# **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  
텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

Download necessary library files:

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

텍스트, 번호, 폰트, 소프트웨어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

**Description with Code**

**ecGPIO2.h**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**

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”**

텍스트, 폰트, 라인, 번호이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

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)

스크린샷, 텍스트, 라인, 도표이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

## **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**

텍스트, 도표, 라인, 스크린샷이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

### **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>

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** |
| 전자제품, 컴퓨터 구성 요소, 전자 공학, 컴퓨터 하드웨어이(가) 표시된 사진  AI 생성 콘텐츠는 정확하지 않을 수 있습니다. | Bright (GPIO\_read(button\_pin) == 1)LED OFF |
| 전자 공학, 전자 부품, 회로 구성요소, 전자제품이(가) 표시된 사진  AI 생성 콘텐츠는 정확하지 않을 수 있습니다. | Dark(GPIO\_read(button\_pin) == o)LED ON |
| 전자 부품, 회로 구성요소, 패시브 회로 부품, 전자 공학이(가) 표시된 사진  AI 생성 콘텐츠는 정확하지 않을 수 있습니다. | Bright (GPIO\_read(button\_pin) == 1)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**](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**](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**](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”**

텍스트, 폰트, 라인, 번호이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

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)

스크린샷, 도표, 라인, 평행이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

## **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**

도표, 텍스트, 평면도, 기술 도면이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

### **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** |
| 전자 공학, 전자 부품, 전자제품, 회로 구성요소이(가) 표시된 사진  AI 생성 콘텐츠는 정확하지 않을 수 있습니다. | 1.Push2.buttonState = 0;prevState = 1;3.prevState = buttonState;buttonState = 0;prevState = 0; |

|  |  |
| --- | --- |
| **Results** | **Analysis** |
| 전자 공학, 전자제품, 전자 부품, 회로 구성요소이(가) 표시된 사진  AI 생성 콘텐츠는 정확하지 않을 수 있습니다. | 1.Release2.buttonState = 1;prevState = 0;3.(prevState == 0 && buttonState == 1)Count++ (0->1)4.(count % 2 == 1)LED ON5.prevState = buttonState;buttonState = 1;prevState = 1; |

|  |  |
| --- | --- |
| **Results** | **Analysis** |
| 전자 부품, 전자 공학, 회로 구성요소, 회로이(가) 표시된 사진  AI 생성 콘텐츠는 정확하지 않을 수 있습니다. | 1.Push2.buttonState = 0;prevState = 1;3.prevState = buttonState;buttonState = 0;prevState = 0; |

|  |  |
| --- | --- |
| **Results** | **Analysis** |
| 전자 공학, 전자 부품, 회로 구성요소, 패시브 회로 부품이(가) 표시된 사진  AI 생성 콘텐츠는 정확하지 않을 수 있습니다. | 1.Release2.buttonState = 1;prevState = 0;3.(prevState == 0 && buttonState == 1)Count++ (1->2)4.(count % 2 == 1)elseLED OFF5.prevState = buttonState;buttonState = 1;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**](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**](https://www.st.com/resource/en/user_manual/um1724-stm32-nucleo64-boards-mb1136-stmicroelectronics.pdf)

**CODINGRUN, 아두이노 예제2. 스위치로 LED 켜기 끄기**

[**https://codingrun.com/101**](https://codingrun.com/101)

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

[**https://docs.arduino.cc/built-in-examples/digital/StateChangeDetection/**](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”**

텍스트, 폰트, 라인, 번호이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

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)

텍스트, 스크린샷, 도표, 평행이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

## **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**

도표, 기술 도면, 평면도, 텍스트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

### **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** |
| 전자제품, 전자 공학, 컴퓨터 구성 요소, 전자 부품이(가) 표시된 사진  AI 생성 콘텐츠는 정확하지 않을 수 있습니다. | 1.Push2.buttonState = 0;prevState = 1;3.prevState = buttonState;buttonState = 0;prevState = 0; |

|  |  |
| --- | --- |
| **Results** | **Analysis** |
| 전자제품, 전자 공학, 전자 부품, 회로 구성요소이(가) 표시된 사진  AI 생성 콘텐츠는 정확하지 않을 수 있습니다. | 1.Release2.buttonState = 1;prevState = 0;3.(prevState == 0 && buttonState == 1)Count++ (0->1)4.switch(count) -> case 1LED\_PIN2 ON5.prevState = buttonState;buttonState = 1;prevState = 1; |

