ATmega128을 이용한 음주운전 방지 자동차 키 보관함 개발

32171095 김진하, 32174380 주소연

<요약> 최근 들어 음주운전으로 인한 교통사고가 빈번하게 발생하고 있다. 따라서 이런 문제를 방지하기 위하여 음주 후에는 자동차 키를 꺼낼 수 없게 하는 자동차 키 보관함을 생각하게 되었다. 본 논문에서는 MCU로 ATmega128을 사용하여 알코올센서를 통해 음주측정 여부를 확인한 후에 보관되어 있는 자동차 키를 꺼낼 수 있도록 하는 보관함 기능을 구현한다.

1. 서론

음주를 한 후 행동 능력과 판단 능력이 떨어진 상태에서 운전을 하는 경우가 있어 많은 문제가 발생하고 있다. 이런 경우를 예방하기 위해 음주 전에 자동차 키를 보관함에 넣어둔 후 그 키를 다시 되찾으려면 음주 측정을 하여 음주를 하지 않았음을 증명해야 그 키를 되찾을 수 있는 보관함을 생각하게 되었다. 또한, 이 보관함은 자동차 키뿐만 아니라 술에 취한 상태에서 만져서는 안 될 물건도 보관할 수 있도록 한다.

이에 본 논문은 음주측정 기능이 들어간 보관함을 구현하게 되었다. 알코올센서를 통해 음주측정을 한 뒤, 음주를 하지 않은 것이 확인이 되면 그 후에 자동차 키가 보관되어 있는 보관함의 도어록을 열 수 있도록 하였다.

2. 개발 과정

2.1 개발환경

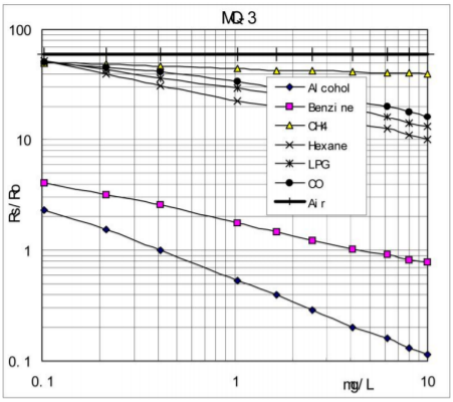
- OS: Windows
- Program tool: Atmel Studio 7.0
- AVR MCU: Atmega128

2.2 구현기술

보관함을 만들기 위하여 음주측정을 위한 알코올센서 mq-3 모듈을 사용하였으며, 도어록을 열기 위해서는 키패드를 통해 비밀번호를 입력하고 입력한 숫자가 FND를 통해 보이도록 구현하였다. 또한, 도어록이 열린 것은 모터가 돌아가는 모습을 통해 나타내도록 구현하였다.

2.2.1 알코올센서 mq-3

음주측정을 위해 mq-3모듈을 사용했다. 센서를 통해 측정한 값을 아날로그 값으로 받으면 그 값을 ADC를 통해 디지털 값으로 변환한다. ATmega128의 ADC는 10비트의 분해능을 가지고 있는 8개의 ADC를 내장하고 있다. 이때, ADC를 위해서는 PF0에서 PF7사이의 핀을 사용한다.



<그림 1>

<그림 1>은 여러 기체에 대한 MQ-3의 대표적인 민감도 특성을 나타낸다.

온도: 20℃

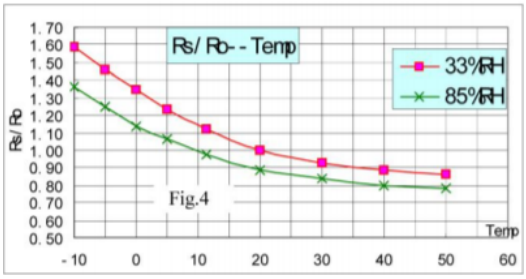
습도: 65%

산소 농도: 21%

RL(부하저항) = 200kΩ

R_O: 깨끗한 공기에서 알코올 0.4mg/L의 센서저항

R_s: 다양한 농도의 기체에서의 센서저항



<그림2>

<그림 2>는 MQ-3가 온도와 습도에 대한 일반적인 의존도를 보여준다.

