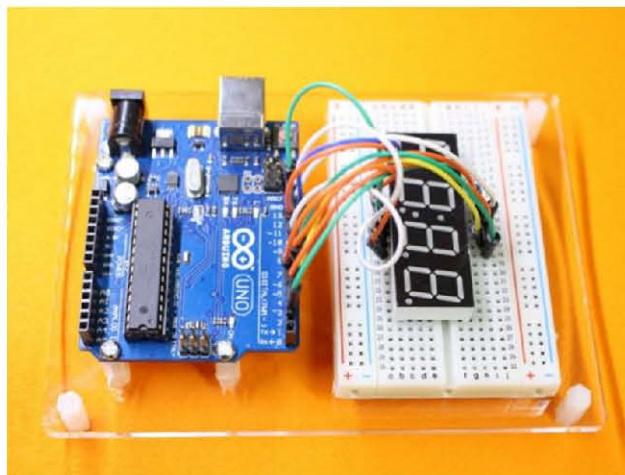


# 아두이노 우노 R3 스타터 최강키트 참고도서

Arduino UNO R3 starter cook book for  
the strongest kit



게임플러스에듀닷컴

<http://www.gameplusedu.com>

## 목차

|        |   |    |
|--------|---|----|
| 1      | > 아두이노란? .....                          | 10 |
| 1.1    | 오픈 소스.....                              | 10 |
| 1.2    | 신개념의 성장기반, 자유와 공유 그리고 오픈 소스, GNU.....   | 11 |
| 1.3    | 쉬운 방식의 프로그래밍 .....                      | 12 |
| 1.4    | 일관된 규칙에서 일어난 커리큘럼.....                  | 13 |
| 1.5    | 오픈 소스의 장점과 단점.....                      | 14 |
| 1.5.1  | 장점 .....                                | 14 |
| 1.5.2  | 단점 .....                                | 14 |
| 2      | 아두이노 우노 R3 보드.....                      | 15 |
| 3      | MCU 기초 정보 .....                         | 17 |
| 3.1    | MCU, MPU, CPU 구분은?.....                 | 18 |
| 4      | ATMEGA328P 기초 정보.....                   | 19 |
| 4.1    | 아두이노의 포트 지정 규칙.....                     | 22 |
| 5      | 아두이노 우노 R3 MCU 보드에 대한 이해 .....          | 24 |
| 5.1    | 하드웨어 통신 I2C & SPI & ICSP 핀맵 기능 요약. .... | 24 |
| 5.1.1  | <b>ICSP 란?</b> .....                    | 25 |
| 5.1.2  | <b>ICSP for USB Interface:</b> .....    | 28 |
| 5.1.3  | <b>ICSP for ATMEGA328P:</b> .....       | 29 |
| 5.1.4  | SPI 포트:.....                            | 31 |
| 5.1.5  | I2C:.....                               | 34 |
| 5.2    | 아두이노 시리얼 통신 .....                       | 36 |
| 5.2.1  | 하드웨어 시리얼 통신. ....                       | 36 |
| 5.2.2  | 소프트웨어 시리얼 통신 .....                      | 39 |
| 5.3    | EEPROM.....                             | 40 |
| 5.4    | ADC (Analog Digital Converter) .....    | 41 |
| 5.5    | AREF (Analog Reference Voltage).....    | 42 |
| 5.6    | PWM 포트.....                             | 44 |
| 5.7    | PWM 포트 이해 .....                         | 45 |
| 5.8    | PWM 사용 예제 .....                         | 47 |
| 5.9    | PWM 직접 구현 예제 코드 .....                   | 49 |
| 5.10   | PWM 포트 사용시 주의점.....                     | 50 |
| 5.11   | 하드웨어 외부 인터럽트 .....                      | 52 |
| 5.11.1 | 내부 인터럽트.....                            | 52 |
| 5.11.2 | 외부 인터럽트 사용 .....                        | 53 |
| 5.12   | DC 입력 7 to 12V DC.....                  | 56 |
| 5.13   | 디지털 포트 & 아날로그 포트 .....                  | 56 |
| 5.13.1 | 디지털 입력/출력 포트.....                       | 57 |
| 5.13.2 | 아날로그 입력 포트 (Analog Input Port).....     | 58 |

|       |                                    |     |
|-------|------------------------------------|-----|
| 5.14  | 디지털 14 핀 & 아날로그 포트 6 포트 사용 정의..... | 61  |
| 6     | 아두이노 보드의 전압, 전류 입, 출력 정보.....      | 62  |
| 7     | 아두이노 시작하기 .....                    | 65  |
| 7.1   | 아두이노 IDE 스케치 프로그램 다운로드.....        | 67  |
| 7.2   | 아두이노 보드 드라이버 설치.....               | 69  |
| 7.3   | 아두이노 드라이버 간편 설치 방법 .....           | 73  |
| 8     | 아두이노 우노 R3 전원 공급 방법 .....          | 74  |
| 8.1   | USB 케이블 연결 .....                   | 74  |
| 8.2   | DC 어댑터 전원 연결.....                  | 75  |
| 8.3   | 9V DC 배터리.....                     | 76  |
| 8.4   | Vin, GND 연결.....                   | 77  |
| 9     | 아두이노 스케치 프로그램 설치 & 설정.....         | 78  |
| 9.1   | 윈도우 탐색기에서 파일 확장자 보이도록 설정 방법 .....  | 81  |
| 9.2   | 스케치 기본 함수 .....                    | 82  |
| 9.3   | 스케치 프로그램 IDE 기본 기능.....            | 84  |
| 9.4   | 보드 종류 & 포트 설정 .....                | 86  |
| 9.4.1 | 시리얼 포트 확인 방법.....                  | 89  |
| 9.5   | 스케치 프로그램 업로드 .....                 | 90  |
| 10    | 아두이노 프로그램 업로드 과정에 대한 이해 .....      | 93  |
| 10.1  | AVR MCU 란?.....                    | 93  |
| 10.2  | AVR MCU 부트로더.....                  | 94  |
| 10.3  | 아두이노 보드의 부트로더 .....                | 94  |
| 11    | 아두이노와 C/C++ 언어 .....               | 96  |
| 12    | 아두이노 IDE 스케치 라이브러리.....            | 97  |
| 12.1  | 스케치 IDE 기본 라이브러리 정보.....           | 97  |
| 12.2  | 아두이노 사용자 라이브러리 설정.....             | 98  |
| 12.3  | 아두이노 IDE 환경 설정 저장 파일 .....         | 99  |
| 13    | 아두이노 스케치 IDE 예제 업로드 & 테스트.....     | 100 |
| 13.1  | 예제 열기.....                         | 100 |
| 14    | 프로그램 빌드 & 업로드.....                 | 102 |
| 15    | 업로드 안 되는 경우 .....                  | 104 |
| 16    | 아두이노 스케치 코드의 main 함수는? .....       | 105 |
| 16.1  | 스케치 IDE 프로그램 소스 코드 분석 .....        | 105 |
| 17    | 아두이노 IDE 하위 디렉터리 정보 .....          | 107 |
| 18    | 아두이노 IDE 라이브러리 추가 및 사용하기.....      | 109 |
| 18.1  | 라이브러리 추가 하기 전 주의사항 .....           | 109 |
| 18.2  | 사용자 라이브러리 디렉터리 (폴더) 확인.....        | 109 |
| 18.3  | 라이브러리 추가 방법 1.....                 | 110 |
| 18.4  | 라이브러리 추가 방법 2.....                 | 111 |
| 19    | 아두블럭 간단한 소개 및 사용법.....             | 113 |

|        |   |     |
|--------|---|-----|
| 19.1   | 아두블럭 설치 방법 .....                              | 113 |
| 20     | 아두이노 프로세싱 .....                               | 117 |
| 20.1   | 프로세싱 프로그램과 아두이노 연동 방식 .....                   | 119 |
| 20.2   | 아두이노 프로세싱의 시리얼 포트 사용시 주의점:.....               | 122 |
| 20.3   | 아두이노와 미니 PC 의 미래 .....                        | 123 |
| 21     | >High quality and large bread board x 1 ..... | 124 |
| 21.1   | 버스영역 .....                                    | 125 |
| 21.2   | IC 영역 .....                                   | 125 |
| 21.3   | 내부 구조.....                                    | 126 |
| 22     | > RFID Module Kit.....                        | 127 |
| 22.1   | > RFID module x 1 .....                       | 128 |
| 22.2   | > IC Keychain x 1 .....                       | 129 |
| 22.3   | > Non-contact type IC card x 1 .....          | 129 |
| 22.4   | 아두이노 보드와 연결 .....                             | 130 |
| 23     | > LCD1602 I2C + IIC Interface Module x 1..... | 140 |
| 23.1   | I2C LCD1602 예제 .....                          | 141 |
| 23.2   | 시리얼로 받은 문자열을 LCD1602 I2C 모듈로 출력 .....         | 142 |
| 24     | LCD1602 디렉트 와이어링 .....                        | 145 |
| 24.1   | 가변저항으로 대비(Contrast) 조절 방법.....                | 145 |
| 24.2   | PWM 포트로 대비 조절방법.....                          | 146 |
| 24.3   | LCD1602 Hello 표시 예제 코드.....                   | 147 |
| 25     | > Opto-couple 1 Channel 5V Relay x 1 .....    | 148 |
| 25.1   | AC 전원선과 릴레이 모듈 연결 방법 .....                    | 151 |
| 25.2   | . 5V 릴레이 테스트 예제 .....                         | 153 |
| 25.2.1 | 릴레이 테스트 코드 – 5초마다 켜고, 끄기.....                 | 154 |
| 25.2.2 | 버튼 누르면 LED 켜지게 하기 .....                       | 155 |
| 25.3   | 릴레이 활용.....                                   | 157 |
| 26     | > DS1302 clock module x 1 .....               | 158 |
| 27     | > Sound Detection Module x 1 .....            | 161 |
| 27.1   | 사운드 센서 모듈 소개.....                             | 161 |
| 27.2   | 기준값 설정.....                                   | 162 |
| 27.3   | 소리 반응 기초 프로그래밍하기.....                         | 165 |
| 27.4   | 소리 반응 LED 켜기와 끄기 .....                        | 166 |
| 27.5   | 소리 반응 LED 반응 PWM 사용하기.....                    | 167 |
| 28     | > Temperature and humidity module x 1.....    | 172 |
| 28.1   | DHT11 온도, 습도 센서 모듈 .....                      | 172 |
| 28.2   | DHT11 센서 .....                                | 176 |
| 29     | > Level detection module x 1 .....            | 180 |
| 29.1   | 여러 개의 아날로그 센서 사용시 주의사항.....                   | 183 |
| 30     | > 4*4 keypad module x 1 .....                 | 184 |

|        |   |     |
|--------|---|-----|
| 31     | > Three-color RGB module x 1 .....      | 188 |
| 31.1   | RGB LED 모듈 형태가 아닌 부품 사용.....            | 189 |
| 31.1.1 | CATHODE (캐소드) Type.....                 | 189 |
| 31.1.2 | ANODE (애노드) TYPE .....                  | 192 |
| 32     | > XY joystick x 1.....                  | 195 |
| 33     | > Servo x 1 .....                       | 199 |
| 33.1   | 서보 모터 전원 공급 참조 .....                    | 204 |
| 34     | > Stepper motor & driver board x 1..... | 205 |
| 34.1   | 스케치 예제 코드 1.....                        | 208 |
| 34.2   | 스케치 예제 코드 2.....                        | 211 |
| 35     | > RED, Green, Yellow LED .....          | 215 |
| 36     | LED 종류와 전압 사용 예제.....                   | 215 |
| 37     | 저항 기초 정보.....                           | 218 |
| 38     | 1K 저항 10 개 .....                        | 219 |
| 39     | 10K 저항 10 개 .....                       | 219 |
| 40     | 220 ohm 저항 10 개 .....                   | 219 |
| 41     | 부저의 종류 .....                            | 220 |
| 41.1   | 패시브 부저.....                             | 220 |
| 41.2   | 액티브 부저.....                             | 220 |
| 42     | > Passive Buzzer x 1 .....              | 221 |
| 43     | > Piezo Buzzer x 1 .....                | 226 |
| 44     | > Key module (with hat) x 4.....        | 228 |
| 45     | Pull-up 풀업 저항, Pull-down 풀다운 저항.....    | 232 |
| 45.1   | 플로팅(floating) .....                     | 232 |
| 45.2   | 풀업(Pull-Up) 저항.....                     | 233 |
| 45.3   | 풀다운(Pull-Down) 저항 .....                 | 234 |
| 46     | 풀업, 풀다운 저항 계산 논리.....                   | 235 |
| 46.1   | 풀다운 버튼 예제 (Pull Down Button).....       | 237 |
| 46.2   | 풀 업 버튼 예제 (Pull Up Button).....         | 240 |
| 47     | Input_PullUP, 풀업 버튼 회로 구성.....          | 242 |
| 48     | > Tilt Switch x 2 .....                 | 244 |
| 49     | > LM35 sensor module x 1 .....          | 249 |
| 50     | > Photo Resistor x 3 .....              | 251 |
| 51     | > Fire x 1 (Flame Sensor) .....         | 254 |
| 51.1   | Flame Sensor Module Type .....          | 255 |
| 51.2   | Flame Sensor 직접 회로 구성 .....             | 256 |
| 52     | > Infrared receiver x 1 .....           | 261 |
| 52.1   | IR 수신 라이브러리 사용시 주의점.....                | 263 |
| 52.2   | IR 수신 모듈 와이어링 .....                     | 265 |
| 53     | > 인체 감지 센서 모듈 x 1 .....                 | 271 |

|        |  |     |
|--------|--|-----|
| 54     | > Adjustable potentiometer x 1 .....         | 276 |
| 55     | > A digital control x 1 .....                | 279 |
| 56     | > 4 digital tube x 1, 4-Digit 7-Segment..... | 288 |
| 57     | > 8 * 8 dot matrix module x 1 .....          | 294 |
| 58     | > 74HC595N IC x 1 .....                      | 304 |
| 59     | > Infrared remote control x 1.....           | 310 |
| 60     | > Breadboard Jumper x 65 .....               | 310 |
| 61     | > Female to Male DuPont lines x 10.....      | 311 |
| 62     | > 9 volt battery snap x 1.....               | 311 |
| 63     | > USB Cable x 1 .....                        | 312 |
| 64     | > HC-SR04 x 1 초음파 센서 .....                   | 313 |
| 64.1   | Direct Trig, Echo 사용 .....                   | 315 |
| 64.2   | pulseIn 함수 사용 설명 .....                       | 316 |
| 64.3   | NewPing 라이브러리 사용 예제 .....                    | 316 |
| 65     | > 블루투스 HC-06 슬레이브 모듈 .....                   | 317 |
| 65.1   | 블루투스 소개.....                                 | 317 |
| 65.2   | 블루투스 모듈 아두이노 .....                           | 317 |
| 65.3   | 블루투스 연결 사용 방법.....                           | 318 |
| 65.3.1 | 블루투스 페어링(Pairing).....                       | 318 |
| 65.3.2 | 하드웨어 시리얼 포트 사용.....                          | 319 |
| 65.3.3 | 소프트웨어 시리얼 블루투스 통신.....                       | 320 |
| 65.4   | 스마트폰 연동.....                                 | 322 |
| 65.5   | PC 연동 .....                                  | 324 |
| 65.6   | 블루투스 모듈 이름 변경.....                           | 327 |
| 66     | 무선통신 RF 433/315 MHz 송/수신 장치 .....            | 332 |
| 67     | 아두이노 네트워크 프로그래밍.....                         | 337 |
| 68     | 웹브라우저에서 아두이노 LED 제어하기 .....                  | 342 |
| 69     | SD 카드 읽기/쓰기 .....                            | 347 |
| 70     | Breadboard Power Board 2 Load 3V3, 5V .....  | 352 |
| 71     | 아두이노 지그비 S2 통신.....                          | 354 |
| 71.1   | XBee 소개.....                                 | 354 |
| 71.2   | 지그비 무선 통신 규약.....                            | 355 |
| 71.3   | 지그비 디바이스 설정 .....                            | 355 |
| 71.4   | XBee S2 소개 .....                             | 357 |
| 71.5   | 지그비 S2 설정 .....                              | 359 |
| 71.5.1 | 지그비 S2 USB Adaptor 연결 .....                  | 359 |
| 71.6   | 지그비 S2 설정 방법 1 X-CTU 사용.....                 | 361 |
| 71.7   | 지그비 S2 설정 방법 2.....                          | 371 |
| 71.8   | 아두이노 지그비 실드 & 블루투스 실드.....                   | 375 |
| 72     | 최강 키트 구성 항목입니다. .....                        | 377 |

|      |                             |     |
|------|-----------------------------|-----|
| 72.1 | 최강 키트.....                  | 377 |
| 72.2 | > 최강 키트 부품 보관 상자 .....      | 379 |
| 73   | 최강 키트 플러스 구성 항목입니다.....     | 380 |
| 74   | 최강 키트 얼티밋 플러스 구성 항목입니다..... | 380 |

## 1 > 아두이노란?

아두이노는 “오픈 소스 기반의 하드웨어와 소프트웨어 개발 통합 환경” 입니다.

아두이노는 하드웨어와 소프트웨어 오픈 소스 기반에서 출발하여 현재까지 오랜 시간 많은 분야에서 활용 되고 있습니다.

공식 아두이노 사이트는 <http://www.arduino.cc> 입니다.

<http://www.arduino.cc> & <http://playground.arduino.cc> 에는 아두이노 프로젝트 진행에 도움이 되는 많은 자료가 있으므로 자주 방문을 하시기 바랍니다.

아두이노를 활용한 기초부터 고급 기술, 응용까지 자세히 설명 되어 있습니다.

아두이노와 스케치 IDE, 아두이노 모듈, 실드, 부품등을 사용하기 위해서는 자주 방문 하여야 할 사이트입니다.

아두이노 통합 환경의 하드웨어 형태로는 아두이노 우노 / 메가 2560 / 나노 / 레오나르도 / 프로미니 / 두에 등의 다양한 아두이노 보드가 있습니다. 최근에도 신규 아두이노 MCU 보드와 다양한 칩셋을 사용하는 보드가 출시 되고 있습니다.

소프트웨어는 대표적으로 “아두이노 IDE 스케치 프로그램” 있습니다.

IDE(Integrated Development Environment)라는 용어의 의미는 통합개발환경을 말합니다.

### 1.1 오픈 소스

아두이노 하드웨어와 소프트웨어 프로그램 모두 오픈 소스입니다.

아두이노와 소프트웨어를 활용한 많은 관련 결과물들을 인터넷에서 찾아 보고 활용할 수 있습니다.

여러 개인, 단체, 기업에서 아두이노를 활용한 많은 분야의 결과물들이 나와 있는 상태입니다.

특히, 흥미로운 부분은 아두이노 보드를 똑같이 만들어서 상업적인 이용도 가능하게 되어 있습니다. 다만 로고와 이름을 제외하고는 모든 것이 공개되어 있습니다.

위에서도 언급 되었듯이 아두이노의 모든 하드웨어는 기초 설계도를 포함하여 모두 오픈 소스입니다. 아두이노 소프트웨어도 모두 오픈 소스입니다.

물론 오픈 소스라고 쉽지는 않습니다. 오픈 소스라고 누구나 사용하지는 않습니다. 방치됨과 잊히는 것들이 너무나도 많습니다.

특이하게도 아두이노를 활용하여 탑재한 여러 프로젝트는 여러 분야에서 사용되고 끊임없이 재생산되고 있습니다.

인터넷과 오프라인에는 끊임없이 결과물들이 생산됨과 동시에 또 다른 누군가에 의해 발전되고 다양한 용도로의 변형이 일어나고 있습니다.

이 모든 현상의 중심에는 “쉽게 누구나 배울 수 있다”, “사용하기 편하다” 는 장점이 있습니다.

사용하기 편하면서 초급, 중급, 고급의 다양한 개발까지 할 수 있어 사용 범위가 넓습니다.

## 1.2 신개념의 성장기반, 자유와 공유 그리고 오픈 소스, GNU

지금 이순간에도 인터넷에는 새로운 아두이노를 활용하여 개인, 단체, 기업에서 제품들이 나오고 있습니다. 물론 기존의 MCU 보드로 개발되었던 제품들과 프로젝트들도 아두이노로 구현 되었을 경우에는 새로운 주목을 받고 있습니다. 주목을 받는 많은 이유중의 하나는 비상업적인 용도로 개인, 단체에서 공개, 공유 한다는 점입니다.

그 중 가장 대표적인 부분은 RepRap 회사의 3D 프린터가 있습니다.

현재의 3D 프린터의 개념과 솔루션은 이미 발표된 시점이 1980년대 초반부터 나왔습니다.

1990년대 초반부터는 특히 기술이 해제 되면서 RepRap 사에서의 오픈 소스로의 전환은 많은 변화의 초석이 되었습니다. 열가소성 재질을 사용한 압출 적층형 방식의 조형제작 기계, 3D 프린터입니다.

그 중 대표적인 회사가 RepRap 회사였습니다. RepRap은 누구나 3D 프린터를 제작 사용 할 수 있도록, 제어보드, 부품, 설명, 구동 프로그램 소스까지 모두 오픈 소스로 개방 하였습니다. 그 후에 여러 과정을 거쳐 아두이노를 사용한 3D 프린터 기계가 생기면서 엄청난 사용자층이 폭발적으로 생기게 되었습니다.

지금은 전세계인들이 사용하게 됩니다.

아두이노라는 것이 있었기에 가능한 일입니다.

아두이노 보드가 3D 프린터 제어 보드로 사용되면서 사용자끼리의 3D 프린터 자체 개발력이 증가되었다고 보여집니다.

아두이노와 3D 프린터 파급력의 모든 출발점은 공유하면서 상세히 설명하여 준다는 것입니다. 누구나 설명서를 보면서, 또는 인터넷 등의 정보를 참조하여 찾아가면서 만들 수 있다는 것입니다. 습득된 정보를 기초로 다시 새로운, 좀더 발전된 개념의 기기가 창조 되고 있습니다.

위의 모든 기본 개념은 모든 성장 기반은 자유와 공유에 있다는 것입니다.

공유하기 위해서는 단순히 문서, 코드(?)를 보여주는 것이 아닌, 어떤 방식으로든 다른 사람들에게 상세히 설명까지 해준다는 것입니다.

진정한 공유를 실천한다고 보여집니다.

아두이노 사용자들이 단순히 아두이노 보드만 사용할까요? 아닙니다.

아두이노와 관련된 주변기기도 같이 사용합니다. 각종 센서 모듈, 디지털 주변 기기를 사용하게 되어 있습니다.

그럼 각종 센서 모듈은 한 개만 사용하는 것이 아니라 용도에 따라 여러 개를 사용합니다. 다른 기능을 가진 주변 모듈은 몇 배 이상의 개수로 사용되고 있습니다..

해당 센서, 모듈들은 개인들이 아닌 대부분 제조사에서만 사용되던 부품들인데, 일반 개인들에게 팔리고 있습니다. 하드웨어 부품 및 제조 분야의 제 2 의 부흥기가 이루어 진다고 보여집니다. 물론 이러한 현상들은 하드웨어와 동반되는 소프트웨어의 공급 및 수요도 기하급수적으로 증가할 것으로 예상됩니다. 한 개, 두 개, 여러 개의 하드웨어 부품 조합을 바탕으로 소프트웨어(정확히 말하자면 펌웨어 & 응용 프로그램 정도) 개발 방향, 소비자의 요구에 따라 무궁무진하게 변화될 수 있습니다. 변화는 창조의 한 분야입니다.

