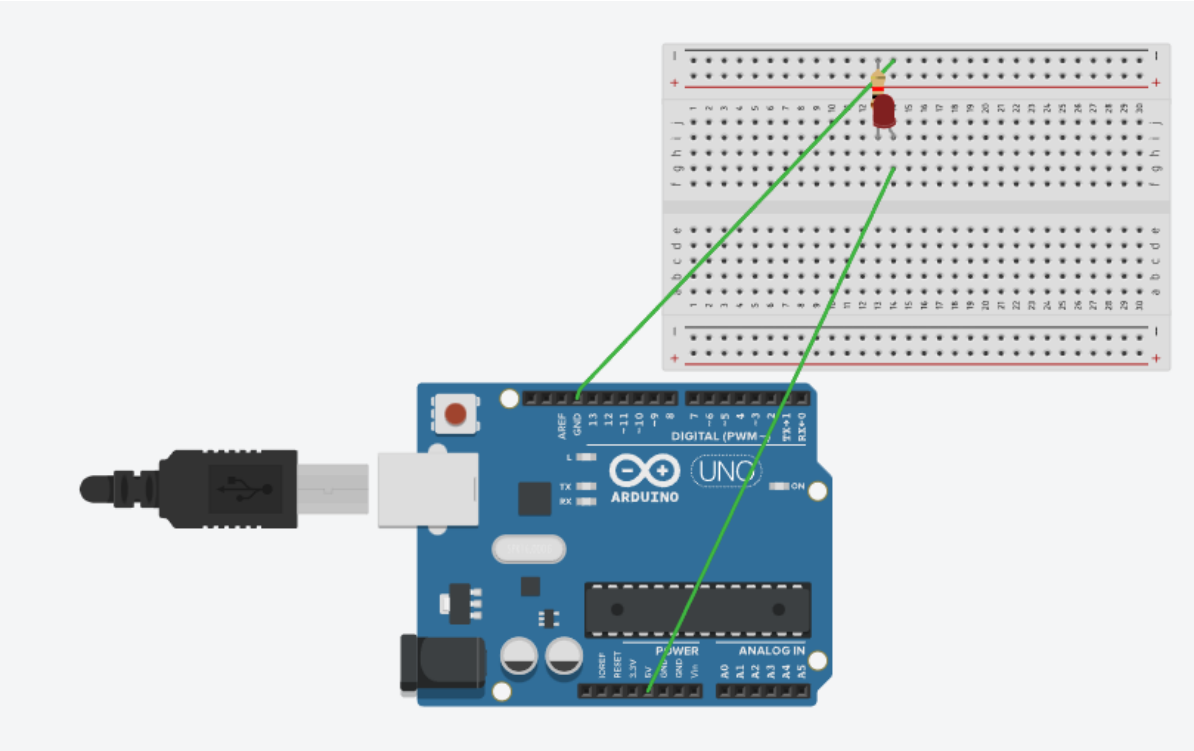
2. Include your library **ecGPIO.h, ecGPIO.c** in `\repos\EC\lib\`

3. Toggle the LED by pushing the button.

Configuration

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

Circuit Diagram



Algorithm

Input : Button

Output: LED

S0: LED Off

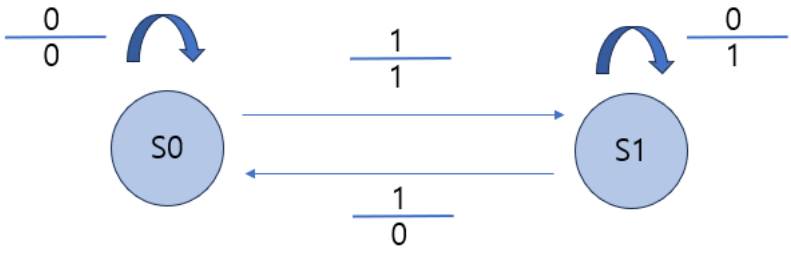
S1: LED On

State table

Present State	Next State		Output	
	Btn=0	Btn= 1	Btn=0	Btn=1
S0	S0	S1	LED=LOW	LED=HIGH
S1	S1	S0	LED=HIGH	LED=LOW

