

Chapter 258. ADC 제어

1. 아두이노

스마트폰과 아두이노를 블루투스로 연결하고 아두이노에 연결된 가변저항의 값이 스마트폰에 표시되는 어플리케이션을 작성해 보자. 아두이노 우노에는 가변저항을 그림 1과 같이 아날로그 입력 핀에 연결한다.

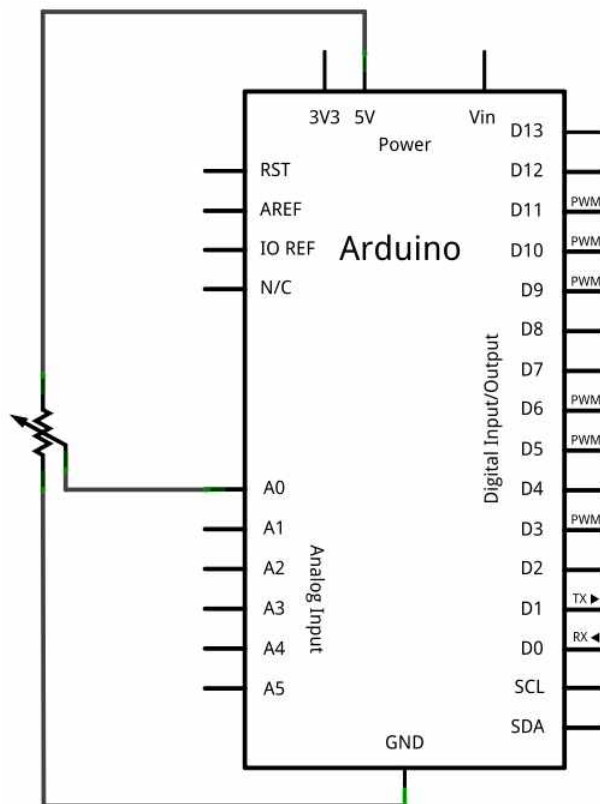


그림 1. RGB LED 연결 회로도

아두이노 우노에 사용된 ATmega328에는 6개의 아날로그 채널이 준비되어 있으며 A0에서 A5까지의 아날로그 입력 핀이 여기에 해당한다. 아날로그 입력 핀은 디지털 입출력 핀으로도 사용이 가능하므로 디지털 입출력 핀과 동일한 방식으로 사용이 가능하다. 아날로그 입력 핀의 수가 6개이지만 이들 핀은 모두 하나의 아날로그-디지털 변환기(Analog-Digital Converter, ADC)에 연결되어 있으므로 동시에 두 개 이상의 아날로그 값을 읽어 들일 수는

없다. 두 개의 아날로그 값을 읽어 들이기 위해서는 채널 전환이 필요하며 이 과정은 시간이 소요되므로 여러 채널을 읽는 경우에는 서로 다른 아날로그 입력 핀의 값을 읽는 문장 사이에 약간의 시간 지연이 필요하다는 점에 유의하여야 한다.

아두이노 우노의 ADC는 10비트의 해상도를 가지고 있다. 따라서 반환하는 최소값은 0이며 최대값은 $1023 (= 2^{10} - 1)$ 이 된다. 아두이노의 동작전압은 5V이므로 5V값이 가해지는 경우 1023을, GND 값이 가해지는 경우 0을 반환하므로 구별할 수 있는 전압은 약 4.88mV 정도가 된다. 아날로그-디지털 변환을 위한 최대 전압은 analogReference 함수로 조절할 수 있다.

스케치 1은 analogRead 함수를 사용하여 가변저항의 값을 디지털로 변환하여 읽어 들이는 스케치의 예로 시리얼 모니터를 통해 현재 가변저항에 가해지는 전압을 0~1023 사이의 값으로 반환한다.

스케치 1. 가변저항 테스트

```
int pin = A0;                                // 가변저항 연결 핀

void setup(){
  Serial.begin(9600);                          // 시리얼 통신 초기화
}

void loop(){
  int value = analogRead(pin);                 // 가변저항 값 읽기

  Serial.print("Current read is ");
  Serial.print(value);                         // 디지털 값
  Serial.print(", approximately ");
  Serial.print(value * 5 / 1023.0);            // 아날로그 전압 값
  Serial.println(" V.");

  delay(1000);
}
```



그림 2. 스케치 1 실행 결과

2. UI 디자인

아두이노에 가변저항을 연결하였으므로 어플리케이션의 UI를 만들어보자. 안드로이드 스튜디오에서 새 프로젝트를 선택하고 프로젝트 이름은 'Bluetooth_Variable_Resistor'로 하자. 프로젝트 생성을 위한 옵션은 디폴트값을 사용한다. 어플리케이션에서는 수신된 현재 가변저항의 값을 탐색바(SeekBar)의 위치와 텍스트뷰(Textview)의 숫자로 표시한다.