또한 위의 같은 현상들은 미래의 새로운 경제 발전의 기초적인 토대가 되리라 예상됩니다. 보다 더 진보된 제조, 생산, 소비, 새로운 형태의 분야가 창출되기 위한 토대입니다. 한마디로 대단한 성장기반이라고 보여집니다.

이에 발맞추어 해외 및 국내의 대 자본에서도 GNU 에 대한 새로운 인식이 일어나고 있는 추세입니다. GNU 와 멀티 플랫폼에 대한 새로운 이해도가 증가되고 있습니다. GNU 가 그만큼 규모가 커졌다고 볼 수도 있겠지만, GNU 자체의 내장된 발전 원동력은 누구의 혼자 힘에서 나오는 것이 아닌 공유와 실천에 있다는 것을 알게 될 것입니다. 미래의 일은 누구도 정확히 예측할 수 없지만, 자유와 공유는 후퇴가 아닌 발전한다는 것은 누구나 예측 할 수 있겠습니다

물론 성장이 이루어지는 가운데 대자본의 독점적인 투자가 일어나서 더욱 발전된 방향으로 이루어 질 수도 있습니다. 하지만 여기에 큰 문제점들이 있습니다. 그 중에 하나는 GNU 는 분기를 한다는 것입니다. 가령 A 항목을 누가 독점 하여 A 를 소멸시키고 B 를 만들었습니다. 하지만 B 자체도 A 의 성격을 기본적으로 가지고 있습니다. 하지만 A 는 또 누군가에 의해 다른 형태로 변형 및 발전되게 되어 있습니다. 그 누군가는 B 도 사용하겠지만, A 도 필요할 이유이겠습니다.

### 1.3 쉬운 방식의 프로그래밍

기존의 AVR 프로그래밍과 ISP 방식은 번거롭습니다. 설정할 것도 많고, 테스트 보드도 고비용입니다. 테스트 보드도 비싸고 전문 개발자도 필요합니다. 시제품에 앞서서 선행 개발 보드(프로토타입 개발용 보드)도 용도에 따라서는 테스트 하드웨어, 펌웨어 개발 단계에서도 엄청난 비용이 소모되었습니다. 전문 용어의 기술서적과 여러 가지 알아야 할 것이 많습니다.

하지만, 아두이노는 간단히 USB 포트와 연결하여 프로그래밍과 업로드가 가능합니다. 참고할 수 있는 자료도 인터넷과 도서 등이 많이 있어 누구나 쉽게 접근 가능하도록 되어 있습니다. 사용하는 PC 도 본인이 사용하는 윈도우, 맥, 리눅스 여러 OS를 지원하고 있습니다.

또한 아두이노는 어떤 이유에서든 비용이 저렴합니다. 3 만~10 만원 정도면 어렵다고 여겨지던 여러 하드웨어 기반 작동 결과물을 직접 체험 할 수 있고, 응용까지 가능합니다.

즐기는 취미 생활로의 접근이 가능하다는 겁니다.

아두이노와 접목하여 사용 가능한 스타터 키트와 센서, 통신 모듈과 실드 형태로 출시 되는 하드웨어 제품들이 많습니다. 온/오프 라인으로 쉽게 구매해서 사용 가능하게 되어 있습니다.

## 1.4 일관된 규칙에서 일어난 커리큘럼

아두이노는 “일관된 규칙” 있습니다.

아두이노 스케치 IDE라는 통합 개발 환경 프로그램으로 용도에 맞게 여러 종류의 아두이노 하드웨어를 선택하여 일관된 규칙(공통 규칙)하에 만들 수 있게 되어 있습니다.

물론, 기본적인 프로그래밍 구조는 C/C++ 언어를 바탕으로 구축하게 되어 있습니다. 하드웨어적인 접근 방식은 디지털, 아날로그 포트와 지정된 함수 명확히 구분되어 있습니다. D0, D1, D2, ~D13, A0~, setup() 과 loop() 함수 사용 용도만 구분만 하면 됩니다. 디지털, 아날로그 포트에 대한 기본 정의를 공통으로 하고 함수 명칭을 지정하여 용도에 따라 아두이노 보드를 선택하여 바로 적용 할 수 있습니다.

가령, 아두이노 우노에서 작성된 코드는 아두이노 메가 2560, 레오나르도 보드에 쉽게 이식이 가능합니다.

소프트웨어의 가장 큰 장점 중에 하나인 추상화 다형성, 유연성을 접목하여 하드웨어에도 적용되어 있습니다.

이러한 일관된 규칙은 또 다른 일관된 규칙을 만들게 되어 있습니다.

접근하기 쉬우면서 사용자가 증가하는 규칙에는 목적에 따라 자연스럽게 커리큘럼이 빠른 속도로 생성되게 되어 있습니다.

즉, 아두이노 또는 오픈 소스의 이러한 사용자 층이 생기게 되면, 자연스럽게 다형성을 띄게 되는 여러 분야의 지식이 공유 되면서 발전 되게 되어 있습니다.

동시에 여러 사용자들은 목적에 따라 새로운 커리큘럼이 생성되게 되어 있습니다.

현재의 아두이노 관련 플랫폼은 IT 선진국 대한민국을 비롯하여, 미국, 캐나다, 호주, 영국을 비롯한 유럽, 동남 아시아의 교육 선진국에서 아두이노를 기반으로 하는 하드웨어, 소프트웨어 기초 교육을 비롯한, 응용 분야, 예술분야, 취미까지 광범위하게 사용되고 있습니다.

## 1.5 오픈 소스의 장점과 단점

### 1.5.1 장점

GNU 방식으로 배포되는 오픈 소스들의 종류와 항목들은 헤아릴 수 없을 정도로 많습니다. 오픈 소스의 큰 장점은 해당 오픈 소스가 필요한 경우에는 그대로 사용해도 무방합니다. 해당 목적에 부합되는 오픈 소스를 적용하는 경우 상당한 시간 단축과 더불어 비용절감이라는 장점이 있습니다. 상당히 깔끔한 결과로 프로젝트 종료 및 차후 발전되는 단계의 프로젝트까지 구성 할 수 있습니다.

물론, GNU 오픈 소스의 규모에 따라 분석하는 시간이 필요하긴 하지만 최종 마무리 단계까지의 부합되는 오픈 소스 프로젝트일 경우, 시간 대비, 비용 절감은 상상 이상으로 극대화 효과를 볼 수 있습니다.

그리고 되도록 GNU 오픈 소스의 선택 및 사용은 90% 이상 형상화 및 적용된 상태 이거나 10% 내외 적용된 소규모 단위 항목, 라이브러리 개념의 GNU 오픈 소스 적용 시 생각보다는 만족하는 결과를 볼 수 있으리라 예상됩니다.

아키텍처의 극한 이익 발생>>

필요한 IT, ICT 기능을 적시에 사용 가능합니다.

유연성 : 적절하게 필요한 기능을 대체하여 사용 가능합니다.

커뮤니티 : 블로그, 카페, it 포럼 사이트를 통한 빠른 문제 해결이 가능합니다.

해당 it 담당자의 빠른 대처가 가능합니다.

### 1.5.2 단점

장점이 있으면 분명, 단점도 있기 마련입니다. 기존의 GNU 오픈 소스 분석 후 적용 및 마무리 단계에서 최종 프로젝트에 부합되지 못하거나, 미비한 부분을 추가 및 변경해야 할 경우가 있습니다. GNU 오픈 소스의 기존 코드를 변경 또는 추가하는 범위가 만약 조금 커지는 경우, 개발 시간의 비약적인 장기화가 발생 될 수 있습니다. 개발 시간의 장기화는 프로젝트 포기, 또는 방향 변경 등의 좋은 않은 결과가 발생될 수 있습니다. 이러한 리스크는 가져다가 사용하는 측에서 부담해야 되는 부분입니다. 물론 GNU 오픈 소스 개발자(그룹)들의 도움으로 마무리 될 수도 있지만, 쉽지는 않습니다.

기존 GNU 오픈 소스의 변경 및 추가의 영역이 커지는 경우에는 나머지 부분의 코드들을 거의 모두 이해해야 변경 및 추가 가능하기 때문입니다. 결국, 전체 코드에 대한 분석 및 파악 후 변경, 추가가 가능하게 됩니다. 원천적인 GNU 오픈 소스 배포자들이 개발에 투자한 시간 그대로 앉고 가야 합니다. 이러한 부분들만 충분히 판단 후 적용, 프로젝트에 적용하기 전에 좀 더 많은 분석 및 단위 시험 테스트가 필요합니다.

## 2 아두이노 우노 R3 보드

아두이노 우노 R3 보드 입니다.

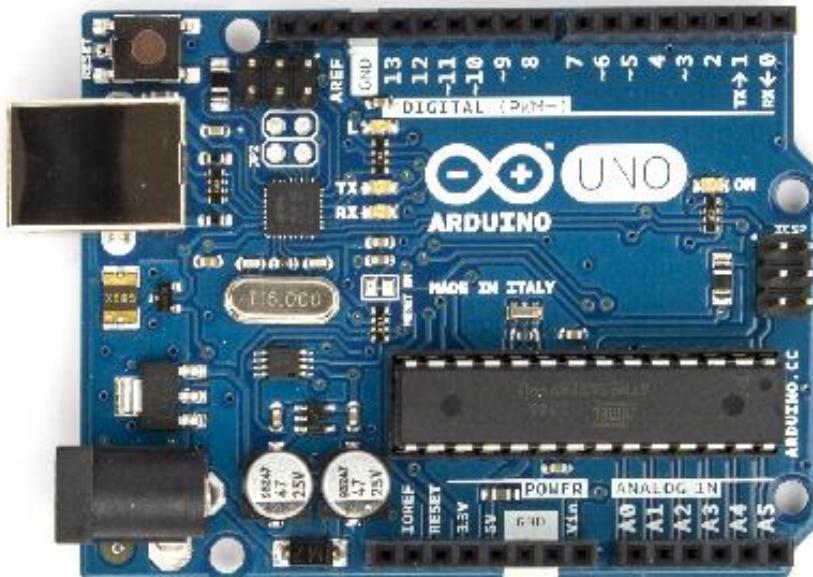


그림 2-1 아두이노 우노 R3

(그림 출처: <http://arduino.cc/en/Main/ArduinoBoardUno>)

아두이노 여러 종류의 보드 중 가장 많이 사용되고 있습니다.

아두이노 보드의 기본 기능은 MCU 보드의 역할을 하고 있습니다.

일종의 마이크로 컴퓨터라고 보면 됩니다.

아두이노 우노 R3 보드는 ATMEGA328P-PU 가 탑재된 형태의 마이크로 컨트롤러 장치 보드입니다.

ATmega328(아트메가 328)칩에서 지원되는 입/출력 제어 포트의 형태 그대로 PCB 보드의 가장자리에 배치된 형태입니다.

아두이노 우노 R3 보드에는 2 개의 MCU 사용되고 있습니다.

ATmega16u2 MCU 와 ATmega328P-PU MCU 입니다..

ATmega16u2 는 USB 시리얼 포트 변환 용도로 사용됩니다.

ATmega328P-PU 는 MCU 로 사용되고 있습니다.

아두이노 스케치 프로그램에서 업로드 후 적재되는 장소는 ATmega328p-pu 의 프로그래밍 공간에 적재됩니다.

## 아두이노 우노 R3 보드 MCU 기술:

Microcontroller ATmega328

Operating Voltage 5V

Input Voltage (recommended) 7-12V

Input Voltage (limits) 6-20V

Digital I/O Pins 14 (of which 6 provide PWM output)

Analog Input Pins 6

DC Current per I/O Pin 40 mA

DC Current for 3.3V Pin 50 mA

Flash Memory 32 KB (ATmega328) of which 0.5 KB used by boot loader.

SRAM 2 KB (ATmega328)

EEPROM 1 KB (ATmega328)

Clock Speed 16 MHz

Length 68.6 mm

Width 53.4 mm

Weight 25 g

MCU는 거의 모두 프로그래밍 가능한 저장 공간이 있습니다.

기본 상태는 공백이라고 보면 됩니다.

아두이노 우노 R3 보드에는 MCU 플래시 메모리 내부에 0.5KB(almost 1KB) 부트로더 개념으로 사용되고 있습니다. 아두이노의 다른 여러 종류의 보드들도 공통적으로 부트로더 적재 되어 있습니다.

부트로더는 간단히 말하자면 시스템 부팅 및 기동 시에 필요한 하드웨어 초기화와 실행 프로그램이 적재 되어 있는 경우 작동 시켜 주는 기능입니다.

또한 아두이노와 PC USB 연결 상태에서 “아두이노 IDE 스케치” 프로그램에서 프로그래밍된 코드의 실행 파일 업로드에도 사용되고 있습니다.

위의 실행 파일은 보통 보통 Hex 파일, Bin 파일, 바이너리 파일, 펌웨어 파일 등으로 부릅니다. (아두이노에서는 스케치 파일이라고 부르는 것이 익숙합니다)

» IC (Integrated Circuit, 집적회로)란 무엇일까요?

IC 는 트랜지스터, 저항, 커패시터 등을 고밀도로 집적하여 회로 구성된 부품을 말합니다. “칩”이라고도 부릅니다. 전자부품들, 트랜지스터, 저항, 커패시터 등의 형태가 소규모, 소형화 형태로 된 것이 아닌, 실리콘 기판 위에 인쇄 형태의 회로로 모두 집적되어 구성된 형태입니다.

### 3 MCU 기초 정보

MCU 라는 단어는 Micro Controller Unit 입니다.

일반적으로 마이크로 컨트롤러 장치라고 부릅니다.

주로 특정 용도의 시스템을 제어하기 위한 핵심 부품입니다. 주로 전자제품에 많이 사용되고 있습니다.

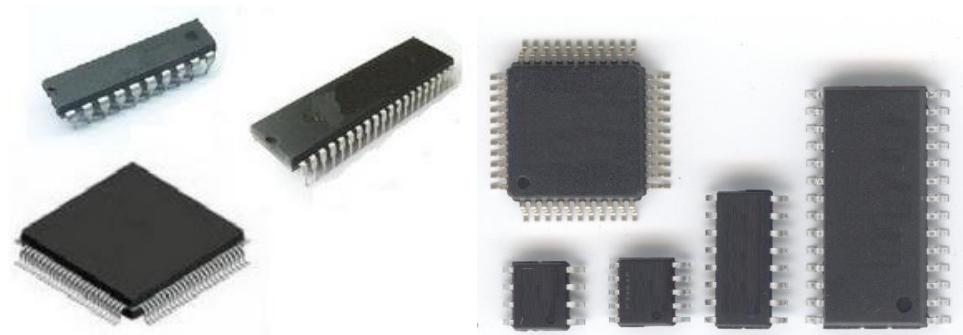


그림 3-1 다양한 MCU 종류와 용도에 따른 크기

전자제품에는 여러 가지의 하드웨어 주변 장치 모듈들이 많이 들어가게 되어 있습니다. 물론 기능에 따라 MCU도 1개 이상 들어갈 수 있습니다.

MCU는 주변장치 모듈에 대한 처리 및 제어를 할 수 있는 IC입니다.

MCU는 제조 개발사의 목적에 맞게 무수히 많은 종류들이 있고, 나름대로의 경쟁이 치열한 분야이기도 합니다. 시간이 경과할수록 독창적이고 다양한 새로운 MCU 들이 등장하고 있습니다. MCU는 사용 및 적용 목적에 따라 내부적으로는 많은 구조 및 성능개선과 변화가 이루어지고 있습니다.

현재까지의 많은 종류의 MCU 들의 기본 기능은, 프로그래밍을 통해 만들어서 적재된 실행 프로그램에 의해, 동작 수행 처리된 결과를 외부로 출력하게 되어 있습니다.

작동 형태는 컴퓨터의 기본 기능과 같다고 보면 됩니다.

MCU의 내부에는 CPU, 프로그래밍 가능 메모리, 보조 모듈, 외부 입출력 포트 등을 하나의 단일 칩에 구성되어 있습니다.

MCU 내부에는 용도에 맞게 적절한 크기의 램(Ram)과 CPU 기능, 데이터 영구 저장 공간, 타이머, 통신 포트, 입력/출력, 필요한 기능 등이 포함되어 있습니다.

MCU는 하나의 작은 미니 컴퓨터라고 보면 됩니다.

### 3.1 MCU, MPU, CPU 구분은?

MPU (Micro Processor Unit)는 컴퓨터의 핵심 기능인 기계어를 해석, 연산을 하는 기능에 중점을 두고 있습니다.

PC 의 CPU(Central Processing Unit)와 같은 장치는 MPU 라고 부릅니다.

CPU 는 순수하게 프로세싱 처리를 위한 유닛입니다. 순수하게 프로세싱 처리 속도를 위한 민감한 장치입니다.

물론, CPU 에도 작은 크기의 메모리와 보조 모듈이 있기는 합니다. 하지만, 빠른 처리를 위한, 즉, 처리 속도를 증가 시키기 위한 용도에 사용되고 있습니다.

PC 에서 사용되는 CPU 가 작동되기 위해서는 메모리, 하드 디스크, 통신 포트 등이 메인보드라는 장치에 의해 별도로 연결되어야 하는 구조입니다.

일반적으로 개인용 컴퓨터, 서버 등에서 사용되는 중앙 처리 장치는 CPU 라고 통칭됩니다. 전자기기, 산업용 기기에서 많이 사용되는 소형화된 CPU 들은 MPU 라고 부르게 됩니다.

#### » 아두이노 MCU 보드들의 새로운 고객 창조

일반적인 전자제품들은 제조사가 생산하고 개인이 소비를 하게 되어 있습니다.

일반적인 전자제품에는 다양한 종류의 MCU 와 전자부품들이 사용되고 있습니다. 하지만, 일반 소비자는 MCU 를 산다고는 생각 안 합니다. 냉장고, 세탁기, TV, 컴퓨터, 마우스, 키보드를 구매 한다고 생각합니다. 방금 언급된 전자제품만 해도 무수히 많은 종류의 MCU 들이 사용되고 있습니다.

기존의 MCU 의 주 고객층은 기업, 단체가 많았습니다. 제조사는 일반 소비자들에게 전자제품을 팔기 위해서는 생산을 해야 하므로 당연히 부품을 사서 제조를 합니다. 즉, 제조사, 관련 직종의 사람들만의 MCU/전자 부품을 활용한 제품 개발이었습니다.

하지만, 아두이노라는 새로운 물건이 나온 다음부터는 일반 개인들이 MCU(정확히 MCU 보드)를 바로 구매하여 사용 가능하게 됩니다.

MCU 등의 부품 제조사의 입장에서는 새로운 고객이 창출되는 구조입니다.

상식적으로 전자/전기와 전혀 관계없다고 보여지는 미술가들도 구매를 합니다.

이는 다른 시각으로 보면 개인이 제조를 할 수 있다는 말도 됩니다.

즉, 새로운 제품 개발 가능 영역이 개인까지 확대 되었다는 의미입니다.

이제는 개인도 목적에 따라 제품 개발이 가능합니다.

미래에는 좀 더 다양하고 뛰어난 아이디어를 가진 신제품들이 많이 나올 것이라고 예상됩니다.

## 4 ATMEGA328P 기초 정보

ATMEGA328P MCU 칩(IC) 그림입니다.

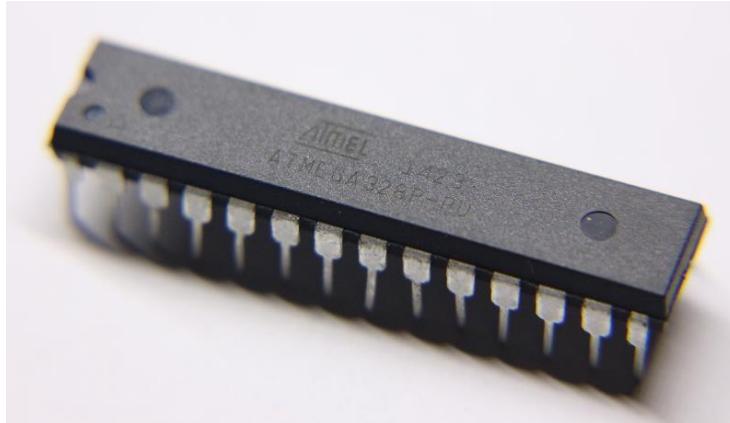


그림 4-1 ATmega328p-pu

왼쪽 14 핀, 오른쪽 14 핀, 모두 28 핀이 할당 되어 있습니다.

해당 제조사의 데이터 자료에 의하면 핀 역할은 아래와 같습니다.

ATmega328P DIP Pin Out 입니다.

|                          |    |    |                          |
|--------------------------|----|----|--------------------------|
| (PCINT14/RESET) PC6      | 1  | 28 | □ PC5 (ADC5/SCL/PCINT13) |
| (PCINT16/RXD) PD0        | 2  | 27 | □ PC4 (ADC4/SDA/PCINT12) |
| (PCINT17/TXD) PD1        | 3  | 26 | □ PC3 (ADC3/PCINT11)     |
| (PCINT18/INT0) PD2       | 4  | 25 | □ PC2 (ADC2/PCINT10)     |
| (PCINT19/OC2B/INT1) PD3  | 5  | 24 | □ PC1 (ADC1/PCINT9)      |
| (PCINT20/XCK/T0) PD4     | 6  | 23 | □ PC0 (ADC0/PCINT8)      |
| VCC                      | 7  | 22 | □ GND                    |
| GND                      | 8  | 21 | □ AREF                   |
| (PCINT6/XTAL1/TOSC1) PB6 | 9  | 20 | □ AVCC                   |
| (PCINT7/XTAL2/TOSC2) PB7 | 10 | 19 | □ PB5 (SCK/PCINT5)       |
| (PCINT21/OC0B/T1) PD5    | 11 | 18 | □ PB4 (MISO/PCINT4)      |
| (PCINT22/OC0A/AIN0) PD6  | 12 | 17 | □ PB3 (MOSI/OC2A/PCINT3) |
| (PCINT23/AIN1) PD7       | 13 | 16 | □ PB2 (SS/OC1B/PCINT2)   |
| (PCINT0/CLKO/ICP1) PB0   | 14 | 15 | □ PB1 (OC1A/PCINT1)      |

그림 4-2 ATmega328 pinout

출처: <http://www.atmel.com/images/8161s.pdf>

대부분의 MCU 프로그래밍 도구 IDE 프로그램은 “AVR Studio IDE” 에서는 위에서 정의된 핀맵을 참조하여 프로그래밍 하도록 되어 있습니다. 우노 R3 보드에서 사용되는 ATMEGA328P 뿐만이 아닌 무수히 많은 AVR MCU 에 프로그래밍 & 업로드 할 수 있습니다. 기존의 AVR 프로그래밍에서 사용하는 핀맵입니다.

물론, 위에 정의된 PIN 지정 용어들도 규칙이 있습니다.

전자 관련 개발자들은 익숙한 명칭들입니다. PDXX, PCXX, PCINTXX 등의 일정한 규칙은 사용하는 개발자들에게는 아두이노 만큼이나 친숙합니다.

핀 속성 파악 후 해당 방향 설정, 인터럽트로 처리해주어야 하고, 클럭 및 타이머 설정, 전압 설정까지, 내부적으로 값 저장하고, 거의 모두 포트 상태/방향을 비트 연산하여 처리 해야 합니다.

아두이노 스케치 IDE를 사용하지 않는다면 AVR Studio등의 프로그램을 사용하여 프로그래밍 해야 합니다.

물론, AVR Studio IDE 프로그램의 장점도 많습니다. 세밀하고 정밀한 하드웨어 펌웨어 프로그래밍이 가능합니다. 모든 AVR MCU 펌웨어 프로그래밍에 지금도 많이 사용되고 있습니다.