State Diagram

Button
LED



Code

```
#define LED_PIN   5
#define BUTTON_PIN 13

int buttonState = 0;
int ledState = 0;

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

void setup(void);
```

buttonState and ledstate were defined.

```
int main(void) {
  // Initialiization
  setup();

  // Inifinite Loop
  while(1){

  int currentButtonState = GPIO_read(GPIOC, BUTTON_PIN);

        if (currentButtonState == 0 && buttonState == 1)
        {
           ledState = !ledState;

            if (ledState == 1)
              {
               GPIO_write(GPIOA, LED_PIN, HIGH);
              }
            else
              {
               GPIO_write(GPIOA, LED_PIN, LOW);
              }
        }

        buttonState = currentButtonState;
    }

    return 0;

  }
```

currentButtonState is set to button pin input.

```
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_pupd(GPIOA, LED_PIN, EC_PU);     // Pull up
  GPIO_otype(GPIOA, LED_PIN, EC_OPEN_DRAIN);        // Open- Drain
  GPIO_ospeed(GPIOA, LED_PIN, EC_MEDIUM);           // medium speed
}
```

The input and output were set to pull up, open drain, and medium speed.

Result

1. The LED stays on when the button is pressed.
2. When you press the button again, the LED turns off.

Demo Video: https://youtube.com/shorts/-IGCFM2Ks4g?si=UJdjGTAwtymDmU3f

Discussion and Analysis

The ledstate is designed to be inverted every time the button is pressed.

1. Find out a typical solution for software debouncing and hardware debouncing.

There is a method of software debouncing that is designed using FSM and defines the state so that the state is switched.

Hardware debouncing methods include using flip-flops or latches to store the button state and update it only when a signal is detected.

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

FSM, one of the software debouncing methods, was designed and a method of converting States was used.

Problem 3: Toggle LED with Button

Procedure

1. Create a new project under the directory `\repos\EC\LAB\`

- The project name is "**LAB_GPIO_DIO_multiLED".**
- Name the source file as "**LAB_GPIO_DIO_multiLED.c"**

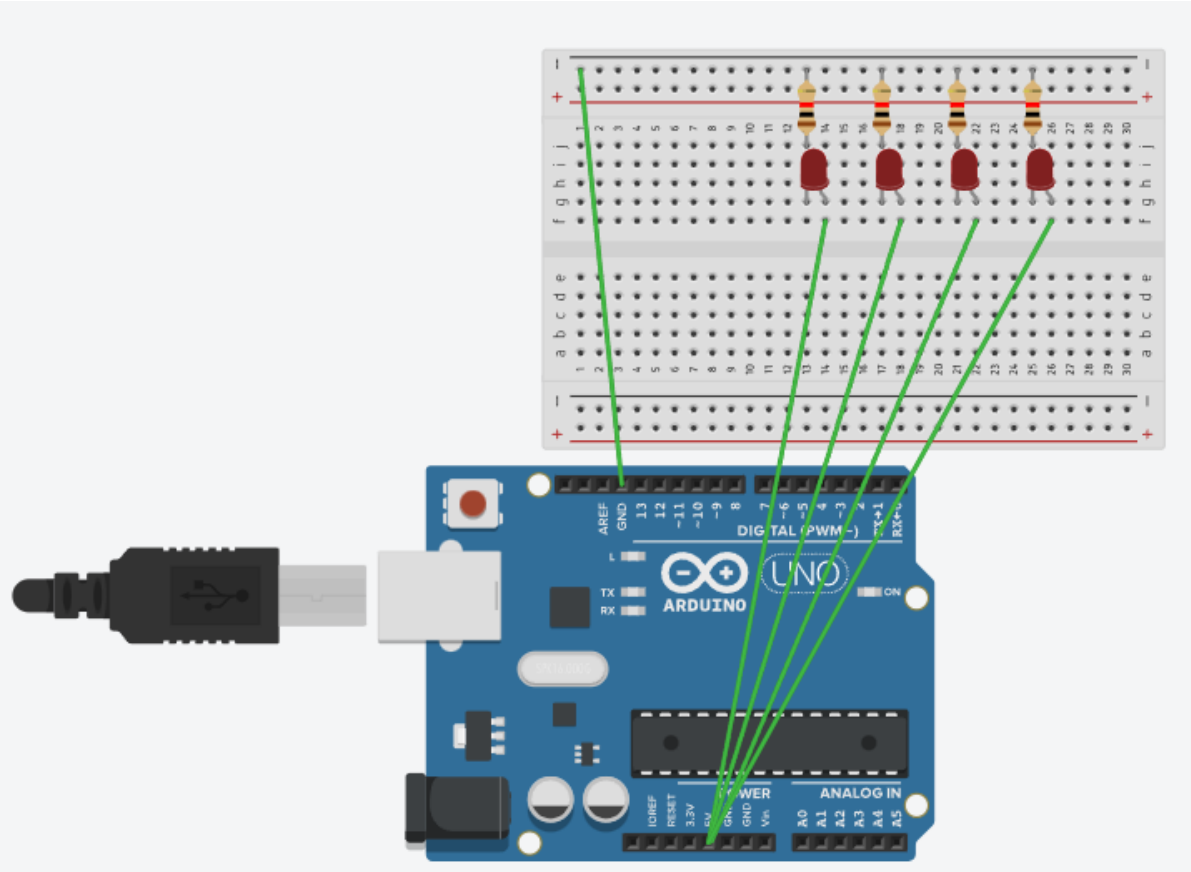
2. Include your library **ecGPIO.h, ecGPIO.c** in `\repos\lib\`.
3. Connect 4 LEDs externally with necessary load resistors.

