LAB: EXTI & SysTick(eval board)

Date: 2025-09-26

Author/Partner: leejeayong

Github: repository link

Demo Video: Youtube link1 Youtube link2

PDF version:

Introduction

In this lab, you are required to create two simple programs using interrupt:

(1) displaying the number counting from 0 to 19 with Button Press

(2) counting at a rate of 1 second

You must submit

- LAB Report (*.md & *.pdf)
- Zip source files(main*.c, ecRCC2.h, ecGPIO2.h, ecSysTick2.c etc...).
  - Only the source files. Do not submit project files

Requirement

Hardware

- MCU
  - NUCLEO-F411RE
- Actuator/Sensor/Others:
  - eval board

Software

- PlatformIO, CMSIS, EC_HAL library

Tutorial: STM-Arduino

{% embed url="https://ykkim.gitbook.io/ec/ec-course/tutorial/tutorial-arduino-stm32/tutorial-arduino-stm32-part1#external-interrupt" %}

We are going to create a simple program that turns LED(LD2) on triggered by **External Interrupt** of user button(BT1)/

attachInterrupt()

```
attachInterrupt(digitalPinToInterrupt(pin), ISR, mode)
```

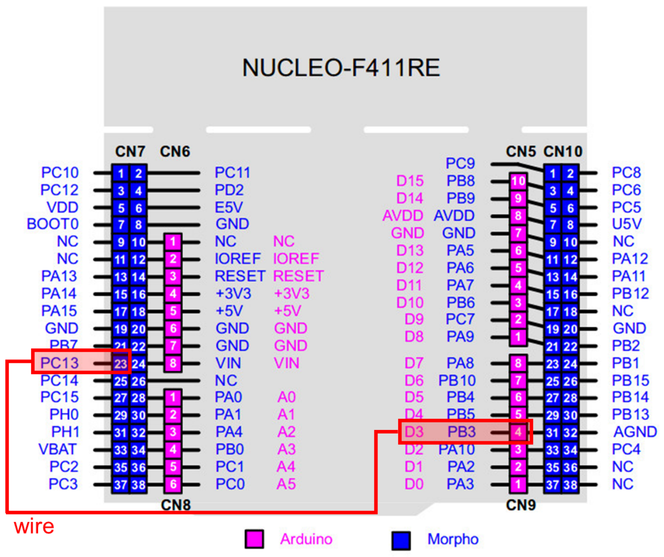
- `digitalPinToInterrupt(pin)`: translate the digital pin to the specific interrupt number.
- `ISR`: a function called whenever the interrupt occurs.
- `mode`: defines when the interrupt should be triggered. (LOW, CHANGE, RISING, FALLING)

Procedure

1. Create a new project under the directory `\EC\lab\LAB_EXTI`
2. Open *Arduino IDE* and Create a new program named as '**TU_arduino_EXTINT.ino**'.
3. Write the following code.

```
const int btnPin = 3;
const int ledPin =  13;

int btnState = HIGH;

void setup() {
    pinMode(ledPin, OUTPUT);
    pinMode(btnPin, INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(btnPin), blink, CHANGE);
}

void loop() {
    // blank
}


void blink(){
    btnState = digitalRead(btnPin);

    if (btnState == HIGH)
        digitalWrite(ledPin, LOW);
    else
        digitalWrite(ledPin, HIGH);
}
```

> The user button pin is `PC13`, but this pin cannot be used in arduino. So, you should connect `PC13` to `pinName D3` by using wire.



4. Click on **upload** button.
5. Whenever the user button(BT1) is pressed (at fall), LED should be ON. When the button is released, the LED should be OFF.

Tutorial: STM32F4xx

1.Tutorial: Managing library header files

Read how to manage library header files for MCU register configurations. Apply it in your LAB.

{% embed url="https://ykkim.gitbook.io/ec/ec-course/tutorial/tutorial-library-header-files#ec-2024" %}

2.Tutorial: Custom Initialization

Instead of writing initial setting functions for each registers, you can call a user defined function e.g. `MCU_init()` for the commonly used default initialization. Follow the tutorial and apply it in your LAB.

{% embed url="https://ykkim.gitbook.io/ec/ec-course/tutorial/tutorial-custom-initialization" %}

Problem 1: Counting numbers on 7-Segment using EXTI Button

Creating EXTI library

1. Download sample header files:

   `ecEXTI2_student.h, ecEXTI2_student.c`

2. Rename these files as **ecEXTI2.h, ecEXTI2.c**

   - You MUST write your name and other information at the top of the library code files.
   - Save these files in your directory `EC \include\`.

3. Declare and define the following functions in your library `ecEXTI2.h`

ecEXTI.h

```c
void EXTI_init(PinName_t pinName, int trig_type, int priority);
void EXTI_enable(uint32_t pin);  // mask in IMR
void EXTI_disable(uint32_t pin);  // unmask in IMR
uint32_t  is_pending_EXTI(uint32_t pin);
void clear_pending_EXTI(uint32_t pin);
```

Procedure

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

- The project name is "**LAB_EXTI".**
- Create a new source file named as "**LAB_EXTI.c"**

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

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

- **ecGPIO2.h, ecGPIO2.c**
- **ecRCC2.h, ecRCC2.c**
- **ecEXTI2.h, ecEXTI2.c**

3. First, check if every number, 0 to 9, can be displayed properly on each 7-segment (there are a total of 4 7-segment display on the evaluation board).
4. Then, create a code to display the number counting from 0 to 19 and repeating.
   - Count up only by pressing the push button
   - Must use External Interrupt
   - Refer to sample codes

Configuration

Digital In for Button (B1)	Digital Out for FND-7-Segment
Digital In	Digital Out
PA4	PB7,PB6,PB5,PB4,PB3,PB2,PB1,PB0 ('a'~~'h', respectively)~~ ~~PC3,PC4,PA11,PA10~~ ~~('LED1'~~'LED4', respectively)
PULL-UP	Push-Pull, No PullUp-PullDown, Medium Speed

Discussion

1. We can use two different methods to detect an external signal: polling and interrupt. What are the advantages and disadvantages of each approach?

> polling has simplicity, Compatibility,Synchronous control for advantage, but it has resource intensive and higher latency, inefficient for disadvantage. interrupt is strong at efficient, responsive but it is

> complex

2. What would happen if the EXTI interrupt handler does not clear the interrupt pending flag? Check with your code

   > Answer discussion questions failing to clear the EXTI interrupt pending flag will result in interrupt flooding making the MCU unable to run its main program as expected and leading to erratic or stuck system behavior. like infinite loop.

Code

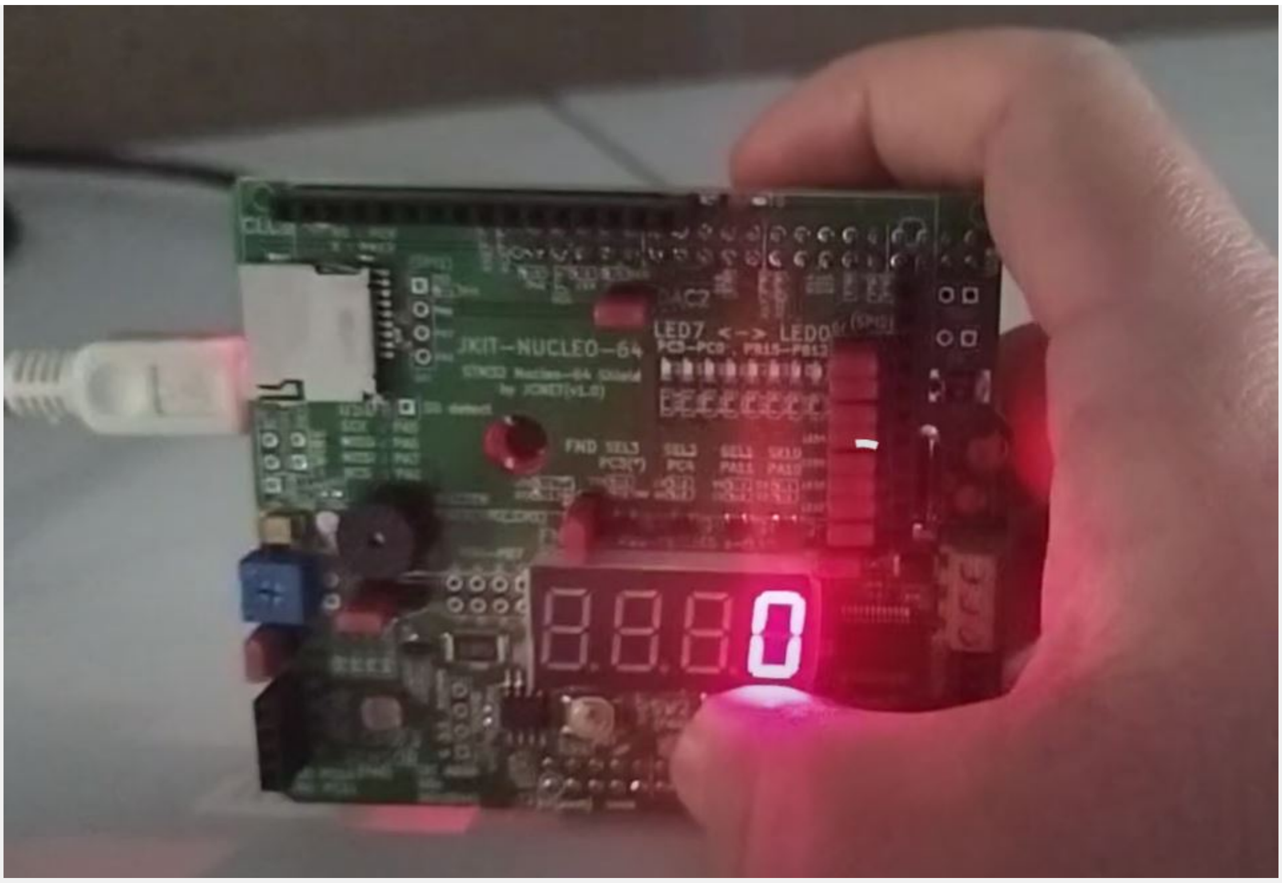
Your code goes here.

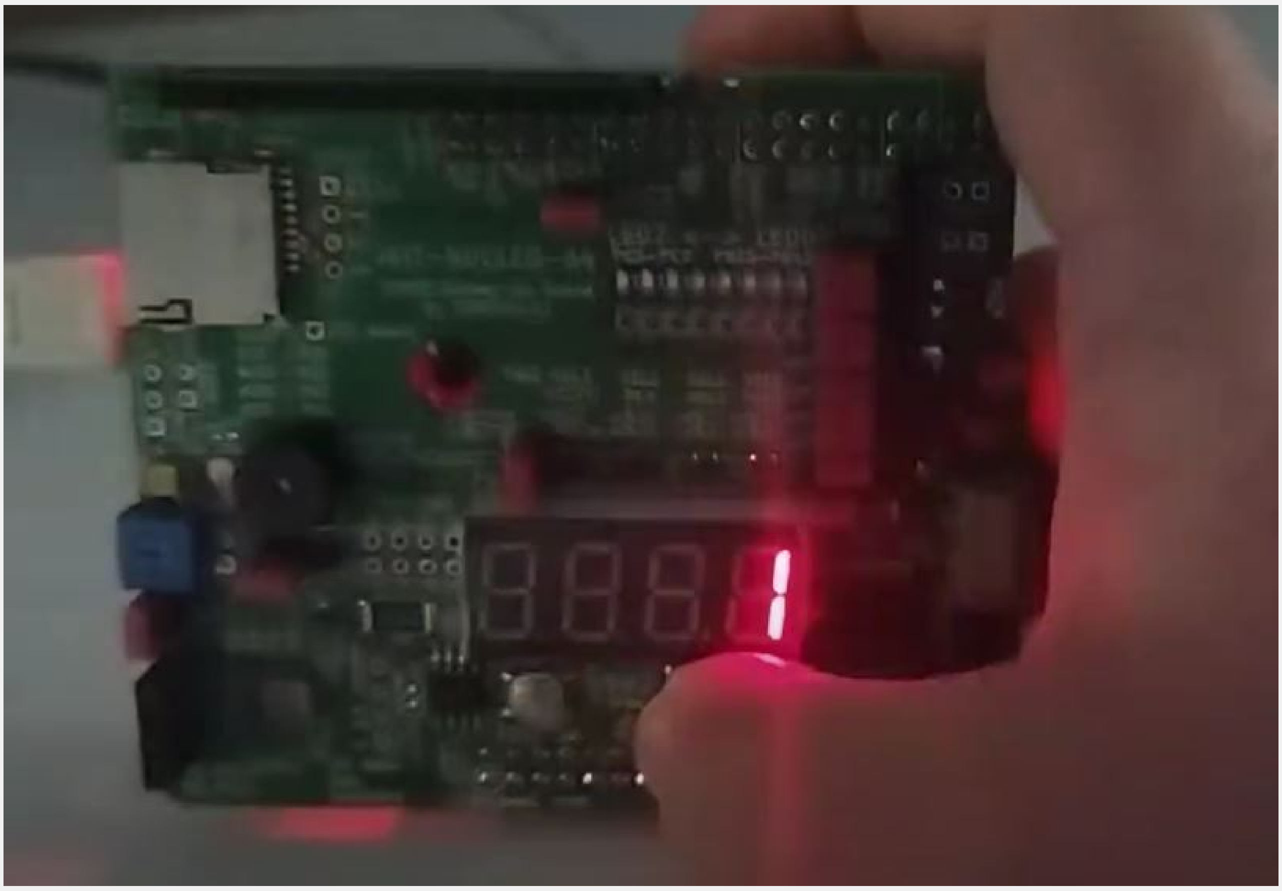
Explain your source code with the necessary comments.

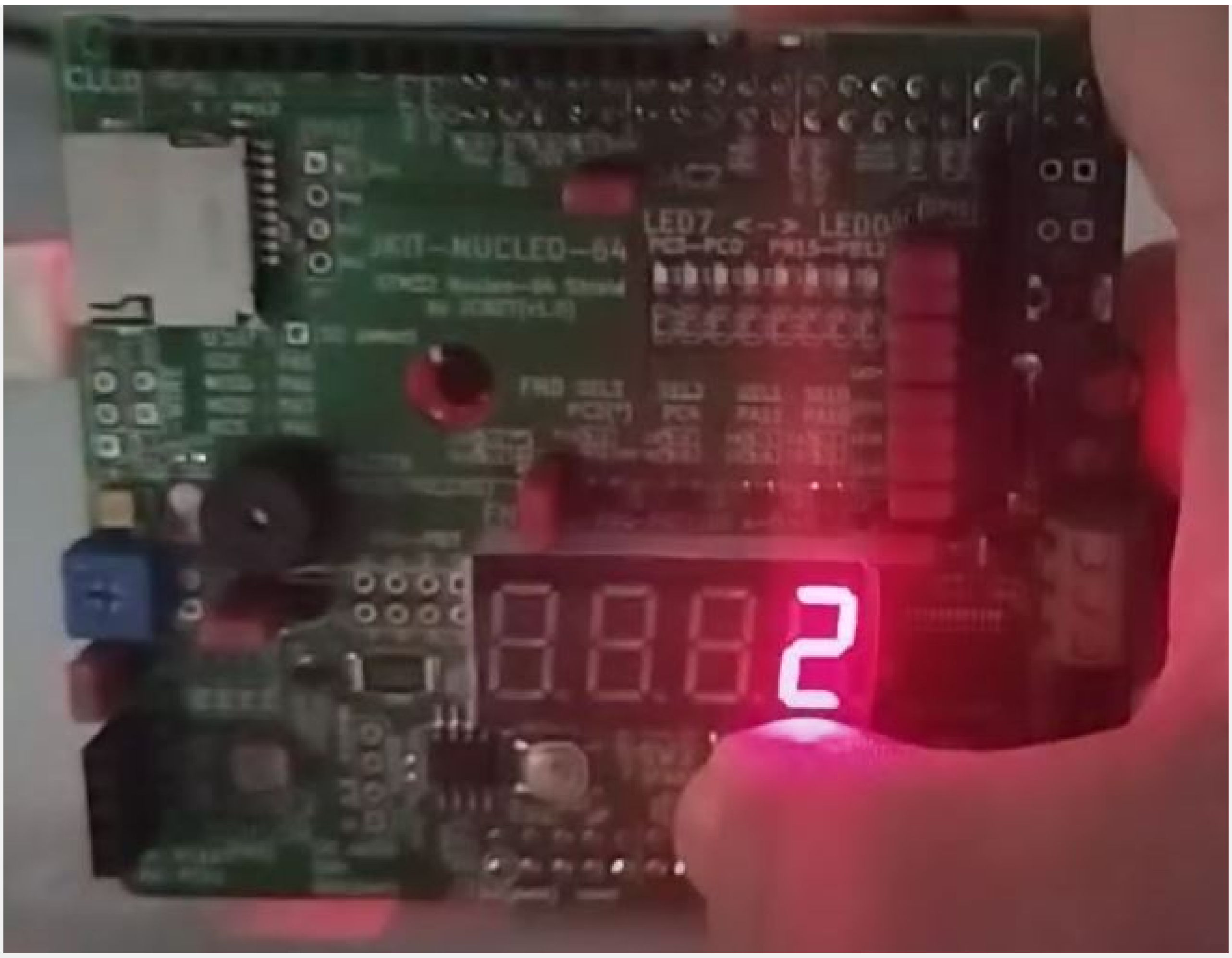
```c
#include "ecSTM32F4v2.h"

#define BUTTON_PIN PA_4

volatile int g_count = 0;          // count should be volatile
volatile uint8_t g_button_flag = 0; // flag that ISR informs main of button input

// 초기화 함수
void setup(void) {
    RCC_PLL_init();
    SysTick_init();
    seven_seg_FND_init(); // 7 seg init

    GPIO_init(BUTTON_PIN, INPUT);
    GPIO_ospeed(BUTTON_PIN, 01); // medium speed
    GPIO_otype(BUTTON_PIN, 0); // Push-Pull
    GPIO_pupd(BUTTON_PIN, 0); // no pull-up, pull-down

    // exti init falling edge trigger, priority 0
    EXTI_init(BUTTON_PIN, FALL, 0);
}

int main(void) {
    setup();

    while (1) {
        // 1. display the current count on the 7-segment display
        sevensegment_display_19(g_count);

        // 2. check if the button was pressed (flag set by ISR)
        if (g_button_flag == 1) {
            delay_ms(50); // 50ms delay for debouncing

            // 3. check if the button is still pressed
            if (GPIO_read(BUTTON_PIN) == 0) {
                g_count = (g_count + 1) % 20; // 0~19 cycle
                // 4. wait until the button is released
                while (GPIO_read(BUTTON_PIN) == 0);
            }

            // 5. clear the button flag
            g_button_flag = 0;
        }
    }
}

// pa4 is extu4, so isr name is EXTI4_IRQHandler
void EXTI4_IRQHandler(void) {
    // 1. check if the interrupt is from the correct pin
    if (is_pending_EXTI(BUTTON_PIN)) {
        // 2. set the button flag to indicate a button press
        g_button_flag = 1;

        // 3. clear the pending bit for the EXTI line
        clear_pending_EXTI(BUTTON_PIN);
    }
}
```

