LAB: Smart Home Security System

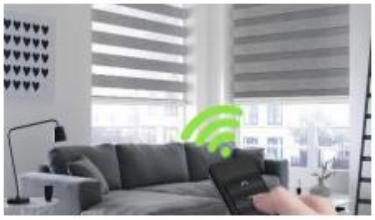
Date: 2021.12.23

Name(ID): Park Jeong Woo

Partner Name: Kim Ji Sung

I. Introduction

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



A. 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.

B. Window Security System

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

C. 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 generates 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.

D. Hard-Ware & Soft-Ware

Hardware

- MCU: NUCLEO -F411RE x 2
- Analog Sensor: IR reflective optical sensor(TCRT5000) x1
  Ultrasonic distance sensor(HC-SR04) x1
  Sound sensor (SZH-EK033) x1
- Digital Sensor: Light intensity sensor(MSE004LSM) x1
  PIR motion sense
- Actuator: Stepper Motor(28BYJ-48) , Motor driver(ULN2003)
- Display: LED, 7-segment display(S-5101ASR)
- Communication: Zigbee module (XB24CZ7WIT-004) x2,
  Zigbee Shield(DFR0015) x2, Bluetooth Module(HC-06)x1

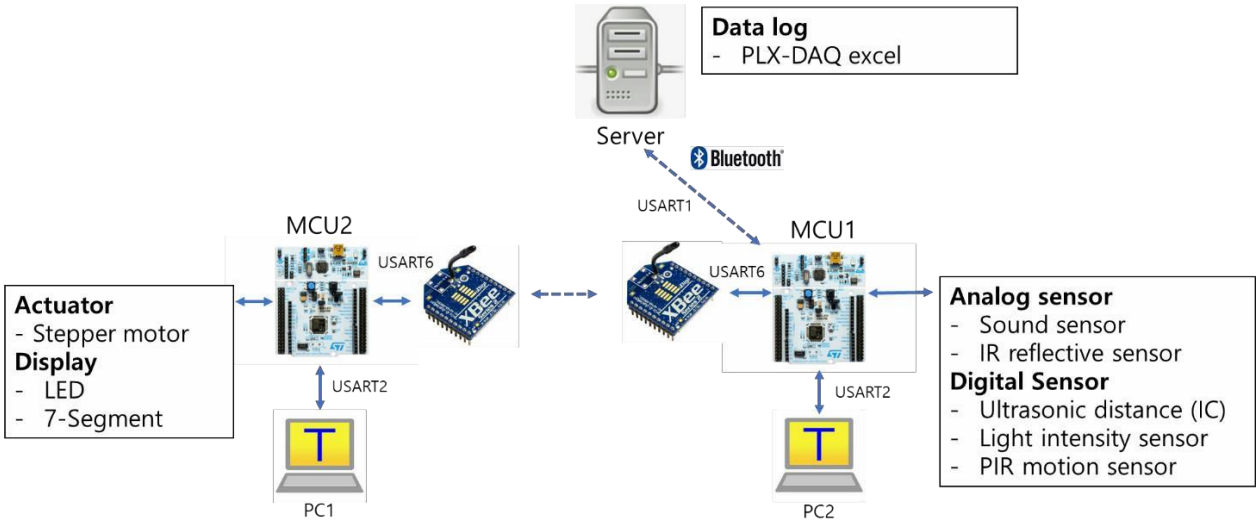
Software

- Keil uVision IDE
- CMSIS
- EC_HAL

II. Problem

A. Description

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

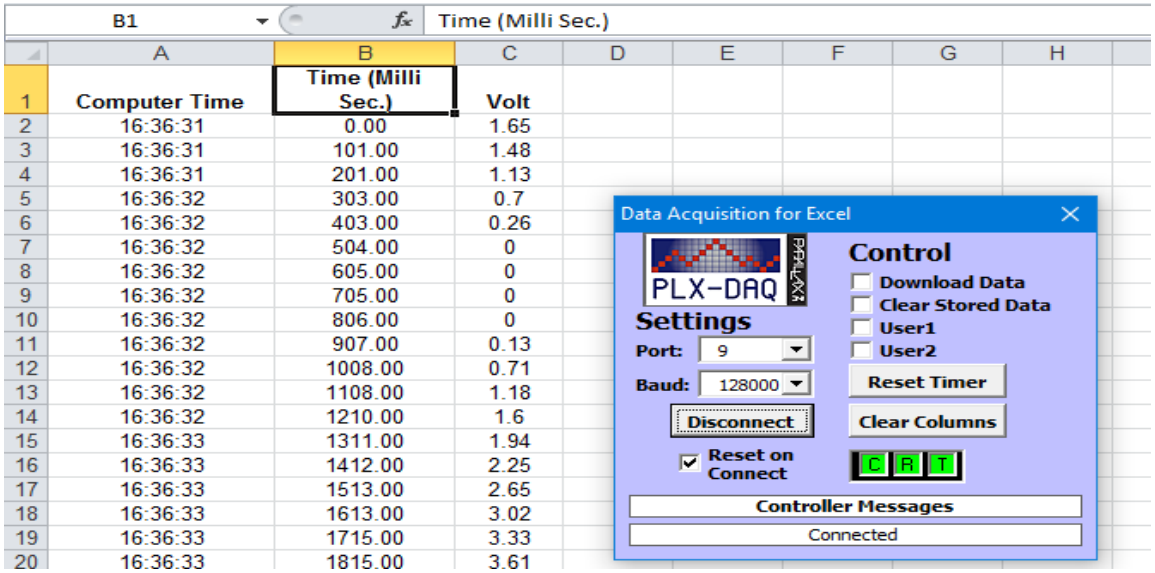


Embedded Controller

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.



	A	B	C
1	Computer Time	Time (Milli Sec.)	Volt
2	16:36:31	0.00	1.65
3	16:36:31	101.00	1.48
4	16:36:31	201.00	1.13
5	16:36:32	303.00	0.7
6	16:36:32	403.00	0.26
7	16:36:32	504.00	0
8	16:36:32	605.00	0
9	16:36:32	705.00	0
10	16:36:32	806.00	0
11	16:36:32	907.00	0.13
12	16:36:32	1008.00	0.71
13	16:36:32	1108.00	1.18
14	16:36:32	1210.00	1.6
15	16:36:33	1311.00	1.94
16	16:36:33	1412.00	2.25
17	16:36:33	1513.00	2.65
18	16:36:33	1613.00	3.02
19	16:36:33	1715.00	3.33
20	16:36:33	1815.00	3.61

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	

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 Display '5'	Reset to SECUR MODE

B. MCU Configuration

I was free to select appropriate configurations for the design problem.

