LAB: GPIO Digital InOut 7-segment

Date: 2023.10.05

Author: Hanmin Kim

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

Demo video: https://youtu.be/1zZtHoalgOA?si=OnFVih_5AYH7inEa

Introduction

In this lab, a simple program will be written to control a 7-segment display to display increasing decimal numbers (0 to 9) when a pushbutton is pressed.

Requirement

Hardware

MCU

- NUCLEO-F411RE

Actuator/Sensor/Others:

- 7-segment display(5101ASR)
- Array resistor (330 ohm)
- decoder chip(74LS47)
- breadboard

Software

Keil uVision, CMSIS, EC_HAL library

Exercise

Port/Pin	Description	Register setting
Port A Pin 5	Clear Pin5 mode	GPIOA->MODER &=~(3<<(5*2))
Port A Pin 5	Set Pin5 mode = Output	GPIOA->MODER |=1<<(5*2)
Port A Pin 6	Clear Pin6 mode	GPIOA->MODER &=~(3<<(6*2))
Port A Pin 6	Set Pin6 mode = Output	GPIOA->MODER |=1<<(6*2)
Port A Pin Y	Clear PinY mode	GPIOA->MODER &=~(3<<(Y*2))
Port A Pin Y	Set PinY mode = Output	GPIOA->MODER |=1<<(Y*2)
Port A Pin 5~9	Clear Pin5~9 mode	GPIOA->MODER &=~(31<<(5*2))
	Set Pin5~9 mode = Output	GPIOA->MODER |=31<<(5*2)
Port X Pin Y	Clear Pin Y mode	GPIOX->MODER &=~(3<<(Y*2))
	Set Pin Y mode = Output	GPIOX->MODER |=1<<(Y*2)
Port A Pin5	Set Pin5 otype=push-pull	GPIOA->OTYPER |=0<<(5*2)
Port A PinY	Set PinY otype=push-pull	GPIOA-> OTYPER |=0<<(5*2)
Port A Pin5	Set Pin5 ospeed=Fast	GPIOA->OSPEEDR |=2<<(5*2)
Port A PinY	Set PinY ospeed=Fast	GPIOA-> OSPEEDR |=2<<(Y*2)
Port A Pin 5	Set Pin5 PUPD=no pullup/down	GPIOA->OTYPER |=0<<(5*2)
Port A Pin Y	Set PinY PUPD=no pullup/down	GPIOA-> OTYPER |=0<<(Y*2)

Problem 1: Connecting 7-Segment Display

Procedure

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

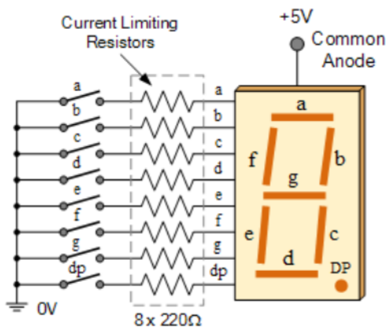
Instead of using the decoder chip, we are going to make the 7-segment decoder with the MCU programming.

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

Apply VCC and GND to the 7-segment display.

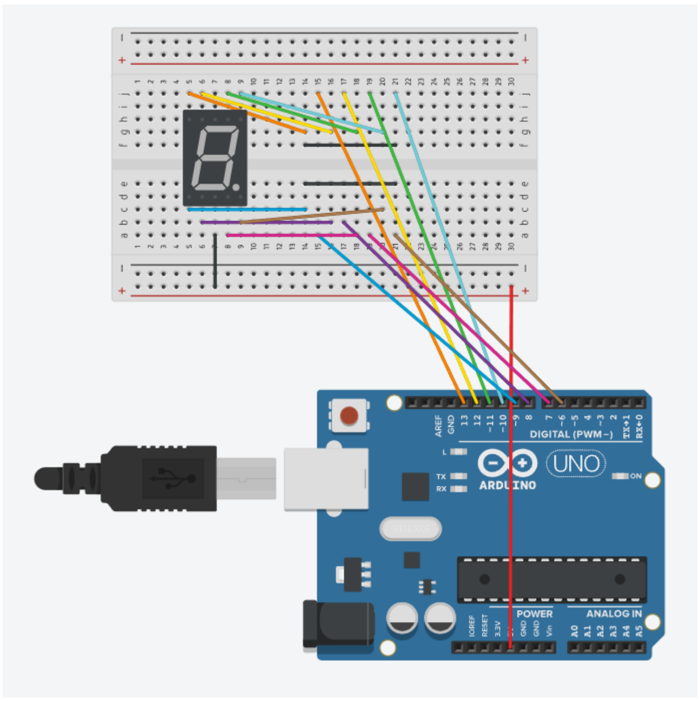
Apply 'H' to any 7-segment pin 'a'~'g' and observe if that LED is turned on or off

