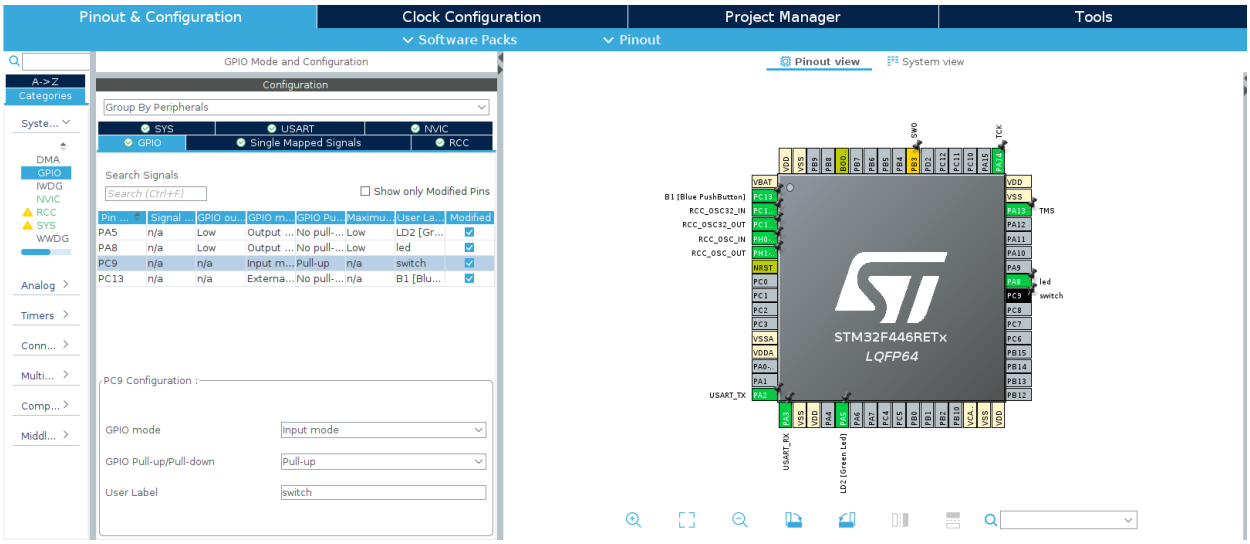
Tasks

@July 26, 2024

Complete the given task Day1 (GPIO)

☐ 1.Write a c program to turn on led when we press the switch and turn off the led when we released the switch
Take Input pin PC9 and Output pin PA8



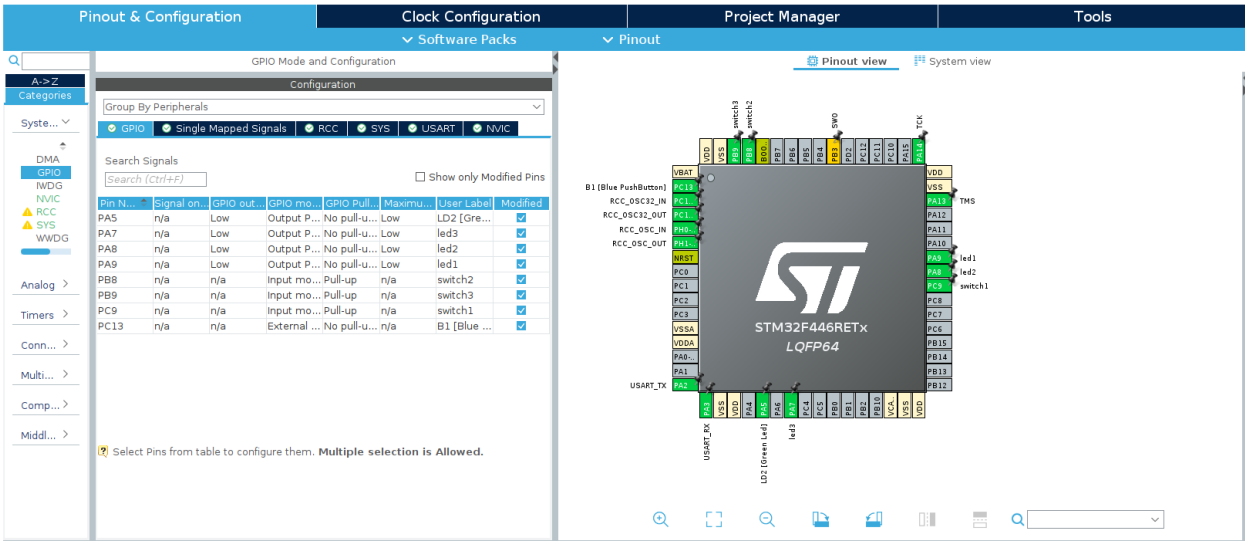

```
//In main.c file
while (1)
  {
      if(!(HAL_GPIO_ReadPin(GPIOC,SW_Pin)))
      {
```

```
          HAL_GPIO_WritePin(GPIOA,LED_Pin,1);//HAL is liabray write means writting value of digi
          HAL_Delay(100);
      }
      else
      {
          HAL_GPIO_WritePin(GPIOA,LED_Pin,0);
          HAL_Delay(100);
      }
  }
```

☐ 2.Write a c program to turn on Led when we press the switch and turn off Led if again we press the switch
Take Input pin PC9 and Output pin PA8

```
//in main.c file
// Variable to keep track of the LED state
uint8_t ledState = 0;

  while (1)
  {    // Read the state of the switch
       if(HAL_GPIO_ReadPin(GPIOC, switch_Pin) == GPIO_PIN_RESET) // Assuming active-low switch
       {
         // Toggle the LED state
         ledState = !ledState;

         // Set the LED pin according to the ledState variable
         HAL_GPIO_WritePin(GPIOA, led_Pin , ledState ? GPIO_PIN_SET : GPIO_PIN_RESET);

         // Wait until the button is released
         while(HAL_GPIO_ReadPin(GPIOC, switch_Pin) == GPIO_PIN_RESET);

         // Debounce delay
         HAL_Delay(100);
       }
      }
  }
```

☐ 3.Write a program to turn on 3 led using 3 switch
conditions :
if we press 1st switch 1st led turn on then off.
if we press 2nd switch 2nd led turn on then off.
if we press 3rd switch 3rd led turn on then off.
Take PA7, PA8 and PA9 as output pin.
PB8, PB9 and PC9 as input pin.




```
while (1)
{
    if(!(HAL_GPIO_ReadPin(GPIOC,switch1_Pin)))
        {
            HAL_GPIO_WritePin(GPIOA,led1_Pin,1);
            HAL_Delay(100);
        }
    else if(!(HAL_GPIO_ReadPin(GPIOB,switch2_Pin)))
    {
        HAL_GPIO_WritePin(GPIOA,led2_Pin,1);
        HAL_Delay(100);
    }
    else if(!(HAL_GPIO_ReadPin(GPIOB,switch3_Pin)))
    {
        HAL_GPIO_WritePin(GPIOA,led3_Pin,1);
        HAL_Delay(100);
    }

    else
        {
            HAL_GPIO_WritePin(GPIOA,led1_Pin,0);
            HAL_GPIO_WritePin(GPIOA,led2_Pin,0);
            HAL_GPIO_WritePin(GPIOA,led3_Pin,0);
            HAL_Delay(100);
        }
  }
```

☐ 4.Write a program to turn on 3 led using 3 switch.
conditions :
if we press the 1st switch 1st led turn on
if we press the 2nd switch 1st and 2nd led turn on
if we press the 3rd switch 1st,2nd and 3rd led turn on
Take PA7, PA8 and PA9 as output pin.
PB8, PB9 and PC9 as input pin.

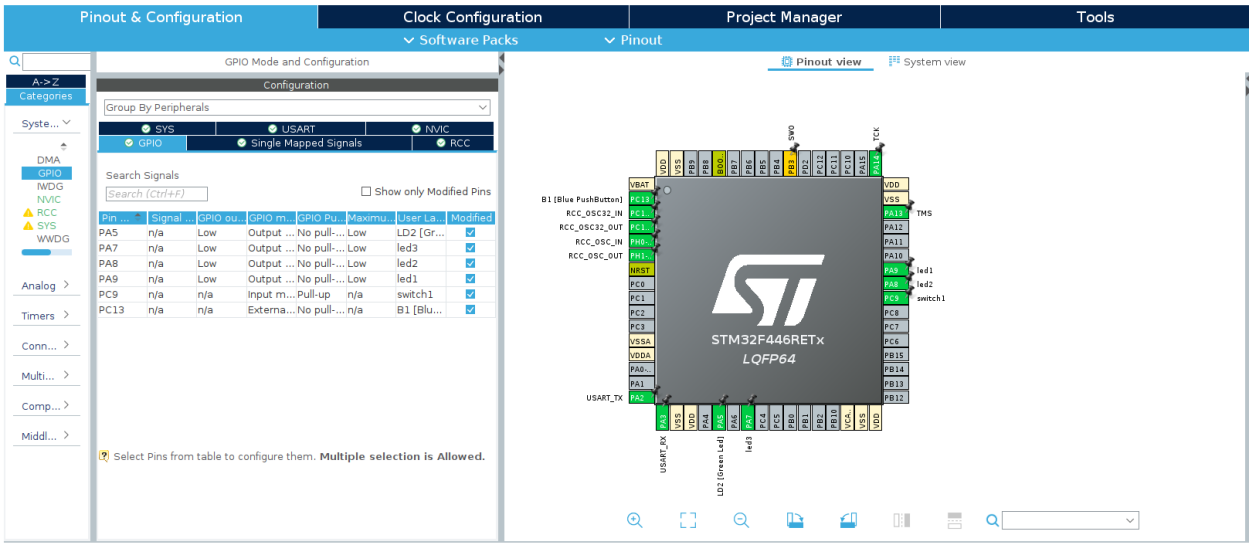
```
while (1)
  {
      if(!(HAL_GPIO_ReadPin(GPIOC,switch1_Pin)))
          {
              HAL_GPIO_WritePin(GPIOA,led1_Pin,1);
              HAL_Delay(100);
          }
      else if(!(HAL_GPIO_ReadPin(GPIOB,switch2_Pin)))
      {
          HAL_GPIO_WritePin(GPIOA,led1_Pin,1);
          HAL_GPIO_WritePin(GPIOA,led2_Pin,1);
          HAL_Delay(100);
      }
      else if(!(HAL_GPIO_ReadPin(GPIOB,switch3_Pin)))
      {
          HAL_GPIO_WritePin(GPIOA,led1_Pin,1);
          HAL_GPIO_WritePin(GPIOA,led2_Pin,1);
          HAL_GPIO_WritePin(GPIOA,led3_Pin,1);
          HAL_Delay(100);
      }

      else
          {
              HAL_GPIO_WritePin(GPIOA,led1_Pin,0);
              HAL_GPIO_WritePin(GPIOA,led2_Pin,0);
              HAL_GPIO_WritePin(GPIOA,led3_Pin,0);
              HAL_Delay(100);
          }
    }
```

☐ 5.Two switch and one led
condition :
if we press the 1st switch led will turn on.
if we press the 2nd switch led will turn off.
Take PA7 as output pin
PB8 and PB9 as input pin

```
  while (1)
  {
      if(!(HAL_GPIO_ReadPin(GPIOC,switch1_Pin)))
          {
              HAL_GPIO_WritePin(GPIOA,led1_Pin,1);
              HAL_Delay(100);
          }
      else if(!(HAL_GPIO_ReadPin(GPIOB,switch2_Pin)))
      {
          HAL_GPIO_WritePin(GPIOA,led1_Pin,0);
          HAL_Delay(100);
      }
  }
```

☐ 6.Write a program using 3 led and 1 switch
condition :

if we press the switch 1st time 1st led will turn on.
if we press the switch 2nd time 2nd led will turn on.
if we press the switch 3rd time 3rd led will turn on.
after releasing switch leds will be in off state.
also show the num of times switch pressed.
Take PA7, PA8 and PA9 as output pin.
PB8 as input pin.




```
uint8_t switchPressCount = 0;

while (1)
  {
      // Read the state of the switch
          if (HAL_GPIO_ReadPin(GPIOC, switch1_Pin) == GPIO_PIN_RESET) // Assuming active-low swi
          {
            // Increment the switch press count
            switchPressCount++;

            // Turn on the corresponding LED based on the switch press count
            if (switchPressCount == 1)
            {
              HAL_GPIO_WritePin(GPIOA, led1_Pin, GPIO_PIN_SET);
            }
            else if (switchPressCount == 2)
            {
              HAL_GPIO_WritePin(GPIOA, led2_Pin, GPIO_PIN_SET);
            }
            else if (switchPressCount == 3)
            {
              HAL_GPIO_WritePin(GPIOA, led3_Pin, GPIO_PIN_SET);
            }

            // Wait until the button is released
            while (HAL_GPIO_ReadPin(GPIOC, switch1_Pin) == GPIO_PIN_RESET);
```

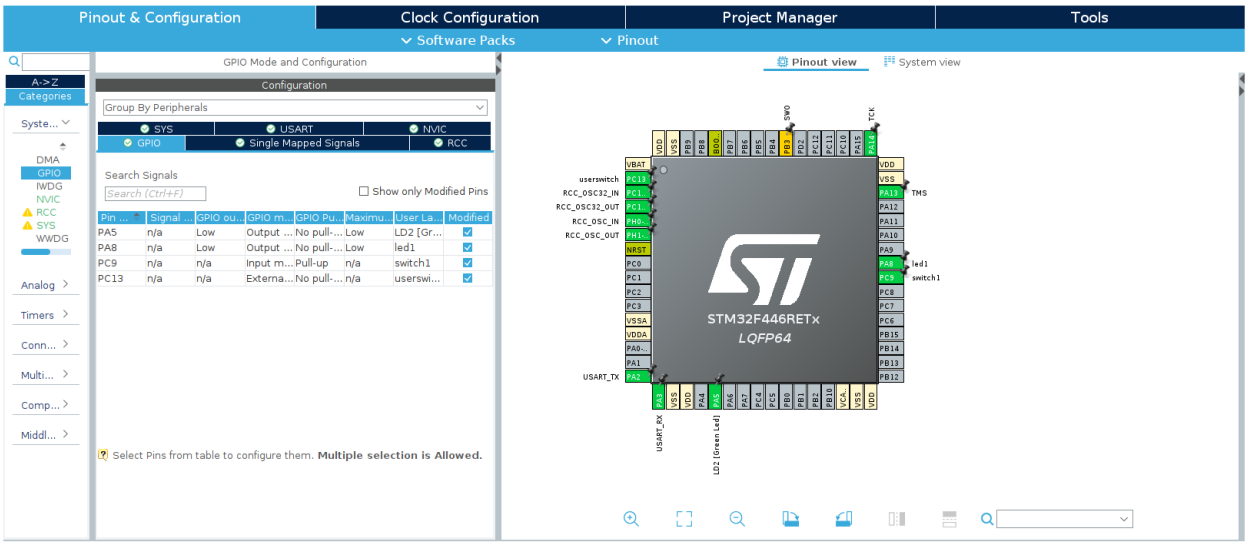
```
            // Turn off all LEDs after releasing the switch
            HAL_GPIO_WritePin(GPIOA, led1_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOA, led2_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOA, led3_Pin, GPIO_PIN_RESET);

            // Reset the switch press count after the third press
            if (switchPressCount == 3)
            {
              switchPressCount = 0;
            }
            else if (switchPressCount == 4)
            {
                switchPressCount=0;
            }


            // Debounce delay
            HAL_Delay(100);
          }

  }
```