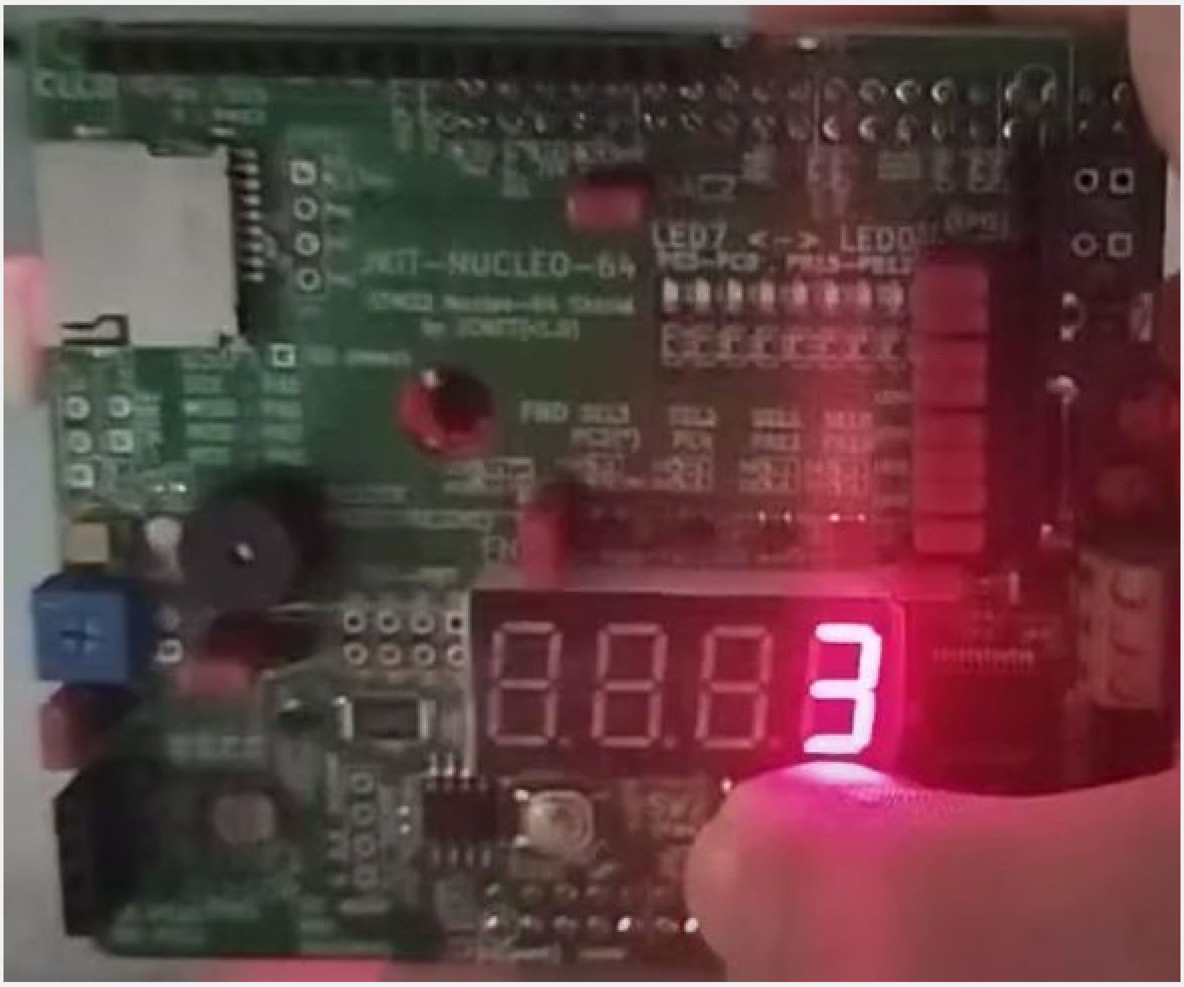
Results

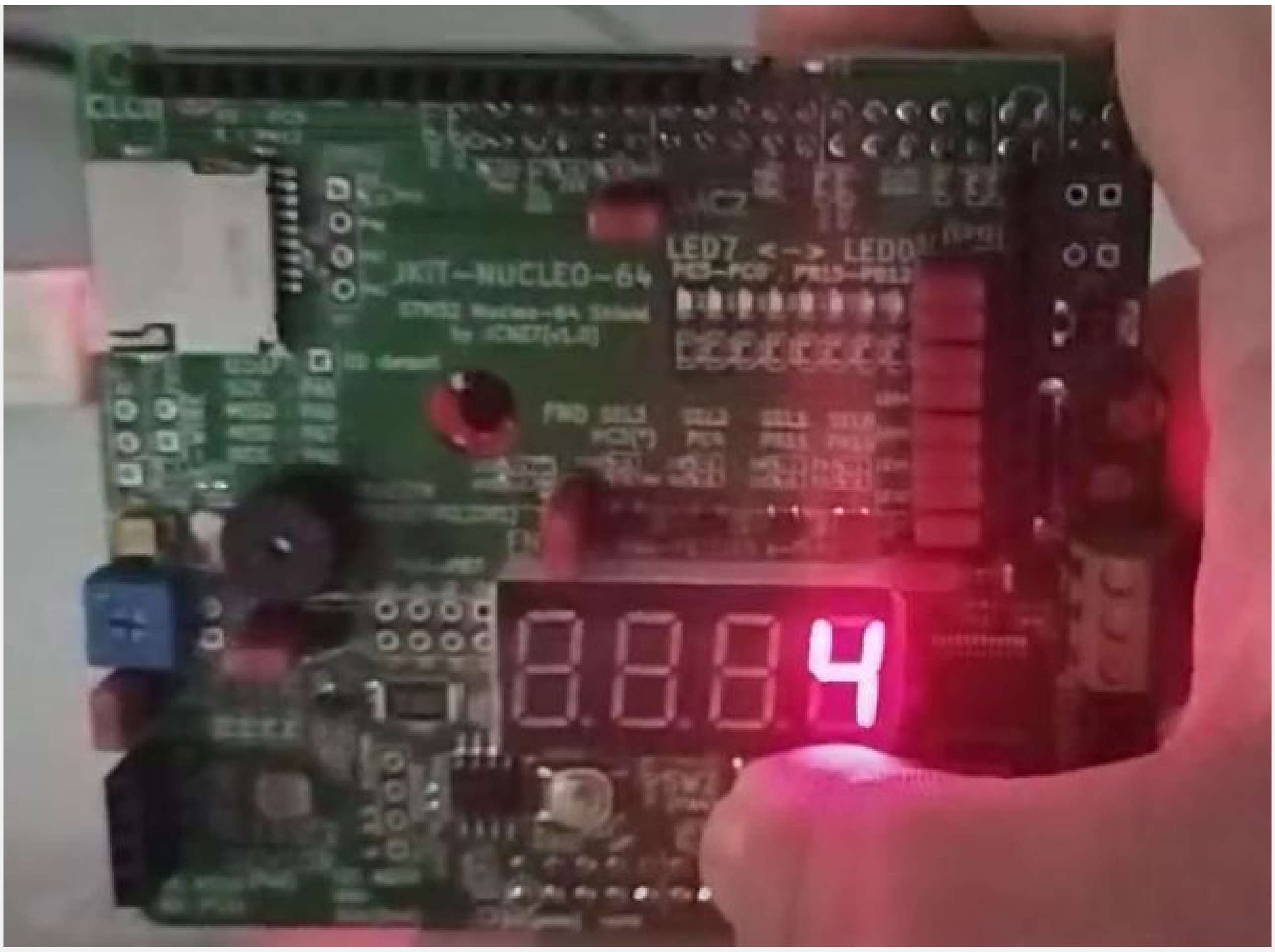
Experiment images and results go here

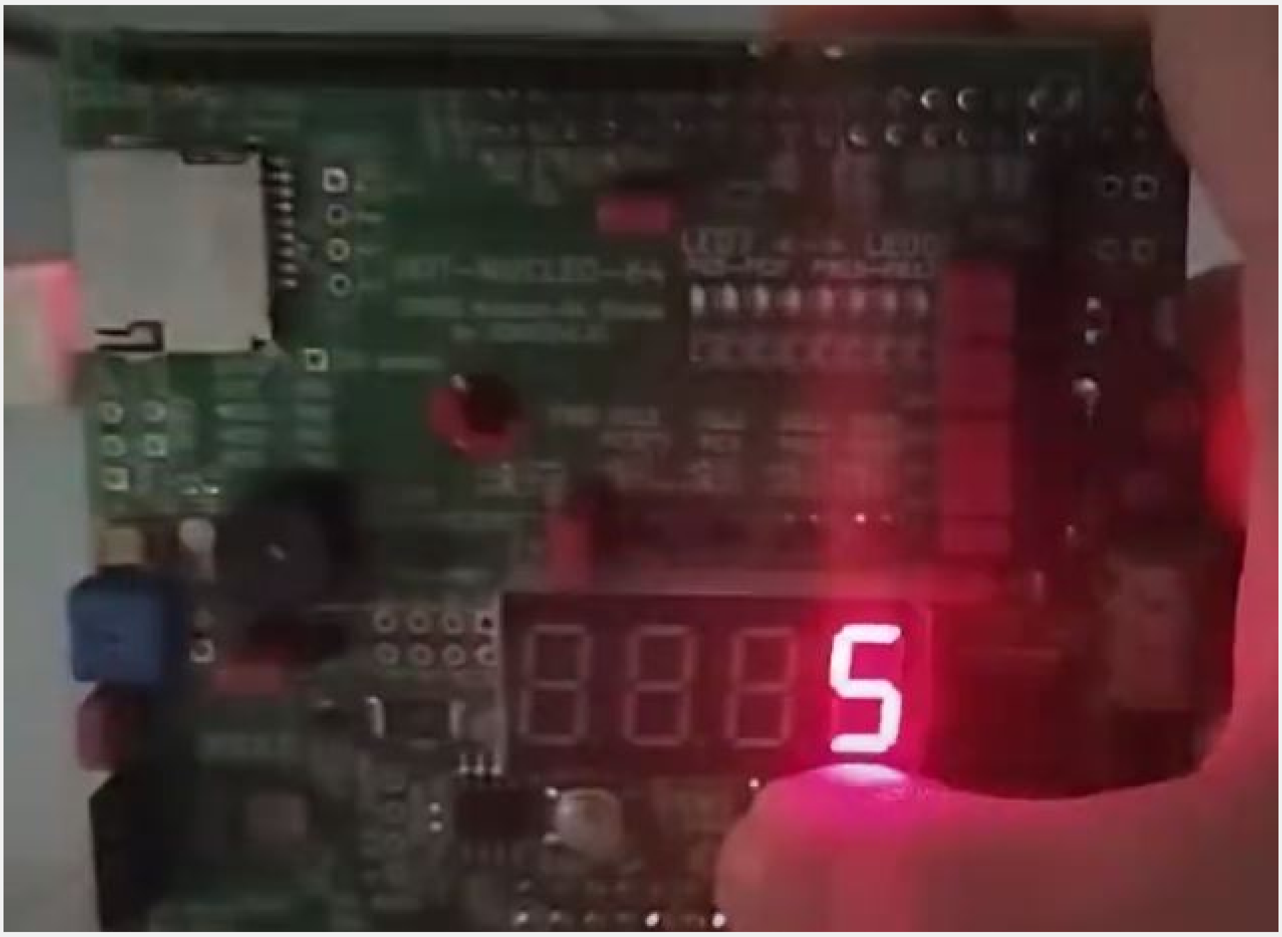

JKIT-NUCLEO-64
LED7 <-> LED0
FND SEL3
8.8.8.0

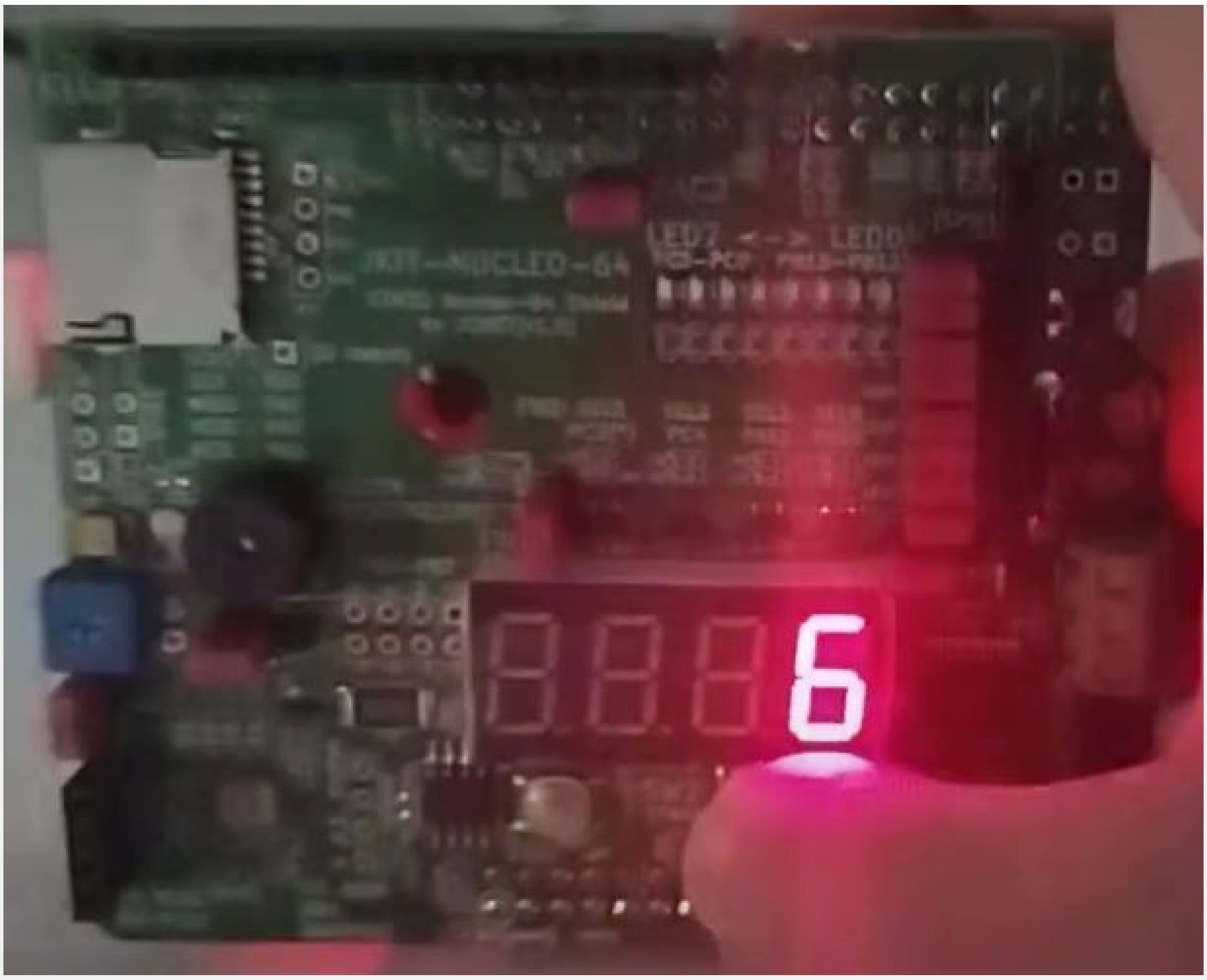

8.8.8.1

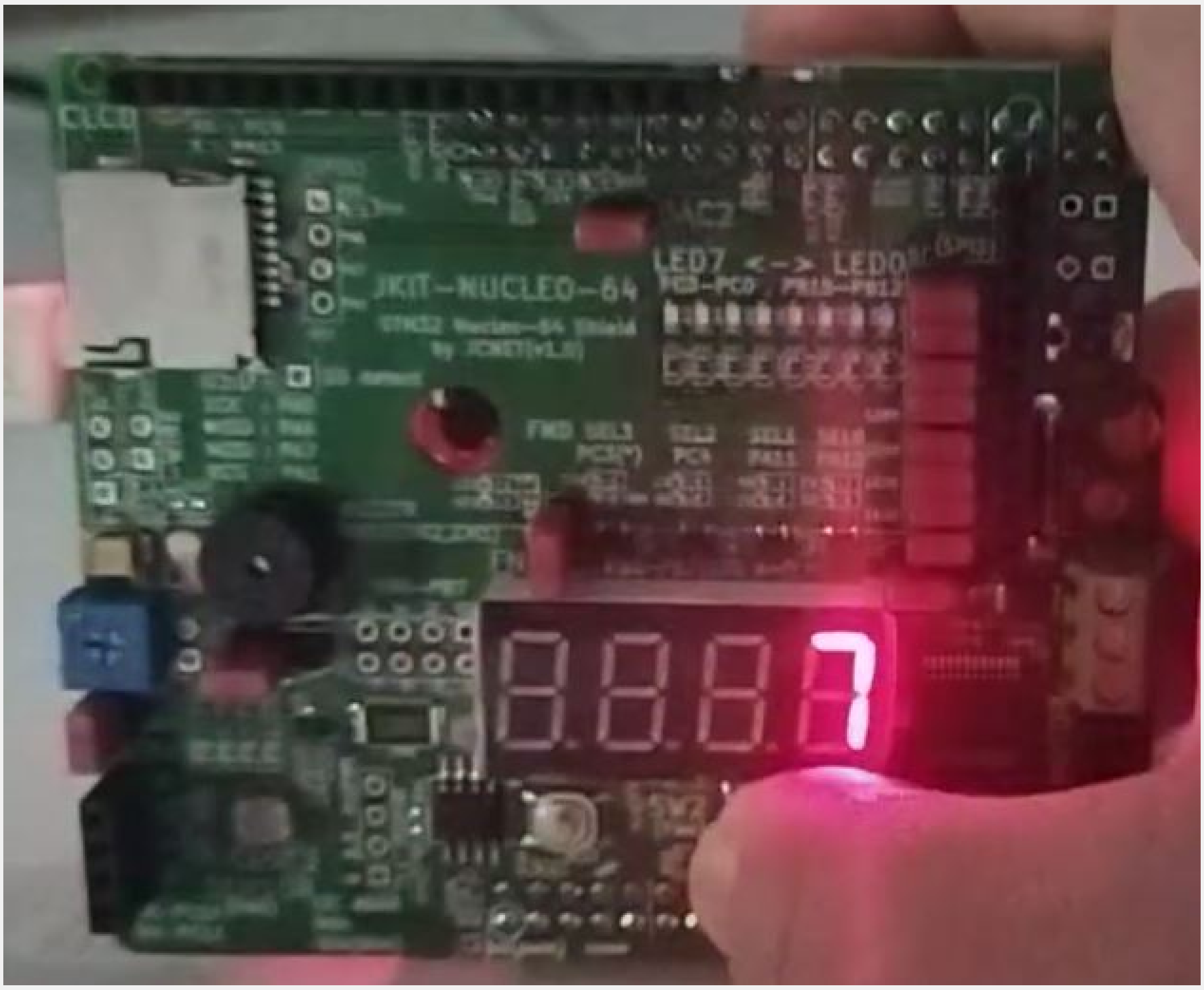

JKIT-NUCLEO-64
LED7 <-> LED0