R_O: 33% RH와 20℃에서 공기중 0.4mg/L의 알코올에서 센서 저항

R_s: 다른 온도와 습도에서 0.4mg/L의 알코올에서 센서 저항

ADC 값을 통해 센서 출력 전압/5V = RL/(RS+RL)

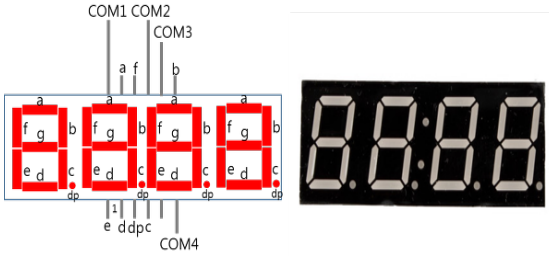
ADC값/1023 = RL/(RS+RL)

음주측정을 하여 알코올이 측정되는 정도가 어느정도 이상이면 빨간색 LED가 켜지면서 도어록을 열기 위해 키패드를 눌러도 FND 화면이 켜지지 않고 아무런 일도 일어나지 않도록 하였다. 이때, 알코올 측정이 되지 않으면 초록색 LED가 켜지면서 도어록을 열 수 있다. 알코올 센서에 관한 코드는 다음과 같다.

```
alcohol_yes=value*5.0/1023.0;

void init_adc()    // ADC 초기 설정 함수
{
 ADMUX = 0x41;    //0b01000001
// '01' : AVCC(+5V) 기준전압 사용, '0' : 디폴트 오른쪽 정렬, "00001" : ADC1 사용, 단극 입력
ADCSRA = 0x87; //0b10000111
// '1' : ADC 허용, '0' : 한번만 변환, '111' : 프리스케일러 128분주
}

unsigned short read_adc()
// AD 변환 시작 및 결과 읽어오는 함수
{
  unsigned char adc_low, adc_high;
  unsigned short value;
  ADCSRA |= 0x40;
  //ADC 변환 시작
  while ((ADCSRA & 0x10) != 0x10)
  //ADC 변환 완료 검사
  ;
  adc_low = ADCL;
  adc_high = ADCH;
  value = (adc_high << 8) | adc_low;
  //value는 High 및 Low 연결 16 비트값
  return value;
}

void show_adc_led(unsigned short value)
//값을 비교하여 LED ON 또는 OFF 함수
{
  if(value<2.0) PORTE=0x00;
   else if(value<4.0) PORTE=0x40; // value값 4.0보다 작으면 초록색 LED on
   else PORTE=0x80; // value값 4.0보다 크거나 같으면 빨간색 LED on
}
```

2.2.2 도어록

도어록 구현을 위해 4×4 키패드와 4digit FND를 사용하였다.