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

Configuration

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

Circuit Diagram



Algorithm

Input: Button

Output: LED1, LED2, LED3, LED4

S0: All LEDs Off

S1: Only LED1 On

S2: Only LED2 On

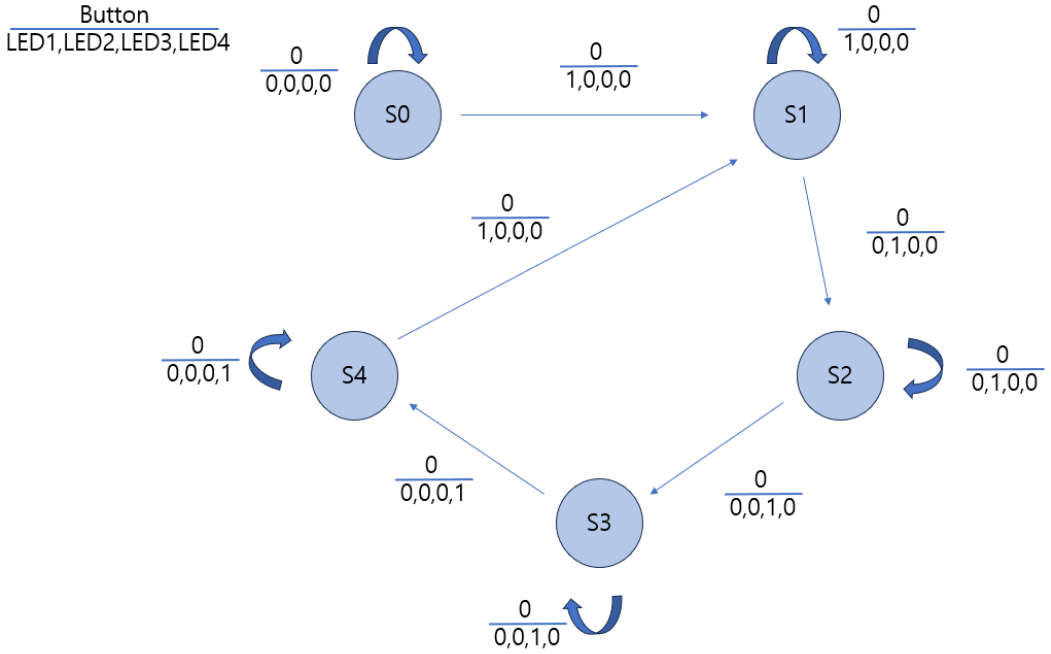
S3: Only LED3 On

S4: Only LED4 On

State table

Present State	Next State		Output	
	Btn=0	Btn= 1	Btn=0	Btn=1
S0	S0	S1	LED1=0, LED2=0 LED3=0, LED4=0	LED1=1, LED2=0 LED3=0, LED4=0
S1	S1	S2	LED1=1, LED2=0 LED3=0, LED4=0	LED1=0, LED2=1 LED3=0, LED4=0
S2	S2	S3	LED1=0, LED2=1 LED3=0, LED4=0	LED1=0, LED2=0 LED3=1, LED4=0
S3	S3	S4	LED1=0, LED2=0 LED3=1, LED4=0	LED1=0, LED2=0 LED3=0, LED4=1
S4	S4	S1	LED1=0, LED2=0 LED3=0, LED4=1	LED1=1, LED2=0 LED3=0, LED4=0

