REPORT

Title: Smart Home Security System

Submission Date	2021.12.23	Major	Mechanical and control engineering
Subject	Embedded Controller	Professor	Young-Keun Kim
Name	Yeong-Won Song	Student Number	21700375
Partner Name	Jin-Su Yu	Partner Number	21700469

Contents

I. Introduction

1.1 Purpose

Design an embedded system to realize a simple smart home security system with the following design criteria.

Figure 1. Smart home system

1.2 Parts List

Parts	Specification	Quantity
Stepper Motor	28BYJ-48	1
DC motor driver	ULN2003	1
IR reflective sensor	TCRT5000	1
Ultrasonic distance sensor	HC-SR04	1
Sound sensor	SZH-EK033	1
Light intensity sensor	MSE004LSM	1
PIR motion sensor	-	1
LED	-	1
7-Segment	S-5101ASR	1
Zigbee module	XB24CZ7WIT-004	2
Zigbee Shield	DFR0015	2
Bluetooth	HC-06	1
breadboard	-	1
NUCLEO-F411RE	Arm®(a) Cortex®-M4 with FPU	2

Table 1. Part List

1.3 System Mode

MODE	Description
Normal Mode (NORM_MODE)	Pressing MODE button (B1 of MCU_1) toggles from NORM mode ←→SECUR _mode
Security Mode (SECUR_MODE):	Pressing STOP button (B1 of MCU_2) resets from SIREN_mode to SECUR_mode.
SIREN Mode (SIREN_MODE):	Turns on the siren after 5 seconds from SIREN_TRG trigger generation.

Table 2. System mode

1.4 Window Security Mode

MODE	Description
Normal Mode (NORM_MODE)	During the daytime: - Automatically opens the curtain. During the night time: - Closes the curtain. Does not generate SIREN _TRG trigger.
Security Mode (SECUR_MODE):	If the window is opened or the glass is chattered unexpectedly: - Opens the curtain and generates the SIREN _TRG trigger

Table 3. Window security mode

1.5 Front Door

MODE	Description
Normal Mode (NORM_MODE)	When a person is detected within the given distance from the door(outside): - Keep turning on the door light as long as the person is present. When a person is detected inside the house nearby the door: - Turn on the door light for given period of time and does not generate the SIREN _TRG trigger.

Security Mode (SECUR_MODE):	When a person is detected within the given distance from the door(outside): - Keep turning on the door light as long as the person is present and sends the VISITOR_LOG message to the server. When a person is detected inside the house nearby the door: - Keep turning on the door light and generates the SIREN _TRG trigger.

Table 4. Front Door

II. Procedure

2.1 Problem Description

The system is consisted of a server, a sensor unit, and an actuation unit.

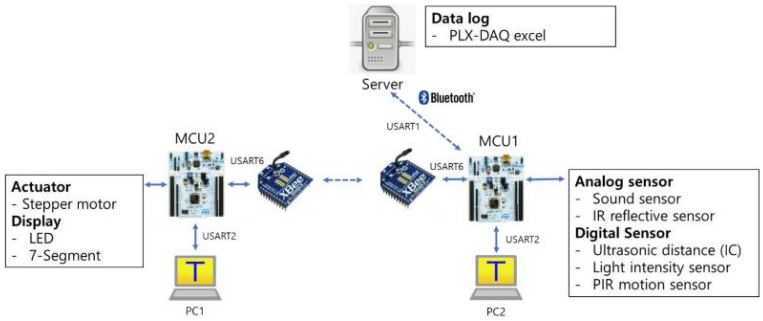

Figure 2. problem description

- **Server**

  Receives each sensor values and necessary status from the Sensor Unit MCU, every 1 second
  I was able to decide what needs to be transmitted to the server.

  For this design problem, I used PLX-DAQ for excel as the data logging.

Download Link is here.

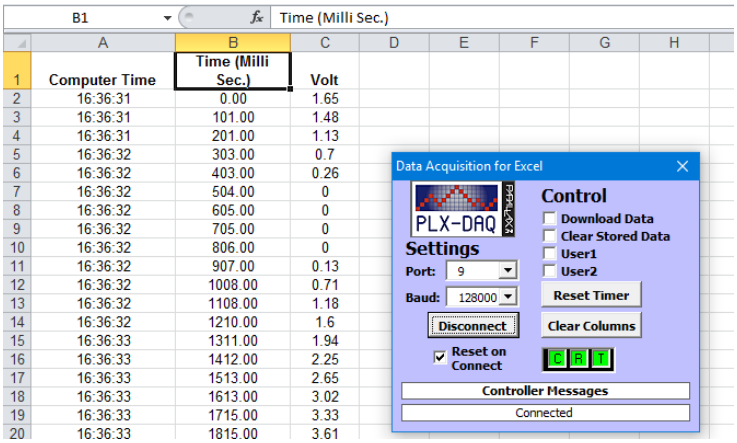

Figure 3. Implementation of PLX-DAQ

- Sensor Unit MCU_1

Function	Sensor	Type	Configuration	Comments
Door Person detect (front)	Ultrasonic distance sensor	Input Capture	Sampling 0.1 sec Check object presence within 30cm Generates PERSON_OUTSIDE flag	Need to check for outlier measurements (at least 7/10 Postive) VISITOR_LOG under SECUR_MODE
Door Person detect (inside)	PIR motion sensor	Digital	Edge trigger Generates PERSON_INSIDE flag	
Glass breaking detect	Sound sensor	Analog	Choose sampling Set a threshold value for breaking Generates WIN_BREAK flag	
Window open detect	IR reflective sensor	Analog	Choose sampling Set a threshold value for open WIN_OPEN flag	Need to check for outlier measurements (at least 7/10 Postive)
Daylight intensity	Light Intensity sensor	Digital	Edge trigger Generates CURTAIN_OPEN Flag	Check for false data. Need to maintain Bright or Dark condition for 2 secs
MODE switch	Button B1	Digital	Edge trigger Toggles SECUR_MODE to NORM_MODE	

Table 5. MCU1 configuration

- Actuator/Display Unit: MCU_2

Function	Actuator	Conditions	Action	Comment
Curtain	Stepper motor	CURTAIN_OPEN=0 CURTAIN_OPEN=1	10 rev CW. Generates CUR_OPENED=1 flag 10 rev CCW. Generates CUR_OPENED=0 flag	Should give complete_flag to MCU1 Open/close only once. Cannot open when it is already opened
Door Light	LED	DLIGHT_ON=1	Turns on light	
MODE Display	7-segment	NORM MODE SECUR MODE	Display ‘1’ Display ‘5’	
SIREN Countdown	7-segment	SIREN _TRG=1	Count 5 to 0 with 1 sec rate. Go to SIREN_ON	
SIREN_ON	7-segment	SIREN_ON=1 If ‘Stop’ button is not pressed in this mode	7-segment blinking with number ‘0’ at rate of 1 sec	
SIREN_STOP	B1 of MCU2 (Input)	Under SECUR_MODE	Edge trigger Turns off SIREN Dısplay ‘5’	Reset to SECUR MODE

Table 6. MCU2 configuration

2.2 MCU Configuration

You are free to select appropriate configurations for the design problem. Create a configuration table to list all the necessary setup.

