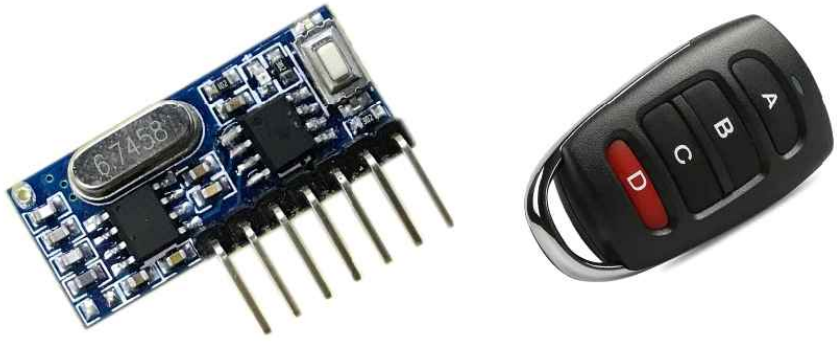
APPENDIX 15

무선 RF 통신

필요한 부품

- 아두이노 우노
- 433MHz RF 수신기
- RF 리모컨
- 피에조 버저
- 220Ω 저항
- 릴레이
- LED

[그림 1]은 433MHz 주파수를 사용하는 리모컨과 수신기다. 수신기는 3.3~5V 전원에 연결하여 사용할 수 있고, 보안 기능이 있어 수신기에 등록한 리모컨만 사용할 수 있다.



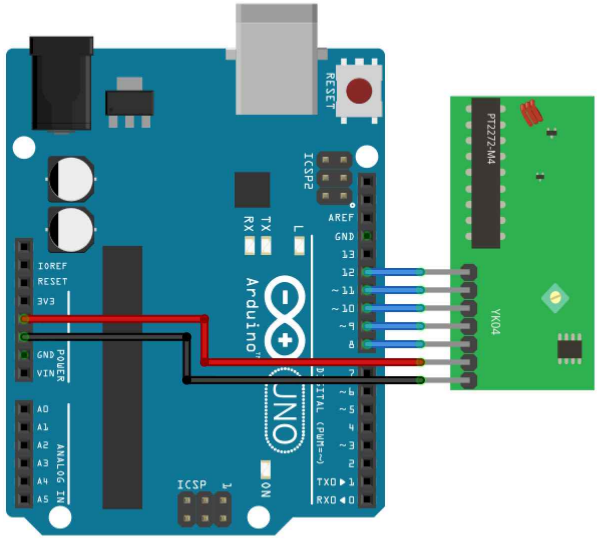
[그림 1] RF 수신기와 리모컨(433MHz)

[그림 1]의 리모컨에는 4개의 버튼이 있으며 각 버튼은 수신기의 D0(리모컨 D 버튼), D1(리모컨 C 버튼), D2(리모컨 B 버튼), D3(리모컨 A 버튼) 핀의 출력을 제어하기 위해 사용된다. 수신기에서 'VT'는 D0에서 D4 핀 중 하나가 HIGH일 때 HIGH가 출력된다.

리모컨으로 수신기의 해당 핀 출력을 제어하는 모드는 다음 3가지가 있다.

1. **모멘터리(momentary) 모드** : 리모컨의 버튼을 누르고 있는 동안에만 수신기의 해당 핀으로 HIGH가 출력되고 버튼을 누르지 않은 상태에서는 LOW가 출력된다.
2. **셀프락(self-lock) 모드** : 리모컨의 버튼을 누르면 수신기의 해당 핀으로 HIGH가 출력되기 시작한다. 리모컨의 버튼을 뗐다가 다시 누르면 수신기의 해당 핀 출력이 LOW로 바뀐다. 토글(toggle) 모드라고도 한다.
3. **인터락(interlock) 모드** : 수신기의 4개 핀 중 하나는 항상 HIGH를 출력하고 있다. 리모컨의 버튼을 누르면 수신기의 해당 핀으로 HIGH가 출력되면서 다른 3개 핀의 출력은 LOW로 바뀐다.

수신기의 D0, D1, D2, D3, VT 핀을 아두이노의 8, 9, 10, 11, 12번 핀에 [그림 2]와 같이 연결하자.



[그림 2] RF 수신기 연결 회로

이 장에서는 리모컨을 모멘터리 모드로 사용하며 리모컨을 모멘터리(momentary) 모드로 등록하는 방법은 다음과 같다.