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

Sensor Unit: MCU_1

Functions	Register	PORT_PIN	Configuration
System Clock	RCC		PLL 84MHz
delay_ms	SysTick		
EXTI_init	EXTI	PA0	BOTH_edge, Light intensity sensor
		PA1	RISE_edge, PIR sensor
		PC13	FALL_edge, Button
RS-232 USB cable(ST-LINK)	USART2		No Parity, 8-bit Data, 1-bit Stop bit 38400 baud-rate
Bluetooth	USART1	TXD: PA9 RXD: PA10	No Parity, 8-bit Data, 1-bit Stop bit 9600 baud-rate
Zigbee	USART6	TXD: PA11 RXD: PA12	No Parity, 8-bit Data, 1-bit Stop bit 9600 baud-rate
TIMER	TIMER2	PA5	Ultra sonic sensor, For PWM(Trig), 100ms, timer interrupt
	TIMER3	PC0	Sound sensor, ADC_TRGO, 100ms
		PC1	IR_sensor, ADC_TRGO, 100ms
	TIMER4	PB6	Ultra sonic sensor, For input capture(echo), 10us, timer interrupt

Actuator/Display Unit: MCU_2

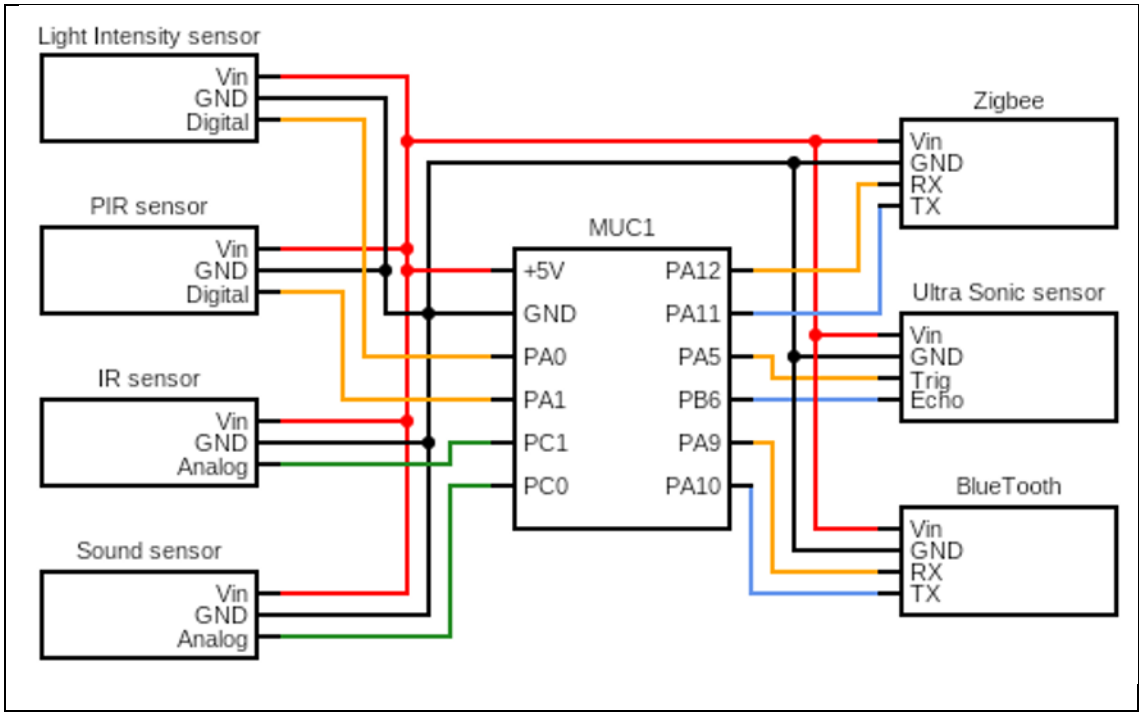
Functions	Register	PORT_PIN	Configuration
System Clock	RCC		PLL 84MHz
delay_ms	SysTick		1ms
EXTI_init	EXTI	PC13	FALL_edge, Button
Inside LED	Digital OUT	PB3	
Outside LED		PA10	
7-Segment		PA5, PA6, PA7, PB6 PC7, PA9, PA8, PB10	
RS-232 USB cable(ST-LINK)	USART2		No Parity, 8-bit Data, 1-bit Stop bit 38400 baud-rate
Zigbee	USART6	TXD: PA11 RXD: PA12	No Parity, 8-bit Data, 1-bit Stop bit 9600 baud-rate
TIMER	TIMER2		Timer Intrrupt, 1s, Inside LED control
	TIMER3		Timer Interrupt, 1s, 7-segment control
Stepper Motor	Digital Out	PA0, PA1, PA4, PB0	Stepper Motor OUTPUT

Embedded Controller

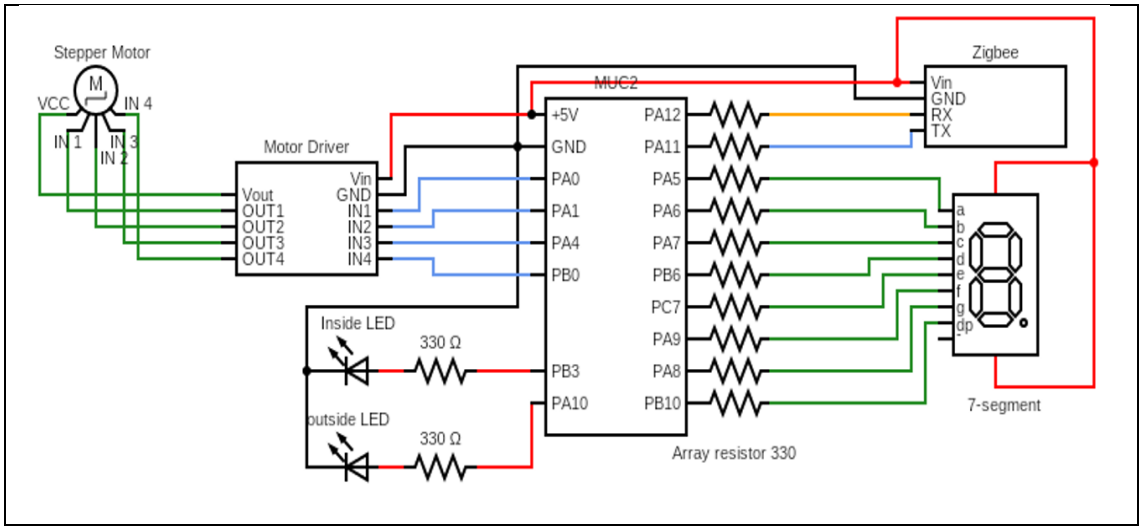
MCU Wiring Connection

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

Sensor Unit: MCU_1



Actuator/Display Unit: MCU_2



Ⅲ. Algorithm

A. Algorithm Overview

Explain with tables or state diagram. Also, should use explain by

- Listing all necessary states (states, input, output etc) to implement this design problem.
- Listing all necessary conditional FLAGS (WIN_BREAK etc) for programming.
- Explaining additional conditions/configurations that were not explained in Introduction.

Sensor Unit: MCU_1

All sensors operate independently and have independent OUTPUTs for each sensor. Therefore, the OUTPUT occurring in the Normal and Security modes is represented by an independent state table for each sensor. And the initial state is the S0 state of all graphs.