MUST condition: Use at least one timer interrupt, at least one polling process in main ().

Functions	Register	PORT_PIN	Configuration
System Clock	RCC		PLL 84MHz
delay_ms	SysTick		
	Digital Out		
Button	Digital In	PA13	Pull-Up, Falling edge
PIR Motion Sensor	Digital In	PB8	Pull-Up, Falling edge
Light Intensity Sensor	Digital In	PB4	Pull-Up, Falling edge
Ultrasonic Distance Sensor	TIMER2	PA6 PB10	Counter step time: 10us TIM2_CH3 as IC3, rising edge detect TIM2_CH3 as IC4, falling edge detect
IR Sensor	ADC	PB0	TRGO
Sound Sensor	ADC	PB1	TRGO
RS-232 USB cable(ST-Link)	USRAT2	TXD: PA2 RXD: PA3	No parity, 8-bit Data, 1-bit stop bit 9600 buad-rate(9600)
Bluetooth	USRAT1	TXD: PA9 RXD: PA10	No parity, 8-bit Data, 1-bit stop bit 9600 buad-rate(9600)
Zigbee	USRAT6	TXD: PA11 RXD: PA12	No parity, 8-bit Data, 1-bit stop bit 9600 buad-rate(9600)

Table 7. MCU configuration

2.3 MCU wiring connection

MUST condition: Show the wiring of each sensor/actuator component to MCUs.

DO NOT draw by hand.

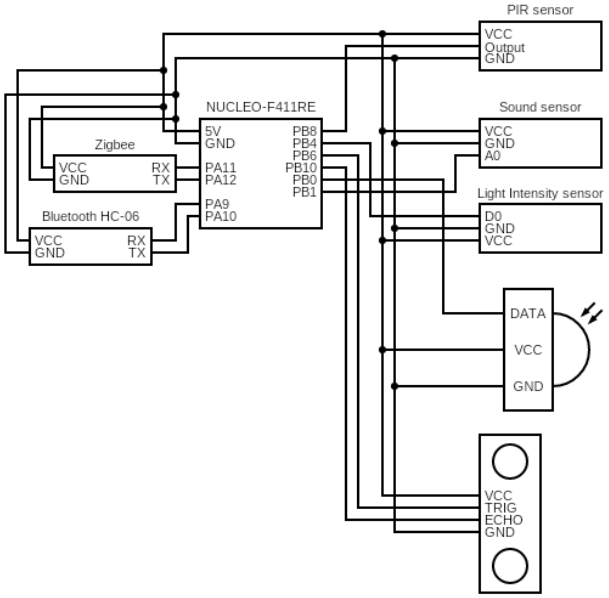

Figure 4. MCU1 external circuit

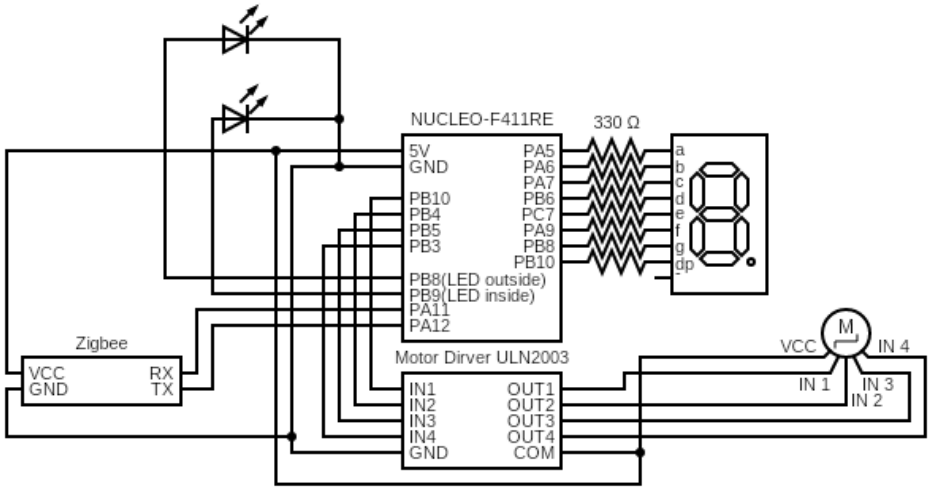

Figure 5. MCU2 external circuit

III. Algorithm

3.1 Algorithm Overview

For the algorithm overview, you need to explain concisely how your system works. Recommended explain with tables or state diagram. Also, you should use explain by

- **Listing all necessary states (states, input, output etc.) to implement this design problem.**

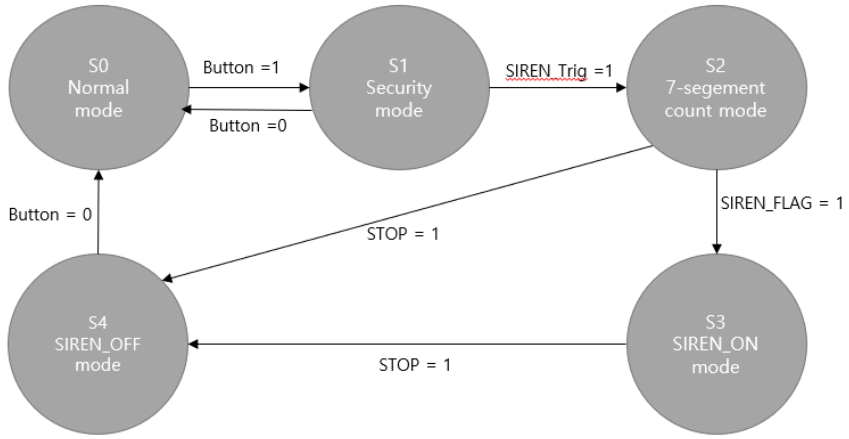

Figure 6. state diagram overview

The following are largely divided into NORMAL MODE, SECURTIY MODE, SIREN_ON, SIREN_OFF, and COUNT MODE. This state diagram shows a schematic diagram of the overall smart home. According to each state, output according to input is expressed as a Moore circuit diagram. The meaning of SIREN_FLAG=1 is a flag value indicating that the countdown of 7-segment is over. In executing this lab, we have renewed the definition of the SIREN_TRIG situation. Originally, pressing the STOP button requires returning to the SECURITY MODE, but then SIREN_TRIG continues to occur and falls into an infinite loop. Therefore, when the STOP button was pressed, the operation of initializing the SECURITY MODE and SRIEN MODE was performed. And in the case of Sound sensor, SIREN_TRIG does not occur in normal mode, so the SIREN_TRIG signal was sent only in the case of SECURTIY MODE. The conditions for all SIREN_TRIG FLAGs were to send signals only in the case of SECURITY MODE.

