

Chapter 260. 모터 제어

1. 아두이노

스마트폰과 아두이노를 블루투스로 연결하고 스마트폰으로 아두이노에 연결된 DC 모터를 제어하는 어플리케이션을 만들어 보자. 모터 제어를 위해 그림 1과 같이 DC 모터를 모터 제어 모듈을 사용하여 아두이노에 연결한다.

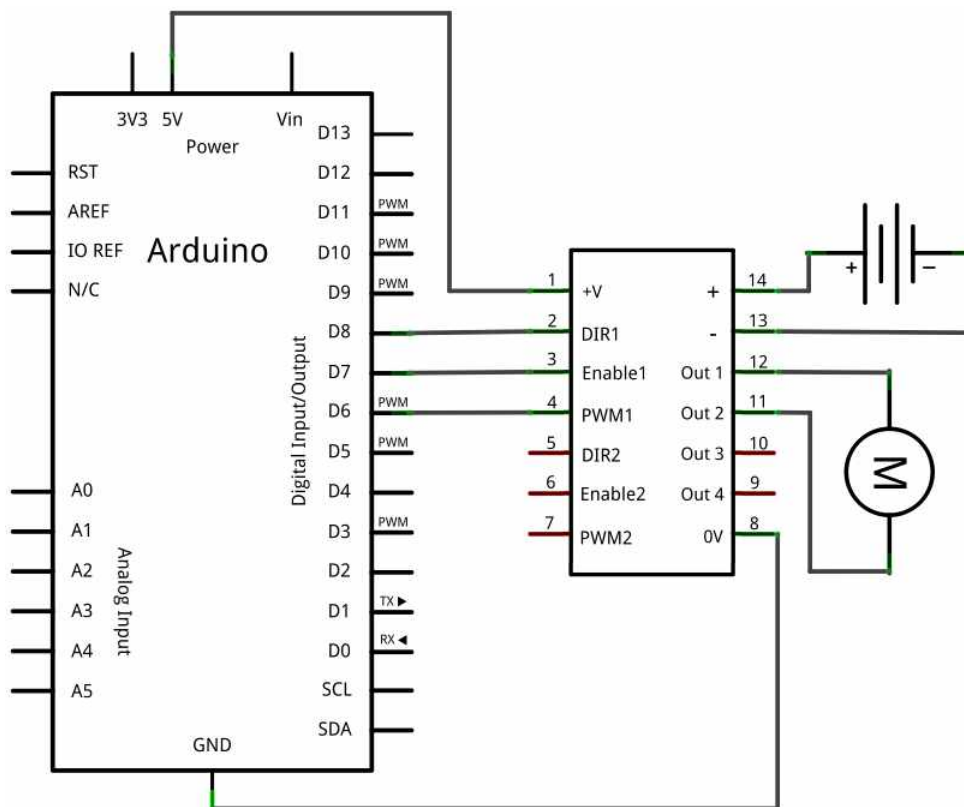


그림 1. 모터 연결 회로도

모터 제어를 위해서는 아두이노의 디지털 출력 핀으로 제공하는 전력 이상의 전력이 필요한 경우가 대부분이며 방향 전환이 필요하므로 전용 모터 제어 모듈을 사용하는 것이 일반적이다. Enable 핀은 연결된 모터를 활성화시키기 위해 사용되며 LOW가 가해진 경우에만 모터를 제어할 수 있고 HIGH가 가해진 경우에는 모터가 정지한다. DIR 핀은 전진 또는 후진을 제어하기 위해 사용되며, PWM 핀은 PWM 신호 출력을 통해 모터의 속도 제어를 위해 사용된다.

스케치 1은 연결된 모터를 정방향으로 서서히 속도를 높이다가 정지하고, 다시 역방향으로 서서히

스케치 1. 모터 제어

```
// 모터 1 제어를 위한 연결 핀
int PWM1 = 6;                // 속도 제어
int Enable1 = 7;             // 활성/비활성 상태 제어
int DIR1 = 8;                // 회전 방향 제어

void setup(){
    // 모터 제어를 위한 핀들을 출력으로 설정
    pinMode(PWM1, OUTPUT);
    pinMode(Enable1, OUTPUT);
    pinMode(DIR1, OUTPUT);

    digitalWrite(Enable1, HIGH);    // 초기 상태는 비활성화 상태
}

void loop(){
    digitalWrite(Enable1, LOW);    // 모터 1 활성화

    digitalWrite(DIR1, HIGH);      // 정방향 회전
    for(int i = 0; i < 256; i++){  // 속도 조절을 위한 PWM 신호 출력
        analogWrite(PWM1, i);
        delay(25);
    }
    analogWrite(PWM1, 0);          // 모터 정지

    delay(1000);

    digitalWrite(DIR1, LOW);       // 역방향 회전
    for(int i = 0; i < 256; i++){  // 속도 조절을 위한 PWM 신호 출력
        analogWrite(PWM1, i);
        delay(25);
    }
}
```

```
analogWrite(PWM1, 0);           // 모터 정지

digitalWrite(Enable1, HIGH);     // 모터 1 비활성화

while(true);
}
```

2. UI 디자인

아두이노에 모터 제어 모듈을 통해 모터 연결이 완료되었으니 어플리케이션의 UI를 만들어보자. 안드로이드 스튜디오에서 새 프로젝트를 선택하고 프로젝트 이름은 'Bluetooth_Motor'로 하자. 프로젝트 생성을 위한 옵션은 디폴트값을 사용한다. 5개의 버튼을 만들고 '전진' 또는 '후진' 버튼이 눌려지면 최대 속도의 50%로 회전하도록 하고, '가속' 또는 '감속' 버튼이 눌려지면 10%씩 속도를 증가 또는 감소하도록 한다. 5개의 버튼은 상대 레이아웃을 사용하여 화면 중앙에 배치한다.



