# LAB: GPIO Digital InOut(eval board)

## LAB: GPIO Digital InOut

**Date:** 2025-09-16

**Author/Partner:** 22201042-Dongjun,Lee

**Github:** https://github.com/henny041520-commits/EC-DJLee-042/tree/main

**Demo Video:**

LAB_GPIO_DIO_LED_Photosensor_22201042 :

https://youtube.com/shorts/xrf6yfmJF2U?feature=share

LAB_GPIO_DIO_LED_Button_22201042 :

https://youtube.com/shorts/p9E1w-iWDRE?feature=share

LAB_GPIO_DIO_multiLED :

https://youtube.com/shorts/MM9G_ndksnY?feature=share

## Introduction

In this lab, you 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.

## Requirement

Write a list of HW/SW requirement

### Hardware

- MCU
  - NUCLEO-F411RE
  - Eval Board
- Sensor
  - Photodetector
- Actuator/Display
  - LED

## Software

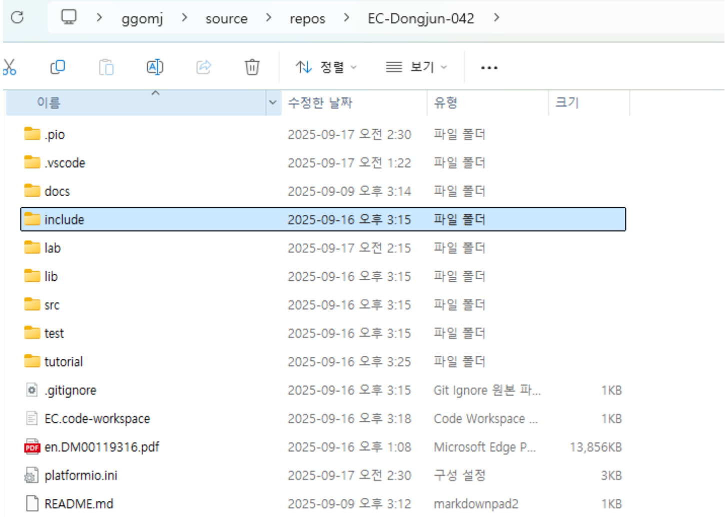
- -PlatformIO, CMSIS, EC_HAL library

# Problem 1 :  Create EC_HAL library

## Procedure

### Library Header Files

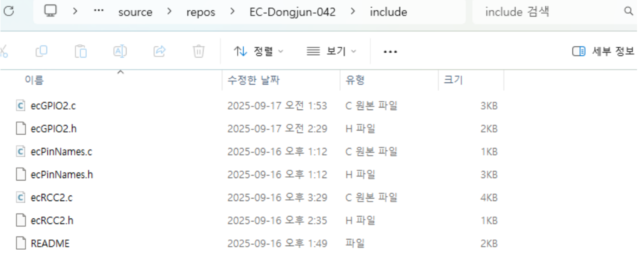
Create the library directory \repos\EC\include



| 이름 | 수정한 날짜 | 유형 | 크기 |
|---|---|---|---|
| .pio | 2025-09-17 오전 2:30 | 파일 폴더 | |
| .vscode | 2025-09-17 오전 1:22 | 파일 폴더 | |
| docs | 2025-09-09 오후 3:14 | 파일 폴더 | |
| include | 2025-09-16 오후 3:15 | 파일 폴더 | |
| lab | 2025-09-17 오전 2:15 | 파일 폴더 | |
| lib | 2025-09-16 오후 3:15 | 파일 폴더 | |
| src | 2025-09-16 오후 3:15 | 파일 폴더 | |
| test | 2025-09-16 오후 3:15 | 파일 폴더 | |
| tutorial | 2025-09-16 오후 3:25 | 파일 폴더 | |
| .gitignore | 2025-09-16 오후 3:15 | Git Ignore 원본 파... | 1KB |
| EC.code-workspace | 2025-09-16 오후 3:18 | Code Workspace ... | 1KB |
| en.DM00119316.pdf | 2025-09-16 오후 1:08 | Microsoft Edge P... | 13,856KB |
| platformio.ini | 2025-09-17 오전 2:30 | 구성 설정 | 3KB |
| README.md | 2025-09-09 오후 3:12 | markdownpad2 | 1KB |

Download necessary library files:

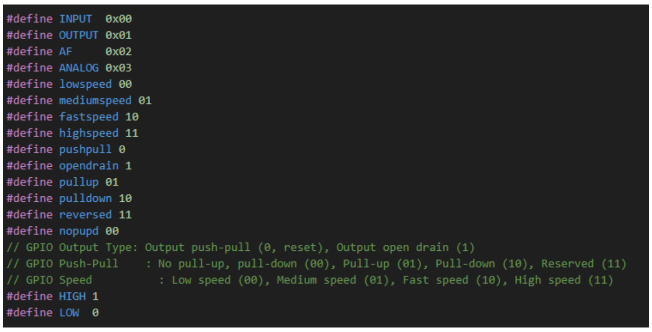
ecRCC2.h, ecRCC2.c,  ecPinNames.h, ecPinNames.c,  ecGPIO2.c, ecGPIO2.h



| 이름 | 수정한 날짜 | 유형 | 크기 |
|---|---|---|---|
| ecGPIO2.c | 2025-09-17 오전 1:53 | C 원본 파일 | 3KB |
| ecGPIO2.h | 2025-09-17 오전 2:29 | H 파일 | 2KB |
| ecPinNames.c | 2025-09-16 오후 1:12 | C 원본 파일 | 1KB |
| ecPinNames.h | 2025-09-16 오후 1:12 | H 파일 | 3KB |
| ecRCC2.c | 2025-09-16 오후 3:29 | C 원본 파일 | 4KB |
| ecRCC2.h | 2025-09-16 오후 2:35 | H 파일 | 1KB |
| README | 2025-09-16 오후 1:49 | 파일 | 2KB |

### Description with Code

### ecGPIO2.h



```
#define INPUT  0x00
#define OUTPUT 0x01
#define AF     0x02
#define ANALOG 0x03
#define lowspeed 00
#define mediumspeed 01
#define fastspeed 10
#define highspeed 11
#define pushpull 0
#define opendrain 1
#define pullup 01
#define pulldown 10
#define reversed 11
#define nopupd 00
// GPIO Output Type: Output push-pull (0, reset), Output open drain (1)
// GPIO Push-Pull    : No pull-up, pull-down (00), Pull-up (01), Pull-down (10), Reserved (11)
// GPIO Speed          : Low speed (00), Medium speed (01), Fast speed (10), High speed (11)
#define HIGH 1
#define LOW  0
```

Additional Definition for lowspeed~nopud for coding convenience

**Discussion**

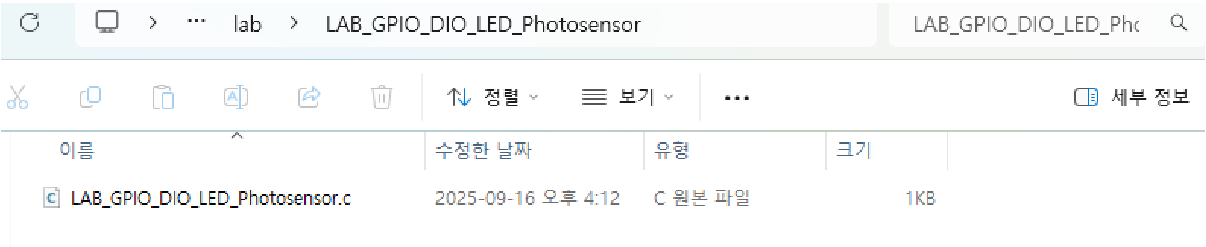
- Find out a typical solution for software debouncing and hardware debouncing.
    - A typical solution for software debouncing is to use time-based filtering or a state change detection
        - Time-based filtering (delay method): After detecting a button press, the program waits for a short delay (about 10-20ms) before confirming the input.
        - State-change detection (edge detection): As used in the LED toggle code, the program only reacts when input changes from 0 -> 1 (rising edge) or 1->0(falling edge). This ensures that the LED toggles once per press
    - A typical solution for hardware debouncing is to use an RC(resistor-capacitor) filter or Schmitt Trigger circuit
        - The RC network smooths out the rapid on/off transitions caused by switch bounce.
        - A Schmitt Trigger provides hysteresis and ensures a clean digital signal at the MCU input.
- What method of debouncing did this NUCLEO board use for the push-button(B1)?
    - The NUCLEO board does not implement hardware debouncing for the B1 push-button. It simply connects the button to the MCU pin with pull-up or pull-down resistor. There for, debouncing must be handled in software. There for, I chose to use ‘State-change detection’ method.

