LAB: GPIO Digital InOut 7-segment

Date: 2022-10-10

Author/Partner: DongMin Kim / Seongjun Park

Github: https://github.com/DongminKim21800064/EC_dmkim-064/tree/main/lab/EC_LAB_GPIO_7segment_21800064_%EA%B9%80%EB%8F%99%EB%AF%BC

Demo Video: https://youtu.be/zCFLeOTo1D0

PDF version:

Introduction

In this lab, you are required to create a simple program to control a 7-segment display to show a decimal number (0~9) that increases by pressing a push-button.

You must submit

- LAB Report (*.md & *.pdf)
- Zip source files(main*.c, ecRCC.h, ecGPIO.h etc...).
    - Only the source files. Do not submit project files

Requirement

Hardware

- MCU
    - NUCLEO-F411RE
- Actuator/Sensor/Others:
    - 7-segment display(5101ASR)
    - Array resistor (330 ohm)
    - breadboard

Software

- Keil uVision, CMSIS, EC_HAL library

Problem 1: Connecting 7-Segment

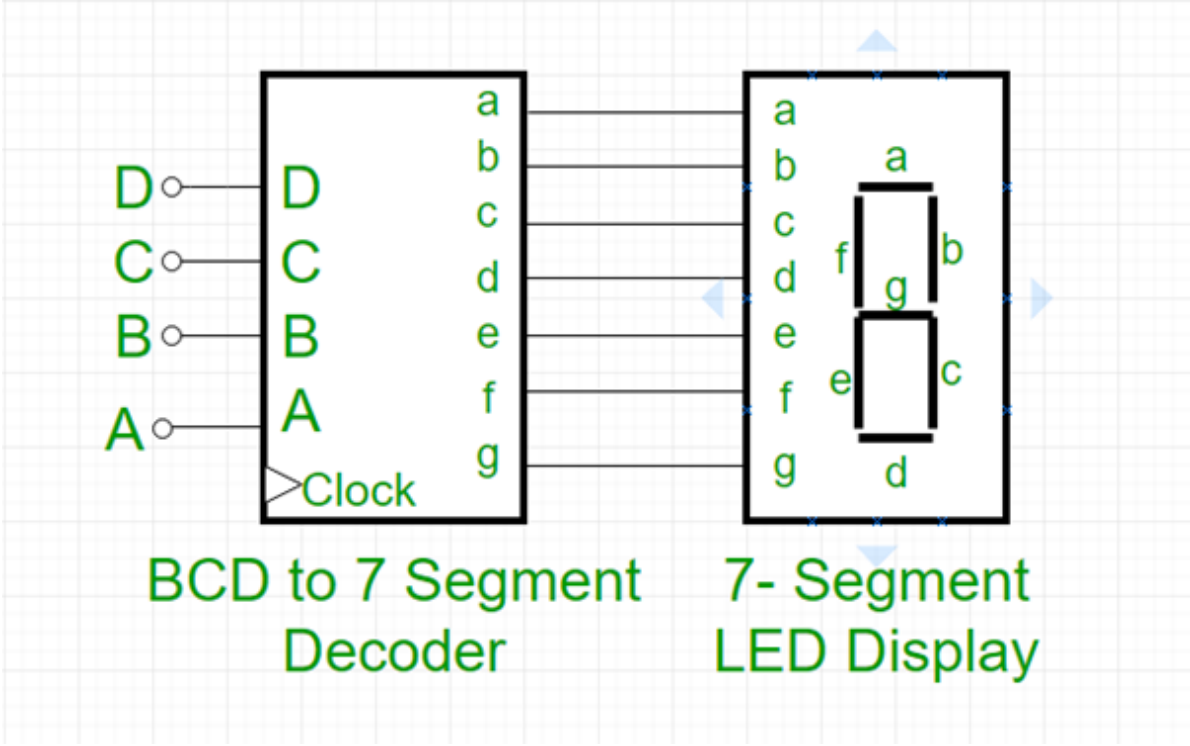
Procedure

Review 7-segment Decoder and Display from Digital Logic lecture.

- Read here: 7-segment BCD tutorial

The popular BCD 7-segment decoder chips are **74LS47 and CD4511**.

Instead of using the decoder chip, we are going to make the 7-segment decoder with the MCU programming.



Connect the common anode 7-segment with the given array resistors.