1. 학습 버튼을 8번 누른다. 버튼을 누를 때마다 수신기에 있는 LED가 깜빡거리고 버튼을 8번 누르고 나면 LED가 7번 깜빡거린 후 꺼진 상태에 있다.
2. 수신기의 LED가 켜질 때까지 학습 버튼을 1번 길게 눌렀다 뗀다. LED가 꺼졌다가 잠시 후 다시 켜지면 리모컨을 등록할 수 있는 상태가 된다.
3. 수신기의 LED가 켜진 상태에서 리모컨의 임의의 버튼을 누르고 있으면 수신기의 LED가 깜빡거리다 켜진 상태로 바뀌면서 리모컨 등록이 끝난다.

등록이 완료되면 리모컨의 임의의 버튼을 눌러 수신기의 핀 출력을 제어할 수 있다. [코드 1]은 리모컨의 각 버튼을 누르거나 뗐을 때, 즉 핀의 상태가 이전과 달라졌을 때 핀의 상태를 시리얼 모니터로 출력하는 스케치의 예다. HIGH가 입력되는 핀, 즉 리모컨의 해당 버튼이 눌러진 경우는 'O'로 LOW가 입력되는 핀은 '.'로 표시하였다.

코드 1 **리모컨 테스트** APPENDIX_15-1_rf_test.ino

```
int PINS[] = {8, 9, 10, 11, 12};        // 수신기 연결 핀
// 수신기 핀의 출력 상태
boolean states[] = {false, false, false, false};
boolean state_change = false;           // 수신기 핀의 출력 변화 여부

void setup() {
  for (int i = 0; i < 5; i++) {         // 수신기 핀이 연결된 핀을 입력으로 설정
    pinMode(PINS[i], INPUT);
  }

  Serial.begin(9600);                   // 시리얼 연결 초기화
}

void loop() {
  state_change = false;

  for (int i = 0; i < 4; i++) {
    boolean current_state = digitalRead(PINS[i]);      // 수신기 핀의 출력을 검사

    // 이전 상태와 다른 경우, 즉 리모컨 버튼이 눌린 경우
    if (current_state != states[i]) {
      state_change = true;              // 상태 변화 플래그 설정
      states[i] = current_state;        // 현재 상태 설정
    }
  }

  // 4개 리모컨 버튼 중 하나라도 상태가 변한 경우에만 시리얼 모니터로 상태 출력
  if (state_change) {
    for (int i = 0; i < 4; i++) {
      Serial.print(states[i] ? 'O' : '.' );
```

```
      Serial.print('\t');
    }
    Serial.println();
  }
}
```

[그림 3] 스케치 1 실행 결과

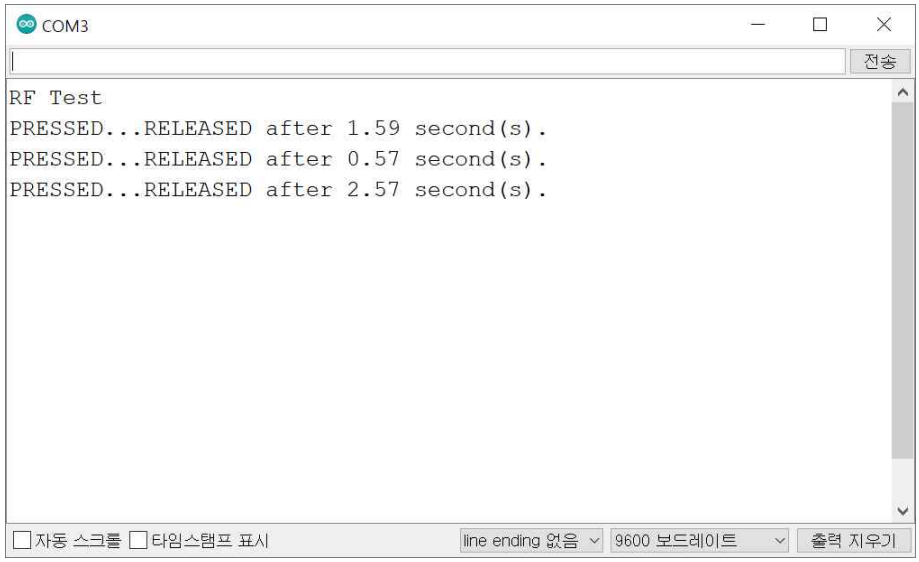
리모컨의 버튼이 눌러진 시간을 측정하여 출력해 보자. 리모컨에서는 A 버튼을 누르는 것으로 하고 아두이노의 11번 핀 상태를 검사하여 버튼이 눌러진 시간을 계산한다. [코드 2]는 리모컨의 A 버튼을 누른 시간을 출력하는 스케치의 예다.