## Problem 2: Toggle a single LED with Digital Sensor(Photodetector)

**Procedure**

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

- The project name is “**LAB_GPIO_DIO_LED_Photosensor”.**
- Name the source file as “**LAB_GPIO_DIO_LED_Photosensor.c”**



| 이름 | 수정한 날짜 | 유형 | 크기 |
|---|---|---|---|
| LAB_GPIO_DIO_LED_Photosensor.c | 2025-09-16 오후 4:12 | C 원본 파일 | 1KB |

2. Include your library **ecGPIO2.h, ecGPIO2.c** in \repos\EC\include\.

3. Toggle the LED by covering the photodetector sensor.

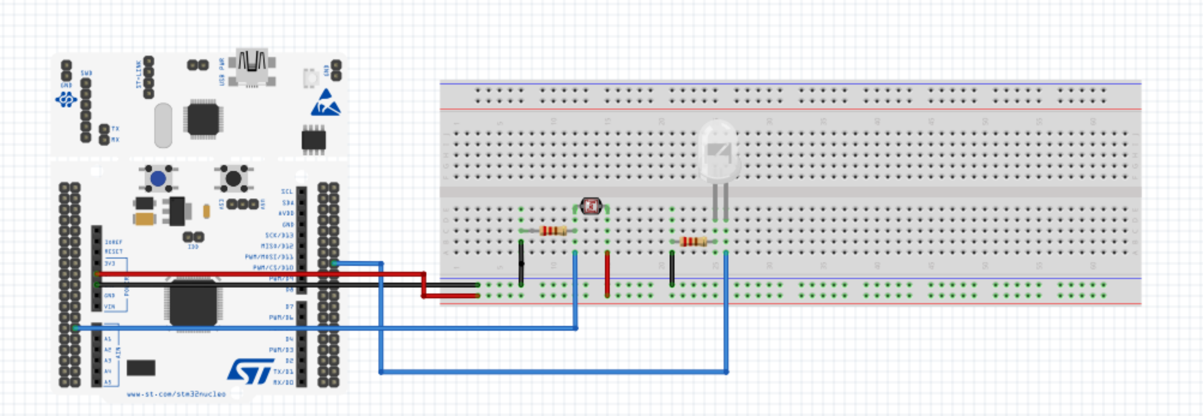
- Dark (LED ON), Bright (LED OFF) and repeat

## Configuration

| Digital Sensor(Photodectector) | LED |
|---|---|
| Digital in | Digital OUT |
| GPIOA, Pin 0 | GPIOC, Pin 3 |
| PULL-UP | Open-Drain, Pull-up, Medium Speed |

## Circuit/Wiring Diagram

External circuit diagram that connects MCU pins to peripherals(sensor/actuator)

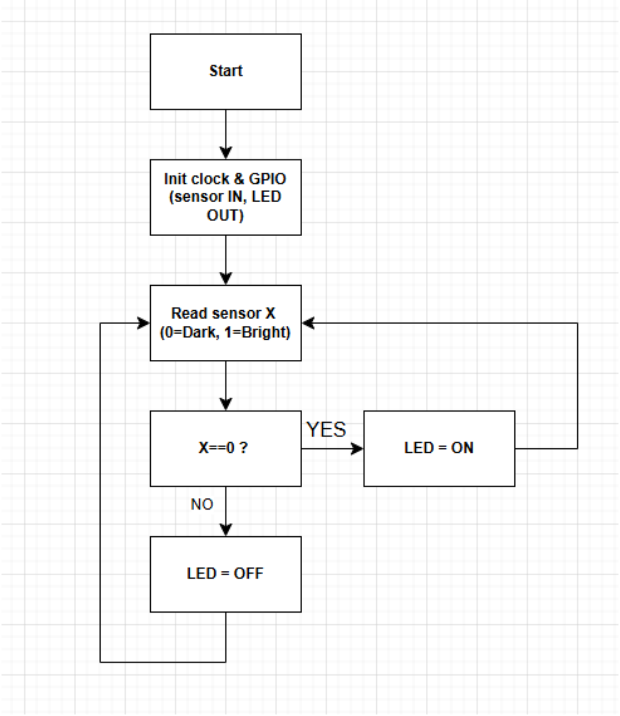


## Algorithm

**Mealy FSM Table**

| Present State | Next State (X=Dark=0) | Next State (X=Bright=1 | Output when X=0 (Dark) | Output when X=1 (Bright) |
|---|---|---|---|---|
| S0 (LED=OFF) | S1 | S0 | LED=ON | LED=OFF |
| S1 (LED=ON) | S1 | S0 | LED=ON | LED=OFF |

**Flowchart**


Start
Init clock & GPIO (sensor IN, LED OUT)
Read sensor X (0=Dark, 1=Bright)
X==0 ?
YES
LED = ON
NO
LED = OFF

## Description with Code

-Lab source code[https://github.com/henny041520-commits/EC-DJLee-042/blob/main/lab/LAB_GPIO_DIO_LED_Photosensor/LAB_GPIO_DIO_LED_Photosensor.c](https://github.com/henny041520-commits/EC-DJLee-042/blob/main/lab/LAB_GPIO_DIO_LED_Photosensor/LAB_GPIO_DIO_LED_Photosensor.c)

Explain your source code with necessary comments

-Description 1

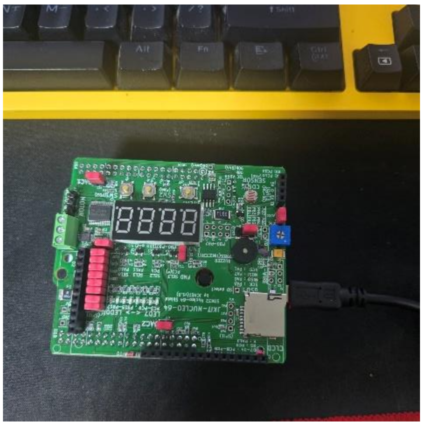
```
Setup

void setup(void)

{

    RCC_HSI_init();                     <- enable 16MHz HSI clock

    GPIO_init(button_pin, INPUT);       <- PA0: photosensor input

    GPIO_pupd(button_pin, pullup);      <- enable internal pull-up

    GPIO_init(LED_pin, OUTPUT);         <- PB12: LED output

    GPIO_otype(LED_pin, opendrain);     <- output type = Open-Drain

    GPIO_pupd(LED_pin, pullup);         <- line held HIGH when not driven

    GPIO_ospeed(LED_pin, mediumspeed);  <- output slew = Medium

}
```