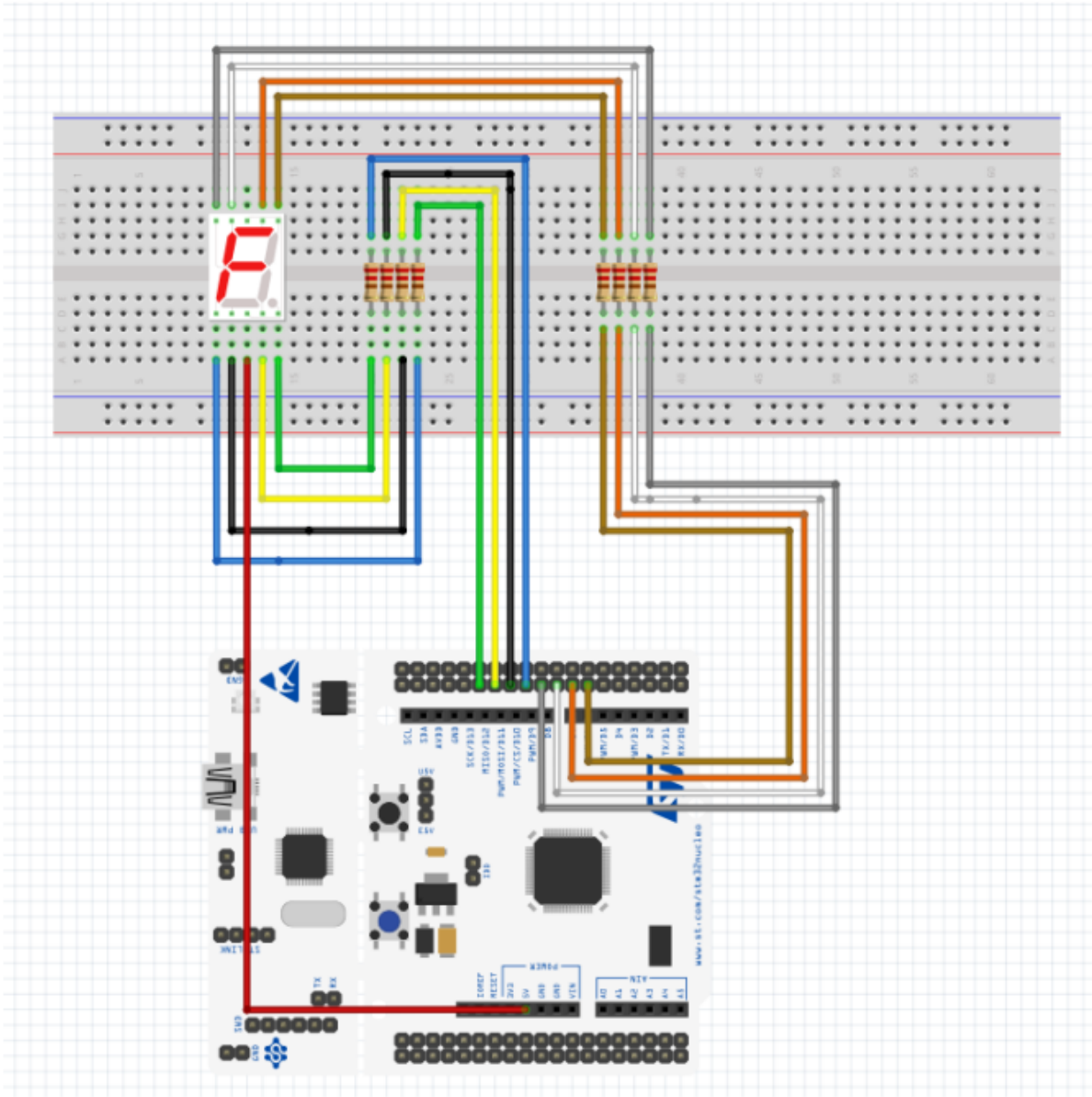
Apply VCC and GND to the 7-segment display.

Apply 'H' to any 7-segment pin 'a'~'g' and observe if that LED is turned on or off

- example: Set 'H' on PA5 of MCU and connect to 'a' of the 7-segment.