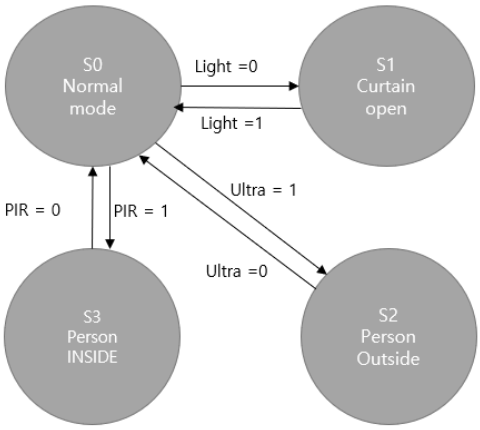

Figure 7. Normal mode state diagram

Figure x. shows the state diagram according to input and output in normal mode. The state of the actuator changed in normal mode according to the input flag was considered. The actuator output executed for each sensor data is independent.

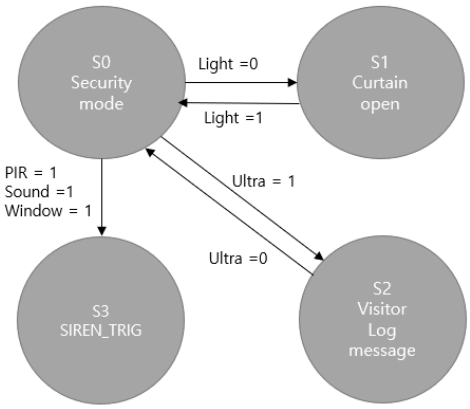

Figure 8. Secure mode state diagram

Figure x. shows the state diagram according to input and output in the case of security mode. In the case of security mode, unlike normal mode, the part of SIREN_TRIG should be specifically considered. Each state output defined an output state different from the normal mode, such as the visual log message or siren trigger.

- **Listing all necessary conditional FLAGS (WIN_BREAK etc.) for programming.**

  The flags for input and output are summarized as follows.

Input flag	Output Flag
Button	MODE_Toggle
Ultra	PERSON_OUTSIDE
PIR	PERSON_INSIDE
Window	WIN_OPEN
Curtain	CURATIN_OPEN
Sound	WIN_BREAKING

Table 8. Used input and output flag

The table above shows the output flag according to the input flag. Information on the input flag collected through each sensor was received in the form of an output flag on the actuator and executed.

- **Explaining additional conditions/configurations that were not explained in Introduction.**

  The additional execution in this lab defines the form of sending data from MCU1 to MCU2. When an interrupt occurs through a value received from each sensor, a signal must be sent independently immediately. First, the form of the output flag that will change according to the input flag was made into an array form. Communication was accurately accepted through the values of the first buffer and the third buffer, and the value of the output flag was determined to be 0 or 1. The first buffer value is a signal that allows us to distinguish which flag came in. Therefore, each sensor was required to send a signal independently, and the actuator classified and recognized the signal according to the buffer value. The figure below is shown in the form of an array, and all initial values are set to 0. As the input flag occurred, the second buffer value was converted to 1 to send a signal.

```
unsigned char BUTTON[3] = {'B','0','E'};
unsigned char Ultra[3] = {'U','0','E'};
unsigned char PIR[3] = {'P','0','E'};
unsigned char WINDOW[3] = {'I','0','E'};
unsigned char CURTAIN[3] = {'C','0','E'};
unsigned char SOUND[3] = {'S','0','E'};
unsigned char SIRENMODE[3] = {'X','0','E'};
```