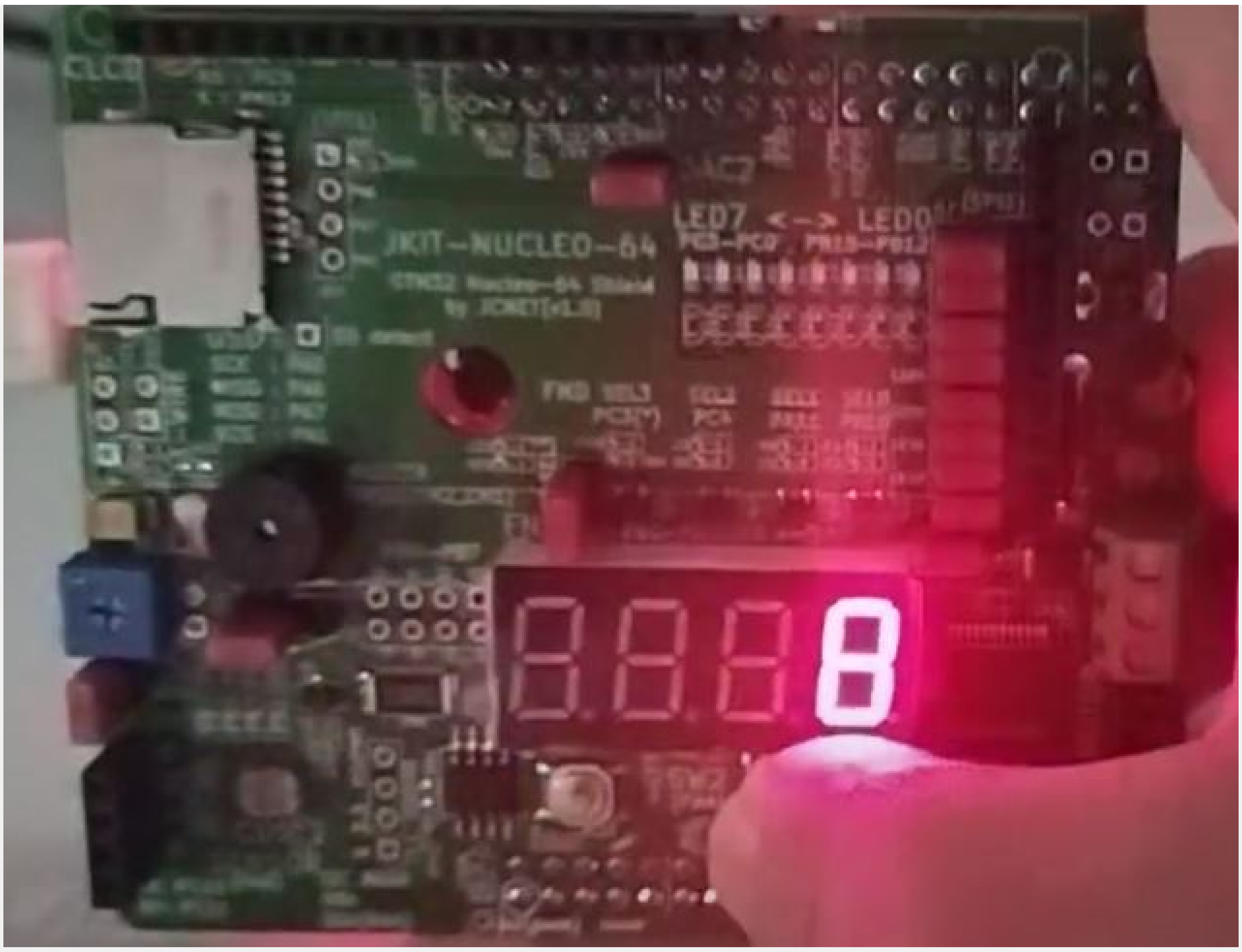

JKIT-NUCLEO-64
LED7 <-> LED0

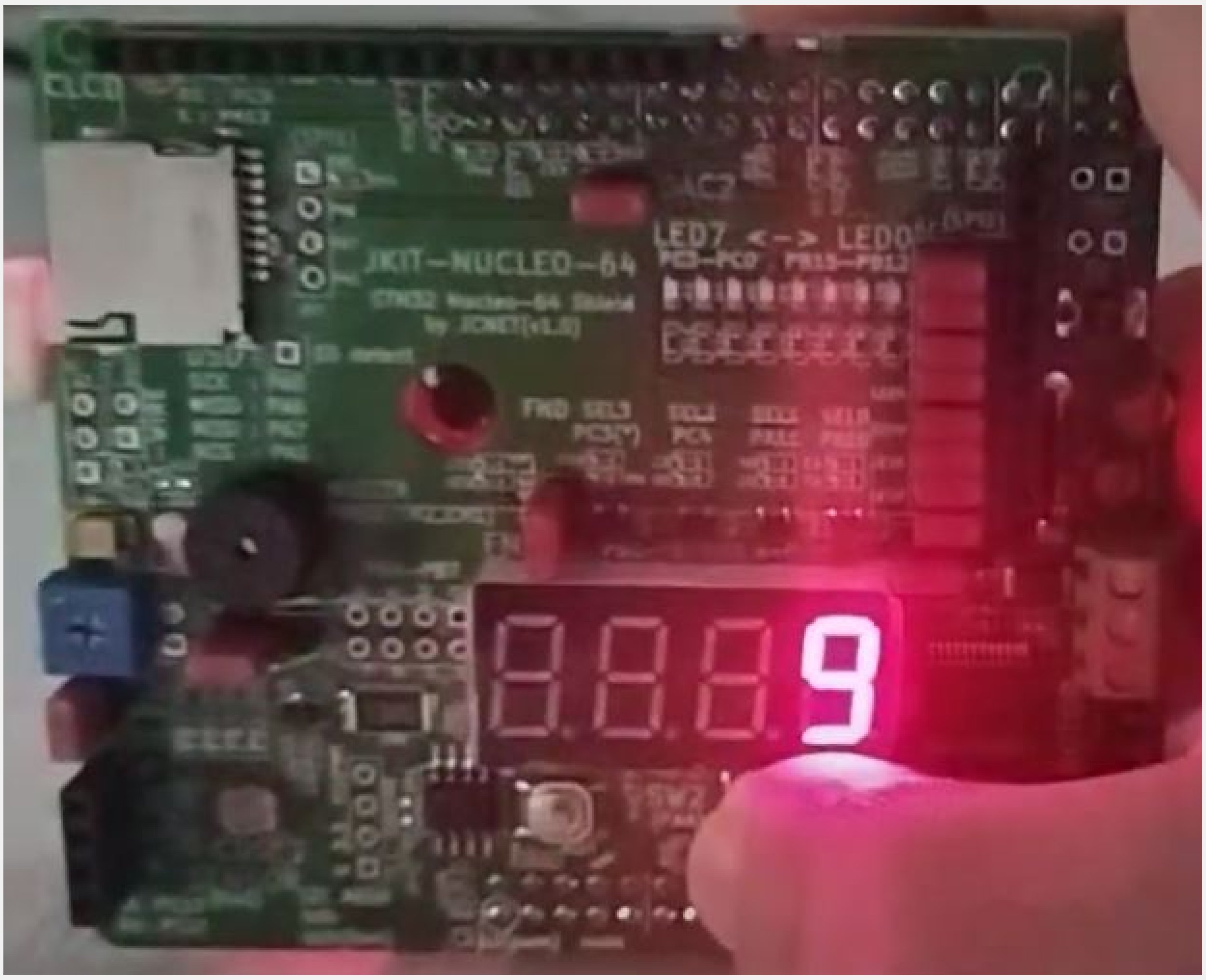

JKIT-NUCLEO-64
LED7 <-> LED0
4

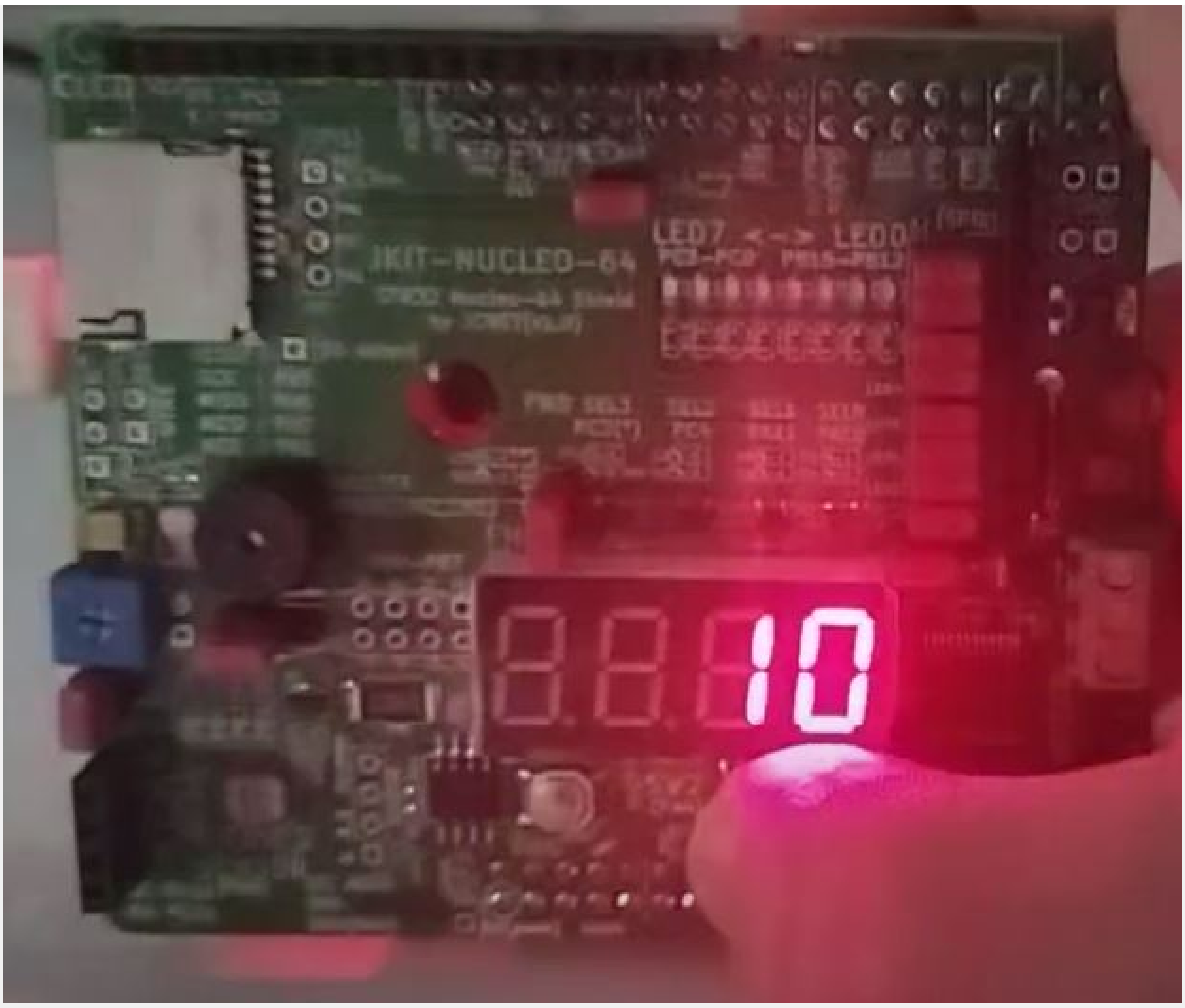

5

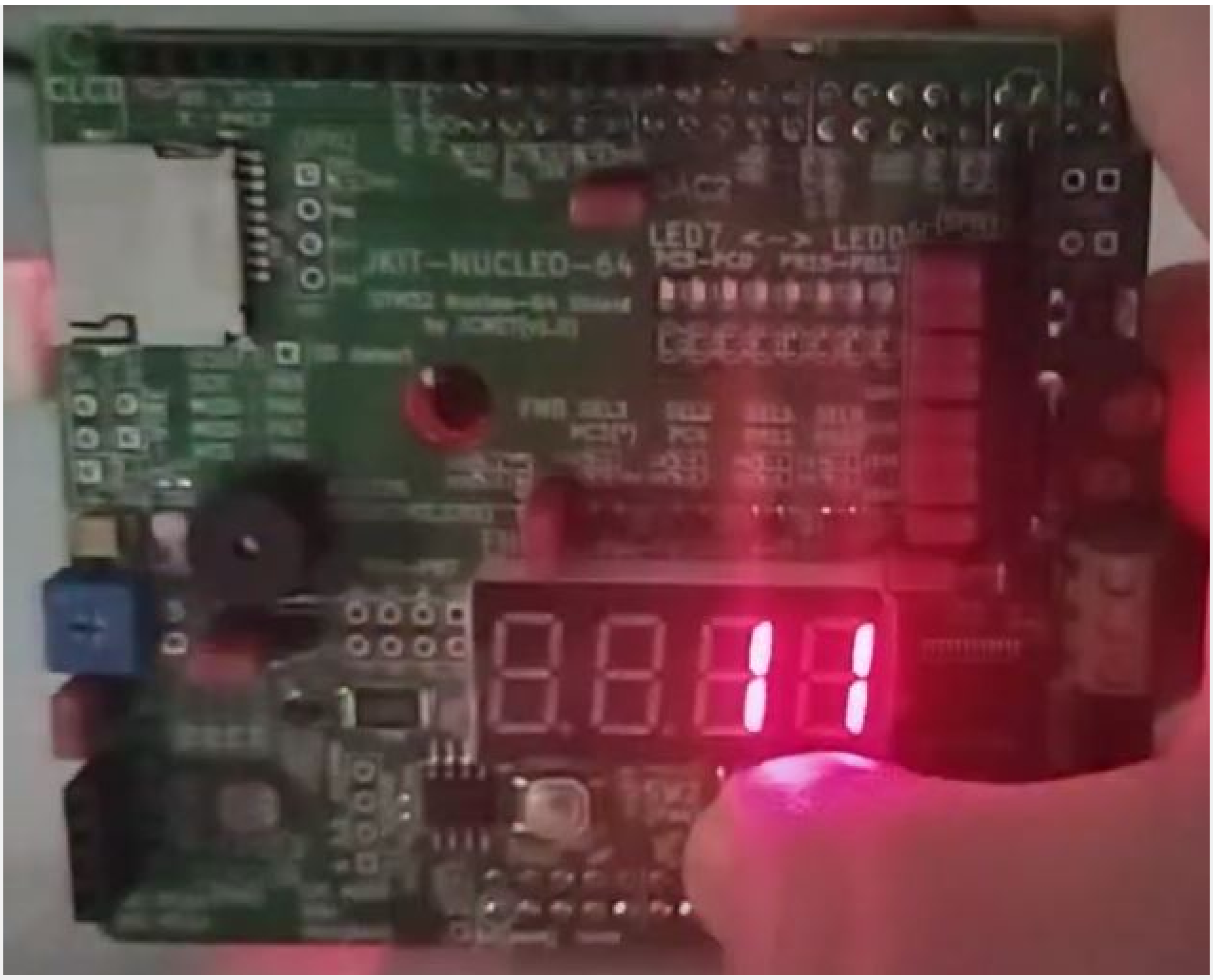

JKIT-NUCLEO-64
LED7 <-> LED0

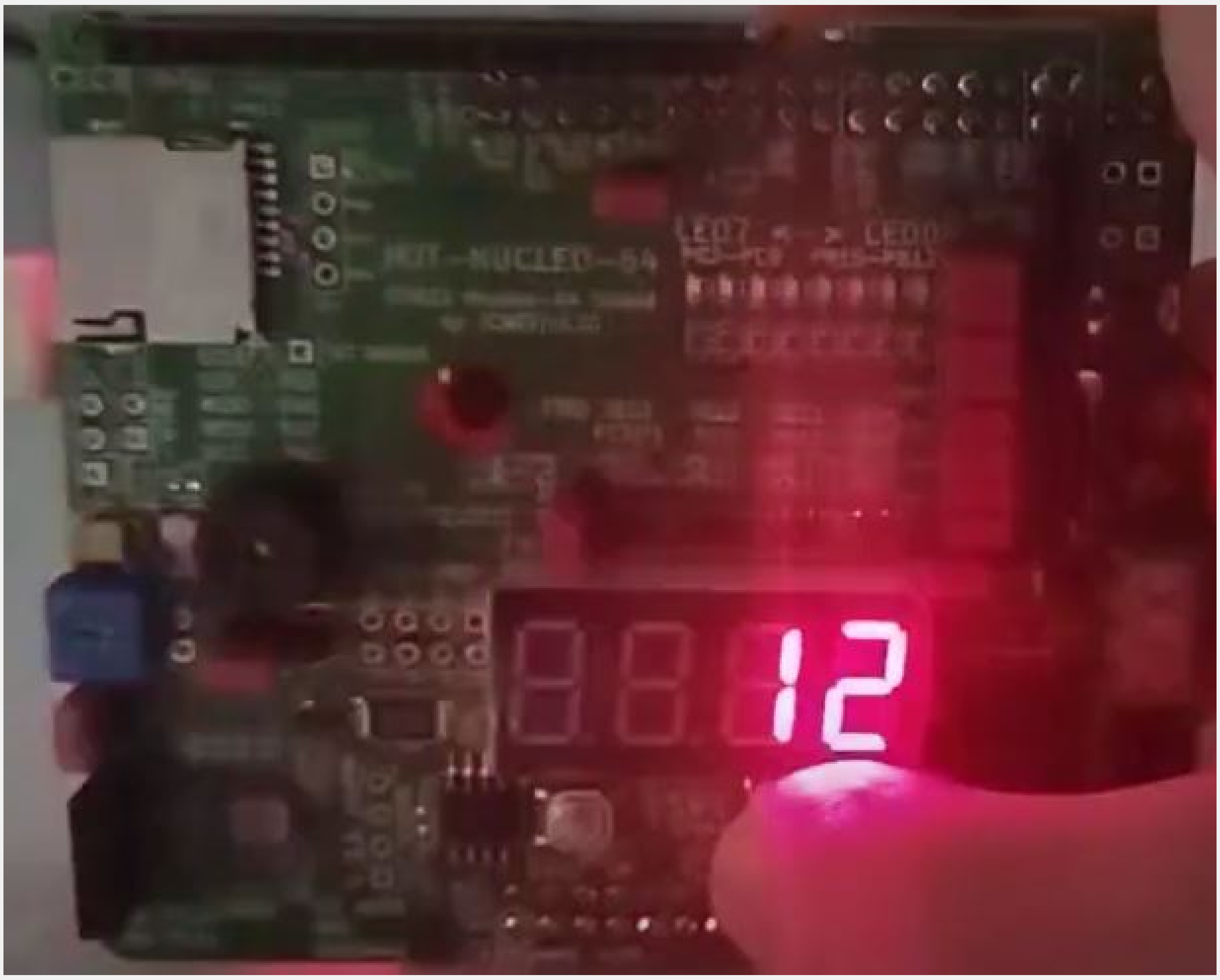

JKIT-NUCLEO-64
LED7 <-> LED0

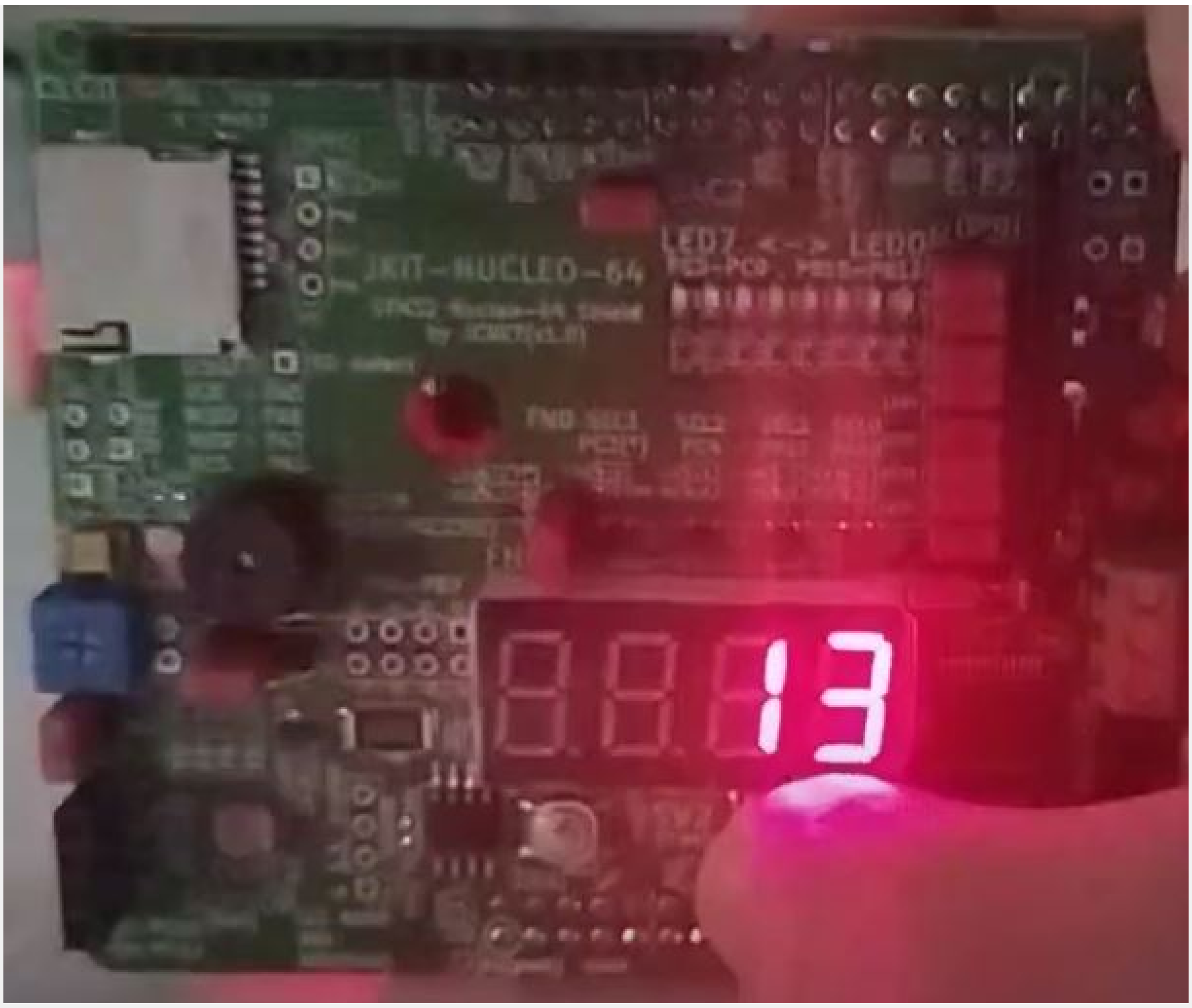