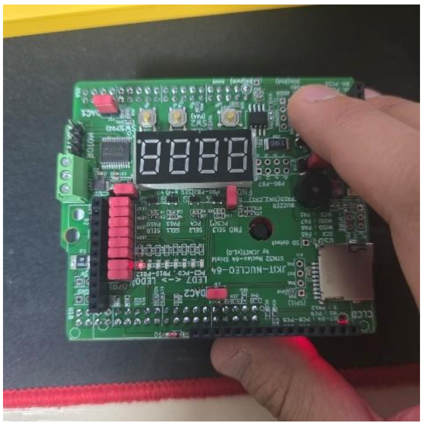
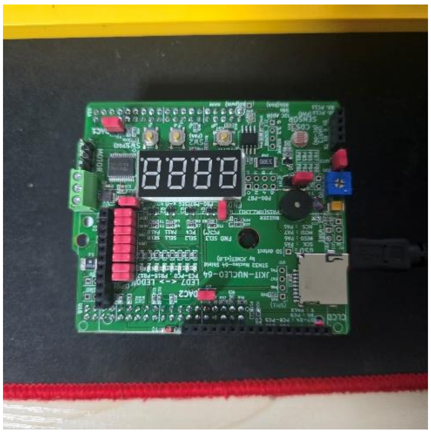
-Description 2

```
int main(void) {
    setup();
    while(1){
     if(GPIO_read(button_pin) == 0)  <- GPIO_read(button_pin) = 0(dark)
                                        GPIO_read(button_pin) = 1 (Bright)
         GPIO_write(LED_pin, 1);     <- Dark -> LED ON
        else
         GPIO_write(LED_pin, 0);      <- Bright -> LED OFF
    }
}
```

# Results and Analysis

## Results

| Results | Analysis |
|---|---|
|  | Bright<br>(GPIO_read(button_pin) == 1)<br>LED OFF |

| | |
|---|---|
|  | Dark<br>(GPIO_read(button_pin) == o)<br>LED ON |
|  | Bright<br>(GPIO_read(button_pin) == 1)<br>LED OFF |

## Demo Video

https://youtube.com/shorts/xrf6yfmJF2U?feature=share

## Analysis

- Work as intended
  - Dark-> LED ON , Bright-> LED OFF, behavior is consistent across repeated trials
- Fast response
  - Simple polling loop; no perceptible latency
- Edge flicker
  - At threshold lighting, minor flicker appeared-> It can be solved by mitigating with 10-20ms software debouncing.

Reference

**STMicroelectronics, RM0383 — STM32F411xC/E Reference Manual**

https://www.st.com/resource/en/reference_manual/rm0383-stm32f411xce-advanced-armbased-32bit-mcus-stmicroelectronics.pdf

**STMicroelectronics, UM1724 — STM32 Nucleo-64 Boards User Manual**

https://www.st.com/resource/en/user_manual/um1724-stm32-nucleo64-boards-mb1136-stmicroelectronics.pdf

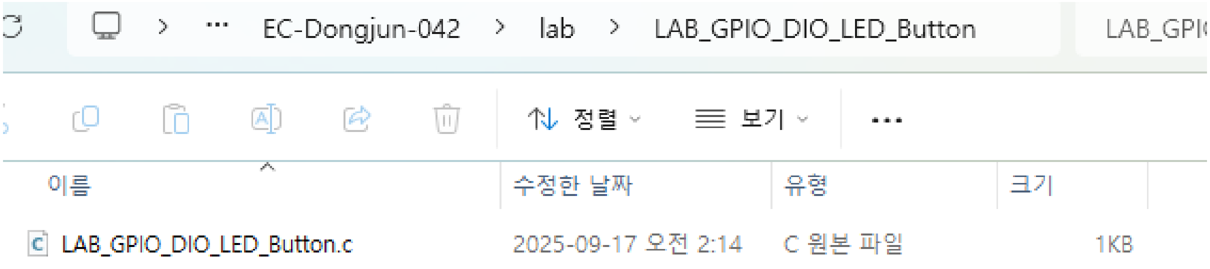
**안경잡이 개발자, 아두이노(Arduino) 빛 감지 센서(Photo Resistor)사용해보기**

https://blog.naver.com/ndb796/221257578214

### Problem 3: Toggle a single LED with a Button Procedure

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

- The project name is “**LAB_GPIO_DIO_LED_Button”.**
- Name the source file as “**LAB_GPIO_DIO_LED_Button.c”**



| 이름 | 수정한 날짜 | 유형 | 크기 |
|---|---|---|---|
| LAB_GPIO_DIO_LED_Button.c | 2025-09-17 오전 2:14 | C 원본 파일 | 1KB |

2. Include your library **ecGPIO2.h, ecGPIO2.c** in \repos\EC\include\.

3. Toggle the LED by pushing the button.

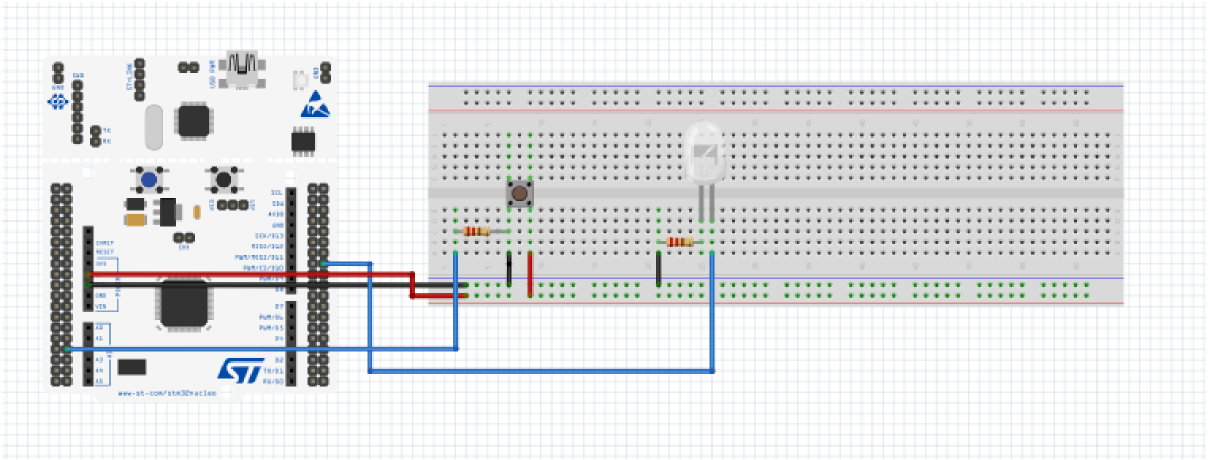
- Push button (LED ON), Push Button (LED OFF) and repeat

**Configuration**

| Button (B1) | LED |
|---|---|
| Digital in | Digital OUT |
| GPIOA, Pin 4 | GPIOB, Pin 12 |
| PULL-UP | Open-Drain, Pull-up, Medium Speed |

## Circuit/Wiring Diagram

External circuit diagram that connects MCU pins to peripherals(sensor/actuator)

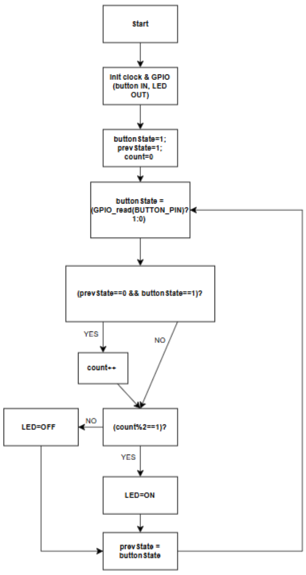


# Algorithm

### Mealy FSM Table

| Present State | Next State (Event=Rise 0→1) | Next State (Else: 1→0 or Hold) | Output when Event=Rise | Output otherwise |
|---|---|---|---|---|
| S0 (LED=OFF) | S1 | S0 | LED=ON | LED=OFF |
| S1 (LED=ON) | S0 | S1 | LED=OFF | LED=ON |