☐ 7.Write a program using 3 led and 1 switch
condition :
if we press the switch 1st time 1st led will turn on.
if we press the switch 2nd time 2nd led will turn on.
if we press the switch 3rd time 3rd led will turn on.
after releasing switch led state will not change.
also show the num of times switch pressed.
Take PA7, PA8 and PA9 as output pin.
PB8 as input pin.

```
uint8_t switchPressCount = 0;

  while (1)
  {
      // Read the state of the switch
          if (HAL_GPIO_ReadPin(GPIOC, switch1_Pin) == GPIO_PIN_RESET) // Assuming active-low swi
          {
            // Increment the switch press count
            switchPressCount++;

            // Turn on the corresponding LED based on the switch press count
            if (switchPressCount == 1)
            {
              HAL_GPIO_WritePin(GPIOA, led1_Pin, GPIO_PIN_SET);
            }
            else if (switchPressCount == 2)
            {
              HAL_GPIO_WritePin(GPIOA, led2_Pin, GPIO_PIN_SET);
            }
            else if (switchPressCount == 3)
            {
```

```
            HAL_GPIO_WritePin(GPIOA, led3_Pin, GPIO_PIN_SET);
            switchPressCount=0;
          }

          // Wait until the button is released
          while (HAL_GPIO_ReadPin(GPIOC, switch1_Pin) == GPIO_PIN_RESET);

          // Debounce delay
          HAL_Delay(100);
        }

  }
```

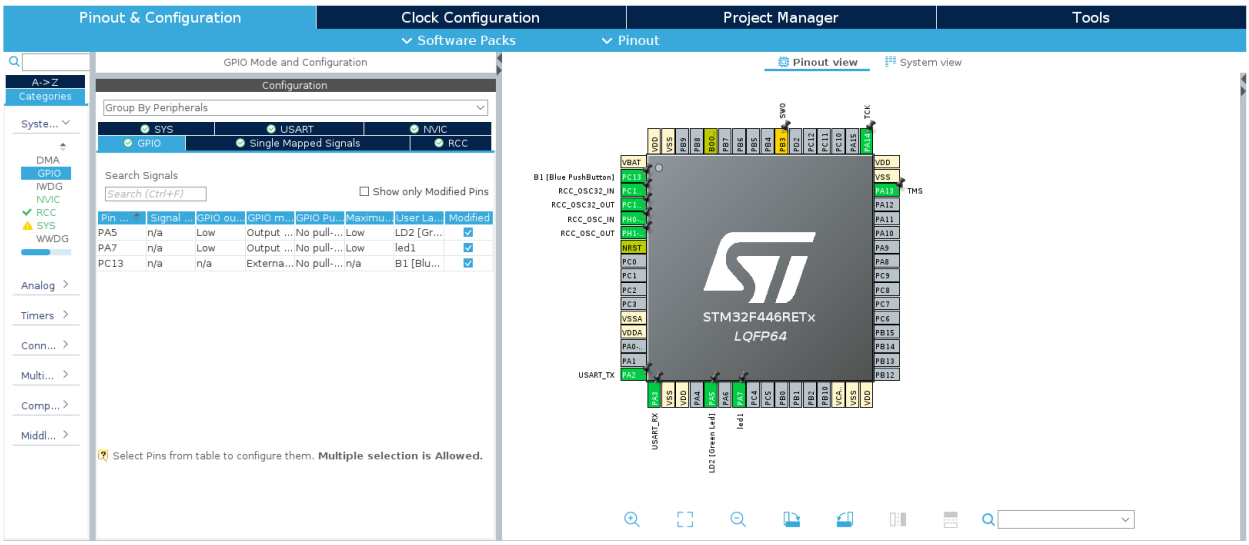
☐ 8.Write a Program using 1 led and 1 switch
conditions:
f we press the switch 1st time led will toggle 1 time
if we press the switch 2nd time led will toggle 2 times.
if we press the switch 3rd time led will toggle 3 times.
and this should be in a loop.
also print the number of times switch pressed.
Take onboard led and switch.




```
 uint8_t switchPressCount=0;
while (1)
{
    if(HAL_GPIO_ReadPin(GPIOC,switch1_Pin)== GPIO_PIN_RESET){
    switchPressCount++;
        for(uint8_t i=0; i < switchPressCount; i++){
                HAL_GPIO_WritePin(GPIOA, led1_Pin, GPIO_PIN_SET);
                HAL_Delay(1000);
                HAL_GPIO_WritePin(GPIOA, led1_Pin, GPIO_PIN_RESET);
                HAL_Delay(1000);
        }
    if (switchPressCount == 3){
    switchPressCount = 0;
```

```
}
while (HAL_GPIO_ReadPin(GPIOC, switch1_Pin) == GPIO_PIN_RESET);
    HAL_Delay(1000);
}}
```

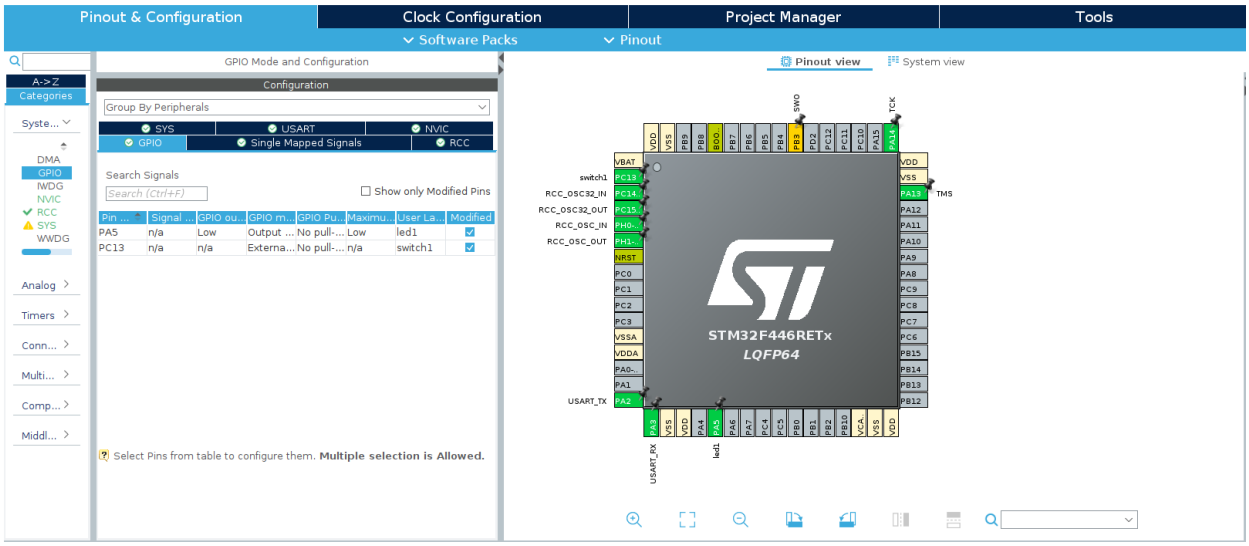
☐ 9.Write a Program using 2 leds and 1 switch
conditions:
if we press the switch 1st time led1 will toggle 1 time and led2 toggle 3 times.
if we press the switch 2nd time led1 will toggle 2 time and led2 toggle 6 times.
if we press the switch 3rd time led1 will toggle 3 time and led2 toggle 9 times.
and this should be in a loop.
also print the number of times switch pressed.
Take onboard switch , led (PA5) and PA7.

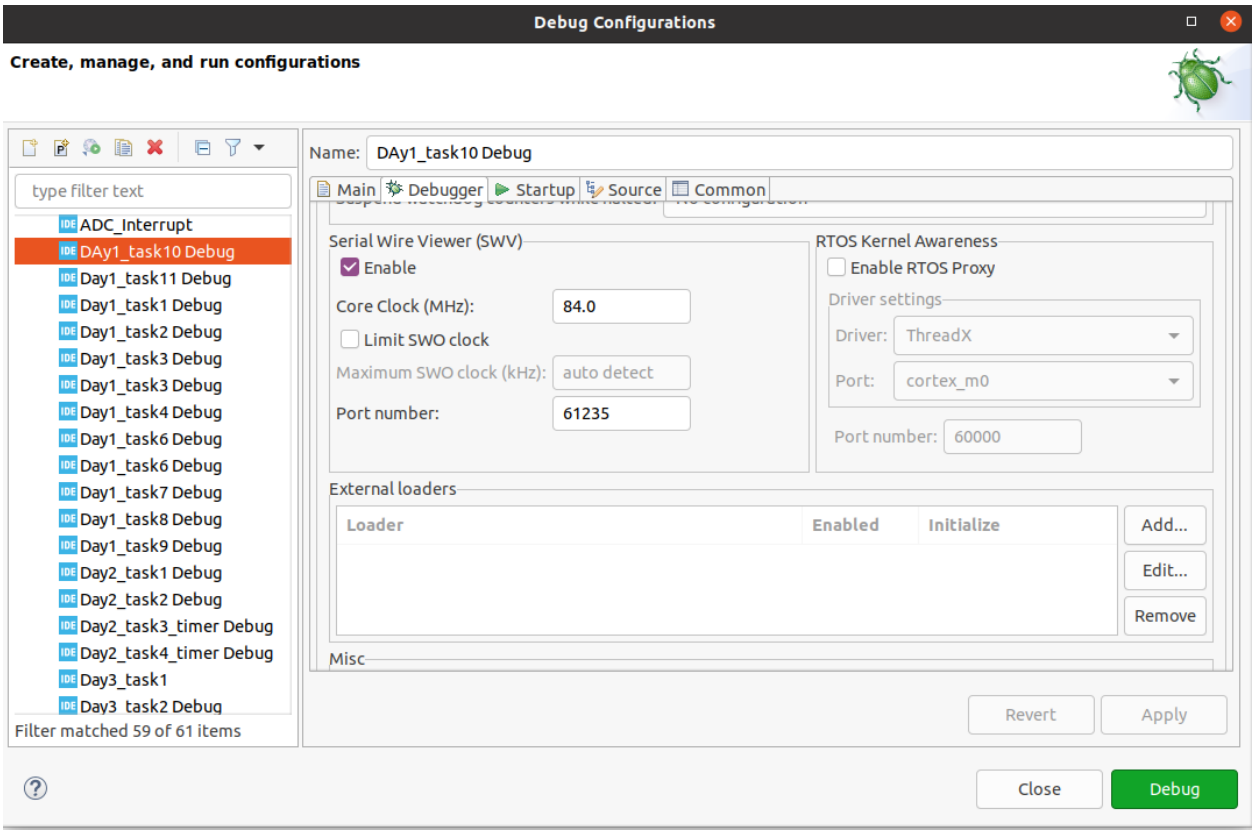



```
uint8_t switchPressCount=0;
while (1)
  {
      if(HAL_GPIO_ReadPin(GPIOC,GPIO_PIN_13)== GPIO_PIN_RESET){
          switchPressCount++;
          for(uint8_t i=0; i< switchPressCount; i++){
              HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
              HAL_Delay(1000);
              HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
              HAL_Delay(1000);
          }
          for(uint8_t i=0; i < switchPressCount*3; i++){
              HAL_GPIO_WritePin(GPIOA, led1_Pin, GPIO_PIN_SET);
              HAL_Delay(1000);
              HAL_GPIO_WritePin(GPIOA, led1_Pin, GPIO_PIN_RESET);
              HAL_Delay(1000);
          }
          if (switchPressCount == 3){
              switchPressCount = 0;
          }
          while (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13) == GPIO_PIN_RESET);
```

```
        }
    }
```

☐ 10.Write a Program using 1 led and 1 switch
conditions:
if we press the switch 1st time led will turn on and print LED ON in SWV ITM Data Console.
if we press the switch 2nd time led will turn off and print LED OFF in SWV ITM Data Console and this should be in a loop.
after releasing switch led state will not change.
also print the number of times switch pressed.
Take onboard led and switch.