JKIT-NUCLEO-64
LED7 <-> LED0

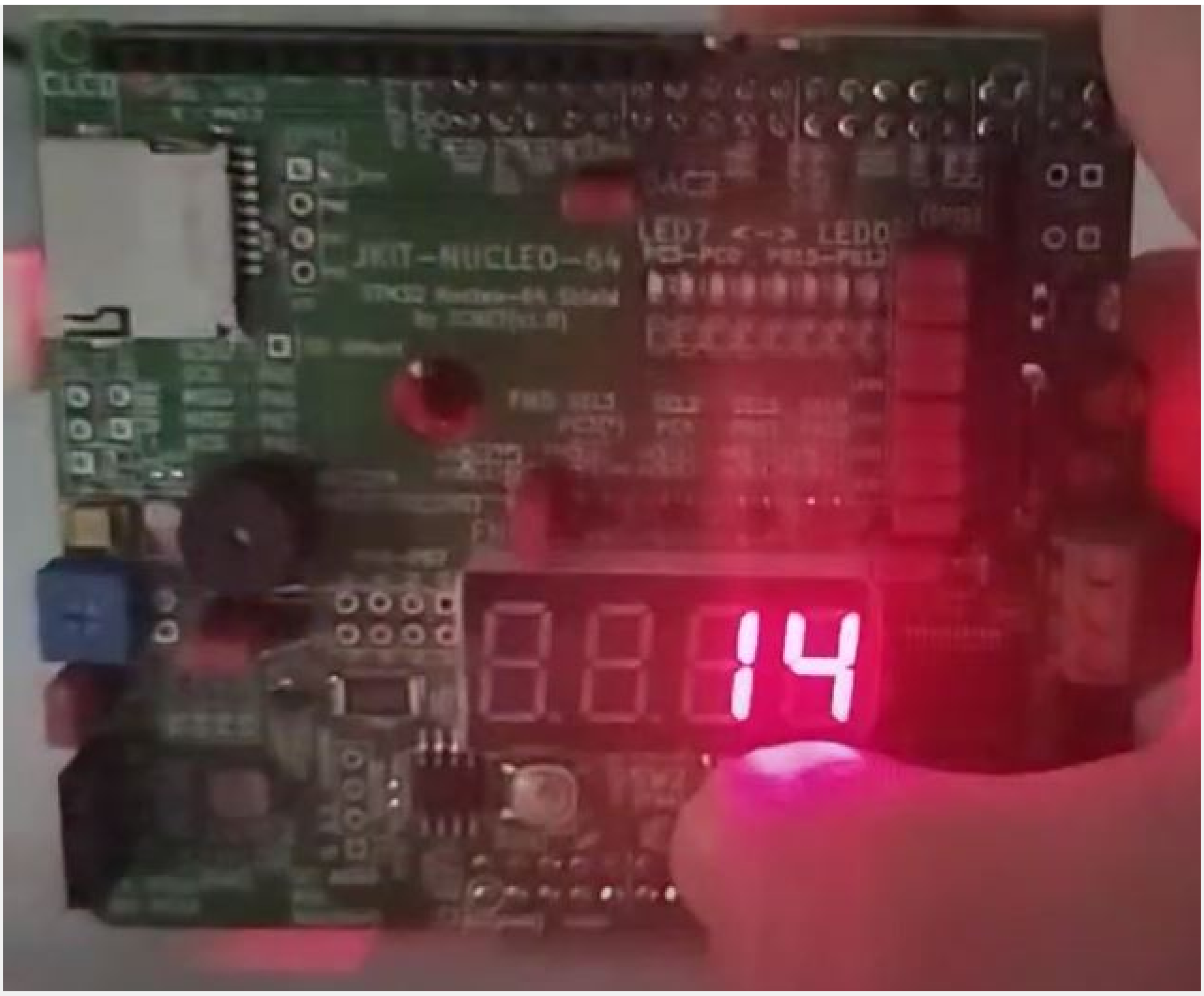

JKIT-NUCLEO-64
LED7 <-> LED0

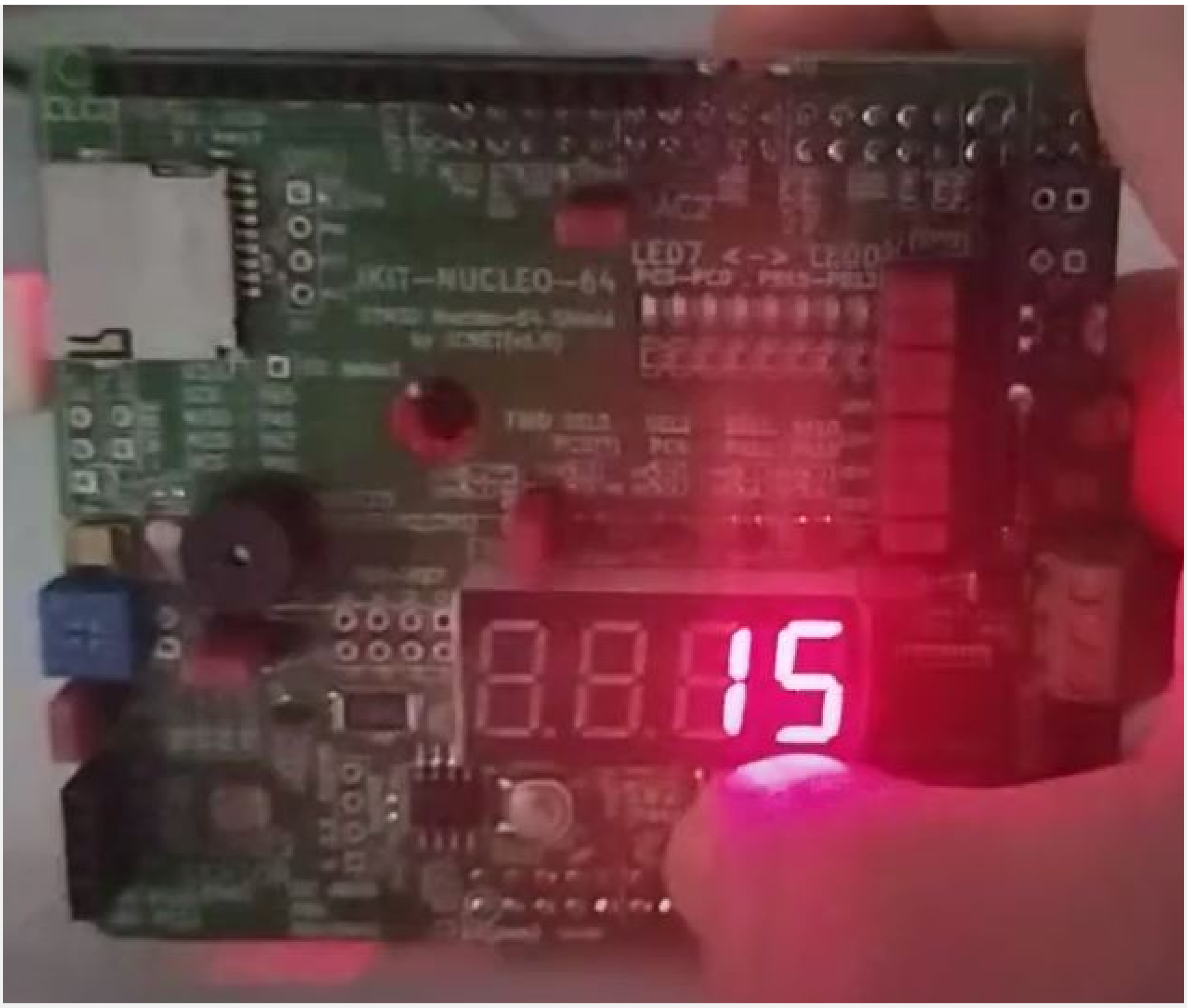

JKIT-NUCLEO-64
LED7 <-> LED0
10

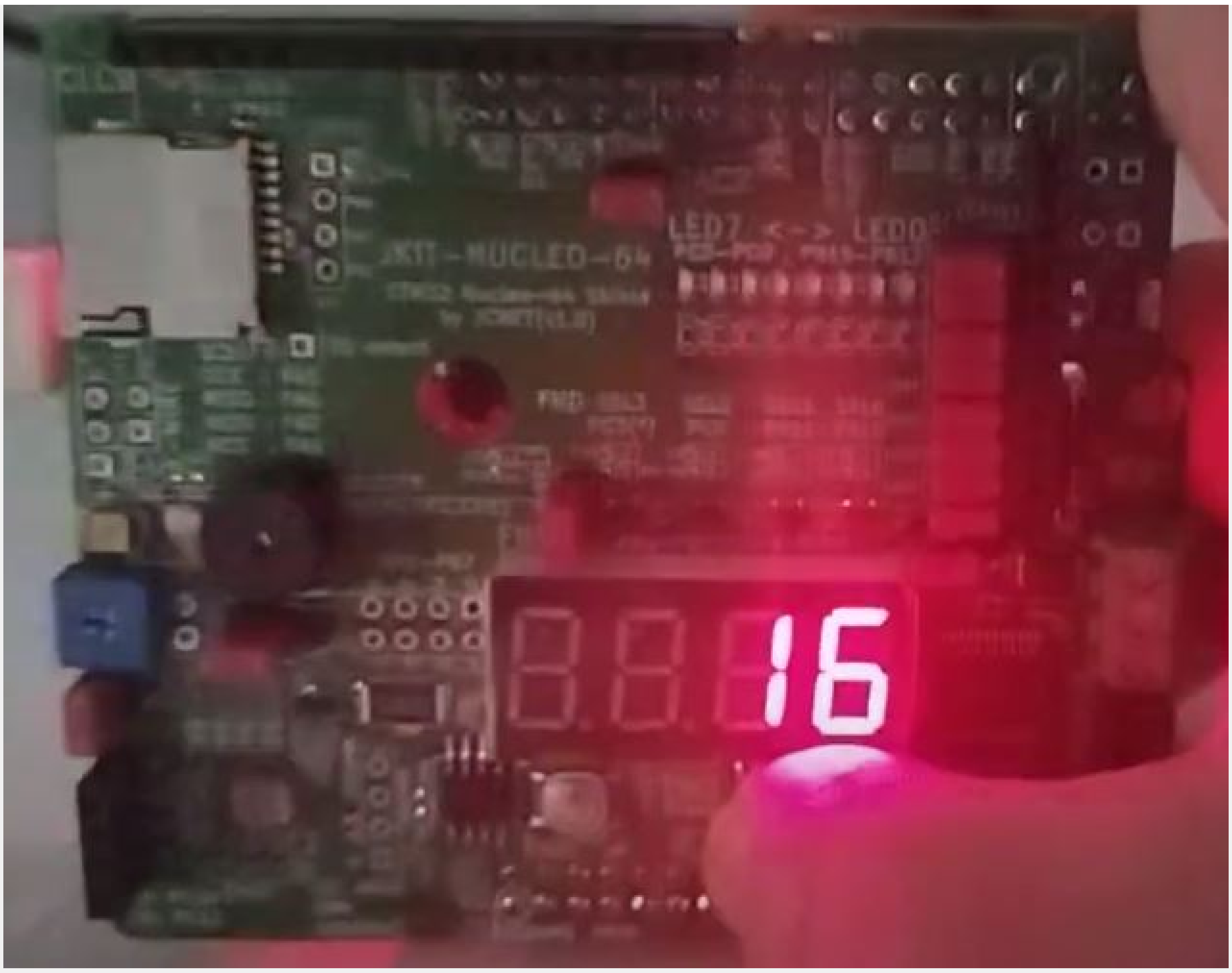

JKIT-NUCLEO-64
LED7 <-> LED0
11

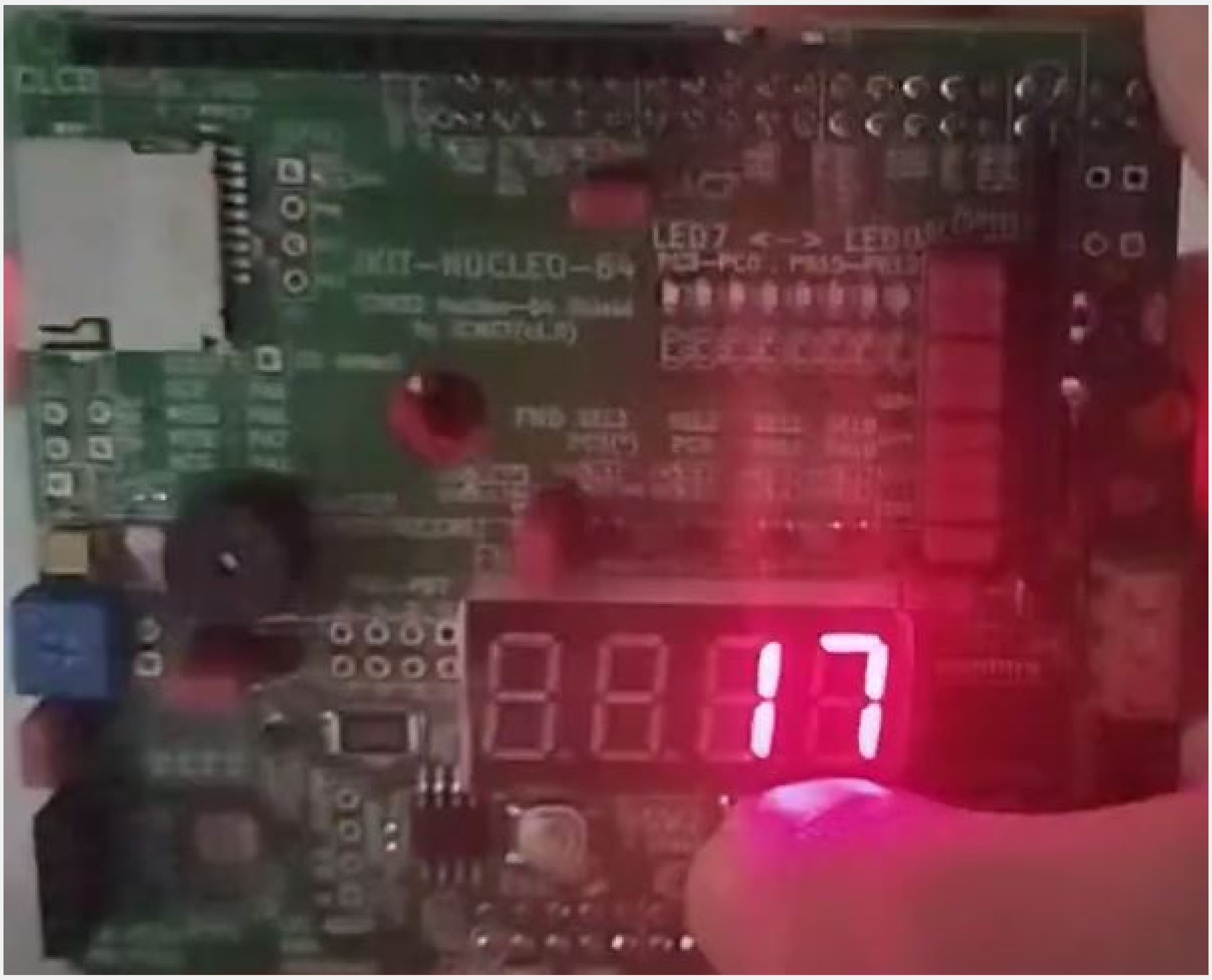

JKIT-NUCLEO-64
LED7 <-> LED0
12

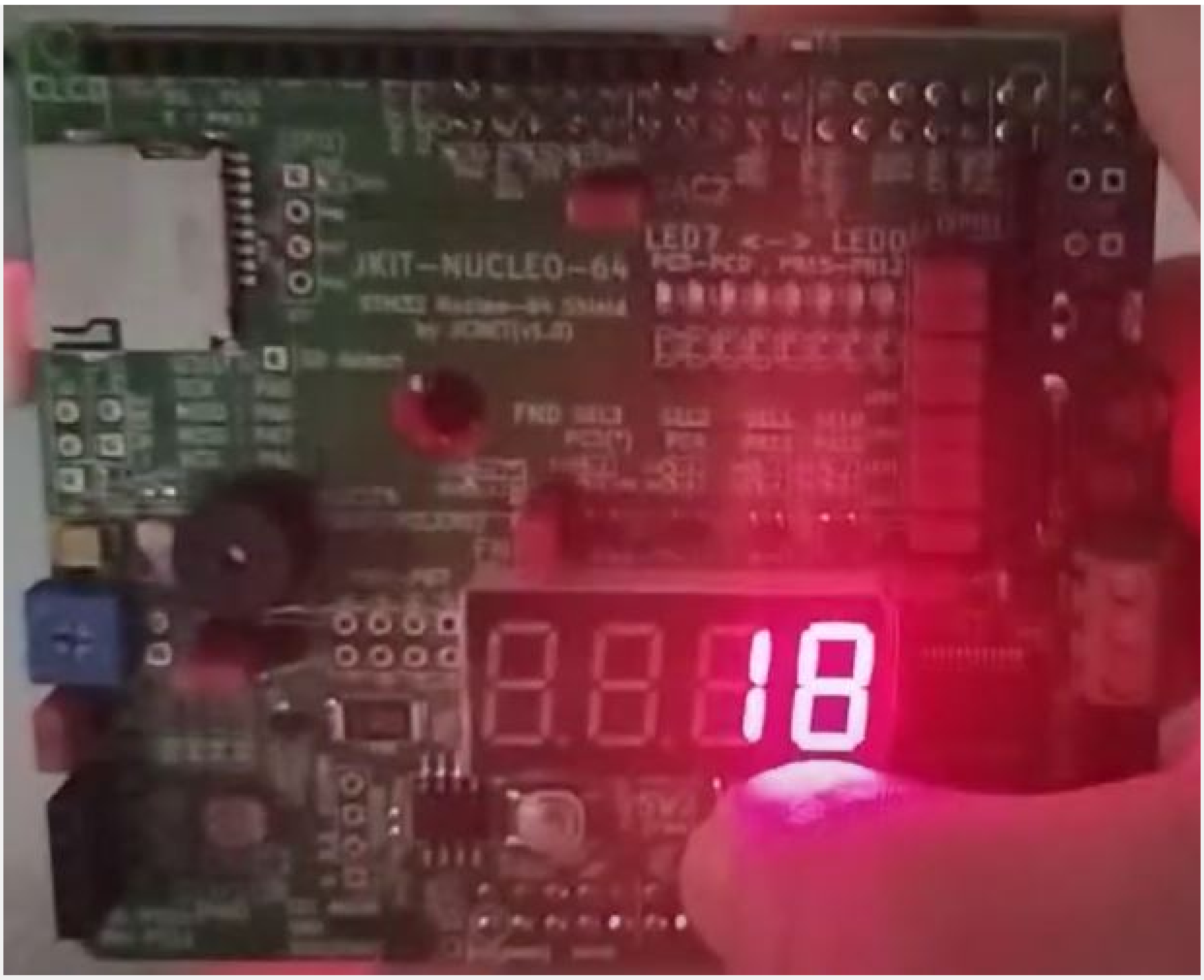