### Flowchart


Start
Init clock & GPIO (button IN, LED OUT)
buttonState=1; prevState=1; count=0
buttonState = (GPIO_read(BUTTON_PIN)? 1:0)
(prevState==0 && buttonState==1)?
YES
NO
count++
(count%2==1)?
NO
LED=OFF
YES
LED=ON
prevState = buttonState

# Description with Code

-Lab source code https://github.com/henny041520-commits/EC-DJLee-042/blob/main/lab/LAB_GPIO_DIO_LED_Button/LAB_GPIO_DIO_LED_Button.c

Explain your source code with necessary comments

-Description 1

```
Setup

void setup(void) {

    RCC_HSI_init();                 <- enable 16MHz HSI clock

    GPIO_init(BUTTON_PIN, INPUT);   <- PA4: push button input (active-low)

    GPIO_pupd(BUTTON_PIN, pullup);  <- enable internal pull-up

    GPIO_init(LED_PIN, OUTPUT);     <- PB12 as digital OUTPUT

    GPIO_otype(LED_PIN, opendrain); <- output type = Open-Drain

    GPIO_pupd(LED_PIN, pullup       <- line held HIGH when not driven

    GPIO_ospeed(LED_PIN, mediumspeed);  <- output slew = Medium

}

void setup(void)

{

    RCC_HSI_init();

    GPIO_init(button_pin, INPUT);

    GPIO_init(LED_pin, OUTPUT);   <- PB12: LED output

}
```

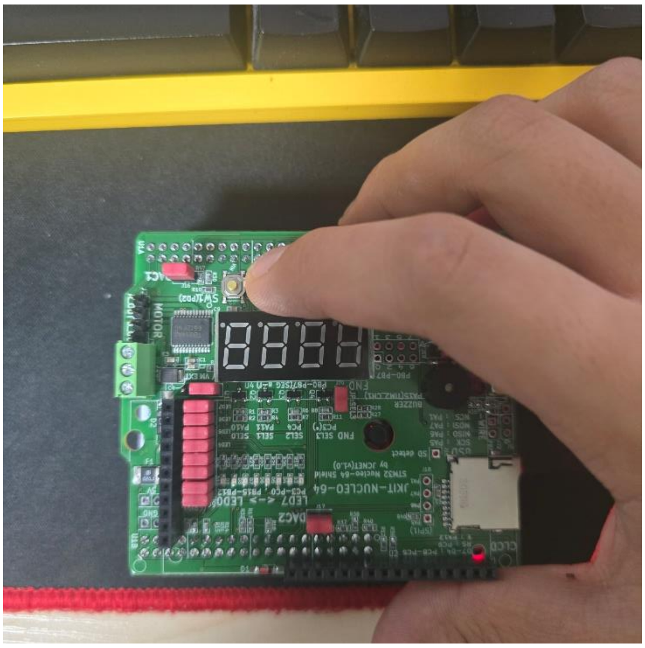
-Description 2

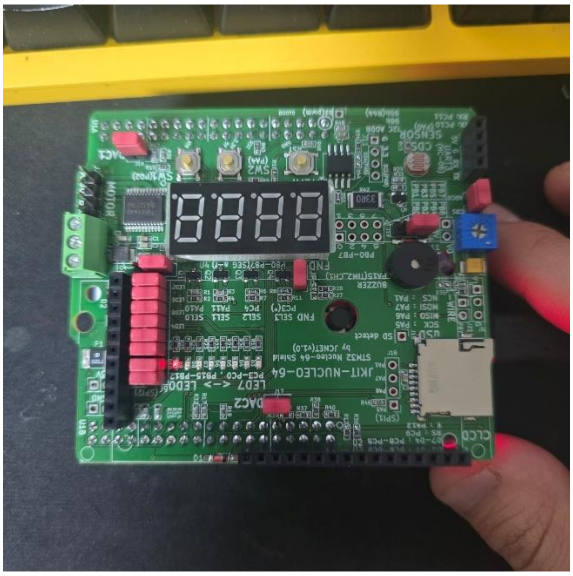
```
int main(void){

    setup();

    int buttonState = 1;              <- start released (1)

    int prevState   = 1;              <- previous input

    int count       = 0;              <- toggle counter (odd=ON, even=OFF)

    while(1){

        buttonState = GPIO_read(BUTTON_PIN) <- normalize: 0 pressed
```

```
                                                      1 released

        rising edge? pressed(0) -> released(1)

    if(prevState == 0 && buttonState == 1)

     {

        count++;                   <- trigger toggle

       }

    if(count % 2 == 1)

      GPIO_write(LED_PIN, 1);     <- LED ON   (odd)

    else

      GPIO_write(LED_PIN, 0);     <- LED OFF  (even)


    prevState = buttonState;        <- update history

}}
```

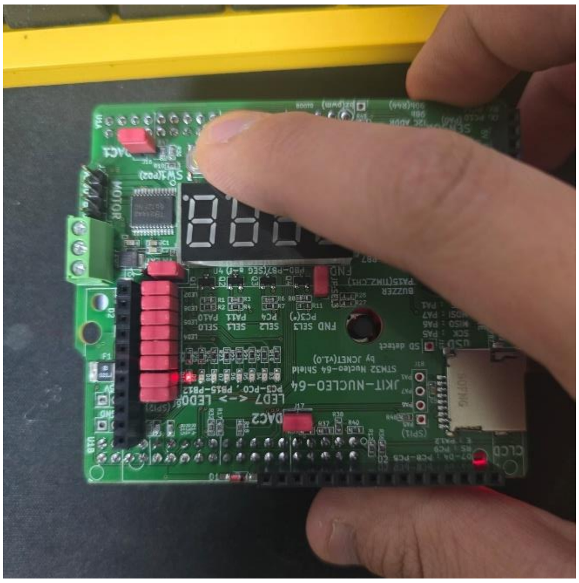
## Results and Analysis

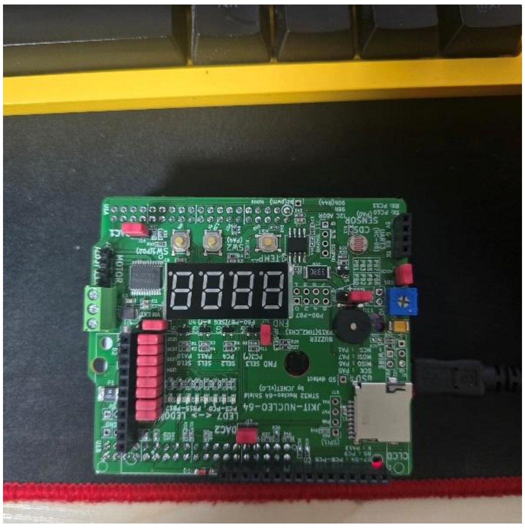
### Results

| Results | Analysis |
|---|---|
|  | 1.Push<br>2.<br>buttonState = 0;<br>prevState = 1;<br>3.<br>prevState = buttonState;<br>buttonState = 0;<br>prevState = 0; |

| Results | Analysis |
|---|---|
|  | 1.Release<br>2.<br>buttonState = 1;<br>prevState = 0;<br>3.<br>(prevState == 0 && buttonState == 1)<br>Count++ (0->1)<br>4.<br>(count % 2 == 1)<br>LED ON<br>5.<br>prevState = buttonState;<br>buttonState = 1;<br>prevState = 1; |

| Results | Analysis |
|---|---|

| | |
|---|---|
|  | 1.Push<br>2.<br>buttonState = 0;<br>prevState   = 1;<br>3.<br>prevState = buttonState;<br>buttonState = 0;<br>prevState   = 0; |

| Results | Analysis |
|---|---|
|  | 1.Release<br>2.<br>buttonState = 1;<br>prevState   = 0;<br>3.<br>(prevState == 0 && buttonState == 1)<br>Count++ (1->2)<br>4.<br>(count % 2 == 1)<br>else<br>LED OFF |

| | |
|---|---|
| | 5.<br>prevState = buttonState;<br>buttonState = 1;<br>prevState   = 1; |