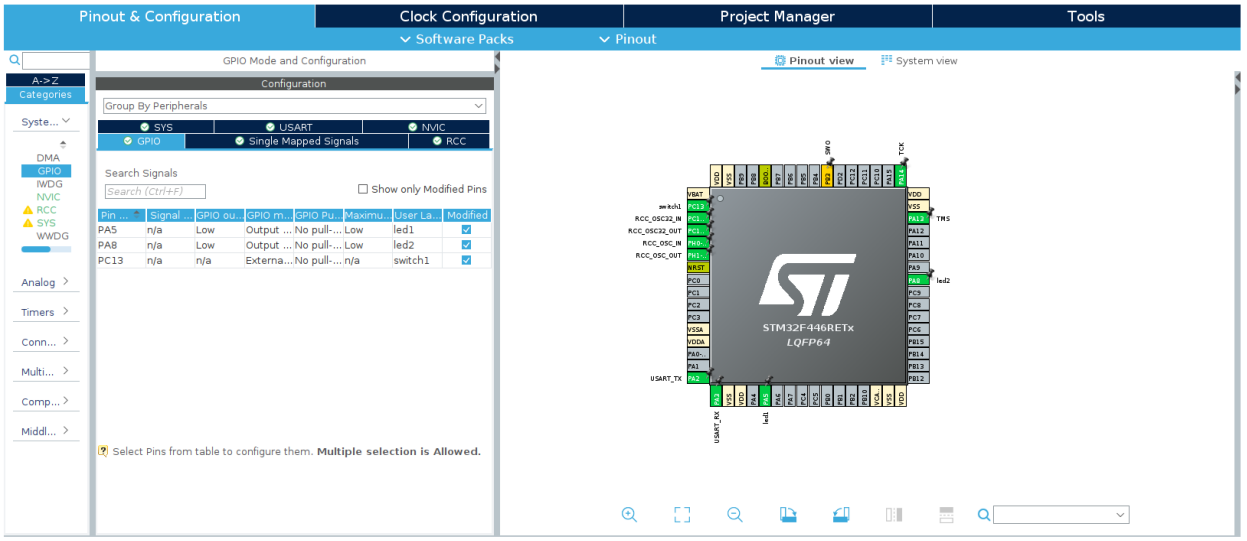


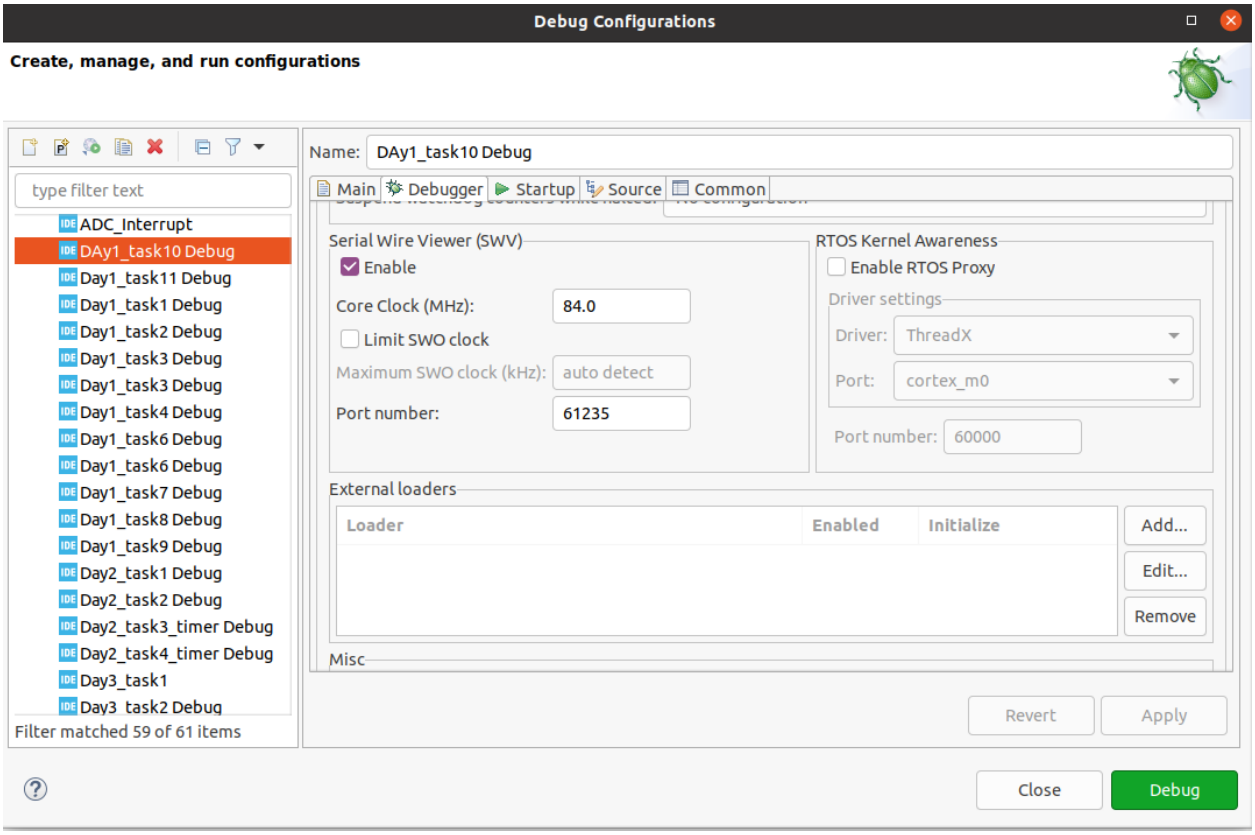



```c
#include "stm32f4xx_hal.h"
#include "stdio.h"


uint8_t switch_count = 0;
//----------------------------------------

#ifdef _GNUC_
#define PUTCHAR_PROTOTYPE int __io_putchar(int ch)
#else
#define PUTCHAR_PROTOTYPE int fputc(int ch, FILE *f)
#endif

PUTCHAR_PROTOTYPE
{
  ITM_SendChar(ch);
  return ch;
}

//-----------------------------------------
uint8_t led_state = 0;
while (1)
  {
    /* USER CODE END WHILE */
        if (HAL_GPIO_ReadPin(GPIOC, switch1_Pin) == GPIO_PIN_RESET) // Assuming the button is on
```

```
        {
          HAL_Delay(200); // Debounce delay
          if (HAL_GPIO_ReadPin(GPIOC, switch1_Pin) == GPIO_PIN_RESET) // Check if button is stil
          {
            switch_count++;
            if (led_state == 0)
            {
              HAL_GPIO_WritePin(GPIOA, led1_Pin, GPIO_PIN_SET); // Assuming the LED is on PA5
              led_state = 1;
              printf("LED ON\n");
            }
            else
            {
              HAL_GPIO_WritePin(GPIOA, led1_Pin, GPIO_PIN_RESET);
              led_state = 0;
              printf("LED OFF\n");
            }
            printf("Switch pressed %d times\n", switch_count);
            while (HAL_GPIO_ReadPin(GPIOC, switch1_Pin) == GPIO_PIN_RESET); // Wait until button
          }
        }
    /* USER CODE BEGIN 3 */
  }

 //out of the main function
  int _write(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;

  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    ITM_SendChar(*ptr++);
  }
  return len;
}
```

☐ 11.Write a Program using 2 leds and 1 switch
conditions:
if we press the switch 1st time 1st led ON and 2nd led OFF
if we press the switch 2nd time 1st led ON and 2nd led OFF
and this should be in loop.
after releasing switch led state will not change.
print on SWV ITM Data Console:
number of times switch pressed.
led1 and led2 state.
Take onBoard led and switch and external led PB7.






```
#ifdef _GNUC_
#define PUTCHAR_PROTOTYPE int __io_putchar(int ch)
#else
#define PUTCHAR_PROTOTYPE int fputc(int ch, FILE *f)
#endif

PUTCHAR_PROTOTYPE
{
```

```
  ITM_SendChar(ch);
  return ch;
}

while (1)
  {
    /* USER CODE END WHILE */
      if (HAL_GPIO_ReadPin(GPIOC, switch1_Pin) == GPIO_PIN_RESET) {
             switchPressCount++;

             printf("Switch pressed %d times", switchPressCount);

             if (switchPressCount % 2 == 1) {
                 HAL_GPIO_WritePin(GPIOA, led1_Pin, GPIO_PIN_SET);
                 HAL_GPIO_WritePin(GPIOA, led2_Pin, GPIO_PIN_RESET);
                 printf("LED1 ON, LED2 OFF \n");
             } else {
                 HAL_GPIO_WritePin(GPIOA, led1_Pin, GPIO_PIN_RESET);
                 HAL_GPIO_WritePin(GPIOA, led2_Pin, GPIO_PIN_SET);
                 printf("LED1 OFF, LED2 ON \n");
             }

             while (HAL_GPIO_ReadPin(GPIOC, switch1_Pin) == GPIO_PIN_RESET);  // Wait for button
         }
    /* USER CODE BEGIN 3 */
  }

int _write(int file, char *ptr, int len)
{
(void)file;
int DataIdx;

for (DataIdx = 0; DataIdx < len; DataIdx++)
{
  ITM_SendChar(*ptr++);
}
return len;
}
```

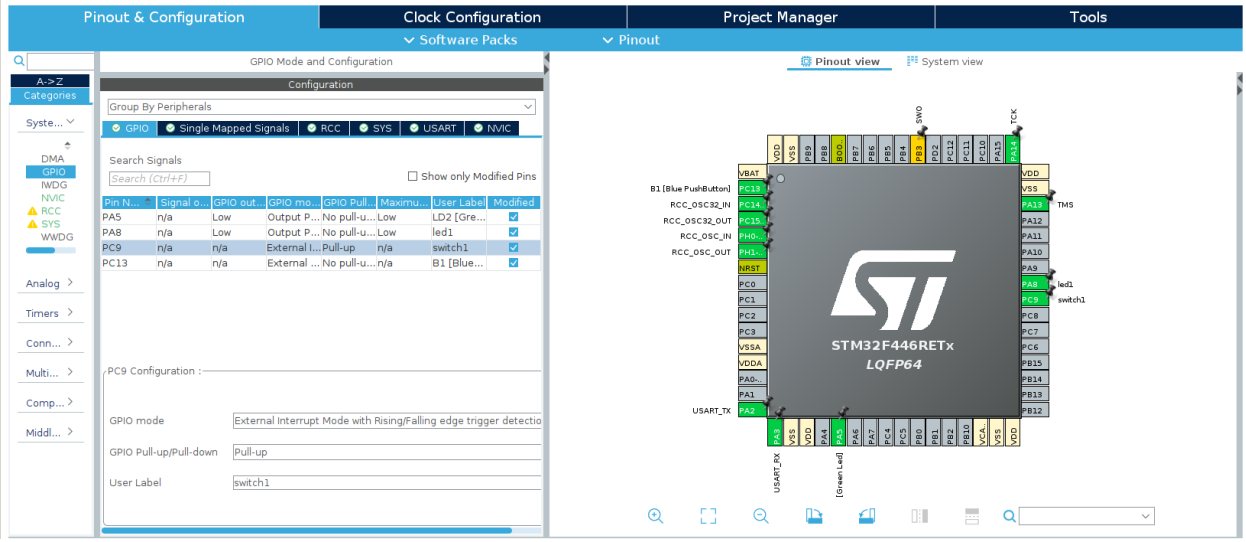
@July 29, 2024

DAY-2 (Interrupt-GPIO and Timer)

- [ ] 1.How to handle interrupt

2.Timer
3.Free RTOS
4.MOD Bus -RS232,RS485
5.LCD ,I2c

- [ ] 1.How to handle GPIO interrupt

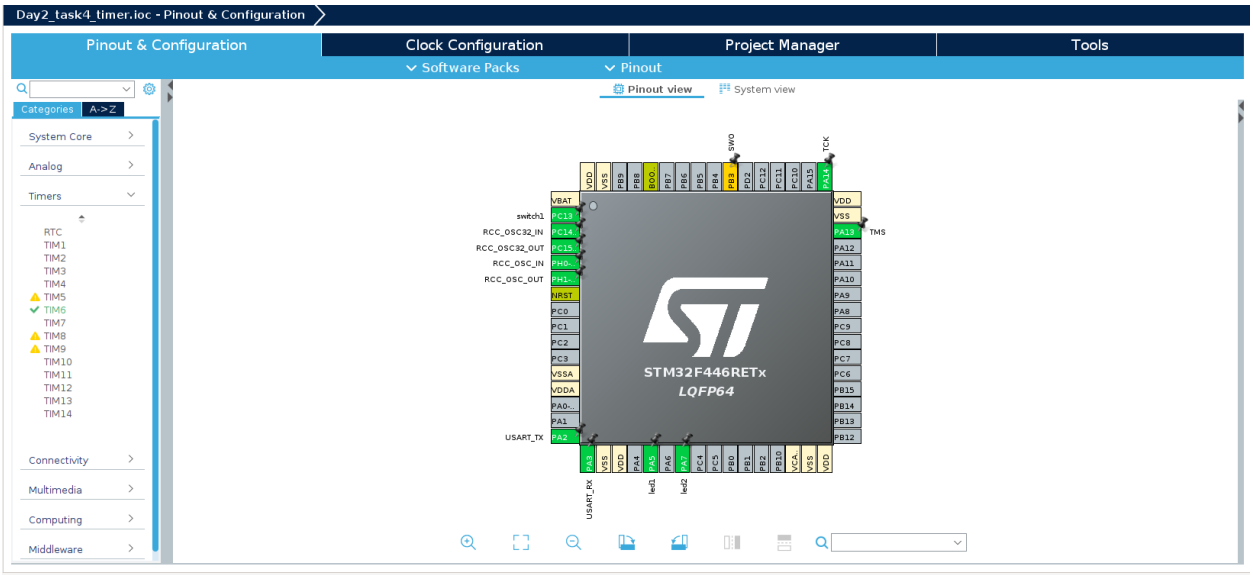



```
//stm32f4xx_it.c

void EXTI9_5_IRQHandler(void)
{
  HAL_GPIO_TogglePin(GPIOA, led1_Pin);

  HAL_GPIO_EXTI_IRQHandler(switch1_Pin);

}
```

☐ 2.How to handle Timer interrupt

```
//stm32f4xx_it.c
void TIM6_DAC_IRQHandler(void)
{
  HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
  HAL_TIM_IRQHandler(&htim6);
}
```

2-2.Both Timer interrupt and GPIO interrupt




```
//main.c
volatile uint8_t switchPressCount = 0;
volatile bool blink = false;

void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) {
    if (GPIO_Pin == switch1_Pin) {
        switchPressCount++;

        if (switchPressCount == 3) {
            switchPressCount = 0;
        }

        if (switchPressCount > 0) {
            blink = true;
            HAL_TIM_Base_Start_IT(&htim6);  // Start the timer interrupt
        } else {
            blink = false;
            HAL_TIM_Base_Stop_IT(&htim6);   // Stop the timer interrupt
            HAL_GPIO_WritePin(led2_GPIO_Port, led2_Pin, GPIO_PIN_RESET);  // Turn off LED
        }

        while (HAL_GPIO_ReadPin(switch1_GPIO_Port, switch1_Pin) == GPIO_PIN_RESET);  // Wait for
    }
}

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
    if (htim == &htim6 && blink) {
        static uint8_t blinkCount = 0;

        if (blinkCount < switchPressCount) {
            HAL_GPIO_TogglePin(led2_GPIO_Port, led2_Pin);  // Toggle LED
            blinkCount++;
        } else if (blinkCount < switchPressCount * 4) {
            HAL_GPIO_TogglePin(GPIOA, led1_Pin);  // Toggle another LED
```

```
            blinkCount++;
        } else {
            blinkCount = 0;  // Reset blink count
        }
    }
}
```