<그림 3> 7세그먼트 외형과 회로

키패드를 통해 네 자리 비밀번호를 입력하면, 입력한 숫자가 FND를 통해 나타난다. 또 비밀번호 입력을 완료한 후, 키패드에 있는 버튼'A'를 눌렀을 때 비밀번호가 맞으면 FND화면에 'OPEn'이 나타나고, 비밀번호가 틀리면 'FAIL'이 나타난다. 도어록에 관한 코드는 다음과 같다.

```
char key='n';
char pWBuf[4] = { 'n' , 'n' , 'n' , 'n' };
char pW[4] = { fnd_num[0], fnd_num[0],
fnd_num[0], fnd_num[0]};


while(1)
  {
    if(alcohol_yes<4.0)
     {
       key = KeySet();
       if ( key == 'A')
         //비밀번호 일치 여부 확인
        {
          if( pWBuf[0] != pW[0])
           {
             pwReset(pWBuf);
             key = 'n';
             ledBuf_nopen();
           }

          else if( pWBuf[1] != pW[1])
           {
             pwReset(pWBuf);
             key = 'n';
             ledBuf_nopen();
           }

          else if( pWBuf[2] != pW[2])
           {
             pwReset(pWBuf);
             key = 'n';
             ledBuf_nopen();
           }

          else if( pWBuf[3] != pW[3])
           {
             pwReset(pWBuf);
             key = 'n';
             ledBuf_nopen();
           }

          else
           {
             ledBuf_open();
             servo_open();
             pwReset(pWBuf);
             continue;
           }
        }
      if (!(key=='n')) //비밀번호 입력 받
음

       {
         if(pWBuf[0] == 'n')
           {
             pWBuf[0] = key;
             _delay_ms(200);
             continue;
           }
         if(pWBuf[1] == 'n')
           {
             pWBuf[1] = key;
             _delay_ms(200);
             continue;
           }
```

```
            if(pWBuf[2] == 'n')
              {
                pWBuf[2] = key;
                _delay_ms(200);
                continue;
              }
            if(pWBuf[3] == 'n')   //비밀번
호 FND화면에 출력
              {
                pWBuf[3] = key;
                _delay_ms(200);
                PORTB = 0xf7;
                PORTA = pWBuf[0];
                _delay_ms(200);

                PORTB = 0xbf;
                PORTA = pWBuf[1];
                _delay_ms(200);

                PORTB = 0xdf;
                PORTA = pWBuf[2];
                _delay_ms(200);

                PORTB = 0xef;
                PORTA = pWBuf[3];
                _delay_ms(200);
                continue;
              }
             pwReset(pWBuf);
          }
       }
   }


char KeySet(void)
{
    char KeyBuff = 'n';   //함수 내에서 사용
될 버퍼 선언
    DDRD = 0x0f; PORTD=0xff;

    DDRD = 0x01; PORTD &= ~0x01;
    _delay_us(5);
    if(!(PIND&0x10)) KeyBuff = fnd_num[1];
    if(!(PIND&0x20)) KeyBuff = fnd_num[2];
    if(!(PIND&0x40)) KeyBuff = fnd_num[3];
    if(!(PIND&0x80)) KeyBuff = 'A';
    DDRD=0x00; PORTD = 0xff;
          // 1 2 3 A 행 검사

    DDRD = 0x02; PORTD &= ~0x02;
    _delay_us(5);
    if(!(PIND&0x10)) KeyBuff = fnd_num[4];
    if(!(PIND&0x20)) KeyBuff = fnd_num[5];
    if(!(PIND&0x40)) KeyBuff = fnd_num[6];
    if(!(PIND&0x80)) KeyBuff = 'B';
    DDRD = 0x00; PORTD = 0xff;
          // 4 5 6 B 행 검사

    DDRD = 0x04; PORTD &= ~0x04;
    _delay_us(5);
    if(!(PIND&0x10)) KeyBuff = fnd_num[7];
    if(!(PIND&0x20)) KeyBuff = fnd_num[8];
    if(!(PIND&0x40)) KeyBuff = fnd_num[9];
    if(!(PIND&0x80)) KeyBuff = 'C';
    DDRD = 0x00; PORTD = 0xff;
          // 7 8 9 C 행 검사

    DDRD = 0x08; PORTD &= ~0x08;
    _delay_us(5);
    if(!(PIND&0x10)) KeyBuff = '*';
    if(!(PIND&0x20)) KeyBuff = fnd_num[0];
    if(!(PIND&0x40)) KeyBuff = '#';
    if(!(PIND&0x80)) KeyBuff = 'D';
    DDRD = 0x00; PORTD = 0xff;
          // * 0 # D 행 검사

    return KeyBuff;
}

void pwReset (char *pwbuf)
{
```

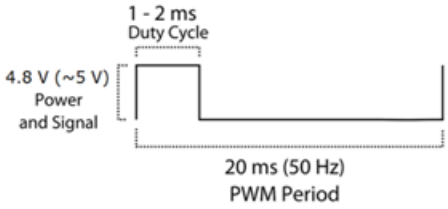
```
    for( unsigned int i = 0 ; i < 4 ; i++)
    {
        pwbuf[i] = 'n';
    }
}


void ledBuf_open(void)
          //비밀번호 맞았을 경우
{
    DDRA = 0xff;
    DDRB = 0xff;

    for( unsigned int i = 0 ; i < 2 ; i++)
    {
        for( unsigned int i = 0 ; i < 50 ; i++)
        {
            PORTA = 0x3f;
            PORTB = 0xf7;
            _delay_ms(5);

            PORTA = 0x73;
            PORTB = 0xbf;
            _delay_ms(5);

            PORTA = 0x79;
            PORTB = 0xdf;
            _delay_ms(5);

            PORTA = 0x54;
            PORTB = 0xef;
            _delay_ms(5);

        }

        PORTB = 0xff;
        _delay_ms(1000);
    }
}
```

```
void ledBuf_nopen(void)
          //비밀번호 틀렸을 경우
{
    DDRA = 0xff;
    DDRB = 0xff;

    for( unsigned int i = 0 ; i < 2 ; i++)
    {
        for( unsigned int i = 0 ; i < 50 ; i++)
        {
            PORTA = 0x71;
            PORTB = 0xf7;
            _delay_ms(5);

            PORTA = 0x77;
            PORTB = 0xbf;
            _delay_ms(5);

            PORTA = 0x06;
            PORTB = 0xdf;
            _delay_ms(5);

            PORTA = 0x38;
            PORTB = 0xef;
            _delay_ms(5);

        }

        PORTB = 0xff;
        _delay_ms(1000);
    }
}
```