- Ultra Sonic sensor

Present	Next State (Button, Outsied Person)				Output	
	00	01	10	11	Out_LED_flag	Visit_LOG_flag
S0	S0	S1	S2	S3	0	0
S1	S0	S1	S3	S3	1	0
S2	S2	S3	S0	S1	0	0
S3	S2	S3	S1	S1	1	1

State Define

S0: Normal mode, No outside person

S1: Normal mode, outside person

S2: Security mode, No outside person

S3: Security mode, outside person

Input: Button, Outside Person

Output: Out_LED, Visit_LOG

Embedded Controller

- PIR sensor

Present	Next State (Button, Inside Person)				Output		
	00	01	10	11	Inside_3sec_LED	Inside_LED	Siren flag
S0	S0	S1	S2	S3	0	0	0
S1	S0	S1	S3	S3	1	0	0
S2	S2	S3	S0	S1	0	0	0
S3	S2	S3	S1	S1	0	1	1

State Define

S0: Normal mode, No Inside person

S1: Normal mode, Inside person

S2: Security mode, No Inside person

S3: Security mode, Inside person

Input: Button, Inside Person

Output: Inside_3sec_LED, Inside_LED, Siren flag

- IR sensor

Present	Next State (Button, Window state)				Output
	00	01	10	11	Siren flag
S0	S0	S1	S2	S3	0
S1	S0	S1	S3	S3	0
S2	S2	S3	S0	S1	0
S3	S2	S3	S1	S1	1

State Define

S0: Normal mode, Window Close

S1: Normal mode, Window Open

S2: Security mode, Window Close

S3: Security mode, Window Open

Input: Button, Window state

Output: Siren flag

Embedded Controller

- Light Intensity sensor

Present	Next State (Button, Daylight)				Output
	00	01	10	11	Stepper_flag
S0	S0	S1	S2	S3	1
S1	S0	S1	S3	S3	1
S2	S2	S3	S0	S1	0
S3	S2	S3	S1	S1	0

State Define

S0: Normal mode, daytime

S1: Normal mode, nighttime

S2: Security mode, daytime

S3: Security mode, nighttime

Input: Button, Daylight

Output: Stepper_flag

- Sound sensor

Present	Next State (Button, Window state)				Output
	00	01	10	11	Siren_flag
S0	S0	S1	S2	S3	0
S1	S0	S1	S3	S3	0
S2	S2	S3	S0	S1	0
S3	S2	S3	S1	S1	1

State Define

S0: Normal mode, No Window break

S1: Normal mode, Window break

S2: Security mode, No Window break

S3: Security mode, Window break

Input: Button, Window state

Output: Siren_flag

Actuator/Display Unit: MCU_2

MUC2 also presented the State Table independently because all actuators and displays are operated independently for Input, such as MCU1.

Embedded Controller

- Inside LED

Present	Next State (Inside LED flag)		Output
	0	1	Inside LED
S0	S0	S1	0
S1	S0	S1	1

State Define	Input: Inside LED flag
S0: Inside LED OFF S1: Inside LED ON	Output: Inside LED

Present	Next State (Inside_3sec_LED)		Output
	0	1	Inside LED
S0	S0	S1	0
S1	S2	S2	1
S2	S3	S3	1
S3	S0	S0	1

State Define	Input: Inside_3sec_LED
S0: LED OFF S1: LED 1sec S2: LED 2sec S3: LED 3sec	Output: Inside LED

Embedded Controller

- Outside LED

Present	Next State (Out LED flag, Visit LOG)				Output	
	00	01	10	11	Outside LED	Visit LOG
S0	S0	X	S1	S2	0	0
S1	S0	X	S1	S2	1	0
S2	S0	X	S1	S2	1	1

State Define	Input: Out LED flag, Visit LOG
S0: LED OFF S1: LED ON S2: LED ON + Visit LOG	Output: Outside LED

- Stepper Motor

Present	Next State (Stepper flag, Curtain flag)				Output
	00	01	10	11	Curtain state
S0	S2	S2	S1	S2	0(OPEN)
S1	S2	S2	S2	S0	1(CLOSE)

State Define	Input: Stepper flag, Curtain flag
S0: Curtain open S1: Curtain close S2: Keeping the current state	Output: Curtain state

Embedded Controller

- 7-segment

Present	Next State (mode, Siren flag)				Output
	00	01	10	11	7-segment Output
S0	S0	X	S4	S4	1
S1	S0	X	S4	S5	2
S2	S0	X	S4	S1	3
S3	S0	X	S4	S2	4
S4	S0	X	S4	S3	5
S5	S0	X	S4	S5	BLINK

