REPORT

과　　목 : 마이크로프로세서2
과 제 명 : 텀프로젝트
담당교수 : 이원영
학　　과 : 전자공학과
학　　번 : 2018144002
이　　름 : 강송구(9조)
제 출 일 : 2022.12.20

텀 프로젝트
-IoT 스마트 하우스-

1. 제안 이유 및 필요성

2. 개발 방법 개요

3. 요소별 개발 내용 및 검증

4. 전체 시나리오에 따른 개발 내용 및 검증

5. 텀 프로젝트 수행에 대한 결론 및 기대효과

6. 소요된 부품 List 및 수행된 일정 정리

1. 제안 이유 및 필요성

시대가 발전해감에 따라 과학기술들이 발전하고 있고 발전한 과학기술은 우리 가정으로도 침투하여 사용자의 편의성을 도모하고, 삶을 더욱 윤택하게 만들어준다. 최근 IoT 기술이 가정에 접목되면서 노인 고독사 방지 시스템, 스마트폰으로 제어 가능한 전구, 스마트 매립형 플러그 등의 제품들이 등장하고 있는데, 우리는 이러한 IoT 제품들을 Atmega128을 이용하여 제작하고 이들을 모아 모델하우스를 만들어서 제어함으로써 IoT 기술에 대한 학습을 통해 이해력을 높이고, 산업적으로 매력이 있는 부분들을 만들려고 한다.

2. 개발 방법 개요

(a) 최종 개발 목표

Atmega128 두 개를 사용하여 하나는 도어락 시스템을 개발하고, 또 다른 하나는 모델하우스의 전반적인 센서와 액추에이터 제어에 사용하는 시스템을 개발하는 것이 목표이다. 앞으로 Atmega128을 이용한 도어락을 도어락이라고 칭하고, 모델하우스의 전반적인 제어를 하는 Atmega128을 제어부라 칭한다.

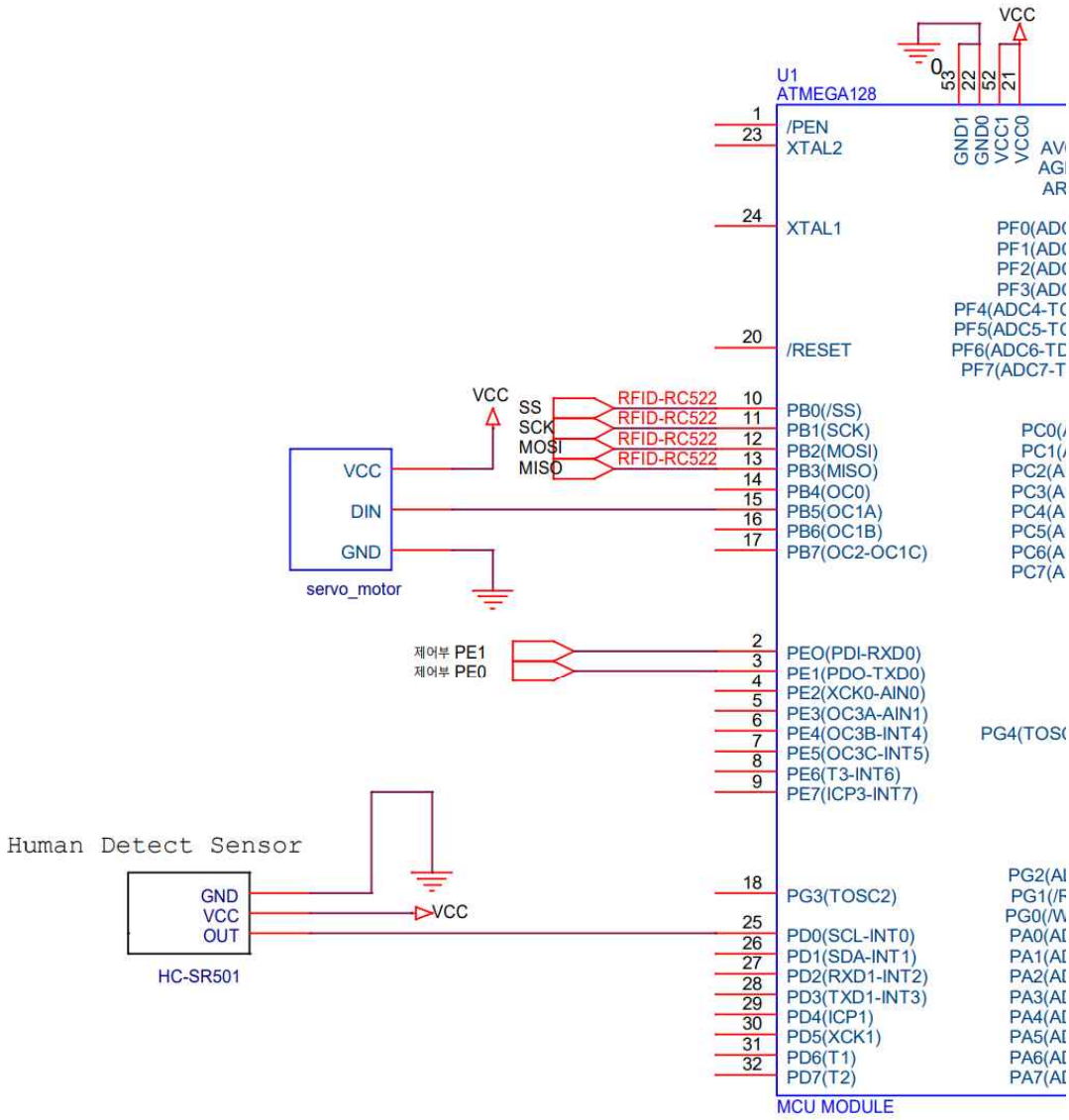
도어락에는 비밀번호를 입력받을 수 있는 4x4 keypad, Rfid 카드를 등록하여 비밀번호 대용으로 사용할 수 있게 하는 Rfid 모듈, 사람을 감지할 수 있는 인체감지센서, 사용자에게 정보를 알릴 수 있는 LCD, 문을 열고 닫을 수 있는 서보모터를 사용하였다.

제어부에는 편지가 도착하였을 때 알림을 줄 수 있는 초음파 거리 센서, 사용자의 스마트폰 등과 통신할 수 있는 블루투스 모듈, 밝기를 측정하여 창문을 열고 닫을 수 있는 광/조도 센서, 집의 가스 유출을 감지할 수 있는 유해가스 감지 센서, 집의 침수 유무를 확인할 수 있는 물 수위 센서, 사용자에게 제어용 인터페이스를 제공할 수 있는 LCD와 인터페이스를 조작할 수 있는 조이스틱, 멀티탭이 켜지고 꺼졌음을 알리는 LED, 창문을 열고 닫을 수 있는 서보모터, 절전과 전원 공급을 위한 8채널 릴레이 모듈, 에어컨을 표시할 FAN, 가습기를 사용하였다. 또한 제어부와 도어락에 안정된 전원을 공급하기 위하여 5v, 3.3v를 공급할 수 있는 전원공급기를 사용하였다.

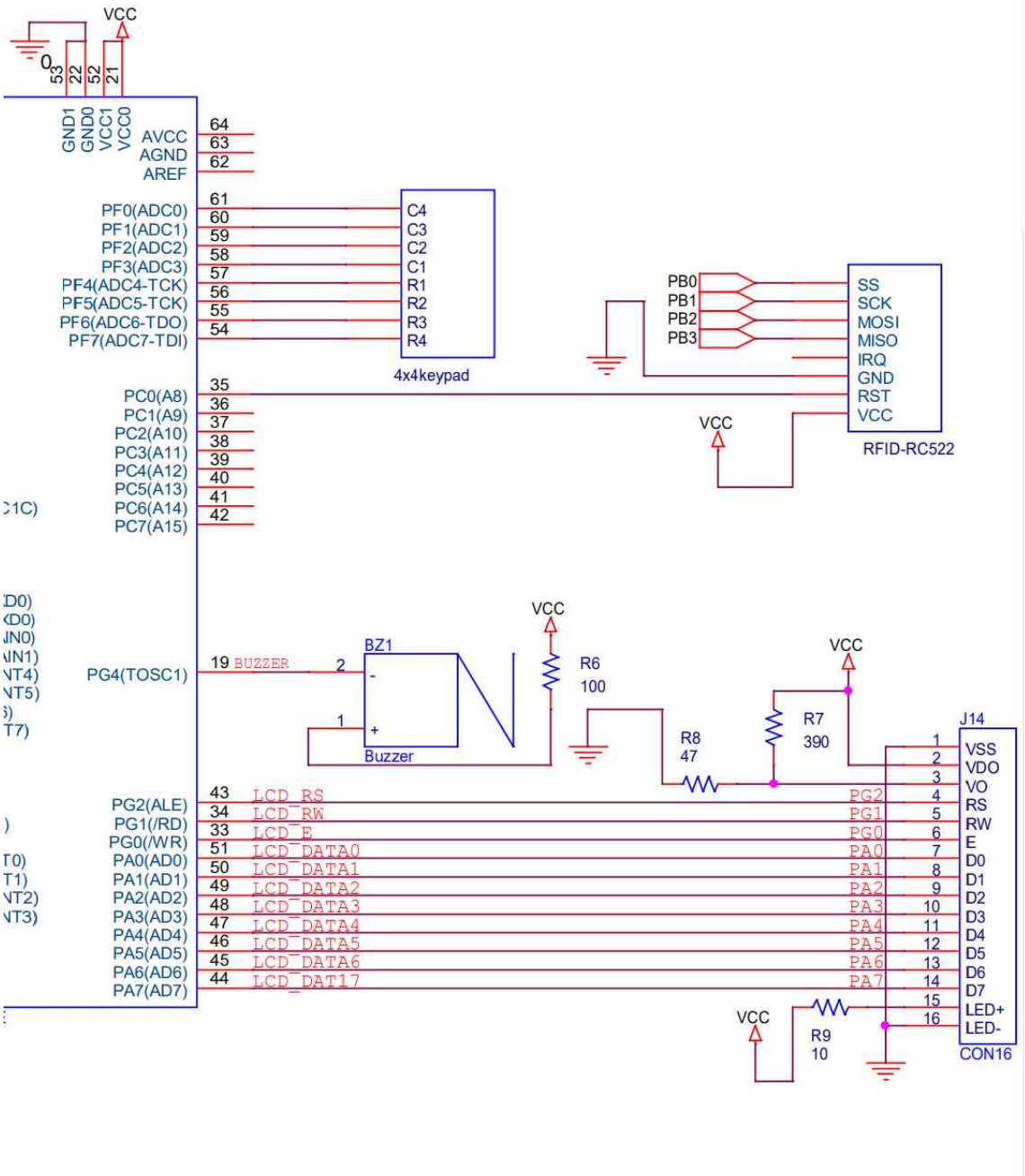
이를 통해 도어락과 제어부를 개발하여 정보를 공유하며 사용자에게 편의와 안전을 제공하는 IoT 스마트 하우스를 개발하는 것이 목표이다.