Connection Diagram

Circuit diagram



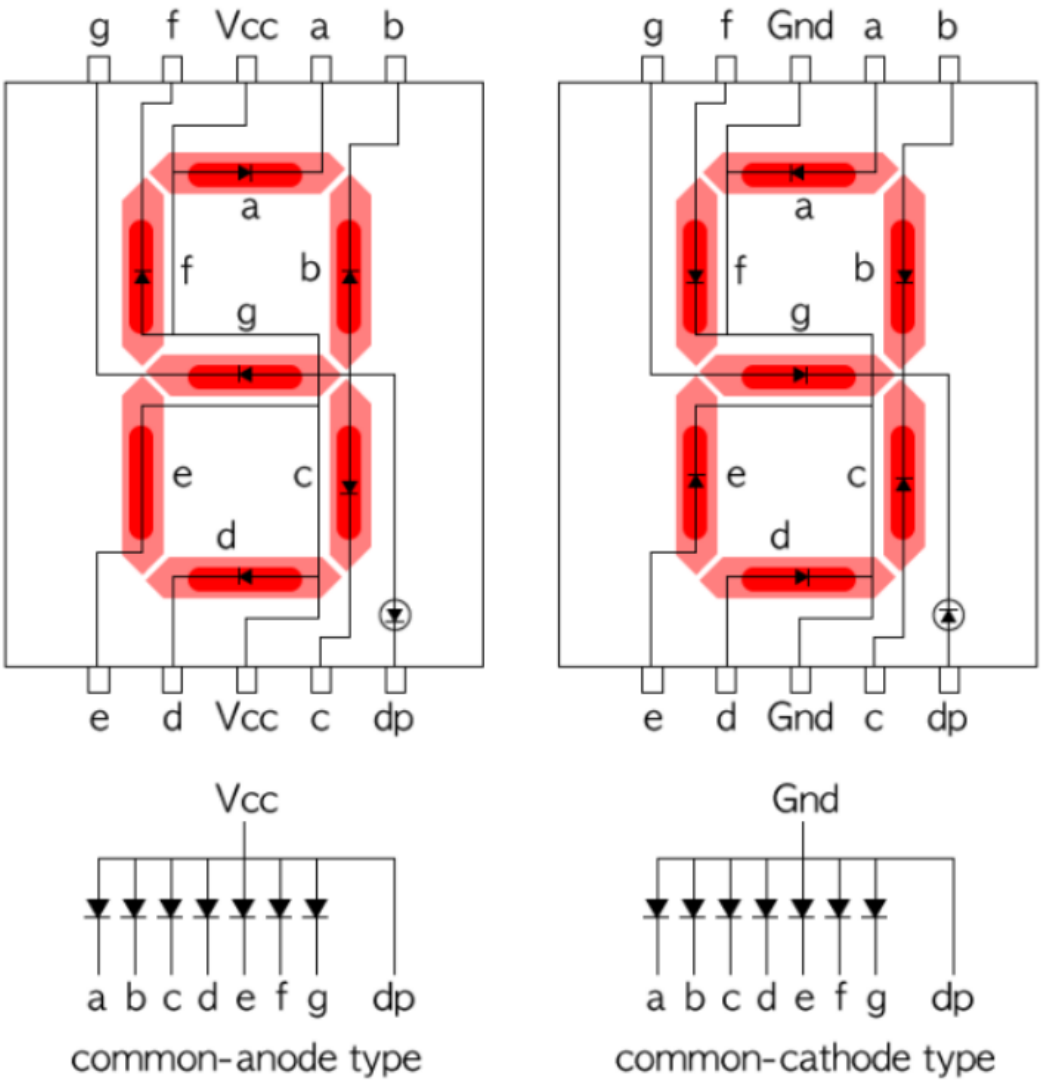
Discussion

1. Draw the truth table for the BCD 7-segment decoder with the 4-bit input.

Num	X3	X2	X1	X0	a	b	c	d	e	f	g	h(DP)
0	0	0	0	0	1	1	1	1	1	1	0	0
1	0	0	0	1	0	1	1	0	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1	0
3	0	0	1	1	1	1	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0	0	1	1	0
5	0	1	0	1	1	0	1	1	0	1	1	0
6	0	1	1	0	1	0	1	1	1	1	1	0
7	0	1	1	1	1	1	1	0	0	1	0	0
8	1	0	0	0	1	1	1	1	1	1	1	0
9	1	0	0	1	1	1	1	1	0	1	1	0

2. What are the common cathode and common anode of 7-segment display?

The way it works has been switched to an anode type and a cathode type. For the anode type, the LED was turned on by the -V (Off) signal, and the LED was turned on by the cathode +V (On) signal.