- 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.

D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	1	1	1
0	0	1	0	0	0	1	0	0	0	1
0	0	1	1	0	0	0	0	1	1	0
0	1	0	0	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0	1	0	0
0	1	1	0	0	1	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	1	0	0

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

Common Cathode (CC): In this setup, all cathodes are connected, and each LED segment has its anode. To light up a segment, you apply a positive voltage to its anode while grounding the common cathode.

Common Anode (CA): Here, all anodes are connected, and each LED segment has its cathode. To illuminate a segment, you apply a ground to its cathode while supplying a positive voltage to the common anode.

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, the light does not turn on because there is not enough voltage difference, so you have to set it to LOW for the LED to turn on.

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”**

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

- **ecGPIO.h**

code

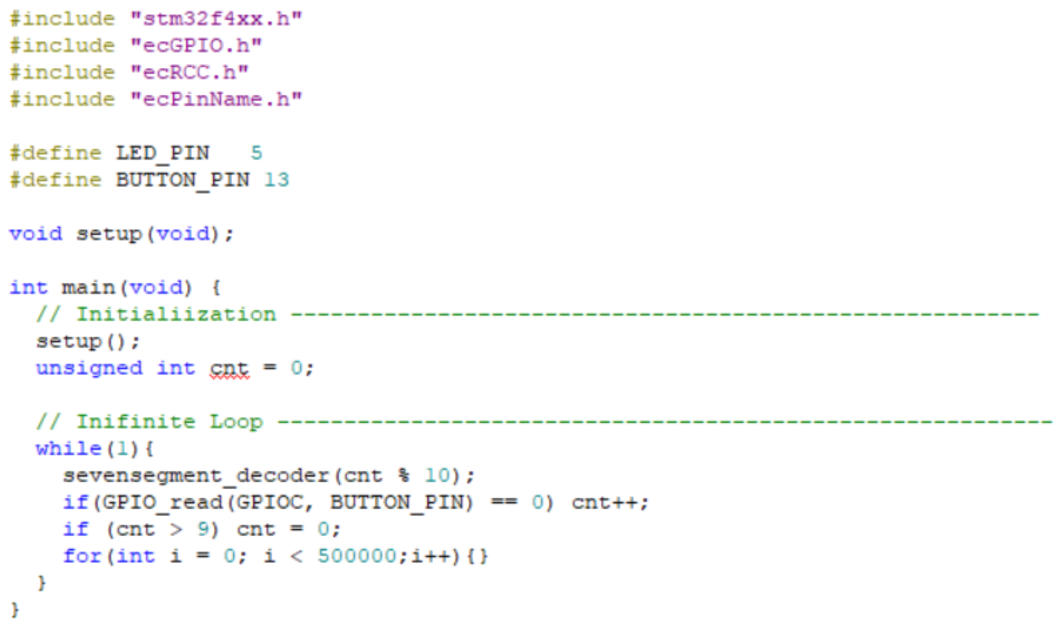
1. First, check if every number, 0 to 9, can be displayed properly
2. Then, create a code to display the number from 0 to 9 with each button press. After the number '9', it should start from '0' again.