(b) 전체 시스템 구성도

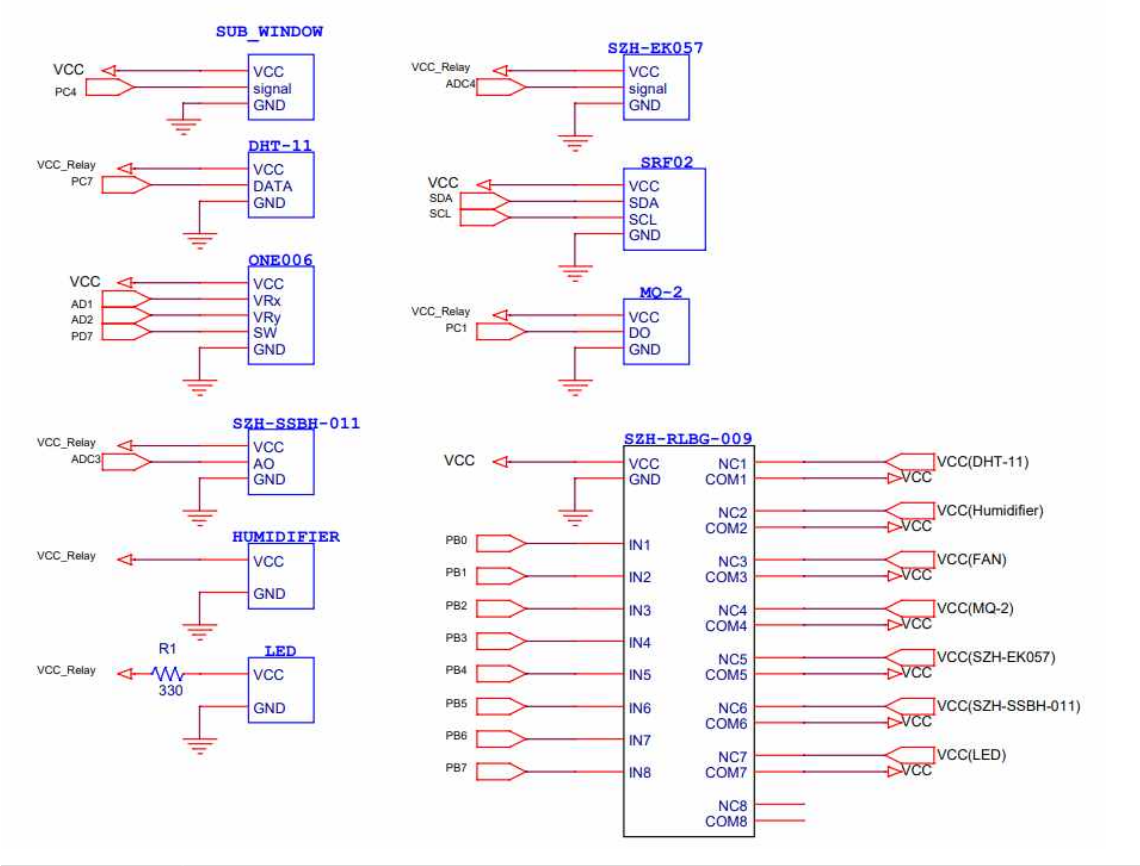
-도어락 H/W 구성도 (좌측)

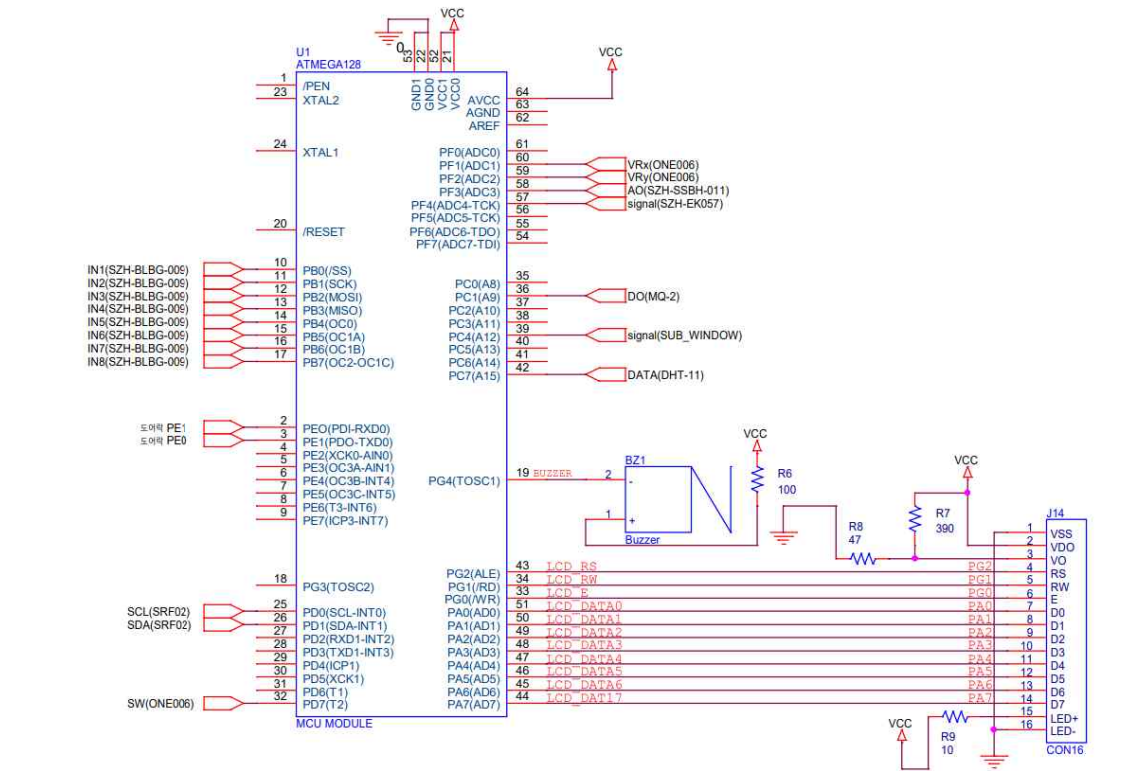


-도어락 H/W 구성도 (우측)


VCC
0
53
22
52
21
GND1
GND0
VCC1
VCC0
AVCC
AGND
AREF
64
63
62
PF0(ADC0)
PF1(ADC1)
PF2(ADC2)
PF3(ADC3)
PF4(ADC4-TCK)
PF5(ADC5-TCK)
PF6(ADC6-TDO)
PF7(ADC7-TDI)
61
60
59
58
57
56
55
54
C4
C3
C2
C1
R1
R2
R3
R4
4x4keypad
PB0
PB1
PB2
PB3
SS
SCK
MOSI
MISO
IRQ
GND
RST
VCC
RFID-RC522
PC0(A8)
PC1(A9)
PC2(A10)
PC3(A11)
PC4(A12)
PC5(A13)
PC6(A14)
PC7(A15)
35
36
37
38
39
40
41
42
PG4(TOSC1)
19 BUZZER
BZ1
2
1
Buzzer
R6
100
R8
47
R7
390
J14
PG2(ALE)
PG1(/RD)
PG0(/WR)
PA0(AD0)
PA1(AD1)
PA2(AD2)
PA3(AD3)
PA4(AD4)
PA5(AD5)
PA6(AD6)
PA7(AD7)
43
34
33
51
50
49
48
47
46
45
44
LCD_RS
LCD_RW
LCD_E
LCD_DATA0
LCD_DATA1
LCD_DATA2
LCD_DATA3
LCD_DATA4
LCD_DATA5
LCD_DATA6
LCD_DAT17
PG2
PG1
PG0
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
VSS
VDO
VO
RS
RW
E
D0
D1
D2
D3
D4
D5
D6
D7
LED+
LED-
CON16
R9
10

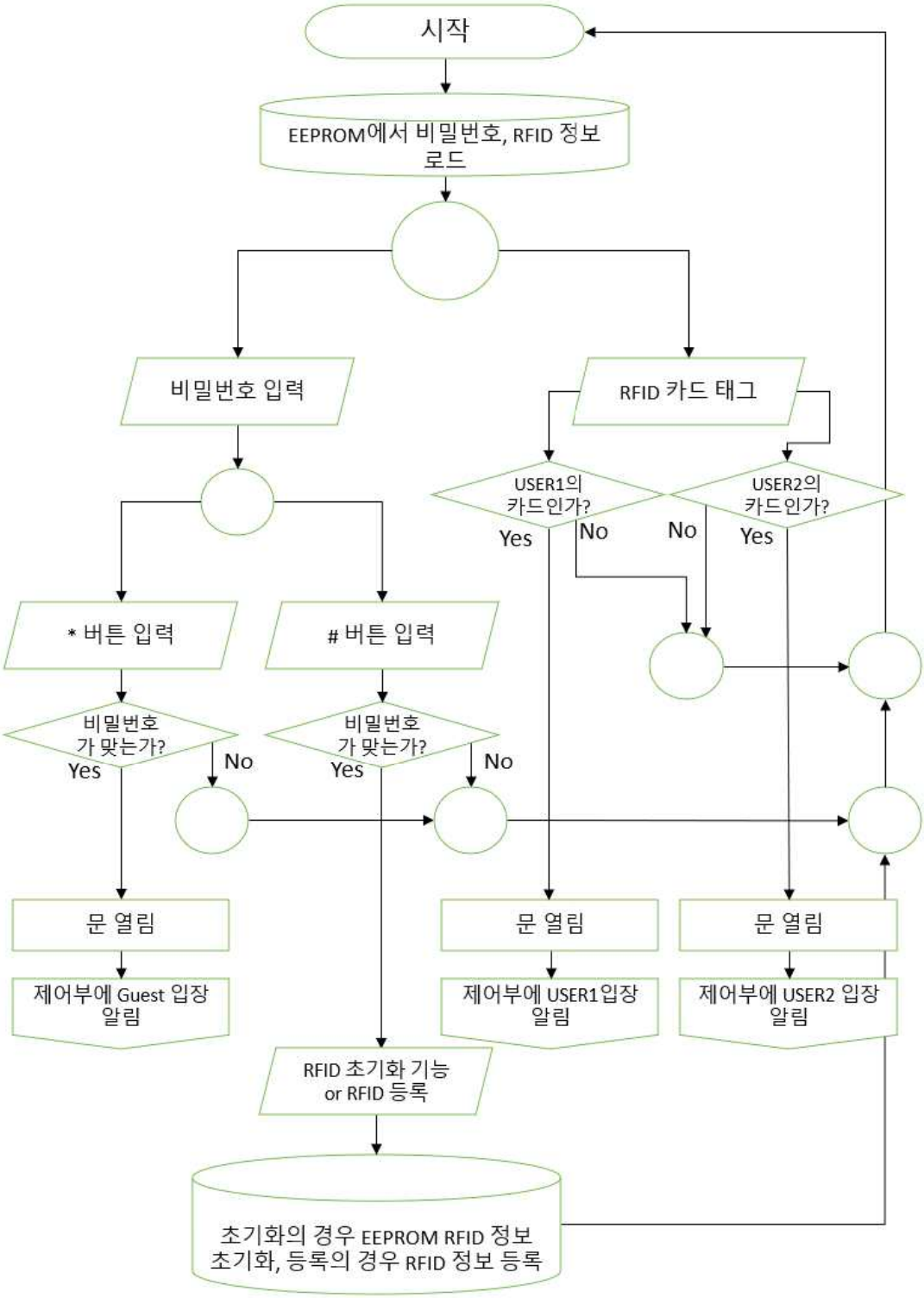
-제어부 H/W 구성도


SUB_WINDOW
VCC
PC4
VCC
signal
GND
SZH-EK057
VCC_Relay
ADC4
VCC
signal
GND
DHT-11
VCC_Relay
PC7
VCC
DATA
GND
SRF02
VCC
SDA
SCL
VCC
SDA
SCL
GND
ONE006
VCC
AD1
AD2
PD7
VCC
VRx
VRy
SW
GND
MQ-2
VCC_Relay
PC1
VCC
DO
GND
SZH-SSBH-011
VCC_Relay
ADC3
VCC
AO
GND
SZH-RLBG-009
VCC
VCC
GND
NC1
COM1
NC2
COM2
NC3
COM3
NC4
COM4
NC5
COM5
NC6
COM6
NC7
COM7
NC8
COM8
VCC(DHT-11)
VCC(Humidifier)
VCC(FAN)
VCC(MQ-2)
VCC(SZH-EK057)
VCC(SSBH-011)
VCC(LED)
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
IN1
IN2
IN3
IN4
IN5
IN6
IN7
IN8
HUMIDIFIER
VCC_Relay
VCC
GND
R1
330
LED
VCC_Relay
VCC
GND