3. Does the LED of a 7-segment display (common anode) pin turn ON when 'HIGH' is given to the LED pin from the MCU?

No. In the common anode type, the LED was turned on by the -V (Off) signal. Therefore, when 'HIGH' is given to the LED pin, LED will turn OFF.

Problem 2: Display 0~9 with button press

Procedure

1. Create a new project under the directory `\repos\EC\LAB\LAB_GPIO_7segment`

- The project name is "**LAB_GPIO_7segment".**
- Create a new source file named as "**LAB_GPIO_7segment.c"**
- Refer to the sample code

> You MUST write your name on the source file inside the comment section.

2. Include your updated library in `\repos\EC\lib\` to your project.

- **ecGPIO.h, ecGPIO.c**
- **ecRCC.h, ecRCC.c**

1. Declare and Define the following functions in your library
    - You can refer to an example code of 7-segment control

ecGPIO.h

```
void sevensegment_init(void);

void sevensegment_decoder(uint8_t  num);
```

1. First, check if every number, 0 to 9, can be displayed properly
2. Then, create a code to display the number from 0 to 9 with each button press. After the number '9', it should start from '0' again.

Configuration

Digital In for Button (B1)	Digital Out for 7-Segment
Digital In	Digital Out
PC13	PA5, PA6, PA7, PB6, PC7, PA9, PA8, PB10 ('a'~'h', respectively)
PULL-UP	Push-Pull, No Pull-up-Pull-down, Medium Speed

Exercise

Port/Pin	Descripstion	Register setting
Port A pin 5	Clear Pin5 mode	GPIOA->MODER &=~(3<<(2*5))
Port A pin 5	Set Pin5 mode = Output	GPIOA->MODER |=(1<<(2*5))
Port A pin 6	Clear Pin6 mode	GPIOA->MODER &=~(3<<(*6))
Port A pin 6	Set Pin6 mode = Output	GPIOA->MODER |=(1<<(2*6))
Port A pin Y	Clear PinY mode	GPIOA->MODER &=~(3<<(2*Y))
Port A pin Y	Set PinY mode = Output	GPIOA->MODER |=(1<<(2*Y))
Port A pin 5~9	Clear Pin5~9 mode	GPIOA->MODER &=~(1023<<2*5)
	Set Pin5~9 mode = Output	GPIOA->MODER |=(341<<(2*5))
Port X pin Y	Clear Pin Y mode	GPIOX->MODER &=~(3<<(2*Y))
	Set Pin Y mode = Output	GPIOX->MODER |=(1<<(2*Y))
Port A pin 5	Set Pin5 otype=push-pull	GPIOA->OTYPER |= (0<<5)
Port A pin Y	Set PinY otype=push-pull	GPIOA-> OTYPER |= (0<<Y)
Port A pin 5	Set Pin5 ospeed=Fast	GPIOA->OSPEEDR |= (2<<2*5)
Port A pin Y	Set PinY ospeed=Fast	GPIOA-> OSPEEDR |= (2<<2*Y)
Port A pin 5	Set Pin5 PUPD=no pullup/down	GPIOA->OTYPER |= (0<<2*5)
Port A pin Y	Set PinY PUPD=no pullup/down	GPIOA-> OTYPER |= (0<<2*Y)

Code

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

#define LED_PIN     5
#define BUTTON_PIN 13

void setup(void);

int main(void) {
    // Initialiization --------------------------------------------------------
    setup();
    unsigned int cnt = 0;


    // Inifinite Loop ----------------------------------------------------------
    while(1){
        sevensegment_decode(cnt % 10);
        if(GPIO_read(GPIOC, BUTTON_PIN) == 0) cnt++;
        if (cnt > 9) cnt = 0;
        for(int i = 0; i < 500000;i++){}
    }
```

```
}

// Initialiization
void setup(void)
{
    RCC_HSI_init();
    GPIO_init(GPIOC, BUTTON_PIN, INPUT);  // calls RCC_GPIOC_enable()
    sevensegment_init();
}
```

Description1

In void function "setup", I declared 'RCC_HSI_init', ''GPIO_init' and 'sevensegment_init'.'RCC_HSI_init' initializes the RCC_HSI. 'GPIO_init' initializes port C and pin 'BUTTON_PIN' to use input. The detail of function 'seven-segment init()' is in Description3.

Description2

In the main function, set up and declare an unsigned integer type of 'cnt' to get the number 0~9. It will perform in the loop to display the numbers sequentially. The "for statement" in the loop performed a kind of delay.

Description3

Code of function **'sevensegment_init'**