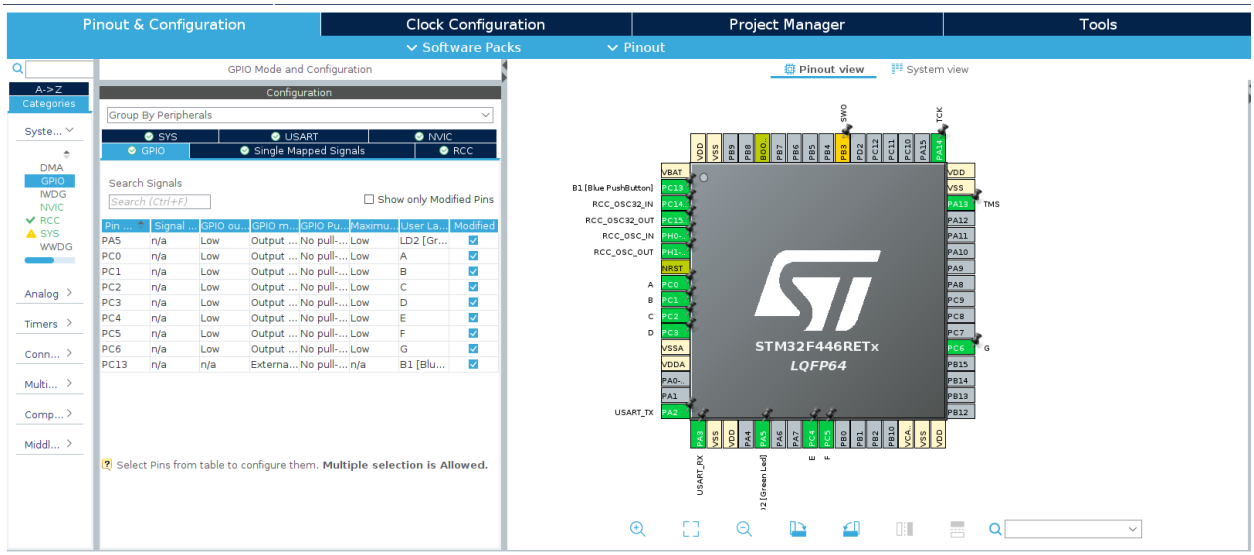
@July 30, 2024

DAY-3 (7-Segment and Keypad)

☐ 1.7seg→1
2.Key-pad→2
3.what are number pesserd in key to display
4.if press 8 it shot display other wise not perssed
5.if i pressed any key it show displayed in LCD

6.what are the number show in lcd it shoud toggel led number of times

☐ 1.Write a program to display the number 0 to 9 using LCD 7-segment and timer interrupt?
To output pin PC0 to PC6
2.Write a program to display the number 0 to 9 using LCD 7-segment and button to increment the number?
To output pin PC0 to PC6 and input PC13

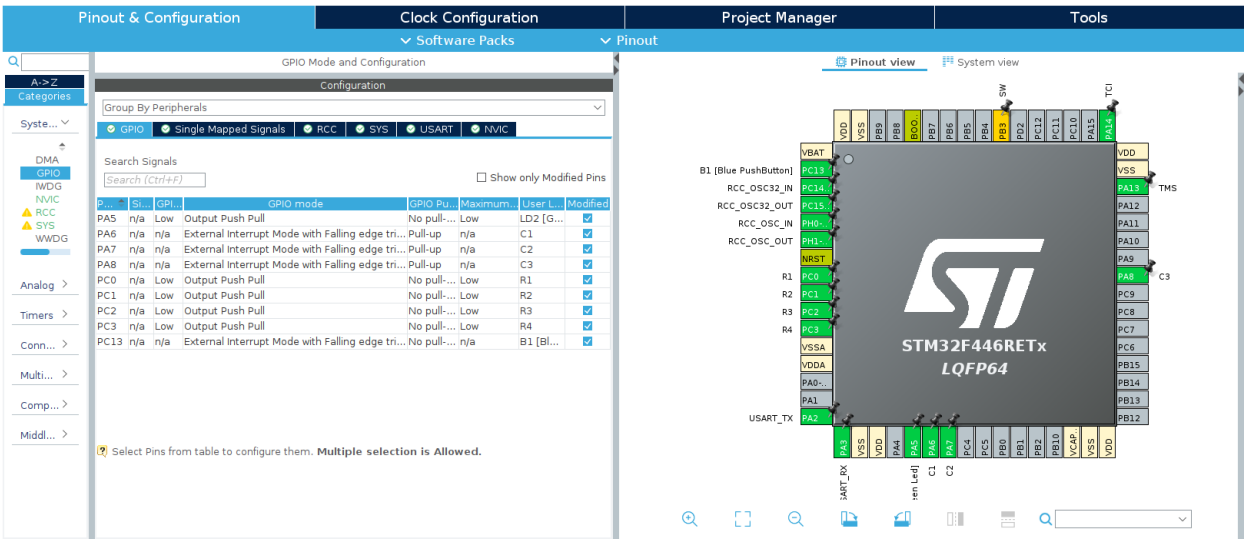



```
//in main.c file
volatile uint8_t current_number = 0;

HAL_TIM_Base_Start_IT(&htim6);

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim6) {
    if (htim6->Instance == TIM6) {  // Replace TIMx with your specific timer
//        display_number(current_number);
//        current_number = (current_number + 1) % 10;  // Increment and wrap around
    }
}
```

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) {
    if (GPIO_Pin == GPIO_PIN_13) {
        current_number = (current_number + 1) % 10;
        display_number(current_number);
    }
}

void display_number(uint8_t number) {
    switch (number) {
        case 0:
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_1, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_2, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_3, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_4, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, SET);
            break;
        case 1:
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0, SET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_1, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_2, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_3, SET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_4, SET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, SET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, SET);
            break;
        case 2:
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_1, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_2, SET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_3, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_4, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, SET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, RESET);
            break;
        case 3:
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_1, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_2, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_3, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_4, SET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, SET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, RESET);
            break;
        case 4:
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0, SET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_1, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_2, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_3, SET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_4, SET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, RESET);
```

```
            break;
        case 5:
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_1, SET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_2, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_3, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_4, SET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, RESET);
            break;
        case 6:
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_1, SET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_2, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_3, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_4, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, RESET);
            break;
        case 7:
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_1, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_2, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_3, SET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_4, SET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, SET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, SET);
            break;
        case 8:
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_1, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_2, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_3, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_4, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, RESET);
            break;
        case 9:
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_1, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_2, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_3, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_4, SET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, RESET);
            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, RESET);
            break;
    }
}
```

- [ ] 3.what are number pressed in key to display

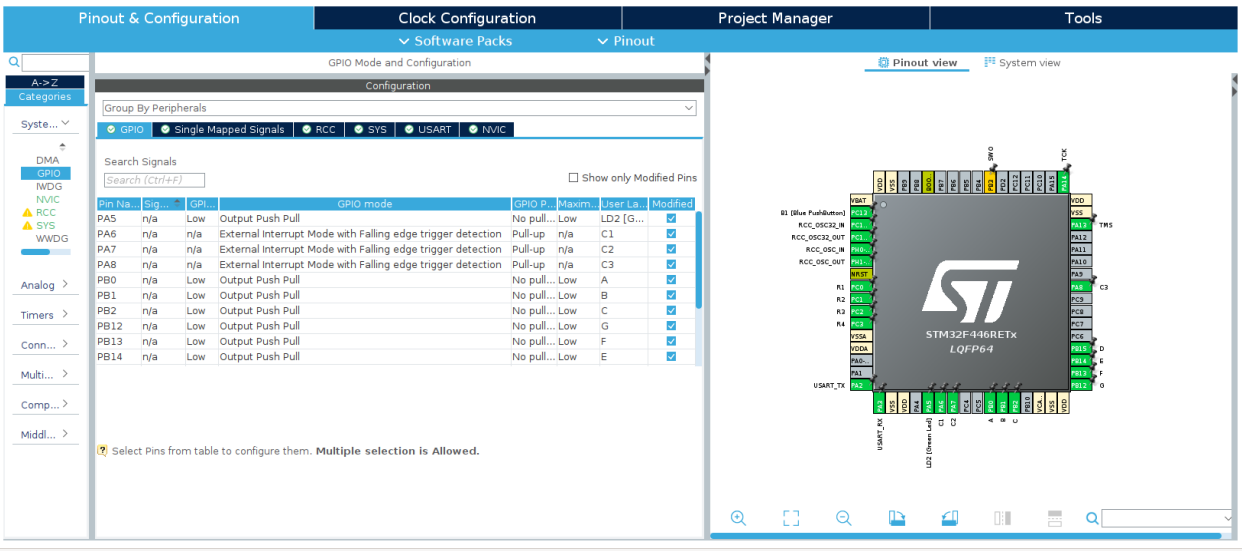


- select the external interrupt also

```
char keypad[4][3] = {
    {'1', '2', '3'},
    {'4', '5', '6'},
    {'7', '8', '9'},
    {'*', '0', '#'}
};

char pressed_key = 0;
int _write(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;

  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    ITM_SendChar(*ptr++);
  }
  return len;
}

void scan_keypad(void)
{
    // Set all rows high initially
    HAL_GPIO_WritePin(GPIOC, R1_Pin | R2_Pin | R3_Pin | R4_Pin, GPIO_PIN_SET);

    // Scan each row
    for (int row = 0; row < 4; row++)
    {
        // Set the current row pin to low
        HAL_GPIO_WritePin(GPIOC, (R1_Pin << row), GPIO_PIN_RESET);

        // Check each column
        for (int col = 0; col < 3; col++)
        {
```

```
            if (HAL_GPIO_ReadPin(GPIOA, (C1_Pin << col)) == GPIO_PIN_RESET)
            {
                // Key is pressed in this row and column
                pressed_key = keypad[row][col];
                printf("Pressed Key: %c\n", pressed_key);
                printf("Scanning Row: %d\n", row);
                printf("Checking Column: %d, Pin State: %d\n", col, HAL_GPIO_ReadPin(GPIOA, (C1_

                // Restore the row pin to high
                HAL_GPIO_WritePin(GPIOC, (R1_Pin << row), GPIO_PIN_SET);

                return; // Exit as soon as a pressed key is found
            }
        }

        // Restore the row pin to high
        HAL_GPIO_WritePin(GPIOC, (R1_Pin << row), GPIO_PIN_SET);

    }
}

void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if (GPIO_Pin == C1_Pin || GPIO_Pin == C2_Pin || GPIO_Pin == C3_Pin)
    {
        scan_keypad();
    }
}
```

- [ ] 4.write a code to displif press 8 it shot display other wise not perssed
- [ ] 5.what are number pressed in key to display in LCD




```
/------------------char keypad[4][3] = {
    {'1', '2', '3'},
```

```
    {'4', '5', '6'},
    {'7', '8', '9'},
    {'*', '0', '#'}
};

char pressed_key = 0;

//-------------------------------------------
int _write(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;

  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    ITM_SendChar(*ptr++);
  }
  return len;
}
//-----------------------------------------

/* USER CODE BEGIN 4 */
void scan_keypad(void)
{
    // Set all row pins high initially
    HAL_GPIO_WritePin(GPIOC, R1_Pin | R2_Pin | R3_Pin | R4_Pin, GPIO_PIN_SET);

    // Scan each row
    for (int row = 0; row < 4; row++)
    {
        // Set the current row pin to low
        HAL_GPIO_WritePin(GPIOC, (R1_Pin << row), GPIO_PIN_RESET);

        //printf("Scanning Row: %d\n", row);  // Debug print to indicate current row being scann

        // Check each column
        for (int col = 0; col < 3; col++)
        {
            int pinState = HAL_GPIO_ReadPin(GPIOA, (C1_Pin << col));  // Read pin state for debu

            // Debug print to show the pin state of each column
           // printf("Checking Column: %d, Pin State: %d\n", col, pinState);

            if (pinState == GPIO_PIN_RESET)
            {
                // Key is pressed in this row and column
                pressed_key = keypad[row][col];
                display_number(pressed_key); // Display the key number
                printf("Pressed Key: %c\n", pressed_key);
                printf("Detected at Row: %d, Column: %d\n", row, col);

                // Restore the row pin to high
                HAL_GPIO_WritePin(GPIOC, (R1_Pin << row), GPIO_PIN_SET);
```

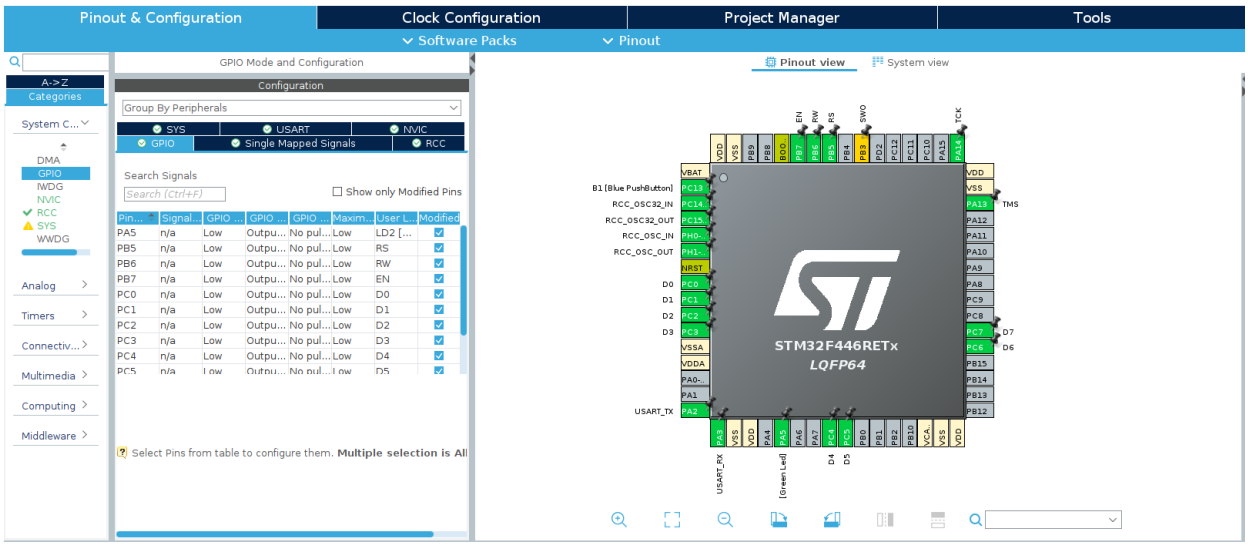
```
                return; // Exit as soon as a pressed key is found
            }
        }