JKIT-NUCLEO-64
LED7 <-> LED0
13

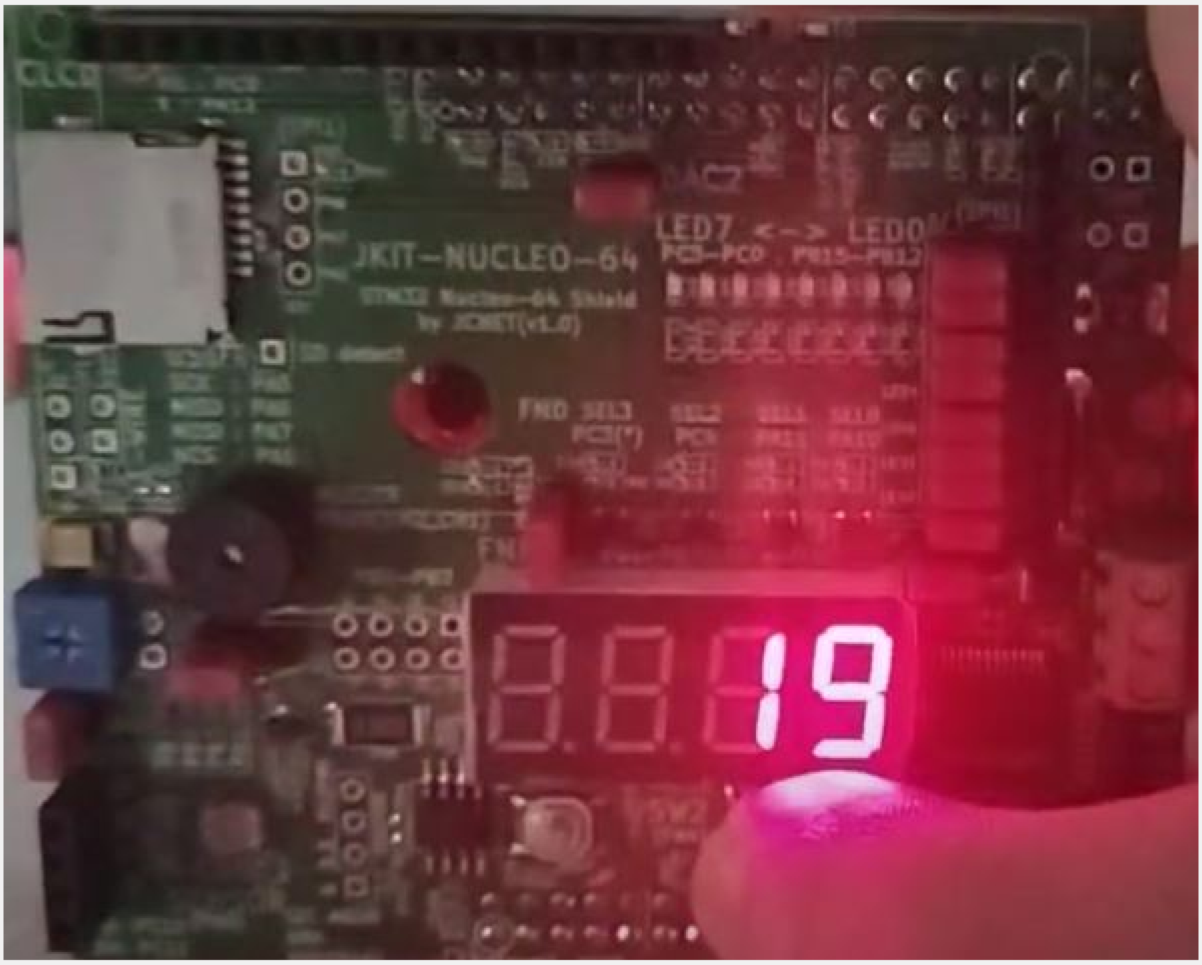

JKIT-NUCLEO-64
LED7 <-> LED0
14


JKIT-NUCLEO-64
LED7 <-> LED0
15


JKIT-NUCLEO-64
LED7 <-> LED0
16


JKIT-NUCLEO-64
LED7 <-> LED0
17





Add demo video link

Problem 2: Counting numbers on 7-Segment using SysTick

Display the number 0 to 9 on the 7-segment LED at the rate of 1 sec.