State Define

S0: 7-segment display by 1

S1: 7-segment display by 2

S2: 7-segment display by 3

S3: 7-segment display by 4

S4: 7-segment display by 5

S5: 7-segment display by 0 Blink

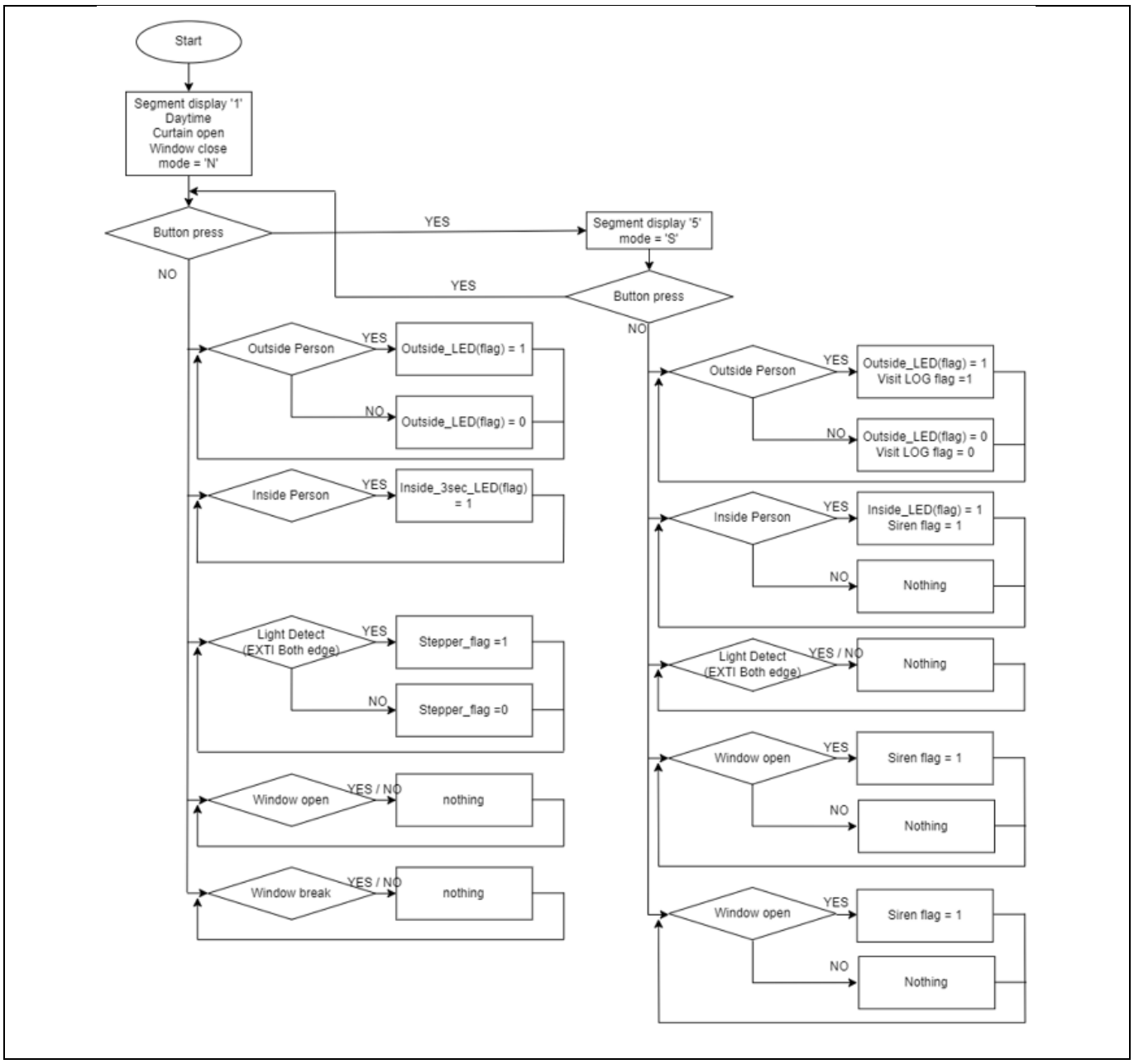
Input: mode, Siren flag

Output: 7-segment Output

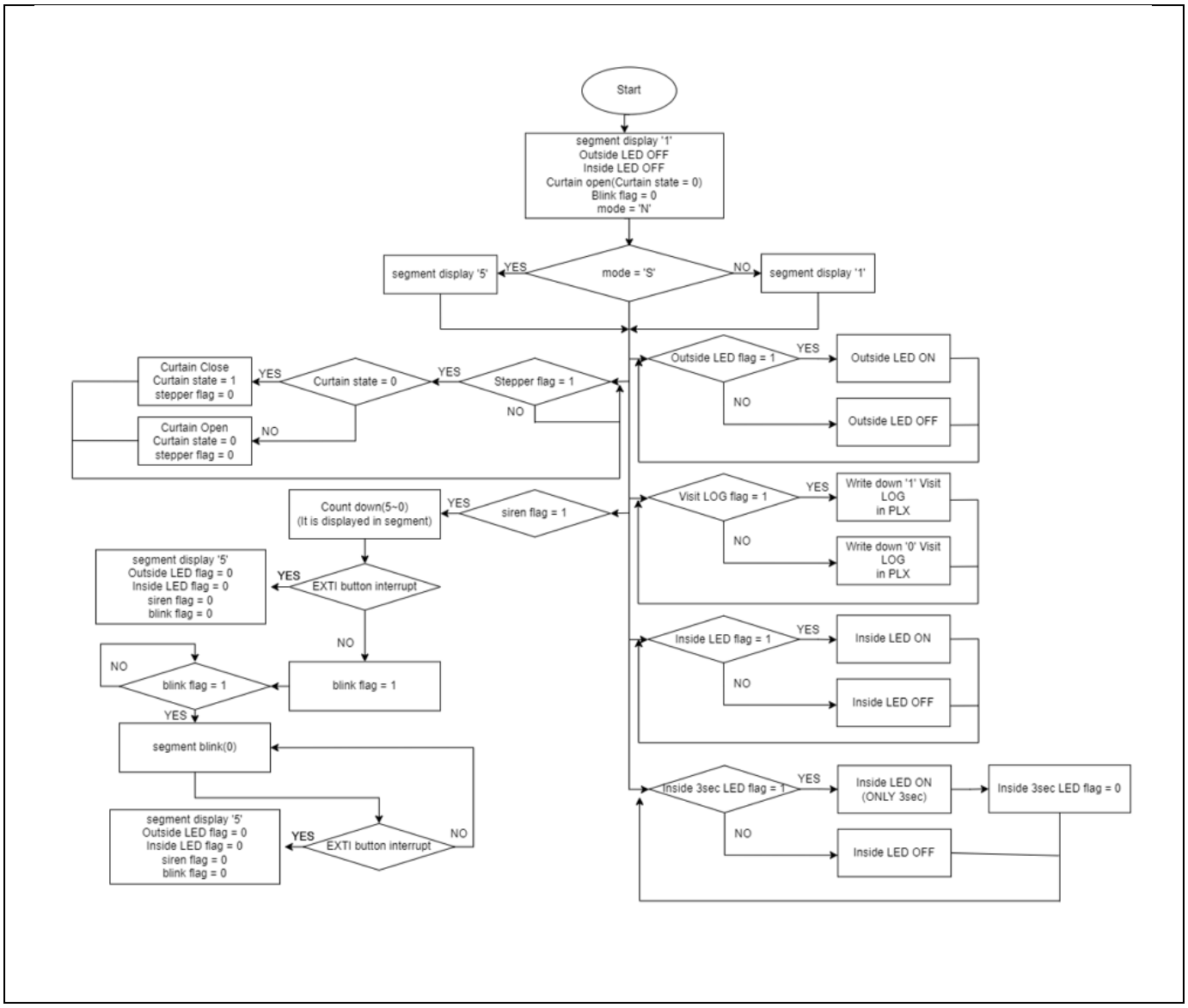
B. Flow Chart [30pt]

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

Sensor Unit: MCU_1



Actuator/Display Unit: MCU_2



IV. Demonstration

DEMO LINK

https://youtu.be/1U6gpvGVZiU

partner: Kim Ji Sung

V. Conclusion & TrobleShooting

A. TrobleShooting

Q. Due to the priority of other interrupts, the order of regular ADC changes when using regular ADC.

A. Increase the priority of ADC and use it. Alternatively, Use an Injected ADC. (I didn't use it in LAB.)

Q. How to solve the error in Setup.

A. Refrain from using the printf function in the interrupt handler.

And Minimize processing in the interrupt handler.