State Diagram



Code

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

#define LED_PIN   5
#define BUTTON_PIN 13

int buttonState = 0;
int ledState = 0;


void setup(void);
```

buttonState and ledState were defined.

```
int main(void) {
  setup();

  while(1){

  int currentButtonState = GPIO_read(GPIOC,BUTTON_PIN);

  if(currentButtonState == 0 && buttonState ==1)
  {
    ledState += 1;

    if(ledState == 1)           //  first LED on
    {
     GPIO_write(GPIOA,LED_PIN,HIGH);
     GPIO_write(GPIOA,6,LOW);
     GPIO_write(GPIOA,7,LOW);
     GPIO_write(GPIOB,6,LOW);
    }
    else if(ledState ==2)      // second LED on
    {
     GPIO_write(GPIOA,LED_PIN,LOW);
     GPIO_write(GPIOA,6,HIGH);
     GPIO_write(GPIOA,7,LOW);
     GPIO_write(GPIOB,6,LOW);
    }
    else if(ledState ==3)      // third LED on
    {
     GPIO_write(GPIOA,LED_PIN,LOW);
     GPIO_write(GPIOA,6,LOW);
     GPIO_write(GPIOA,7,HIGH);
     GPIO_write(GPIOB,6,LOW);
    }
    else if(ledState==4)      // Forth LED on
    {
     GPIO_write(GPIOA,LED_PIN,LOW);
     GPIO_write(GPIOA,6,LOW);
     GPIO_write(GPIOA,7,LOW);
     GPIO_write(GPIOB,6,HIGH);
     ledState = 0;
    }
  }
    buttonState = currentButtonState;
}
}
```

When controlling each LED on and off with an if statement, one LED is set to turn on at a time with each button.

```
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(GPIOA, LED_PIN, EC_PU);                // Pull up
  GPIO_otype(GPIOA, LED_PIN, EC_PUSH_PULL);        // push pull
  GPIO_ospeed(GPIOA, LED_PIN, EC_MEDIUM);          // medium speed

  GPIO_pupd(GPIOA, 6, EC_PU);                      // Pull up
  GPIO_otype(GPIOA, 6, EC_PUSH_PULL);              // push pull
  GPIO_ospeed(GPIOA, 6, EC_MEDIUM);                // medium speed

  GPIO_pupd(GPIOA, 7, EC_PU);                      // Pull up
  GPIO_otype(GPIOA, 7, EC_PUSH_PULL);              // push pull
  GPIO_ospeed(GPIOA, 7, EC_MEDIUM);                // medium speed

  GPIO_pupd(GPIOB, 6, EC_PU);                      // Pull up
  GPIO_otype(GPIOB, 6, EC_PUSH_PULL);              // push pull
  GPIO_ospeed(GPIOB, 6, EC_MEDIUM);                // medium speed


}
```

Pull up, push pull, and medium speed were set for each port.

Result

1. The first LED turns on when the button is pressed.
2. The second LED turns on when the button is pressed.
3. The third LED turns on when the button is pressed.
4. The fourth LED turns on when the button is pressed.

Demo Video: https://youtube.com/shorts/-IGCFM2Ks4g?si=UJdjGTAwtymDmU3fReference

Discussion and Analysis

Each LED was controlled by defining the button state and ledstate and writing an if statement in which the states were added one by one.

1. 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)?

There is a method of software debouncing that is designed using FSM and defines the state so that the state is switched. Hardware debouncing methods include using flip- flops or latches to store the button state and update it only when a signal is detected.

Here, FSM, one of the software debouncing methods, was designed and a method of converting Statef was used.

Reference

https://github.com/ykkimhgu

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