## Demo Video

https://youtube.com/shorts/p9E1w-iWDRE

## Analysis

- Work as intended
  - Each release (rising edge 0->1) toggles the LED once
  - Holding the button does not change the state
  - Across repeated trials at different speeds, the behavior remained consistent
- Fast response
  - Simple polling loop; no perceptible latency for human interaction

## Reference

**STMicroelectronics, RM0383 — STM32F411xC/E Reference Manual**

**https://www.st.com/resource/en/reference_manual/rm0383-stm32f411xce-advanced-armbased-32bit-mcus-stmicroelectronics.pdf**

**STMicroelectronics, UM1724 — STM32 Nucleo-64 Boards User Manual**

**https://www.st.com/resource/en/user_manual/um1724-stm32-nucleo64-boards-mb1136-stmicroelectronics.pdf**

**CODINGRUN, 아두이노 예제 2. 스위치로 LED 켜기 끄기**

**https://codingrun.com/101**

**ARUINODOCS, State Change Detection (Edge Detection) for pushbuttons**

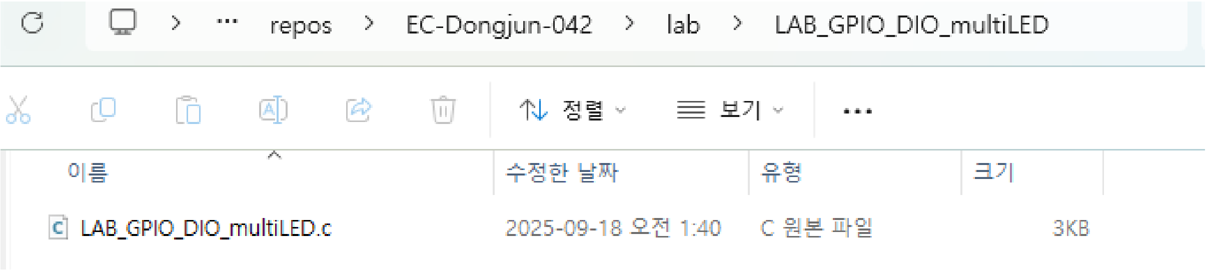
https://docs.arduino.cc/built-in-examples/digital/StateChangeDetection/

**Problem 4: Toggle multiple LEDs with a 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”**



| 이름 | 수정한 날짜 | 유형 | 크기 |
|---|---|---|---|
| LAB_GPIO_DIO_multiLED.c | 2025-09-18 오전 1:40 | C 원본 파일 | 3KB |

2. Include your library **ecGPIO2.h, ecGPIO2.c** in \repos\EC\include\.

3. Connect 4 LEDs **externally** with necessary load resistors.

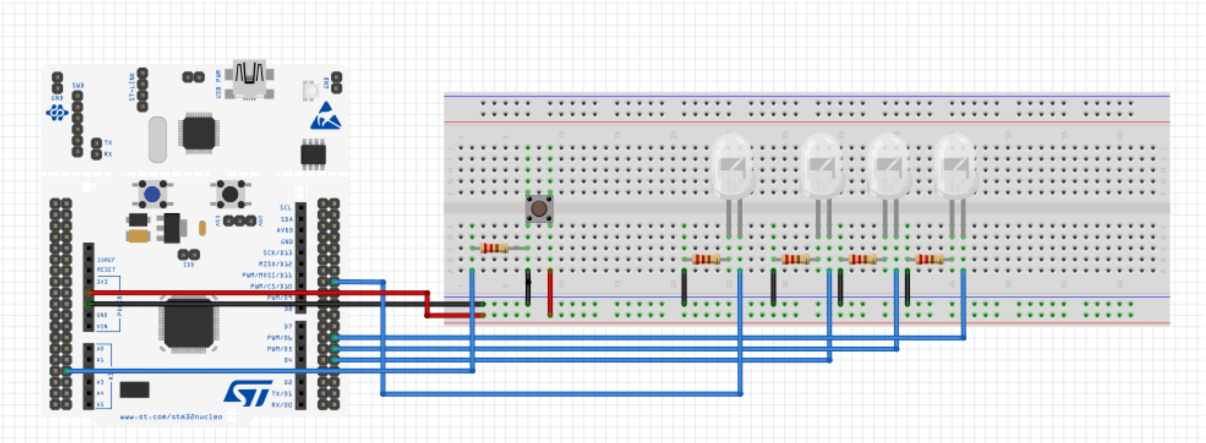
- As Button B1 is Pressed, light one LED at a time, in sequence.
- Example: LED0--> LED1--> …LED3--> …LED0….

## Configuration

| Button | LED |
|---|---|
| Digital in | Digital OUT |
| GPIOA, Pin 4 | PB12,PB13,PB14,PB15 |
| PULL-UP | Push-Pull, Pull-up, Medium Speed |

## Circuit/Wiring Diagram

External circuit diagram that connects MCU pins to peripherals(sensor/actuator)