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

Code

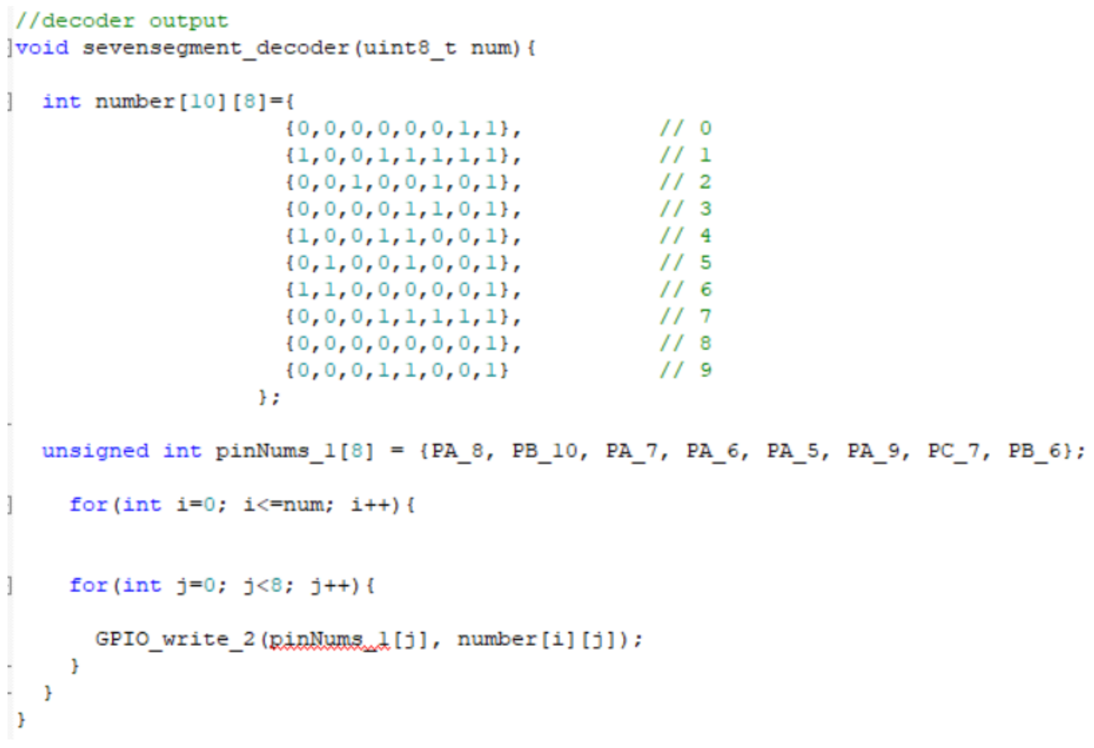
LAB_GPIO_7segment.c



```c
#include "stm32f4xx.h"
#include "ecGPIO.h"
#include "ecRCC.h"
#include "ecPinName.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_decoder(cnt % 10);
    if(GPIO_read(GPIOC, BUTTON_PIN) == 0) cnt++;
    if (cnt > 9) cnt = 0;
    for(int i = 0; i < 500000;i++){}
  }
}
```