        // Restore the row pin to high
        HAL_GPIO_WritePin(GPIOC, (R1_Pin << row), GPIO_PIN_SET);

    }
}

void display_number(char key) {
    uint8_t number = key - '0'; // Convert char to integer
    switch (number) {
        case 0:
            HAL_GPIO_WritePin(GPIOB, A_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, B_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, C_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, D_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, E_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, F_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, G_Pin, GPIO_PIN_SET);
            break;
        case 1:
            HAL_GPIO_WritePin(GPIOB, A_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOB, B_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, C_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, D_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOB, E_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOB, F_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOB, G_Pin, GPIO_PIN_SET);
            break;
        case 2:
            HAL_GPIO_WritePin(GPIOB, A_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, B_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, C_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOB, D_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, E_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, F_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOB, G_Pin, GPIO_PIN_RESET);
            break;
        case 3:
            HAL_GPIO_WritePin(GPIOB, A_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, B_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, C_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, D_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, E_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOB, F_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOB, G_Pin, GPIO_PIN_RESET);
            break;
        case 4:
            HAL_GPIO_WritePin(GPIOB, A_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOB, B_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, C_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, D_Pin, GPIO_PIN_SET);
```

```
            HAL_GPIO_WritePin(GPIOB, E_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOB, F_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, G_Pin, GPIO_PIN_RESET);
            break;
        case 5:
            HAL_GPIO_WritePin(GPIOB, A_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, B_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOB, C_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, D_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, E_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOB, F_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, G_Pin, GPIO_PIN_RESET);
            break;
        case 6:
            HAL_GPIO_WritePin(GPIOB, A_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, B_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOB, C_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, D_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, E_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, F_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, G_Pin, GPIO_PIN_RESET);
            break;
        case 7:
            HAL_GPIO_WritePin(GPIOB, A_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, B_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, C_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, D_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOB, E_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOB, F_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOB, G_Pin, GPIO_PIN_SET);
            break;
        case 8:
            HAL_GPIO_WritePin(GPIOB, A_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, B_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, C_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, D_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, E_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, F_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, G_Pin, GPIO_PIN_RESET);
            break;
        case 9:
            HAL_GPIO_WritePin(GPIOB, A_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, B_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, C_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, D_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, E_Pin, GPIO_PIN_SET);
            HAL_GPIO_WritePin(GPIOB, F_Pin, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOB, G_Pin, GPIO_PIN_RESET);
            break;
        default:
            // If key is not a valid digit, turn off all segments
            HAL_GPIO_WritePin(GPIOB, A_Pin | B_Pin | C_Pin | D_Pin | E_Pin | F_Pin | G_Pin, GPIO
            break;
    }
```

```
}


void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if (GPIO_Pin == C1_Pin || GPIO_Pin == C2_Pin || GPIO_Pin == C3_Pin)
    {
        scan_keypad();
    }
}
```

DAY-4 (LCD and UART)

- [ ] 1. Write a code to display string on lcd in 8 bit mode




```
//in main.c file

 LCD_Init();

 LCD_String("Hello world");

//functions
void LCD_String(char str[])
{
    for (int i = 0; str[i] != '\0'; i++)
    {
        LCD_Data(str[i]);
    }
}

void LCD_Init(void)
{
    PORTS_Init();
```

```
    HAL_Delay(30); /* initialization sequence */
    LCD_Command(0x30);
    HAL_Delay(10);
    LCD_Command(0x30);
    HAL_Delay(1);
    LCD_Command(0x30);
    LCD_Command(0x38);      /* set 8-bit data, 2-line, 5x7 font */
    LCD_Command(0x06);      /* move cursor right after each char */
    LCD_Command(0x01);      /* clear screen, move cursor to home */
    LCD_Command(0x0F);      /* turn on display, cursor blinking */
}

void PORTS_Init(void)
{
    __HAL_RCC_GPIOB_CLK_ENABLE();    /* enable GPIOB clock */
    __HAL_RCC_GPIOC_CLK_ENABLE();    /* enable GPIOC clock */

    GPIO_InitTypeDef GPIO_InitStruct = {0};

    /* Configure GPIO pins : RS_Pin RW_Pin EN_Pin */
    GPIO_InitStruct.Pin = RS_Pin | RW_Pin | EN_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

    /* Turn off EN and R/W */
    HAL_GPIO_WritePin(GPIOB, EN_Pin | RW_Pin, GPIO_PIN_RESET);

    /* Configure GPIO pins : PC0 to PC7 as output for data lines */
    GPIO_InitStruct.Pin = GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3 |
                          GPIO_PIN_4 | GPIO_PIN_5 | GPIO_PIN_6 | GPIO_PIN_7;
    HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
}

void LCD_Command(uint8_t command)
{
    HAL_GPIO_WritePin(GPIOB, RS_Pin | RW_Pin, GPIO_PIN_RESET); /* RS = 0, RW = 0 */
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_All, GPIO_PIN_RESET); /* Clear previous data */
    HAL_GPIO_WritePin(GPIOC, command, GPIO_PIN_SET);        /* Put command on data bus */
    HAL_GPIO_WritePin(GPIOB, EN_Pin, GPIO_PIN_SET); /* Pulse EN high */
    delayMs(0);
    HAL_GPIO_WritePin(GPIOB, EN_Pin, GPIO_PIN_RESET); /* Clear EN */
    if (command < 4)
        HAL_Delay(2); /* Commands 1 and 2 need up to 1.64ms */
    else
        HAL_Delay(1); /* All others 40us */
}

void LCD_Data(uint8_t data)
{
    HAL_GPIO_WritePin(GPIOB, RS_Pin, GPIO_PIN_SET);  /* RS = 1 */
    HAL_GPIO_WritePin(GPIOB, RW_Pin, GPIO_PIN_RESET); /* RW = 0 */
    HAL_Delay(1);
```

```
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_All, GPIO_PIN_RESET); /* Clear previous data */
    HAL_GPIO_WritePin(GPIOC, data, GPIO_PIN_SET);           /* Put data on data bus */
    HAL_GPIO_WritePin(GPIOB, EN_Pin, GPIO_PIN_SET); /* Pulse EN high */
    delayMs(0);
    HAL_GPIO_WritePin(GPIOB, EN_Pin, GPIO_PIN_RESET); /* Clear EN */
}

/* delay n milliseconds (assuming 16 MHz CPU clock) */
void delayMs(int n)
{
    HAL_Delay(n);
}
```

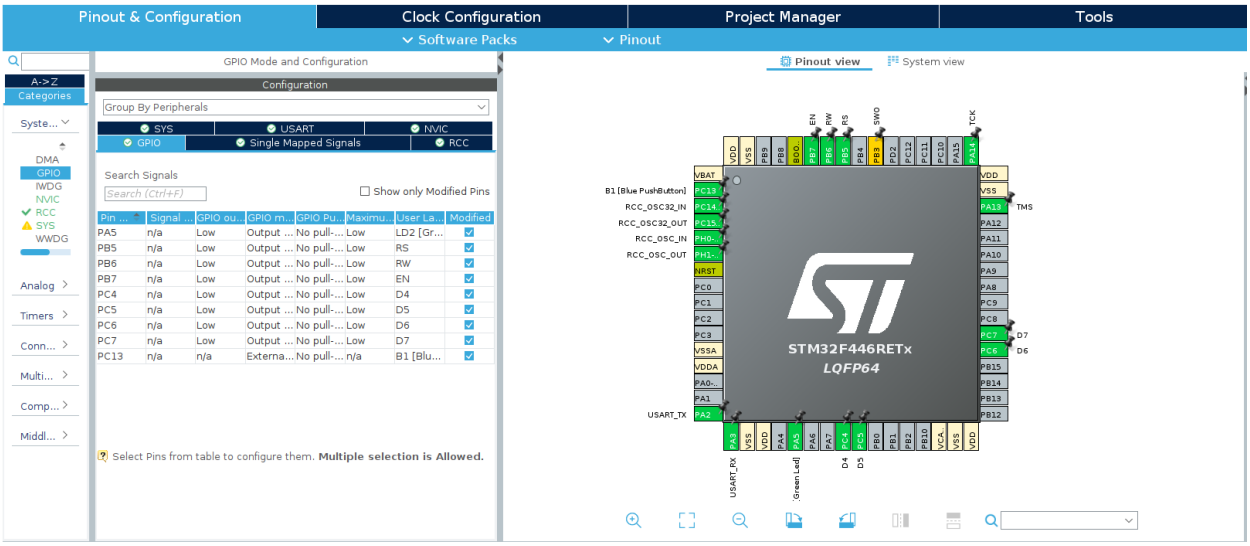
☐ 2. Write a code to set the cursor and display data on both the lines of LCD using 8-bit mode

```
LCD_Init();
 /* Display strings on both lines of the LCD */
LCD_SetCursor(0, 0);
LCD_String("DEVA N");
LCD_SetCursor(1, 0);
LCD_String("LCD Line2 Test");
//Functions
void LCD_String(char str[])
{
    for (int i = 0; str[i] != '\0'; i++)
    {
        LCD_Data(str[i]);
    }
}

void LCD_Init(void)
{
    PORTS_Init();
    HAL_Delay(30); /* initialization sequence */
    LCD_Command(0x30);
    HAL_Delay(10);
    LCD_Command(0x30);
    HAL_Delay(1);
    LCD_Command(0x30);
    LCD_Command(0x38);      /* set 8-bit data, 2-line, 5x7 font */
    LCD_Command(0x06);      /* move cursor right after each char */
    LCD_Command(0x01);      /* clear screen, move cursor to home */
    LCD_Command(0x0F);      /* turn on display, cursor blinking */
}

void LCD_SetCursor(uint8_t row, uint8_t col)
{
    uint8_t address;
    if(row == 0)
        address = LCD_LINE_1 + col;
    else
        address = LCD_LINE_2 + col;
```

```
    LCD_Command(address);
}

void PORTS_Init(void)
{
    __HAL_RCC_GPIOB_CLK_ENABLE();    /* enable GPIOB clock */
    __HAL_RCC_GPIOC_CLK_ENABLE();    /* enable GPIOC clock */

    GPIO_InitTypeDef GPIO_InitStruct = {0};

    /* Configure GPIO pins : RS_Pin RW_Pin EN_Pin */
    GPIO_InitStruct.Pin = RS_Pin | RW_Pin | EN_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

    /* Turn off EN and R/W */
    HAL_GPIO_WritePin(GPIOB, EN_Pin | RW_Pin, GPIO_PIN_RESET);

    /* Configure GPIO pins : PC0 to PC7 as output for data lines */
    GPIO_InitStruct.Pin = GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3 |
                          GPIO_PIN_4 | GPIO_PIN_5 | GPIO_PIN_6 | GPIO_PIN_7;
    HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
}

void LCD_Command(uint8_t command)
{
    HAL_GPIO_WritePin(GPIOB, RS_Pin | RW_Pin, GPIO_PIN_RESET); /* RS = 0, RW = 0 */
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_All, GPIO_PIN_RESET); /* Clear previous data */
    HAL_GPIO_WritePin(GPIOC, command, GPIO_PIN_SET);        /* Put command on data bus */
    HAL_GPIO_WritePin(GPIOB, EN_Pin, GPIO_PIN_SET); /* Pulse EN high */
    delayMs(0);
    HAL_GPIO_WritePin(GPIOB, EN_Pin, GPIO_PIN_RESET); /* Clear EN */
    if (command < 4)
        HAL_Delay(2); /* Commands 1 and 2 need up to 1.64ms */
    else
        HAL_Delay(1); /* All others 40us */
}

void LCD_Data(uint8_t data)
{
    HAL_GPIO_WritePin(GPIOB, RS_Pin, GPIO_PIN_SET);  /* RS = 1 */
    HAL_GPIO_WritePin(GPIOB, RW_Pin, GPIO_PIN_RESET); /* RW = 0 */
    HAL_Delay(1);

    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_All, GPIO_PIN_RESET); /* Clear previous data */
    HAL_GPIO_WritePin(GPIOC, data, GPIO_PIN_SET);           /* Put data on data bus */
    HAL_GPIO_WritePin(GPIOB, EN_Pin, GPIO_PIN_SET); /* Pulse EN high */
    delayMs(0);
    HAL_GPIO_WritePin(GPIOB, EN_Pin, GPIO_PIN_RESET); /* Clear EN */
}
```

```
/* delay n milliseconds (assuming 16 MHz CPU clock) */
void delayMs(int n)
{
    HAL_Delay(n);
}
```

☐ 3.Write a code to display string on both line lcd in 4 bit mode



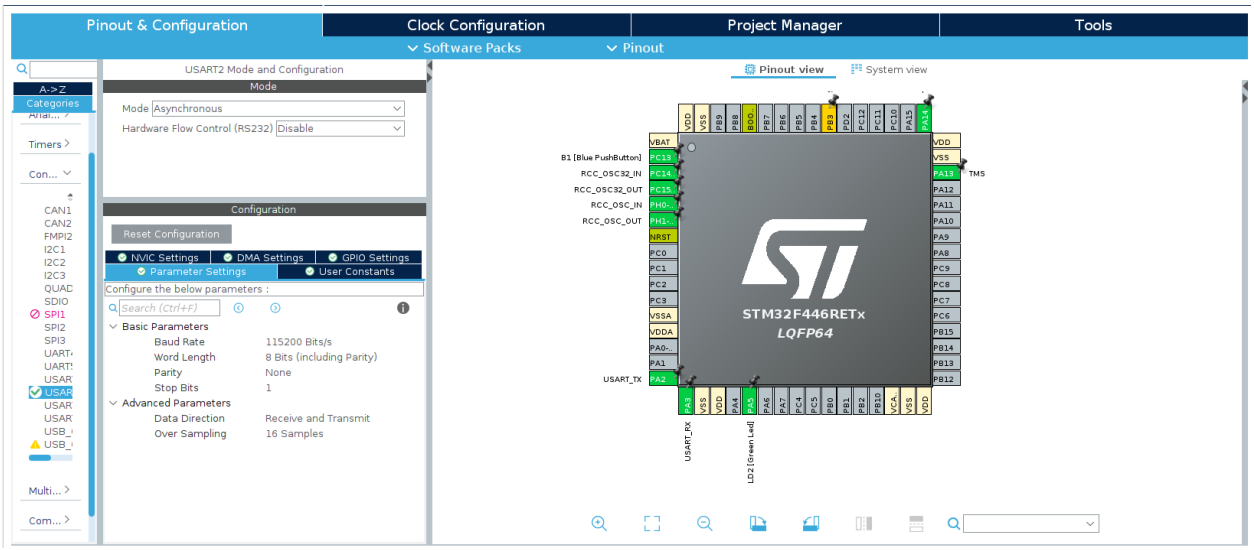