After displaying up to 9, then it should display ‘0’ and continue counting.

When the button is pressed, the number should be reset ‘0’ and start counting again.

SysTick Library

1. Download sample header files: **ecSysTick_student.h, ecSysTick_student.c**
2. Rename these files as **ecSysTick2.h, ecSysTick2.c**
   - You MUST write your name and other information at the top of the library code files.
   - Save these files in your directory `EC \include\`.
3. Declare and define the following functions in your library : **ecSysTick2.h**

ecSysTick.h

```
void SysTick_init(uint32_t msec);
void delay_ms(uint32_t msec);
uint32_t SysTick_val(void);
void SysTick_reset (void);
void SysTick_enable(void);
void SysTick_disable (void)
```

2-2. Procedure

1. Create a new project under the directory

   `\EC\lab\LAB_EXTI_SysTick`

- The project name is “**LAB_EXTI_SysTick”.**
- Create a new source file named as “**LAB_EXTI_SysTick.c”**

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

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

- **ecGPIO2.h, ecGPIO2.c**
- **ecRCC2.h, ecRCC2.c**
- **ecEXTI2.h, ecEXTI2.c**
- **ecSysTick2.h, ecSysTick2.c**

3. First, check if every number, 0 to 9, can be displayed properly on the 7-segment.

4. Then, create a code to display the number counting from 0 to 9 and repeat at the rate of 1 second. (Use only one digit)

5. When the button is pressed, it should start from '0' again.

   > Use EXTI for this button reset.

Configuration

Digital In for Button (B1)	Digital Out for FND-7-Segment
Digital In	Digital Out
PA4	PB7,PB6,PB5,PB4,PB3,PB2,PB1,PB0 ('a'~'h', respectively) PA10 ('LED4', respectively)
PULL-UP	Push-Pull, No Pull-up-Pull-down, Medium Speed

Code

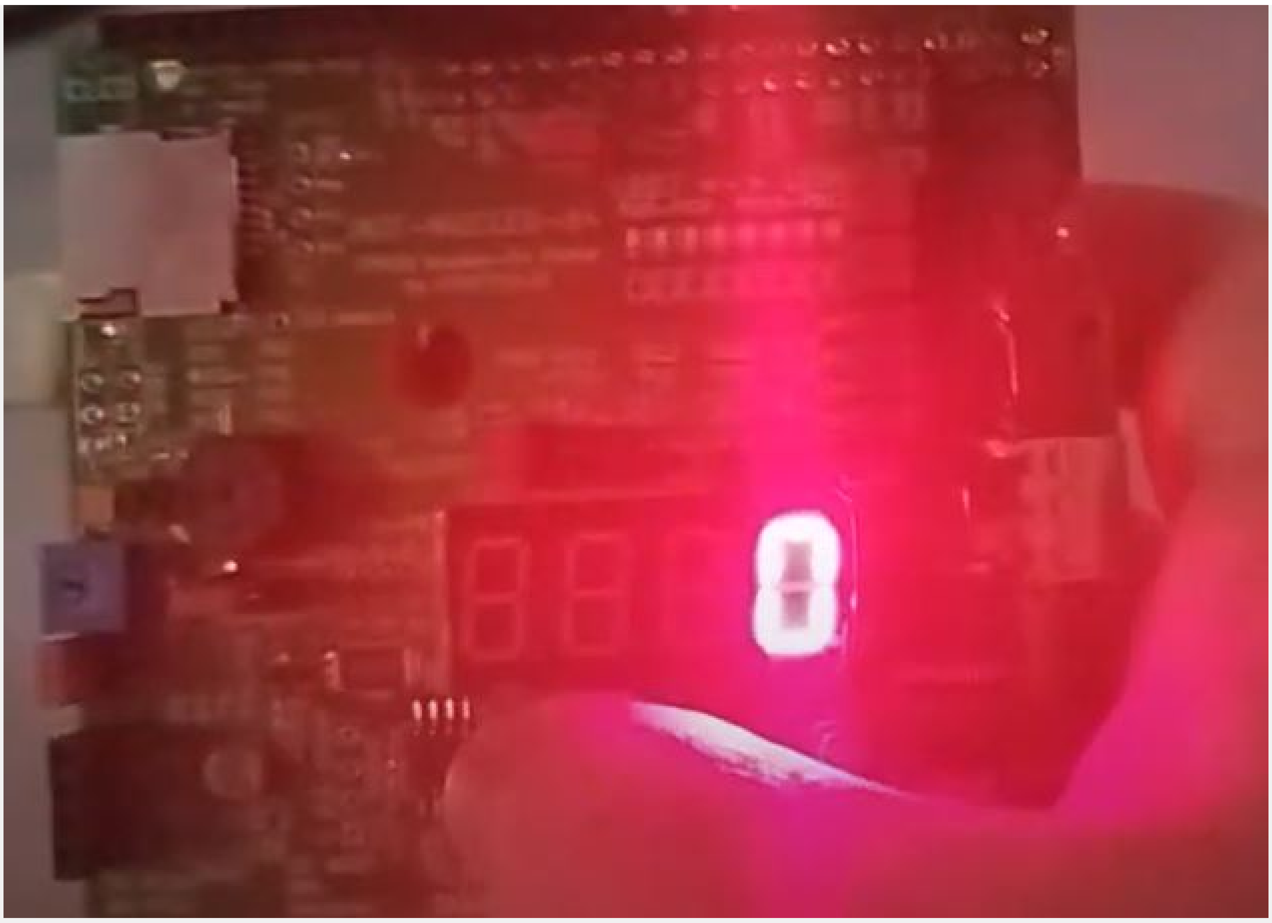
Your code goes here.

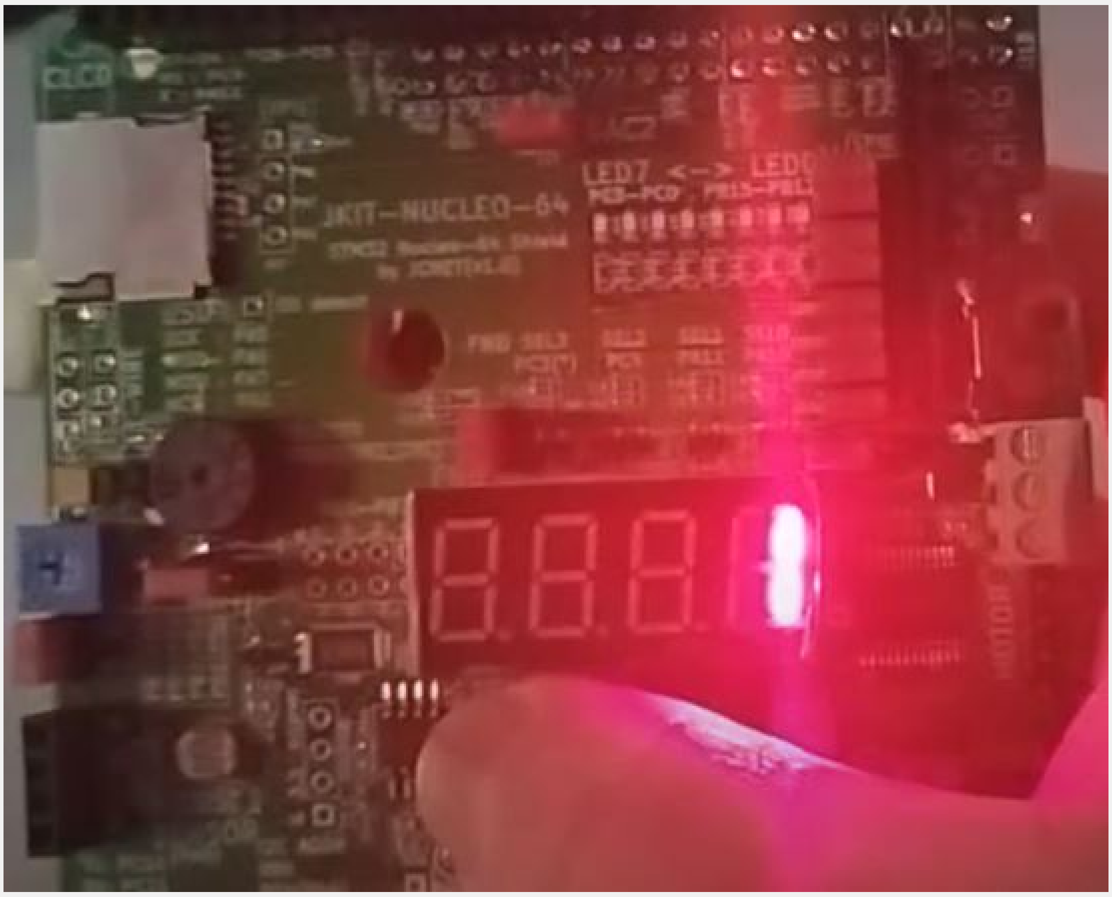
Explain your source code with necessary comments.

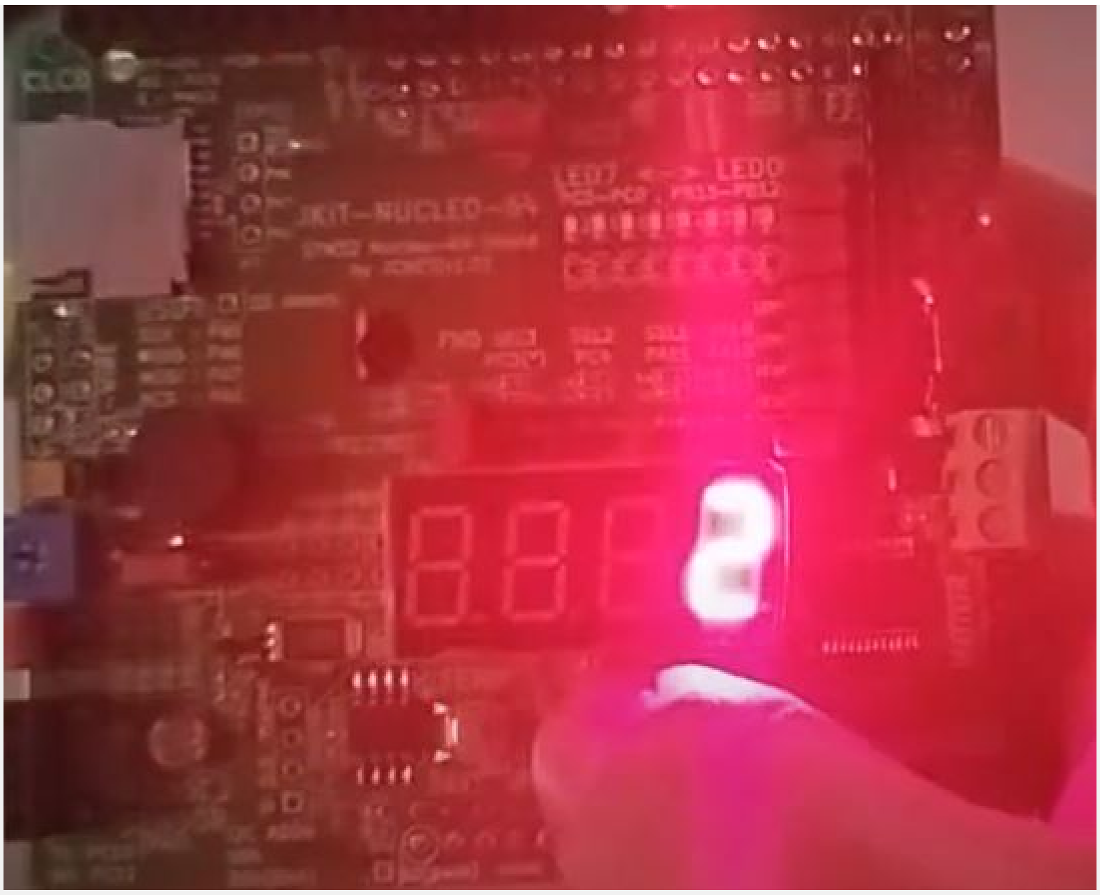
```c
#include "ecSTM32F4v2.h"

volatile int g_count = 0;         // Current digit to display
volatile uint8_t g_reset_flag = 0; // Set by EXTI ISR when button pressed
#define BUTTON_PIN PA_4

void setup(void)
{
    RCC_PLL_init();
    SysTick_init();
    seven_seg_FND_init();
    // Button pin EXTI setup (assuming BUTTON_PIN is defined as PA4)
    GPIO_init(BUTTON_PIN, INPUT);
    GPIO_ospeed(BUTTON_PIN, 01); // medium speed
    GPIO_otype(BUTTON_PIN, 0); // Push-Pull
    GPIO_pupd(BUTTON_PIN, 0); // no pull-up, pull-down
    // exti init falling edge trigger, priority 0
    EXTI_init(BUTTON_PIN, FALL, 0);

}

int main(void) {
    setup();
    while (1) {
         // 1. display the current count on the 7-segment display
        seven_seg_FND_display(g_count, 0);

        // 2. delay for 1 second
        delay_ms(1000);

        // 3. increment the count
        g_count++;
        // 4. 0~9 cycle
        if (g_count > 9) {
            g_count = 0;
        }

        // 5. if reset flag is set by EXTI ISR, reset count
        if (g_reset_flag == 1) {
            g_count = 0;      // reset count to 0
            g_reset_flag = 0; // reset the flag
        }
}
}


// EXTI4 IRQ Handler for PA4 button
void EXTI4_IRQHandler(void) {
    if (is_pending_EXTI(BUTTON_PIN)) {
        g_reset_flag = 1; // Signal main to reset count
        clear_pending_EXTI(BUTTON_PIN);
    }
}
```