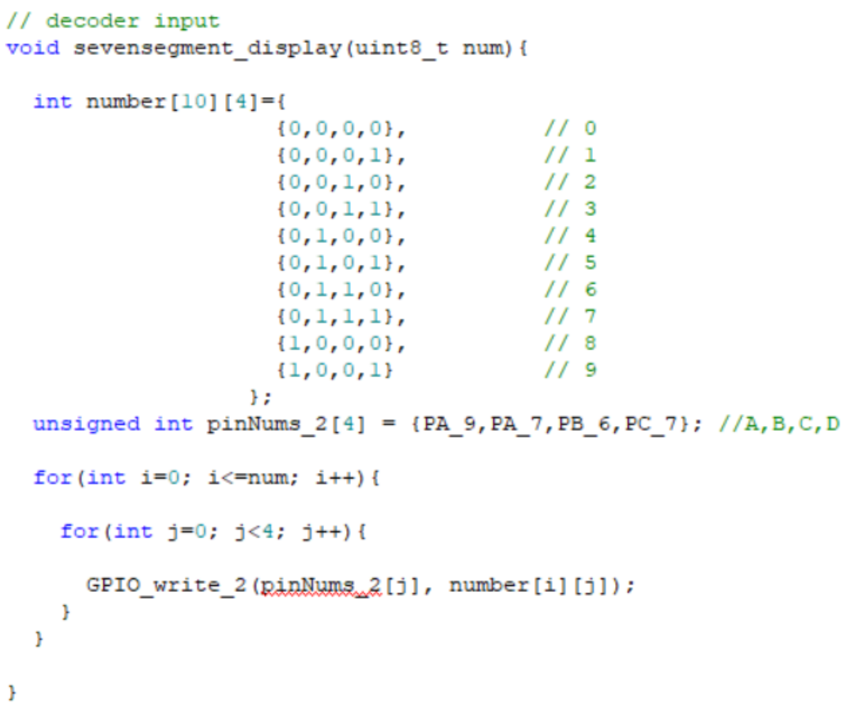
ecGPIO.c

Create a header file called ecPinName and configure it to operate like Arduino code.



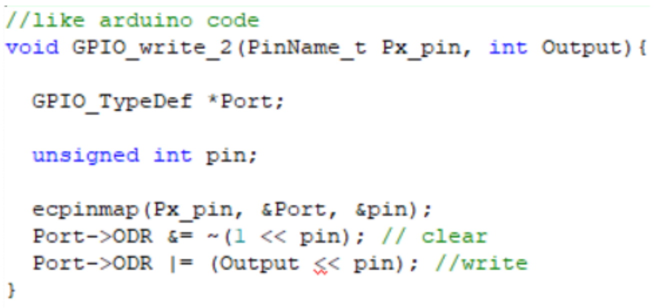
```c
//decoder output
void sevensegment_decoder(uint8_t num){

  int number[10][8]={
                    {0,0,0,0,0,0,1,1},          // 0
                    {1,0,0,1,1,1,1,1},          // 1
                    {0,0,1,0,0,1,0,1},          // 2
                    {0,0,0,0,1,1,0,1},          // 3
                    {1,0,0,1,1,0,0,1},          // 4
                    {0,1,0,0,1,0,0,1},          // 5
                    {1,1,0,0,0,0,0,1},          // 6
                    {0,0,0,1,1,1,1,1},          // 7
                    {0,0,0,0,0,0,0,1},          // 8
                    {0,0,0,1,1,0,0,1}           // 9
                  };

  unsigned int pinNums_1[8] = {PA_8, PB_10, PA_7, PA_6, PA_5, PA_9, PC_7, PB_6};

    for(int i=0; i<=num; i++){


    for(int j=0; j<8; j++){

      GPIO_write_2(pinNums_1[j], number[i][j]);
    }
  }
}
```

Design the output of the decoder.



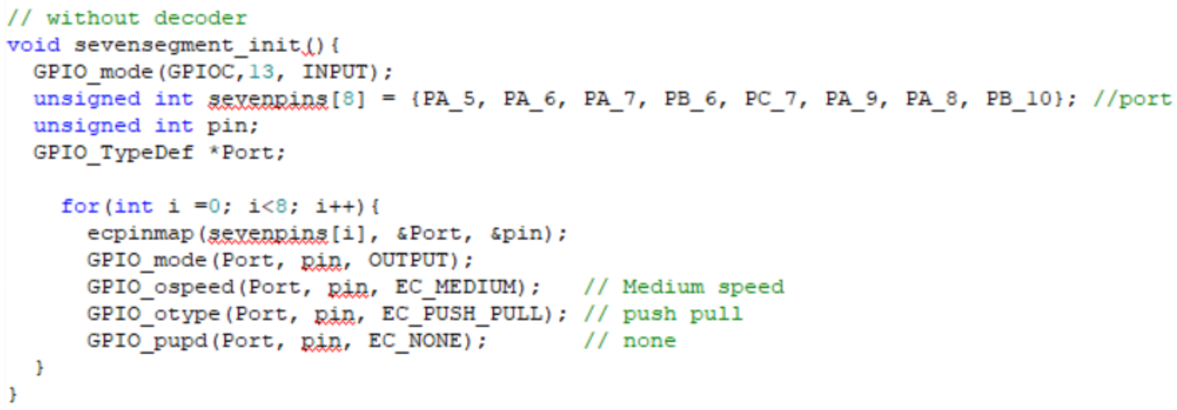
```c
// decoder input
void sevensegment_display(uint8_t num){

  int number[10][4]={
                  {0,0,0,0},          // 0
                  {0,0,0,1},          // 1
                  {0,0,1,0},          // 2
                  {0,0,1,1},          // 3
                  {0,1,0,0},          // 4
                  {0,1,0,1},          // 5
                  {0,1,1,0},          // 6
                  {0,1,1,1},          // 7
                  {1,0,0,0},          // 8
                  {1,0,0,1}           // 9
                };
  unsigned int pinNums_2[4] = {PA_9,PA_7,PB_6,PC_7}; //A,B,C,D

  for(int i=0; i<=num; i++){

    for(int j=0; j<4; j++){

      GPIO_write_2(pinNums_2[j], number[i][j]);
    }
  }

}
```