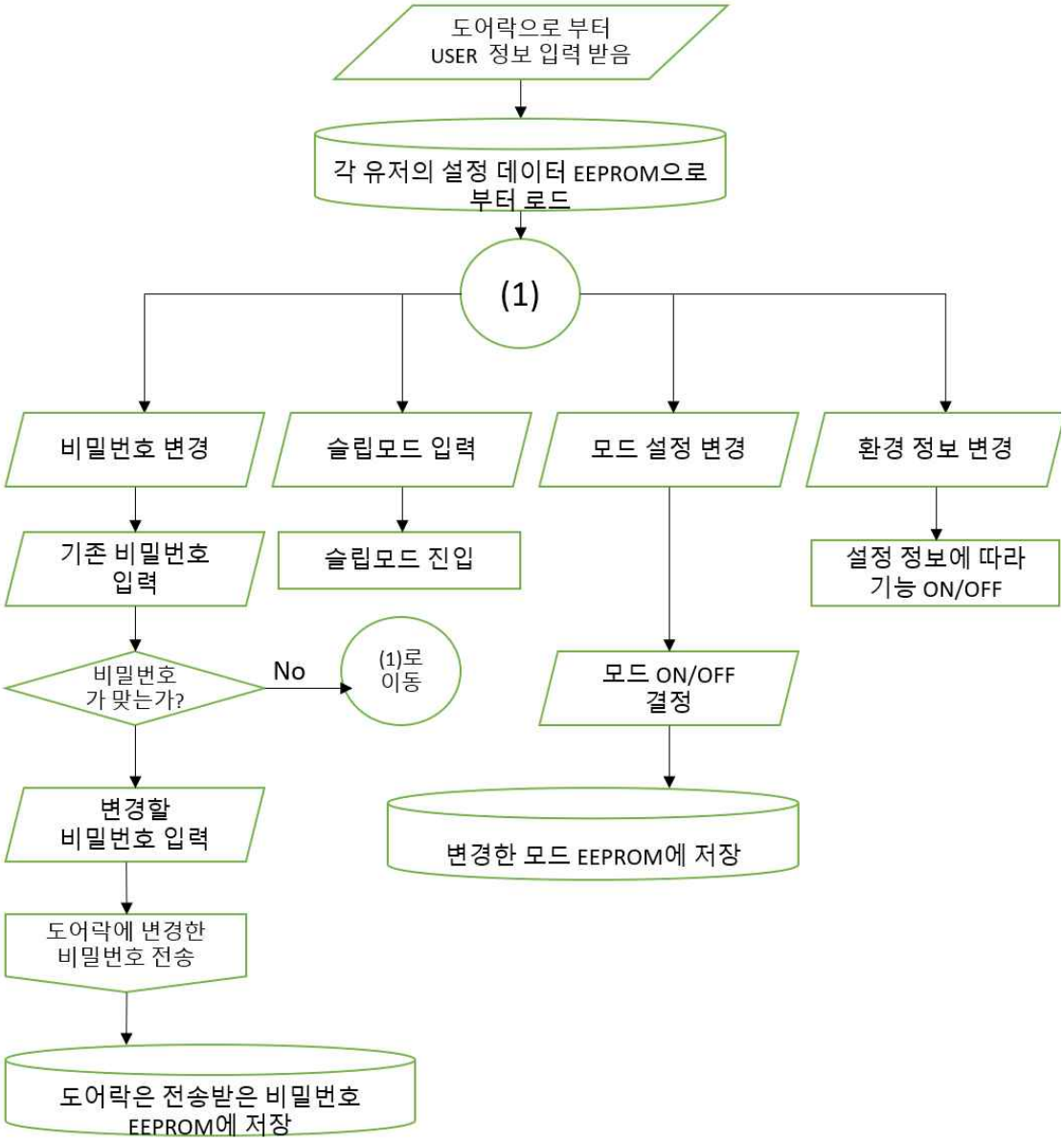

U1
ATMEGA128
MCU MODULE
VCC
/PEN
XTAL2
XTAL1
/RESET
AVCC
AGND
AREF
PF0(ADC0)
PF1(ADC1)
PF2(ADC2)
PF3(ADC3)
PF4(ADC4-TCK)
PF5(ADC5-TCK)
PF6(ADC6-TDO)
PF7(ADC7-TDI)
VRx(ONE006)
VRy(ONE006)
AO(SZH-SSBH-011)
signal(SZH-EK057)
PB0(/SS)
PB1(SCK)
PB2(MOSI)
PB3(MISO)
PB4(OC0)
PB5(OC1A)
PB6(OC1B)
PB7(OC2-OC1C)
IN1(SZH-BLBG-009)
IN2(SZH-BLBG-009)
IN3(SZH-BLBG-009)
IN4(SZH-BLBG-009)
IN5(SZH-BLBG-009)
IN6(SZH-BLBG-009)
IN7(SZH-BLBG-009)
IN8(SZH-BLBG-009)
PC0(A8)
PC1(A9)
PC2(A10)
PC3(A11)
PC4(A12)
PC5(A13)
PC6(A14)
PC7(A15)
DO(MQ-2)
signal(SUB_WINDOW)
DATA(DHT-11)
PE0(PDI-RXD0)
PE1(PDO-TXD0)
PE2(XCK0-AIN0)
PE3(OC3A-AIN1)
PE4(OC3B-INT4)
PE5(OC3C-INT5)
PE6(T3-INT6)
PE7(ICP3-INT7)
PG3(TOSC2)
PG4(TOSC1)
PD0(SCL-INT0)
PD1(SDA-INT1)
PD2(RXD1-INT2)
PD3(TXD1-INT3)
PD4(ICP1)
PD5(XCK1)
PD6(T1)
PD7(T2)
SCL(SRF02)
SDA(SRF02)
SW(ONE006)
BZ1
Buzzer
R6
100
R7
390
R8
47
R9
10
J14
VSS
VDO
VO
RS
RW
E
D0
D1
D2
D3
D4
D5
D6
D7
LED+
LED-
CON16
LCD_RS
LCD_RW
LCD_E
LCD_DATA0
LCD_DATA1
LCD_DATA2
LCD_DATA3
LCD_DATA4
LCD_DATA5
LCD_DATA6
LCD_DAT17

(c) 동작 시나리오

-도어락 FlowChart



- 제어부 FlowChart



도어락과 제어부의 경우 초기 비밀번호는 “0000”이며 비밀번호의 변경은 제어부에서만 가능하다. 도어락의 경우 비밀번호를 입력 후 “*" 버튼과 "#" 버튼을 눌렀을 때의 두 가지 기능이 있는데, 올바른 비밀번호 입력 후 ”*“ 버튼을 누르게 되면 문이 열리고 제어부에 Guest 입장을 알린다. "#" 버튼을 누르게 되면 RFID

카드 정보를 관리할 수 있는데, 버튼의 입력에 따라 USER1, USER2의 카드 정보를 등록할 수 있고, RFID 카드 정보들을 초기화 할 수 있다. USER1, USER2의 RFID 카드 등록이 완료된 상태이면 초기 화면에서 비밀번호 입력 없이 카드 태그만으로 문을 열 수 있으며, 각각의 카드로 문을 열면 제어부에 USER1, USER2의 입장을 알린다.

제어부의 경우 도어락에서 유저 정보를 수신 받으면 각 유저가 마지막에 설정했던 모드를 EEPROM에서 불러와서 실행한다. 제어부의 초기화면은 아래와 같은 5개의 선택지가 있으며, 조이스틱을 상하좌우로 움직이거나 누름으로써 인터페이스를 조작할 수 있게 하였다.

1. SETTIGNS
2. CHANGE PWD
3. ENV SETTING
4. DOOR OPEN
5. SLEEP MODE

1. SETTIGNS의 경우 진입하게 되면 총 8개의 센서나 엑추에이터들을 끄고 킬 수 있으며, 릴레이 모듈을 활용하여 사용하지 않는 센서나 엑추에이터들은 전원 자체가 공급되지 않게 하여 절전도 가능하게 하였다. 여기서 끄고 킬 수 있는 센서는 온습도 센서, 조도 센서, 초음파 거리 센서, 유해가스 검출 센서, 물 수위 센서가 있으며, 엑추에이터는 에어컨, 멀티탭, 가습기가 있다.

-온습도 센서를 키게 되면 사용자가 설정한 온도, 습도에 따라 가습기와 에어컨이 자동으로 ON/OFF가 되는 기능을 한다.

-조도 센서를 키게 되면 사용자가 설정한 밝기 정도에 따라 어두울 때는 창문이 열리고, 밝을 때는 창문이 닫히게 하였다.

-초음파 거리 센서를 키게 되면 우체통에 편지가 들어올 시 측정되는 거리가 달라지는 것을 이용하여서 우편물이 왔을 시에 사용자에게 알림이 가도록 설정하였다.

-유해가스 검출 센서를 키게 되면 유해가스가 검출되었을 시 창문이 열리고 닫힘을 반복함으로써 환기를 시키고 사용자에게 유해가스 검출 알림을 주게 하였다.

-물 수위 센서의 경우 침수가 되면 측정값이 올라가는 것을 이용하여 침수가 감지가 되면 1채널 릴레이 모듈을 활용하여서 집안의 전기 기구들의 전원 공급을 모두 차단되게 하였다.

2. CHANGE PWD의 경우 올바른 비밀번호를 입력 후 새로운 비밀번호를 입력하면 변경된 비밀번호가 EEPROM에 저장되며, 도어락에 변경된 비밀번호를 전송하고, 비밀번호를 수신 받은 도어락은 그 비밀번호를 EEPROM에 저장 후 로드하여 비밀번호를 사용자가 변경할 수 있게 하였다.

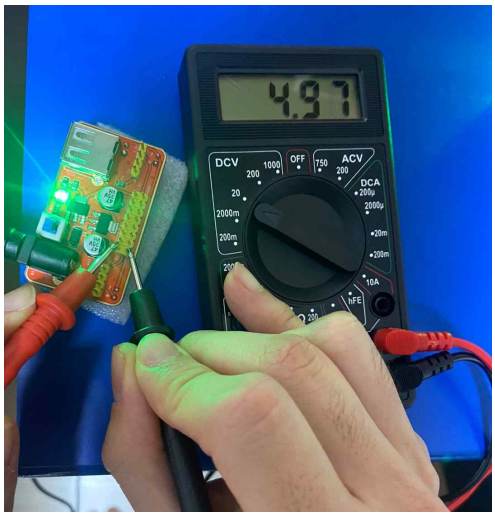
3. ENV SETTING의 경우 사용자가 1.SETTINGS에서 온습도 센서, 조도 센서, 물 수위 센서를 ON 하였을 때 설정 값에 따라 엑추에이터들의 작동이 달리하게 하였다. 사용자는 이 항목에 들어가게 되면 현재 온습도를 볼 수 있으며 자신이 원하는 온습도를 설정하면 이에 따라 에어컨과 가습기가 자동으로 켜지고 꺼진다. 조도 센서의 경우 원하는 밝기 레벨을 입력하면 측정되는 레벨에 따라 창문이 자동으로 열고 닫히게 하였다. 물 수위 센서의 경우 조도센서와 마찬가지로 사용자가 레벨을 선택하면 그 레벨까지 물 수위가 측정되었을 때 작동되게 하였다.
4.DOOR OPEN의 경우 선택하게 되면 도어락에 정보를 송신하여 문이 열리게 하였다.
5. SLEEP MODE의 경우 선택하게 되면 제어부 Atmega128이 슬립 모드에 진입하게 하였다.

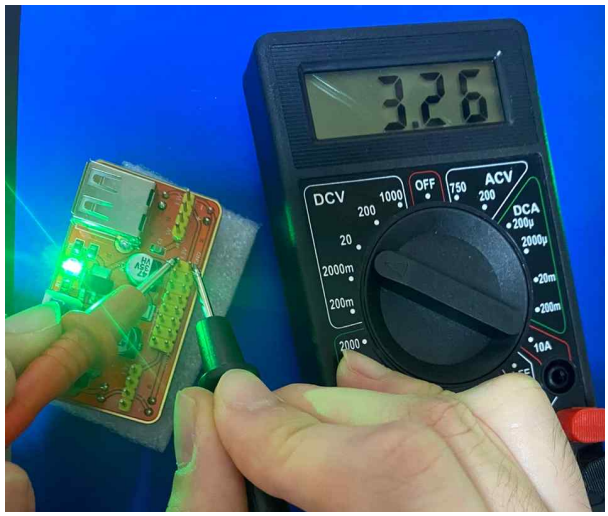
3. 요소별 개발 내용 및 검증

(a) 각 요소별 회로의 설계/구현 결과

각 요소별 회로의 경우 2. 개발 방법 개요에서 회로도를 첨부하였으므로 회로도는 여기에 첨부하지 않겠다.