```
void sevensegment_init(void){

    GPIO_init(GPIOA, 8, OUTPUT);        // A
    GPIO_init(GPIOB, 10, OUTPUT);       // B
    GPIO_init(GPIOA, 7, OUTPUT);        // C
    GPIO_init(GPIOA, 6, OUTPUT);        // D
    GPIO_init(GPIOA, 5, OUTPUT);        // E
    GPIO_init(GPIOA, 9, OUTPUT);        // F
    GPIO_init(GPIOC, 7, OUTPUT);        // G
    GPIO_init(GPIOB, 6, OUTPUT);        // DP

    //Set BUTTON_PIN to PULL-UP Mode
    GPIO_pupd(GPIOC, BUTTON_PIN, EC_PU);            // PULL-UP

    //Set 7segment_PIN to NO PULL-UP, PULL-DOWN Mode
    GPIO_pupd(GPIOA, 5, NONE); // no pull-up, pull-down
    GPIO_pupd(GPIOA, 6, NONE);
    GPIO_pupd(GPIOA, 7, NONE);
    GPIO_pupd(GPIOA, 8, NONE);
    GPIO_pupd(GPIOA, 9, NONE);
    GPIO_pupd(GPIOB, 6, NONE);
    GPIO_pupd(GPIOB,10, NONE);
    GPIO_pupd(GPIOC, 7, NONE);

    //Set 7segment_PIN to Push-Pull Mode
    GPIO_otype(GPIOA, 5, PUSH_PULL); //push-pull
    GPIO_otype(GPIOA, 6, PUSH_PULL);
    GPIO_otype(GPIOA, 7, PUSH_PULL);
    GPIO_otype(GPIOA, 8, PUSH_PULL);
    GPIO_otype(GPIOA, 9, PUSH_PULL);
    GPIO_otype(GPIOB, 6, PUSH_PULL);
    GPIO_otype(GPIOB,10, PUSH_PULL);
```

```
    GPIO_otype(GPIOC, 7, PUSH_PULL);

    //Set 7segment_PIN to mid speed Mode
    GPIO_ospeed(GPIOA, 5, MEDIUM_SPEED ); //mid-speed
  GPIO_ospeed(GPIOA, 6, MEDIUM_SPEED );
    GPIO_ospeed(GPIOA, 7, MEDIUM_SPEED );
    GPIO_ospeed(GPIOA, 8, MEDIUM_SPEED );
    GPIO_ospeed(GPIOA, 9, MEDIUM_SPEED );
    GPIO_ospeed(GPIOB, 6, MEDIUM_SPEED );
    GPIO_ospeed(GPIOB,10, MEDIUM_SPEED );
    GPIO_ospeed(GPIOC, 7, MEDIUM_SPEED );
}
```

'sevensegment_init()' initializes digital out pin (PA5, PA6, PA7, PB6, PC7, PA9, PA8, PB10) with to set push-pull, NO PULL-UP-PULL-DOWN and mid speed mode. Plus, set BUTTON_PIN to PULL-UP Mode.

Description4

Code of function **sevensegment_decode**

```
void sevensegment_decode(uint8_t  num){

    // 7-segments Reversed TruthTable
        int number[10][8] = {
                // A B C D E F G DP
                    {0,0,0,0,0,0,1,1},          //zero
                    {1,0,0,1,1,1,1,1},          //one
                    {0,0,1,0,0,1,0,1},          //two
                    {0,0,0,0,1,1,0,1},          //three
                    {1,0,0,1,1,0,0,1},          //four
                    {0,1,0,0,1,0,0,1},          //five
                    {0,1,0,0,0,0,0,1},          //six
                    {0,0,0,1,1,1,1,1},          //seven
                    {0,0,0,0,0,0,0,1},          //eight
                    {0,0,0,0,1,0,0,1},          //nine
                        };