Design the input of the decoder.



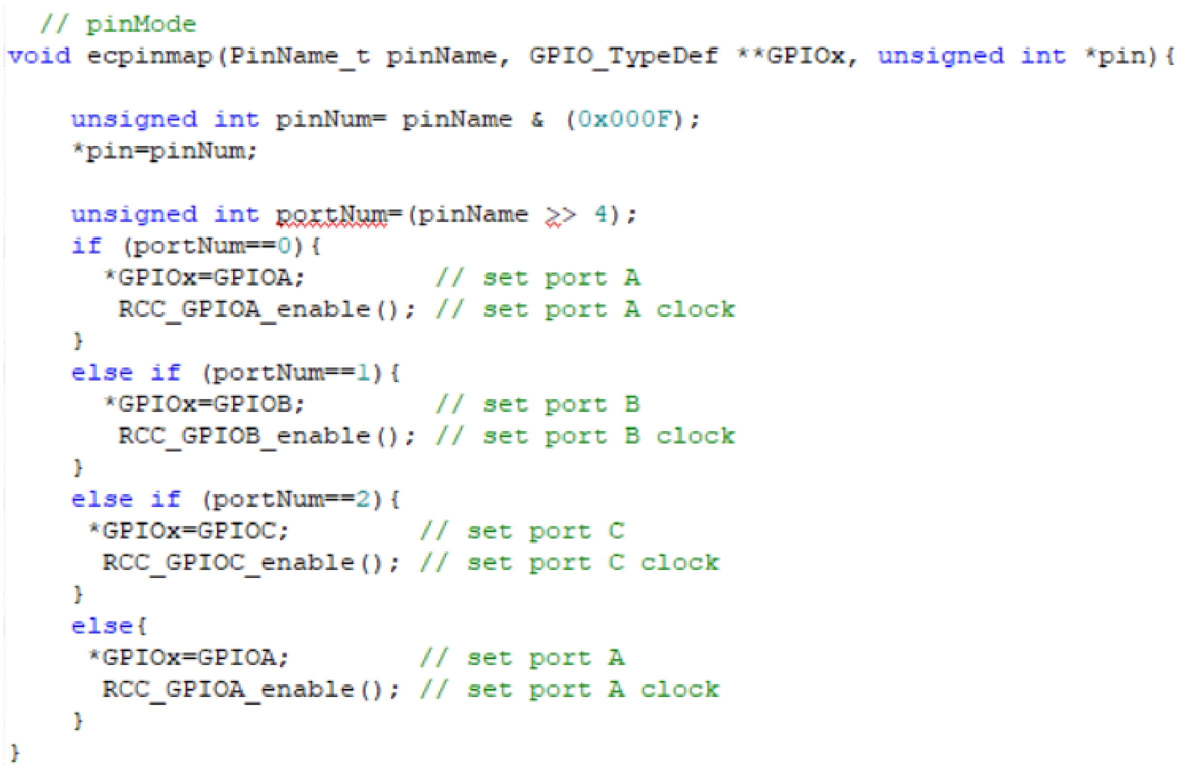
```c
//like arduino code
void GPIO_write_2(PinName_t Px_pin, int Output){

  GPIO_TypeDef *Port;

  unsigned int pin;

  ecpinmap(Px_pin, &Port, &pin);
  Port->ODR &= ~(1 << pin); // clear
  Port->ODR |= (Output << pin); //write
}
```