아래의 코드는 AVR Studio IDE에서의 PB5 포트 LED On/Off 하는 예제입니다.  
0.5 초마다 LED On / Off 반복하는 코드입니다.

```
#define F_CPU 1000000UL

#include <avr/io.h>
#include <util/delay.h>

int main (void)
{
    DDRB |= _BV(DDB0);
    while(1)
    {
        PORTB ^= _BV(PB0);
        _delay_ms(500);
    }
}
```

위의 코드를 AVR Studio IDE 설치 후 AVR MKII 등의 펌웨어 프로그래밍 장치 연결 후 업로드 하면 아두이노 우노의 보드에 있는 작은 LED 점등 됩니다.

아두이노 IDE 스케치에서 위의 정의된 코드를 거의 그대로 사용 가능합니다.

다만 main() 함수는 “아두이노 IDE 스케치 기본 빌드”에 포함되어 있는 관계로, 제공 되는 setup() 함수와 loop() 함수를 사용하여 아래와 같이 코드를 집어넣어야 합니다.

```

void setup() {
    // put your setup code here, to run once:
    DDRB |= _BV(DDB0);

    while(1)
    {
        PORTB ^= _BV(PB0);
        _delay_ms(500);
    }
}

void loop() {
    // put your main code here, to run repeatedly:
}

```

아두이노 IDE 스케치 업로드 후 LED 0.5 초마다 On / Off 반복됩니다.  
 물론 위의 코드 중 setup 함수에 있는 while(1)을 제거 후 loop 함수에서 기입 하여  
 도 같은 결과입니다.

```

void setup()
{
    // put your setup code here, to run once:
    DDRB |= _BV(DDB0);
}

void loop()
{
    // put your main code here, to run repeatedly:
    PORTB ^= _BV(PB0);
    _delay_ms(500);
}

```

» C/C++에서의 main() 함수는?

C/C++ 언어는 main이라는 함수가 있습니다. 코드가 실행 되는 경우 제일 처음 만나게 되는 함수입니다.

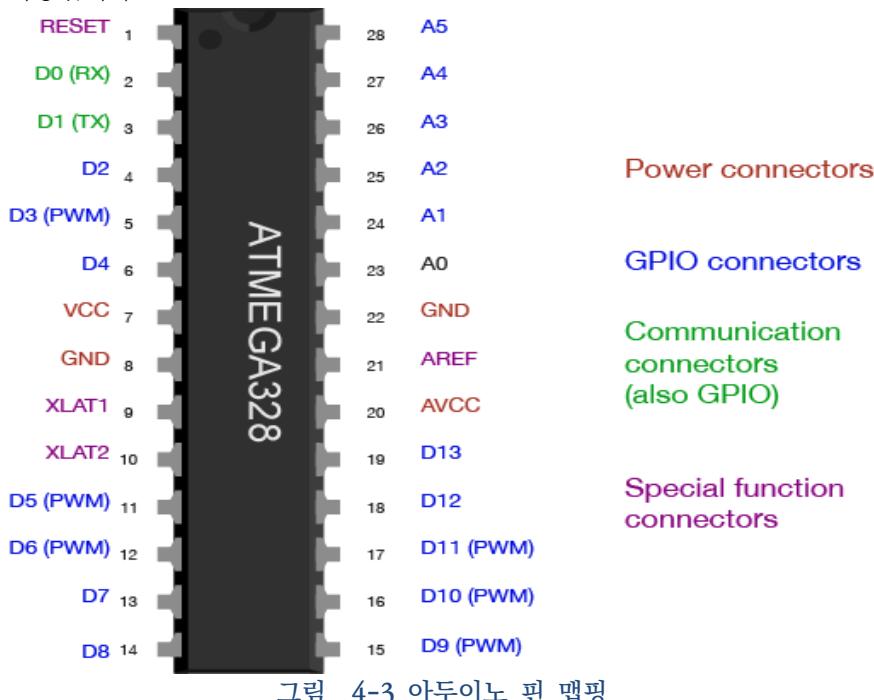
함수 본체는 보통 `int main(int argc, char** argv)` 입니다.

아두이노 스케치 프로그래밍에서는 이런 부분들을 기본 라이브러리 빌드에 포함되어 있어서, 사용자는 일부러 찾아보지 않으면 볼 수 없도록 되어 있습니다.

물론, 아두이노의 대부분의 프로그래밍에는 C/C++에서의 main() 함수 개념인 setup()과 loop() 함수만 사용해도 무방합니다.

## 4.1 아두이노의 포트 지정 규칙

아두이노 우노 R3 보드 & 아두이노 스케치 ID에서는 아래와 같은 명칭으로 일관적으로 사용됩니다.



아두이노는 별도의 부트로더가 올려져 있어서 아두이노 IDE 스케치 프로그램에서 위의 핀 설명으로 사용 가능하게 되어 있습니다.

아두이노 우노 R3 / 메가 2560 / 나노 / 프로미니 / 레오나르도 / 두에 등의 보드가 일관되게 사용됩니다. 아날로그 핀 추가 될 경우에는 A6, A7, A8 ~ A15 같이 상위 번호로 계속 번호가 매겨집니다.

디지털 핀 추가 또한 같은 규칙으로 D14, D15, D16 ~ D52 까지 있습니다.

위와 같이 크게 2 가지, 디지털 포트, 아날로그 포트에 대한 단순한 정의, 번호만 있을 뿐입니다.

간단하지만, 직관적이기도 합니다. 이런 사항들이 많은 사용자들에게 꾸준히 사용 되는 이유중의 하나입니다. 대부분의 사람들은 “디지털” “아날로그” 이해는 하고 있습니다.

그럼 아두이노 IDE 스케치 전용 함수를 사용하여 같은 기능을 구현하여 봅니다.

아두이노 IDE 기본 예제 Blink 코드.

아두이노에서 지정된 핀 명칭과 함수를 사용하여 구현된 예제 코드입니다.

```
// blink example code.
// the setup function runs once when you press reset
or power the board
void setup() {
    // initialize digital pin 13 as an output.
    pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
    digitalWrite(13, HIGH);    // turn the LED on (HIGH is
the voltage level)
    delay(1000);              // wait for a second
    digitalWrite(13, LOW);     // turn the LED off by making
the voltage LOW
    delay(1000);              // wait for a second
}
```

위의 코드가 아두이노 사용자들에게 익숙한 코드와 함수 명칭입니다.

기능은 같지만 위와 같이 좀 더 편하게 사용할 수도 있으면서도 기존의 함수 체계도 가지고 있어 용도에 따라 적절히 사용할 수 있습니다.

위에서 사용된 코드의 digitalWrite, pinMode, delay 함수와 digitalRead 함수 등만 이해 하여도 많은 아두이노 관련 모듈들을 사용 할 수 있습니다.

## 5 아두이노 우노 R3 MCU 보드에 대한 이해

아두이노 보드의 기본 기능은 MCU 하드웨어 제어 보드입니다.

각종 포트들은 해당하는 MCU 보드의 기능을 만족시켜 주고 있습니다.

차후 좀더 진보적인 프로젝트를 위해서는 MCU의 포트, 즉 아두이노 보드에 대한 간단한 이해가 필요합니다.

### 5.1 하드웨어 통신 I2C & SPI & ICSP 핀맵 기능 요약.

아두이노 보드와 같은 MCU 탑재 보드는 외부 기기를 제어하기 위한 용도가 대부분입니다. 펌웨어 프로그래밍 및 디버깅, 외부 기기, 또는 다른 MCU 탑재 보드와도 통신이 가능합니다. 통신을 하기 위해 지정된 포트는 일반 포트(GPIO) 말고도 규약된 통신 포트로 사용할 수 있습니다.

보드 구성 이해를 하기 위한 핀맵(Pin Map) 다이어그램입니다.

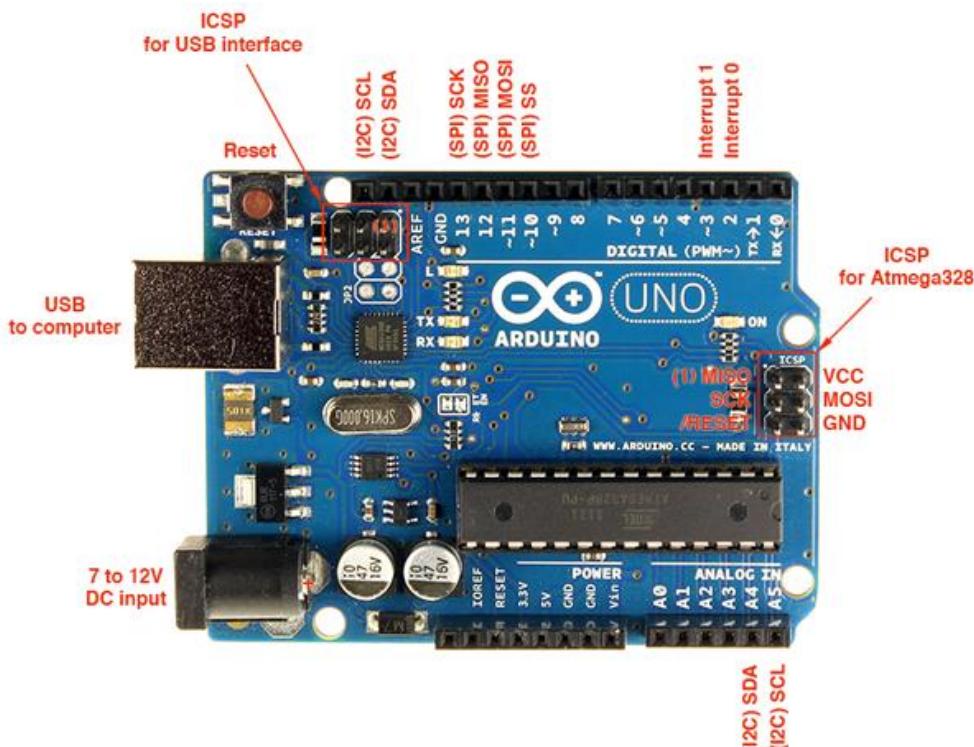


그림 5-1 우노 R3 보드 기능 포트

(그림 출처: <http://www.gammon.com.au/forum/?id=11473>)

보드 위에 있는(빌드 되어 있는) 여러 포트들은 ATMEGA328P MCU 의 기능을 전부 반영 하고 있습니다. SPI, I2C, ICSP, 리셋 스위치, 인터럽트 포트와, 디지털 포트, 아날로그 포트, PWM 포트, 안정적인 전원 공급을 위한 레귤레이터까지 기본적인 MCU 보드의 기능을 충실히 지원되고 있습니다.

USB 연결 방식에 의한 프로그램 업로드를 위해 ATmega16u2 IC도 사용하고 있습니다. 시리얼포트와 I2C(IIC)와 SPI 통신 포트는 지원되는 기기와의 통신이 가능합니다. 물론 다른 MCU 보드와도 통신이 가능합니다.

### 5.1.1 ICSP 란?

In Circuit Serial Programming, 용어 그대로 MCU에 직접 프로그래밍 가능한 통신 포트를 말합니다. 보통 펌웨어 프로그래밍/디버깅을 위한 포트로 사용되고 있습니다. ICSP 포트를 통한 아두이노 IDE 스케치 프로그램을 사용하여 펌웨어 업로드 할 수 있으며, 동시에 기존 프로그래밍 방식의 AVR Studio 등의 개발환경까지 사용할 수 있습니다.

ICSP 지원되지 않는 MCU는 대부분 간단한 기능만을 위한 MCU가 대부분입니다. ICSP 기능이 없는 MCU는 별도의 룸(ROM)에 프로그래밍, 업로드 후 사용하는 방식입니다.

ICSP 포트는 일반적으로 6 핀(포트)(SPI 통신), 10 포트(SPI 포트와 여러 용도의 추가 포트)가 사용됩니다.



그림 5-2 ICSP 6 포트

ICSP 포트를 통한 부트로더 & 펌웨어 도구로는 AVR MKII 또는 USBTinyIsp 등이 많이 사용됩니다.

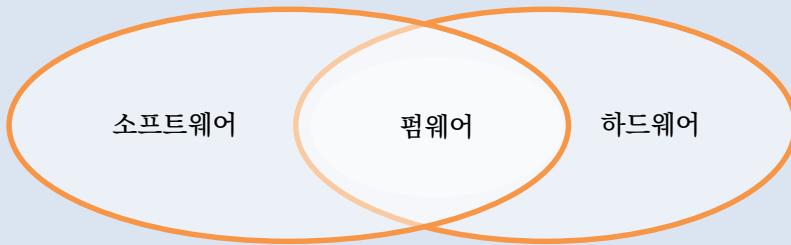
아두이노의 거의 모든 보드는 USB 시리얼 포트를 통한 펌웨어 업로드 방식이지만, 아두이노 부트로더 업로드 필요할 경우에는 AVR MKII, USBTinyIsp 등을 사용합니다. 물론, AVR MKII 기기로 아두이노 IDE 스케치 프로그램에서 만들어진 HEX 파일도 업로드 가능합니다.

참고로 아두이노 IDE 스케치 프로그램에서 빌드(컴파일&링크)된, 만들어진 HEX 파일은 USB 시리얼 포트로 업로드 하기 위해 윈도우즈 사용자 임시 디렉터리에 있습니다.

임시 디렉터리 아래에 프로젝트명칭의 알파벳명칭+임시번호.tmp 디렉터리 입니다.  
HEX 파일은 직접 ICSP 업로드 장치를 통해 업로드 해도 무방합니다.

### » 펌웨어란? (Firmware)

펌웨어는 소프트웨어와 하드웨어의 중간 개념의 소프트웨어를 말합니다.



하드웨어의 MCU 또는 ROM 등에 들어가는 소프트웨어가 펌웨어라고 불려지고 있습니다. 즉, 하드웨어에 들어가는 소프트웨어를 말하고 있습니다.

하드웨어를 작동하게 하는 기본적인 실시간 운영체제 개념의 프로그램이기도 합니다.

펌웨어는 C/C++, ASM, BASIC 등의 언어로 구현 되고 있습니다. 만들어진 프로그램은 MCU의 프로그램 실행 영역에 적재되면서 프로그램이 실행되게 됩니다.

펌웨어는 하드웨어에 적재되어 작동되는, 하드웨어 의존적인 구조입니다.

펌웨어는 결국 하드웨어 구성과 조합되어 완성되면, 하드웨어 수명이 다 할 때까지 대부분 지속적으로 사용하게 됩니다. 그래서 “FirmWare”(펌웨어)라는 명칭으로 오래 전부터 불리어온 이유이기도 합니다.

대부분의 펌웨어는 코드의 크기가 작은 간단한 코드들로 구성되어 있습니다. 1회 개발 후 지속적으로 사용되고 있습니다.

요즘에는 점점 더 복잡한 전자기기가 많아지는 관계로 MCU의 프로그래밍 저장 공간 및 속도가 빨라지고, “펌웨어 업그레이드” 방식으로 1회성의 펌웨어 업로드가 아닌, 일반적인 소프트웨어와 비슷한 개념으로 제조사에서의 지속적인 업데이트 개발이 많아지고 있는 추세입니다.

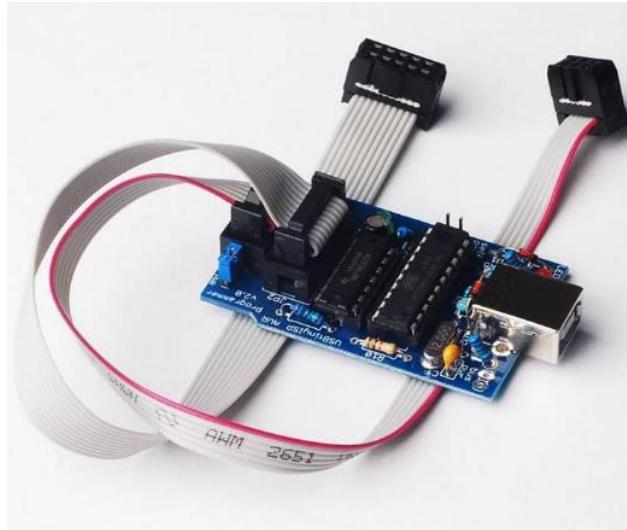


그림 5-3 USBTinyISP for Arduino



그림 5-4 AVR MK II (시리즈 2)

ATmega328 MCU 등의 AVR 계열의 칩셋은 AVR MK 프로그래머를 사용하는 경우 전체 영역에 대한 변경 및 추가가 가능합니다.

USBTinyIsp 같은 경우에는 거의 모든 기능을 사용할 수 있습니다. 하지만 AVR Studio IDE 프로그램 등에서 사용하지는 않고 아두이노 IDE 프로그램 등에서만 사용 가능합니다.

AVR MK 프로그래머 장치에 비해 접근 및 변경 용도가 제한적이기도 합니다. 아두이노 IDE에서 AVR MK II를 사용하여 업로드를 하는 경우 libUsb 드라이버를 별도로 설치 후 사용해야 합니다.

### 5.1.2 ICSP for USB Interface:

ATMEGA16U2 펌웨어 업로드 연결 ISP 포트.

아두이노 우노 R3 보드는 ATMEGA16U2 사용됩니다.

ATMEGA16U2 IC는 USB 가상 시리얼 포트의 용도로 사용되고 있습니다.

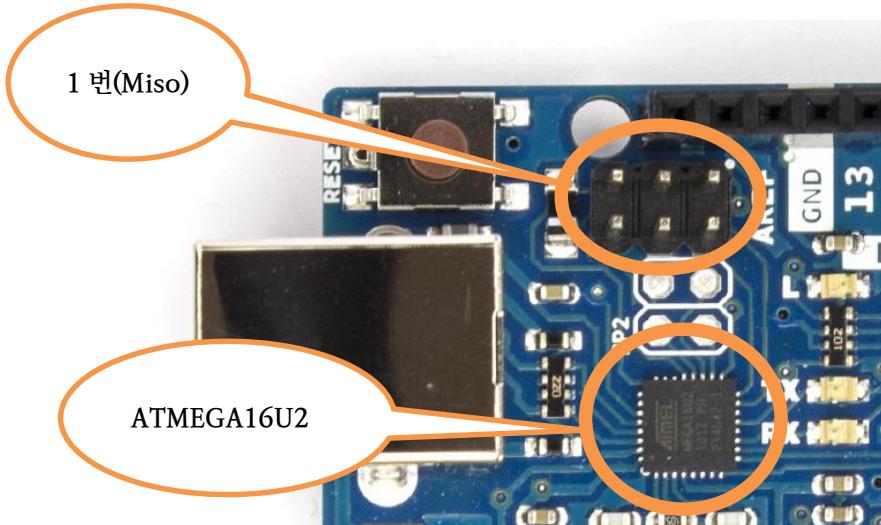


그림 5-5 ICSP 포트와 ATMEGA16U2 위치

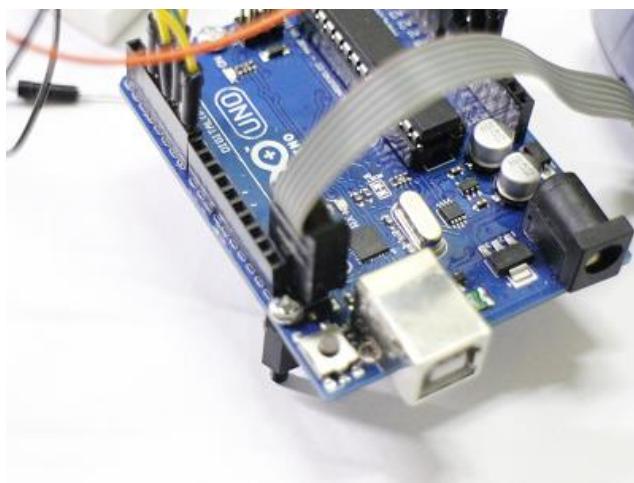


그림 5-6 ICSP ATMEGA16U2 사용

참고로 아두이노 우노 R3 에서의 ATMEGA16U2 펌웨어는

아두이노 IDE 스케치 프로그램이 있는 디렉터리 아래에 있습니다.

“arduino-1.5.8\hardware\arduino\avr\firmwares\atmegaxxu2”

아래의 파일 이름은 우노 R3 의 ATmega16U2 펌웨어입니다.

“Arduino-COMBINED-dfu-usbserial-atmega16u2-Uno-Rev3.hex”

### 5.1.3 ICSP for ATMEGA328P:

ATMEGA328P 펌웨어 업로드 연결 ISP 포트..

부트로더 또는 펌웨어까지 업로드 할 수 있는 통신 포트입니다.

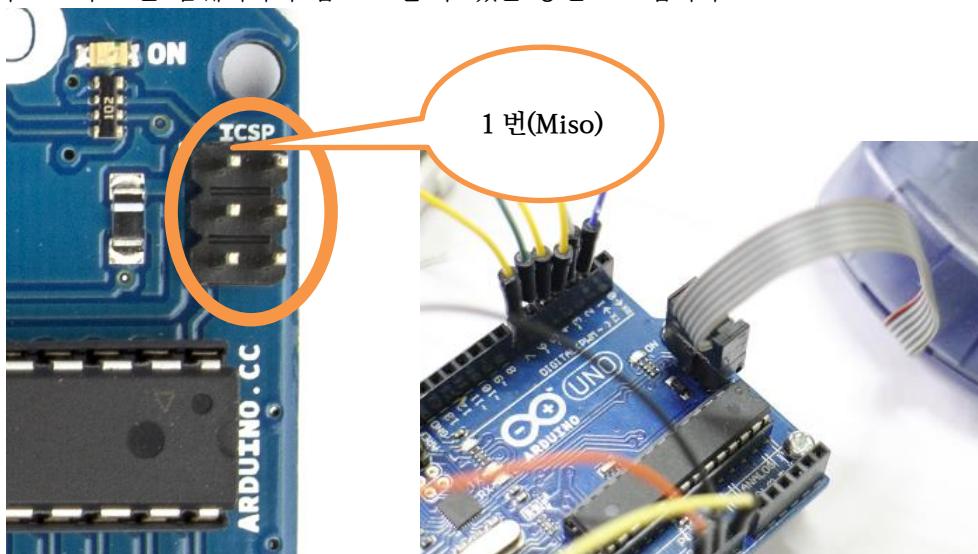


그림 5-7 ICSP ATMEGA328P 참조

아두이노 우노 R3 에서의 ATmega328p 펌웨어 파일은 아두이노 IDE 스케치 프로그램이 위치한 아래에 있습니다.

“\arduino-1.5.8\hardware\arduino\avr\bootloaders\optiboot”

“optiboot\_atmega328.hex” 파일 이름은 우노 R3 펌웨어 파일입니다.

“arduino-1.5.8\hardware\arduino\avr\bootloaders” 디렉터리 아래에 아두이노 보드의 거의 모든 부트로더 파일이 있습니다.

“caterina” 디렉터리는 “아두이노 레오나르도” 부트로더 펌웨어 파일이 있습니다.

“stk500v2” 디렉터리는 “아두이노 메가 2560” 부트로더 펌웨어 파일이 있습니다.

아두이노 IDE 프로그램에서의 부트로더 파일을 업로드 할 경우는 “USBTinyISP” 또는 여러 종류의 장치가 지원되고 있습니다. 아두이노 IDE 의 메뉴 선택만으로 바로 부트로더 업로드가 가능하게 되어 있습니다.

먼저 도구->보드-> 업로드 할 보드를 선택 합니다.

다음은 도구->프로그램-> “사용중인 업로드 장치 선택을 합니다.  
도구->”부트로더 굽기” 선택하면 현재 선택된 아두이노 보드에 맞게 자동으로 업로드 됩니다.