        GPIO_write(GPIOA, 8, number[num][0]);  // A
        GPIO_write(GPIOB, 10, number[num][1]); // B
        GPIO_write(GPIOA, 7, number[num][2]);  // C
        GPIO_write(GPIOA, 6, number[num][3]);  // D
        GPIO_write(GPIOA, 5, number[num][4]);  // E
        GPIO_write(GPIOA, 9, number[num][5]);  // F
        GPIO_write(GPIOC, 7, number[num][6]);  // G
        GPIO_write(GPIOB, 6, number[num][7]);  // DP
    }
```

The function 'sevensegment_decode' is consist of integer type structure 'number' and GPIO_writes for displaying 7-segment.

The structure consists of A to G plus DP pins which are connected with the led of 7-segment.

Refer to truth table of 7-segment, I write the '1' and '0' in the "number", but they are reversed. Because this lab used common anode type of 7-segment. Therefore, only the LEDs set to not send current (-V) will be turn on.

Results

Experiment images and results

Image1 : Whole circuit and MCU.

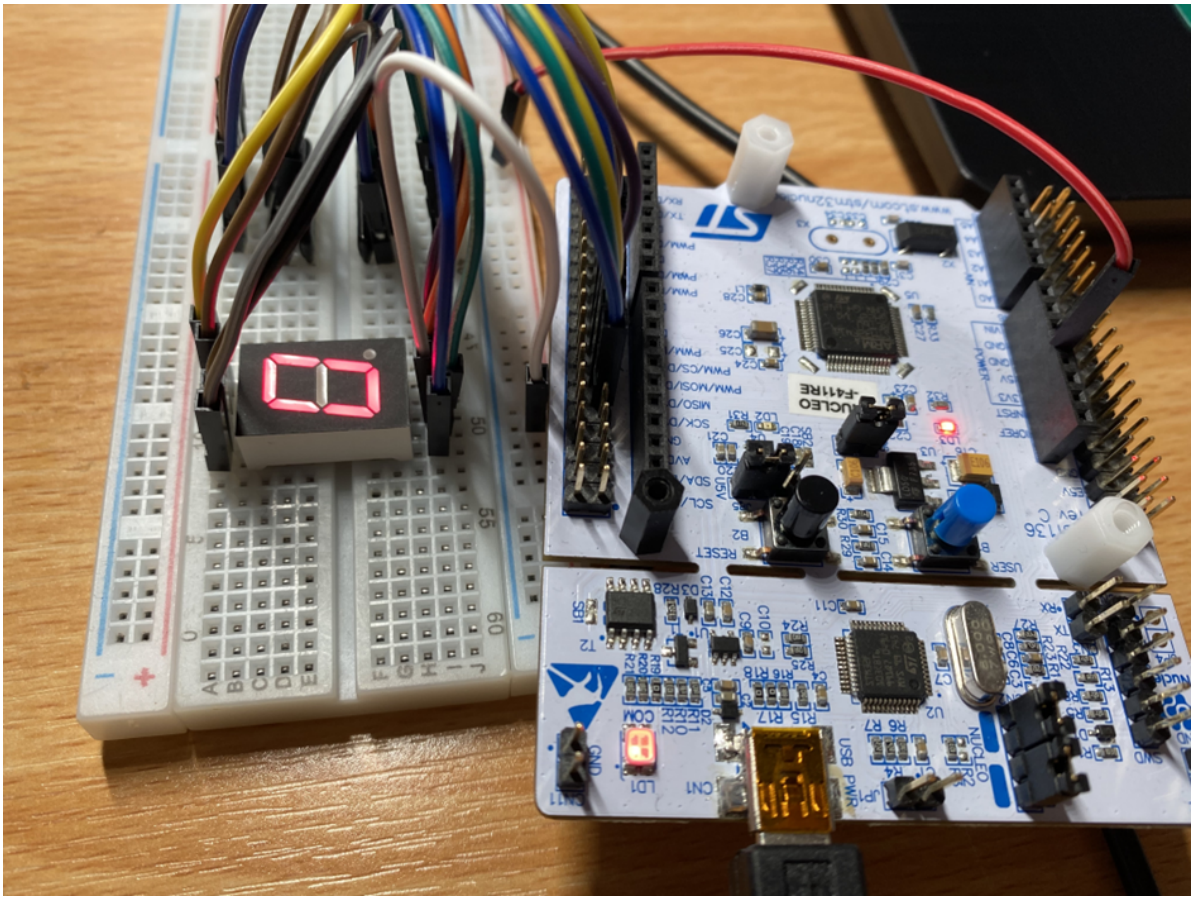
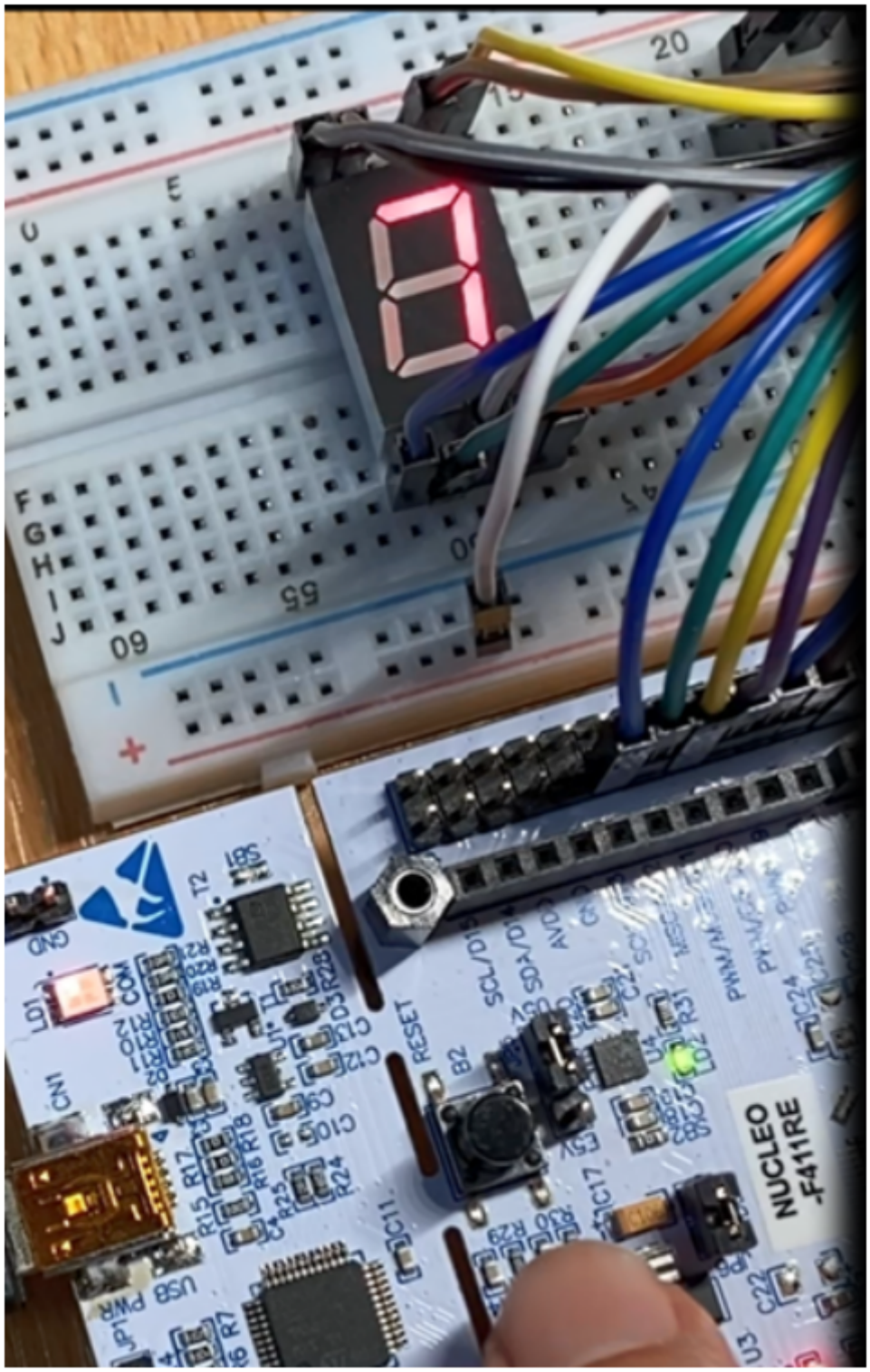


Image2 : Zoom in the 7-segment



Demo Video Link : https://youtu.be/zCFLeOTo1D0

Reference

- **common anode and cathode type of 7-segment**
  - https://kocoafab.cc/tutorial/view/351
- **Lecture provided**
  - https://github.com/ykkimhgu/EC-student/tree/main/tutorial/tutorial-student