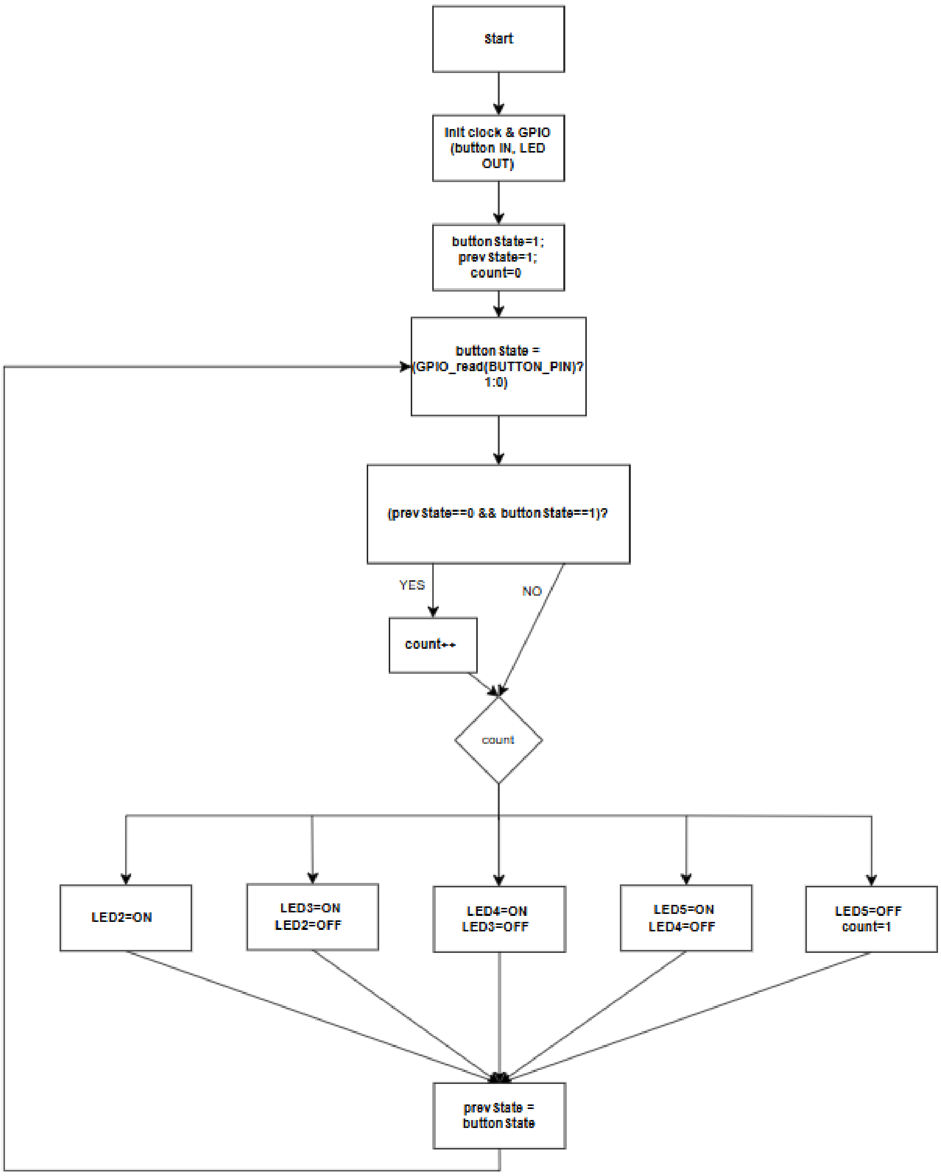


# Algorithm

## Mealy FSM Table

| Present State | Next State on Rise (0→1) | Next State on Else | Output |
|---|---|---|---|
| S0 (ALL OFF) | S1 | S0 | 0000 |
| S1 (LED2 ON) | S2 | S1 | 1000 |
| S2 (LED3 ON) | S3 | S2 | 0100 |
| S3 (LED4 ON) | S4 | S3 | 0010 |
| S4 (LED5 ON) | S1 | S4 | 0001 |

## Flowchart


Start
Init clock & GPIO (button IN, LED OUT)
buttonState=1; prevState=1; count=0
buttonState = (GPIO_read(BUTTON_PIN)? 1:0)
(prevState==0 && buttonState==1)?
YES
NO
count++
count
LED2=ON
LED3=ON LED2=OFF
LED4=ON LED3=OFF
LED5=ON LED4=OFF
LED5=OFF count=1
prevState = buttonState

## Description with Code

[-Lab source codehttps://github.com/henny041520-commits/EC-DJLee-042/blob/main/lab/LAB_GPIO_DIO_multiLED/LAB_GPIO_DIO_multiLED.c](https://github.com/henny041520-commits/EC-DJLee-042/blob/main/lab/LAB_GPIO_DIO_multiLED/LAB_GPIO_DIO_multiLED.c)

Explain your source code with necessary comments

-Description 1

```
Setup:

void setup(void) {
    RCC_HSI_init();                         <- enable 16MHz HSI clock
```

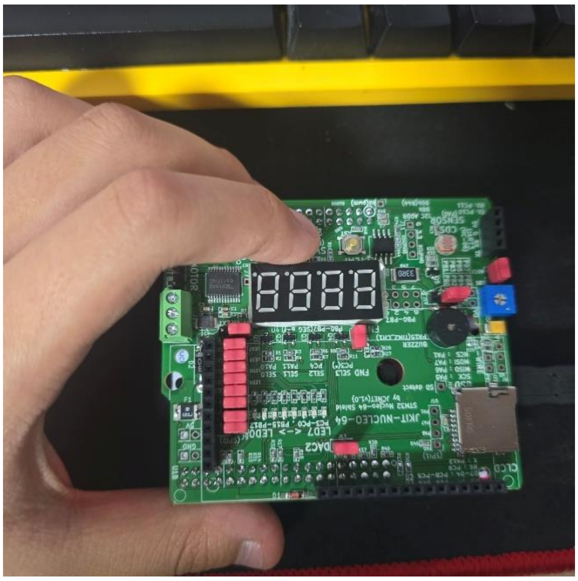
```
  Button B1 (PA4): digital input with pull-up → active-low (pressed=0)

    GPIO_init(BUTTON_PIN, INPUT);

    GPIO_pupd(BUTTON_PIN, pullup);


  LEDs (PB12~PB15): digital outputs (push-pull, medium speed)

    GPIO_init(LED_PIN2, OUTPUT);

    GPIO_init(LED_PIN3, OUTPUT);

    GPIO_init(LED_PIN4, OUTPUT);

    GPIO_init(LED_PIN5, OUTPUT);


    GPIO_otype(LED_PIN2, pushpull);

    GPIO_otype(LED_PIN3, pushpull);

    GPIO_otype(LED_PIN4, pushpull);

    GPIO_otype(LED_PIN5, pushpull);

   GPIO_ospeed(LED_PIN2, mediumspeed);

    GPIO_ospeed(LED_PIN3, mediumspeed);

    GPIO_ospeed(LED_PIN4, mediumspeed);

    GPIO_ospeed(LED_PIN5, mediumspeed);

  LEDs (PB12~PB15): digital outputs with pull-up

    GPIO_pupd(LED_PIN2, pullup);

    GPIO_pupd(LED_PIN3, pullup);

    GPIO_pupd(LED_PIN4, pullup);

    GPIO_pupd(LED_PIN5, pullup);}
```

-Description 2

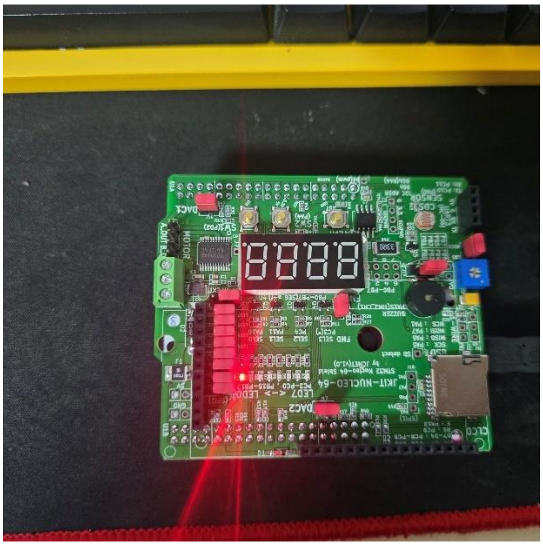
```
int main(void) {

    setup();


    int buttonState = 1;       <-start released
    int prevState   = 1;          <- previous input
    int count       = 0;          <- toggle counter
)
    while(1){
        normalize: 0 pressed 1 released
        buttonState = GPIO_read(BUTTON_PIN) ? 1 : 0;
        rising edge? pressed(0) -> released(1) → advance step
        if(prevState == 0 && buttonState == 1){
            count++; }
        one-hot selection: exactly one LED ON
        switch(count){
        case 1:
            GPIO_write(LED_PIN2, 1);       <- LED_PIN2 ON
            break;
        case 2:
            GPIO_write(LED_PIN2, 0);       <- LED_PIN2 OFF
            GPIO_write(LED_PIN3, 1);       <- LED_PIN3 ON
            break;
        case 3:
            GPIO_write(LED_PIN3, 0);       <- LED_PIN3 OFF
            GPIO_write(LED_PIN4, 1);       <- LED_PIN4 ON
            break;
        case 4:
            GPIO_write(LED_PIN4, 0);       <- LED_PIN4 OFF
            GPIO_write(LED_PIN5, 1);       <- LED_PIN5 ON
            break;
        case 5:
            GPIO_write(LED_PIN5, 0);    <- LED_PIN5 OFF
            count = 1;                  <- wrap back to step 1(LED_PIN2 ON)
            break;}
        prevState = buttonState; }}      <- update history
```