|  |  |
| --- | --- |
| **Results** | **Analysis** |
| 전자제품, 전자 공학, 전자 부품, 회로 구성요소이(가) 표시된 사진  AI 생성 콘텐츠는 정확하지 않을 수 있습니다. | 1.Push2.buttonState = 0;prevState = 1;3.prevState = buttonState;buttonState = 0;prevState = 0; |

|  |  |
| --- | --- |
| **Results** | **Analysis** |
| 전자제품, 전자 공학, 전자 부품, 회로 구성요소이(가) 표시된 사진  AI 생성 콘텐츠는 정확하지 않을 수 있습니다. | 1.Release2.buttonState = 1;prevState = 0;3.(prevState == 0 && buttonState == 1)Count++ (1->2)4.switch(count) -> case 2LED\_PIN2 OFFLED\_PIN3 ON5.prevState = buttonState;buttonState = 1;prevState = 1; |

|  |  |
| --- | --- |
| **Results** | **Analysis** |
| 전자 공학, 전자제품, 전자 부품, 회로 구성요소이(가) 표시된 사진  AI 생성 콘텐츠는 정확하지 않을 수 있습니다. | 1.Push2.buttonState = 0;prevState = 1;3.prevState = buttonState;buttonState = 0;prevState = 0; |

|  |  |
| --- | --- |
| **Results** | **Analysis** |
| 전자제품, 전자 공학, 전자 부품, 회로 구성요소이(가) 표시된 사진  AI 생성 콘텐츠는 정확하지 않을 수 있습니다. | 1.Release2.buttonState = 1;prevState = 0;3.(prevState == 0 && buttonState == 1)Count++ (2->3)4.switch(count) -> case 3LED\_PIN3 OFFLED\_PIN4 ON5.prevState = buttonState;buttonState = 1;prevState = 1; |

|  |  |
| --- | --- |
| **Results** | **Analysis** |
| 텍스트, 전자제품, 전자 공학, 전자 부품이(가) 표시된 사진  AI 생성 콘텐츠는 정확하지 않을 수 있습니다. | 1.Push2.buttonState = 0;prevState = 1;3.prevState = buttonState;buttonState = 0;prevState = 0; |

|  |  |
| --- | --- |
| **Results** | **Analysis** |
| 텍스트, 전자 부품, 회로 구성요소, 전자제품이(가) 표시된 사진  AI 생성 콘텐츠는 정확하지 않을 수 있습니다. | 1.Release2.buttonState = 1;prevState = 0;3.(prevState == 0 && buttonState == 1)Count++ (3->4)4.switch(count) -> case 4LED\_PIN4 OFFLED\_PIN5 ON5.prevState = buttonState;buttonState = 1;prevState = 1; |

|  |  |
| --- | --- |
| **Results** | **Analysis** |
| 전자제품, 전자 공학, 텍스트, 전자 부품이(가) 표시된 사진  AI 생성 콘텐츠는 정확하지 않을 수 있습니다. | 1.Push2.buttonState = 0;prevState = 1;3.prevState = buttonState;buttonState = 0;prevState = 0; |

|  |  |
| --- | --- |
| **Results** | **Analysis** |
| 전자 부품, 회로 구성요소, 회로, 전자 공학이(가) 표시된 사진  AI 생성 콘텐츠는 정확하지 않을 수 있습니다. | 1.Release2.buttonState = 1;prevState = 0;3.(prevState == 0 && buttonState == 1)Count++ (4->5)4.switch(count) -> case 5LED\_PIN5 OFFCount=1 -> case 1LED\_PIN2 ON5.prevState = buttonState;buttonState = 1;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**](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**](https://www.st.com/resource/en/user_manual/um1724-stm32-nucleo64-boards-mb1136-stmicroelectronics.pdf)

**CODINGRUN, 아두이노 예제2. 스위치로 LED 켜기 끄기**

[**https://codingrun.com/101**](https://codingrun.com/101)

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

[**https://docs.arduino.cc/built-in-examples/digital/StateChangeDetection/**](https://docs.arduino.cc/built-in-examples/digital/StateChangeDetection/)