그림 2. 모터 제어를 위한 버튼 배치

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" >
```

```
<Button
    android:id="@+id/button_stop"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:paddingTop="40dp"
    android:paddingBottom="40dp"
    android:text="정지" />

<Button
    android:id="@+id/button_increase"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:paddingTop="40dp"
    android:paddingBottom="40dp"
    android:text="가속"
    android:layout_below="@+id/button_forward"
    android:layout_toRightOf="@+id/button_stop"
    android:layout_toEndOf="@+id/button_stop" />

<Button
    android:id="@+id/button_decrease"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:paddingTop="40dp"
    android:paddingBottom="40dp"
    android:text="감속"
    android:layout_below="@+id/button_forward"
    android:layout_toLeftOf="@+id/button_stop"
    android:layout_toStartOf="@+id/button_stop" />

<Button
    android:id="@+id/button_forward"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:paddingTop="40dp"
```

```

        android:paddingBottom="40dp"
        android:text="전진"
        android:layout_above="@+id/button_stop"
        android:layout_alignLeft="@+id/button_stop"
        android:layout_alignStart="@+id/button_stop" />

<Button
    android:id="@+id/button_backward"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:paddingTop="40dp"
    android:paddingBottom="40dp"
    android:text="후진"
    android:layout_below="@+id/button_stop"
    android:layout_alignLeft="@+id/button_stop"
    android:layout_alignStart="@+id/button_stop" />

</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" />

```

먼저 버튼이 눌러진 경우 이벤트를 처리하기 위해 MainActivity 클래스가 OnClickListener를 구현하도록 하자. OnClickListener를 구현하면 onClick 멤버 함수를 반드시 구현하여야 하므로 onClick 함수 정의 역시 추가하자. 다음은 onCreate 함수에서 버튼의 이벤트 처리를

MainActivity 클래스가 담당하도록 지정해 주어야 한다.

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

    Button btn;
    btn = (button)findViewById(R.id.button_backward);
    btn.setOnClickListener(this);
    btn = (button)findViewById(R.id.button_forward);
    btn.setOnClickListener(this);
    btn = (button)findViewById(R.id.button_stop);
    btn.setOnClickListener(this);
    btn = (button)findViewById(R.id.button_increase);
    btn.setOnClickListener(this);
    btn = (button)findViewById(R.id.button_decrease);
    btn.setOnClickListener(this);

    checkBluetooth();
}
```

버튼이 눌리면 MainActivity 클래스의 onClick 함수가 호출되므로 onClick 함수에서는 현재 눌려진 버튼에 대한 정보를 블루투스를 통해 아두이노로 전달하면 된다.

```
@Override
public void onClick(View v) {
    String str = new String();

    switch(v.getId()){
        case R.id.button_backward:
            str = "1";
            break;
        case R.id.button_forward:
```

```
        str = "2";
        break;
    case R.id.button_stop:
        str = "3";
        break;
    case R.id.button_increase:
        str = "4";
        break;
    case R.id.button_decrease:
        str = "5";
        break;
    }

    sendData(str);
}
```

어플리케이션을 스마트폰에 설치하고 아두이노에는 스케치 3을 업로드 하자. 스케치 3은 블루투스를 통해 수신된 데이터로 모터를 제어하는 스케치의 예이다.

스케치 3. 원격 모터 제어

```
#include <SoftwareSerial.h>

// SoftwareSerial(RX, TX) 형식으로 블루투스 모듈과 교차하여 연결된다.
SoftwareSerial BTSerial(2, 3);

int PWM1 = 6;           // 모터 1 제어를 위한 연결 핀
int Enable1 = 7;
int DIR1 = 8;

int current_speed = 0;

void setup(){
    // 모터 제어를 위한 핀들을 출력으로 설정
    pinMode(PWM1, OUTPUT);
```

```
pinMode(Enable1, OUTPUT);
pinMode(DIR1, OUTPUT);

digitalWrite(Enable1, LOW);           // 활성화 상태
digitalWrite(DIR1, HIGH);             // 전진
analogWrite(PWM1, 0);                 // 속도 0

Serial.begin(9600);
BTSerial.begin(9600);
}

void loop(){
  if(BTSerial.available()){
    byte data = BTSerial.read();
    if(data >= '1' && data <= '5'){
      switch(data){
        case '1':           // 후진
          Serial.println("** Backward...");
          digitalWrite(DIR1, LOW);
          break;
        case '2':           // 전진
          Serial.println("** Forward...");
          digitalWrite(DIR1, HIGH);
          break;
        case '3':           // 정지
          Serial.println("** Stop...");
          analogWrite(PWM1, 0);
          break;
        case '4':           // 속도 10% 증가
          current_speed += 10;
          if(current_speed > 100)           // 최대 속도는 100%
            current_speed = 100;
          Serial.println(String("** Increase speed to ") + current_speed + "%.");
          analogWrite(PWM1, current_speed * 255 / 100);
          break;
        case '5':           // 속도 10% 감소
```

```
        current_speed -= 10;
        if(current_speed < 10)           // 최소 속도는 10%
            current_speed = 10;
        Serial.println(String("** Decrease speed to ") + current_speed + "%.");
        analogWrite(PWM1, current_speed * 255 / 100);
        break;
    }
}
}
```

어플리케이션에서 버튼을 눌러 모터를 제어해 보자.



그림 3. 스케치 3 실행 결과