```
 LCD_Init();

  /* Display strings on both lines of the LCD */
  LCD_SetCursor(0, 0);
  LCD_String("DEVA N");

  LCD_SetCursor(1, 0);
  LCD_String("LCD 4-Bit Test");

//Functions
void LCD_Init(void)
{
    HAL_Delay(30); /* initialization sequence */
    LCD_Command(0x02); /* Initialize in 4-bit mode */
    LCD_Command(0x28); /* 2-line, 5x7 font */
    LCD_Command(0x0C); /* Display ON, Cursor OFF */
    LCD_Command(0x06); /* Increment cursor */
    LCD_Command(0x01); /* Clear display */
    HAL_Delay(2);      /* Delay for clear display */
}

void LCD_SetCursor(uint8_t row, uint8_t col)
{
    uint8_t address = (row == 0) ? (LCD_LINE_1 + col) : (LCD_LINE_2 + col);
    LCD_Command(address);
}
```

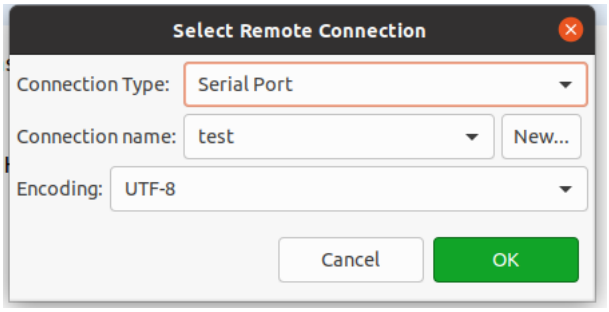
```
void LCD_String(char *str)
{
    while (*str)
    {
        LCD_Data(*str++);
    }
}

void LCD_Command(uint8_t cmd)
{
    HAL_GPIO_WritePin(GPIOB, RS_Pin, GPIO_PIN_RESET); /* RS = 0 for command */
    LCD_SendNibble(cmd >> 4); /* Send higher nibble */
    LCD_SendNibble(cmd & 0x0F); /* Send lower nibble */
    if (cmd < 4)
        HAL_Delay(2); /* Commands 1 and 2 need up to 1.64ms */
    else
        HAL_Delay(1); /* All others 40us */
}

void LCD_Data(uint8_t data)
{
    HAL_GPIO_WritePin(GPIOB, RS_Pin, GPIO_PIN_SET); /* RS = 1 for data */
    LCD_SendNibble(data >> 4); /* Send higher nibble */
    LCD_SendNibble(data & 0x0F); /* Send lower nibble */
    HAL_Delay(1);
}

void LCD_SendNibble(uint8_t nibble)
{
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_4, (nibble & 0x01) ? GPIO_PIN_SET : GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, (nibble & 0x02) ? GPIO_PIN_SET : GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, (nibble & 0x04) ? GPIO_PIN_SET : GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7, (nibble & 0x08) ? GPIO_PIN_SET : GPIO_PIN_RESET);

    HAL_GPIO_WritePin(GPIOB, EN_Pin, GPIO_PIN_SET);
    HAL_Delay(1);
    HAL_GPIO_WritePin(GPIOB, EN_Pin, GPIO_PIN_RESET);
    HAL_Delay(1);
}
```

- [ ] 4.Write a code to scroll the data on the LCD

- [ ] 5.Write a code to display the special character ‘alpha’ , ‘beta’ ,’pie’ and ‘ohm symbol’ on LCD

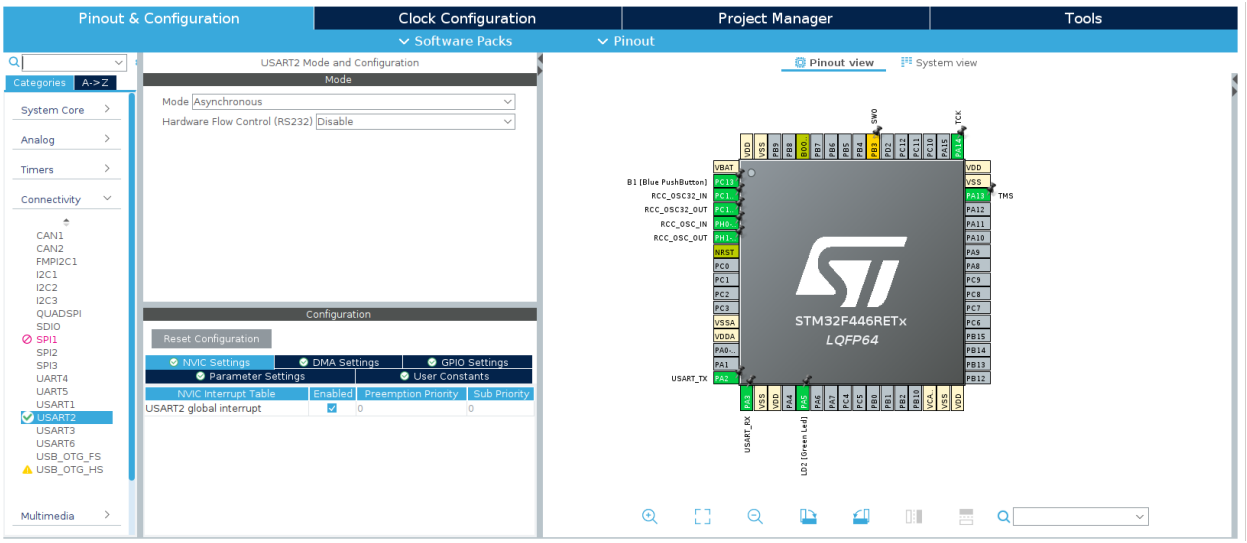
- [ ] 6. Write a code to transfer and receive data



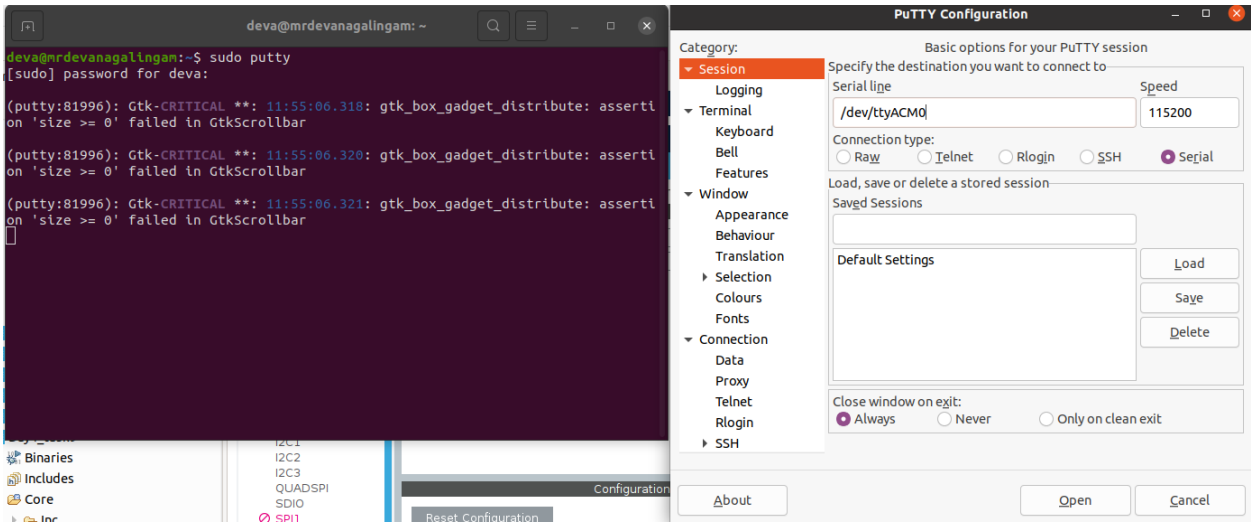



```
   // Transmit data
char *msg = "Hello, UART!\r\n";
HAL_UART_Transmit(&huart2, (uint8_t*)msg, strlen(msg), HAL_MAX_DELAY);

  // Receive data
uint8_t rxData[10];
HAL_UART_Receive(&huart2, rxData, 10, HAL_MAX_DELAY);
```

☐ 7. Write a code to control the LED PA5 using UART when we press cha ‘N’ or ‘n’ then led will turn on and if we press char ‘F’ or ‘f’ then led will turn off.




```
uint8_t received;
 // Enable the UART receive interrupt
 HAL_UART_Receive_IT(&huart2, &received, 1);

void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    if (huart->Instance == USART2)
    {
        if (received == 'F' || received == 'f')
        {
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
        }
        else if (received == 'N' || received == 'n')
        {
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
        }

        // Re-enable UART receive interrupt
        HAL_UART_Receive_IT(&huart2, &received, 1);
    }
}
```

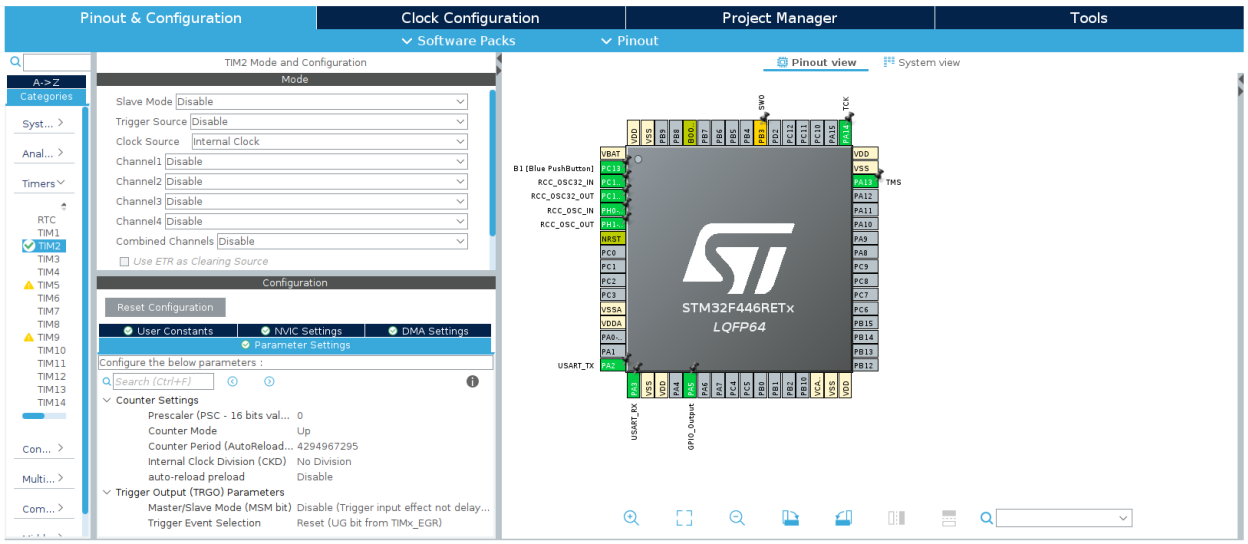


- [ ] 8.Write a code to transfer and receive data between two STM32 boards

- [ ] 9.Write a code to display data written in UART on LCD

DAY-5 (ACD and PWM)

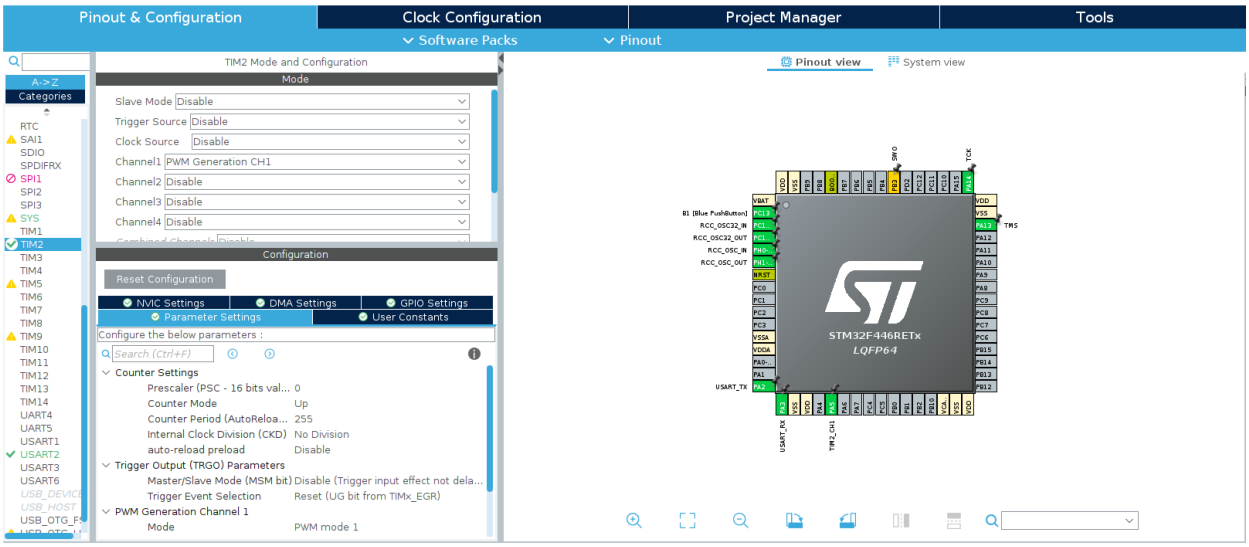
- [ ] 1. Write a function that will create delay in millisecond using systick delay