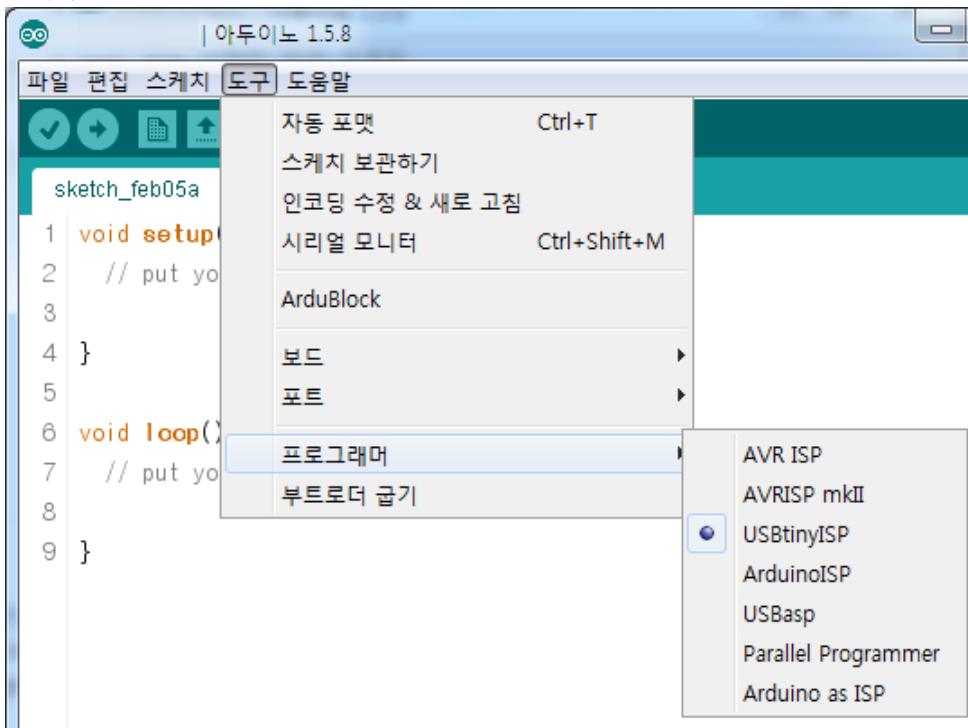


그림 5-8 업로더 장치 선택

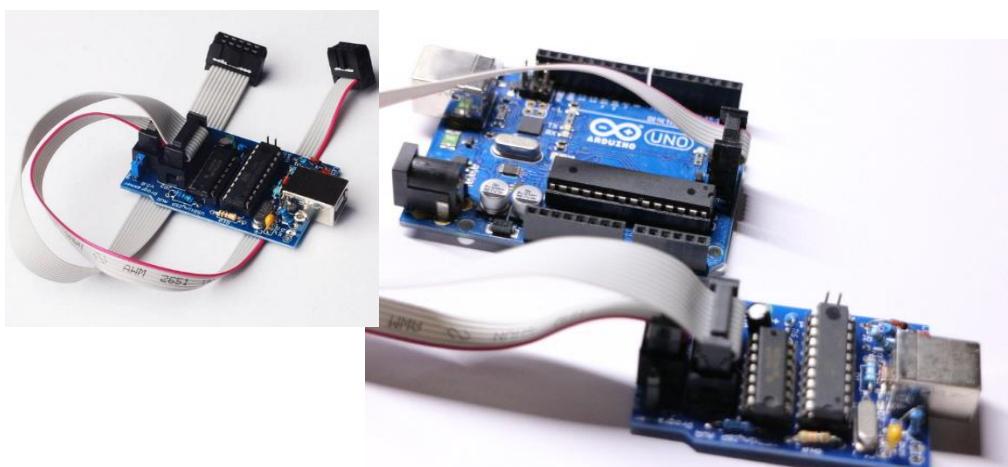


그림 5-9 USBtinyISP 장치와 우노 R3 ICSP 연결

#### 5.1.4 SPI 포트:

SPI(Serial Peripheral Interface) 통신을 위한 포트입니다.

SCK, MISO, MOSI, SS 명칭의 포트가 있습니다.

지금은 구글에 합병된 회사인 Motorola 사에서 처음 고안된 통신 인터페이스입니다.

MCU 와 주변 장치간의 시리얼 통신을 하기 위한 규약입니다.

주변 장치와의 Clock 을 통하여 동기화하여 동기식 통신 방식입니다.

전이중(Full Duplex) 통신방식이라고 합니다. 데이터 통신을 위해 정의된 포트가 2 개입니다. 통신을 위한 포트가 2 개이므로 데이터 송신/수신이 동시에 가능한 방식입니다.

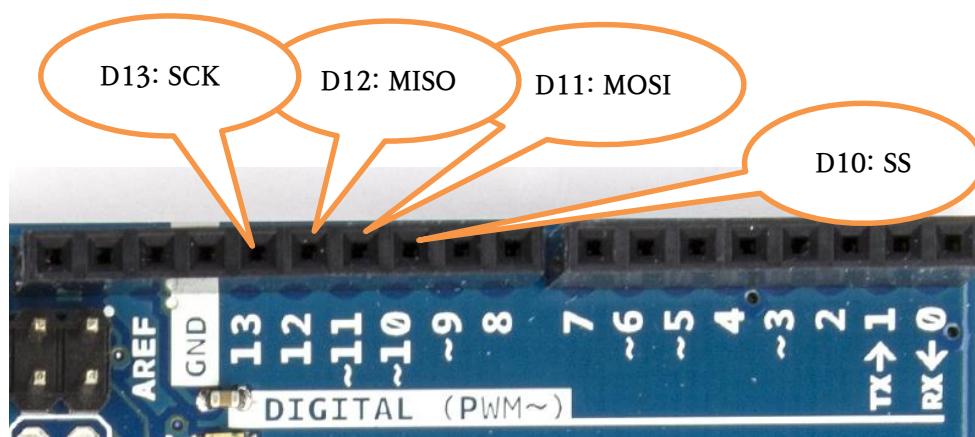
1:N 통신, 하나의 마스터와 다수의 슬레이브간의 통신이 가능합니다.

아두이노 보드의 MCU 와 부트로더 펌웨어에 의해 아래와 같은 SPI 포트가 할당 되어 있습니다.

〈SPI 보드 종류별 포트〉

| Arduino Board        | MOSI         | MISO         | SCK          | SS<br>(slave) | SS (master) |
|----------------------|--------------|--------------|--------------|---------------|-------------|
| Uno or Duemilanove   | 11 or ICSP-4 | 12 or ICSP-1 | 13 or ICSP-3 | 10            | -           |
| Mega1280 or Mega2560 | 51 or ICSP-4 | 50 or ICSP-1 | 52 or ICSP-3 | 53            | -           |
| Leonardo             | ICSP-4       | ICSP-1       | ICSP-3       | -             | -           |
| Due                  | ICSP-4       | ICSP-1       | ICSP-3       | -             | 4, 10, 52   |

아두이노 우노 R3 보드에서는 위의 표와 같이 D11, D12, D13, D10 사용되고 있습니다.



〈SPI 포트 표시 용어 설명 참조〉

| Signal Name | Alternative Names     | Description                            |
|-------------|-----------------------|--|
| SCLK        | SCK, CLK              | Serial Clock                           |
| MOSI        | SDI, DI, SI           | Master Output Slave Input              |
| MISO        | SDO, DO, SO           | Master Input Slave Output              |
| SS          | nCS, CS, nSS, STE, CE | Slave Select, Chip Enable, Chip Select |

SCK (Serial Clock) - 데이터 전송의 동기화를 맞추기 위해 마스터가 생성하는 clock pulse. SCK, CLK 등으로 표시가 됩니다.

MISO (Master In Slave Out) - 마스터로 데이터를 전송하는 포트. 모듈, 기기에 따라 SDI, DI, SI 등으로 표시가 됩니다.

MOSI (Master Out Slave In) - 마스터에서 디바이스로 전송. 모듈, 기기에 따라 SDO, DO, SO 등으로 표시 됩니다.

SS (Slave Select) - 마스터가 특정 디바이스를 활성화/비활성화 하기 위해 사용되는 포트. 모듈, 기기에 따라 nCS, CS, nSS, STE, CE 등으로 표시가 되기로 합니다.

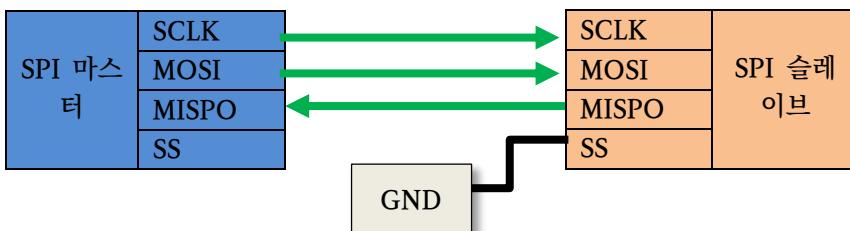
일반적으로 SS 포트의 값이 HIGH 상태인 경우 SS 포트에 연결된 SPI 기기는 정지 상태가 됩니다. 반대로 LOW 상태인 경우는 SS 포트에 연결된 SPI 기기와 통신을 하게 됩니다. 아두이노와 사용을 하는 대부분의 요즘 모듈, 기기들은 위의 SCK, MOSI, MISO, SS 등으로 동일하게 표시되고 있는 추세입니다.

SPI 연결 방식에는 4-Wire, 3-Wire, 멀티 슬레이브 방식이 있습니다.

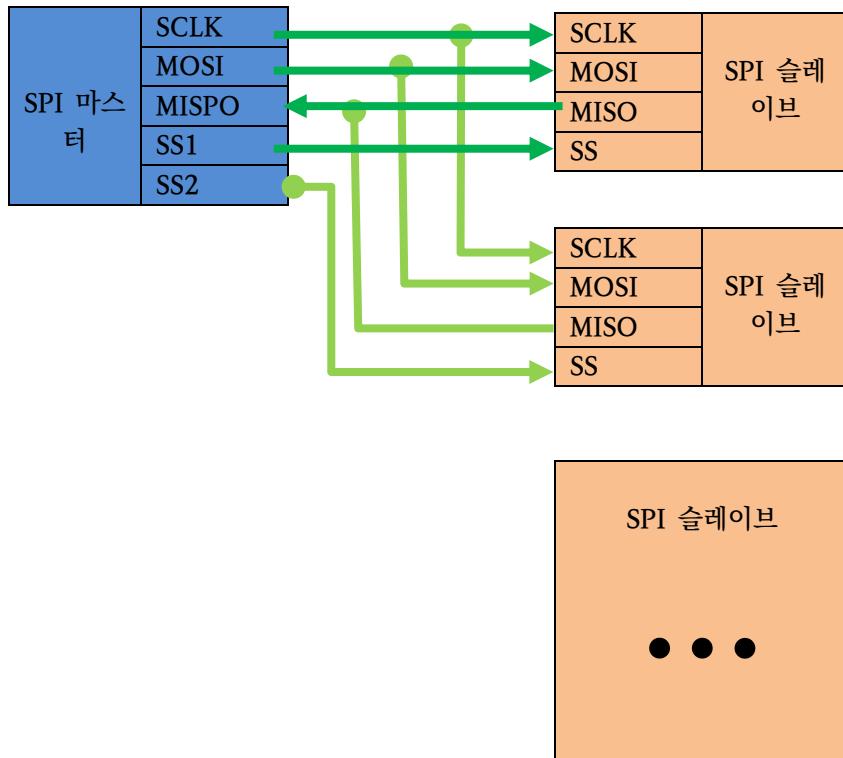
» 4-Wire 방식 연결 – 마스터에서 슬레이브 장치의 작동 Enable / Disable 기능 사용



» 3-Wire 방식 연결 – 마스터에서 슬레이브 장치 지속적으로 사용 연결 방식  
슬레이브의 SS 포트를 GND (LOW) 상태 유지하면서 계속 사용되게 됩니다.



» 여러 개의 슬레이브 연결 방식. SS 포트를 2 개 지정해서 SS1, SS2로 사용합니다.  
SS1, SS2 포트는 디지털 포트를 사용하면 됩니다.  
그리고 SPI 통신의 특이점은 마스터 장치는 1 개입니다.



아두이노 프로그래밍 코드에서는 SPI 통신을 사용하기 위해 `#include <SPI.h>` 선언하여 함수, 클래스 등을 사용할 수 있습니다.  
아두이노에서 많이 사용되는 SPI 통신 모듈로는 SD 카드 모듈, 이더넷 네트워크 모듈이 있습니다. 그 외에 RFID, NFC 모듈과 아두이노와의 SPI 통신 등이 있습니다.

#### » 아두이노에서 2 개의 SPI 통신 기기 사용하기

프로젝트의 목적에 따라 1 개 이상의 SPI 모듈을 사용해야 하는 경우가 있습니다.  
아두이노 보드가 마스터이고 나머지 2 개의 SPI 기기는 슬레이브 방식으로 통신을하게 됩니다.  
SPI 기본 포트의 설명을 조금이라도 이해되는 상태라면 안다면 어렵지 않게 1 개 이상의 SPI 모듈을 아두이노 보드에 적용하여 사용 할 수 있습니다.

SPI 통신 포트 중 SS (Slave Select)포트는 고정적이지 않습니다. 아두이노에서의 1개 이상의 SPI 기기를 사용하는 경우에는 SS 포트 연결 시 다른 포트를 지정하여 사용할 수 있습니다. 용어 그대로 SS로 지정된 포트가 LOW(or HIGH)일 경우에는 해당 SPI 포트에 연결된 장치와 통신을 한다는 의미입니다. SS 포트는 사용자가 정할 수 있습니다. SS 포트는 다른 디지털 포트로 지정하여 사용 가능합니다. D9, D8, D7 등의 포트를 프로그래밍에서 사용할 수 있습니다.

SPI 방식의 RFID 리더기 모듈에서 읽어 들인 출입 카드의 내용을 SPI 방식의 SD 카드에 저장하는 경우 2개의 SPI 기기와 통신을 해야 합니다.

이런 경우에는 위에서도 언급 되어 있듯이 SS 포트만 다른 디지털 포트(HIGH, LOW 가능 포트)를 지정하여 사용할 수 있습니다.

SPI 통신 포트 SCK, MISO, MOSI 포트에는 2개의 기기를 병렬로 연결하면 됩니다.

### 5.1.5 I2C:

Inter-Integrated Circuit, I2C라는 단어와 IIC는 같은 의미입니다.

I2C(Inter Integrated Circuit)는 MCU(마이크로프로세서장치)와 저속 주변 장치 사이의 통신을 위한 규약입니다.

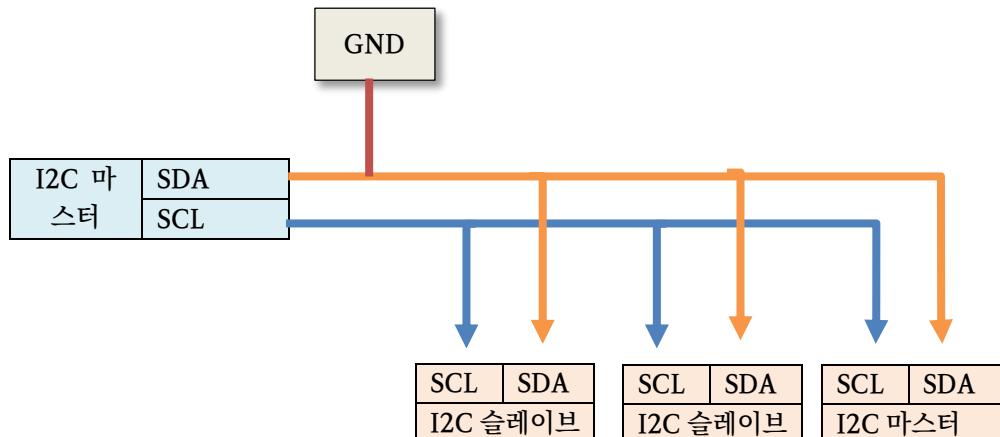
필립스 회사에서 개발되어 현재도 많이 사용되는 방식입니다.

TWI (Two Wire Interface), eye-squared-see, Inter IC Control, 통신 방식에 대한 용도 및 접근 방식에 따른 표현 단어가 조금 다를 뿐 같은 의미입니다.

SPI 통신과는 다르게 2라인을 사용합니다. 2가닥의 선으로 통신한다는 것이 큰 장점 중에 하나이기도 합니다.

SCL은 동기화 클럭 조절 통신을 위한 포트입니다.

SDA는 데이터 통신을 위한 포트입니다. I2C 통신은 데이터 전송을 위해 하나의 포트(연결)가 사용되므로, 데이터는 한번에 한 방향으로만 전송이 됩니다. 단방향 통신이라고 합니다. 송신/수신이 동시에 할 수 없기 때문에 저속 통신 기기에 적용하여 대부분 사용합니다. 단방향 통신은 비동기 통신이라고도 합니다.



I2C 통신 규약도 1:N 통신이 가능한 규약입니다. I2C 마스터, I2C 슬레이브, 고유 식별 ID 지정 기능이 있어, 지정된 곳으로 데이터 통신이 가능합니다.

아두이노에서 I2C 통신을 사용하기 위해서는 Wire 라이브러리를 사용합니다.

아두이노 코드상에서는 `#include <Wire.h>` 선언하여 사용합니다.

아두이노 우노 R3 보드에서는 SDA, SCL 포트 A4, A5 포트가 사용되고 있습니다.

아두이노 우노 R3 (Revision(리비전 또는 개선판 3 번째) 보드에서는 I2C 포트가 2 군데 있습니다.

우노 R3 보드의 왼쪽 위, 오른쪽 아래 있습니다. Rev 3 에서 왼쪽 위 I2C 포트 추가 되었습니다.

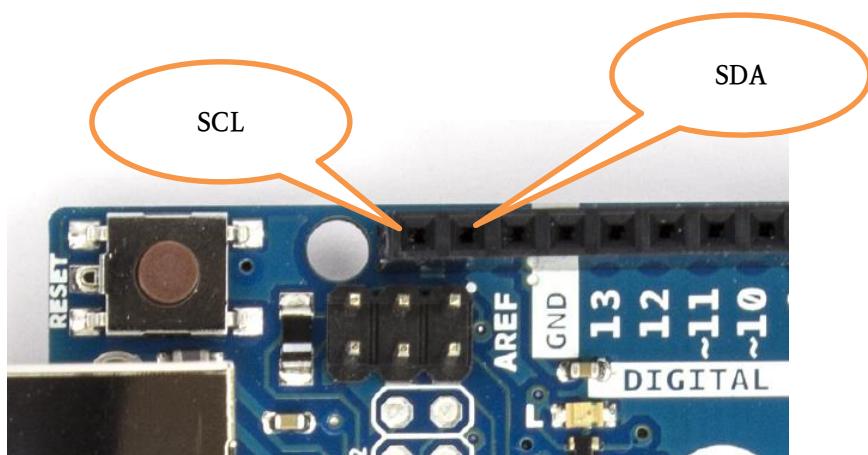


그림 5-10 i2c 포트 2 번째 위치

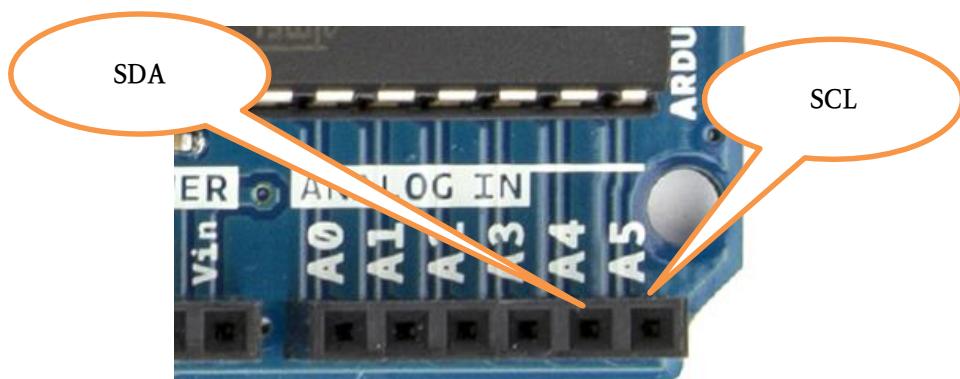


그림 5-11 i2c 포트

## » 아두이노에서 2 개 이상 I2C 기기 사용하기

아두이노의 Wire 라이브러리 사용시에는 간편하게 I2C 통신을 위한 I2C 기기의 연결된 주소 검색 코드의 예제가 있습니다. 아두이노에서 사용되는 대부분의 I2C 모듈들은 예제코드가 지원되어 있고, I2C 모듈의 주소도 기입되어 있어 사용하기 무리 없지만, I2C 주소를 직접 알아봐야 할 경우도 있습니다.

<http://playground.arduino.cc/Main/I2cScanner> 사이트에서 자세한 설명을 하고 있습니다. Wire 라이브러리를 사용하여 I2C 주소를 가져오는, 검색하는 예제 코드를 보여주고 있습니다.

## 5.2 아두이노 시리얼 통신

시리얼 통신(Serial Communication)은 기기들간의 데이터를 주고받는 방법 중 하나입니다. 아주 오래된 방식이지만 PC, MCU 등의 눈부시게 발전하는 기술의 기초적이고도 중요한 통신 방법입니다.

아두이노 우노 R3 보드는 시리얼 포트가 1 개 지원되고 있습니다.

참고로 메가 2560은 하드웨어 시리얼 포트가 4 개 있습니다.

ATmega328 MCU에서 1 개의 하드웨어 시리얼 포트가 지원됩니다.

1 개의 UART(universal asynchronous receiver and transmitter) 있습니다.

UART는 범용 비동기 송수신 직렬 포트입니다.

참고로 USART라는 것도 있습니다. 이것은 범용 동기식 송수신 직렬 포트입니다.

비동기는 송/수신이 한쪽에서만 가능한 구조입니다. 동기식은 송/수신 양쪽에서 동시에 가능합니다.

### 5.2.1 하드웨어 시리얼 통신.

아두이노에서는 D0, D1 포트가 시리얼 통신에 사용되고 있습니다. 일반적인 시리얼 포트의 RX/TX 기능만 사용되고 있습니다.

ATmega328p MCU는 1 개의 UART를 가지고 있습니다. UART에서 사용되는 포트는 우노에서는 D0, D1에 연결되어 있습니다.

PC와 USB 연결된 상태에서 아두이노 IDE 스케치 프로그램에서의 펌웨어 업로드에 사용되고 있습니다. PC와 USB 연결된 상태에서 D0, D1은 독점 예약 시리얼 포트라고 보면 됩니다.

아두이노 보드와 PC 와 USB 연결시 “가상 USB 시리얼 포트” 역할을 해주는 ATmega16u2 MCU가 사용되고 있습니다. ATmega16u2는 USB 통신을 시리얼 데이터를 변환하여 ATmega328과 통신을 하고 있습니다. 이때는 항상 ATmega16u2와 ATmega328p의 UART 포트와 연결된 상태로 서로 통신이 됩니다.

ATmega16u2 와 ATmega328p 와의 시리얼 데이터 교환이 발생되는 경우에는 우노 보드의 RX/TX 의 LED 깜박이게 됩니다.