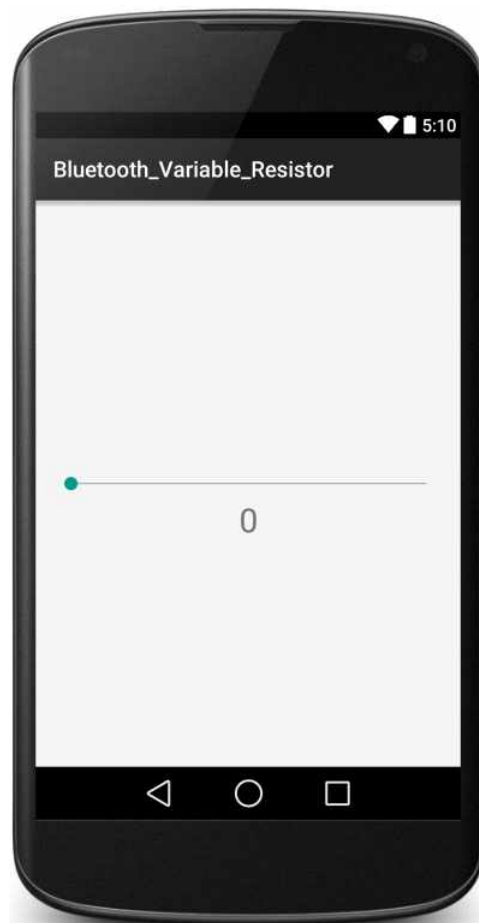


그림 3. 가변저항 값 표시

코드 2. UI 디자인

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity"
    android:gravity="center" >
```

```
<TextView
    android:text="0"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/seekBarVariableResistor"
    android:layout_centerHorizontal="true"
    android:textSize="30sp"
    android:id="@+id/textViewVariableResistor" />

<SeekBar
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:id="@+id/seekBarVariableResistor"
    android:layout_centerVertical="true" />

</RelativeLayout>
```

3. 블루투스 통신

블루투스 통신을 위한 어플리케이션의 기본 코드는 블루투스 프로그래밍 장의 코드를 사용한다. package 문장 이후 모든 코드를 블루투스 기본 코드로 대체하자. AndroidManifest.xml 파일에서 블루투스 사용을 위한 퍼미션을 최상위 요소 아래에 추가하는 것도 잊지 말아야 한다.

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

탐색바는 0에서 1023 사이의 값을 가지므로 먼저 탐색바의 최대 및 최소값을 이에 맞게 수정해 주어야 한다. 탐색바의 범위 설정은 MainActivity 클래스의 onCreate 함수에서 수행한다.

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    SeekBar bar = (SeekBar)findViewById(R.id.seekBarVariableResistor);
    bar.setMax(1023);

    checkBluetooth();
}

```

블루투스를 통한 데이터 전달은 문자열 기반으로 종료문자가 포함되어 전달되므로 종료문자를 수신하면 현재 수신된 가변저항의 값에 따라 탐색바의 썸(thumb) 위치를 변경하고 텍스트뷰의 문자열을 수정하도록 Handler 클래스의 멤버함수인 run 함수를 수정하면 된다.

```

public void run(){
    // 수신된 문자열 데이터에 대한 처리 작업
    int variable_resistor = Integer.parseInt(data);    // 수신 데이터
    // 수신된 값이 범위 이내의 값인 경우
    if(variable_resistor >= 0 && variable_resistor <= 1023) {
        TextView txtView = (TextView) findViewById(R.id.textViewVariableResistor);
        txtView.setText(data);                        // 텍스트뷰에 문자열로 표시
        SeekBar bar = (SeekBar) findViewById(R.id.seekBarVariableResistor);
        bar.setProgress(variable_resistor);            // 시크바 위치 조정
    }
}

```

스케치 1은 블루투스를 통해 데이터를 수신할 수 있도록 스케치 3과 같이 수정한다.

스케치 3. 블루투스를 통한 RGB LED 값 전달

```
#include <SoftwareSerial.h>
```

```
// SoftwareSerial(RX, TX) 형식으로 블루투스 모듈과 교차하여 연결된다.
SoftwareSerial BTSerial(2, 3);
int pin = A0; // 가변저항 연결 핀
int value_previous, value_current;

void setup(){
  Serial.begin(9600); // 시리얼 모니터 초기화
  BTSerial.begin(9600); // 블루투스 모듈 초기화

  value_previous = analogRead(pin); // 초기값
  sendData(value_previous);
}

void sendData(int value){ // 블루투스 모듈을 통해 데이터 전송
  String message = String(value) + '\n';
  BTSerial.print(message);
}

void loop(){
  value_current = analogRead(pin); // 가변저항 값 읽기

  if(value_current != value_previous){ // 가변저항 값이 바뀐 경우에만 전송
    value_previous = value_current;

    Serial.print("Current read is ");
    Serial.print(value_current); // 디지털 값
    Serial.print(", approximately ");
    Serial.print(value_current * 5 / 1023.0); // 아날로그 전압 값
    Serial.println(" V.");

    sendData(value_current); // 블루투스 모듈을 통해 데이터 전송
  }
  delay(1000);
}
```

스케치 3을 아두이노에 업로드 하고 아두이노에 연결된 가변저항을 돌리면 현재 가변저항의 값이 시리얼 모니터에 출력되며, 스마트폰에는 텍스트로 표시됨과 동시에 탐색바의 위치가 이 동함을 확인할 수 있다.