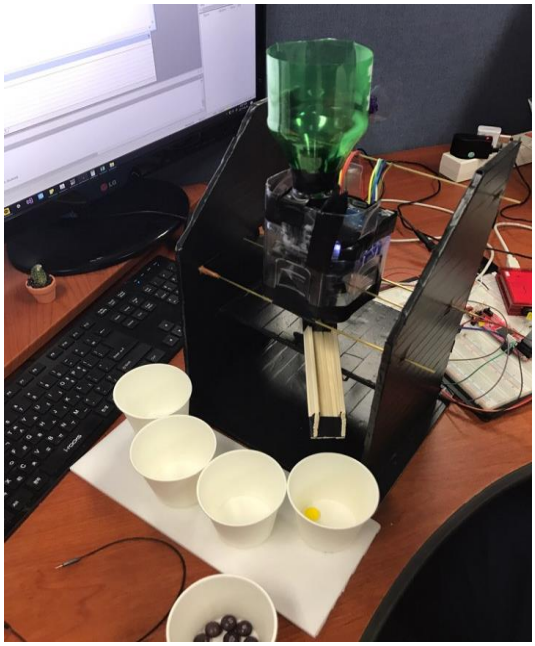
# 분류기 제작 결과보고서

20153110 정예지



## 1. 시스템

1) 디바이스를 작동시킨다.

- 어플리케이션을 통해 블루투스 연결을 하고, Start 버튼을 터치한다.
- 브레드보드의 interrupt 5번 port와 연결된 스위치를 누른다.

2) 깔대기를 통해 첫 번째 칸에서 초코볼을 받는다.

- 첫 번째 칸은 한 개의 서보모터와 컬러센서로 이루어진다.
- 초코볼을 하나씩 받는다.

3) 받은 초코볼의 색을 판별하고 두 번째 칸으로 보낸다.

- 서보모터는 3개의 각도로 움직인다.
- 첫 번째 각도( 0 도 )는 깔대기로부터 초코볼을 받는 역할.
- 두 번째 각도( 150 도 )는 컬러센서를 통해 색을 분류하는 역할.
- 세 번째 각도( 180 도 )는 바닥의 구멍을 통해 초코볼을 두 번째 칸으로 보내는 역할.

4) 두 번째 칸의 미끄럼틀을 통해 바닥의 종이컵으로 초코볼을 보낸다.

- 두 번째 칸은 하나의 서보모터와 그에 붙은 미끄럼틀로 이루어진다.

- 서보모터는 4개의 각도로 움직인다.

- 첫 번째 각도( 0 도 )는 빨간색 초코볼을 보내는 역할.

- 두 번째 각도( 40 도 )는 주황색 초코볼을 보내는 역할.

- 세 번째 각도( 80 도 )는 초록색 초코볼을 보내는 역할.

- 네 번째 각도( 120 도 )는 노란색 초코볼을 보내는 역할.

5) 디바이스를 멈춘다.

- 어플리케이션의 Stop 버튼을 터치한다.

- 브레드보드의 interrupt 6번 port와 연결된 스위치를 누른다.

## 2. 코드

```
#pragma interrupt_handler int5_isr:iv_INT5
void int5_isr(void)
{
    run_Chk = 1;
}
```

- 동작 실행 스위치.

- run_Chk가 true가 되면 Main함수의 동작의 실행조건을 만족한다.

```
#pragma interrupt_handler int6_isr:iv_INT6
void int6_isr(void)
{
    run_Chk = 0;
}
```

- 동작 정지 스위치.

```
#pragma interrupt_handler usart0_rx_isr:iv_USART0_RX
void usart0_rx_isr(void)
{
    data = UDR0;
    if(data == '1')
        run_Chk = 1;
    else if(data == '0')
        run_Chk = 0;

}
```

- 어플리케이션을 통한 동작 실행/정지 스위치

- 블루투스 통신을 통해 받은 데이터에 따라 동작을 실행하거나 정지시킨다.

```
void color_check(void)
{
    PORTD = RED;
    delay_ms(50);
    clk_red = clock;
    PORTD = BLUE;
    delay_ms(50);
    clk_blue = clock;
    PORTD = WHITE;
    delay_ms(50);
    clk_white = clock;
    PORTD = GREEN;
    delay_ms(50);
    clk_green = clock;
}
```

- PORTD의 값을 변경하여 컬러센서의 측정모드를 변경시킨다.

- 50ms 주기에 빨강, 파랑, 하양, 초록 순서대로 측정한다.