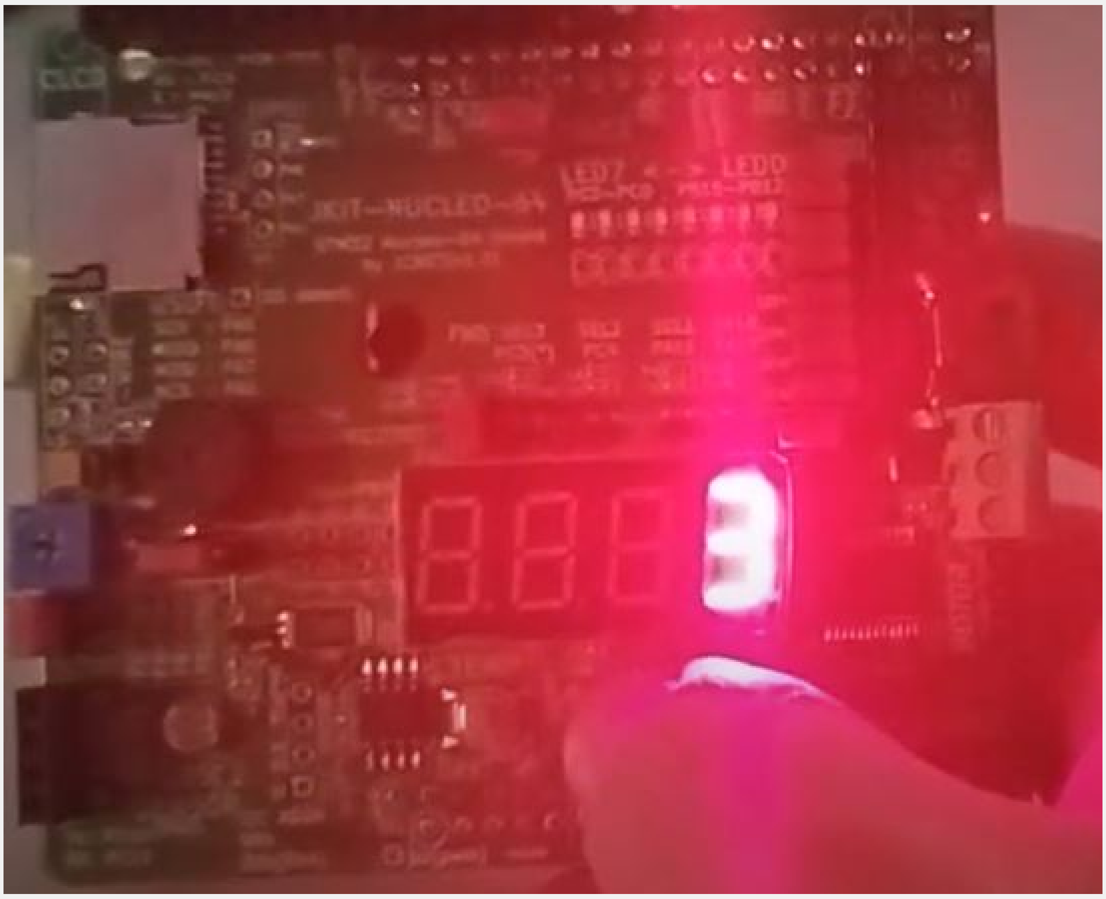
Results

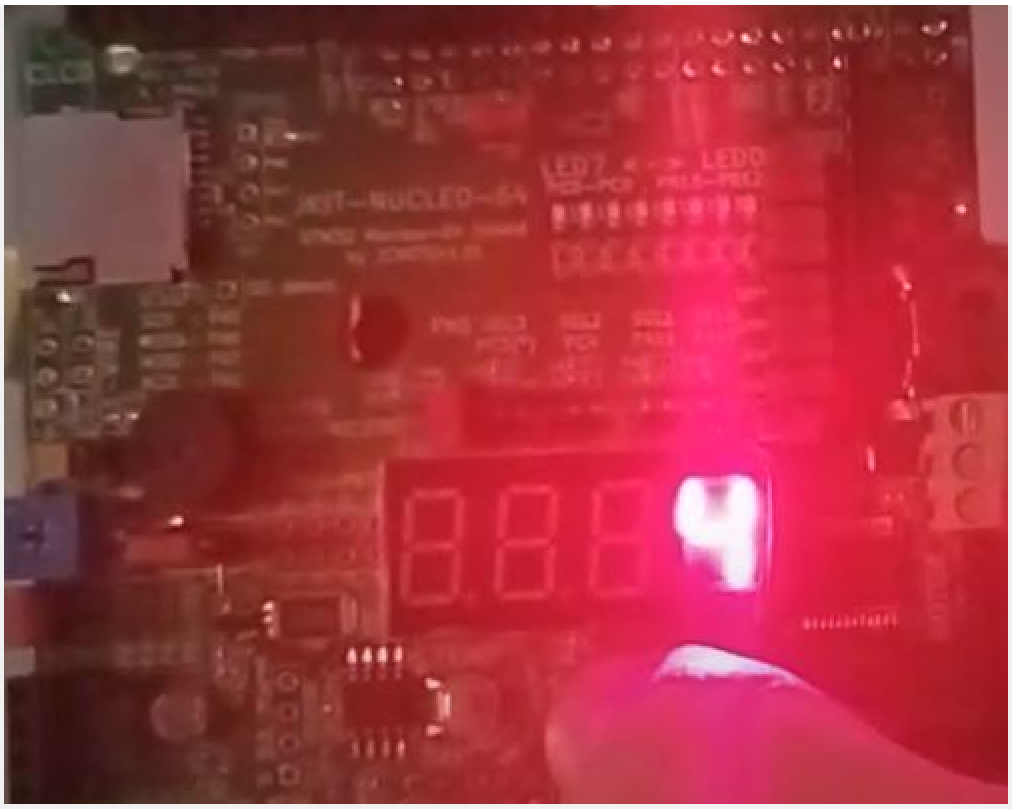
Experiment images and results

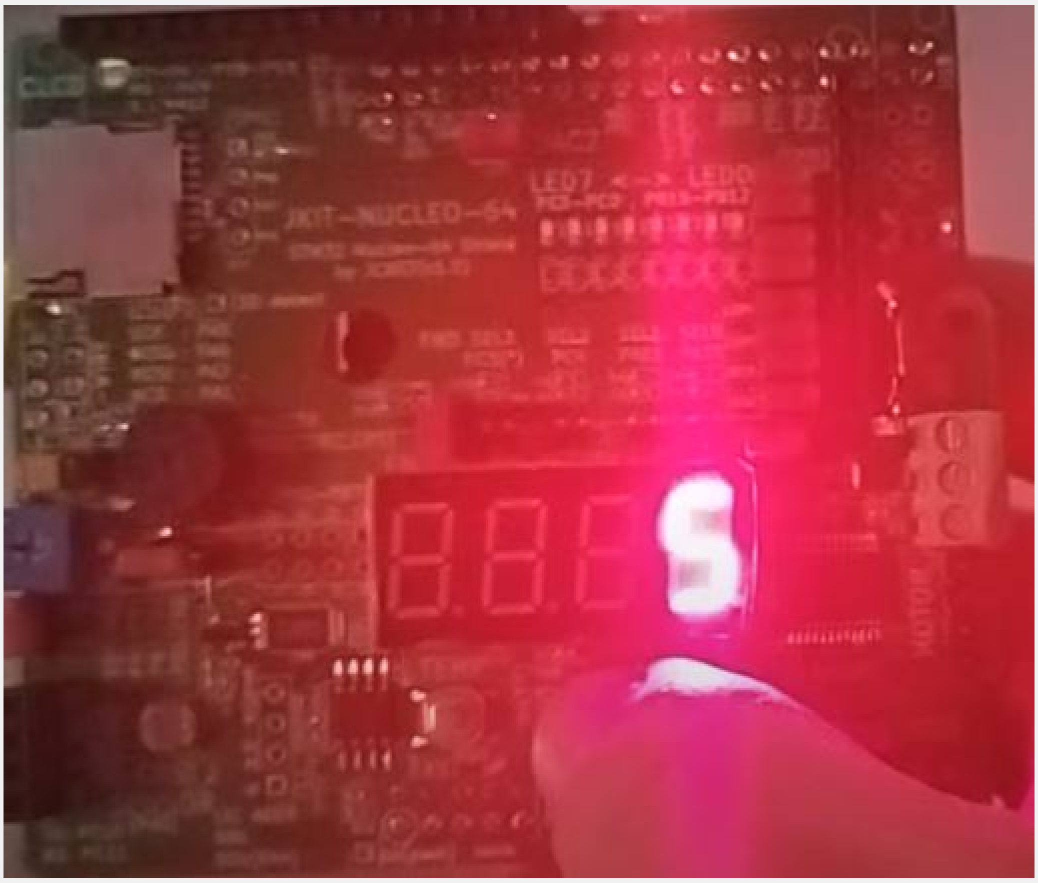


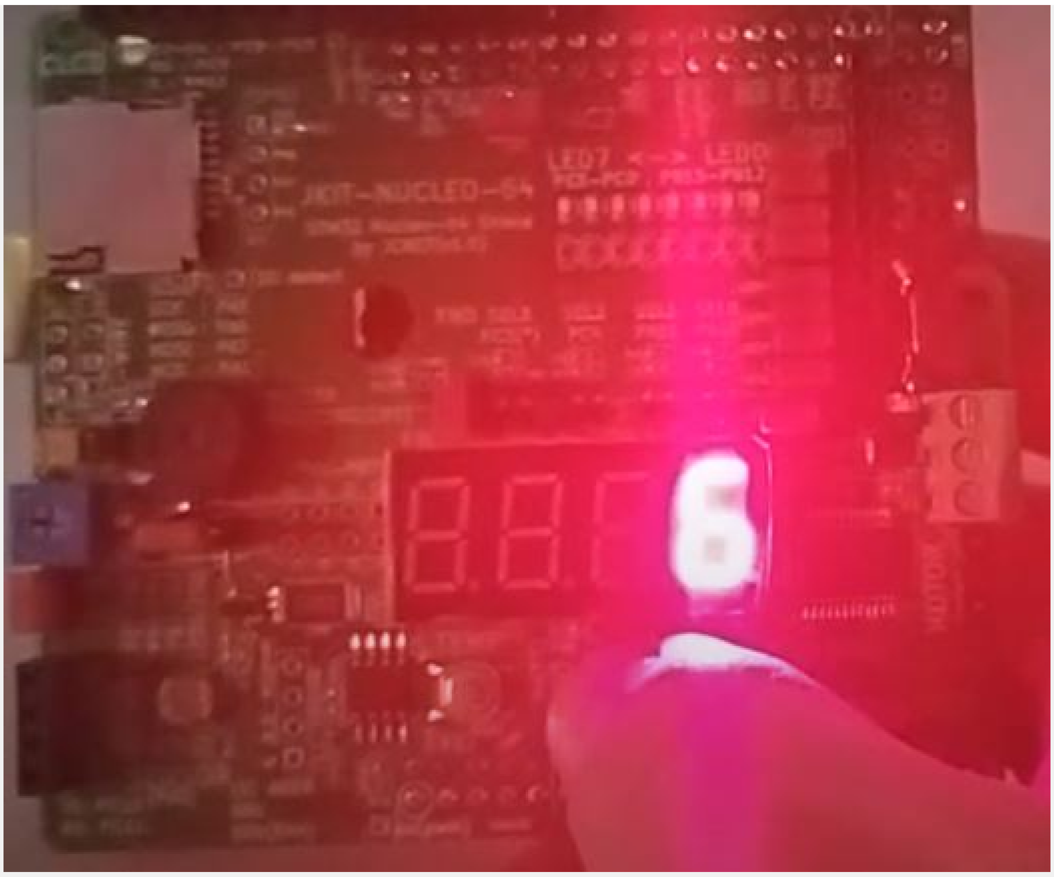

JKIT-NUCLEO-64
LED7 <-> LED0

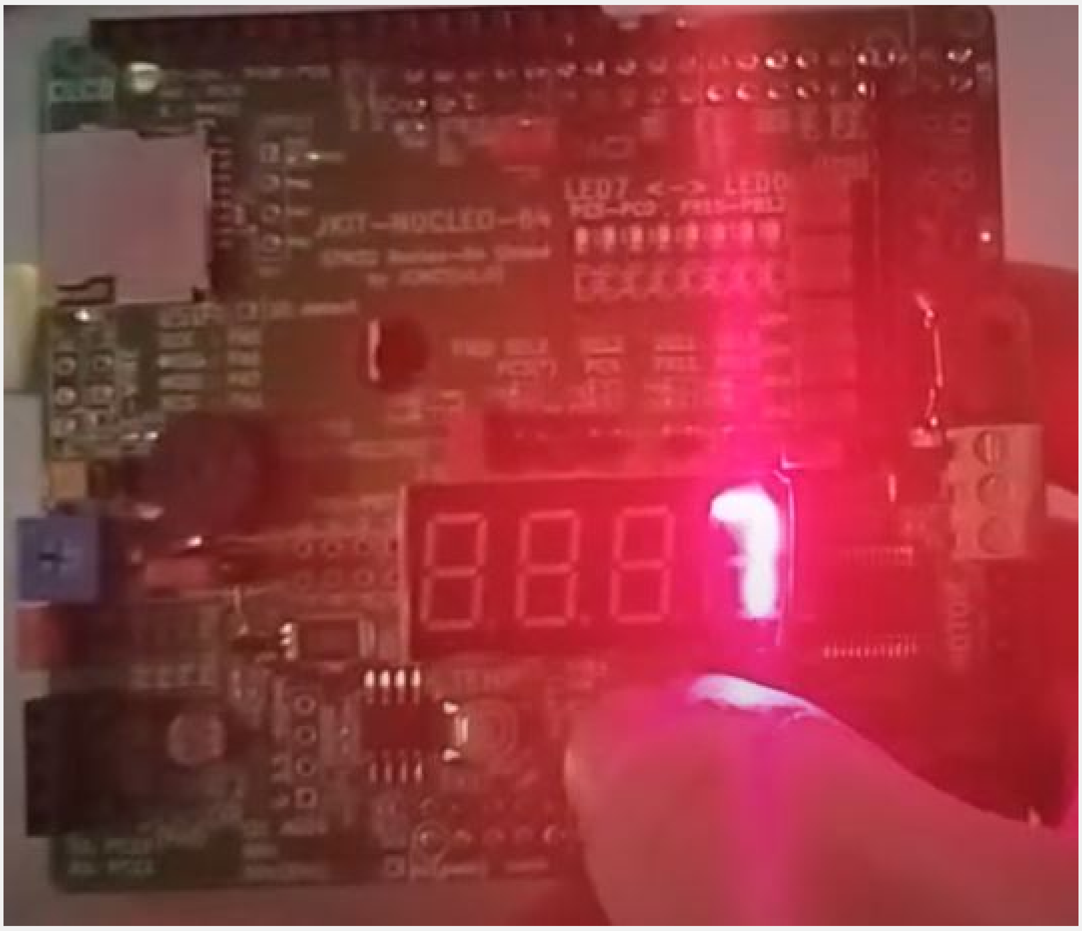

EXIT-NUCLEO-64
LED7 <-> LED0

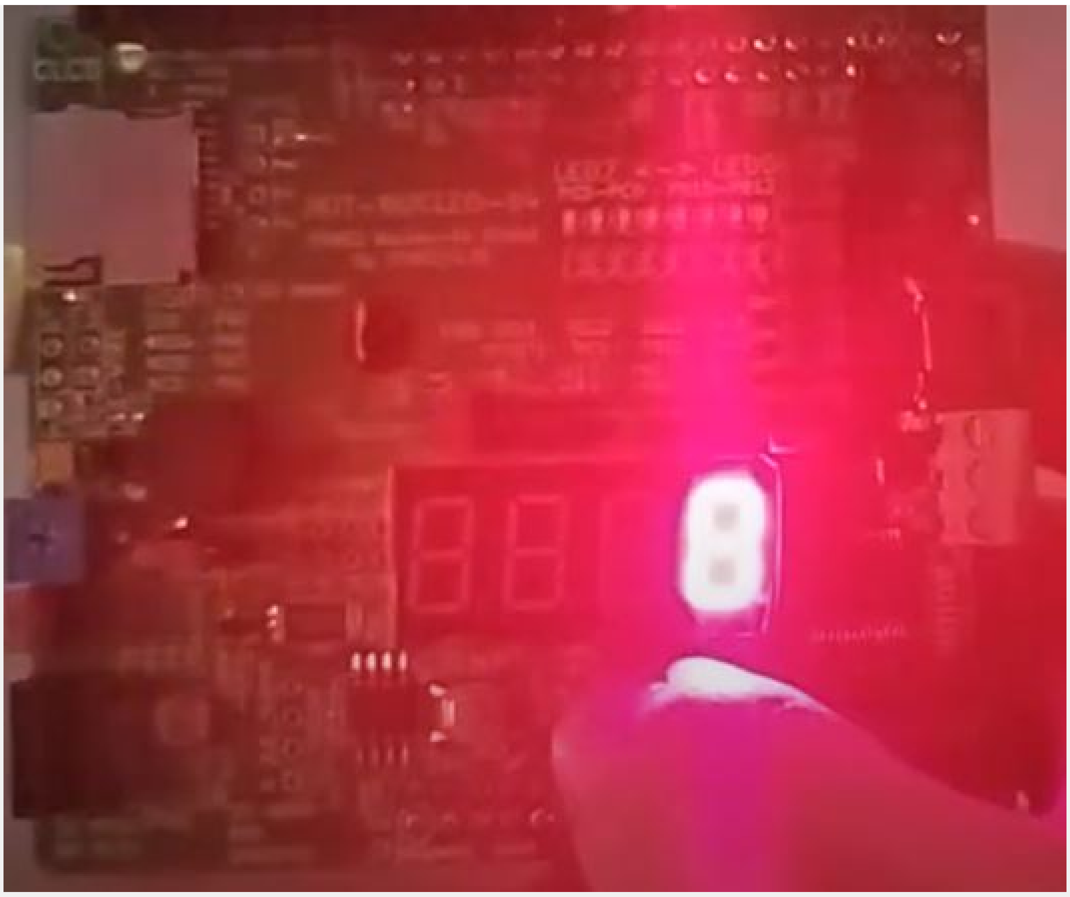

EXIT-NUCLEO-64
LED7 <-> LED0

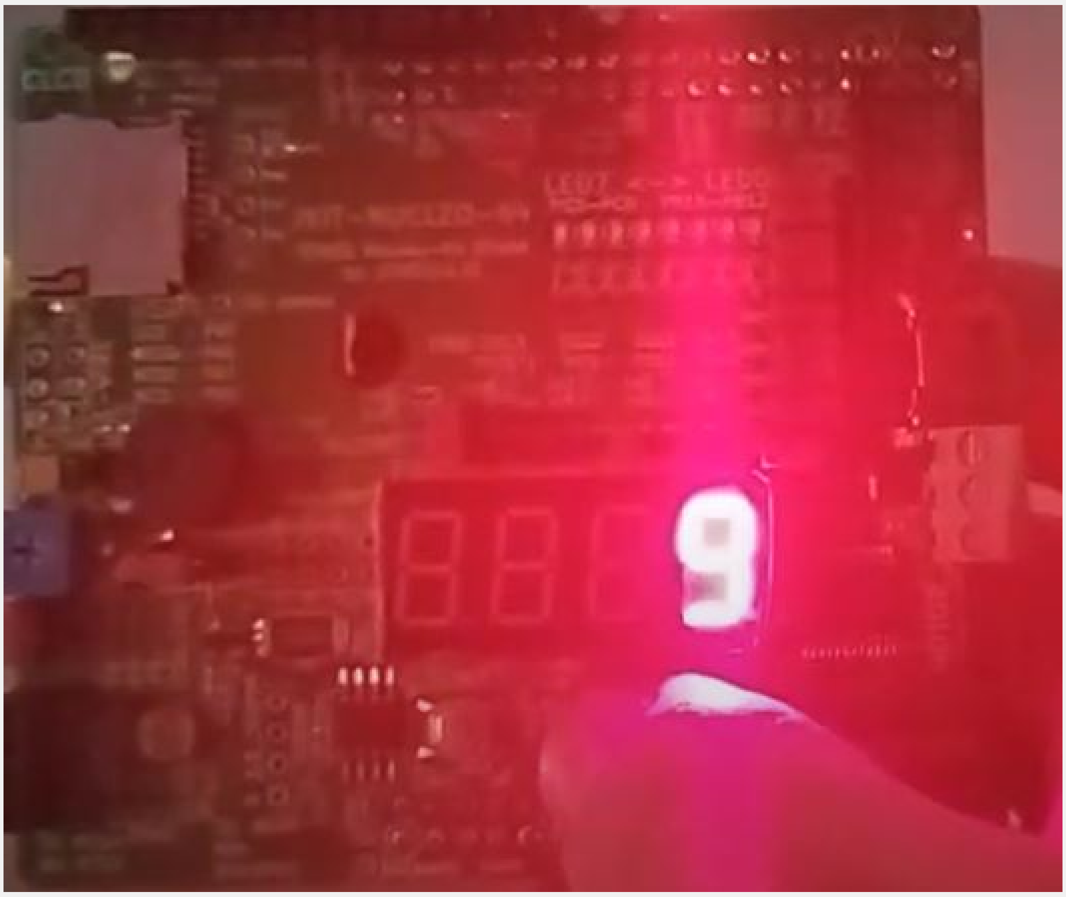

JKIT-NUCLEO-64
LED7 <--> LED0

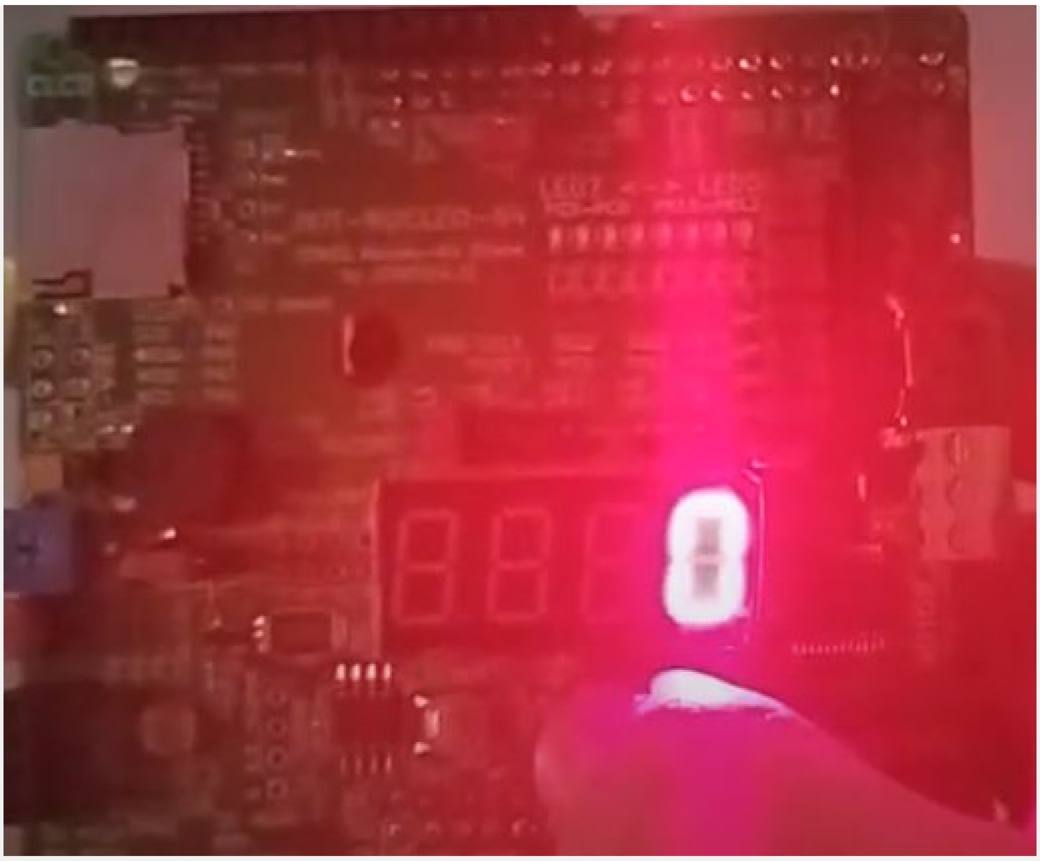

JKIT-NUCLEO-64
LED7 <--> LED0

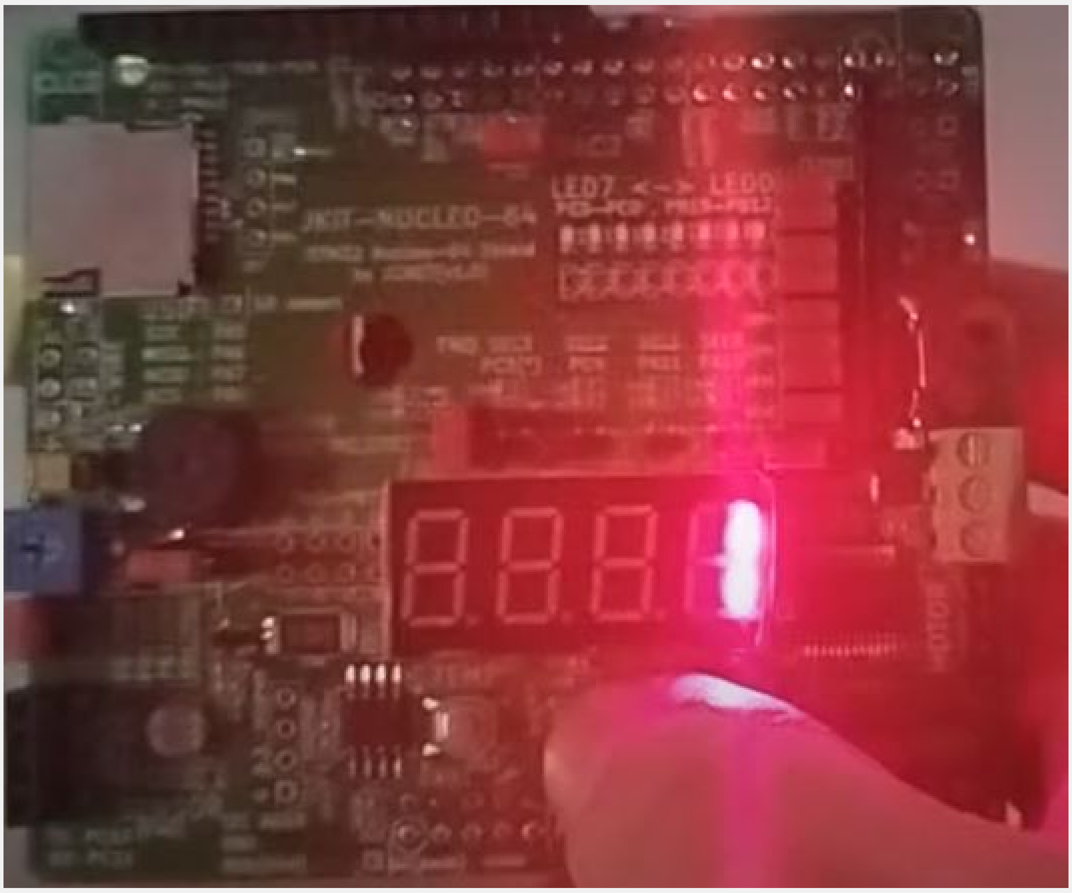