Q. When the buffer value is not sent smoothly during communication.

A.

```
if(modebuf[0] != 'N' || modebuf[0] != 'S' || modebuf[1] != 'X' || modebuf[2] != 'X' || modebuf[3] != '*' )
    memset(modebuf,0,sizeof(uint8_t)*4);
```

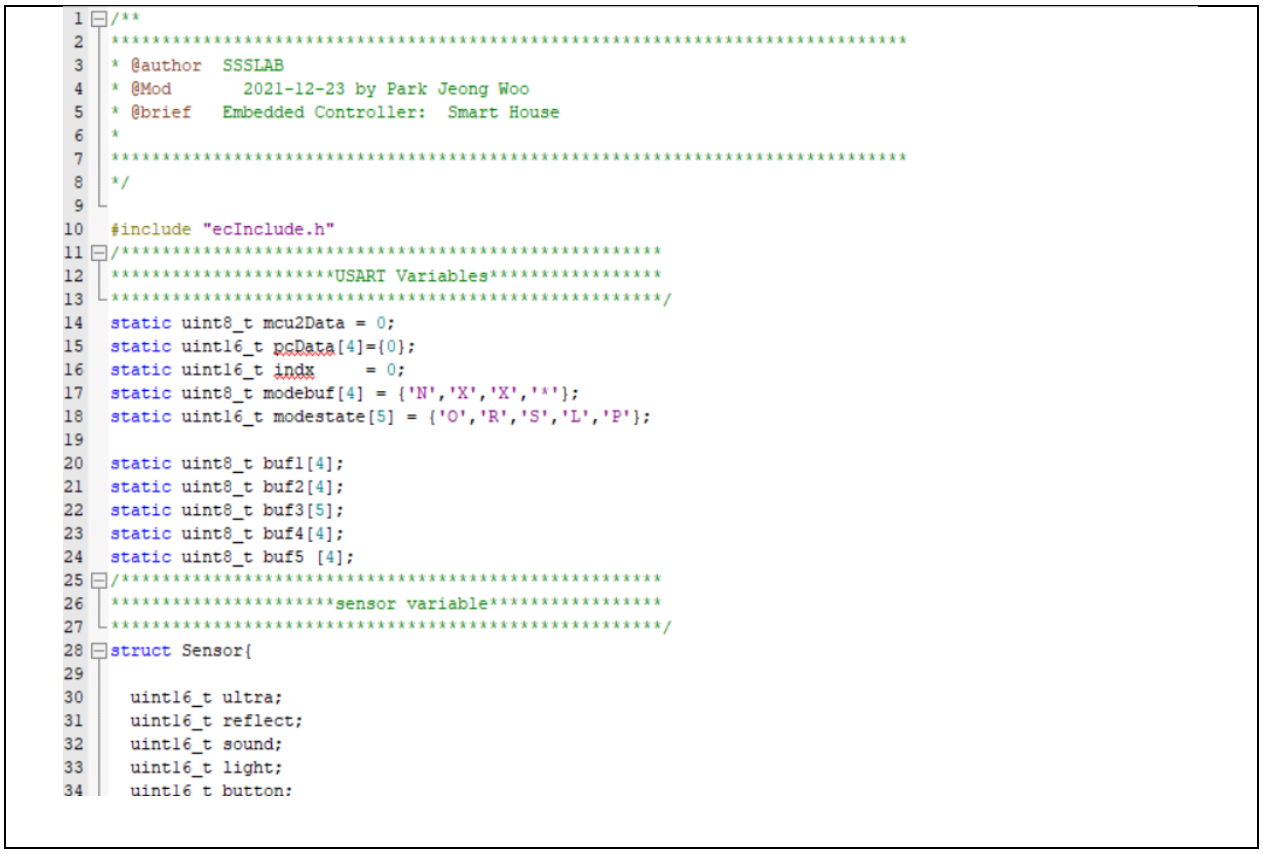
All buffers were initialized if the desired value was not input when the button was pressed through the above processing statement.

B. Conclusion

In this experiment, all the sensors used in the previous experiment were used. It was confirmed that the operating principles of all sensors were understood and the sensors were operating normally. And I found that the priority of interrupt is important when using many sensors.

VI. Appendix

- MCU1.c



```c
/**
********************************************************************************
* @author  SSSLAB
* @Mod       2021-12-23 by Park Jeong Woo
* @brief   Embedded Controller:  Smart House
*
********************************************************************************
*/

#include "ecInclude.h"
/*****************************************************
**********************USART Variables*****************
*****************************************************/
static uint8_t mcu2Data = 0;
static uint16_t pcData[4]={0};
static uint16_t indx     = 0;
static uint8_t modebuf[4] = {'N','X','X','*'};
static uint16_t modestate[5] = {'O','R','S','L','P'};

static uint8_t buf1[4];
static uint8_t buf2[4];
static uint8_t buf3[5];
static uint8_t buf4[4];
static uint8_t buf5 [4];
/*****************************************************
**********************sensor variable*****************
*****************************************************/
struct Sensor{

  uint16_t ultra;
  uint16_t reflect;
  uint16_t sound;
  uint16_t light;
  uint16_t button;
```

Embedded Controller

```
  uint16_t pir;

};


static struct Sensor flag;
static struct Sensor prev_flag;
static struct Sensor value;

void flag_init(void);
void flag_init(void)
{
  flag.ultra = 0;
  flag.reflect = 0;
  flag.sound  = 0;
  flag.light  = 0;
  flag.button = 0;
  flag.pir    = 0;

  prev_flag.ultra = 0;
  prev_flag.reflect = 0;
  prev_flag.sound  = 0;
  prev_flag.light  = 0;
  prev_flag.button = 0;
  prev_flag.pir    = 0;
}


static uint16_t light_value_prev = 0;
static uint8_t outside_flag_count = 0;
static uint8_t reflect_flag_count = 0;
/*******************************************************
**********************Other variable*******************
*******************************************************/

static uint16_t seq[2] = {10, 11};
static uint16_t ovf_cnt = 0;
static float distance = 0.f;
static double timeInterval = 0;
static double timeSt = 0;
static double timeEnd= 0;
static uint8_t print_count= 0;
static uint8_t ADC_seq_flag= 0;

static uint8_t visitor_flag =0;
static uint8_t siren_flag =0;

void save_buf(uint8_t sensor, uint8_t state);
static uint16_t light_count = 0;
static uint32_t distance_count = 10;
static volatile uint16_t flag_ultra = 0;
static volatile uint16_t prev_flag_ultra = 0;
static uint16_t reflect_count = 0;

void setup(void);

int main (void){

  setup();                                    // Initialization
  printf("Welcome to Smart House\r\n");       // Finish Init
  delay_ms(500);

  USART_write(USART1,(unsigned char*) "CLEARSHEET\r\n",12);
  USART_write(USART1,(unsigned char*) "LABEL,Date,Time,Timer,Reflect,Sound,Dist,VISIT_LOG,SINEN\r\n",59);