사용하는 RFID 모듈을 제외하고는 5V 전압을 요구하였으며, RFID 모듈은 3.3V를 요구하였고, 사용한 전원공급기에서는 5V와 3.3V의 출력을 모두 지원하였으며 멀티미터로 측정한 결과 오차 범위 내로 5V와 3.3V의 출력을 제대로 다하고 있음을 확인하였다.





또한 각 센서와 엑추에이터의 경우 LCD출력을 통하여 검증하였고, 센서와 모듈이 많은 관계로 각 센서 및 엑추에이터의 동작 확인 코드를 개별 첨부하겠다.

(b) 각 요소의 동작을 위한 프로그램 설계

본 텀프로젝트에서는 총 2개의 Atmega128을 이용하여 도어락, 제어부를 구현하였기에 도어락, 제어부에 나누어 코드 차례로 기술하겠다. 또한 분량의 문제상 각 기능별 코드를 개별 설명하겠다.

-Atmega128 도어락

(EEPROM)

```
void EEPROM_Write(unsigned int EE_Addr, unsigned char EE_Data)
{
while(EECR&(1<<EEWE));
EEAR = EE_Addr
EEDR = EE_Data// EEDR <- EE_Data
cli();// Global Interrupt Disable
EECR |= (1 << EEMWE);// SET EEMWE
EECR |= (1 << EEWE); // SET EEWE
sei(); // Global Interrupt Enable
}

unsigned char EEPROM_Read(unsigned int EE_addr)
{
while(EECR&(1<<EEWE));
EEAR = EE_addr
EECR |= (1 <<EERE);// SET EERE
return EEDR // Return Read Value
}
```

먼저 EEPROM 관련 코드의 경우 수업 시간에 실습으로 진행하였던 코드를 활용하였고, EEPROM_Write(EEPROM 주소, EEPROM에 기록할 데이터)를 활용하여 EEPROM에 데이터를 쓸 수 있고, EEPROM_Read(EEPROM 주소)를 활용하여 EEPROM에 저장된 데이터를 읽을 수 있게 하였다.

(타이머/카운터)

```
void init_timer2(void) // 오버플로우발생시0.01초씩증가, 타이머카운터2
{
   TCCR2 = 0;
   TCCR2 |=(1<<WGM21)|(1<<WGM20)|(1<<CS22)|(1<<CS20);// fast PWM 모드 사용, 자동출력x 오버플로우인터럽트활성화
   TIMSK = 0x40;
   TCNT2 = 112;
```

```
}

ISR(TIMER2_OVF_vect)
{
   second_0p01++;
   TCNT2 = 112;
}

void init_timer1(void) // 타이머카운터1, Fast PWM 16비트타이머사용, 4608번세면주기20ms PB5 자동출력
{
TCCR1A= 0x82; // PB5(OC1A)핀자동출력, 컴패어매치때클리어, TOP에도달하였을때셋
TCCR1B=0x1B; // 16비트Fast PWM 모드사용, 분주비64 ICR1  = 4607일때파형의주기20ms
OCR1A=0;
ICR1=4607;
}
```

타이머/카운터의 경우 총 2개를 이용하였는데 타이머/카운터 1은 Fast PWM 16비트 타이머를 사용하였고, OC1A핀 자동출력을 통해 Servo Motor을 제어하여 도어락이 문의 열고 닫힘을 제어할 수 있게 하였으며 클럭 소스를 64분주하여 4608번 세면 20ms의 주기가 되게 하였고, 컴페어 매치 때 Clear, 오버플로우때 Set이 되게 하여 OCR1A의 값 설정에 따라 Duty ratio를 조정할 수 있게 하였다.

타이머/카운터 2은 Fast Pwm 모드를 사용하고 TIMSK에서 타이머/카운터2의 오버플로우 인터럽트를 활성화 시켜주어서 TCNT2값이 112부터 세서 오버플로우가 되면 10ms가 되게 하였다. 이는 이후에 인체감지센서를 활용하여서 사람이 10초동안 감지가 되지 않을 시 자동적으로 슬립모드에 들어가는 기능으로 활용하였다.

(서보 모터 제어)

```
void door_open(void)
{
OCR1A = 550; //  20% duty
}

void door_close(void)
{
OCR1A = 230; // 10% duty
_delay_ms(800);
OCR1A = 0; // 다시서보모터가작동하면안되므로duty 0}
```

door_open과 door_close의 경우 20ms의 주기에서 Duty ratio를 제어하여 회전 각도를 제어할 수 있었기에, OCR1A값을 변경하여 Servo Motor을 제어하였다.

(슬립 모드)

```
ISR(INT1_vect)
{
   second_0p01 = 0; // 사람이감지되면시간초기화
   TCCR2 |=(1<<WGM21)|(1<<WGM20)|(1<<CS22)|(1<<CS20);
}

void sleepmode(void)
{
   if(second_0p01 > 1000)
   {  //10초가넘으면슬립모드활성화
TCCR2 = 0;
       lcd_clear();
       _delay_ms(10);
       lcd_data(0,0);   lcd_read('S');   lcd_read('L');   lcd_read('E');   lcd_read('E');
lcd_read('P'); lcd_read(' ');
       lcd_read('M'); lcd_read('O'); lcd_read('D'); lcd_read('E');
       sleep_cpu();
   }
}
```

슬립 모드의 경우 인체감지센서를 PORTD 1번에 연결하였는데, 인체감지센서는 LOW신호를 유지하다가 사람이 감지되면 HIGH신호를 일정 시간동안 유지하고 다시 LOW신호로 돌아가기에 외부 인터럽트가 감지되면 타이머/카운터 2에서 증가시키던 시간 변수를 초기화하게 하였다. 만약 사람이 감지되지 않으면 시간 변수가 초기화 되지 않으므로 슬립 모드에 들어가게 설정하였고, <avr/sleep.h>를 이용하여 슬립 모드에 들어가게 하였다.

(Uart 설정 부분)

```
int parsingPacket(char *recBuff, int length) // 들어온패킷을처리하는함수 return -1, -2,-3 이면각각의종류의상태이상발생한것임
{
   if(recBuff[0] != 0xff) return -1; // 패킷검사1. 만약시작데이터가정해진데이터(0xFF)가아닌경우종료.
   if(recBuff[1] != 0x02) return -2; // 패킷검사2. 장치의이름이1이아닌경우종료, 0x01은도어락, 0x02는제어부아트메가
   if(recBuff[length -1] != chksum(recBuff,length-1)) return -3;
```

```
    // 패킷검사3. 패킷의체크섬을확인한뒤일치하지않은경우종료.
    // 수신된체크섬데이터는recBuff[length -1]
    // 체크섬계산값은체크섬데이터앞까지더하는chksum(recBuff,length-1)
    if( recBuff[3] == 0x06)
    { // 데이터가쓰기인경우
if(recBuff[2] == 0x02)
{
pw1 = RecBuff[5];
pw2 = RecBuff[6];
pw3 = RecBuff[7];
pw4 = RecBuff[8];
EEPROM_Write(0x10, pw1);
EEPROM_Write(0x11, pw2);
EEPROM_Write(0x12, pw3);
EEPROM_Write(0x13, pw4);
}
if(recBuff[2] == 0x03)
{
door_open();
lcd_clear();
_delay_ms(10);
lcd_data(0,0); lcd_read('D'); lcd_read('O'); lcd_read('O'); lcd_read('R'); lcd_read(' ');
lcd_read('O'); lcd_read('P'); lcd_read('E'); lcd_read('N');
_delay_ms(2000);
door_close();
}

    }
    return 0;
}

void USART0_process(void)
{
    if(RecFlg == 1)
    {
        // 임시저장후또다시패킷수신을위한버퍼및관련변수초기화
        memcpy(lRecBuff, RecBuff, RecBuff_estLength);
        lRecBuffLength = RecBuff_estLength
        RecBuff_estLength = REC_BUFF_MAX_LENGTH
        RecBuffindex = 0;
```

```
        RecFlg = 0;
        // 수신된패킷을파싱하는함수
        usart0_state = parsingPacket(lRecBuff,lRecBuffLength);
        memset(RecBuff, 0, REC_BUFF_MAX_LENGTH);

        if (usart0_state < 0)
        {
            // 상태이상알림
        }
    }
}

void send_Packet(unsigned char type, unsigned char rw, unsigned char length, unsigned char *sendData) // 데이터보내는함수
{
    unsigned char resBUFF[REC_BUFF_MAX_LENGTH]; // REC_BUFF_MAX_LENGTH는데이터의최대길이, 코딩하고마지막에수정
    unsigned char resBUFFLength = 0;
    int i = 0;
    resBUFF[resBUFFLength++] = 0xff; // 시작바이트0xff
    resBUFF[resBUFFLength++] = 0x01; // 장치이름0x01은도어락0x02는제어부, 이장치는제어부임으로0x02 전송
    resBUFF[resBUFFLength++] = type // 데이터의종류, 0x02는패스워드, 0x01은유저데이터0x03은문열림데이터
    resBUFF[resBUFFLength++] = rw    // read, write 종류
    resBUFF[resBUFFLength++] = length // 데이터들의길이절대전체의길이가아님
    for(i = 0; i<length i++)
    {
        resBUFF[resBUFFLength++] = sendData[i]; // 데이터를보냄 send_Pachet(..... ,보낼데이터)
    }
    resBUFF[resBUFFLength] = chksum(resBUFF, resBUFFLength); // 체크섬계산후체크섬전송
    resBUFFLength++;
    sendBuff_USART0(resBUFF, resBUFFLength); // 데이터전송
}
```

Uart 설정 부분의 경우 분량의 관련 레지스터 및 인터럽트 부분을 제외하고 설계한 데이터 패킷 위주로 설명하겠다. 관련 내용은 강의 자료의 “10. 직렬 통신 포트의 동작_실습_배포.pptx"를 참고하였다.

설계한 데이터 패킷은 아래와 같은 구조를 지니고 있다.

Byte 0	Byte 1	Byte 2	Byte3	Byte 4	Byte 5~n	Byte n+1
0xFF	장치 이름	데이터 종류	읽기/쓰기	데이터 길이	데이터	체크섬

장치 이름의 경우 0x01를 도어락, 0x02를 제어부로 설정하였다.

데이터의 종류는 0x01를 유저 데이터(도어락이 제어부로 보내는 유저 입장 데이터), 0x02를 패스워드(제어부가 PW를 변경한 후 도어락으로 보내는 PW데이터), 0x03을 문 열림 데이터(제어부가 도어락에 문 열기를 요청하는 데이터)로 지정하였다.

읽기 쓰기의 경우 강의 자료와 같이 읽기 0x03, 쓰기 0x06으로 설정하였다.

받는 데이터를 처리하는 int parsingPacket(char *recBuff, int length) 경우 시작비트, 장치 이름, 체크섬이 모두 일치할 경우에 데이터를 처리하는데, PW데이터를 처리하는 경우 PW를 수신 받은 PW로 변경하고, EEPROM에 수신 받은 PW를 저장한다. 문 열림 데이터를 수신 받은 경우에는 문을 열고, LCD에 “DOOR OPEN"을 출력하고 다시 닫는다.