JKIT-NUCLEO-64
LED7 <--> LED0

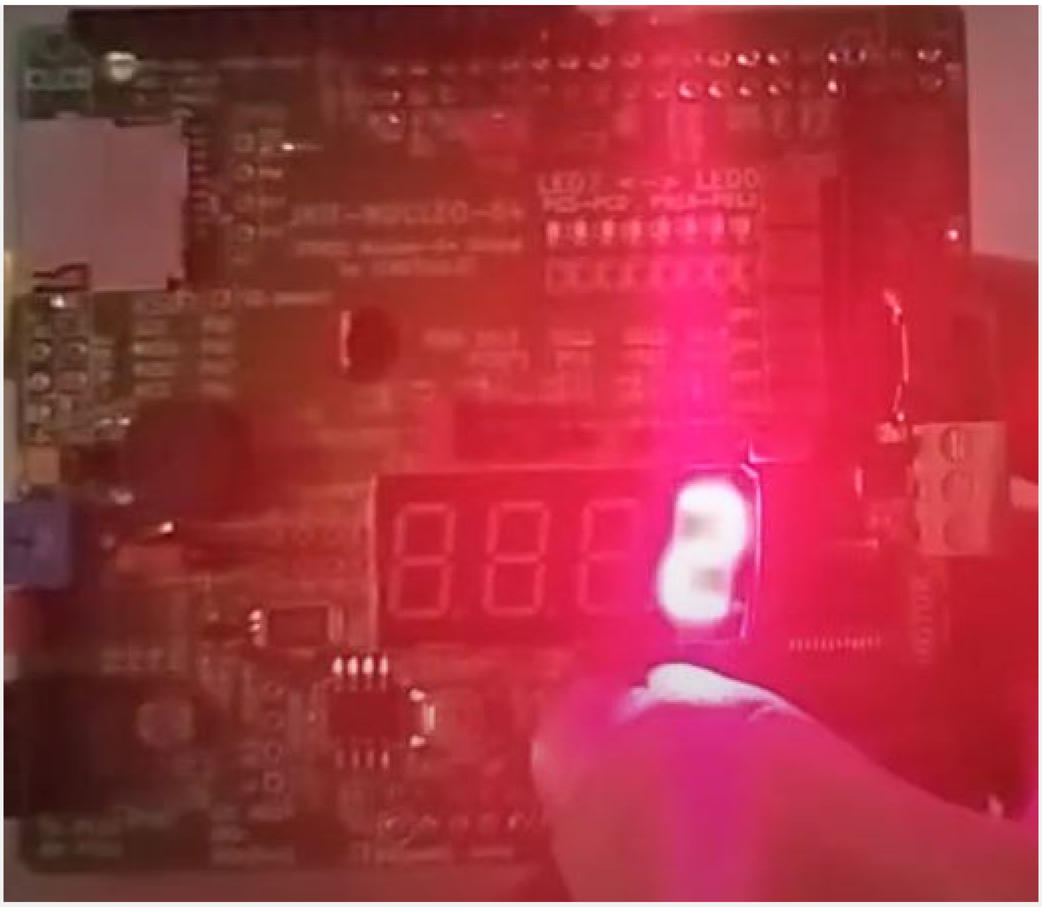

JKIT-NUCLEO-64
LED7 <--> LED0

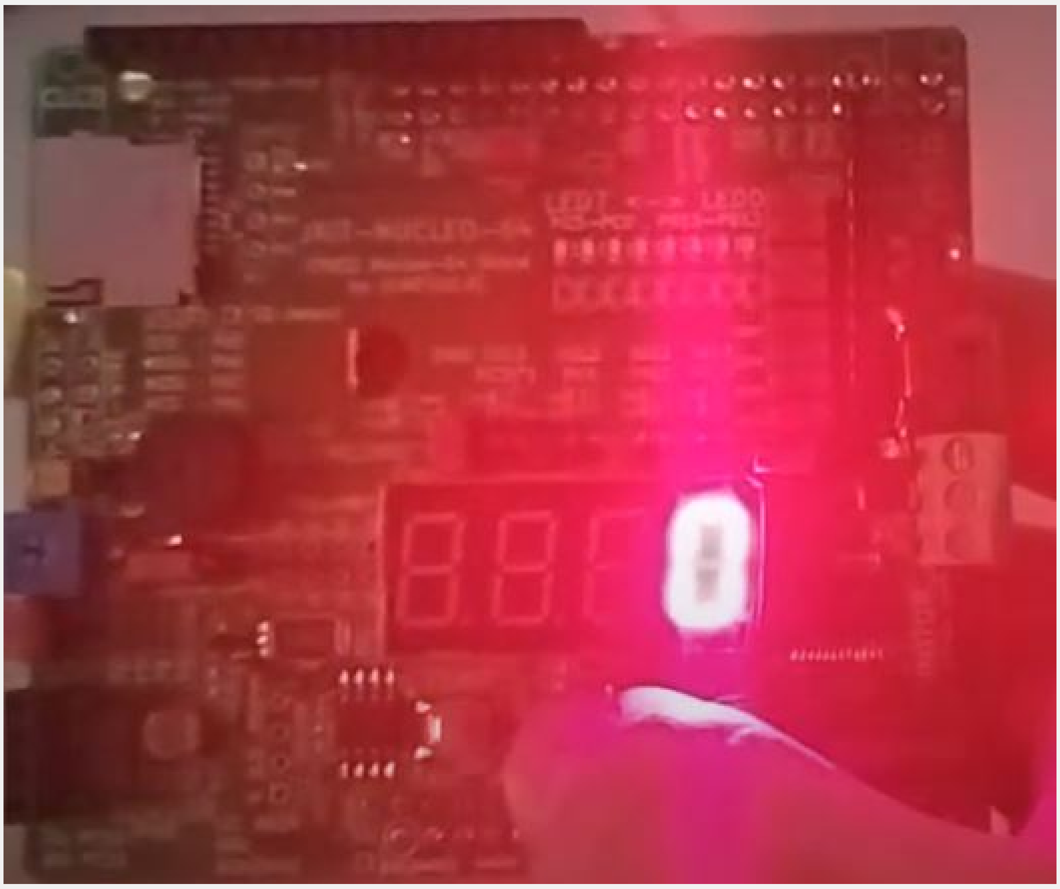





JKIT-NUCLEO-64
LED7 <--> LED0


JKIT-NUCLEO-64
LED7 <--> LED0





Add demo video link

Reference

Complete list of all references used (github, blog, paper, etc)

```
https://eteo.tistory.com/70
https://deepbluembedded.com/stm32-button-debounce-code-examples-tutorial/
https://stm32f4-discovery.net/2014/08/stm32f4-external-interrupts-tutorial/
```

Troubleshooting

(Option) You can write a Troubleshooting section