Figure 9. Array of sensor

3.2 Flow Chart

Draw a flow chart or state diagram to show the flow of your system

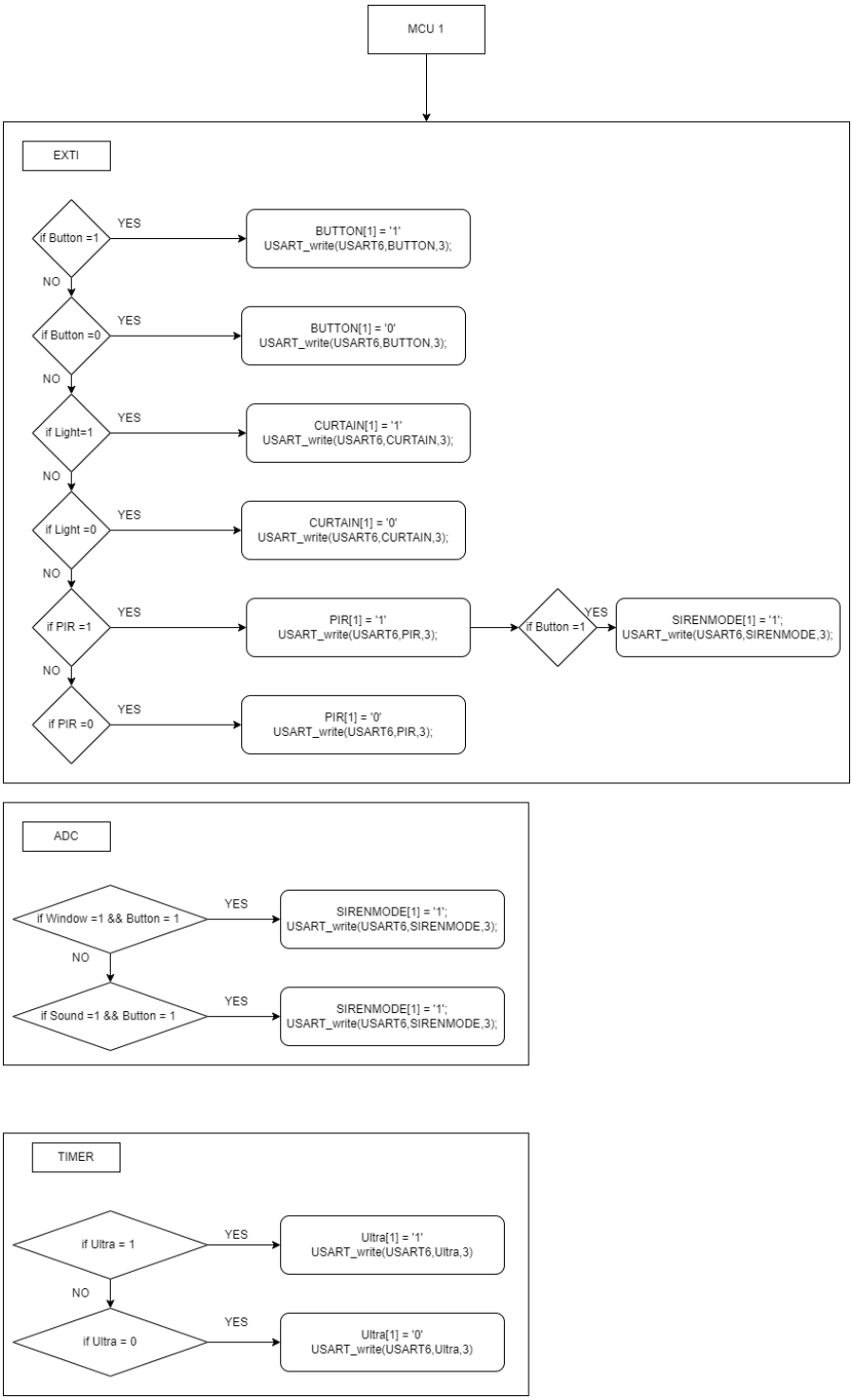

Figure 10. MCU 1 Flow Chart

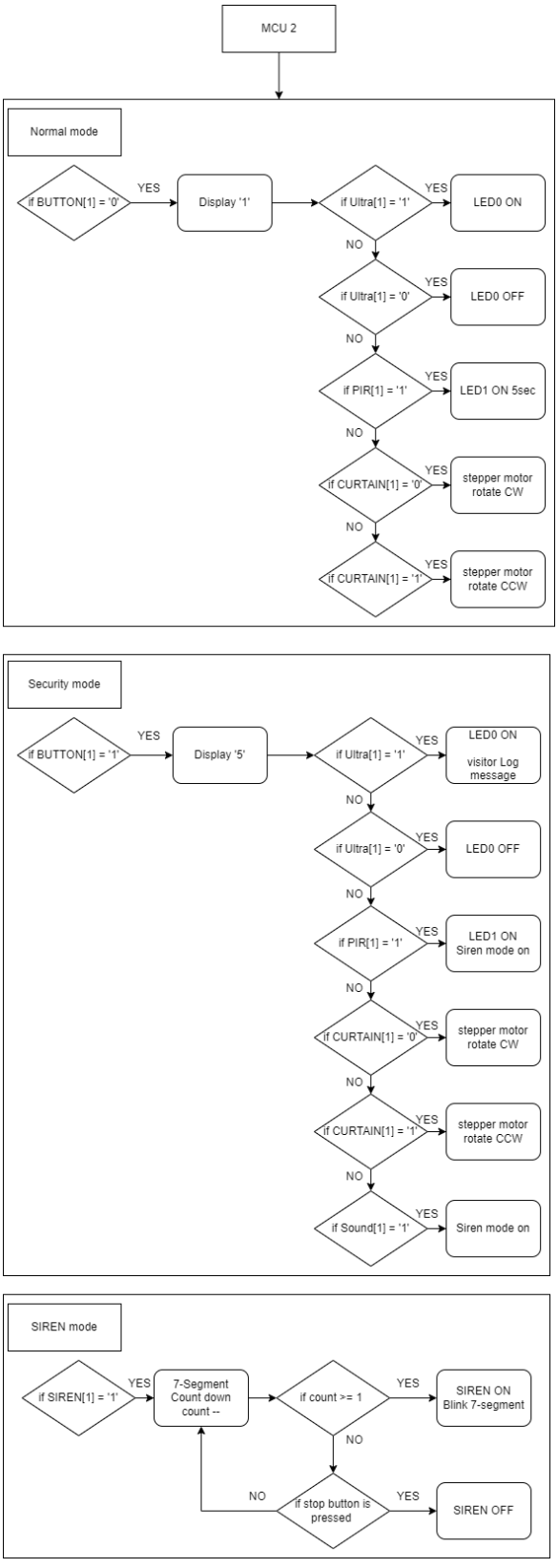

Figure 11. MCU 2 Flow Chart

IV. Demonstration

- Video demo Link

  Click here watch video.

V. Conclusion

Through this rap, I dealt with the overall things of embed HAL that I have learned so far. From the basic operating principle, we could learn about various functions. The most important thing was communication. Unlike the previous lab, there were many errors because information had to be exchanged through communication. At first, communication was coded by periodically sending it once at a time. However, this method was unnecessarily sending too many signals, resulting in communication problems. So, we changed the method to send signals independently only when an interrupt occurs. Because unnecessary signals were not repeatedly sent, smooth communication was possible. Through this embedded class, I was able to learn how to deal with various sensors in combination, starting with turning on and off one led. In addition, it was possible to cultivate the ability to communicate with the other party through communication.

- **Trouble shooting**

  Q: How could the communication problem be solved?

  A: When communicating for the first time, there were many cases where data was not properly transmitted, or signals were not sent because too much data was processed. To solve this problem, the signal was sent only once when the signal changed when the interrupt occurred. Then, it was possible to prevent continuous transmission of the same signal, reducing unnecessary signals and confirming that communication was going smoothly without interruption.