```
static volatile uint32_t timing_delay;


void SysTick_Init(void) {
    // Assuming system clock is 80 MHz
    // This will give you a 1 microsecond tick with a 1MHz SysTick clock
    uint32_t tick_rate = HAL_RCC_GetHCLKFreq() / 1000000; // 1MHz
```

```
    // Configure SysTick to interrupt every 1 microsecond
    SysTick_Config(tick_rate);
}

    while (1) {
        /* USER CODE END WHILE */
        HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
        //delay_us(1000);
        delay_ns(50);
        /* USER CODE BEGIN 3 */
    }

/* Delay function for microseconds */
void delay_us(uint32_t microseconds) {
    // Set timing delay in microseconds
    timing_delay = microseconds;

    // Wait until timing_delay reaches zero
    while (timing_delay != 0);
}

/* Delay function for nanoseconds */
void delay_ns(uint32_t nanoseconds) {
    // Calculate number of ticks needed for the delay
    // SysTick is configured for 1MHz, so each tick is 1 microsecond
    // For nanoseconds, divide by 1000
    uint32_t ticks = (nanoseconds + 999) / 1000; // Approximation

    // Set timing delay in microseconds
    timing_delay = ticks;

    // Wait until timing_delay reaches zero
    while (timing_delay != 0);
}
```

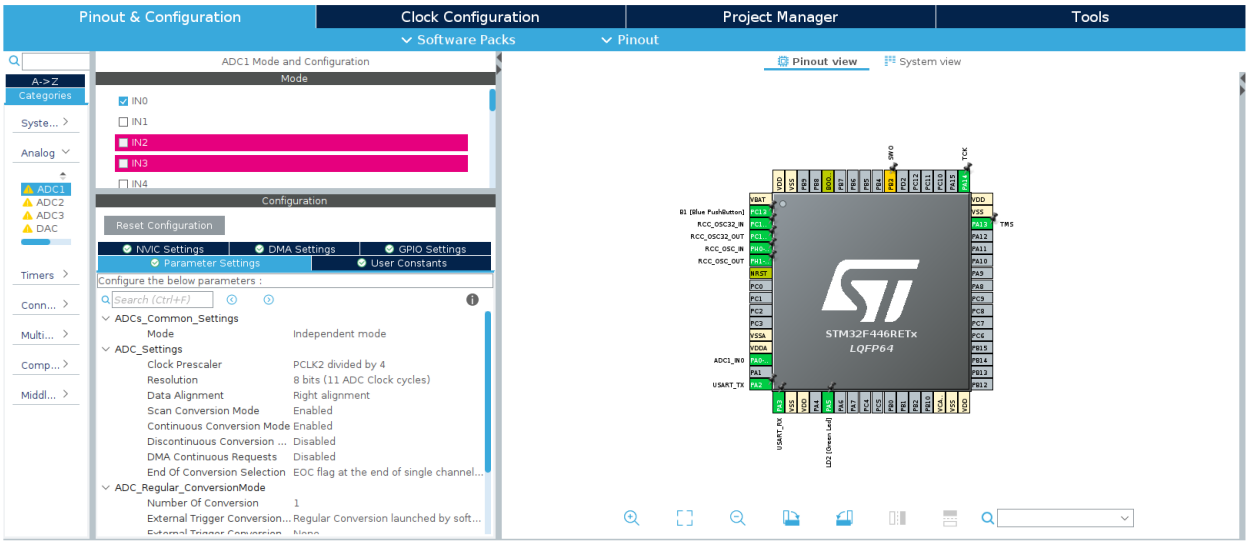
☐ 2. Write a code to change the intensity of light it decreases from 100 % to 25 % after one second




```
  HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);

while (1)
  {
    /* USER CODE END WHILE */
        for (int i = 0; i < 255; i++) {
            __HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, i);
            HAL_Delay(10);
        }
        for (int i = 255; i > 0; i--) {
            __HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, i);
            HAL_Delay(10);
        }
    /* USER CODE BEGIN 3 */
  }
```

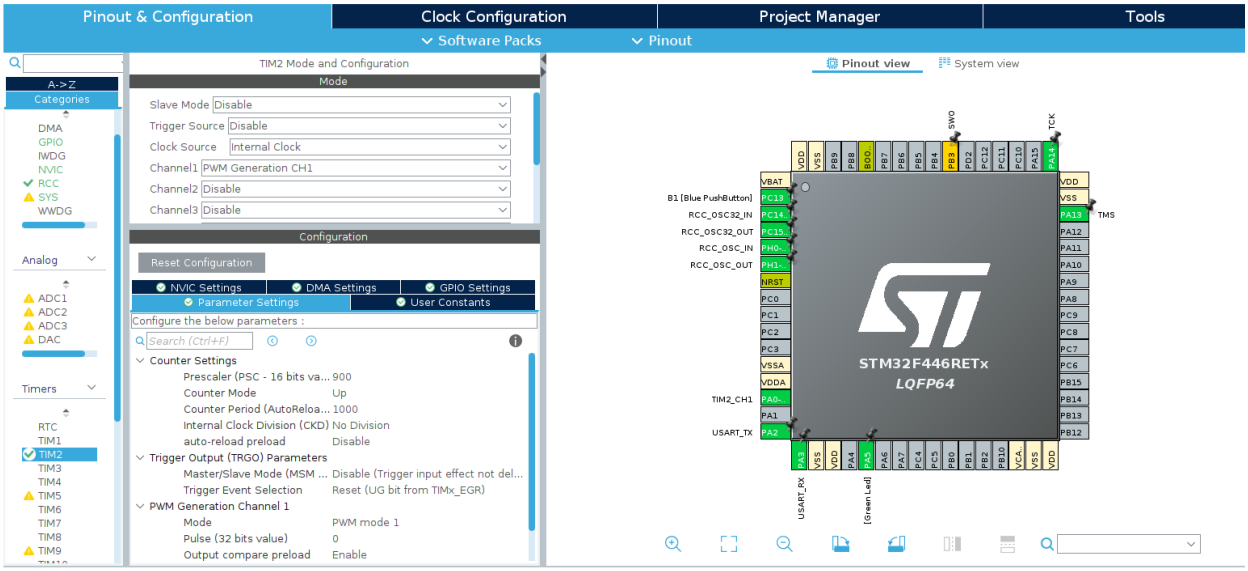
- [ ] 3. Write a code to read analog value from POT and according to value intensity of the light should change.

- [ ] 4.Write a code to read analog value from KY-015 module to print the value of Temperature and Humidity.




```
uint8_t adc_value;

    while (1) {
        /* USER CODE END WHILE */
        HAL_ADC_Start(&hadc1);
        if (HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY) == HAL_OK) {
            adc_value = HAL_ADC_GetValue(&hadc1);

            float temperature = (adc_value * 5 / 255.0) / 0.05;
            float humidity = (((adc_value * 5 / 255.0) - 0.8) * 100) / 2.5;
            printf("Temp:%.2f C, Hum:%.2f %%\r\n", temperature, humidity);
        }
        HAL_Delay(100);
        /* USER CODE BEGIN 3 */
    }
//Function to print in SWV ITM data console
int _write(int file, char *ptr, int len)
{
(void)file;
int DataIdx;

for (DataIdx = 0; DataIdx < len; DataIdx++)
{
  ITM_SendChar(*ptr++);
}
return len;
}
```

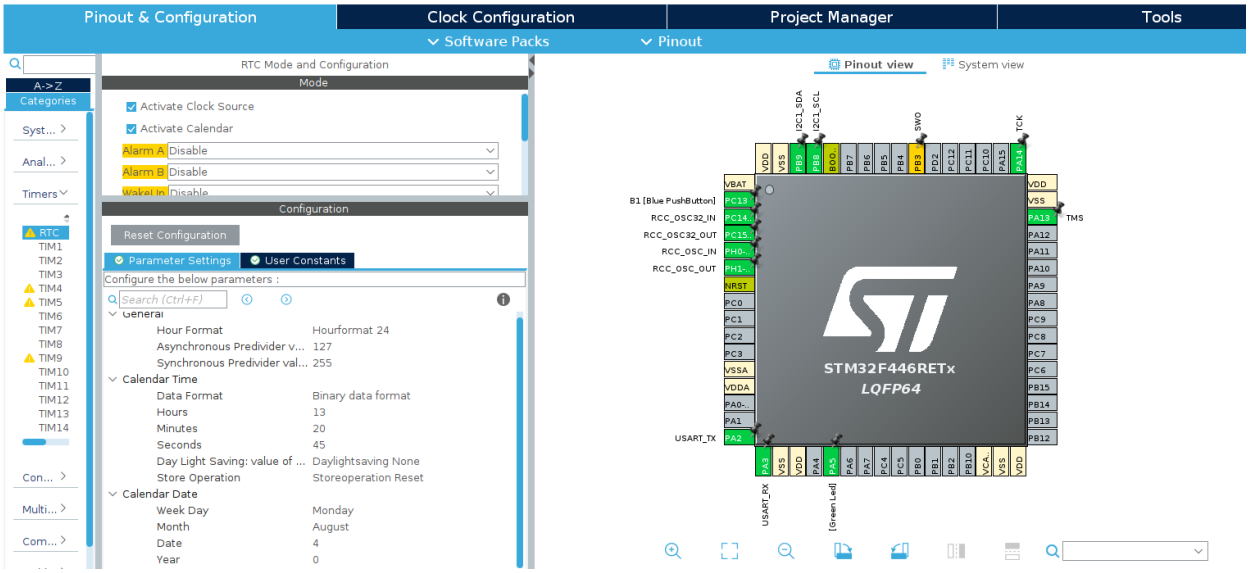
- [ ] 5.Write a code to Control the Servo motor

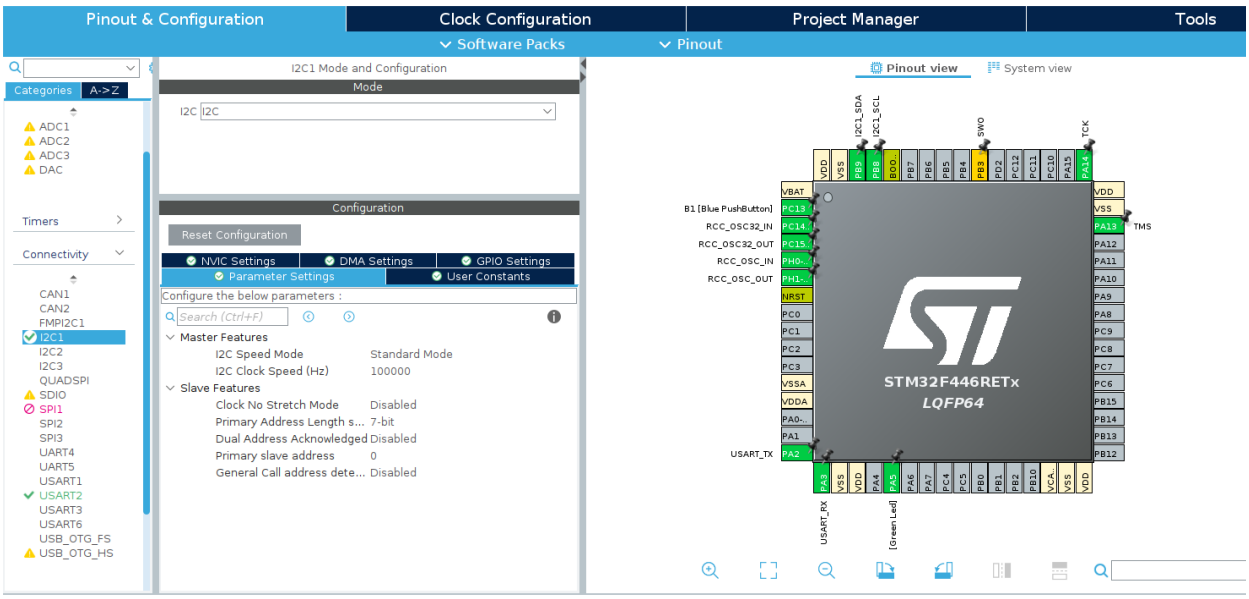



```
 HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);

while (1)
 {
   /* USER CODE END WHILE */
       htim2.Instance->CCR1 = 25;  // duty cycle is .5 ms
       HAL_Delay(2000);
       htim2.Instance->CCR1 = 100;  // duty cycle is 1.5 ms
       HAL_Delay(2000);
       htim2.Instance->CCR1 = 150;  // duty cycle is 2.5 ms
       HAL_Delay(2000);
   /* USER CODE BEGIN 3 */
 }
```

DAY-6 (RTC and I2C)

- [ ] 1. Wrtie a code to integrate I2C and display date and time with internal RTC






```
#define I2C_ADDRESS_LCD 0x27 << 1

/* USER CODE BEGIN PFP */
void set_time(void);
void get_time(void);
void display_time(void);
void lcd_send_cmd(char cmd);
void lcd_send_data(char data);
void lcd_init(void);
void lcd_send_string(char *str);
/* USER CODE END PFP */

/* USER CODE BEGIN 0 */
char time_str[9];
```

```
char date_str[11];
/* USER CODE END 0 */

/* USER CODE BEGIN 2 */
lcd_init();
set_time();
/* USER CODE END 2 */

while (1)
  {
      /* Get and display the time and date */
         get_time();
         display_time();

         /* Add a delay to update the display every second */
         HAL_Delay(1000);
  }

void lcd_send_cmd(char cmd)
{
    char data_u, data_l;
    uint8_t data_t[4];
    data_u = (cmd & 0xf0);
    data_l = ((cmd << 4) & 0xf0);
    data_t[0] = data_u | 0x0C;  // en=1, rs=0
    data_t[1] = data_u | 0x08;  // en=0, rs=0
    data_t[2] = data_l | 0x0C;  // en=1, rs=0
    data_t[3] = data_l | 0x08;  // en=0, rs=0
    HAL_I2C_Master_Transmit(&hi2c1, I2C_ADDRESS_LCD, (uint8_t *)data_t, 4, 100);
}

void lcd_send_data(char data)
{
    char data_u, data_l;
    uint8_t data_t[4];
    data_u = (data & 0xf0);
    data_l = ((data << 4) & 0xf0);
    data_t[0] = data_u | 0x0D;  // en=1, rs=1
    data_t[1] = data_u | 0x09;  // en=0, rs=1
    data_t[2] = data_l | 0x0D;  // en=1, rs=1
    data_t[3] = data_l | 0x09;  // en=0, rs=1
    HAL_I2C_Master_Transmit(&hi2c1, I2C_ADDRESS_LCD, (uint8_t *)data_t, 4, 100);
}

void lcd_init(void)
{
    HAL_Delay(50);  // Wait for >40ms after VCC rises to 2.7V
    lcd_send_cmd(0x30);  // Wake up
    HAL_Delay(5);  // Wait for >4.1ms
    lcd_send_cmd(0x30);  // Wake up #2
    HAL_Delay(1);  // Wait for >100us
    lcd_send_cmd(0x30);  // Wake up #3
    HAL_Delay(10);
    lcd_send_cmd(0x20);  // 4-bit mode
```