```

Embedded Controller

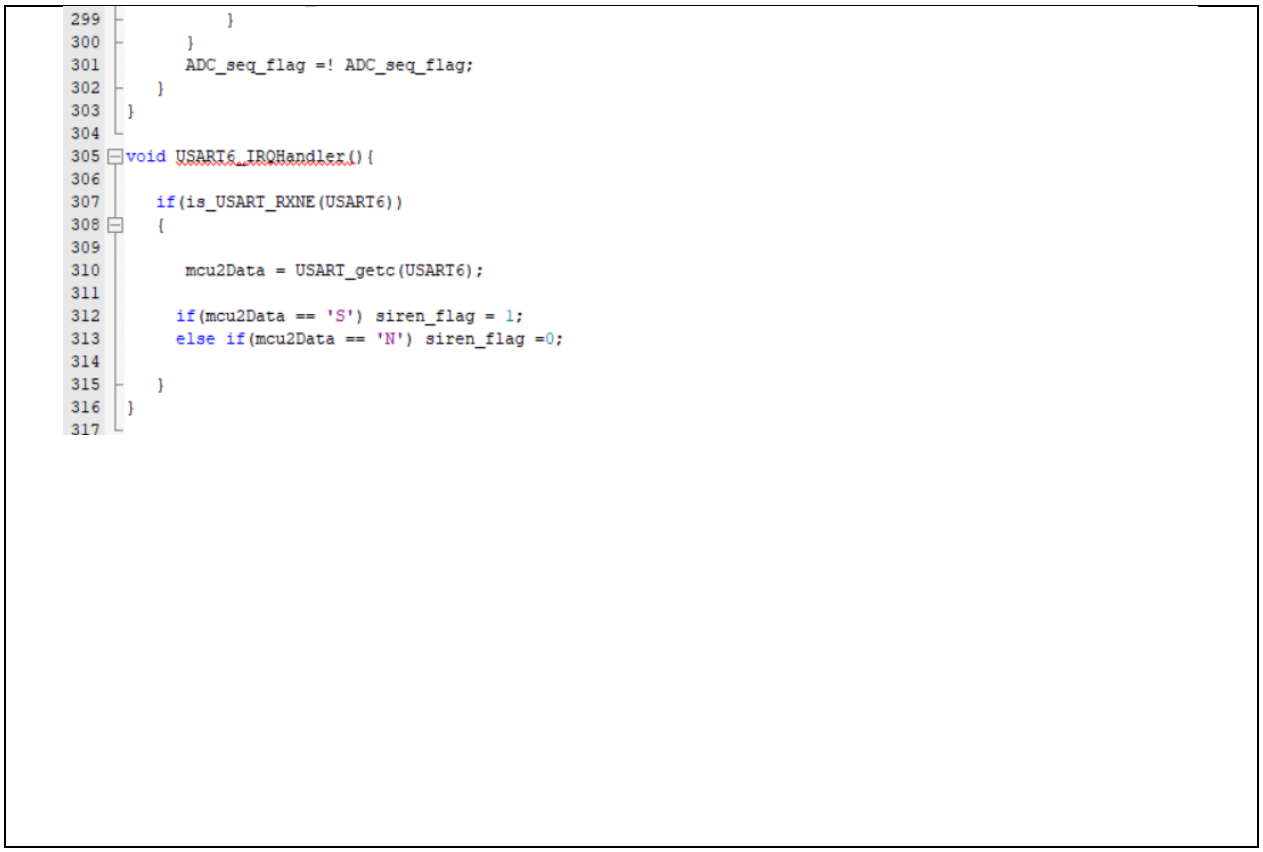
```
100  while(1){
101
102    sprintf(buf1, "%d", value.reflect);
103    sprintf(buf2, "%d", value.sound);
104    sprintf(buf3, "%f", distance);
105    sprintf(buf4, "%d", visitor_flag);
106    sprintf(buf5, "%d", siren_flag);
107
108     if(print_count == 20){
109
110          USART_write(USART1,(unsigned char*) "DATA,DATE,TIME,TIMER,",21);  // transmit char to USART6
111          USART_write(USART1,buf1,4);
112          USART_write(USART1,(unsigned char*) ",",1); // transmit char to USART6
113          USART_write(USART1,buf2,4);
114          USART_write(USART1,(unsigned char*) ",",1); // transmit char to USART6
115          USART_write(USART1,buf3,5);
116          USART_write(USART1,(unsigned char*) ",",1); // transmit char to USART6
117          USART_write(USART1,buf4,1);
118          USART_write(USART1,(unsigned char*) ",",1); // transmit char to USART6
119          USART_write(USART1,buf5,1);
120          USART_write(USART1,(unsigned char*) ",AUTOSCROLL_20\r\n",16); // transmit char to USART6
121
122          print_count =0;
123      }
124    }
125  }
126
127  void setup(void)
128  {
129
130    RCC_PLL_init();                              //PLL Clock ON
131    Systick_Init(1);                             //Systick ON
132
133    USART_init(USART2, 9600);                    //USART2 PA2 (D1) - RXD , PA3  (D0) - TXD
134    USART_begin(USART1, GPIOA,9,GPIOA,10, 9600); //USART1 PA9 (D8) - RXD , PA10 (D2) - TXD
135    USART_begin(USART6, GPIOA,11,GPIOA,12, 9600); //USART6 PA11(D12) - RXD , PA12 (D13) - TXD
136
137    GPIO_init(GPIOC, 13, INPUT);                   //Button Input PC13
138    GPIO_init(GPIOA, 0, INPUT);                  //Light Sensor Input PA0 (A0)
139    GPIO_init(GPIOA, 1, INPUT);                  //PIR Sensor Input PA1 (A1)
140
141    EXTI_init(GPIOC, BUTTON_PIN, FALL_EXTI, 2);  //Button EXTI PC13
142    EXTI_init(GPIOA, 0, BOTH_EXTI, 2);           //Light Sensor Input PA0 (A0)
143    EXTI_init(GPIOA, 1, RISE_EXTI, 2);           //PIR sensor EXTI PA1 (A1)
144
145    ADC_init(GPIOC, 1, TRGO);                    //Reflect Sensor PC1 (A4)
146    ADC_init(GPIOC, 0, TRGO);                    //Sound Sensor PC0 (A5)
147    ADC_continue(SINGLE);                        //timer 3 100msec
148    ADC_sequence(2,seq);                         //when i use sequence
149    ADC_start();                                 //Star ADC
150
151    PWM_t trig;                                  //PWM1 for trig
152    PWM_init(&trig,GPIOA,5);                     //timer 2 100msec
153    PWM_period_us(&trig,100000);                 //PWM of 100ms period.
154    PWM_pulsewidth_us(&trig,10);                 //Ultrasonic trig puls (D13)
155
156    IC_t echo;                                   //Input Capture for echo //timer 4 1msec
157    ICAP_init(&echo,GPIOB,6);                    //ICAP init as PB6 (D10)
158    ICAP_counter_us(&echo, 10);                  //ICAP counter step time as 10us
159    ICAP_setup(&echo, 1, RISE_TIM);              //TIM4_CH1 as IC1 , rising edge detect
160    ICAP_setup(&echo, 2, FALL_TIM);              //TIM4_CH2 as IC2 , falling edge detect
161    TIM_INT_enable(TIM4);                        //TIM4 interrup init
162
163    TIM_INT_init(TIM2,100);                      //Light sensor interrup timer
164    TIM_INT_enable(TIM2);                        //TIM2 enable
165
```