VI. Appendix

- Source code: main.c

- MCU 1

```
/**
********************************************************************************
* @author  SONG YEONG WON
* @Mod     2021-12-23 by YWSONG
* @brief   Embedded Controller:  USART communication
*
********************************************************************************
*/

#include "stm32f411xe.h"
#include "math.h"
#include "ecGPIO.h"
#include "ecRCC.h"
#include "ecTIM.h"
#include "ecPWM.h"
#include "ecUART.h"
#include "ecSysTick.h"
#include "ecADC.h"
#include "ecEXTI.h"
#include "string.h"

// USART2 : MCU to PC via usb
// USART1 : MCU to MCU2 via zigbee

// USART2 : MCU to PC via usb

uint8_t buf1[4];
uint8_t buf2[4];
uint8_t buf3[4];
uint8_t buf4[4];
uint8_t buf5[4];
uint8_t buf6[4];
uint8_t buf7[4];
uint8_t buf8[4];
uint8_t buf9[4];

uint8_t mcu2Data = 0;
uint8_t pcData = 0;
int indx =0;
int maxBuf=50;
uint8_t buffer[100]={0,};
int bReceive=0;
int ledOn = 0;
int endChar = 13;
```

```
//-------Ultrasonic variable------------//
static int overflow = 0;
float distance = 0;
static float timeSt = 0;
static float timeEnd= 0;
static float sec = 0;
static int PERSON_OUTSIDE =0;     // Door front Flag
static int Distance_validity =0;      // Door front Flag


//--------IR sensor parameter------------------//
static int flag =0 ;
static uint32_t IR1=0;
static uint32_t IR2=0;
static int seqCHn[2] = {8,9};
static int WIN_OPEN =0;              // Window open detect Flag
static int WINDOW_Validity =0;

//--------PIR motion sensor parameter------------//
static int PERSON_INSIDE =0;

//--------Light Intensity Sensor parameter-------//
static uint32_t Light_prev = 0;
static uint32_t Light_next = 0;

//--------Button mode parameter------------------//
static int button_pressed = 0;

//---------TIM5 DAQ parameter--------------------//
static int DAQ_FLAG= 0;
static int DAQ_cnt= 0;
static int DAQ_cnt2= 0;
static int Light_TRUE= 0;

static int Light_FlAG= 0;
static int Light_PREV= 0;
//--------Ultra send---------------//
static int ULT_FlAG= 0;
static int ULT_prev= 0;
static int ULT_next= 0;

//---------ADC WINDOW------------//
static int WIN_FlAG= 0;
static int WIN_prev= 0;
static int ADC_SEND= 0;
```

```
//-----------VISITOR-----------//
static int sound =0;
static int visitor =0;
static int siren = 0;
static int stop = 0;


//MODE, Ultrasonic, PIR, Window, Curtain, Sound, end of char

unsigned char BUTTON[3] = {'B','0','E'};
unsigned char Ultra[3] = {'U','0','E'};
unsigned char PIR[3] = {'P','0','E'};
unsigned char WINDOW[3] = {'I','0','E'};
unsigned char CURTAIN[3] = {'C','0','E'};
unsigned char SOUND[3] = {'S','0','E'};
unsigned char SIRENMODE[3] = {'X','0','E'};


void setup(void);
void TIM2_IRQHandler(void);     //Front door -> Ultrasonic distance sensor
void ADC_IRQHandler(void);      //Window open detect and sound sensor
void EXTI9_5_IRQHandler(void);  //Inside door -> PIR motion sensor
void EXTI4_IRQHandler(void);    //Light Intensity Sensor
void USART6_IRQHandler(void);   //Zigbee comunication
void USART2_IRQHandler(void);
void USART1_IRQHandler(void);
void EXTI15_10_IRQHandler(void); //Button mode
void TIM5_IRQHandler(void);     //DAQ log
void TIM4_IRQHandler(void);

int main(void) {
  // Initialiization --------------------------------------------------------
  setup();
  ADC_start();
  printf("Hello Nucleo\r\n");

  //USART1 excel_DAQ initialize
  USART_write(USART1,(unsigned char*) "CLEARSHEET\r\n",12);
  USART_write(USART1,(unsigned char*) "LABEL,Date,Time,Timer,MODE,Dist,PIR,WINDOW,Curtain,SOUND,siren,Visitor,stop\r\n",77);
  // Inifinite Loop ----------------------------------------------------------
  while (1){


    if(DAQ_FLAG==1){
    sprintf(buf1, "%d", BUTTON[1]-48);
    sprintf(buf2, "%d", Ultra[1]-48);
    sprintf(buf3, "%d", PIR[1] - 48);
    sprintf(buf4, "%d", WINDOW[1]- 48);
    sprintf(buf5, "%d", CURTAIN[1]-48);
    sprintf(buf6, "%d", sound);
    sprintf(buf7, "%d", siren);
    sprintf(buf8, "%d", visitor);
    sprintf(buf9, "%d", stop);

    //USART1 Trasnmit sensor value to server
    USART_write(USART1,(unsigned char*) "DATA,DATE,TIME,TIMER,",21);  // transmit char to USART6
    USART_write(USART1,&buf1,4);
    USART_write(USART1,(unsigned char*) ",",1); // transmit char to USART6
    USART_write(USART1,&buf2,4);
    USART_write(USART1,(unsigned char*) ",",1); // transmit char to USART6
    USART_write(USART1,&buf3,4);