# Results and Analysis

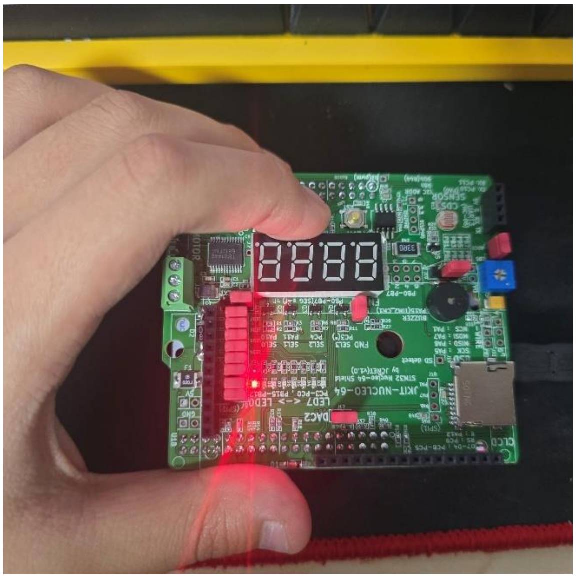
## Results

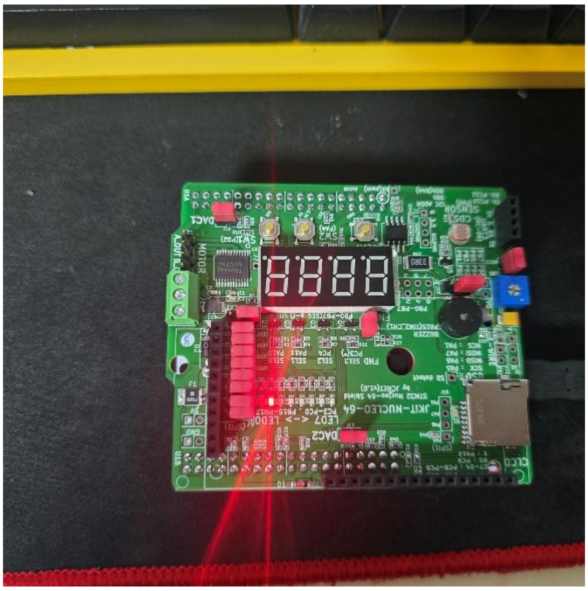
| Results | Analysis |
|---|---|
|  | 1.Push<br>2.<br>buttonState = 0;<br>prevState = 1;<br>3.<br>prevState = buttonState;<br>buttonState = 0;<br>prevState = 0; |

| Results | Analysis |
|---|---|

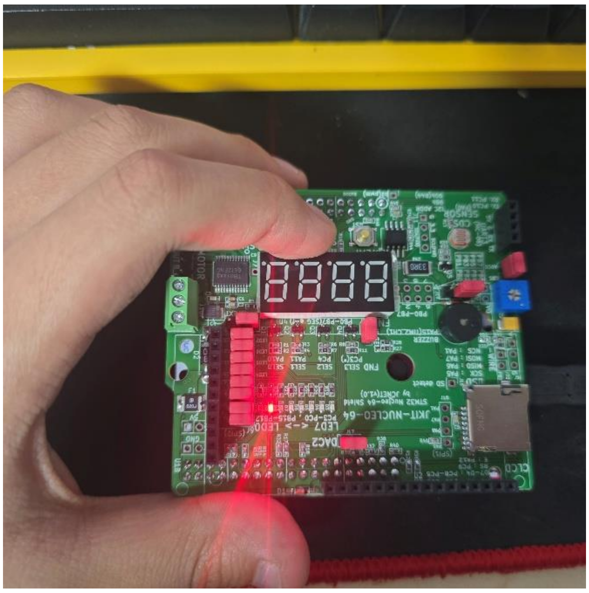
| | |
|---|---|
|  | 1.Release<br>2.<br>buttonState = 1;<br>prevState   = 0;<br>3.<br>(prevState == 0 && buttonState == 1)<br>Count++ (0->1)<br>4.<br>switch(count)-> case 1<br>LED_PIN2 ON<br>5.<br>prevState = buttonState;<br>buttonState = 1;<br>prevState   = 1; |

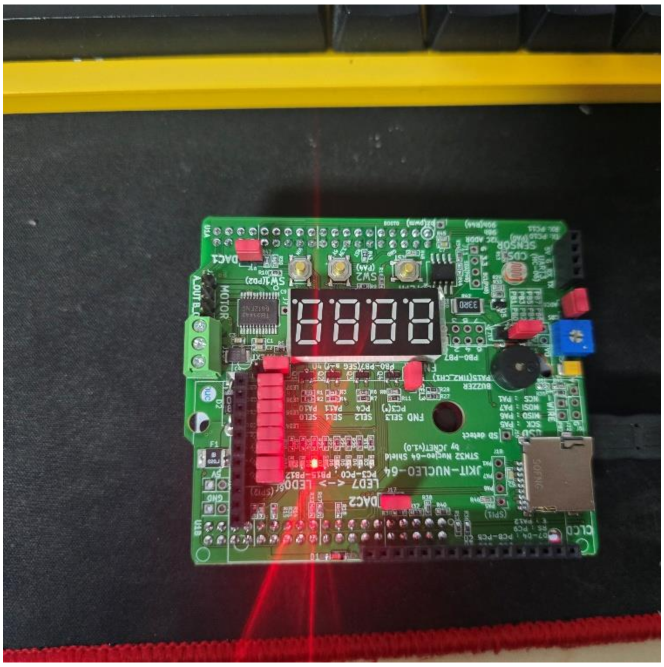
| Results | Analysis |
|---|---|

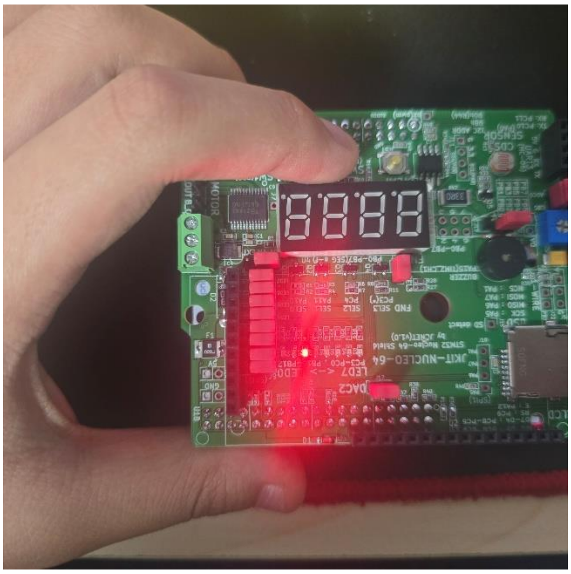
| | |
|---|---|
|  | 1.Push<br>2.<br>buttonState = 0;<br>prevState  = 1;<br>3.<br>prevState = buttonState;<br>buttonState = 0;<br>prevState  = 0; |