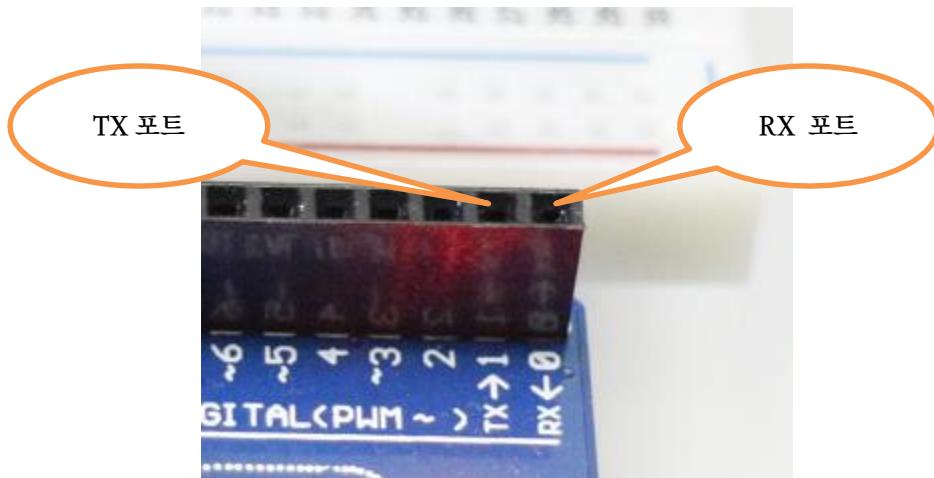


그림 5-12 RX, TX 포트

시리얼 포트로부터 받은 문자를 비교하여 “1” 인 경우에는 LED 를 ON, “0” 일때는 OFF 하는 예제입니다.

아두이노에서는 setup() 함수와 loop() 함수와 더불어 예약된 함수 serialEvent()라는 함수가 있습니다.

```
void serialEvent () {  
//statements  
}
```

D0, D1 의 시리얼 포트에 데이터가 있을 경우에 호출되는 이벤트 함수와 비슷합니다.

예제코드)

```
void setup0 {  
// put your setup code here, to run once:  
Serial.begin(9600);  
pinMode(13,OUTPUT);  
}  
  
void loop0 {  
// put your main code here, to run repeatedly:  
}  
  
void serialEvent0 // 예약된 함수.  
{
```

```

int read_byte = Serial.read();
Serial.println(read_byte);
if(read_byte=='1')
{
    digitalWrite(13,HIGH);
    Serial.println("LED ON");
}
else
if(read_byte=='0')
{
    digitalWrite(13,LOW);
    Serial.println("LED OFF");
}
}

```

시리얼 모니터 장을 열어서 1 또는 0 을 입력하여 봅니다.

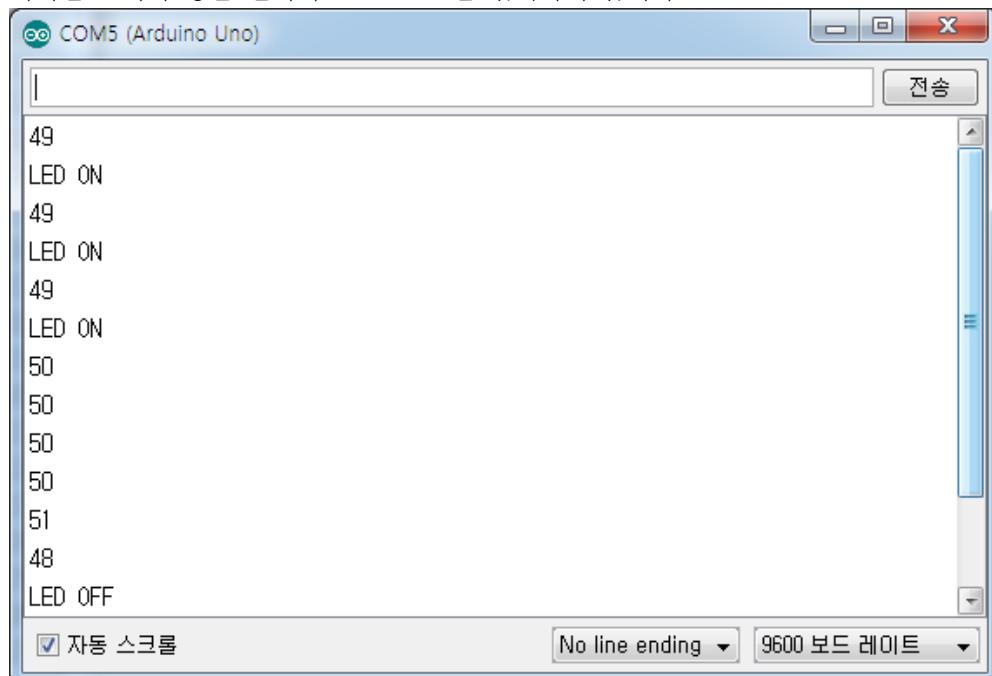


그림 5-13 시리얼 모니터 창 로그

참고로 “아두이노 메가 2560” 보드는 4 개의 하드웨어 시리얼 통신 포트를 지원하고 있습니다. 동시에 4 개의 시리얼 포트 연결이 가능합니다. 물론 소프트웨어 시리얼 통신도 가능합니다. 보다 많은 기기와의 시리얼 통신을 사용하고자 하는 경우 아두이노 메가 2560 보드를 사용합니다.

## 5.2.2 소프트웨어 시리얼 통신

아두이노에서는 디지털 포트를 이용하여 시리얼 통신을 할 수 있습니다.

아두이노의 기본 라이브러리 중 “SoftwareSerial” 있습니다.

<http://arduino.cc/en/Reference/SoftwareSerial>

소프트웨어 시리얼 통신이란 하드웨어에서 지원되는 UART or USART 포트를 사용하지 않고 일반적인 포트를 사용하여 시리얼 통신을 합니다. 소프트웨어적인 방법으로 시리얼 통신을 하게 되므로 하드웨어 시리얼 통신 보다는 속도 점 등의 여러 단점이 있습니다.

기존의 AVR 프로그래밍에서는 꽤나 어려운 펌웨어 개발이었지만, 아두이노에서도 여러 번의 수정을 거쳐 안정적인 소프트웨어 시리얼 통신을 하게 된 거 같습니다.

아래의 코드는 10,11 번 포트를 이용하여 시리얼 통신을 하는 예제입니다.

속도는 4800 baud rate (보레이트)로 테스트를 하지만 9600, 또는 그 이상도 안정적으로 통신이 가능합니다. 하드웨어 시리얼 포트와 소프트웨어 시리얼 포트에 데이터가 있으면 읽어서 다른 시리얼 포트에 출력하는 프로그램입니다.

```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 11); // RX, TX

void setup()
{
    // Open serial communications and wait for port to open:
    Serial.begin(57600);
    while (!Serial) {
        ; // wait for serial port to connect. Needed for Leonardo only
    }

    Serial.println("Goodnight moon!");

    // set the data rate for the SoftwareSerial port
    mySerial.begin(4800);
    mySerial.println("Hello, world?");
}

void loop() // run over and over
{
    if (mySerial.available())
        Serial.write(mySerial.read());
    if (Serial.available())
        mySerial.write(Serial.read());
}
```

10 번과 11 번 포트에는 시리얼 통신이 되는 다른 모듈을 연결하여 사용합니다.  
대표적으로 블루투스 시리얼 통신 모듈, GPS 모듈 등이 많이 사용되고 있습니다.  
소프트웨어 시리얼 통신 사용시 주의할 점은 PCINT 지원되는 핀만 사용해야 됩니다.  
10,11 번 이외에 2,3 번이 안정적입니다. 아두이노 공식 사이트에서 소프트웨어 시리얼 문서가 많으므로 참조하기 바랍니다.  
참고로 “아두이노 메가 2560” 과 “아두이노 우노 R3” 보드의 소프트웨어 시리얼 통신 가능한 포트는 다릅니다.  
아두이노 우노 R3 다음으로 많이 사용되는 “아두이노 메가 2560” 보드에서의 시리얼 통신 가능 포트는 아래와 같습니다.

Mega 2560 support change interrupts, so only the following can be used for RX: 10, 11, 12, 13, 14, 15, 50, 51, 52, 53, A8 (62), A9 (63), A10 (64), A11 (65), A12 (66), A13 (67), A14 (68), A15 (69).

주의할 점은 여러 개의 소프트웨어 시리얼 통신을 사용하는 경우 하드웨어 시리얼 통신과는 다르게 1 개의 소프트웨어 시리얼 포트에서만 데이터 수신이 가능합니다.  
동시에 소프트웨어 시리얼 통신 수신은 되지 않습니다.

### 5.3 EEPROM

EEPROM(electrically erasable and programmable read only memory), 보통 이 투피롬(E2PROM)으로 불려지고 있습니다.

ATMEGA328P MCU 내부에는 1 KB 크기의 EEPROM 이 있습니다. 최대 1KB(키로 바이트)까지 저장 가능한 메모리입니다.

EEPROM은 전기 신호로만 삭제, 프로그래밍 가능한 읽기 전용 메모리입니다.  
참고로 ATMEGA168, ATMEGA8은 512BYTE, ATMEGA1280, ATMEGA2560 MCU는 4KB의 크기를 가지고 있습니다.

1KB 크기는 하드디스크, SD 카드에 비하면 엄청 작은 크기라고 볼 수 있습니다.  
실제 아두이노 등을 통하여 프로그래밍 시에 1KB 정도면 웬만한 프로젝트의 환경 설정 기본값들은 저장하고도 남는 크기입니다.

EEPROM은 MCU 내부의 다른 메모리와는 사용 용도가 명확하게 다릅니다.  
데이터를 영구 저장하기 위한 메모리로 사용됩니다. 불휘발성 메모리입니다. 전원이 없어도 기억 내용이 지워지지 않습니다.

아두이노에서는 기본 라이브러리에서 EEPROM에 데이터 읽기/쓰기 가능합니다.  
EEPROM 라이브러리 사용하기 위해서는 `#include <EEPROM.h>` 선언하여 사용할 수 있습니다.

아두이노의 EEPROM 사용 라이브러리도 물론, C++ 클래스 형태로 되어 있습니다.

아래의 예제코드는 EEPROM 의 주소 0 번째부터 512 번까지 읽어서 시리얼 프린트 하는 예제입니다. EEPROM 의 1KB 의 용량의 절반 512 BYTE 를 읽어서 시리얼 프린트 합니다. 1KB 는 1024 BYTE 입니다.

```
#include <EEPROM.h>

int a = 0;
int value;

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

void loop()
{
    value = EEPROM.read(a);

    Serial.print(a);
    Serial.print("\t");
    Serial.print(value, DEC);
    Serial.println();

    a = a + 1; // a 변수값 증가.

    if (a == 512) // a 변수의 값이 512 인 경우 a 변수의 값을 0 설정
        a = 0;

    delay(500); // 0.5 초 실행 지연.
}
```

## 5.4 ADC (ANALOG DIGITAL CONVERTER)

ADC (Analog Digital Converter)는 아날로그 신호를 디지털 신호로 변환해주는 역할 또는 장치입니다. A/D Converter 로도 표기 됩니다.

빛, 온도, 소리, 압력 등의 변화 값들이 전압으로 바뀌면, 전압은 다시 기준 전압에 의해 일정 범위의 디지털 값으로 변경할 수 있습니다. 아날로그 입력->샘플링 전압 변환->디지털 신호 출력 개념입니다. ADC 의 기능이 좋을수록 고비용의 MCU 가 되기도 합니다.

보통 MCU 에서의 ADC 입력 포트는 여러 개입니다. MCU 의 ADC 기능에서는 위와 같은 과정을 여러 개의 아날로그 입력 채널의 값을 하나의 ADC 에서 처리해 주어야 하므로 멀티플렉싱, 멀티플렉서 회로도 들어가 있습니다.

ADC 기능의 속도와 정확도가 좋을수록 정확한 데이터 출력을 당연한 사항입니다.

속도는 입력된 아날로그 값(신호)을 디지털로 변환하는 시간, 정확도는 아날로그 입력 전압의 범위의 세밀한 단계를 얼마만큼의 큰 세밀한 디지털로 변환하는가에 있습니다. 정확도는 분해능(Resolution)이라고 합니다.

한가지 중요한 부분은 ADC에서의 변환 과정 중에도 일정 시간 소요가 됩니다. 소요 시간 내에 기준전압이 일정치 않고 널뛰는 증상에서는 정확하지 않으므로 고정(HOLD) 개념의 회로도도 적용되어 있습니다.

아두이노 보드에서 각종 센서 모듈을 사용 할 수 있는 이유는 ADC 기능이 있기 때문입니다.

여러 개의 아날로그 입력 처리시 약간의 지연 현상이 발생 될 수도 있습니다.

ADC에서 사용되는 기준 전압은 내부 MCU의 사용전압입니다. ADC의 처리 결과는 아래와 같은 식으로 지정 되어 있습니다.

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}}$$

즉,  $ADC = (V_{in}) * 1024 / V_{ref}$  입니다.

1024는 10비트입니다. ATMEGA328P의 ADC 분해능은 10비트입니다.

ADC 기능은 자체적인 온도 상승, 또는 여러 요소에 의해 정밀 오차를 방지 하여 사용하기 위한 AREF 같은 외부 참조 전압을 사용 할 수도 있습니다.

MCU 프로그래밍에서는 ADC 설정 레지스트리에 기준전압 사용 또는 외부 참조전압을 사용하도록 설정할 수 있습니다.

아두이노에서는 analogReference() 함수를 사용하여 지정할 수 있습니다.

## 5.5 AREF (ANALOG REFERENCE VOLTAGE)

AREF 아날로그 외부 참조 전압 포트입니다.

ADC(Analog Digital Converter)의 기본 기능은 아날로그로 입력되는 신호(전압)를 디지털로 변환하고 있습니다.

아날로그의 범위는 무한대일수 있으므로 기준 전압을 바탕으로 입력되는 아날로그 신호를 디지털 값으로 변경할 수 있습니다.

AREF는 ADC에서 기준으로 참조하는 전압을 입력하기 위한 포트입니다.

즉, A/D Converter에서 참조하여 사용되는 아날로그 외부 입력 전압 포트(핀)입니다. MCU는 기본적으로 내부 MCU에서의 AREF는 좀더 정밀한 아날로그 측정값을 위한 사용자에게는 중요한 부분이기도 합니다. 각각의 포트에 대한 아날로그 입력 전압을 측정하기 위한 ADC의 기준값이기 때문입니다.

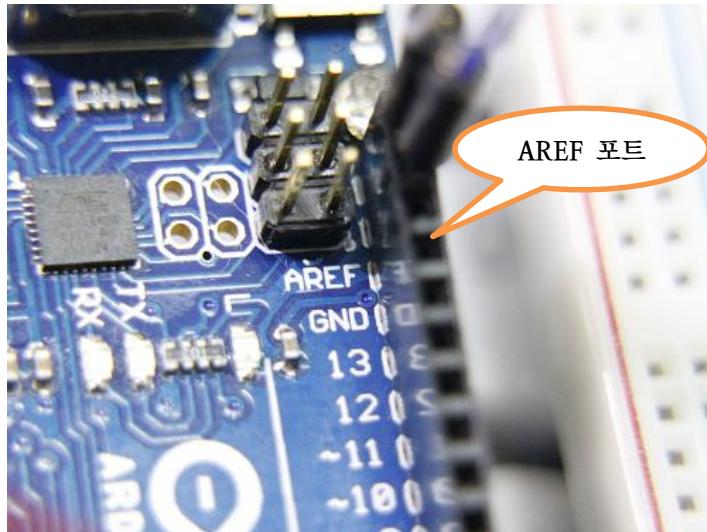


그림 5-14 AREF 포트 위치

아두이노에서 사용되는 ATmega328 MCU의 ADC는 기본 전압을 내부에서 사용되는 기준 전압으로 작동 됩니다.

`analogRead(A0);` 함수를 사용하여 A0 포트의 값을 읽어보면 0~5V 전원 입력하는 경우 0부터 1023으로 나옵니다.

ATmega328P MCU는 내부 기준전압은 5V 기준으로 사용합니다.

하드웨어, MCU 등에서의 기준/기본 전압이라는 용어는 대부분 최대값 기준이라고 보면 됩니다. 즉, 기준전압이라는 말은 최대전압 값이라는 겁니다. 중간 값이나 최소값이 아닙니다. 최대값과 0 까지(Ground, GND)의 기준값이라는 의미입니다.

아두이노 우노 R3는 5V 기본 기준값으로 설정 되어 있습니다.

아두이노의 다른 보드 중 일부는 3V3 (3.3V)로 되어 있는 것도 있습니다.

AREF를 필요로 하는 경우는 그렇게 많지는 않습니다. MCU 자체 또는 사용되는 외부 부품 사용시 온도 변화에 따른 아날로그 값이 변경되는 경우 단독으로 AREF에 전압을 입력하여 주면 오차가 적은 값을 구할 수도 있습니다.

아두이노 우노 R3는 AREF에는 5V가 설정 되어 있습니다. HIGH는 5V로 되어 있는 것도 그 이유입니다. 만약 아두이노 우노 R3 보드가 설계부터 3.3V 사용하는 보드였다면 HIGH는 3.3V입니다.

아두이노에서는 AREF 포트를 사용하기 위한 함수가 지원 되고 있습니다.

`analogReference()` 함수입니다.

함수의 파라 미터는 아래와 같이 여러 종류가 있습니다.

-DEFAULT: the default analog reference of 5 volts (on 5V Arduino boards) or 3.3 volts (on 3.3V Arduino boards)

- INTERNAL: an built-in reference, equal to 1.1 volts on the ATmega168 or ATmega328 and 2.56 volts on the ATmega8 (not available on the Arduino Mega)
- INTERNAL1V1: a built-in 1.1V reference (Arduino Mega only)
- INTERNAL2V56: a built-in 2.56V reference (Arduino Mega only)
- EXTERNAL: the voltage applied to the AREF pin (0 to 5V only) is used as the reference

AREF에 외부 전원 입력을 사용하는 경우 analogReference(EXTERNAL); 라고 해야 합니다. 기본값은 위의 설명대로 DEFAULT입니다.

그리고 AREF에 전원입력을 하는 경우에는 0~5V 까지라고 명시 되어 있습니다. 내부의 ADC의 처리 기준값은 5V입니다. 최대값은 5V입니다.

또 한가지 주의할 점은 AREF에 전압 입력 시 흔들리지 않는 일정한 전압을 사용해야 정확한 수치를 구할 수 있습니다. SMPS 또는 레귤레이터와 커패시터 같은 부품을 사용하여 AREF 포트에 균일한 전압을 입력하여 사용하도록 회로를 구성해야 합니다. 기준 전압이 흔들리는 경우 ADC의 데이터는 무의미하게 됩니다.

## 5.6 PWM 포트

### PWM – Pulse Width Modulation – 펄스폭 변조.

펄스폭 변조 발생시켜 디지털 출력으로 0과 1 출력을 아날로그인 것처럼 출력할 수 있습니다. 일종의 디지털 전압이라고도 볼 수 있습니다. PWM은 MCU의 사용 용도에 따라 계측, 통신에서의 전력 제어, 변환까지 광범위하게 사용되고 있습니다.

아두이노 프로젝트에서 가장 많이 사용되는 용도는 DC 모터의 속도 제어, LED 등의 광량 조절하는데 많이 사용되고 있습니다.

PWM은 디지털 신호 HIGH와 LOW 상태의 지속시간을 변화시켜 전압을 변환합니다. 파형에서의 지속 되는 구간은 뉴티(Duty)라고 합니다.

아두이노에서의 PWM 기능은 통신, 계측, 서보 모터, DC 모터 제어 등에 유용하게 사용됩니다. 아두이노에서는 복잡한 타이밍과 인터럽트 등을 일부러 사용하지 않아도 1개의 함수만 사용하면 PWM 기능을 바로 사용 할 수 있습니다.

analogWrite 함수입니다. analogWrite 함수의 파라미터로 PWM 수치를 변경하여 전압 조절이 가능합니다.

아두이노 우노 R3 보드에서는 디지털 포트로 사용되는 14개 중 PWM 지원 포트가 있습니다.

6개 포트를 사용할 수 있습니다.

~3, ~5, ~6, ~9, ~10, ~11 출력 핀으로 analogWrite() 함수를 사용 할 수 있습니다. PWM 지원 포트(핀)에 256개의 (0부터 255) 범위의 값을 출력 할 수 있습니다.

analogWrite 함수 파라미터 255는 절대적인 HIGH입니다. analogWrite(3,255); 와 digitalWrite(3,HIGH); 함수와 같은 기능이 되게 됩니다.

보드에 표시된 포트 명칭의 (~) 표시는 PWM 기능 표시입니다.



그림 5-15 PWM 포트 표시 설명

함수 `analogWrite` 설명:

`analogWrite(pin,value)`

pin: 포트 번호

value: duty cycle 값: 0 ~ 255

`analogWrite` 함수는 256 개의 값을 사용 합니다. (0 ~ 255). 255 는 HIGH 입니다. 최대 전압입니다.

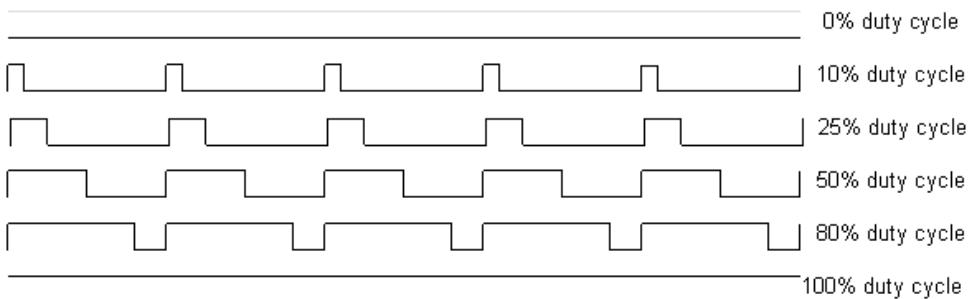
PWM 용도로의 사용시 `analogWrite` 함수는 3,5,6,9,10,11 포트에만 적용됩니다.

물론 ~3, ~5, ~6, ~9, ~10, ~11 핀도 디지털 핀 방식으로 읽고 쓰기를 할 수 있습니다. `digitalWrite0`, `digitalRead0` 함수 사용시 아날로그 범위 값은 무시되고 0, 1 값 을 읽고/쓰기 됩니다.

## 5.7 PWM 포트 이해

PWM 포트 사용시 `analogWrite` 함수 사용시 0 퍼센트 또는 25 퍼센트, 80 퍼센트로 설정 하는 경우 아래의 함수를 사용합니다.

`analogWrite (3,0);`   `analogWrite (3, 255 * 0.25);`   `analogWrite (3, 255 * 0.8 );`  
위 함수 사용시 3 번 포트의 HIGH, LOW 는 아래의 그림처럼 변동됩니다.

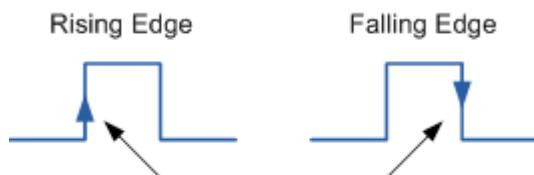


`analogWrite` 함수 파라미터로 255 값이 사용될 경우 5V에 대한 100% 이므로 5V 출력됩니다. 64라는 값이 사용된다면  $255 * 0.25 = 64$  (63.75 반올림) 5V에 대한 25퍼센트이므로 1.25V 출력됩니다.