    USART_write(USART1,(unsigned char*) ",",1); // transmit char to USART6
    USART_write(USART1,&buf4,4);
    USART_write(USART1,(unsigned char*) ",",1); // transmit char to USART6
    USART_write(USART1,&buf5,4);
    USART_write(USART1,(unsigned char*) ",",1); // transmit char to USART6
    USART_write(USART1,&buf6,4);
    USART_write(USART1,(unsigned char*) ",",1); // transmit char to USART6
    USART_write(USART1,&buf7,4);
    USART_write(USART1,(unsigned char*) ",",1); // transmit char to USART6
    USART_write(USART1,&buf8,4);
    USART_write(USART1,(unsigned char*) ",",1); // transmit char to USART6
    USART_write(USART1,&buf9,4);
    USART_write(USART1,(unsigned char*) ",AUTOSCROLL_20\r\n",16); // transmit char to USART6
    stop=0;

      DAQ_FLAG=0;
    }
  }
}
```

```
// Initialiization
void setup(void)
{
  RCC_PLL_init();
  SysTick_init(1);

  //-----------------Button Mode configuration-------------------//
  EXTI_init(GPIOC, BUTTON_PIN, FALL_EXTI, 1);
  GPIO_init(GPIOC, BUTTON_PIN, INPUT);
  GPIO_pupdr(GPIOC, BUTTON_PIN, PD);

  //-----------------PIR motion sensor(inside door)--------------//
  GPIO_init(GPIOB, 8,INPUT);
  EXTI_init(GPIOB,8,BOTH_EXTI,2);
  GPIO_pupdr(GPIOB, 8, PU);

  //-----------------Light intensity sensor(Curtain)--------------//
  GPIO_init(GPIOB, 4, INPUT);
  EXTI_init(GPIOB,4,BOTH_EXTI,3);
  GPIO_pupdr(GPIOB, 4, PU);


  // USART congfiguration
  USART_init(USART2, 9600);
  USART_begin(USART1, GPIOA,9,GPIOA,10, 9600);  // PA9 - RXD , PA10 - TXD
  USART_begin(USART6, GPIOA,11,GPIOA,12, 9600);   // PA11 - RXD , PA12 - TXD


  //------------Ultrasonic distance sensor(Front door)--------//

  // PWM configuration ---------------------------------------------------------------
  PWM_t trig;                     // PWM1 for trig
  PWM_init(&trig,GPIOB,6);        // PWM init as PA_1: Ultrasonic trig pulse TIM2_CH2
  PWM_period_us(&trig,100000);      // PWM of 50ms period. Use period_us()
  PWM_pulsewidth_us(&trig,10);    // PWM pulse width of 10us

  // Input Capture configuration -----------------------------------------------------
  IC_t echo;                      // Input Capture for echo
  ICAP_init(&echo,GPIOB,10);      // ICAP init as PB10 as input caputre
  ICAP_counter_us(&echo, 10);     // ICAP counter step time as 10us
  ICAP_setup(&echo, 3, RISE_TIM);     // TIM2_CH3 as IC3 , rising edge detect
  ICAP_setup(&echo, 4, FALL_TIM);     // TIM2_CH3 as IC4 , falling edge detect
```

```
  // Enable TIMx interrupt      ------------------------------------------------------
  TIM_INT_enable(TIM2);           // TIM2 Interrupt Enable

  //------------Sound sensor and IR reflective sensor--------//
  ADC_init(GPIOB,0,TRGO);   //window open detect
  ADC_init(GPIOB,1,TRGO);   //Glass breaking detect
  ADC_sequence(2,seqCHn);

  //------------TIM5 configuration---------------------------//
  TIM_INT_init(TIM5,100);
}


//Front Door
void TIM2_IRQHandler(void){
  if(is_UIF(TIM2)){                     // Update interrupt
    overflow++;                         // overflow count
    clear_UIF(TIM2);                    // clear update interrupt flag
  }
  if(is_CCIF(TIM2,3)){                  // TIM4_Ch1 (IC1) Capture Flag. Rising Edge Detect
      timeSt = TIM2->CCR3;                // Capture TimeStart from CC3
    clear_CCIF(TIM2,3);                 // clear capture/compare interrupt flag
  }
  else if(is_CCIF(TIM2,4)){             // TIM4_Ch1 (IC2) Capture Flag. Falling Edge Detect
      timeEnd = TIM2->CCR4;
      sec =((timeEnd-timeSt)+(overflow*(0xFFFF+1)))*10;
      distance = (float) sec*340/(2*10000); // Ultrasonic speed[m/s] * echo pulse duration[us]
      PERSON_OUTSIDE++;

    ULT_prev = ULT_FlAG;
    if(distance<30)Distance_validity++;

    if(Distance_validity>=5 && PERSON_OUTSIDE >=10){
      ULT_FlAG = 1;
      Ultra[1] = '1';
      PERSON_OUTSIDE=0;
      Distance_validity=0;
      if(ULT_prev != ULT_FlAG){
        USART_write(USART6,Ultra,3);
      }
      if(BUTTON[1]=='1'){
        visitor=1;
      }if(BUTTON[1]=='0'){
        visitor=0;
      }
```

```
    }

    else if(Distance_validity<5 && PERSON_OUTSIDE >=10){
      ULT_FlAG = 0;
      Ultra[1] = '0';
      PERSON_OUTSIDE=0;
      Distance_validity=0;
      ULT_next=Ultra[1];
      if(ULT_prev != ULT_FlAG){
        USART_write(USART6,Ultra,3);
      }
      if(BUTTON[1]=='1'){