Embedded Controller

```
  flag_init();
}

void save_buf(uint8_t sensor, uint8_t state)
{
  modebuf[1] = sensor;
  modebuf[2] = state + '0';
  modebuf[3] = '*';

  if(modebuf[0] == 'S' && sensor == 'O' && state == 1)  visitor_flag = 1;
  else if(modebuf[0] == 'S' && sensor == 'O' && state == 0)  visitor_flag = 0;
  USART_write(USART6,modebuf,4);
}
//Button press
void EXTI15_10_IRQHandler(void) {
   if (is_pending_EXTI(BUTTON_PIN))
   {
      flag.button =! flag.button;

      if (flag.button == 0) modebuf[0] = 'N';
      else if (flag.button == 1) modebuf[0] = 'S';

      modebuf[1] = 'X';
      modebuf[2] = 'X';
      modebuf[3] = '*';

      USART_write(USART6,modebuf,4);

      if(modebuf[0] != 'N' || modebuf[0] != 'S' || modebuf[1] != 'X' || modebuf[2] != 'X' || modebuf[3] != '*' ) memset(modebuf,0,sizeof(uint8_t)*4);
      clear_pending_EXTI(BUTTON_PIN); // cleared by writing '1'
   }
}
//PIR sensor
void EXTI1_IRQHandler(){
  if(is_pending_EXTI(1)){
      flag.pir = 1;
      save_buf(modestate[4],flag.pir);
      clear_pending_EXTI(1); // cleared by writing '1'
 }
}
 //Light sensor
void EXTI0_IRQHandler(void) {
   if (is_pending_EXTI(0)){

      delay_ms(10);
      value.light = GPIO_read(GPIOA,0);
      clear_pending_EXTI(0);
 }
}
 //Light sensor

void TIM2_IRQHandler (void){
 if(is_UIF(TIM2))
 {
   print_count ++;

   if(light_value_prev != value.light)  light_count = 0;
   else if(light_value_prev == value.light) light_count ++;

    if(light_count == 20){
      flag.light = value.light;
      light_count = 0;
    }
    if(prev_flag.light != flag.light) save_buf(modestate[3],flag.light);

    prev_flag.light = flag.light;
```

Embedded Controller

```
    light_value_prev = value.light;

   clear_UIF(TIM2);
 }
}
//Ultra Sonic Senor
void TIM4_IRQHandler(void){
   if(is_UIF(TIM4)){                     // Update interrupt
      ovf_cnt++;                         // overflow count
      clear_UIF(TIM4);                   // clear update interrupt flag
   }
   if(is_CCIF(TIM4,1)){                  // TIM4_Ch1 (IC1) Capture Flag. Rising Edge Detect
      timeSt = TIM4->CCR1;               // Capture TimeStart from CC1
      clear_CCIF(TIM4,1);                // clear capture/compare interrupt flag
   }
   else if(is_CCIF(TIM4,2)){             // TIM4_Ch2 (I2) Capture Flag. Falling Edge Detect
      timeEnd = TIM4->CCR2;              // Capture TimeEnd from CC4
      timeInterval = 10*((timeEnd - timeSt) + (ovf_cnt * ((TIM4->ARR)+1)));      // Total time of echo pulse
      distance = (float) (timeInterval*340)/(2*10000); // [cm]
      ovf_cnt = 0;                       // overflow reset
      if(distance < 30) distance_count--;

      if(outside_flag_count == 10){

          if (distance_count < 5) flag_ultra = 1;
          else  flag_ultra = 0;

         if(prev_flag_ultra != flag_ultra) save_buf(modestate[0], flag_ultra);

            prev_flag_ultra = flag_ultra;
            outside_flag_count =0;
            distance_count = 10;
      }
      outside_flag_count++;
      clear_CCIF(TIM4,2);                          // clear capture/compare interrupt flag
   }
}

//sound and reflect sensors

void ADC_IRQHandler(void){

   if(is_ADC_OVR()){       //after finishing sequence
     clear_ADC_OVR();
   }
   if(is_ADC_EOC()){
      if(ADC_seq_flag==0){
         value.sound = ADC_read();

         flag.sound = 1;
        if(value.sound > 1000) save_buf(modestate[2],flag.sound);

      } else if(ADC_seq_flag==1){
        value.reflect = ADC_read();
        reflect_flag_count++;
       if(value.reflect < 500) reflect_count++;

       if(reflect_flag_count == 10){

          if (reflect_count >= 7) flag.reflect = 1;
          else if (reflect_count < 6)  flag.reflect= 0;

          if(prev_flag.reflect != flag.reflect) save_buf(modestate[1],flag.reflect);
            prev_flag.reflect = flag.reflect;

           reflect_flag_count =0;
           reflect_count = 0;
```



```
299             }
300         }
301         ADC_seq_flag =! ADC_seq_flag;
302     }
303 }
304
305 void USART6_IRQHandler(){
306
307     if(is_USART_RXNE(USART6))
308     {
309
310         mcu2Data = USART_getc(USART6);
311
312       if(mcu2Data == 'S') siren_flag = 1;
313       else if(mcu2Data == 'N') siren_flag =0;
314
315     }
316 }
317
```

- MCU2.c

```c
/**
******************************************************************************
* @author  SSSLAB
* @Mod       2021-11-27 by Park Jeong Woo
* @brief   Embedded Controller:  MCU2
*
******************************************************************************
*/

#include "ecInclude.h"

PWM_t pwm2;
PWM_t pwm1;

uint8_t mcu2Data = 0;
uint8_t pcData = 0;
int idx =0;
uint8_t buf[4] = {0};
int maxBuf=10;
uint8_t buffer[100]={0,};
uint8_t buffer2 = '\r\n';
int endChar = 13;

void mode_scan(uint8_t *buf);

static volatile uint16_t stepper_flag = 0;
static volatile uint16_t siren_flag = 0;

static volatile uint32_t led_3sec_flag = 0;
static volatile uint32_t led_count = 0;

static volatile int count = 0;
static volatile uint32_t blink_flag = 0;
static volatile uint32_t segment_blink = 1;
static volatile uint32_t stop_flag = 0;
static volatile uint32_t curtain_state = 0;

static uint8_t state[2] = {'S','N'};
static uint8_t state_flag = 0;


static uint8_t N_stepper_flag = 0;
void setup(void);

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

    // Inifinite Loop ----------------------------------------------------------
    while (1){

     if(state_flag){
       USART_write(USART6,&state[0], 1);
       state_flag = 0;
     }
   if(stepper_flag == 1 && curtain_state == 0)
   {
       Stepper_step(2048, 0,FULL);   //close
        curtain_state = 1;
        stepper_flag = 0;
   }
    else if(stepper_flag == 1 && curtain_state == 1)
    {
        Stepper_step(2048, 1,FULL);  //open
        curtain_state = 0;
```