```
    HAL_Delay(10);

    // Function set: 4-bit mode, 2 line, 5x8 dots
    lcd_send_cmd(0x28);
    HAL_Delay(1);

    // Display on, cursor off, blink off
    lcd_send_cmd(0x0C);
    HAL_Delay(1);

    // Clear display
    lcd_send_cmd(0x01);
    HAL_Delay(2);

    // Entry mode set: increment automatically, no shift
    lcd_send_cmd(0x06);
    HAL_Delay(1);
}

void lcd_clear(void)
{
    lcd_send_cmd(0x01);  // Clear display
    HAL_Delay(2);  // Wait for the command to complete
}

void lcd_send_string(char *str)
{
    while (*str) lcd_send_data(*str++);
}

void set_time(void)
{
    RTC_TimeTypeDef sTime;
    RTC_DateTypeDef sDate;
    sTime.Hours = 0x10;  // set hours
    sTime.Minutes = 0x20;  // set minutes
    sTime.Seconds = 0x30;  // set seconds
    sTime.DayLightSaving = RTC_DAYLIGHTSAVING_NONE;
    sTime.StoreOperation = RTC_STOREOPERATION_RESET;
    if (HAL_RTC_SetTime(&hrtc, &sTime, RTC_FORMAT_BCD) != HAL_OK)
    {
        Error_Handler();
    }
    sDate.WeekDay = RTC_WEEKDAY_THURSDAY;  // day
    sDate.Month = RTC_MONTH_AUGUST;  // month
    sDate.Date = 0x09;  // date
    sDate.Year = 0x18;  // year
    if (HAL_RTC_SetDate(&hrtc, &sDate, RTC_FORMAT_BCD) != HAL_OK)
    {
        Error_Handler();
    }
    HAL_RTCEx_BKUPWrite(&hrtc, RTC_BKP_DR1, 0x32F2);  // backup register
}
```

```
void get_time(void)
{
    RTC_DateTypeDef gDate;
    RTC_TimeTypeDef gTime;
    /* Get the RTC current Time */
    HAL_RTC_GetTime(&hrtc, &gTime, RTC_FORMAT_BIN);
    /* Get the RTC current Date */
    HAL_RTC_GetDate(&hrtc, &gDate, RTC_FORMAT_BIN);
    /* Display time Format: hh:mm:ss */
    sprintf(time_str, "%02d:%02d:%02d", gTime.Hours, gTime.Minutes, gTime.Seconds);
    /* Display date Format: dd-mm-yy */
    sprintf(date_str, "%02d-%02d-%02d", gDate.Date, gDate.Month, 2000 + gDate.Year);
}

void display_time(void)
{
    lcd_clear();  // Clear the LCD before displaying new data
    lcd_send_cmd(0x80);  // send cursor to 0,0
    lcd_send_string(time_str);
    lcd_send_cmd(0xC0);  // send cursor to 1,0
    lcd_send_string(date_str);
}
```

- [ ] 2.Write a code to display time in 24 hrs format, date, day using external RTC.

- [ ] 3.Write a code to display time in 12 hrs format, date using external RTC.

- [ ] 4.Write a code to integrate I2C and external RTC

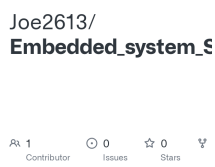
DAY-7 (SPI and RFID-RC522)

- [ ] 1.Write a code to read the RFID-RC522 id number and print the RFID ID on the console.

Embedded_system_STM32_program/Day7_task_RFID at main · Joe2613/Embedded_system_STM32_program
Contribute to Joe2613/Embedded_system_STM32_program development by creating an account on GitHub.
https://github.com/Joe2613/Embedded_system_STM32_program/tree/main/Day7_task_RFID

- [ ] 2.Write a code to read three different RFID card and trun on three different leds.

Embedded_system_STM32_program/Day7_task_RFID_GPIO at main · Joe2613/Embedded_system_STM32_program
Contribute to Joe2613/Embedded_system_STM32_program development by creating an account on GitHub.
https://github.com/Joe2613/Embedded_system_STM32_program/tree/main/Day7_task_RFID_GPIO



- [ ] 3.Write a code to read the RFID-RC522 and crossponded the RFID 3 tag mapping three different name and print the name on LCD.

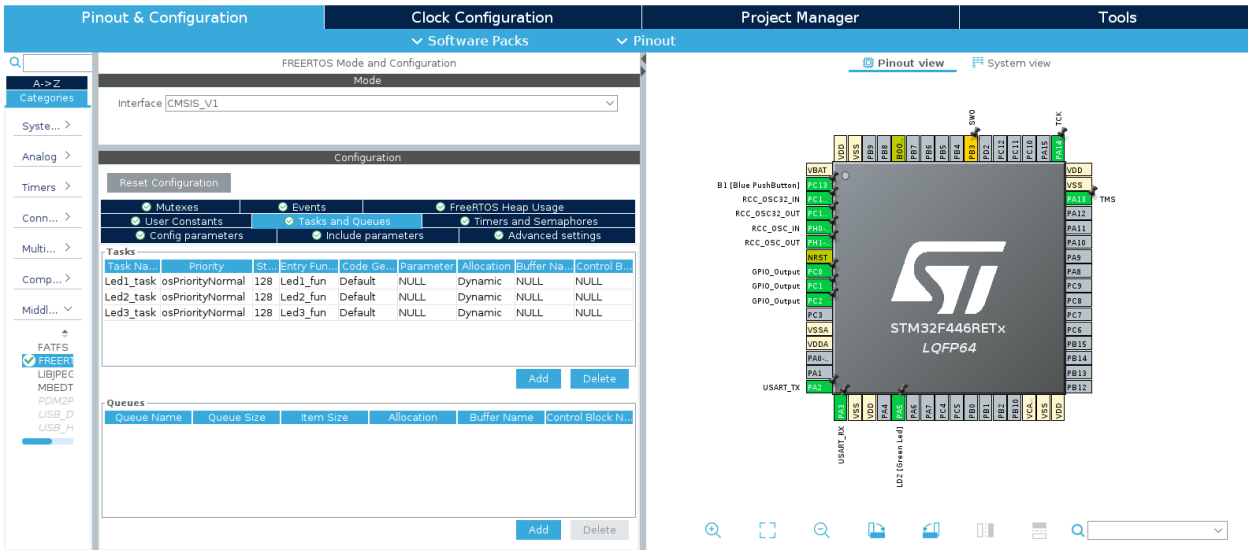
Embedded_system_STM32_program/Day7_task_RFID_LCD at main · Joe2613/Embedded_system_STM32_program

Contribute to Joe2613/Embedded_system_STM32_program development by creating an account on GitHub.

https://github.com/Joe2613/Embedded_system_STM32_program/tree/main/Day7_task_RFID_LCD

DAY-8 (FreeRTOs)

☐ 1.Take a three led and toggle each leds using with FreeRTOs and condition of LED1→ toggle after each 1sec ,LED2 → toggle after 800msec, LED3 →toggle after 400 msec.




```
//check weather thread is created are not
osThreadId Led1_taskHandle;
osThreadId Led2_taskHandle;
osThreadId Led3_taskHandle;

//check funtion are created are not
void Led1_fun(void const * argument);
void Led2_fun(void const * argument);
void Led3_fun(void const * argument);

 /* Create the thread(s) */
  /* definition and creation of Led1_task */
  osThreadDef(Led1_task, Led1_fun, osPriorityNormal, 0, 128);
  Led1_taskHandle = osThreadCreate(osThread(Led1_task), NULL);

  /* definition and creation of Led2_task */
  osThreadDef(Led2_task, Led2_fun, osPriorityNormal, 0, 128);
  Led2_taskHandle = osThreadCreate(osThread(Led2_task), NULL);

  /* definition and creation of Led3_task */
  osThreadDef(Led3_task, Led3_fun, osPriorityNormal, 0, 128);
  Led3_taskHandle = osThreadCreate(osThread(Led3_task), NULL);
```

```
  /* Start scheduler */
  osKernelStart();

  /* We should never get here as control is now taken by the scheduler */

void Led1_fun(void const * argument)
{
  /* USER CODE BEGIN 5 */
  /* Infinite loop */
  for(;;)
  {
      HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_0);
    osDelay(1000);
  }
  /* USER CODE END 5 */
}

/* USER CODE BEGIN Header_Led2_fun */
/**
* @brief Function implementing the Led2_task thread.
* @param argument: Not used
* @retval None
*/
/* USER CODE END Header_Led2_fun */
void Led2_fun(void const * argument)
{
  /* USER CODE BEGIN Led2_fun */
  /* Infinite loop */
  for(;;)
  {
      HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_1);
    osDelay(800);
  }
  /* USER CODE END Led2_fun */
}

/* USER CODE BEGIN Header_Led3_fun */
/**
* @brief Function implementing the Led3_task thread.
* @param argument: Not used
* @retval None
*/
/* USER CODE END Header_Led3_fun */
void Led3_fun(void const * argument)
{
  /* USER CODE BEGIN Led3_fun */
  /* Infinite loop */
  for(;;)
  {
      HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_2);
    osDelay(400);
  }
```

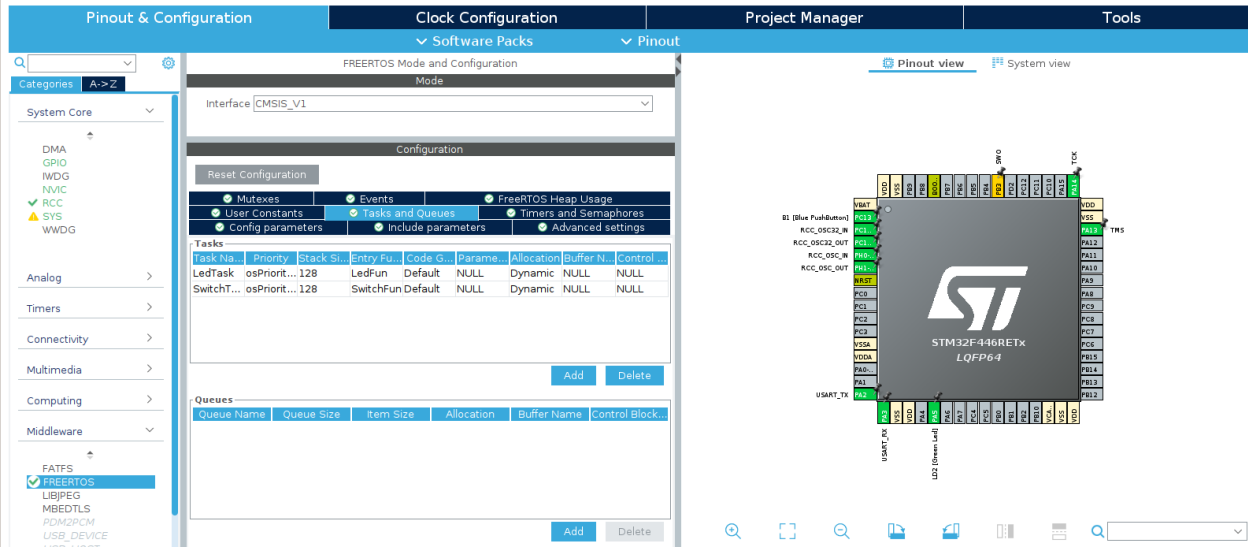
```
  /* USER CODE END Led3_fun */
}
```

☐ 2.Take a three led and toggle each leds using Without FreeRTOs with condition of LED1→ toggle after each 1sec ,LED2 → toggle after 800msec, LED3 →toggle after 400 msec.

- set GPIO pin as output PC0, PC1, PC2

```
uint32_t led1_last_toggle = 0;
uint32_t led2_last_toggle = 0;
uint32_t led3_last_toggle = 0;

while (1)
  {
    /* USER CODE END WHILE */
      uint32_t current_time = HAL_GetTick();

          // Toggle LED1 every 1 second
          if (current_time - led1_last_toggle >= 1000)
          {
            HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_0);
            led1_last_toggle = current_time;
          }

          // Toggle LED2 every 800 milliseconds
          if (current_time - led2_last_toggle >= 800)
          {
            HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_1);
            led2_last_toggle = current_time;
          }

          // Toggle LED3 every 400 milliseconds
          if (current_time - led3_last_toggle >= 400)
          {
            HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_2);
            led3_last_toggle = current_time;
          }
    /* USER CODE BEGIN 3 */
  }
```

☐ 3.Write a Program using 1 led and 1 switch
conditions:(Using FreeRTOs)
if we press the switch 1st time led will turn on and print LED ON in SWV ITM Data Console.
if we press the switch 2nd time led will turn off and print LED OFF in SWV ITM Data Console and this should be in a loop.
after releasing switch led state will not change.
also print the number of times switch pressed.
Take onboard led and switch.




```
void SwitchFun(void const * argument)
{
  /* USER CODE BEGIN SwitchFun */
  /* Infinite loop */
  for(;;)
  {
      if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13) == GPIO_PIN_RESET) // Assuming the button is on P
          {
            HAL_Delay(200); // Debounce delay
            if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13) == GPIO_PIN_RESET) // Check if button is st
            {
              switch_count++;
              led_state = !led_state;
              HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, (led_state) ? GPIO_PIN_SET : GPIO_PIN_RESET);
              printf("Switch pressed %d times\n", switch_count);
              printf("LED %s\n", (led_state) ? "ON" : "OFF");

              while (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13) == GPIO_PIN_RESET); // Wait until butt
            }
          }
          osDelay(100);
  }
  /* USER CODE END SwitchFun */
}

int _write(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;

  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    ITM_SendChar(*ptr++);
  }
```

```
  return len;
}
```