        visitor=0;
      }
    }
      overflow=0;
      clear_CCIF(TIM2,4);                // clear capture/compare interrupt flag
  }
}


//Window Open detect
void ADC_IRQHandler(void){
  if(is_ADC_OVR()){
    clear_ADC_OVR();
  }
  if(is_ADC_EOC()){       //after finishing sequence
    WIN_OPEN++;
    ADC_SEND++;

    WIN_prev = WIN_FlAG;
    if(flag==0){
      IR1 = ADC1->DR;
    if(IR1<1000)WINDOW_Validity++;

    if(WINDOW_Validity>=7 && WIN_OPEN >=20){
      WINDOW[1] = '1';
      WIN_OPEN=0;
      WINDOW_Validity=0;
      WIN_FlAG=1;

      if(WIN_prev != WIN_FlAG){
        if(BUTTON[1]=='1'){
          SIRENMODE[1]='1';
          siren=1;
```

```
        USART_write(USART6,SIRENMODE,3);
        }if(BUTTON[1]=='0'){
          siren=0;
        }
      }
    }
    else if(WINDOW_Validity<7 && WIN_OPEN >=20){
      WINDOW[1] = '0';
      WIN_OPEN=0;
      WINDOW_Validity=0;
      WIN_FlAG=0;
      }
    }

  if(flag==1){
    if(BUTTON[1]=='0'){
          siren=0;
        }
      IR2 = ADC1->DR;
      if(IR2>=1000){
        sound =1;
        if(BUTTON[1]=='1'){
          SIRENMODE[1]='1';
          siren=1;
          USART_write(USART6, SIRENMODE,3);
        }
      }else sound =0;
  }
  flag =!flag;
  }
}

//Door person detect (inside)
void EXTI9_5_IRQHandler(void){
   if(is_pending_EXTI(8)){
     if(GPIO_read(GPIOB,8) == 256){
       PIR[1] = '1';
       if(BUTTON[1]=='1'){
         SIRENMODE[1] = '1';
         siren=1;
         USART_write(USART6,SIRENMODE,3);
       }if(BUTTON[1]=='0'){
          siren=0;
        }
```

```
      }if(GPIO_read(GPIOB,8) == 0){
        PIR[1] = '0';
      }
      USART_write(USART6,PIR,3);
      clear_pending_EXTI(8);
    }
}

//Daylight intensity
void EXTI4_IRQHandler(void){
    if(is_pending_EXTI(4)){
      if(GPIO_read(GPIOB,4) == 16){
        CURTAIN[1] = '1';
      }if(GPIO_read(GPIOB,4) == 0){
        CURTAIN[1] = '0';
      }
      clear_pending_EXTI(4);
    }
}


//Comunication Zigbee
void USART6_IRQHandler(){    //USART1 INT
  if(is_USART_RXNE(USART6)){
     mcu2Data = USART_getc(USART6);
     buffer[indx] = mcu2Data;
     indx++;

     if(mcu2Data=='E'){
       indx = 0;
       switch(buffer[0]){
         case 'T': stop=1; siren=0; break;
       default: ;
     }
       printf("%c %c\r\n",buffer[0],buffer[1]);
  }
     else{
       if(indx>maxBuf){
         indx =0;
         memset(buffer, 0, sizeof(char) * maxBuf);
         printf("ERROR : Too long string\r\n");
       }
     }
  }
}
```

```
//Comunication mucl
void USART2_IRQHandler(){   //USART2 INT
  if(is_USART_RXNE(USART2)){
    pcData = USART_getc(USART2);
    USART_write(USART6,&pcData,1);  // transmit char to USART1
    printf("%c",pcData);            // echo to sender(pc)

    if(pcData==endChar){
      printf("\r\n");               // to change line on PC display
    }
  }
}

//Comunication DAQ
void USART1_IRQHandler(){   //USART1 INT
  if(is_USART_RXNE(USART1)){
    mcu2Data = USART_getc(USART1);
    if(mcu2Data==endChar) {
      bReceive=1;
      indx = 0;
    }
    else{
      if(indx>maxBuf){
        indx =0;
        memset(buffer, 0, sizeof(char) * maxBuf);
        printf("ERROR : Too long string\r\n");
      }
      buffer[indx] = mcu2Data;
      indx++;
    }
  }
}


void EXTI15_10_IRQHandler(void){
   if(is_pending_EXTI(BUTTON_PIN)){
     if(button_pressed==0){
       BUTTON[1] = '1';
      }if(button_pressed==1){
       BUTTON[1] = '0';
     }
      button_pressed=!button_pressed;
     USART_write(USART6,BUTTON,3);
     clear_pending_EXTI(BUTTON_PIN);
   }
}
```

```
void TIM5_IRQHandler(void){
  if(is_UIF(TIM5)){                     // Update interrupt
    DAQ_cnt++;
    DAQ_cnt2++;
    if(DAQ_cnt2 == 1) Light_prev=CURTAIN[1];
    else if(DAQ_cnt2 == 2){
      Light_next =CURTAIN[1];
      DAQ_cnt2 = 0;
    }
    Light_PREV = Light_FlAG;
    if(Light_prev == Light_next){
      Light_TRUE++;
    }
    if(Light_TRUE==20){
      Light_TRUE=0;
      if(Light_prev==48){
        Light_FlAG=0;
        if(Light_PREV != Light_FlAG){
        USART_write(USART6,CURTAIN,3);
        }
      }
      if(Light_prev==49){
        Light_FlAG=1;
        if(Light_PREV != Light_FlAG){
        USART_write(USART6,CURTAIN,3);
        }
      }
    }
    if(DAQ_cnt==20){
      DAQ_FLAG=1;
      DAQ_cnt=0;
    }
    clear_UIF(TIM5);                    // clear update interrupt flag
  }
}
```