Embedded Controller

```
        stepper_flag = 0;
    }
      else if(blink_flag == 1 &&  curtain_state == 1){

         Stepper_step(2048, 1, FULL);
         curtain_state = 0;
         N_stepper_flag = 0;

    }
    else if(N_stepper_flag == 1 &&  curtain_state == 0){

         Stepper_step(2048, 0, FULL);
         curtain_state = 1;
         N_stepper_flag = 0;
    }
    }
 }
// Initialiization
void setup(void)
{
  RCC_PLL_init();
  Systick_Init(1);

  // USART congfiguration
  USART_init(USART2, 9600);
  USART_begin(USART6, GPIOA,11,GPIOA,12, 9600);   // PA11 (D12) - RXD , PA12 (D13) - TXD

  //Stepper motor setting                          // MCU2

  Stepper_init(GPIOA,0,GPIOA,1,GPIOA,4,GPIOB,0);  // (D6) (D5) (D4) (D3) Stepper GPIO pin initialization
  Stepper_setSpeed(12,FULL);                      //  set stepper motor speed


  TIM_INT_init(TIM3,1000);
  TIM_INT_enable(TIM3);

  TIM_INT_init(TIM2,1000);
  TIM_INT_enable(TIM2);



  EXTI_init(GPIOC, BUTTON_PIN, FALL, 1);
  //Button
  GPIO_init(GPIOC, BUTTON_PIN, INPUT);


  //LED + segment setting
  GPIO_init(GPIOB,3,OUTPUT); //IN_LED
  GPIO_init(GPIOA,10,OUTPUT); //OUT_LED


  sevensegment_init();
  sevensegment_decode(1);


}
void EXTI15_10_IRQHandler(void) {
   if (is_pending_EXTI(BUTTON_PIN))
   {

       siren_flag = 0;
       sevensegment_decode(5);
       GPIO_write(GPIOB,3,0);
       GPIO_write(GPIOA,10,0);
       count = 0;
```

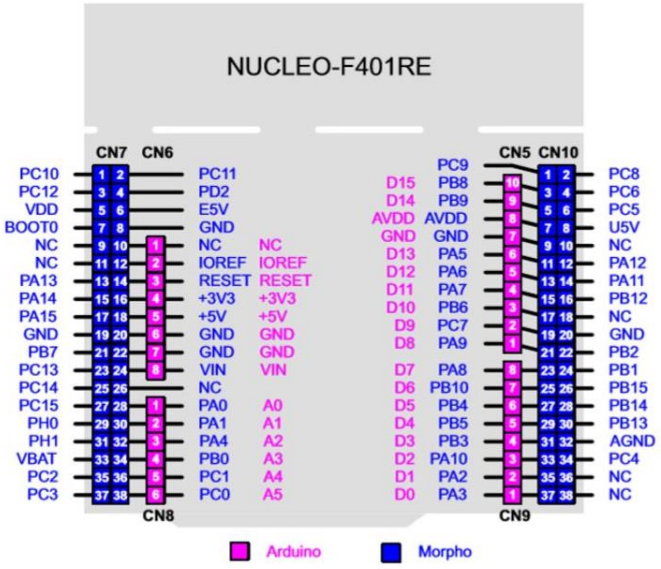
```
133         blink_flag = 0;
134         segment_blink = 1;
135         USART_write(USART6,&state[1], 1);
136         clear_pending_EXTI(BUTTON_PIN); // cleared by writing '1'
137       }
138   }
139
140 void USART6_IRQHandler(){
141
142     if(is_USART_RXNE(USART6))
143     {
144
145       mcu2Data = USART_getc(USART6);
146
147       if(mcu2Data == '*') {
148         USART_write(USART2, buf, 4);
149         printf("\r\n");
150         mode_scan(buf);
151         idx = 0;
152       }
153       else{
154         if(idx>maxBuf){
155           idx =0;
156           memset(buf, 0, sizeof(char) * maxBuf);
157           printf("ERROR : Too long string\r\n");
158         }
159         buf[idx] = mcu2Data;
160         idx++;
161       }
162
163     }
164   }
165
166
167 void TIM2_IRQHandler (void){
168   if(is_UIF(TIM2))
169   {
170
171     if(led_3sec_flag  ==1){
172
173           GPIO_write(GPIOB,3,1);
174           led_count++;
175
176       if(led_count == 4) {
177
178         GPIO_write(GPIOB,3,0);
179         led_count =0;
180         led_3sec_flag  = 0;
181       }
182     }
183     clear_UIF(TIM2);
184   }
185 }
186 void TIM3_IRQHandler (void){
187   if(is_UIF(TIM3))
188   {
189
190     if(siren_flag == 1){
191       if(blink_flag == 0){
192
193         count++;
194         sevensegment_decode(5-count);
195         if(count == 5) blink_flag = 1;
196       }
197
198
```

Embedded Controller

```
    if(blink_flag == 1){

      state_flag = 1;
      segment_blink =! segment_blink;
      sevensegment_decode(segment_blink*10);


     }
    }
   clear_UIF(TIM3);
 }
}

void mode_scan(uint8_t *buf)
{
  if(buf[0] == 'N'){

    sevensegment_decode(1);
    if(buf[1] == 'X') N_stepper_flag = 1;
    if(buf[1] == 'P'){

       switch (buf[2]){

         case '1' : led_3sec_flag = 1; break;//IN_LED ON
       }
    }
    else if(buf[1] == 'O'){

       switch (buf[2]){
         case '0' : GPIO_write(GPIOA,10,0);  break;  //OUT_LED OFF
         case '1' : GPIO_write(GPIOA,10,1);  break;  //OUT_LED ON
       }
    }
    else if(buf[1] == 'L'){
       switch (buf[2]){
         case '0' : stepper_flag = 1;   break;
         case '1' : stepper_flag = 1;   break;
       }
    }


  }
  else if(buf[0] == 'S'){

    if(siren_flag == 0)sevensegment_decode(5);
    if(buf[1] == 'P'){
       switch (buf[2]){
         case '1' : GPIO_write(GPIOB,3,1);  siren_flag =1 ; break;
       }
    }
    else if(buf[1] == 'R'){
       switch (buf[2]){
         case '0' : siren_flag =1 ; break;
       }
    }
    else if(buf[1] == 'O'){

       switch (buf[2]){
         case '0' : GPIO_write(GPIOA,10,0);  break;
         case '1' : GPIO_write(GPIOA,10,1);  break;
       }
    }
    else if(buf[1] == 'S'){
       switch (buf[2]){

         case '1' :  siren_flag =1;  break;
       }
    }
  }
}
```

- Pinmap

Figure 18. NUCLEO-F401RE



Timer PinOut Map

Advanced Timer

Timer	Channel	Port	Pin
1	1	A	8
	1N	A	7
		B	13
	2	A	9
	2N	B	0
		B	14
	3	A	10
	3N	B	1
		B	15
	4		
	4N		

General Purpose Timer

Timer	Channel	Port	Pin
2	1	A	0
		A	5
		A	15
	2	A	1
		B	3
	3	B	10
	4		
3	1	A	6
		B	4
		C	6
	2	B	5
		C	7
	3	C	8
	4	C	9
4	1	B	6
	2	B	7
	3	B	8
	4	B	9

ADC1

Channel	Port	Pin
0	A	0
1	A	1
2		
3		
4	A	4
5	A	5
6	A	6
7	A	7
8	B	0
9	B	1
10	C	0
11	C	1
12	C	2
13	C	3
14	C	4
15	C	5