코드 2 **리모컨 수신 시간** APPENDIX_15-2_push_time.ino

```
// 모멘터리 형식의 RF 수신기가 사용된 것으로 가정한다.
// 모멘터리 형식 RF 수신기는 푸시 버튼과 유사한 방식으로 동작한다.
// 리모컨의 버튼이 눌린 상태이면 수신기 모듈의 D3 핀은 HIGH 상태가 되고
// 리모컨의 버튼이 눌리지 않은 상태이면 수신기 모듈의 D3 핀은 LOW 상태가 된다.

// 수신기 모듈의 "D3" 핀이 연결된 아두이노 입출력 핀 번호
const int TRIGGER_PIN = 11;
```

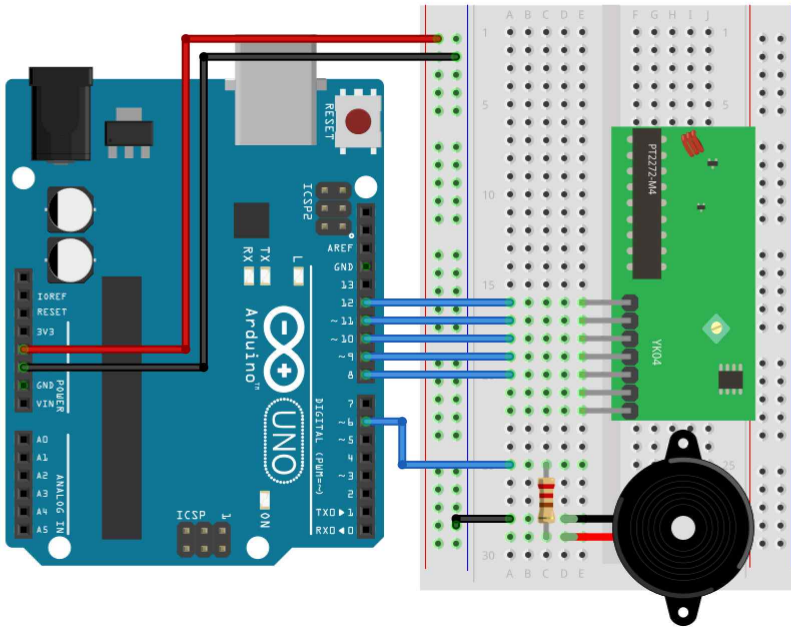
```
// 리모컨 A 버튼이 눌렸을 때 켜지는 LED가 연결된 핀 번호
const int LED_PIN = 13;

// 버튼이 눌리기 시작한 시간 저장을 위한 변수
unsigned long start_time;

// 버튼이 눌리기 시작했을 때 메시지 출력을 위한 변수
boolean announced;

void setup() {
  Serial.begin(9600);
  Serial.println("RF Test");

  pinMode(LED_PIN, OUTPUT);             // LED 연결 핀을 출력으로 설정
}

void loop() {
  digitalWrite(LED_PIN, LOW);           // 버튼이 눌리지 않았을 때 LED 끔
  announced = false;                    // 현재 버튼은 눌리지 않은 상태에 있음

  // 버튼이 눌린 동안은 while 루프에 머무름
  while (digitalRead(TRIGGER_PIN)) {
    // 버튼이 눌리기 시작할 때 한 번만 동작
    if (!announced) {
      start_time = millis();
      Serial.print("PRESSED...");
      announced = true;                 // 버튼을 누르기 시작할 때 한 번만 메시지 출력
    }
    digitalWrite(LED_PIN, HIGH);
  }

  // 버튼을 눌렀다 뗄 때 버튼을 누른 시간을 출력
  if (announced) {
    Serial.print("RELEASED after ");
    unsigned long duration = millis() - start_time;
    // 소수점 이하 2자리의 초 단위로 버튼을 누른 시간 출력
```

```
    Serial.print(duration / 1000.0);
    Serial.println(" second(s).");
  }
}
```



[그림 4] 코드 2 실행 결과

[그림 5]와 같이 아두이노 우노의 6번 핀에 피에조 버저나 스피커를 연결한다. [코드 3]은 리모컨의 A 버튼을 누르면 연결한 피에조 버저나 스피커로 멜로디를 재생하는 스케치다. 스케치를 업로드하고 리모컨의 A 버튼을 눌러 멜로디가 재생되는 것을 확인해 보자.