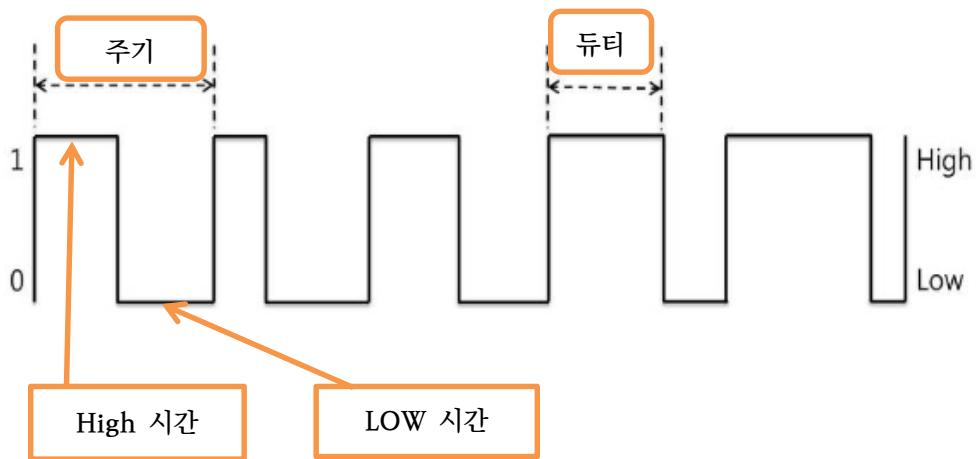
HIGH, LOW를 시간 X축, Y축을 전압(HIGH, LOW)보면 아래와 같습니다.  
파형이라고 볼 수 있습니다. 파형이란 시간 또는 거리에 대한 양의 변화를 형상과 모양으로 나타낸 것입니다.  
PWM에 적용되는 전압 파형(신호)을 아래의 그림으로 표현 될 수 있습니다.  
파도 형태와 비슷합니다. 주파수(frequency)라고도 합니다.



주기는 HIGH 신호를 기준으로 잡습니다. 즉, Rising edge (올라가는 형태)와 다음 올라가는 형태의 신호가 보이는 구간이 주기입니다. 내려가는 형태는 Falling edge입니다.



Duty(듀티)는 HIGH, LOW 지속되는 구간을 뷰티라고 합니다.  
듀티비율의 차이에 의해 전압의 변화가 생기게 됩니다.



아두이노에서의 PWM 주기는 490Hz로 발생되고 있습니다.

조금 더 정확히 말하자면 아두이노의 PWM 주기는 1초에 490회 진동하고 있습니다.  
진동은 HIGH, LOW 왕복입니다.

아두이노에서의 PWM 주기는 고정 되어 있습니다.

1초에 490번 진동하게 되어 있습니다.  $1/490 = 0.0020408163265306$  sec 입니다.

0.002초 정도에 1번 진동 합니다.

HIGH 1, LOW 0으로 가정합니다.

그럼 0.002초에 1회 진동을 합니다. 즉, 0.002초에 1회 HIGH, LOW 반복됩니다.

그럼 0.002초동안의 HIGH 상태, LOW 상태의 시간의 비율을 변경하면 전압의 평균 값이 나오게 됩니다.

0.002초 시간에 지속적으로 HIGH가 되어야 5V입니다.

0.002초 동안의 60프로는 HIGH, 40퍼센트는 LOW로 신호를 주게 되면 현재 전압의 평균값이 60퍼센트가 됩니다.

#### » 헤르츠란? (Hz)

1초에 진동되는 횟수를 말하고 있습니다. 1 헤르츠, 1Hz는 1초에 1회 진동을 말하고 있습니다. 여기서의 진동은 왔다/갔다 왕복입니다.

아두이노에서의 PWM 주기는 490Hz 이므로 1초에 490회 진동을 한다는 의미입니다.

## 5.8 PWM 사용 예제

`analogWrite` 함수를 사용하여 가장 직관적이 아두이노 예제는 아날로그 포트에 가변 저항을 연결하여 읽어 들이는 값의 비율을 변화시켜 `analogWrite` 함수를 사용합니다..

가변저항 10K 와 LED (5 pi 크기) 예제

출처: <http://arduino.cc/en/Tutorial/AnalogInput>

가변저항 10K 와 LED 를 연결합니다.

LED 는 13 번 핀에 연결, 가변저항의 A0 (아날로그 입력 포트 1 번째)에 연결합니다.

가변저항을 돌려서 LED 의 밝기 조절할 수 있습니다.

가변저항의 값을 읽어서 PWM 포트로의 0~255 값 출력 예제 코드입니다.

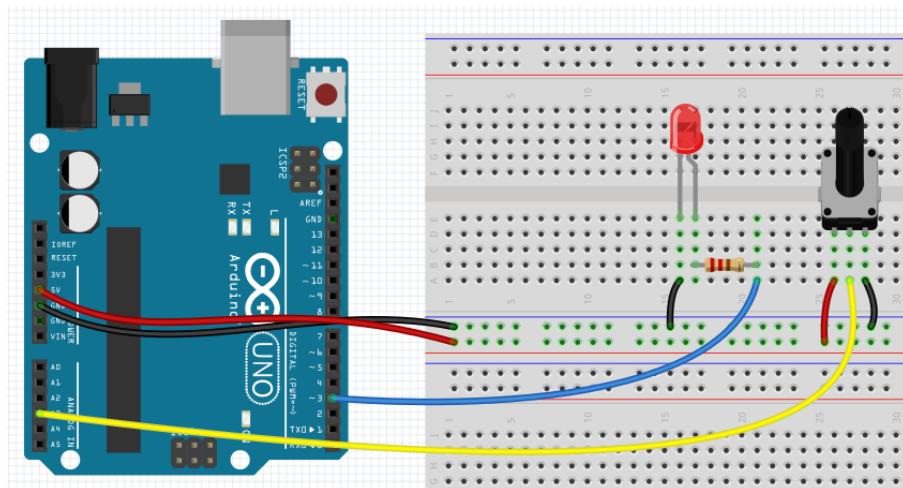


그림 5-16 브레드보드 구성 회로도

예제코드)

```
int sensorPin = A3;      // 아날로그 포트 A3 에 가변저항 출력 연결
int ledPin = 3;          // LED 3 번 핀에 연결. PWM 가능 포트.

int sensorValue = 0;     // 가변저항에서 입력된 값 저장 변수.

void setup()
{
    pinMode(ledPin, OUTPUT); // 3 번 포트는 출력 전용으로 설정.
}

void loop()
{
    // read the value from the sensor:
    sensorValue = analogRead(sensorPin);

    // 읽어들인 값을 pwm 범위 값으로 변환.
    // analogRead 는 0~1023 이므로 pwm 포트에 출력 가능한
    // 0~255 사이의 값으로 변경을 하는 함수입니다.
    int ledValue= map(sensorValue, 0, 1023, 0, 255);
```

```
analogWrite(ledPin, ledValue);

delay(100);           // 0.1 초 DELAY
}
```

## 5.9 PWM 직접 구현 예제 코드

정밀하지는 않지만 delayMicroseconds 0, digitalWrite(포트, HIGH or LOW)를 사용하여 PWM과 비슷한 프로그래밍을 하여 LED의 밝기를 조절 할 수도 있습니다.

```
void setup()
{
    pinMode(13, OUTPUT);
}

void loop()
{
    digitalWrite(13, HIGH);
    delayMicroseconds(100); // 대략 10% duty cycle @ 1KHz 지연
    digitalWrite(13, LOW);
    delayMicroseconds(1000 - 100);
}
```

함수 delayMicroseconds(지연시간);

함수의 파라미터 지연시간은 delay() 함수와 다릅니다.

시간 표시 단위는 1초를 기준으로 다음과 같이 표시 가능합니다.

1초는 1000 밀리초(밀리세컨드)입니다. 1 sec = 1,000 ms;

1 밀리초는 1000 마이크로초(마이크로세컨드) 1 ms = 1,000 us;

다시 1초는 1,000,000 us입니다.

그럼 위에 코드의 delayMicroseconds(100);

100 us (100 마이크로초) 지연 하라는 의미가 됩니다.

함수의 100 아규먼트는 100us 의미입니다. 그럼, 1초를 마이크로초(us)로 변환하면 1,000,000 us입니다.  $100 / 1,000,000 = 0.0001$  sec입니다.

100 us는 0.0001초입니다 0.0001초동안 지연시키게 됩니다.

PWM 변경 및 상세한 정보는 아래의 아두이노 사이트를 보시기 바랍니다.

<http://arduino.cc/en/Tutorial/SecretsOfArduinoPWM>

» C/C++ 함수의 파라미터(Parameter)는?

단어 자체의 의미는 매개변수입니다. 함수를 정의할 때 외부로부터 받아들이는 임의의 값을 말합니다.

아래와 같이 myFunction 함수에 파라미터 int x, int y 입니다.

함수 실행 후 반환되는 값은 int 라는 정수입니다.

“정수형 반환 함수 마이펑션(정수형 x 라는 이름을 가진 매개 변수, 정수형 y 라는 이름을 가진 매개변수)”

```
int myFunction(int x, int y)
{
    return x+y;
}
```

풀어서 말하면

“정수형의 값을 반환하는 myFunction 이 있습니다.

myFunction 함수는 정수형 x 와 y 라는 이름의 매개변수를 정의합니다.”

» C/C++ 함수의 아규먼트(Argument)란?

Argument에 대한 단순 번역은 인수입니다.

함수에 파라미터로 사용되는 값으로 이해하면 됩니다.

위에서 언급된 myFunction(int x, int y); 형태를 가진 함수를 코딩에서 사용하는 경우 int addValue = myFunction(4,6); 이라고 사용할 수 있습니다.

그럼 addValue 는 10 이 됩니다.

여기서 사용된 4 와 6 이라는 값은 아규먼트(인수)입니다.

## 5.10 PWM 포트 사용시 주의점

PWM 기능으로 사용하는 포트는 MCU 의 내부 하드웨어 타이머에 의해 적용되고 있습니다.

아두이노 우노 R3 에서 사용하는 ATMega328 MCU 에는 3 개의 하드웨어 타이머가 있습니다.

아두이노 우노 R3 부트로더에서는 하나의 타이머마다 2 개의 PWM 포트를 연동하도록 되어 있습니다.

한 개의 타이머에는 2 개의 PWM 포트를 연동하게 되어 있습니다.

그리고 3 개의 타이머는 조금 다른 Frequency로 설정 되어 있습니다.

```
void setup()
{
    pinMode(5,OUTPUT);
    pinMode(11,OUTPUT);

    analogWrite(5,200); // 5 번 pwm ; 200 8Pvd.
    analogWrite(11,200); // 11 번 pwm ; 200 8Pvd.
}

void loop()
{
```

위의 코드를 업로드 후에 5 번핀, 11 번 핀의 전압을 테스터기로 측정하면 약간의 전압 오차가 있습니다.

255 값이 5V로 보았을 경우 200 의 출력 값은 4V입니다.

그럼 균등하게 5 번핀과 11 번 핀의 출력 전압은 4V로 되어야 합니다.

실제 측정된 값은 약간 다릅니다.

아두이노 프로젝트에서 PWM 포트를 2 개 사용하는 경우에는 짹을 지어서 사용해야 합니다.

5,6 번 TIMER0 에 의해 제어됩니다.

9,10 번 TIMER1 에 의해 제어됩니다.

3,11 번 TIMER2 에 의해 제어됩니다.

2 개의 PWM 포트를 사용하는 경우에는 위에 지정된 포트를 사용해야 합니다.

PWM 제어가 필요한 1 개의 모듈을 사용하는 경우 PWM 지원되는 포트, 구분 없이 사용하면 됩니다. 하지만, 2 개의 모듈을 사용하는 경우, 동일한 전압레벨 차이를 이용하는 경우에는 짹지어진 PWM 포트를 사용하면 결과적으로 정확한 제어가 가능합니다.

## 5.11 하드웨어 외부 인터럽트

INT0, INT1 2 개 사용 가능합니다.

하드웨어 인터럽트 포트는 D2, D3 에 연결하여 사용됩니다.



인터럽트 사용은 하드웨어 프로그래밍 요소 중 중요한 부분입니다.

인터럽트(Interrupt)의 기본 개념은 프로세서에서의 처리시 일정한 작동, 조건 발생시 지정된 함수를 호출합니다. 직관적인 이해는 스마트폰에서 문자, 동영상 등을 보는 중간에 전화가 걸려오면 전화 수신 알림이 보입니다. 사용자는 전화 수신, 통화를 합니다. 전화 통화가 끝나면 전에 작동 중이던 문자, 동영상 화면이 다시 나오게 됩니다. CPU에서는 이러한 하드웨어 작동을 레지스트리의 값을 변경, MCU내부에 있는 CPU는 무언가를 계속 처리를 하고 있습니다. CPU에서의 여러 작동 진행상황, 신호 상태 변화, 타이머 작동 등에 대한 모든 인터럽트가 존재합니다.

### 5.11.1 내부 인터럽트

MCU 는 인터럽트 처리를 하고 있습니다. 펌웨어 개발자, 사용자는 ISR(인터럽트 서비스 루틴, Interrupt Service Routine) 함수를 이용하여 인터럽트 처리를 할 수 있습니다. 즉, 사용자가 원하는 형태의 인터럽트 처리를 할 수 있습니다. 아두이노 우노 MCU 를 비롯하여 일반적인 MCU 하드웨어에서의 인터럽트 종류로는 타이머 기능, 리셋 기능, 외부포트 인터럽트와 UART(시리얼포트) 인터럽트가 있다고 보면 됩니다.

아두이노 보드를 사용한 프로젝트에서는 거의 인터럽트는 사용하지 않아도 됩니다. 아두이노 프로그래밍에서는 대부분 사용하지 않아도 됩니다. 인터럽트를 사용하지 않고도 프로젝트의 대부분은 구현 가능하도록 되어 있습니다.  
하지만, 조금 더 정밀한 타이밍을 요구하는 프로젝트일 경우에는 인터럽트를 사용할 수 있습니다.

### 5.11.2 외부 인터럽트 사용

아두이노 우노 R3 의 MCU(ATmega328p-pu)에서는 하드웨어 인터럽트 포트(핀) 2 개 지원 됩니다. 해당 포트는 아두이노 포트의 D2, D3 포트를 사용하여 처리할 수 있습니다. 사용하기 위한 함수도 지원합니다.

즉, 지정된 포트의 신호 레벨이 변경될 때마다, 변경 상황에 맞게 함수를 처리할 수 있습니다.

attachInterrupt(), detachInterrupt() 함수입니다.

D2 포트는 인터럽트 0 번, D3 포트는 인터럽트 1 번으로 명명 되어 있습니다.

자세한 세부 설명은 아두이노에서의 인터럽트에 대한 자세한 정보는 아래의 주소에 있습니다.

<http://playground.arduino.cc/Code/Interrupts>

외부 인터럽트 작동 발생시, 함수를 등록하여 처리할 수 있습니다. 등록과 해제 함수가 지원 됩니다.

인터럽트 등록 함수입니다.

**attachInterrupt(interrupt, ISR, mode)**

interrupt: 인터럽트 포트 번호입니다.

2 개가 있으므로, D2 포트 사용하는 경우 0 을, D3 포트 사용 하려면 1 을 사용합니다.

ISR: 인터럽트 서비스 루틴(Interrupt Service Routine) 함수 명칭을 등록합니다.

Mode: 해당 인터럽트로 지정된 하드웨어 포트의 신호 이벤트 경우를 기입할 수 있습니다.

하드웨어 포트의 신호 이벤트는 LOW, CHANGE, RISING, FALLING, HIGH(아두이노 두에 보드 전용) 사용될 수 있습니다.

필요한 포트의 Mode 에 따라 여러 가지 ISR 함수를 등록하여 처리할 수 있습니다.

인터럽트 해제 함수입니다.

**detachInterrupt(interrupt)**

interrupt: 인터럽트 포트 번호입니다. 2 개가 있으므로 0 번과 1 이 들어갈 수 있습니다. (우노 R3)

아래의 예제 코드는 CHANGE (LOW에서 HIGH, 또는 HIGH에서 LOW) 경우에 blink라는 함수를 실행하는 예제 코드입니다.

```
// http://arduino.cc/en/Reference/AttachInterrupt
int pin = 13;
volatile int state = LOW;

void setup()
{
    pinMode(pin, OUTPUT);
    attachInterrupt(0, blink, CHANGE);
}

void loop()
{
    digitalWrite(pin, state);
}

void blink()
{
    state = !state;
}
```

명확히 이해하기 위해 코드를 아래와 같이 추가 합니다.

CHANGE 인터럽트 이벤트 발생시 실행되는 state 변화 횟수를 저장하여, 시리얼 포트로 출력해주는 예제 코드입니다.

CHANGE 인터럽트 이벤트를 발생시키기 위해서는 D2 포트에 간단한 버튼, 로터리 엔코더 등을 연결하여 HIGH, LOW 변경해 보도록 합니다.

로터리 엔코더는 대부분 시계방향, 또는 반시계방향 변경하면 LH->HH, 또는 HL->LL 등의 신호가 발생됩니다.

```
int pin = 13;
volatile int state = LOW;
int old_state=state;
int count_change=0;

void setup()
{
    Serial.begin(9600);
    pinMode(pin, OUTPUT);
    attachInterrupt(0, blink, CHANGE);
    Serial.println("setup end");
}

void loop()
```

```

{
    digitalWrite(pin, state);
    if( state !=old_state )
    {
        Serial.print("changeed ");
        old_state=state;
        count_change+=1; // count_change = count_change + 1;
        Serial.print("state counting");
        Serial.print("count_change -->");
        Serial.println(count_change);
    }
}

void blink()
{
    state = !state;
}

```

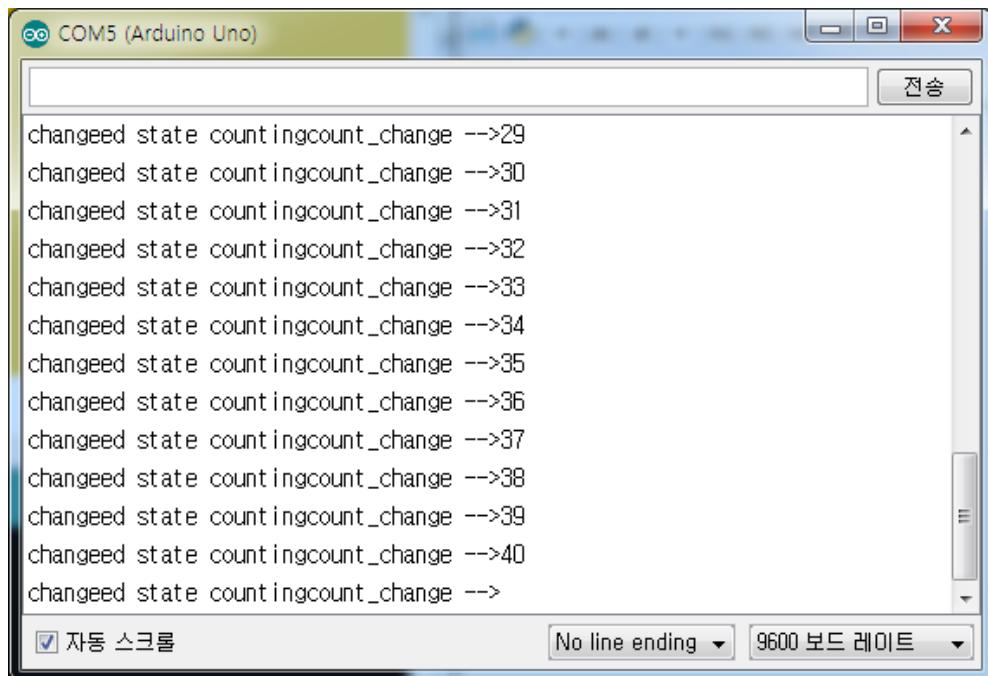


그림 5-18 로터리 엔코더 작동 시리얼 모니터

## 5.12 DC 입력 7 TO 12V DC

DC 전원 입력 포트입니다.

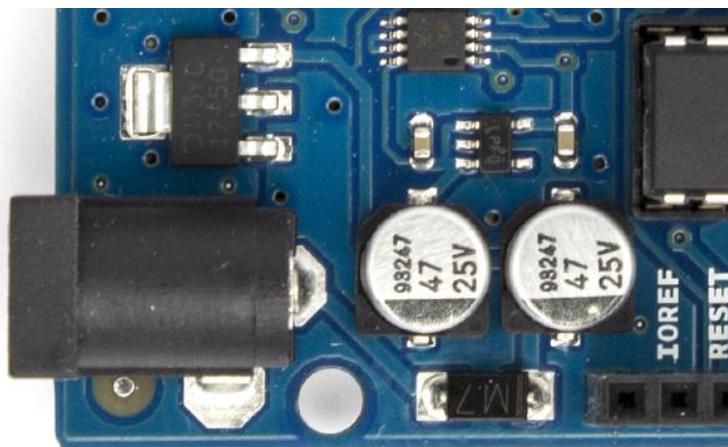


그림 5-19 DC 입력 포트 위치

7v to 12v DC 입력 전원 포트입니다. 보통 가정에서 많이 사용되는 9V, 12V DC 어댑터 그대로 사용하면 됩니다.

DC JACK 규격은 5.5 파이, 내경 2.1 mm 입니다. (Plug diameter 2.1mm ID, 5.5mm OD)

보통 아두이노 우노 R3 의 추천 DC 어댑터는 9V 1A 어댑터, 12V 1A 어댑터를 사용합니다.

물론, PC 와 USB 연결 후 동시에 사용하여도 무방합니다. 동시에 USB, DC 입력되는 경우에는 높은 입력 전압이 사용됩니다.

## 5.13 디지털 포트 & 아날로그 포트

아두이노 우노 R3 버전 디지털 포트와 아날로그 포트 설명 다이어그램입니다.

아두이노 우노 보드는 MCU(Micro Controller Unit) ATMEGA328P 를 사용하고 있습니다.

MCU 의 작동 결과 및 통신은 모두 디지털 입력/출력, 아날로그 입력/출력 기능이 있어야 합니다.

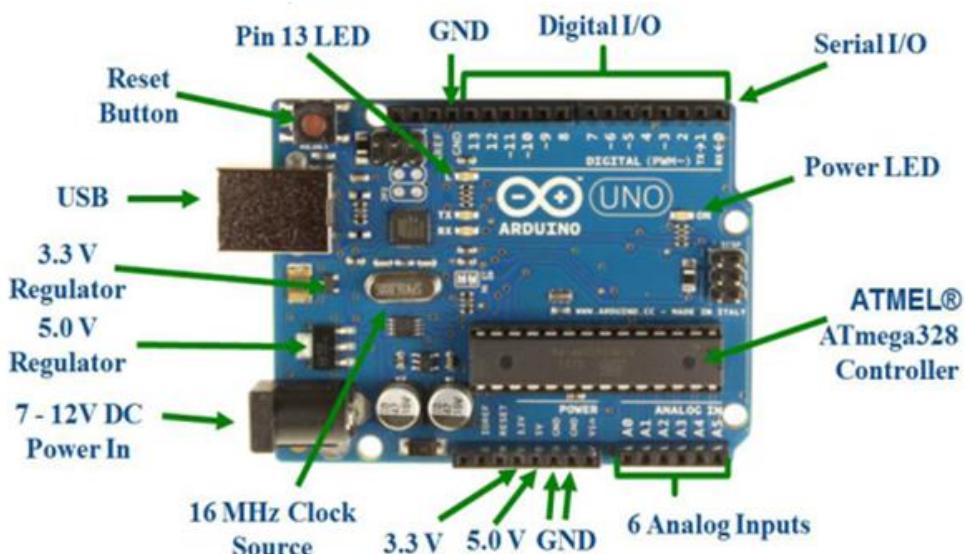


그림 5-20 우노 R3 보드 핀 맵

>> 디지털 In/Out 포트: 0~13, 14 Digital input/output Ports.

디지털 포트는 1, 0의 값을 읽고 쓰기 할 수 있습니다. 0은 LOW, 1은 HIGH  
digitalRead(), digitalWrite() 함수를 사용하여 각각의 포트에 값을 읽기, 쓰기를 합니다.

>> 아날로그 Input 포트: A0 ~ A5, 6 개의 Analog Input Ports.