(4x4 Keypad 설정 부분)

```
void num_pad_test(void)// PB4~PB7 => C1~C4 PB0~PD3 => R4~R1
{
   PORTF = ~0x01;
   _delay_ms(DEBOUNCING_DELAY);
   if(!(PINF&0x10)) input_signal = star_input
   if(!(PINF&0x20))
   {
      input_number = 7;//
      input_number_count++;
   }
   if(!(PINF&0x40))
   {
      input_number = 4;//
      input_number_count++;
   }
   if(!(PINF&0x80))
   {
      input_number = 1;//
      input_number_count++;
   }

   PORTF = ~0x02;
   _delay_ms(DEBOUNCING_DELAY);
```

```
if(!(PINF&0x10))
 {
    input_number = 0;//
    input_number_count++;
 }
if(!(PINF&0x20))
{
   input_number = 8;//
   input_number_count++;
}
if(!(PINF&0x40))
{
   input_number = 5;
   input_number_count++;
}
if(!(PINF&0x80))
{
   input_number = 2;//
   input_number_count++;
}


PORTF = ~0x04;
_delay_ms(DEBOUNCING_DELAY);
if(!(PINF&0x10)) input_signal = pound_input // #버튼을눌렀을때
if(!(PINF&0x20))
{
   input_number = 9;//
   input_number_count++;
}
if(!(PINF&0x40))
{
   input_number = 6;//
   input_number_count++;
}
if(!(PINF&0x80))
{
   input_number = 3;
   input_number_count++;
}
```

```
    PORTF = ~0x08;
    _delay_ms(DEBOUNCING_DELAY);
    if(!(PINF&0x10)) input_signal = D_input// D버튼을눌렀을때
    if(!(PINF&0x20)) input_signal = C_input // C버튼눌렀을때
    if(!(PINF&0x40)) input_signal = B_input //B버튼눌렀을때
    if(!(PINF&0x80)) input_signal = A_input // A버튼눌렀을때
}

void input_pw(void)
{
    if (input_number_count == 1) input_pw1 = input_number
    if (input_number_count == 2) input_pw2 = input_number
    if (input_number_count == 3) input_pw3 = input_number
    if (input_number_count == 4) input_pw4 = input_number

}
```

4x4 Keypad의 경우 PORTF의 0~3번 핀을 출력, 4~7번 핀을 입력으로 설정하여서, 0~3번 핀에 50ms의 간격으로 HIGH신호를 순서대로 출력하게 하여서 PIND의 4~7번 핀 중에서 어떤 핀으로 HIGH신호가 들어오는지 확인하여서 1~9의 숫자와 A~D, *, #의 문자 중 사용자가 어떤 문자를 입력하였는지 파악할 수 있게 했다. 또한 숫자 핀을 눌렀을 시에는 input_number의 값이 1씩 증가하게 하여서 사용자가 입력한 숫자가 몇 번째 숫자인지 파악할 수 있게 하였다. 이는 사용자가 비밀번호를 입력할 때 사용자가 입력한 비밀번호를 LCD에 출력할 때 활용하였다.

(RFID 설정 부분)

RFID 코드의 경우 분량 문제로 전체 코드를 설명하지 않고 부분적으로 설명하겠다. 교수님께서 제공하신 코드에서는 RFID_loop 함수에서 RFID 카드를 태그하면 2를 return하는데, 이를 활용하여서 RFID_new_reading 함수를 만들었다.

이 함수에서는 res_SPI = 2일 때 card_byte_USER[]의 배열에 읽힌 카드 정보를 넣고 이를 활용하여서 RFID카드 정보를 등록할 수 있게 하였다.

```
if(lcd_state == 3)
{    // 사용자가USER1 입력모드선택하여card_USER1_USER2 [0~3]에rfid 카드정보입력
card_USER1_USER2[0]  =  card_byte_USER[0];      card_USER1_USER2[1]  =  card_byte_USER[1];
card_USER1_USER2[2]  =  card_byte_USER[2];      card_USER1_USER2[3]  =  card_byte_USER[3];
```

```
// EEPROM 0x01~0x04에rfid 카드정보저장
EEPROM_Write(0x01,card_USER1_USER2[0]);
EEPROM_Write(0x02,card_USER1_USER2[1]);
EEPROM_Write(0x03,card_USER1_USER2[2]);
EEPROM_Write(0x04,card_USER1_USER2[3]);
//card_check_buffer에EEPROM에저장한카드정보를불러옴
card_check_buffer[0] = EEPROM_Read(0x01); card_check_buffer[1] = EEPROM_Read(0x02);
card_check_buffer[2] = EEPROM_Read(0x03); card_check_buffer[3] = EEPROM_Read(0x04);
// EEPROM에제대로저장되었는지확인되면
if(card_check_buffer[0] == card_USER1_USER2[0] && card_check_buffer[1] == card_USER1_USER2[1]&&
card_check_buffer[2] == card_USER1_USER2[2] && card_check_buffer[3] == card_USER1_USER2[3])
{   //카드등록이성공하였다는메세지출력후초기화면으로돌아감이하아래lcd_state = 4의경우도동일하게작동함.
lcd_clear();
_delay_ms(10);
lcd_data(0,0); lcd_read('R'); lcd_read('F'); lcd_read('I'); lcd_read('D'); lcd_read(' ');
lcd_read('I'); lcd_read('N'); lcd_read('F'); lcd_read('O'); lcd_read('R'); lcd_read('M');
lcd_read('A'); lcd_read('T'); lcd_read('I'); lcd_read('O'); lcd_read('N');
lcd_data(1,0); lcd_read('S'); lcd_read('A'); lcd_read('V'); lcd_read('E'); lcd_read('D');
lcd_read(' '); lcd_read('S'); lcd_read('U'); lcd_read('C'); lcd_read('C'); lcd_read('E');
lcd_read('E'); lcd_read('D');
_delay_ms(2000);
lcd_state = 0; input_number_count = 0; input_signal = 0;
}

}
```

lcd_state = 3 이 의미하는 것은 사용자가 올바른 비밀번호를 입력한 후 "#" 버튼을 누른 뒤 USER1의 RFID 카드를 등록하는 상황에 온 것을 의미하는데, 이때 card_USER1_USER2[7]배열에 0~3에 태그 된 카드 정보를 저장하고 이를 EEPROM에 저장한다.(이 때 card_USER1_USER2의 0~3은 USER1의 카드 정보를 저장하고, 4~7의 USER2의 카드 정보를 저장한다.) 그 후 card_check_buffer[3]에 EEPROM에 저장하였던 데이터를 다시 불러와 card_USER1_USER2 0~3 와 card_check_buffer 0~3을 비교하여 둘이 같음을 확인하면 EEPROM에 사용자 카드 정보가 제대로 저장된 것이므로 사용자에게 LCD로 카드 정보가 제대로 등록이 되었다고 알리고, 초기 화면으로 돌리고 사용자가 입력하여 변하였던 변수들을 초기화 시켜준다.

(인터페이스 설정 부분)

menu 함수의 경우 위의 함수들을 사용하여 사용자가 입력하는 변수에 따라 LCD에 디스플레이 되는 문자와 화면들이 변경되게 만들었다. 분량 문제로 전체 코드를 첨부하기는 힘들어 간단하게 기술하여 설명하면, lcd_state = 0 의 경우에는 초기 화면으로 "TAG RFID OR PUT PWD : ____"를 LCD에 출력하는데, 위의 4x4 Keypad에서 사용자가 숫자를 누르면 input_number의 값이 1씩 증가한다고 설명하였는데 이를 활용하여서 사용자가 숫자를 순서대로 누르면 LCD의 좌표 값이 변하게 만들어 입력한 4자리의 비밀번호가 순서대로 출력되게 설계하였다.

4자리의 비밀번호를 다 누르면 사용자가 “*” 버튼이나 “#"버튼 중 어떤 버튼을 누르는지에 따라 각각 lcd_state = 1, lcd_state = 2로 lcd_state의 값이 변하는데 각 조건식에 들어가면 비밀번호 일치, 불일치를 판단한 뒤에 문을 열거나 RFID 정보를 등록하거나, 초기화 시키는 등의 작동을 하도록 코드를 작성하였다.

(메인문)

```
int main(void)
{
   /*EEPROM_Write(0x10, 0);
   EEPROM_Write(0x11, 0);
   EEPROM_Write(0x12, 0);
   EEPROM_Write(0x13,   0);*/   //비밀번호초기화코드각주지우고사용하면EEPROM에
0000저장

   pw1 = EEPROM_Read(0x10);
   pw2 = EEPROM_Read(0x11);
   pw3 = EEPROM_Read(0x12);
   pw4 = EEPROM_Read(0x13);
   //  도어락이만약꺼져도eeprom에저장된비밀번호를불러옴,  비밀번호변경데이터가수신
되면
   // eeprom에각비밀번호가저장됨.

   // 기존사용자의RFID  정보를EEPROM으로부터읽어옴USER1는0x01~0x04,  USER2
는0x05~0x08에정보저장됨
   card_USER1_USER2[0]  =  EEPROM_Read(0x01);  card_USER1_USER2[1]  =
EEPROM_Read(0x02);      card_USER1_USER2[2]      =      EEPROM_Read(0x03);
card_USER1_USER2[3] = EEPROM_Read(0x04);
   card_USER1_USER2[4]  =  EEPROM_Read(0x05);  card_USER1_USER2[5]  =
EEPROM_Read(0x06);      card_USER1_USER2[6]      =      EEPROM_Read(0x07);
card_USER1_USER2[7] = EEPROM_Read(0x08);
   // EEPROM에저장되었던카드2개의정보를읽어태그하는카드와비교할수있게함
```

```
    Set_pin_set_atmega128();
    USART0_init();
    init_timer1();
    init_timer2();

    RFID_settings();
    lcd_pic();
    lcd_mv();
    lcd_func();
    rfid_ic_ver = PCD_DumpVersionToSerial();
    DDRB |= 0x20; // RFID에서DDRB 레지스터제어하기때문에자동출력PORTB5번핀출
력활성화
     lcd_clear();
     _delay_ms(10);
    while (1)
    {
        sleepmode();
        menu();
        num_pad_test();
        input_pw();
        USART0_process();
res_SPI = RFID_loop();
if(input_number_count > 4 ) input_number_count = 4;

    }
}
```

메인문의 경우 무한 루프 이전에 EEPROM에서 PW정보와 RFID 정보를 불러오는 것으로 혹시 모를 MCU의 종료 후 재부팅에도 비밀번호와 RFID 정보를 안전하게 불러올 수 있게 하였고, Set_pin_set_atmega128 함수 안에 각종 레지스터 설정 정보를 넣어서 편하게 관리하였으며, lcd 세팅을 해주는 함수를 넣었고, DDRB |= 0x20을 하였는데 이는 제공해주신 RFID 코드에서 DDRB설정을 하는 코드들이 있었는데, 이를 바꾸지 않고 간편하게 0x20과 OR연산 후 DDRB의 설정 값을 그 값으로 바꾸어주는 방법을 선택하였다.

무한 루프 안을 보면 if(input_number_count > 4) input_number_count = 4; 부분이 있는데 이는 테스트 중 사용자가 초기에 비밀번호를 4개 이상 눌렀을 때 오류가 나는 부분이 있었는데 입력 가능한 최대 비밀번호는 4자리지만 범위를 설정해주지 않았기 때문에 오류가 났으므로 아래 조건문을 적어 오류를 방지하였다.

-Atmega128 제어부
(타이머/카운터)

```
void init_timer0(void)    //타이머기본설정
{
TCCR0 = 0x0f;//(1<<WGM01)|(1<<CS00)|(1<<CS01)|(1<<CS02);
TCNT0 = 0x00;
OCR0 = 14;
TIMSK = 0x14;
TIMSK |= (1<<OCIE0);
}

ISR(TIMER0_COMP_vect)
{
ti_Cnt_1ms ++;
LCD_DelCnt_1ms ++;
}

void init_timer1(void) // 타이머카운터1, Fast PWM 16비트타이머사용, 4608번세면주기20ms
{
TCCR1A = 0x02; // 컴패어매치때클리어, TOP에도달하였을때셋
TCCR1B = 0x1B; // 16비트Fast PWM 모드사용, 분주비64 ICR1  = 4607일때파형의주기20ms
OCR1A = 0;
ICR1 = 4607;
}

ISR(TIMER1_COMPA_vect)
{
BIT_CLEAR(PORTC, 4);
}

ISR(TIMER1_OVF_vect)
{
BIT_SET(PORTC, 4);
}
```

타이머/카운터는 0, 1 총 두 개를 사용하였는데, 타이머/카운터 0의 경우 조원이 작성하였기에 따로 기술하지 않고, 타이머/카운터 1에 대해 설명하면, 도어락의 타이머/카운터 1과 자동출력을 비활성화 한 것 빼고 동일하다. 자동출력을 비활성한 이유는 기획 단계에서 OC1A핀에 이미 릴레이 모듈을 쓰기로 예정해 놓았기

때문에 자동출력을 비활성화 시켰다. 자동출력을 사용할 수 없기 때문에 컴패어 매치 인터럽트와 오버플로우 인터럽트를 활성화 시켜서 컴패어 매치 때 PORTC의 4번핀을 클리어, 셋을 시켜서 PWM 신호를 PORTC로 출력할 수 있게끔 하였다.
그 다음 아래의 서보모터 부분은 위의 도어락과 동일하지만, 창문을 여닫는 코드가 있는데 이 코드만 설명하겠다.