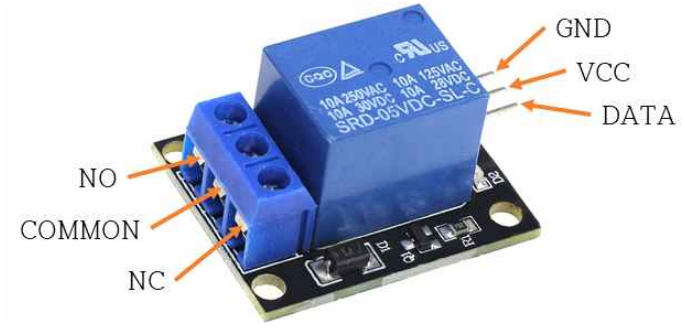
[그림 5] RF 수신기와 피에조 버저 연결 회로

코드 3 **RF 통신을 이용한 초인종** APPENDIX_15-3_doorbell.ino

```
// 모멘터리 형식의 RF 수신기가 사용된 것으로 가정한다.
// 모멘터리 형식 RF 수신기는 푸시 버튼과 유사한 방식으로 동작한다.
// 리모컨의 버튼이 눌린 상태이면 수신기 모듈의 D3 핀은 HIGH 상태가 되고
// 리모컨의 버튼이 눌리지 않은 상태이면 수신기 모듈의 D3 핀은 LOW 상태가 된다.

#include "pitches.h"                    // 음높이를 정의하는 헤더 파일 포함

// 수신기 모듈의 "D3" 핀이 연결된 아두이노 입출력 핀 번호
const int TRIGGER_PIN = 11;

const int SPEAKER = 6;                  // 스피커 연결 핀

// 재생할 멜로디의 음 높이 배열
int notes[] = {
  NOTE_A4, NOTE_E3, NOTE_A4, 0,
  NOTE_A4, NOTE_E3, NOTE_A4, 0,
  NOTE_E4, NOTE_D4, NOTE_C4, NOTE_B4, NOTE_A4, NOTE_B4, NOTE_C4, NOTE_D4,
  NOTE_E4, NOTE_E3, NOTE_A4, 0
};
```

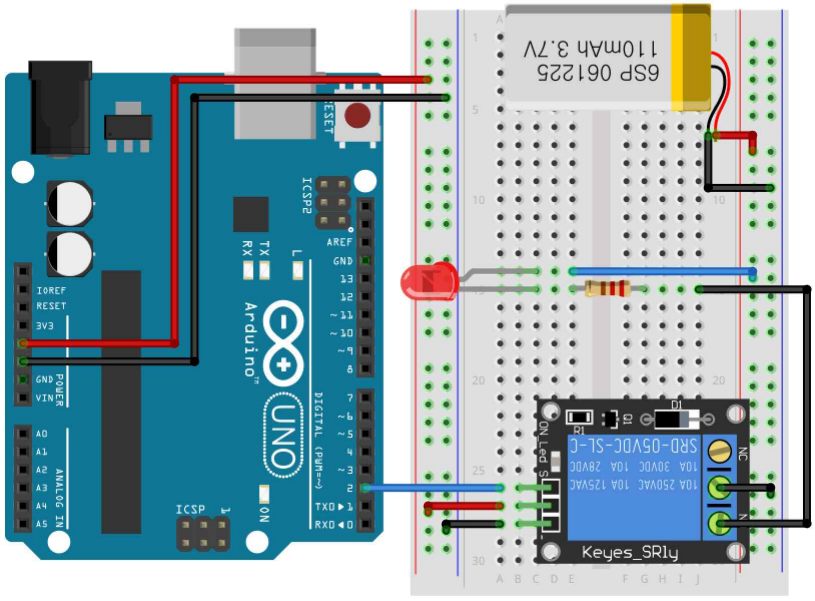
```
// 밀리초 단위의 음길이 배열
int times[] = {
  250, 250, 250, 250,
  250, 250, 250, 250,
  125, 125, 125, 125, 125, 125, 125, 125,
  250, 250, 250, 250
};

void setup() {
  // 초기화는 필요하지 않다.
}

void loop() {
  // 버튼이 눌려 아두이노 핀으로 HIGH가 입력되면 멜로디를 한 번 재생한다.
  if (digitalRead(TRIGGER_PIN)) {
    for (int i = 0; i < 20; i++) {
      tone(SPEAKER, notes[i], times[i]);
      delay(times[i]);
    }
    // 멜로디 재생이 끝난 후에도 버튼이 눌린 상태에 있으면
    // 버튼을 뗄 때까지 기다려 멜로디가 연속해서 재생되지 않도록 한다.
    while (digitalRead(TRIGGER_PIN));
  }
}
```

릴레이는 스위치의 일종으로 아두이노 우노의 5V 신호로 가전제품에 사용되는 220V 전원을 스위칭할 수 있다. [그림 6]은 이 장에서 사용하는 릴레이를 나타낸다.