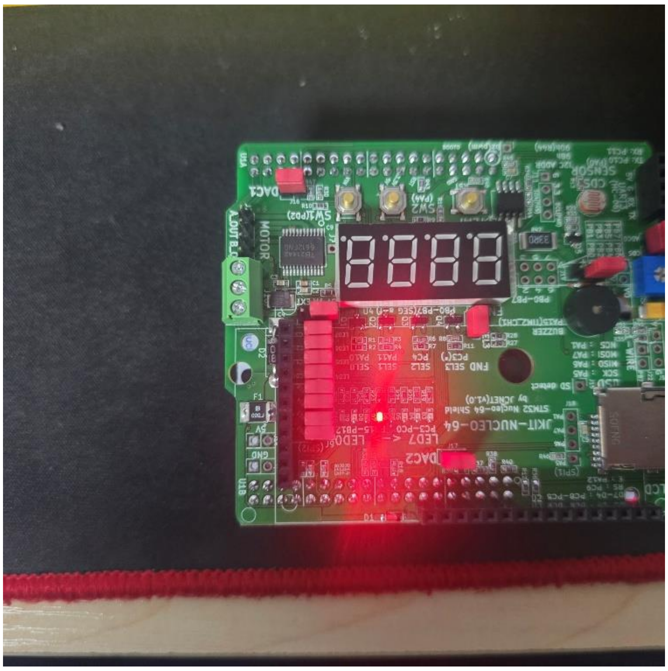
| Results | Analysis |
|---|---|
|  | 1.Release<br>2.<br>buttonState = 1;<br>prevState  = 0;<br>3.<br>(prevState == 0 && buttonState == 1)<br>Count++ (1->2)<br>4.<br>switch(count)-> case 2<br>LED_PIN2 OFF<br>LED_PIN3 ON |

|  |  |
|---|---|
|  | 5.<br>prevState = buttonState;<br>buttonState = 1;<br>prevState   = 1; |

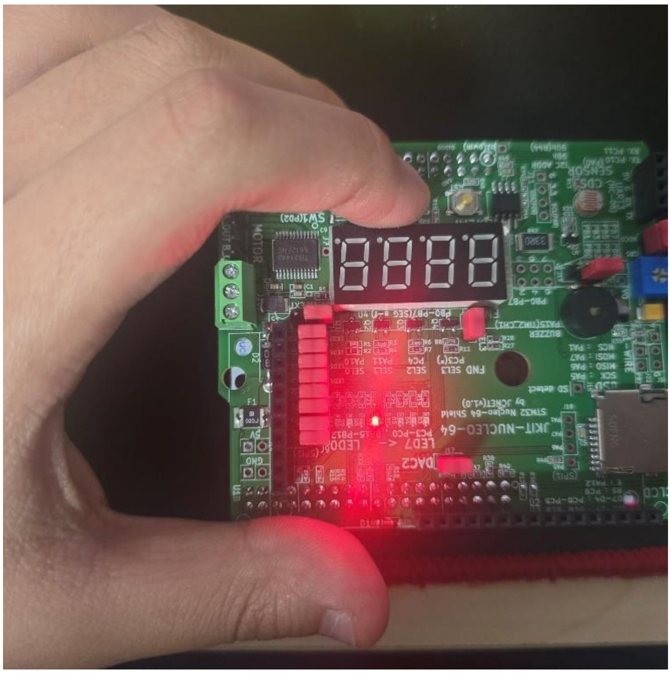
| Results | Analysis |
|---|---|
|  | 1.Push<br>2.<br>buttonState = 0;<br>prevState   = 1;<br>3.<br>prevState = buttonState;<br>buttonState = 0;<br>prevState   = 0; |

| Results | Analysis |
|---|---|
|  | 1.Release<br>2.<br>buttonState = 1;<br>prevState = 0;<br>3.<br>(prevState == 0 && buttonState == 1)<br>Count++ (2->3)<br>4.<br>switch(count)-> case 3<br>LED_PIN3 OFF<br>LED_PIN4 ON<br>5.<br>prevState = buttonState;<br>buttonState = 1;<br>prevState = 1; |

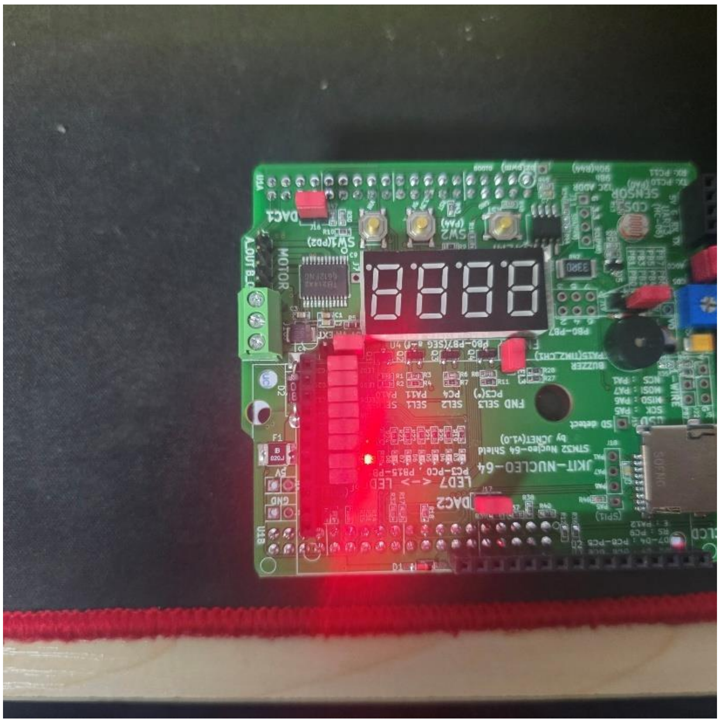
| Results | Analysis |
|---|---|
|  | 1.Push<br>2.<br>buttonState = 0;<br>prevState   = 1;<br>3.<br>prevState = buttonState;<br>buttonState = 0;<br>prevState   = 0; |

| Results | Analysis |
|---|---|
|  | 1.Release<br>2.<br>buttonState = 1;<br>prevState   = 0;<br>3.<br>(prevState == 0 && buttonState == 1)<br>Count++ (3->4)<br>4.<br>switch(count)-> case 4<br>LED_PIN4 OFF |

| | |
|---|---|
| | LED_PIN5 ON<br>5.<br>prevState = buttonState;<br>buttonState = 1;<br>prevState   = 1; |

| Results | Analysis |
|---|---|
|  | 1.Push<br>2.<br>buttonState = 0;<br>prevState   = 1;<br>3.<br>prevState = buttonState;<br>buttonState = 0;<br>prevState   = 0; |

| Results | Analysis |
|---|---|

| | |
|---|---|
|  | 1.Release<br>2.<br>buttonState = 1;<br>prevState = 0;<br>3.<br>(prevState == 0 && buttonState == 1)<br>Count++ (4->5)<br>4.<br>switch(count)-> case 5<br>LED_PIN5 OFF<br>Count=1-> case 1<br>LED_PIN2 ON<br>5.<br>prevState = buttonState;<br>buttonState = 1;<br>prevState = 1; |

## Demo Video

https://youtube.com/shorts/MM9G_ndksnY?feature=share

## Analysis

- Work as intended
  - Each rising edge (0→1) of B1 advances exactly one step: LED12 → LED 13 → LED 14 → LED 15 → wrap. Only one LED is ON at any time (one-hot), matching the FSM table.
- Consistency across speeds
  - Even with varying actuation speeds, repeated tests produced consistent results (not moving steps while holding).
- Next steps
  - The code was messy, but since this lab only required verifying a simple outcome, it wasn't an issue. However, we should identify ways to improve in preparation for future labs.

## Reference

**STMicroelectronics, RM0383 — STM32F411xC/E Reference Manual**

https://www.st.com/resource/en/reference_manual/rm0383-stm32f411xce-advanced-armbased-32bit-mcus-stmicroelectronics.pdf

**STMicroelectronics, UM1724 — STM32 Nucleo-64 Boards User Manual**

https://www.st.com/resource/en/user_manual/um1724-stm32-nucleo64-boards-mb1136-stmicroelectronics.pdf

**CODINGRUN, 아두이노 예제 2. 스위치로 LED 켜기 끄기**

https://codingrun.com/101

**ARUINODOCS, State Change Detection (Edge Detection) for pushbuttons**

https://docs.arduino.cc/built-in-examples/digital/StateChangeDetection/