(서보모터 설정 부분)

```
void window_open_and_close(void)
{
OCR1A = 550;
_delay_ms(600);
OCR1A = 230;
_delay_ms(600);
}
```

window_open_and_close 함수는 OCR1A값을 600ms 딜레이 사이로 550, 230으로 전환하여서 창문이 계속 여닫히게 하였다. I2C 초음파 관련 함수는 조원이 설계하였지만, 관련 테스트 중 초음파 값이 알 수 없는 이유로 튀는 현상이 발생하여서 이를 해결하기 위한 코드를 작성하였고 아래에 기술하겠다.

(초음파 거리센서 설정 부분)

```
void SRF02_running(void)
{
startRanging(Sonar_Addr);
if(ti_Cnt_1ms > 66){  //66ms이상일때동작// 충분한시간이있어야함
res = getRange(Sonar_Addr, &Sonar_range);
if(LCD_DelCnt_1ms > 66)
{ // 거리를30번더하고30으로나누어평균값구함(1번측정하면값이오류가나는경우가많음)
Sonar_range_buffer_plus += Sonar_range
LCD_DelCnt_1ms = 0;
k++;
}
startRanging(Sonar_Addr);
ti_Cnt_1ms = 0;
readCnt = (readCnt + 1)%10;
}
   if(k > 29) // 30번카운팅후
   {
Sonar_range_avr = Sonar_range_buffer_plus / 30; // Sonar_range_buffer_plus 을30으로나누어평균값구함
```

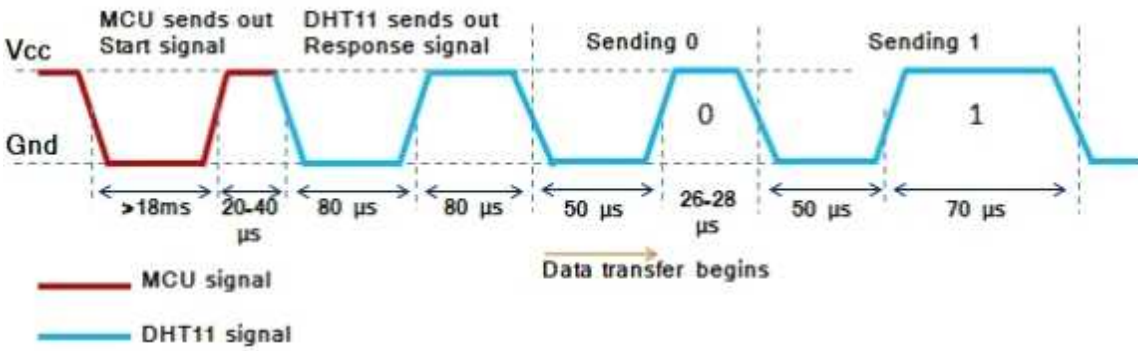
```
// 이후관련변수초기화
Sonar_range_buffer_plus = 0;
k = 0;
    }
}
```

Sonar_range에 현재 측정되고 있는 거리 값인데, 가끔씩 튀는 값이 있어서 이를 활용하기가 힘든 부분이 있었다. 따라서 Sonar_range_buffer_plus 라는 int 변수에 Sonar_range 값을 30번 더하고, 30번이 카운팅 되면 30으로 나누어 평균값을 Sonar_range_avr 변수에 저장하고, 이후 이 과정을 반복하여 몇 번 튀는 값이 있어도 평균값에 따라 액추에이터를 구동하게 하여서 이를 해결하였다.

(DHT-11 설정 부분)

DHT-11의 경우 한 개의 선으로 통신을 하는 방식으로, 기존의 Uart, I2C, SPI 통신 방식이 아닌 센서만의 통신 방식을 사용하기에 이에 맞춰 코드를 작성하였다.

코드 설명에 앞서 DHT-11의 통신 방식을 간단하게 설명하면 다음과 같다.



DHT-11의 경우 Atmega128이 18ms의 LOW신호 뒤에 20~40us의 HIGH 신호를 DHT-11에 보내면, DHT-11은 80us의 LOW신호와 80us의 HIGH신호를 보낸 뒤 아래 표와 같은 40bit의 데이터를 순차적으로 전송한다.

parameter	습 도		온 도		체크섬
총 40bit	8bit integral Data	8bit Decimal Data	8bit integral Data	8bit Decimal Data	Check Sum

DHT-11에서 보내는 데이터에서 0과 1을 구분하는 방법은 보내는 신호에서 80us LOW신호 뒤에 HIGH신호가 26~28us면 '0'이고, 70us면 '1'임을 구분할 수 있다.

이에 따라 DHT-11에서 온습도 정보를 전달받는 코드는 다음과 같다.

```
void send_signal()
{
DDRC = 0x80;
PORTC = 0x80;
PORTC = 0x00;  // low
_delay_ms(20);  // wait 18ms
PORTC = 0x80;  // high
_delay_us(40);  // wait 20~40us
}

int response()
{
DDRC = 0x00;
if((PINC & 0x80) != 0)   // low가아니면
{
return 1;
}
_delay_us(80);  // until 80us
if((PINC & 0x80) == 0)  // high가아니면
{
return 1;
}
_delay_us(80);  // until 80us
return 0;
}

void send_data()
{
if(response() == 0)
{
DDRC = 0x00;
for(g = 0; g < 5; g++)   // RH, T, Parity bit
{
for(h=0; h < 8; h++)   // 8bit
{
while((PINC & 0x80) == 0)
{

}
_delay_us(30);
```

```
if((PINC & 0x80) == 0)     // 26 ~ 28us
{
data[g][h] = 0;
}
else    //  70us
{
data[g][h] = 1;
while((PINC & 0x80) == 128)  // while(PINC & (1<<7))
{

}
}
}
}
}
else
{

}
}

void  dht11_work(void)
{
send_signal();
response();
send_data();
RH_integral = data[0][0] * 128 + data[0][1] * 64 + data[0][2] * 32 +
data[0][3] * 16 + data[0][4] * 8 + data[0][5] * 4 + data[0][6] * 2 +
data[0][7];
RH_decimal = data[1][0] * 128 + data[1][1] * 64 + data[1][2] * 32 +
data[1][3] * 16 + data[1][4] * 8 + data[1][5] * 4 + data[1][6] * 2 +
data[1][7];
T_integral = data[2][0] * 128 + data[2][1] * 64 + data[2][2] * 32 +
data[2][3] * 16 + data[2][4] * 8 + data[2][5] * 4 + data[2][6] * 2 +
data[2][7];
T_decimal = data[3][0] * 128 + data[3][1] * 64 + data[3][2] * 32 +
data[3][3] * 16 + data[3][4] * 8 + data[3][5] * 4 + data[3][6] * 2 +
data[3][7];
parity = data[4][0] * 128 + data[4][1] * 64 + data[4][2] * 32 + data[4][3] *
16 + data[4][4] * 8 + data[4][5] * 4 + data[4][6] * 2 + data[4][7];
```

```
if((RH_integral + RH_decimal + T_integral + T_decimal) == parity)
{
H_10 = RH_integral / 10; H_01 = RH_integral % 10; H_0p1 = RH_decimal /10;
H_0p01 = RH_decimal % 10;
T_10 = T_integral / 10; T_01 = T_integral % 10; T_0p1 = T_decimal /10;
T_0p01 = T_decimal % 10;
}

}
```

send_signal 함수를 보면 DHT-11에 Start signal을 보내는 것을 확인할 수 있다. 20ms의 LOW신호 뒤 40us의 HIGH신호를 보내고 response 함수로 넘어가게 된다. response함수에서는 DDRC = 0x00으로 하여 입력으로 전환하고, LOW신호가 아니면 1을 리턴, 80us 뒤에 HIGH가 아니면 1을 리턴 한다. 1을 리턴하면 DHT-11이 정상적으로 작동을 하지 않았음을 알 수 있다.

그 다음 send_data함수를 보면 response함수에서 0을 리턴 하였을 때, 온습도 값을 받아온 뒤, 마지막으로 체크섬 값을 받아온다. 마지막으로 데이터를 다 더해서 체크섬과 일치하면 온습도의 정수 값과 소수 값을 각각 T_10, T_01, T_0p1, T_0p01/ H_10, H_01, H_0p1, H_0p01에 저장함으로 DHT-11과의 통신을 마친다.

(Uart 설정 부분)

Uart 설정 부분은 도어락 부분과 거의 유사하므로, 수신한 데이터를 처리하는 부분만 설명하겠다.

```
int parsingPacket(char *recBuff, int length) // 들어온패킷을처리하는함수 return -1, -2,-3 이면각각의종류의상태이상발생한것임
{
   if(recBuff[0] != 0xff) return -1; // 패킷검사1. 만약시작데이터가정해진데이터(0xFF)가아닌경우종료.
   if(recBuff[1] != 0x01) return -2; // 패킷검사2. 장치의이름이1이아닌경우종료, 0x01은도어락, 0x02는제어부아트메가
   if(recBuff[length -1] != chksum(recBuff,length-1)) return -3;
   // 패킷검사3. 패킷의체크섬을확인한뒤일치하지않은경우종료.
   // 수신된체크섬데이터는recBuff[length -1]
   // 체크섬계산값은체크섬데이터앞까지더하는chksum(recBuff,length-1)

      if(RecBuff[3] == 0x06)
      {
         USER_WHO = RecBuff[5];
         if(USER_WHO == 1) mode = EEPROM_Read(0x01);
         if(USER_WHO == 2) mode = EEPROM_Read(0x02);
```

```
            if(USER_WHO == 3) mode = EEPROM_Read(0x03);
        }


    return 0;
}
```

제어부의 경우, 데이터를 수신 받는 경우가 사용자가 도어락을 해제하였을 때 어떤 사용자가 들어왔는지에 대한 정보를 수신 받는 경우가 전부이다. 따라서 데이터가 쓰기일 때, USER_WHO에 수신된 데이터를 저장하고, 이를 조건문으로 구별하여 각 사용자의 mode값을 EEPROM에서 불러온다. 여기서 mode는 unsigned char인데, 각 비트별로 센서와 액추에이터의 ON/OFF를 구분한다. 각 비트가 의미하는 센서 및 액추에이터는 아래와 같으며, '0' 일 때는 OFF, '1' 일 때는 ON을 의미한다.

(7bit~4bit)

DHT-11	가습기	조도 센서	x

(3bit~0bit)

초음파 센서	수위 센서	유해가스 센서	에어컨

잠시 이에 대해 더 설명하면 8비트의 mode를 PORTB의 출력으로 바로 사용하여 릴레이 모듈을 제어함으로 절전 또한 가능하게 하였다. 또한 mode를 수신 받은 데이터를 처리하여 어떤 사용자가 집에 들어왔는지 확인하여 그 사용자가 설정하였었던 값을 가져올 수 있다.