2.2.3 서브 모터 s90

보관함의 문이 열리는 것을 나타내기 위해 서보 모터를 사용하였다. 서보 모터는 지정된 각도만큼 회전할 수 있으며, 위치, 방위, 자세 등의 정밀 제어가 가능하다. 0에서 180도 사이만 회전이 가능하며

PWM 신호를 통해 위치를 제어한다. 서보 모터에는 3개의 연결선이 있는데, 붉은색 선은 VCC, 갈색 선은 GND, 그리고 주황색 선은 위치 설정을 위한 제어선이다.



<그림 4>

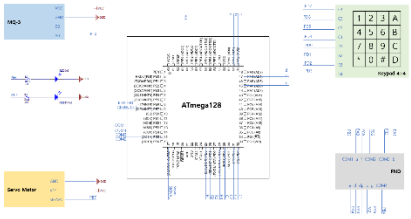
<그림 4>와 같이 PWM Period를 20ms(50Hz)로 설정하고 Duty Cycle을 1.5ms pulse로 설정하면 위치가 "0"으로 되고, ~2ms pulse로 설정하면 오른쪽으로 "90", 그리고 ~1ms pulse로 설정하면 왼쪽으로 "90"이 된다.

도어록을 여는데 성공해서 FND 화면에 'OPEn'이 나타나면 모터가 돌아가고, 실패해서 FND 화면에 'FAIL'이 나타나면 모터가 돌아가지 않는다. 서보 모터에 관한 코드는 다음과 같다.

```
#define TC1_TOP 28338L
#define WIDTH2 2211L //1ms
#define WIDTH3 4400L // 2ms

DDRB = 0xff; //PB7에 연결
TCCR1A |= (1<<COM1C1) | (0<<COM1C0);
TCCR1A |= (1<<WGM11) | (0<<WGM10);
TCCR1B |= (1<<WGM13) | (1<<WGM12);
TCCR1B |= 0x02; //분주비 clk/8
ICR1H = TC1_TOP>>8;
ICR1L = TC1_TOP & 0xFF;
OCR1CH = WIDTH2>>8;
OCR1CL = WIDTH2&0xFF;
_delay_ms(1000);
```

```
void servo_open(void)
{
      OCR1CH = WIDTH3>>8;
    OCR1CL = WIDTH3&0xFF;
    _delay_ms(1000);
}
```

2.2.4 회로도



https://drive.google.com/file/d/1Ly3edViGrzZYRliW-28vfVXeI08J8ox5/view?usp=sharing

3. 결론 및 고찰

제일 먼저 센서를 통해 음주측정을 한 후, 초록색 LED가 켜져야 키패드 입력을 통해 도어록을 열 수 있도록 해야 하는데, 구현한 결과를 보면 처음부터 초록색 LED가 켜져 있는 것을 볼 수 있다. 이렇게 되면 알코올을 측정하기 전부터 도어록을 열 수 있다는 문제점이 생기게 된다. 추후 이 문제를 보완하여 센서를 통해 측정하기 전에는 키패드를 입력할 수 없도록 수정해볼 계획이다. 또한 이번 구현에서는 계획했던 비밀번호 불일치, 알코올 감지 시 5분 동안 비밀번호를 입력할 수 없는 기능을 추가하지 못했다. delay 함수 혹은 다른 방법을 이용하여 수정해보려 한다.

문제점을 개선하여 이 보관함을 실제로 만들어서 실생활에 사용해 보면 좋을 것 같다. 자

동차 키를 보관하는 것 외에도 음주 후에 핸드폰의 버튼을 잘못 눌러 전화를 잘못 거는 상황을 방지하기 위해 음주 전 보관함에 핸드폰을 보관할 수도 있다. 또한, 파손의 위험이 있는 귀중품도 음주 후에 들고 다니다가 떨어뜨릴 수도 있으므로 음주 전에 미리 보관함에 보관해 둘 수도 있다. 이처럼 이 보관함이 실제로 개발이 된다면 매우 유용하게 사용될 것 같다.

4. 참고자료

[1] https://kogun.tistory.com/11

[2] https://blog.naver.com/cho6716666/221915081127

[3] https://jdselectron.tistory.com/42

[4] http://hanjindata.iptime.org/myweb/shop/P0174/MQ3.pdf

[5] https://www.teachmemicro.com/mq-3-alcohol-sensor/