아날로그 입력 포트입니다. 각종 아날로그 센서들을 장착하여 측정값들을 가져올 수 있습니다. analogRead() 함수를 사용하여 아날로그 입력된 값을 받을 수 있습니다.  
입력되는 측정값들의 범위는 0 ~ 1023 범위(1024 개)입니다.

### 5.13.1 디지털 입력/출력 포트

디지털 Digital Input Output Port 14 개를 사용할 수 있습니다.

GPIO (General Purpose Input-Output) 포트라고도 합니다.

D0 ~ D13 입니다. 14 개입니다.



그림 5-21 우노 R3 디지털 포트

D0, D1은 하드웨어 시리얼 포트로 예약되어 사용됩니다. 물론 다른 용도로의 디지털 포트로 사용 가능합니다.

`digitalRead`, `digitalWrite` 함수와 방향 설정 `pinMode`라는 함수를 주로 사용하여 프로그래밍 할 수 있습니다.

5 번 포트의 값을 읽어 오는 경우에는 아래와 같은 `digitalRead`라는 함수를 사용합니다.

```
int d5_Value = digitalRead(5);
```

반대로 5 번 포트에 1 ( HIGH) 값을 써주는 경우는 아래와 같이 `digitalWrite` 함수를 사용합니다.

```
digitalWrite(5,HIGH);
```

물론 해당 포트로의 읽기/쓰기 함수는 사용되기 전에 `pinMode`라는 함수를 사용하여 포트의 속성을 미리 지정해야 합니다

```
pinMode(5,OUTPUT); // 5 번 포트 출력 모드로 설정.
```

```
int d5_Value = digitalWrite(5,HIGH); // HIGH 신호 설정
```

### 5.13.2 아날로그 입력 포트 (Analog Input Port)

6 개의 아날로그 입력 전용 포트입니다.

A0~A5, 총 6 개의 입력 전용 포트입니다.

즉, 외부로부터의 아날로그 입력만 허용되게 예약 되어 있습니다.

하지만 용도에 따라 출력 방향을 변경하여 사용도 가능합니다.

일반 디지털 포트로의 사용은 `pinMode(A0, INPUT);` 또는 `pinMode(A0, OUTPUT);` 등의 포트 방향 설정을 해주면, 일반 디지털포트, PWM 포트로의 사용이 가능합니다.

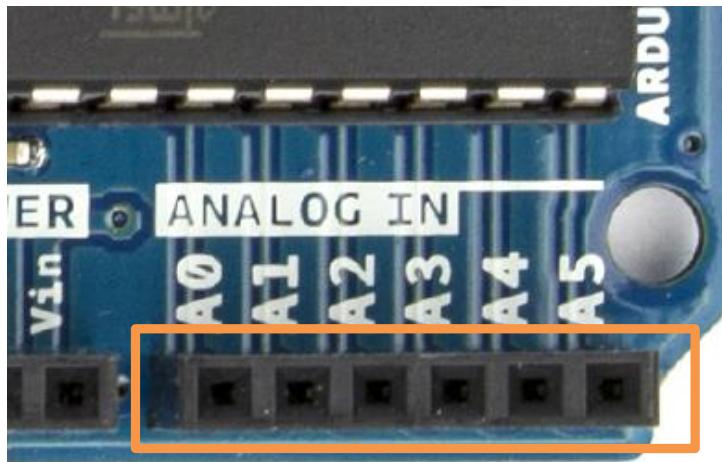


그림 5-22 우노 R3 아날로그 포트 위치

스케치에서의 `analogRead()` 함수 사용시 아날로그 포트 번호로 A0 ~ A5 사용됩니다. 1 개의 아날로그 인풋 핀에서는 1024 개의 아날로그 범위 값을 읽을 수 있습니다. 1024 개의 아날로그 범위 값을 읽을 수 있다는 것은 0 ~ 1023 범위의 integer type(정수) 입니다. C/C++ 에서의 정수의 기본 인덱스는 0 부터 시작되므로 정수로 1024 개는 0~1023 이 됩니다.

아두이노 공식 사이트에서도 상세히 설명 되어 있습니다.

<http://arduino.cc/en/Reference/analogRead>

예제 함수:

// 6 번째 아날로그 포트에서 값을 읽는다.

```
int a5_Value = analogRead(A5);
```

// 1 번째 아날로그 포트에서 값을 읽는다.

```
int a0_Value = analogRead(A0);
```

`analogRead()` 함수는 1024 범위의 값을 반환 합니다. (0 ~ 1023)

가변저항 10K 모듈을 사용하여 아날로그 입력 값을 알아봅니다.

가변저항을 돌려 입력되는 전압의 크기를 수치로 볼 수 있습니다.

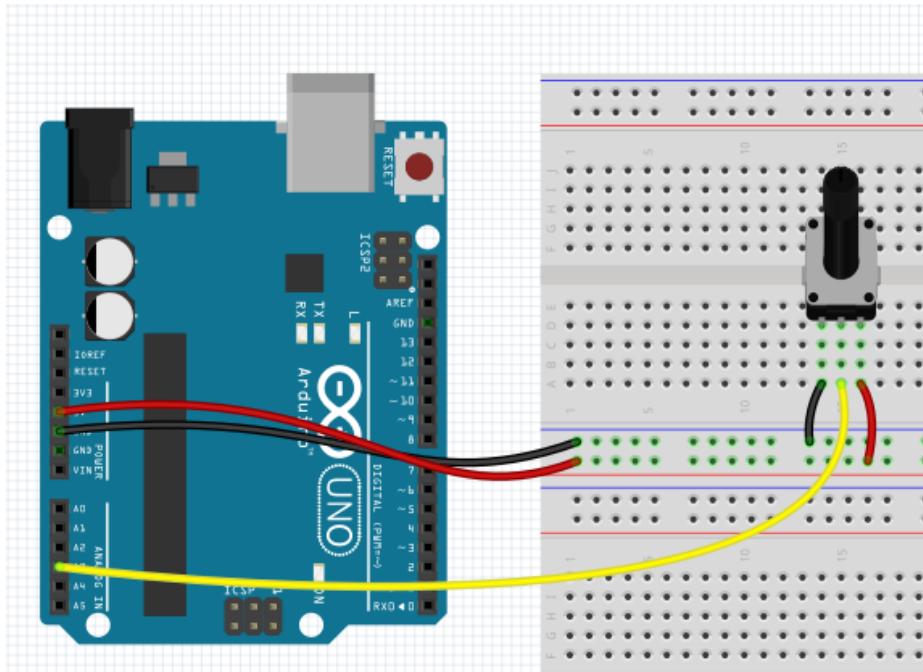


그림 5-23 가변저항 브레드보드 회로도

예제 코드: 가변저항의 값을 읽어서 시리얼 포트로 출력해 줍니다.

```
// 가변저항 10K 사용 예제 코드
int analogPin = 3
int val = 0;

void setup()
{
    Serial.begin(9600);
}

void loop()
{
    val = analogRead(analogPin);
    Serial.println(val);
}
```

## 5.14 디지털 14 핀 & 아날로그 포트 6 포트 사용 정의

디지털 포트와 아날로그 입력 포트는 위에서도 설명 되어 있습니다.

우노 보드에서는 총 20 개의 입/출력 핀들이 제공되고 있습니다.

디지털 14 핀 D0 ~ D13

아날로그 6 핀 A0 ~ A5

만약, 프로젝트 진행 시 디지털 입력/출력 포트가 부족할 경우에는 A0~A5 포트도 디지털 포트로 속성을 변경하여 입력/출력으로도 사용 가능합니다.

pinMode 라는 함수를 사용합니다.

```
void pinMode(pin, mode)
```

pin 값으로 디지털 핀들은 0 ~ 13 값으로 할당됩니다.

아날로그 A0 ~ A5 핀들은 14 ~ 19로 사용됩니다.

A0 와 A5 는 상수로 정의되어 있어 mode 파라미터 값으로 같이 사용할 수 있습니다.

A0 혹은 14 를 사용하여도 같은 결과입니다.

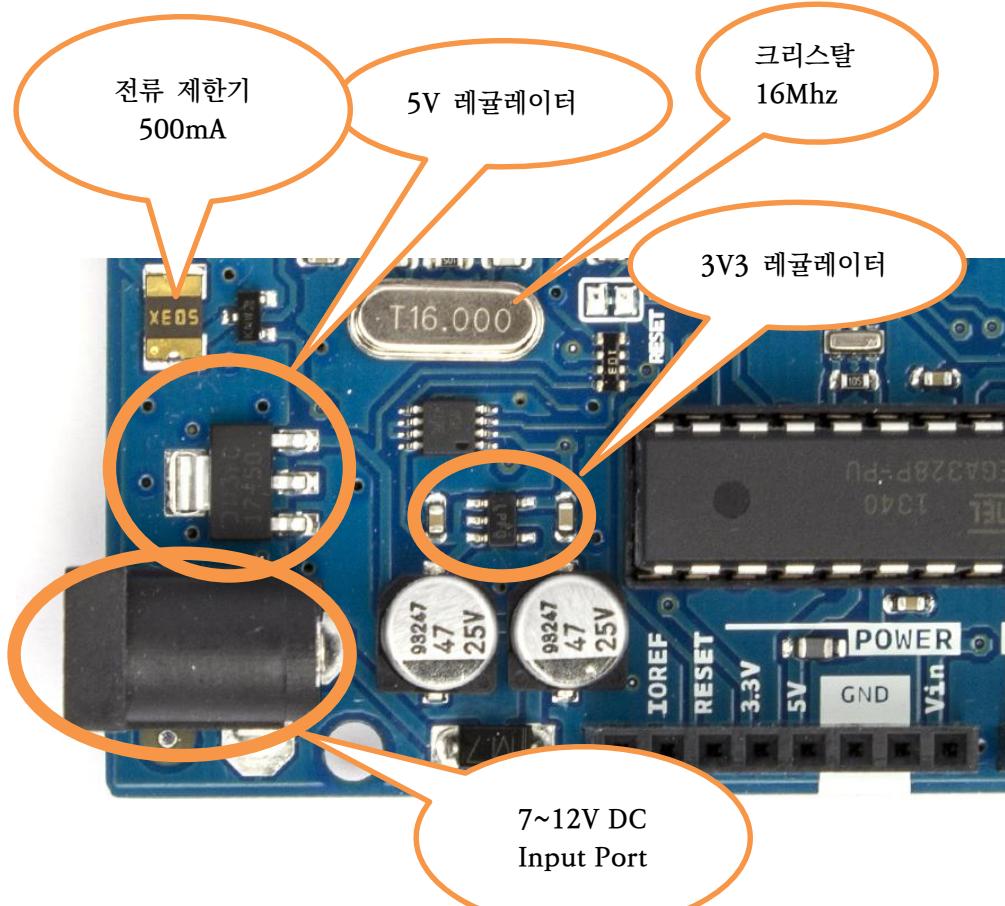
```
void setup()
{
// 아날로그 포트 A0 포트를 디지털 i/o 포트로 사용.
//
// 아래 2 줄의 코드는 같은 의미입니다.
pinMode(14, OUTPUT); // 인덱스 번호로 대입
pinMode(A0, OUTPUT); // 정의된 명칭으로 대입

}

void loop()
{
// 아래 2 줄의 코드는 같은 의미입니다.
digitalWrite(14,HIGH);
// 또는
digitalWrite(A0,HIGH);
}
```

## 6 아두이노 보드의 전압, 전류 입, 출력 정보

아두이노 우노 R3 보드에는 7~12V 입력 가능한 레귤레이터가 사용되고 있습니다.  
DC Jack을 통한 입력 가능 DC 크기는 7~12V입니다.



아두이노 우노 R3에서 7~12V 입력 가능하다고 명시된 이유는, 사용되는 레귤레이터 부품의 영향을 받고 있습니다.

<http://www.arduino.cc/en/Main/arduinoBoardUno> 사이트에 소개되고 있는 아두이노 우노 R3 보드의 하드웨어 기본 기술 사양입니다.

Microcontroller ATmega328

Operating Voltage 5V

Input Voltage (recommended) 7-12V

Input Voltage (limits) 6-20V

Digital I/O Pins 14 (of which 6 provide PWM output)

Analog Input Pins 6

DC Current per I/O Pin 40 mA

DC Current for 3.3V Pin 50 mA

Flash Memory 32 KB (ATmega328) of which 0.5 KB used by  
bootloader

SRAM 2 KB (ATmega328)

EEPROM 1 KB (ATmega328)

Clock Speed 16 MHz

Length 68.6 mm

Width 53.4 mm

Weight 25 g

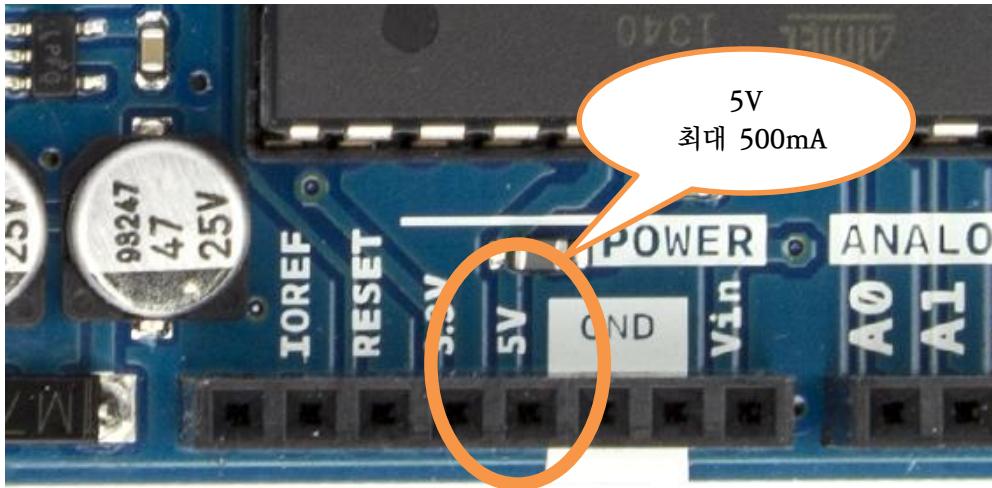
DC Current per I/O Pin 40mA 라는 문구가 보입니다. 입력과 출력으로 사용되는 핀은 40mA (40미리암페어)라고 나와 있습니다. 최대 40mA 출력이라는 뜻입니다. 항상 40mA 출력은 아닙니다.

DC Current for 3.3V pin 50mA 라는 문구는 3.3V 출력에 사용되는 핀은 50mA 입니다.

의아스러운 부분은 아두이노 우노 R3 보드의 5V 전원 출력 핀에 대한 암페어 값이 나와있지 않습니다.

사용되는 부품 중 전압과 전류에 관련된 레귤레이터의 데이터 쉬트를 찾아보면 7805 등의 5V 레귤레이터 부품들은 최대 5V 800mA 의 전압과 전류가 나옵니다.

또한, 아두이노 우노 R3/메가2560 등의 보드에는 레귤레이터 부품과 함께 전류 제한 부품도 같이 붙어 있습니다. 전류 제한은 500mA 까지 제한하고 있습니다.



아두이노 보드에서의 5V 출력 포트는 레귤레이터를 통해서 나오는 전압과 전류입니다. 레귤레이터를 통하여 나오는 전압은 동일하지만, 전류의 크기는 다를 수 있습니다. 실제 USB 연결 상태에서의 5V 출력과 전압, DC 9V 1A 어댑터 연결, DC 9V 배터리 연결 후, 전류를 측정하여 보면 전류의 크기는 다르게 나옵니다. 물론, 최대 500mA 정도의 전류가 나올 수 있습니다.

## 7 아두이노 시작하기

“아두이노 우노 R3”

아두이노를 시작하면서 가장 많이 사용되는 마이크로 컨트롤러 보드입니다.

여러 종류의 보드들이 있지만, 직관적인 사용이 가능한 전원 입력과, 포트의 배열, 전원 출력 연결의 편리성 등에 의해 가장 많이 사용되는 보드중의 하나입니다.

그리고, USB 케이블만 연결하여 프로그래밍과 전원공급 2 가지 모두 해결 됩니다.

기존 AVR 프로그래밍 프로토타입 보드(선행 개발 보드)는 전원과 프로그래밍 포트가 분리되어 있는 형태가 대부분이었습니다.

☞ 3 아두이노 우노 R3 보드를 사용하기 위해서는 PC 에 USB 연결 후 사용합니다.



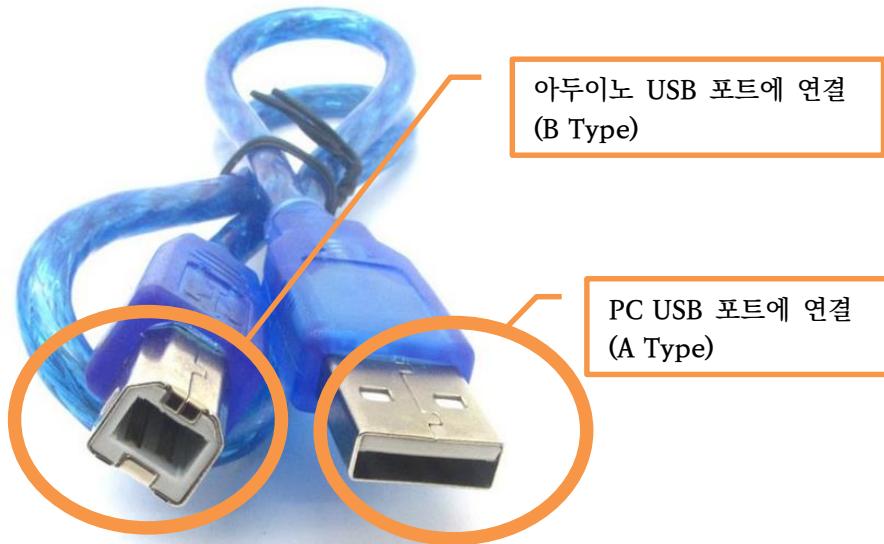
그림 7-1 아두이노 우노 R3 보드 & USB B Type 케이블

PC에서 사용하기 위해서는 드라이버를 설치해 주어야 작동됩니다.

아두이노 우노를 비롯하여 대부분의 아두이노 보드들은 PC 와 연결 시 별도의 전원이 없이 USB 케이블 연결로만 작동 가능합니다.

USB 케이블을 통해 시리얼 통신 신호와 더불어 전원도 공급 받아서 사용됩니다.

사용되는 USB 케이블은 B-Type USB 연결 케이블을 사용합니다.



USB Type 은 용도에 따라 여러 종류가 있습니다. 아두이노 우노 R3 보드는 B Type입니다.

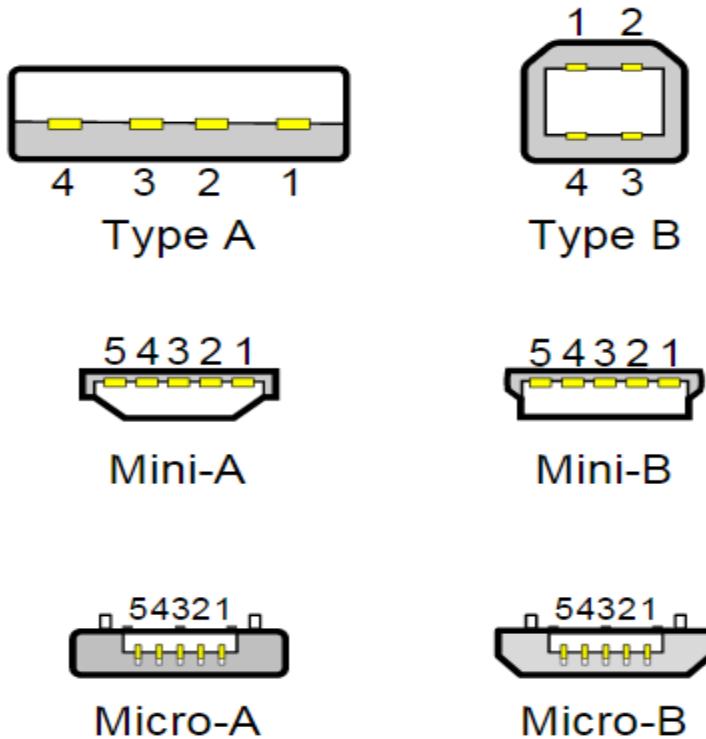
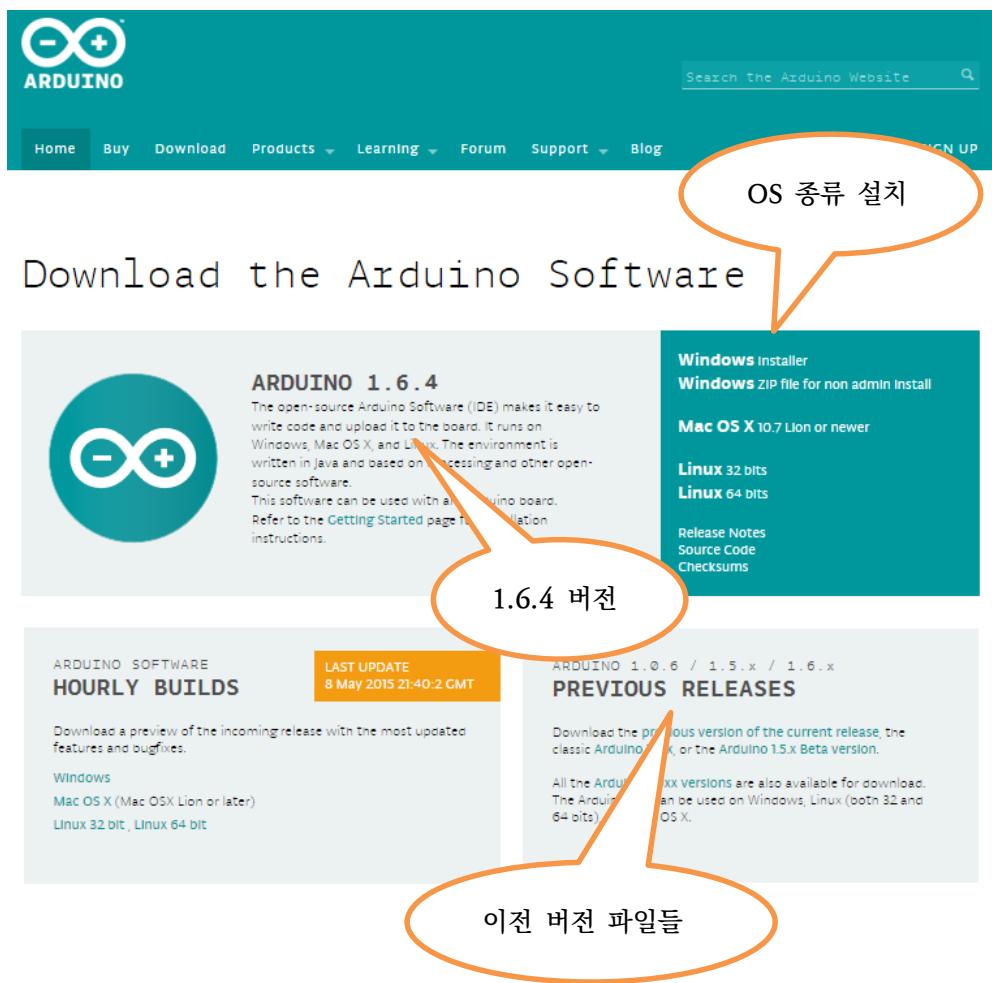


그림 7-2 USB Connector Type

## 7.1 아두이노 IDE 스케치 프로그램 다운로드

드라이버 파일은 아두이노 IDE 스케치 프로그램에 포함 되어 있습니다.  
다운로드 주소는 <http://arduino.cc/en/Main/Software> 입니다.