[그림 6] 전기기계식 릴레이

왼쪽의 단자에는 제어하고자 하는 장치의 전원선 2개 중 하나를 잘라 자른 두 선을 연결한다. 이때 COMMON은 공통 연결 단자로 자른 두 개의 선 중 하나를 연결한다. 자른 두 개의 선 중 나머지 하나는 NO나 NC에 연결한다. NO는 Normal Open의 약자로 릴레이가 동작하지 않을 때 열린 상태에 있는 스위치로, 릴레이에 전원을 연결하고 DATA 핀으로 HIGH를 출력하면 닫힌다. 반면 NC는 Normal Close의 약자로 릴레이가 동작하지 않을 때 닫힌 상태에 있는 스위치로, 릴레이에 전원을 연결하고 DATA 핀으로 HIGH를 출력하면 열린다. 오른쪽의 GND와 VCC에는 릴레이 제어에 사용되는 전원을 연결하고 DATA에는 릴레이 개폐를 제어할 신호를 입력한다. [그림 7]과 같이 아두이노 우노에 릴레이를 연결하자. 릴레이로 220V 전원을 제어할 수 있지만, 안전을 위해 여기서는 별도 전원을 사용하는 LED를 제어하는 예를 살펴본다. 연결에서 주의할 점은 아두이노의 전원과 제어하는 LED의 전원은 별개라는 점이다. 즉, LED 전원의 GND와 아두이노의 GND를 서로 연결하지 않는다. [그림 7]에는 나타내지 않았지만, [코드 4]를 테스트하기 위해서는 [그림 5]의 RF 수신기와 피에조 부조 역시 연결된 상태에 있어야 한다.



[그림 7] 릴레이 연결 회로도

[코드 4]는 리모컨으로 LED의 점멸을 제어하고 LED가 켜질 때와 꺼질 때 다른 멜로디가 재생되도록 하는 스케치의 예다.

코드 4 **RF 통신을 이용한 LED 제어** APPENDIX_15-4_lamp_remote.ino

```
// 모멘터리 형식의 RF 수신기가 사용된 것으로 가정한다.
// 모멘터리 형식 RF 수신기는 푸시 버튼과 유사한 방식으로 동작한다.
// 리모컨의 버튼이 눌린 상태이면 수신기 모듈의 D3 핀은 HIGH 상태가 되고
// 리모컨의 버튼이 눌리지 않은 상태이면 수신기 모듈의 D3 핀은 LOW 상태가 된다.

#include "pitches.h"                    // 음높이를 정의하는 헤더 파일 포함

// 수신기 모듈의 "D3" 핀이 연결된 아두이노 입출력 핀 번호
const int TRIGGER_PIN = 11;

const int SPEAKER = 6;                  // 스피커 연결 핀
const int LAMP = 2;                     // LED 제어 핀

// 재생할 멜로디의 음 높이 배열
int notes[] = {NOTE_E3, NOTE_A4, NOTE_C5};
```

```
// 밀리초 단위의 음길이 배열
int times[] = {250, 250, 250};

// LED의 디폴트 상태는 OFF
bool lamp_on = false;

void setup() {
  pinMode(LAMP, OUTPUT);                // LED 제어 핀을 출력으로 설정
  digitalWrite(LAMP, lamp_on);          // LED는 꺼진 상태로 시작
}

void loop() {
  // 버튼이 눌리면 LED의 상태를 바꿈
  if (digitalRead(TRIGGER_PIN)) {
    lamp_on = !lamp_on;                 // LED 제어를 위한 변수 상태 반전
    digitalWrite(LAMP, lamp_on);        // LED를 현재 상태로 설정
    // LED의 상태 변화 방향(ON->OFF 또는 OFF->ON)에 따라 다른 멜로디 재생
    if (lamp_on) {
      // LED가 켜질 때의 멜로디 재생 : 순방향
      for (int i = 0; i < 3; i++) {
        tone(SPEAKER, notes[i], times[i]);
        delay(times[i]);
      }
    }
    else {
      // LED가 꺼질 때의 멜로디 재생 : 역방향
      for (int i = 2; i >= 0; i--) {
        tone(SPEAKER, notes[i], times[i]);
        delay(times[i]);
      }
    }
    // 멜로디 재생이 끝난 후에도 버튼이 눌러진 상태에 있으면 버튼을 뗄 때까지 대기
    // 리모컨 신호에 대한 디바운싱 효과를 얻을 수 있음
    while (digitalRead(TRIGGER_PIN));
  }
}
```