GPIO_write_2 was created to operate like Arduino code.



```c
// without decoder
void sevensegment_init(){
  GPIO_mode(GPIOC,13, INPUT);
  unsigned int sevenpins[8] = {PA_5, PA_6, PA_7, PB_6, PC_7, PA_9, PA_8, PB_10}; //port
  unsigned int pin;
  GPIO_TypeDef *Port;

    for(int i =0; i<8; i++){
      ecpinmap(sevenpins[i], &Port, &pin);
      GPIO_mode(Port, pin, OUTPUT);
      GPIO_ospeed(Port, pin, EC_MEDIUM);   // Medium speed
      GPIO_otype(Port, pin, EC_PUSH_PULL); // push pull
      GPIO_pupd(Port, pin, EC_NONE);       // none
  }
}
```

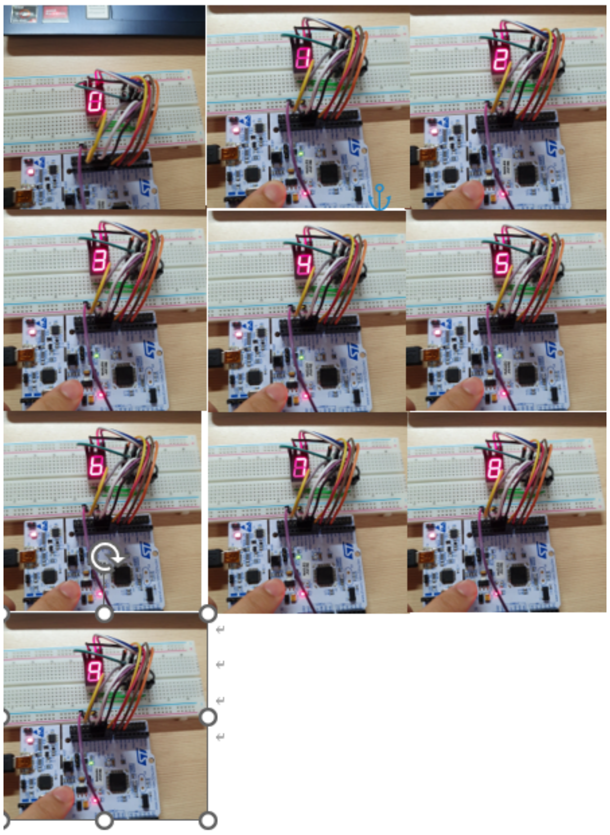
The output pins and characteristics of the decoder were set.



```
  // pinMode
void ecpinmap(PinName_t pinName, GPIO_TypeDef **GPIOx, unsigned int *pin){

    unsigned int pinNum= pinName & (0x000F);
    *pin=pinNum;

    unsigned int portNum=(pinName >> 4);
    if (portNum==0){
      *GPIOx=GPIOA;         // set port A
       RCC_GPIOA_enable(); // set port A clock
    }
    else if (portNum==1){
      *GPIOx=GPIOB;         // set port B
       RCC_GPIOB_enable(); // set port B clock
    }
    else if (portNum==2){
     *GPIOx=GPIOC;         // set port C
      RCC_GPIOC_enable(); // set port C clock
    }
    else{
     *GPIOx=GPIOA;         // set port A
      RCC_GPIOA_enable(); // set port A clock
    }
}
```

An ecpinmap was created to operate like Arduino code.

Results



In the initial state, the LED outputs 0, and each time the button is pressed, numbers from 0 to 9 are repeatedly output.