(ADC 설정 부분)

```
void ADC_init(void)
{
    ADCSRA = 0x00;
    ADMUX = 0x40 | (0<<ADLAR) | (0 << MUX0);
    ADCSRA = (1<<ADEN)|(3<<ADPS0)|(0<<ADFR); //
}

unsigned int read_ADC4_data(void)
{
unsigned int adc_Data_ADC0 = 0;
ADMUX = 0x44;
ADCSRA |=(1<<ADSC);
while(!(ADCSRA & (1<<ADIF)));
adc_Data_ADC0 = ADCL
     adc_Data_ADC0 |= ADCH<<8;
```

```
return adc_Data_ADC0
}

unsigned int read_ADC1_data(void)
{
unsigned int adc_Data_ADC1 = 0;
ADMUX = 0x41;
ADCSRA |=(1<<ADSC);
while(!(ADCSRA & (1<<ADIF)));
adc_Data_ADC1 = ADCL
adc_Data_ADC1 |= ADCH<<8;

return adc_Data_ADC1
}

unsigned int read_ADC2_data(void)
{
unsigned int adc_Data_ADC2 = 0;
ADMUX = 0x42;
ADCSRA |=(1<<ADSC);
while(!(ADCSRA & (1<<ADIF)));
adc_Data_ADC2 = ADCL
adc_Data_ADC2 |= ADCH<<8;

return adc_Data_ADC2
}

unsigned int read_ADC3_data(void)
{
unsigned int adc_Data_ADC3 = 0;
ADMUX = 0x43;
ADCSRA |=(1<<ADSC);
while(!(ADCSRA & (1<<ADIF)));
adc_Data_ADC3 = ADCL
adc_Data_ADC3 |= ADCH<<8;

return adc_Data_ADC3
}

void adc_sensor(void)
```

```
{
// 물수위센서값변환시작
water_level = read_ADC4_data();
water_level_100 = water_level / 100;
// 조이스틱변환시작
joystick_x = read_ADC1_data();
joystick_y = read_ADC2_data();
    // 조도센서변환시작
Luminous = read_ADC3_data();
Luminous_100 = Luminous / 100;
}
```

이번 프로젝트에서는 물 수위 센서, 조도 센서, 조이스틱x축, y축의 총 4개의 아날로그 값을 ADC를 통해 변환하여야 했다. 하지만 ADC변환은 4개 다 동시에 이루어질 수 없었기 때문에 순차적으로 4번, 1번, 2번, 3번 채널의 데이터를 변환하였으며 프리러닝 비활성화, AVCC기준 전압을 활용하였다. joystick_x, joystick_y의 경우 RawData를 그대로 활용하였고, 물 수위 센서의 값과 조도 센서의 값은 100으로 나누어 0~10의 값을 활용하였다.

(조이스틱 설정 부분)

```
void joystick(void)
{
   / / 조 이 스 틱 이 오 른 쪽 으 로 움 직 일 때 의 동 작
------------------------------------------------------------
   if(BIT_CHECK(lcd_state_and_joystick,0) == 0)//조이스틱이중앙에있을때
   {
      if(joystick_x > 700)// 조이스틱이오른쪽으로움직였을때
      {
         BIT_SET(lcd_state_and_joystick,0); // 조이스틱이오른쪽으로움직인상태라는
것을알수있게1로셋
      }
   }
   if(BIT_CHECK(lcd_state_and_joystick,0) == 1)//조이스틱이오른쪽에있을때
   {
      if(joystick_x < 580)//조이스틱이중앙으로돌아왔을때
      {
          if(BIT_CHECK(pwd_insert_mode, 0) == 0)
          {
            x++;
          }
```

```
if(BIT_CHECK(env_insert_mode,0)==0)
{
x_lcd_state_5++;
}

          BIT_CLEAR(lcd_state_and_joystick,0);//조이스틱이중앙에있을때의상태로돌아
감
      }
   }
```

조이스틱의 코드는 조이스틱이 중앙에서 오른쪽으로 움직였을 때의 코드를 예시로 설명하겠다. 조이스틱을 오른쪽으로 움직이면 ADC 변환 값이 약 500~600에서 증가하였는데, 값이 증가하였을 때 어떠한 변수를 1씩 증가하게 하면 조이스틱을 오른쪽으로 기울였을 때 값이 엄청 빠른 속도로 증가하여 이를 활용할 수 없었다. 따라서 조이스틱을 오른쪽으로 기울였을 때 lcd_state_and_joystick의 0번 비트를 셋한 뒤 1로 셋 되어있을 때 다시 중앙으로 돌아오면 x값이 1 증가하게 만들어 오른쪽으로 기울였다가 다시 중앙으로 돌아오면 x좌표가 1 증가하게 활용하였다. 또한 코드 내에서 사용자가 어떤 메뉴 내에 위치해 있는지 확인하여 특수한 값들이 증가하게 하였다.

(인터페이스 설정 부분)

settings_page 함수의 경우 lcd_state = 1일 때 LCD에 ON/OFF를 간편하게 출력하기 위해 mode의 각 bit를 체크하여 출력하는 함수이다.

menu함수의 경우 사용자가 입력하는 값에 따라 인터페이스 화면의 변화를 주기 위해 만든 함수로, lcd_state = 0일 때는 메인 화면이고 lcd_state = 1일 때는 1. SETTINGS의 화면으로 진입하였을 때의 화면, lcd_state = 2, 3, 4 일 때는 비밀번호 변경 상태일 때의 화면, lcd_state = 5일 때는 ENV SETTING에 진입하였을 때의 화면, lcd_state = 6일 때는 슬립 모드에 진입하였을 때의 화면을 나타낸다.

또한 가스 검출 센서 혹은 메일 알람을 ON 해놓았을 때는 가스가 검출되거나 메일이 감지되면 그에 알맞은 화면을 LCD에 출력하도록 하였다.

사용자는 인터페이스 화면과 조이스틱 하나의 조작으로 각종 센서의 ON/OFF나 집의 환경을 설정할 수 있고, 도어락의 비밀번호를 변경하거나 슬립모드에 진입하는 등의 각종 환경 설정을 실행할 수 있다.

1. SETTINGS의 경우 커 조이스틱을 누르게 되면 lcd_state_and_joystick의 4번 비트가 1로 SET되게 되고, y좌표 값을 바탕으로 mode의 비트를 토글 시켜 ON,OFF가 가능하게 하였고, 다시 lcd_state_and_joystick의 4번 비트를 CLEAR시켜 다음 조이스틱의 누름을 감지할 수 있게 하였다. 다시 돌아가고 싶으면 back으로 커서를 옮겨 누르면 초기 화면으로 돌아가게 된다.

(조이스틱의 입력 설정 부분)

```
void pind_test_fix(void)
{
   if (PIND != 0xff)
   {
      sww = ~PIND & 0xff;
      _delay_ms(DEBOUNCING_DELAY);
         if (sww == 0x80)
         {
            BIT_SET(lcd_state_and_joystick,4);
            if(lcd_state == 2)
            {
               x++;
            }
         }
   }
}
```

2. PWD_CHANGE의 경우 들어가게 되면 x좌표를 옮김으로서 변경을 원하는 4자리의 비밀번호의 자리를 각각 찾아갈 수 있게 하였고, 여기서 조이스틱을 누르게 되면 Insert Mode로 진입하게 되고, y축을 움직여 비밀번호를 0~9의 숫자로 변경할 수 있게 하였다. 이후 OK버튼으로 커서를 옮기게 되면 올바른 비밀번호를 입력하면 CHANGE PWD 화면으로 넘어가게 되고, 올바른 비밀번호를 입력하지 않았으면 WRONG PWD 출력 후 초기화면으로 넘어간다. CHANGE PWD에서는 위와 동일한 메커니즘으로 비밀번호를 입력하고, OK로 커서를 옮겨 누르게 되면 PW가 변경되고 Uart통신으로 도어락에 변경된 PW를 전송하게 된다.

3. ENV SETTING 에서는 현재의 온/습도 값과, 조도 값, 수위 값을 볼 수 있으며 유의할 점은 1.SETTINGS에서 DHT-11, 조도 센서, 물 수위 센서를 ON 하여야 측정값이 나온다는 것이다. 또한 사용자는 여기서 조이스틱을 한번 누르게 되면 원하는 습도 값, 온도 값, 조도 값, 수위 값을 설정할 수 있게 되고 각각의 기능은 설정값에 따라 가습기, 에어컨, 창문의 열고 닫힘, 절전의 기능을 수행할 수 있게 한다.

4. DOOR OPEN 에 커서가 위치할 때는 조이스틱을 누르게 되면 도어락에 문 열림 정보를 송신한다.

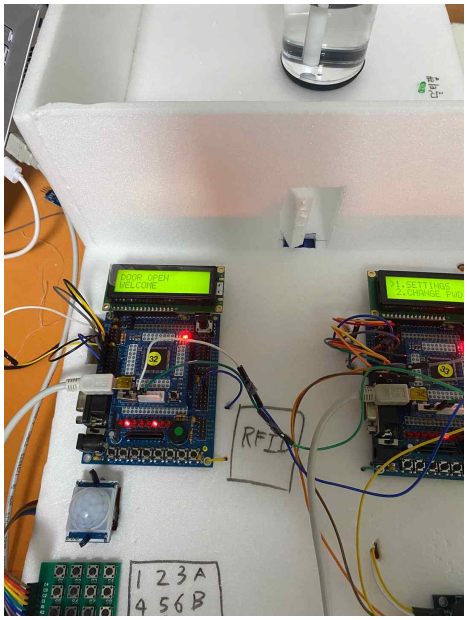
5. SLEEP MODE 에 커서가 위치할 때는 조이스틱을 누르게 되면 제어부 Atmega128이 슬립모드에 진입하게 된다.

(센서 및 액추에이터 설정 부분)

DHT-11부터 aircon_on_off까지의 함수는 위에 서술했듯 mode의 각 비트를 체크하여 '1'이면 각 센서 및 엑추에이터의 기능 ON, '0'이면 각 센서 및 엑추에이터의 기능을 OFF되게 만들었다.

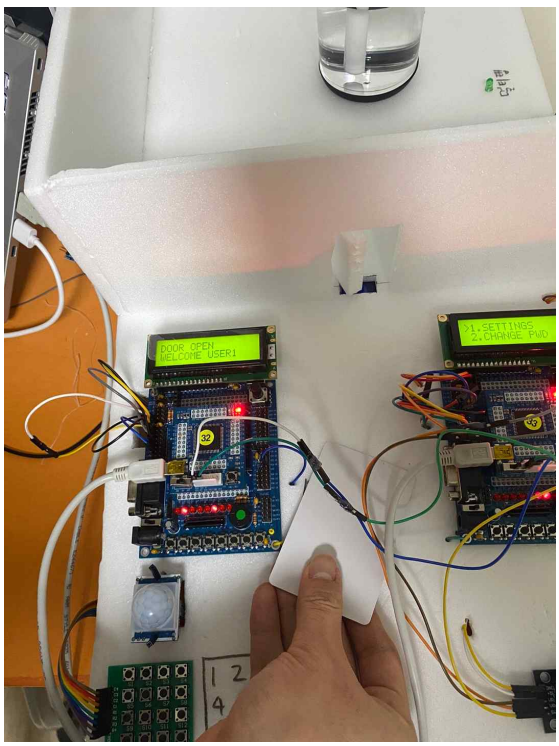
4. 전체 시나리오에 따른 개발 내용 및 검증

(a) 도어락 비밀번호 입력



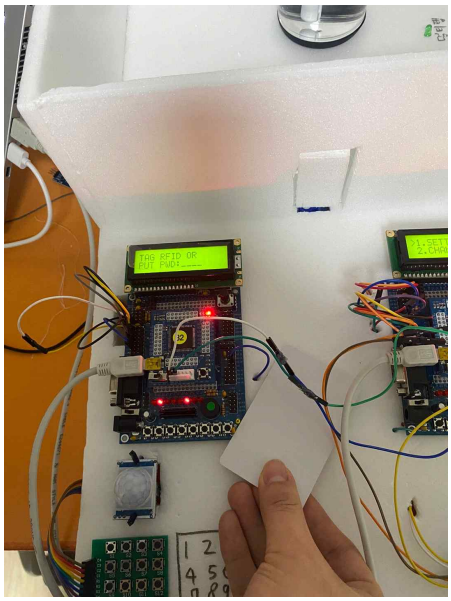
도어락에 비밀번호를 입력하여 문이 열림을 확인하였다.