- 모드별로 얻은 clock을 색깔별로 분류된 변수에 저장한다.

```
void init_TCS3200(void)
{
    DDRD = 0x0f;
    PORTD = 0x02;
}
```

- 컬러센서의 초기화 함수.

```
void init_servoPin(void)
{
    DDRE = 0x18;
}
```
- 서보모터의 초기화 함수.

```
void init_Timer1(void)
{
    TCCR1A= 0x00;
    TCCR1B = 0x43;
    TCCR1C = 0x00;
    TIMSK = 0x24;
}
```

- Timer1의 초기화 함수.

- 컬러센서를 위해 쓰인다.

```
void init_Timer3(void)
{
    TCCR3A = 0xA2;
    TCCR3B = 0x1B;
    TCCR3C = 0x00;
    ICR3 = 4999;
}
```

- Timer3의 초기화 함수.

- 두 개의 서보모터를 위해 쓰인다.

- Timer3는 16bit-timer로 OCR을 A, B, C로 세 개를 사용할 수가 있어서 편하다.

```
void init_Uart(void)
{
    UCSR0A = 0x00;
    UCSR0B = 0x98;
    UCSR0C = 0x06;
    UBRR0H = 0x00;
    UBRR0L = 103;
}
```

- UART 통신을 위한 초기화 함수.

- 송/수신 허가.

- Baud Rate = 9600

```
void servo_angle1(int angle)
{
    TCCR3A = 0x82;
    OCR3A = angle;
}
```

- 첫 번째 서보모터의 각도를 제어하는 함수.

- 첫 번째 서보모터는 첫 번째칸에 존재한다.

```
void servo_angle2(int angle)
{
    TCCR3A = 0x22;
    OCR3B = angle;
}
```

- 두 번째 서보모터의 각도를 제어하는 함수.

- 두 번째 서보모터는 두 번째칸에 존재한다.

```
char rx_char(void)
{
    while((UCSR0A & 0x80) == 0);
    return UDR0;
}
```

- UART 통신을 통해 Atmega에서 받은 데이터를 반환하는 함수.

```
void tx_char(char tx_data)
{
    while((UCSR0A & 0x20) == 0 );
    UDR0 = tx_data;
}
```

- UART 통신을 통해 Atmega에서 인자를 데이터로 보내는 함수.

```
void delay_ms(int ms)
{
    int i, j;
    for(i = 0; i < ms; i++)
        for(j = 0; j < 2100; j++);
}
```

- 코드의 흐름을 인자로 받은 시간만큼 중단시키는 함수.

```
#pragma interrupt_handler timer1_ovf_isr:iv_TIMER1_OVF
void timer1_ovf_isr(void)
{
    cnt_timer1_ovf1++;
}
```

- Timer1에서 overflow가 생길때의 인터럽트 함수.

```
#pragma interrupt_handler timer1_capt_isr:iv_TIMER1_CAPT
void timer1_capt_isr(void)
{
    ending_edge = ICR1H<<8;
    ending_edge |= ICR1L;
    if(cnt_timer1_ovf1){
        starting_edge = ending_edge;
        cnt_timer1_ovf1 = 0;
        return;
    }
    clock = ending_edge - starting_edge;
    starting_edge = ending_edge;
}
```

- Timer1의 캡처 인터럽트 함수.

- 컬러센서는 이를 통해 얻은 clock 값을 참조한다.