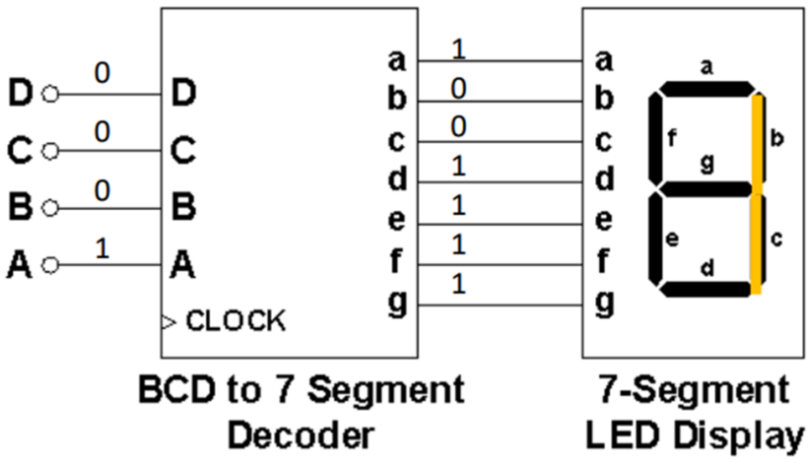
video link: https://www.youtube.com/watch?si=OnFVih_5AYH7inEa&v=1zZtHoalgOA&feature=youtu.be

Problem 3: Using both 7-Segment Decoder and 7-segment display

Procedure

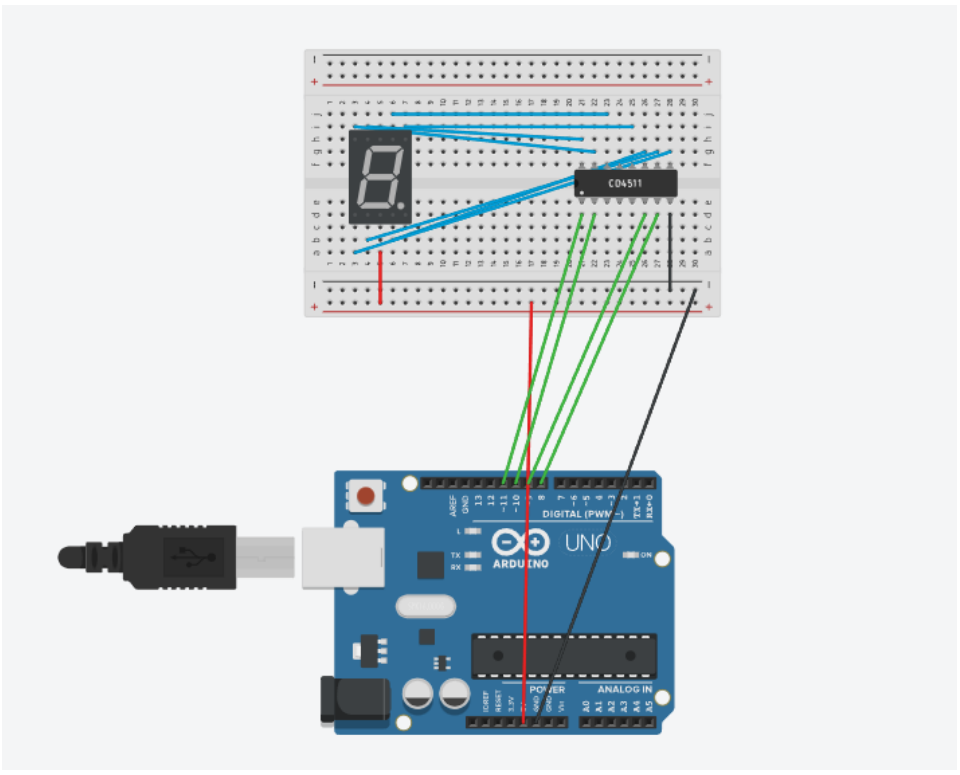
Now, use the decoder chip (**74LS47**). Connect it to the bread board.

Then, you need only 4 Digital out pins of MCU to display from 0 to 9.



Connection Diagram

circuit diagram



Configuration

Digital In for Button (B1)	Digital Out for 7-Segment
Digital In	Digital Out
PC13	PA7, PB6, PC7, PA9
PULL-UP	Push-Pull, No Pull-up-Pull-down, Medium Speed

Reference

https://ykkim.gitbook.io/ec/ec-course/lab/lab-gpio-digital-inout-7segment

https://ykkim.gitbook.io/ec/stm32-m4-programming/peripheral-api/pinmap

74LS47 Data sheet, NUCLEO-F411RE Data sheet