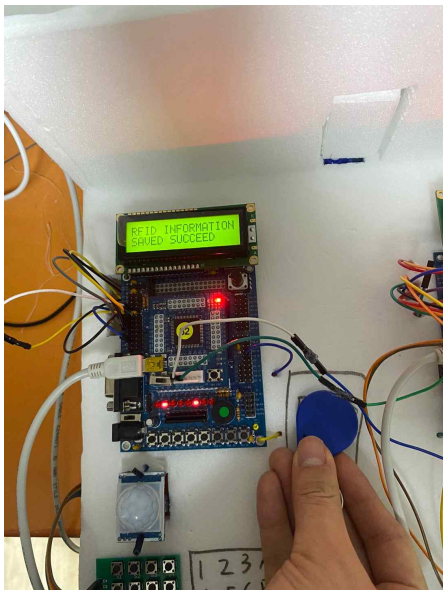
(b) RFID로 도어락 해제



RFID 카드로 도어락을 해제하는 것을 확인하였다

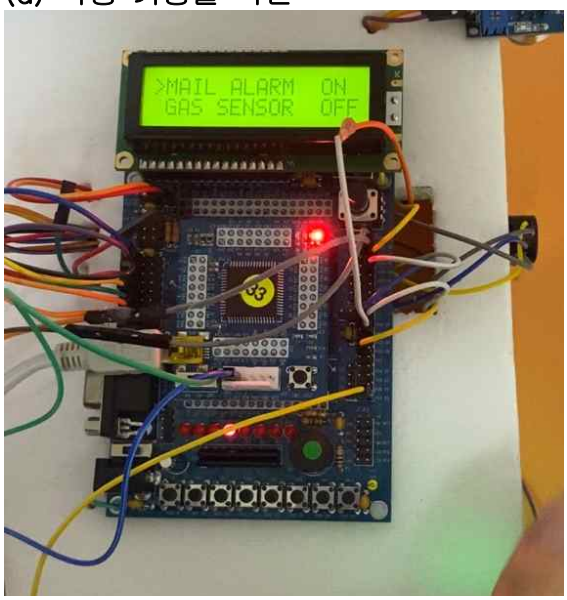
(c) RFID정보 초기화 및 정보 등록





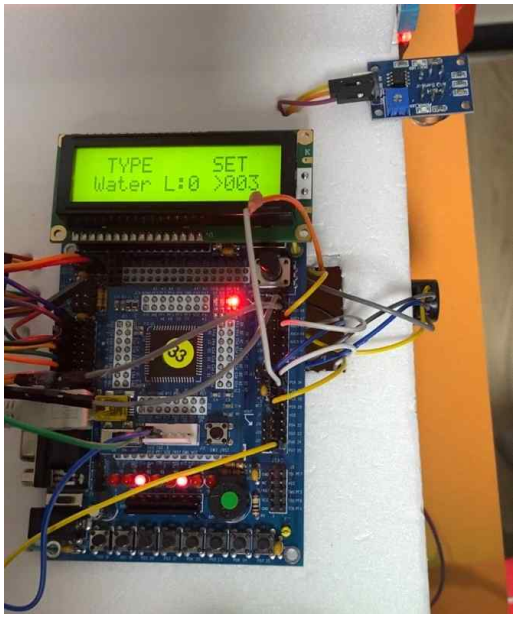
RFID 정보를 초기화 하여 인식이 안 됨을 확인하였고, 이후 RFID 카드를 새로 등록할 수 있는 것을 확인하였다.

(d) 각종 기능들 확인

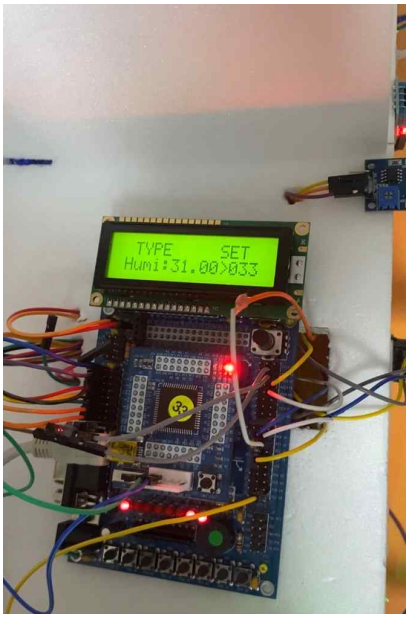


(메일 알람 확인)

(조도 센서 확인)



(수위 센서 확인)



(온습도 기능 확인)

이하 기능들은 제출 시 동영상과 함께 제출하겠다.

5. 텀 프로젝트 수행에 대한 결론 및 기대효과

(a) 초기 계획 대비 진행 내용 비교

초기 계획에서는 블루투스 모듈을 활용한 스마트폰과의 통신으로 각종 센서와 엑츄에이터들을 제어할 수 있게 하는 내용과 부저를 활용한 각종 소리 효과, 슬립 모드의 활용, 사람이 일정 시간동안 감지되면 보안 알림의 계획들도 있었으나 이 세 가지는 개발 시간의 한계로 구현하지 못하였다. 또한 인체감지센서는 외부 인터럽트를 활용하여 슬립 모드의 해제 및 타이머/카운터2의 초기화를 통해 활용을 추가적인 활용을 시도하였으나 인체감지센서의 오작동 등으로 추정되는 문제로 구현하지 못하였다.

그 외의 도어락 기능 및 제어부 기능을 수행하는 프로젝트는 성공적으로 진행되었다고 생각한다.

(b) 전체적인 구현에 대한 Overview

비밀번호 입력과 RFID 카드를 활용할 수 있는 도어락을 개발하였고, 각종 센서와 액츄에이터들을 제어하고 센서의 값을 LCD에 출력하며 이 값들에 따라

액츄에이터들을 제어하였으며, EEPROM을 활용하여 전원이 OFF되어도 사용자의 설정 값들과 패스워드, RFID카드 정보 등의 사용자의 보안과 편의성을 도모하는 정보들을 저장할 수 있었고 Uart통신을 활용해 Atmega128 두 개의 데이터 송수신, SPI통신을 활용해 RFID 모듈을 사용하였고, TWI통신을 활용해 초음파 거리 센서와의 데이터 송수신을 하였다.

(c) 프로젝트 진행 중 어려웠던 내용과 어떻게 해결하였는지

이번 텀프로젝트를 진행하면서 전반적인 프로그래밍을 맡고 회로적인 내용 또한 참여하였다.

프로그래밍을 하면서 어려웠던 점은 Atmega128끼리의 Uart통신을 구현하는 것이 어려웠다. 데이터 패킷을 설계하는 개념부터 Uart통신에 필요한 각종 레지스터를 제어하고 포인터 개념을 활용하여 문자열을 전송하는 과정까지 많은 이해력이 필요하였고, 테스트 코드들을 만들고 프로테우스로 시뮬레이션을 해가면서 데이터 패킷에 대한 이해도를 늘렸고 이를 코드로 구현하는데 성공하였다.

RFID 모듈을 활용하는 것도 난이도가 높았는데 교수님께서 코드를 제공해 주시기 전부터 데이터 시트를 찾아서 해석하고 읽어보며 코드를 짜기 위한 노력을 하였지만 방대한 데이터 시트의 양와 SPI통신에 대한 이해도가 부족하여 구현하기 힘들었다. 만약 코드를 제공받지 못하였다면 RFID모듈을 활용하기가 힘들었을 것이라고 생각한다.

초음파 거리 센서의 값을 활용하는 것도 난이도가 있었는데, 정확한 값을 출력하다가도 가끔씩 오차 범위를 훨씬 넘어서는 값을 출력할 때가 있었다. 메일이 오면 즉각적으로 사용자에게 알람을 줘서 알리려 했지만, 이를 활용하기가 상당히 까다로웠고, 고민 끝에 30번의 거리 값을 더해서 30으로 나누어 평균값을 취하고, 이를 데이터로 활용하여 사용자에게 알림을 줄 수 있게 하는 방법으로 해서 오차 범위를 넘어서는 값이 나오더라도 이를 무시할 수 있는 방법을 구현하였다.

조이스틱의 경우도 조이스틱의 쏠림에 따라 가변 저항의 값이 바뀌어 ADC측정값이 바뀌는데, 이를 활용하여 인터페이스를 조작하기 위해서는 x, y좌표를 1씩 증가시켜야 하였고, 조이스틱이 일정 값 이상으로 넘어가면 어떠한 자료형을 1로 셋 그 자료형이 1로 셋 돼있을 때 조이스틱이 중앙으로 넘어가면 좌표 값을 1 증가시키고 다시 그 자료형을 0으로 바꾸어 좌표 값을 1씩 증가시키는데 성공하였다. 또한 각종 센서가 제대로 값을 출력하는지, 혹은 내가 제작한 코드를 Atmega128에 넣었을 때 제대로 값을 출력하고 있는지 확인하기 어려운 부분이 있었고, 프로테우스의 오실로스코프와 소지하고 있는 휴대용 멀티미터를 사용해서 많은 도움이 되었다. 실제 오실로스코프가 있었으면 더욱 빠르게 해결할 수 있었을 것이라 생각해서 아쉬웠다.

회로적인 부분에서 어려웠던 점은 점퍼선의 특성상 잘 빠지는데 이를 글루건으로 살짝 붙여서 빠지지 않게 해결하였다. 또한 회로를 구성할 때 전압만을 고려하고

전류를 고려하지 않은 부분과 안정적으로 전원을 공급하는 회로를 짜는 지식이 많이 부족했던 점이 아쉬웠다. 만약 다음에 다시 이러한 프로젝트를 진행하는 기회가 온다면, 프로그래밍뿐만이 아니라 회로에 대한 지식 또한 넓히고 싶다는 생각이 들었다.

(d) 진행 미비점 분석 및 발전방향

블루투스 모듈을 활용하여서 스마트폰으로 각종 센서 및 액추에이터들을 제어할 수 없게 된 것이 가장 아쉬웠다. 이는 단지 Atmega128의 프로그램을 추가해야 될 뿐만이 아니라 안드로이드나 IOS앱을 개발할 수 있는 능력까지 있어야 하기에 이후 졸업작품 등을 수행할 때에는 그러한 능력을 길러서 추가하고 싶다는 생각이 들었다. 또한 이런 기능을 추가하면 더욱 완성도가 높은 결과물이 나올 것 이라고 생각한다.

C언어 프로그래밍 능력 또한 부족했다고 생각한다. 특히 LCD쪽 출력 부분에서 포인터와 문자열을 제대로 활용 못하였다고 생각하고, C언어의 spirnf등의 함수들의 지식 및 활용도가 많이 부족하다고 느꼈다. 이러한 능력을 향상시키면 더욱 깔끔하고 활용적인 코드를 만들 수 있을 것이라고 생각한다.

마지막으로 프로그래밍뿐만이 아니라 센서나 액츄에이터의 데이터 시트를 분석할 수 있는 능력과, 전반적인 아이디어를 바탕으로 잘 작동할 수 있는 회로를 꾸미는 것 또한 중요하다고 생각했고 이런 것들이 많이 부족하였기 때문에 하드웨어적으로 부족했다고 생각한다. 이 또한 보완할 수 있으면 완성도가 더욱 높아질 것이라고 생각한다.

(e) 본 과제물의 기대 효과

본 과제물의 장점은 릴레이 모듈을 활용하여 쓰지 않는 제품의 절전을 하여 대기전력을 차단하는 것과, 물 수위 센서를 사용하여 집이 침수가 되었을 때 전원을 OFF 시켜 감전에 의한 추가 사고가 나지 않게끔 하는 기능이 있다는 것이라고 생각한다. 또한 IoT기능을 가미한 스마트 하우스 모델을 학우들에게 선보여 IoT에 대한 이해력과 관심도를 상승시킬 수 있을 것이라고 기대된다.

6. 소요된 부품 List 및 수행된 일정 정리

(a) 소요된 부품 List

1. Atmega128 (2개) (제어부, 도어락)
2. HC-SR501 : 인체감지센서
3. DHT-11 : 온습도 센서
4. SZH-SSBH-011 : 광/조도 센서
5. SZH-SSBH-038 : 유해가스 측정 센서
6. HC-SR04 : 초음파 거리 센서
7. 물 수위 센서
8. 서브 모터 2개
9. LCD
10. ONE006 : 조이스틱
11. 4x4 keypad
12. 릴레이 모듈
13. 릴레이 모듈 8채널
14. 저항 및 점퍼선등
15. RFID 모듈
16. FAN
17. humidifier
18. Breadboard Power Supply Module

(b) 수행 일정

-11/14 ~ 11/20 : 프로젝트 계획서 발표 및 필요 물품 조달
-11/21 ~ 12/04 : 프로젝트에 필요한 센서 테스트 및 인터페이스 구축
-12/05 ~ 12/11 : 테스트 모델 제작 및 프로그램 테스트
-12/12 : 프로젝트 중간 발표
-12/13 ~ 12/21 : 모델 제작 및 프로그램 제작 및 검증, 버그 테스트
-12/22 : 프로젝트 최종 발표