Figure 12. Main code MCU1

- MCU2

```
/**
*****************************************************************************************
* @author  SONG YEONG WON
* @Mod     2021-12-23 by YWSONG
* @brief   Embedded Controller:  USART communication
*
*****************************************************************************************
*/

#include "stm32f411xe.h"
#include "math.h"
#include "ecGPIO.h"
#include "ecRCC.h"
#include "ecTIM.h"
#include "ecPWM.h"
#include "ecUART.h"
#include "ecSysTick.h"
#include "ecADC.h"
#include "ecEXTI.h"
#include "string.h"
#include "ecStepper.h"


uint8_t mcu2Data = 0;
uint8_t pcData = 0;
int indx =0;
int maxBuf=50;
uint8_t buffer[100]={0,};
int bReceive=0;
int ledOn = 0;
//int endChar = 13;
int endChar = 69;


//SIREN MODE count number
static uint32_t SRFG_count = 6;
static uint32_t SRFG_toggle = 0;

static uint32_t mode =0;
static uint32_t siren =1;
static uint32_t siren_MODE =1;
```

```
//------------TIM5 Stepper Motor------------//
static int stepper_Toggle =0;
static int Stepper_Flag = 0;
static int Stepper_day = 1;
static int Stepper_night = 1;
static int Stepper_MODE = 1;
static int Stepper_print = 0;

//----------SIREN TRIGER----------------//
static int SIREN_sevenprint = 0;
static int SIREN_PRINTMODE = 1;
static int Curtain_siren =0;
static int Stepper_siren =1;

//-----------INSIDE 5sec maintain-------------//
static int INSIDE_cnt = 0;
static int INSIDE_FLAG = 0;
static int INSIDE_FLAG_SECURE =0;
//-----------SECURE MODE--------------//

unsigned char BUTTON[3] = {'B','0','E'};
unsigned char Ultra[3] = {'U','0','E'};
unsigned char PIR[3] = {'P','0','E'};
unsigned char WINDOW[3] = {'I','0','E'};
unsigned char CURTAIN[3] = {'C','0','E'};
unsigned char SOUND[3] = {'S','0','E'};


unsigned char STOP_buffer[3] = {'T','1','E'};

static  uint8_t Button_v = 48;
static  uint8_t Ultrasonic_v = 48;
static  uint8_t PIR_v = 48;
static  uint8_t Sound_v = 48;
static  uint8_t IR_v = 0;
static  uint8_t Light_v = 48;
static  uint8_t Siren_v = 48;
void setup(void);
void TIM2_IRQHandler(void);     //Front door -> Ultrasonic distance sensor
void USART6_IRQHandler(void);
void EXTI15_10_IRQHandler(void); //Button mode
void TIM5_IRQHandler(void);
void TIM2_IRQHandler(void);
void SMART_HOME(void);


int main(void) {
  // Initialiization --------------------------------------------------------
  setup();
  printf("Hello Nucleo\r\n");
  // Inifinite Loop ---------------------------------------------------------
  while (1){

      //end of Siren
      if(SIREN_sevenprint==1){
        Stepper_step(2048*2, 1, HALF);
        Stepper_siren=0;
        SIREN_sevenprint=0;
      }

      //NORMAL MODE
      if(Stepper_print==1){
        Stepper_step(2048*2, 1, HALF);
        //Stepper_day=0;
        stepper_Toggle=1;
        Stepper_print=0;
        //Stepper_Flag =0;
      } if(Stepper_print==2){
        Stepper_step(2048*2, 0, HALF);
        //Stepper_night =0;
        stepper_Toggle=0;
        Stepper_print=0;
    }
  }
}
```

```
// Initialiization
void setup(void)
{
  RCC_PLL_init();
  SysTick_init(1);

  //----------SIREN MODE 7-segement configuration-----------//
  sevensegment_init();

  //OUTSIDE LED
  GPIO_init(GPIOB, 8, OUTPUT);
  GPIO_ospeed(GPIOB, 8, MEDIUM_SPEED);
  GPIO_otype(GPIOB, 8, OUTPUT_PP);
  GPIO_pupdr(GPIOB, 8, NOPUPD);

  //INSIDE LED
  GPIO_init(GPIOB, 9, OUTPUT);
  GPIO_ospeed(GPIOB, 9, MEDIUM_SPEED);
  GPIO_otype(GPIOB, 9, OUTPUT_PP);
  GPIO_pupdr(GPIOB, 9, NOPUPD);

    //----------------Button Mode configuration-------------------//
  EXTI_init(GPIOC, BUTTON_PIN, FALL_EXTI, 1);
  GPIO_init(GPIOC, BUTTON_PIN, INPUT);
  GPIO_pupdr(GPIOC, BUTTON_PIN, PD);

  //------------TIM5 configuration---------------------------//
  //TIM_INT_usec_init(TIM5,100);

  //-------------Stepper Motor-------------------------------//
  Stepper_init(GPIOB,10,GPIOB,4,GPIOB,5,GPIOB,3); // Stepper GPIO pin initialization
  Stepper_setSpeed(10,HALF);                          //  set stepper motor speed


  //------------TIM2 configuration---------------------------//
  TIM_INT_init(TIM2,1000);

  // USART congfiguration
  USART_init(USART2, 9600);
  USART_begin(USART6, GPIOA,11,GPIOA,12, 9600);   // PA11 - RXD , PA12 - TXD
}
```

```
void USART6_IRQHandler(){    //USART1 INT
  if(is_USART_RXNE(USART6)){
    mcu2Data = USART_getc(USART6);
    //printf("%c",mcu2Data);
    buffer[indx] = mcu2Data;
    indx++;

    if(mcu2Data=='E'){
      indx = 0;
      switch(buffer[0]){
        case 'B': Button_v = buffer[1]; break;
        case 'U': Ultrasonic_v = buffer[1]; break;
        case 'P': PIR_v = buffer[1]; break;
        case 'I': IR_v = buffer[1]; break;
        case 'C': Light_v = buffer[1]; break;
        case 'S': Sound_v = buffer[1]; break;
        case 'X': Siren_v = buffer[1]; break;
      default: ;
    }
      printf("%c %c\r\n",buffer[0],buffer[1]);
  }
    else{
      if(indx>maxBuf){
        indx =0;
        memset(buffer, 0, sizeof(char) * maxBuf);
        printf("ERROR : Too long string\r\n");
      }
    }
  }
}
```

```
//STOP BUTTON
void EXTI15_10_IRQHandler(void){
   if(is_pending_EXTI(BUTTON_PIN)){
     siren=0;
     sevensegment_decoder(5);
     Siren_v =0;
     INSIDE_FLAG_SECURE=0;
     GPIO_write(GPIOB,9,LOW);
     stepper_Toggle=1;
     USART_write(USART6,STOP_buffer,3);
     clear_pending_EXTI(BUTTON_PIN);
   }
}


void TIM2_IRQHandler(void){
  if(is_UIF(TIM2)){                     // Update interrupt
    if(INSIDE_FLAG==1){
        GPIO_write(GPIOB,9,HIGH);
        INSIDE_cnt++;
      if(INSIDE_cnt==3){
        GPIO_write(GPIOB,9,LOW);
        INSIDE_cnt=0;
        INSIDE_FLAG=0;
      }
    }
    SMART_HOME();
    clear_UIF(TIM2);                    // clear update interrupt flag
  }
}

void SMART_HOME(void){
  //-------------------------------------------------------------------------//
  //-----------------------------NORMAL MODE---------------------------------//
  //-------------------------------------------------------------------------//

  if(Button_v == 48){
    //Normal mode steup
    sevensegment_decoder(1);
    SRFG_count=6;
    siren=1;
    Stepper_siren=1;

    //OUTSIDE PEOPEL DETECT
    if(Ultrasonic_v == 49){
      GPIO_write(GPIOB,8,HIGH);
    }if(Ultrasonic_v == 48){
      GPIO_write(GPIOB,8,LOW);   //5 sec on and off
    }

    //INSIDE PEOPEL DETECT
    if(PIR_v == 49){
      INSIDE_FLAG=1;
    }

    //Curtain Daytime Lighttime
    if(Light_v == 48 &&  stepper_Toggle==0){
      Stepper_Flag = Stepper_day;
      if(Stepper_Flag == Stepper_MODE){
        Stepper_print=1;
      }
    }if(Light_v == 49 && stepper_Toggle==1){
      Stepper_Flag = Stepper_night;
      if(Stepper_Flag == Stepper_MODE){
        Stepper_print=2;
      }
    }

  }
```

```
  //------------------------------------------------------------------------------//
  //------------------------------SECURE MODE-------------------------------------//
  //------------------------------------------------------------------------------//
  else if(Button_v == 49){
    sevensegment_decoder(5);
    Stepper_day=1;
    Stepper_night=1;

    //OUTSIDE PEOPEL DETECT
    if(Ultrasonic_v == 49){
      GPIO_write(GPIOB,8,HIGH);
      //VSITOR_LOG MESSAGE TO the server
    }else GPIO_write(GPIOB,8,LOW);

    //INSIDE PEOPEL DETECT
    if(PIR_v == 49){
      INSIDE_FLAG=1;
    }

    if(INSIDE_FLAG==1){
      GPIO_write(GPIOB,9,HIGH);
    }
    if(INSIDE_FLAG==0){
      GPIO_write(GPIOB,9,LOW);
    }
      if(Siren_v ==49){
        mode = siren ;

    if(mode == siren_MODE){
      if(SRFG_count>=1){
          SRFG_count--;
        sevensegment_decoder(SRFG_count);
        }
        if(SRFG_count<1){
            sevensegment_decoder(SRFG_toggle);
            Curtain_siren=Stepper_siren;
          if(Curtain_siren==SIREN_PRINTMODE){
            SIREN_sevenprint=1;
          }
            if(SRFG_toggle==0){
              SRFG_toggle=10;
            }else SRFG_toggle=0;
        }
      }
    }
  }
}
```

Figure 13. Main code MCU2