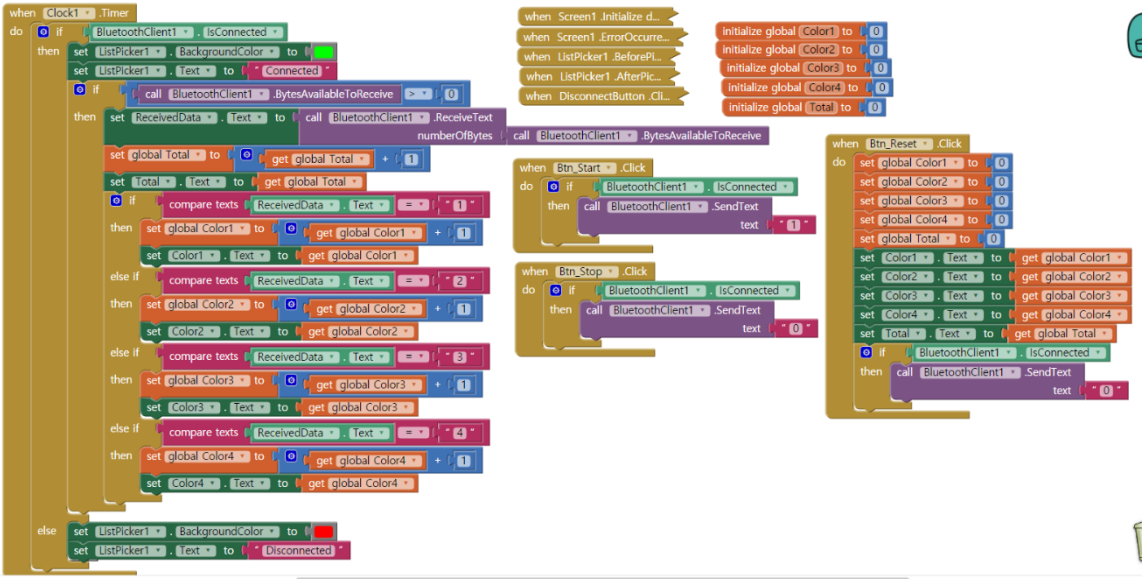
```
int main(void)
{
    DDRE = 0x9f;
    EICRA = 0x00;
    EICRB = 0x28;
    EIMSK = 0x60;
    init_TCS3200();
    init_servoPin();
    init_Timer1();
    init_Timer3();
    init_Uart();

    SREG |= 0x80;

    while(1){
        while(run_Chk){
            /*angle1 is top, angle2 is bottom.*/
            servo_angle1(130);
            delay_ms(500);
            servo_angle1(450);
            delay_ms(500);
            color_check();
            if(clk_red > 21000 && clk_blue > 12000){   //Red
                servo_angle2(130);
                tx_char(CRED);   //Count up in app.
                delay_ms(300);
            }
            else if(clk_red > 15000 && clk_blue > 8000){   //Orenge
                servo_angle2(190);
                tx_char(CORENGE);
                delay_ms(300);
            }
            else if(clk_red > 12000 && clk_blue > 6000){   //Green
                servo_angle2(250);
                tx_char(CGREEN);
                delay_ms(300);
            }
            else if(clk_red > 7000 && clk_blue > 3000){   //Yellow
                servo_angle2(310);
                tx_char(CYELLOW);
                delay_ms(300);
            }
            else{                        //Nothing
                servo_angle2(130);
                delay_ms(300);
            }
            servo_angle1(550);
            delay_ms(300);
        }

    }
    return 0;
}
```

- while문 위의 내용은 초기값 설정에 대한 내용이다.

- 맨 위의 시스템 흐름에 따라 while문 안이 동작한다.

- 모드별로 측정된 네가지 clock 값 중에서 clk_red와 clk_blue로 색을 판별한다.

## 3. 어플리케이션




- 앱인벤터를 사용한 코드

- 스마트폰 어플리케이션 제작

- Atmega와의 블루투스 통신이 주목적.

## 4. 느낀 점

이번에 조별로 프로젝트를 진행함으로써, 다른 이들이 다르게 생각하는 폭넓은 견해를 배웠고, 조원들과 의사소통을 통해 하나씩 개선해 나가는 것이 좋은 경험이 되었다.

Atmega를 흥미있게 공부를 하면서 아두이노 또한 접하게 되었다. 아두이노에 비해 AVR로 각종 센서를 사용할 때, 설정해야할 레지스터값들이 많고 복잡하다고 느끼게 되었다. 하지만 이번 제작을 통해 AVR을 쓰는데 많은 레지스터값들을 변경하는 것이 크게 어렵지않고, 오히려 아두이노에 비해 더 자세한 내용을 수정할 수 있어 더 전문적인 것을 알게되어 많은 인식의 변화를 느꼈다. 이번 제작에는 앞서 말한 것과 같이 소프트웨어 부분에서는 디버깅하는 것 외에 큰 시간이 들지 않았다. 반면에, 하드웨어 부분에서 많은 고생을 하였는데, 부품의 결함을 확인하지 않고 무턱대

고 만들어서 분해하고 다시 만들기를 3번 반복하였다. 하드웨어를 제작하면서 중간중간 테스트를 해야한다는 것을 알았고, 완성이 된다고 해서 기대한대로 동작하는 것이 아니란 것을 알게되었다. 눈으로만 봤을때는 간단해보였던 것들이 실제로 제작하면서 여러 난해한 점을 경험하게 되었고, 앞으로 설계를 할 때, 이번에 보지 못했던 부분들을 꼼꼼히 하나하나 살펴보며 그전보다 더 나은 결과물을 내보일 것이라는 자신감을 얻게 되었다. 이번 경험을 살려 앞으로도 더 참신하고 남들에게 도움이되는 발명품을 만들고싶다.