공식 사이트에서 배포되는 버전은 1.0.6 버전과 새로운 최신 버전, 2 종류가 있습니다.  
원하는 버전을 다운로드 받아 사용하면 됩니다.  
(현재 새로운 버전은 1.6.4입니다)



배포되는 형식은 프로그램 설치 형식(Windows Installer), 압축 파일 다운로드 후 해제하여 사용되는 방식(Windows ZIP file), 2 가지입니다.  
여기서는 압축 파일 다운로드 형식의 파일(Windows ZIP file)을 사용합니다.

### » Zip 파일이란?

ZIP 파일은 여러 개의 파일들이 압축되어 있는 형태입니다.

압축되어 있는 파일은 “알집”, “WinZip”, “7-Zip” 등으로 풀어서 사용해야 합니다.

압축 파일의 형태 중 가장 많이 사용되고 있습니다.

ZIP 파일을 사용하는 상용, 공개 형태의 많은 프로그램들이 있습니다. ZIP 파일 다루는 C/C++, 또는 다른 언어 형식으로도 공개 소스가 많이 있습니다.

### » 디렉터리, 폴더는?

참고로 PC 사용에 관한 여러 글들을 접하다 보면 디렉터리, 폴더라는 용어가 있습니다.

같은 의미입니다.

디렉터리는 초기에 지정된 명칭입니다.

콘솔 방식(명령어를 손으로 타이핑 입력하여 사용하는 방식)의 리눅스/도스/터 오래된 시스템에서 사용되던 용어입니다.

컴퓨터 사용자 환경이 GUI 시대가 오면서 폴더라는 용어가 나오게 된 것입니다. 의미는 같다고 보시면 됩니다. 물론 지금은 윈도우를 비롯하여, 리눅스, 스마트폰도 지금은 모두 GUI 방식입니다.

다운받은 아두이노 IDE 스케치 프로그램 파일을 원하는 디렉터리에 풀어 놓으면 하위 디렉터리의 `\Arduino\arduino-1.5.8-windows\arduino-1.5.8\drivers`라는 디렉터리에 관련된 파일들이 있습니다.

또는 설치 형식으로 받은 경우에는 “`C:\Program Files\Arduino`” 디렉터리 아래에 설치 됩니다. 또는 원하는 위치의 디렉터리를 지정하여 설치를 합니다.

### » 아두이노 스케치 디렉터리 지정.

아두이노 설치 및 사용자 라이브러리 디렉터리는 별도로 지정하기 권장합니다.

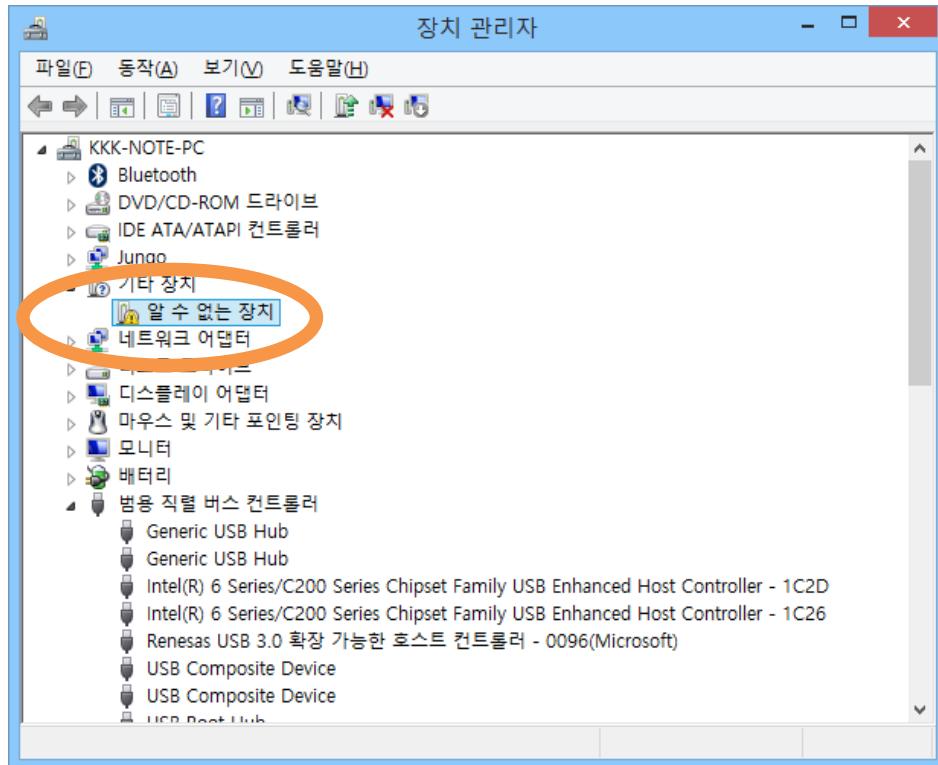
아두이노 스케치 IDE 사용하는 경우에는 많은 모듈, 기기들을 사용하게 됩니다. 해당 모듈, 기기에는 아두이노에서 사용 가능하게 하는 라이브러리들이 있습니다. 라이브러리들이 많아 지는 경우 업로드 시 컴파일, 링크 과정에서 에러가 있을 수 있습니다. 되도록 같은 종류의 라이브러리는 1 개만 사용하면 라이브러리 헤더 파일 중복에 의한 에러는 없습니다. 에러 발생시 프로젝트 진행 및 디버깅에 많은 어려움이 있을 수 있으니 반드시 정리하시기 바랍니다.

## 7.2 아두이노 보드 드라이버 설치

아두이노 보드와 PC의 USB 포트에 처음 연결 시 드라이버 설정이 안되어 있을 경우 드라이버를 설치 해주어야 합니다.

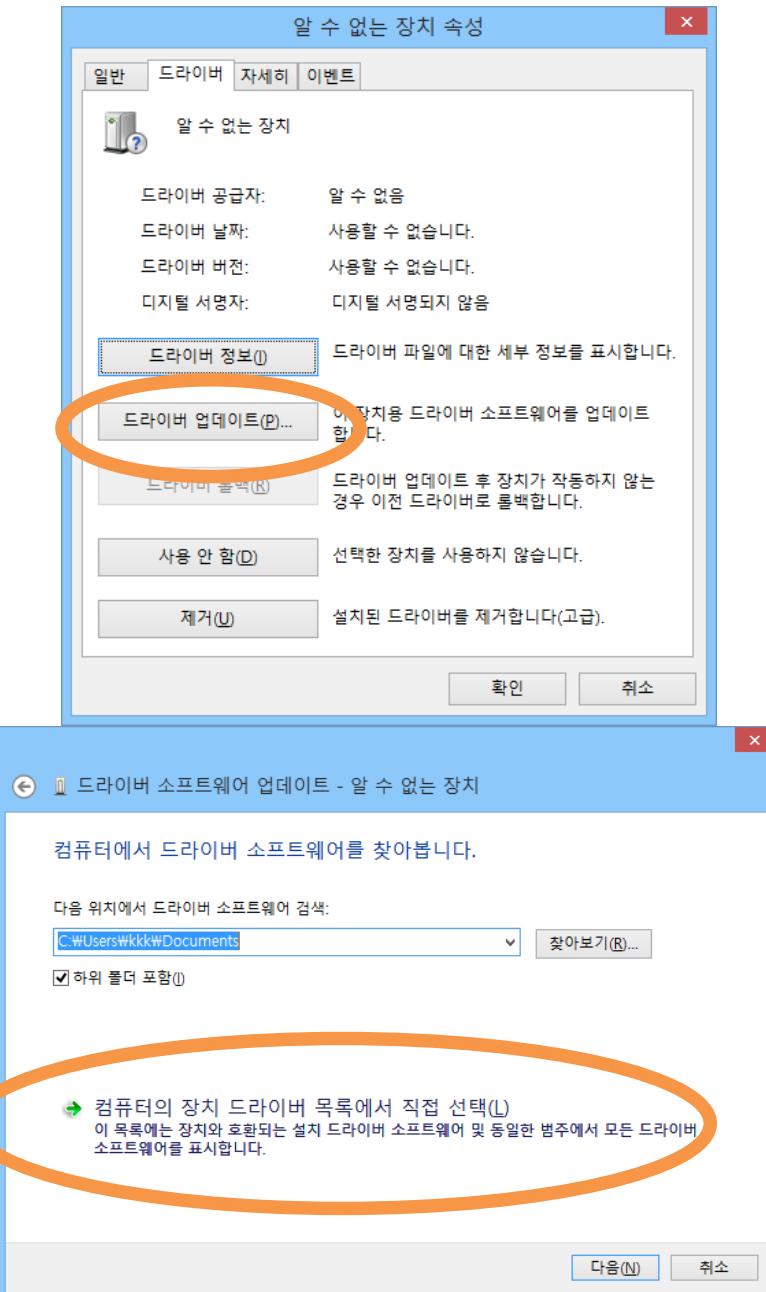
PC USB 포트에 연결 후 “장치 관리자” 창을 열어서 보면 아래와 같은 기타장치에 “알 수 없는 장치” 명칭으로 표시되고 있습니다.

또는 “ATMEGA16U2” 명칭으로 표시될 수 있습니다.



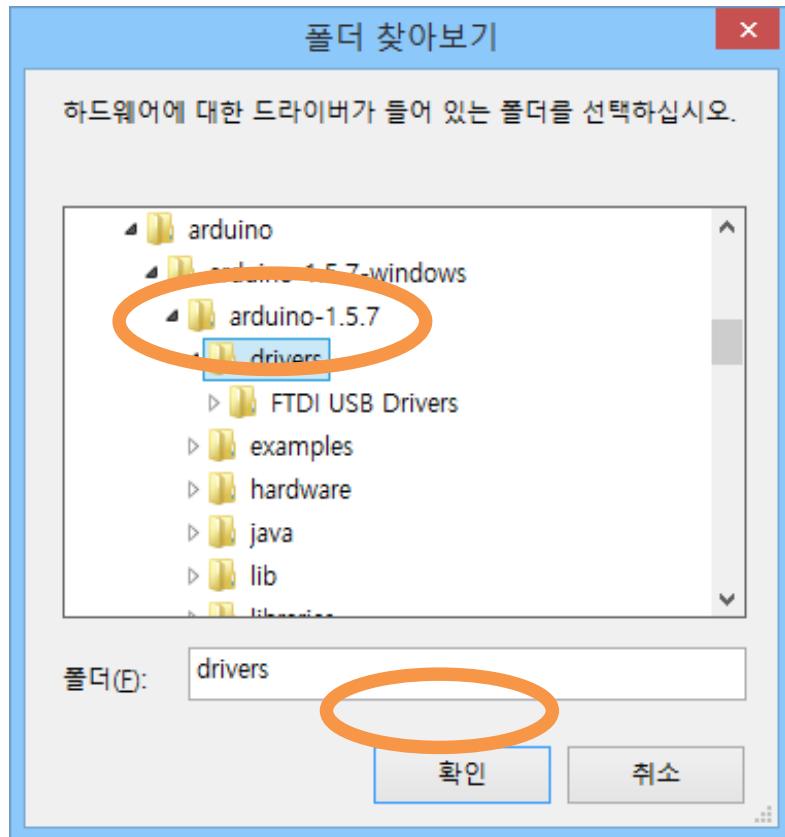
“알 수 없는 장치” 텍스트 부분을 더블 클릭 또는 속성 보기 를 하면 아래와 같은 대화 정보 창이 나옵니다.

“드라이버 업데이트” 버튼을 눌러서 드라이버 파일이 있는 디렉터리를 지정 해줍니다



“컴퓨터의 장치 드라이버 목록에서 직접 선택” 버튼을 누르게 되면 아래와 같은 팔더 찾아보기ダイアログ가 나옵니다.

위에서 아두이노 IDE 스케치 프로그램 다운로드 후 압축 해제 (또는 설치된 디렉터리) 디렉터리의 아래의 “drivers” 지정하여 줍니다.



정상적으로 드라이버 찾고 업데이트 되는 경우 위와 같은 업데이트 성공 메시지가 보이게 됩니다.

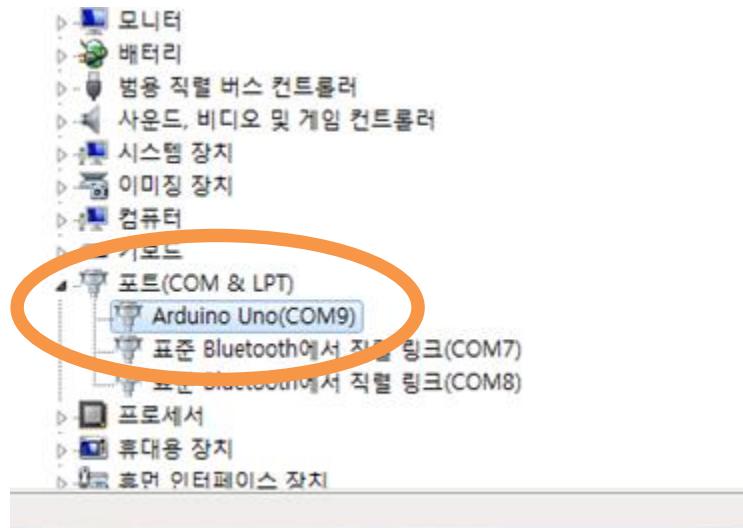


그림 7-3 드라이버 정상 적용된 상태의 제어판의 포트

드라이버 적용 완료 후 “장치 관리자” 포트의 하위 목록에는 “Arduino Uno(COM9)”라고 보이게 됩니다.

괄호 안의 COM9 표시는 시리얼 포트 장치 번호 명칭입니다. 사용자 PC 의 환경에 따라 COM 포트 번호는 다르게 설정이 됩니다.

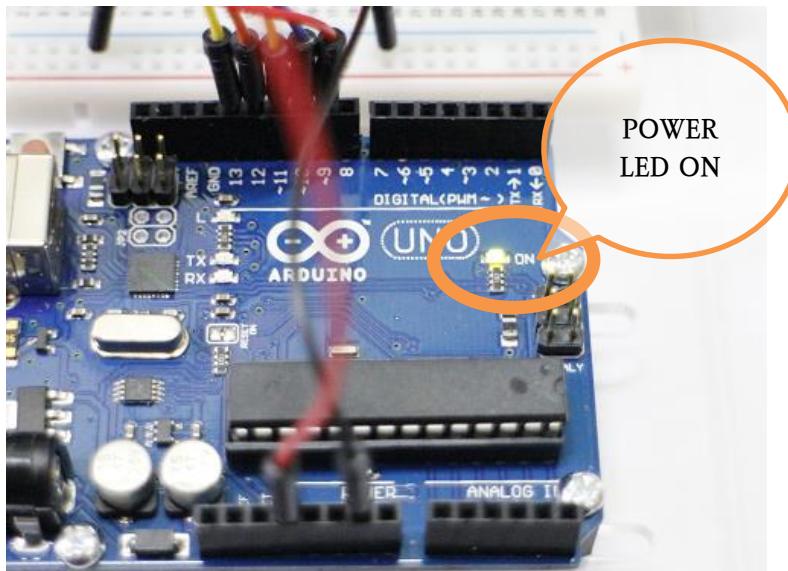


그림 7-4 우노 R3 보드 내장 LED

위의 시리얼 포트 번호를 기억하여 아두이노 IDE 스케치 프로그램에서 포트 설정을 해주어야 정확히 업로드가 됩니다.

최근의 아두이노 IDE 버전에서는 포트 메뉴 선택 시 포트에 대한 명칭도 같이 보입니다. 전원 공급, 드라이버 설정 등이 정상이면 아래와 같은 위치에 “ON” LED 가 켜져 있게 됩니다. 초록(연두색상) 또는 빨간 색상입니다.

### 7.3 아두이노 드라이버 간편 설치 방법

아두이노 IDE 프로그램 설치된 디렉터리 아래에 drivers라는 디렉터리가 있습니다.

\arduino-1.6.4\drivers

디렉터리의 내용을 탐색기로 보면 dpinst-amd64.exe, dpinst-x86.exe 파일이 보입니다. 2개의 실행 파일은 간편 설치 파일입니다. 1회 실행 하여주면 드라이버 정보 및 파일들이 설치 됩니다.

| 이름                      | 수정한 날짜           | 유형             | 크기      |
|-------------------------|------------------|----------------|---------|
| 파일 폴더 (1)               |                  |                |         |
| FTDI USB Drivers        | 2015-04-03 오전... | 파일 폴더          |         |
| 설치 파일 (2)               |                  |                |         |
| dpinst-amd64.exe        | 2015-04-02 오전... | 응용 프로그램        | 1,024KB |
| dpinst-x86.exe          | 2015-04-02 오전... | 응용 프로그램        | 901KB   |
| 설치 정보 (2)               |                  |                |         |
| arduino.inf             | 2015-04-02 오전... | 설치 정보          | 7KB     |
| arduino-org.inf         | 2015-04-02 오전... | 설치 정보          | 8KB     |
| 보안 카탈로그 (2)             |                  |                |         |
| arduino.cat             | 2015-04-02 오전... | 보안 카탈로그        | 10KB    |
| arduino-org.cat         | 2015-04-02 오전... | 보안 카탈로그        | 9KB     |
| TXT 파일 (1)              |                  |                |         |
| README.txt              | 2015-04-02 오전... | TXT 파일         | 1KB     |
| ALZip ZIP File (1)      |                  |                |         |
| Old_Arduino_Drivers.zip | 2015-04-02 오전... | ALZip ZIP File | 17KB    |

그림 7-5 아두이노 IDE 드라이버 디렉터리 파일들

아두이노 우노 보드를 PC 의 USB 포트에 연결(또는 연결하지 않은 상태여도 무방)합니다.

Dpinst-amd64.exe 설치 실행 파일은 64 비트 윈도우.

Dpinst-x86.exe 설치 실행 파일은 32 비트 윈도우.

위의 파일을 실행 후 아두이노 보드와 PC 의 USB 연결 시 인식되면서 사용 가능하게 됩니다.

## 8 아두이노 우노 R3 전원 공급 방법

전원 공급 방법은 우노 R3 보드 포함 다른 아두이노 보드들도 공통입니다.

### 8.1 USB 케이블 연결

PC의 USB 포트와 연결합니다.

프로그램 업로드를 하기 위해서도 사용됩니다.



PC의 USB 포트의 전원을 공급 받아 아두이노 보드가 작동 됩니다.

물론 USB 포트와 연결 되면서 USB 시리얼 포트가 생성되면서 아두이노 IDE 프로그램에서의 프로그램 업로드가 됩니다.

특별히 다른 AVR ICSP 프로그래머 기기를 사용하는 경우를 제외하고는 프로그램 업로드 후 테스트 완료까지는 계속 USB 연결한 상태로 프로그래밍하도록 되어 있습니다.

## 8.2 DC 어댑터 전원 연결

7V~12V DC 어댑터 연결하여 사용합니다.

보통 9V 1A, 12V 1A DC 어댑터가 많이 사용되고 있습니다.



그림 8-1 DC 9V 어댑터



그림 8-2 우노 R3 보드에 DC 9V 연결된 모습

### 8.3 9V DC 배터리

아두이노 우노 R3 보드 자체 단독으로 사용시 많이 적용되는 전원 공급 구조입니다.



그림 8-3 우노 R3 보드에 DC 9V 배터리 연결된 모습

아두이노를 활용한 여러 가지 용도에 적절히 사용될 수 있는 전원 공급 방식입니다. 다만 1회용 배터리이므로 지속적으로 소모, 구매 된다는 단점이 있습니다.

그래서 재충전 가능한 3.7V 18650 배터리 2 개 또는 3 개 사용되는 형태도 많이 사용되고 있습니다. 아래는 18650 배터리 2 개 사용되는 그림입니다.

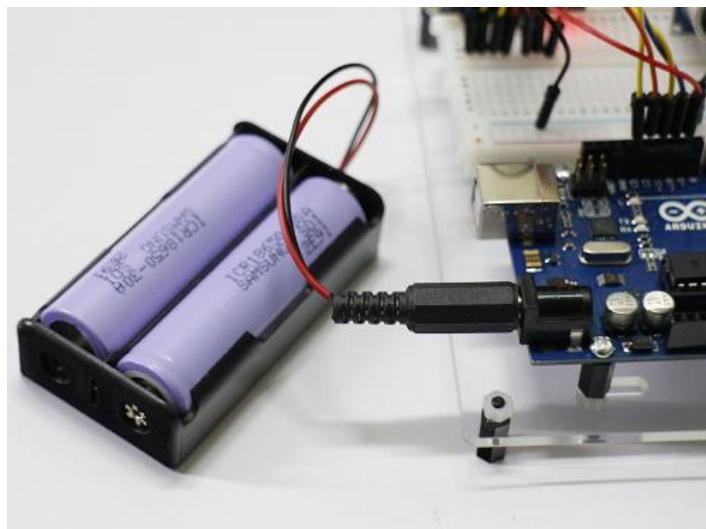
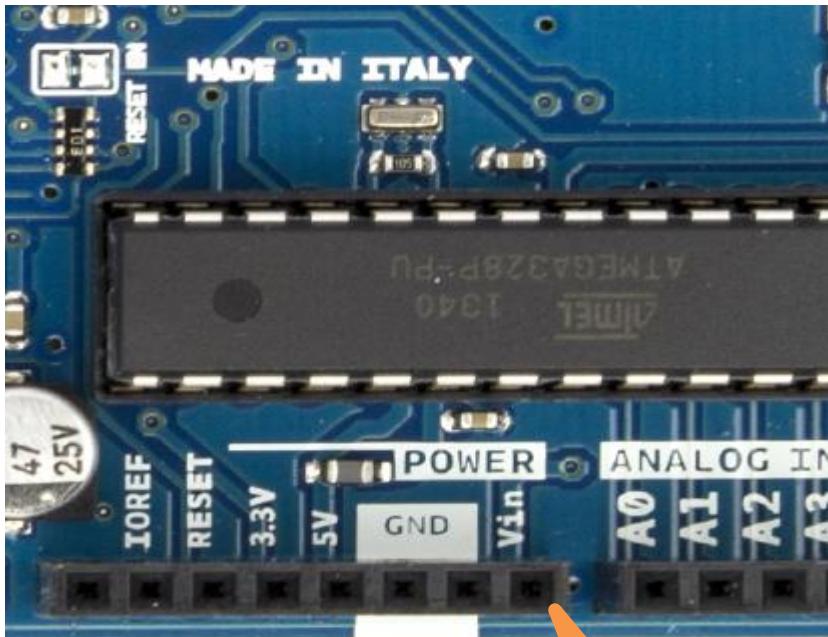


그림 8-4 18650 배터리 연결된 모습

## 8.4 VIN, GND 연결

아두이노 보드는 Vin 포트로 전원을 공급하여 사용 할 수도 있습니다. 여러 용도로의 사용시 전원 배선 시 Vin(+) 입력, GND, 2 가닥만 간단히 연결만 하면 작동 되게도 가능한 구조입니다.



Vin 전원 공급하여 사용할 경우 주의 사항은  
7~12V 범위내의 전원을 공급 해야 합니다.

주의하여 전원 입력 하기 바랍니다.

Vin 7~12 V  
가능

## 9 아두이노 스케치 프로그램 설치 & 설정.

아두이노 공식 사이트에서 스케치 프로그램을 다운로드 받습니다.

<http://arduino.cc/en/Main/Software>

예제의 설명은 1.5.8 버전 설명입니다.

필요에 따라 하위 또는 상위 버전을 사용하여도 무방합니다.

### Download

Arduino 1.5.8 ([release notes](#)):

- Windows: [Installer](#)
- Windows: [ZIP file](#) (for non-administrator install)
- Mac OS X: [ZIP file for Java 6](#)
- Mac OS X: [ZIP file for Java 7](#). NOTE: the ZIP for Java 7 is experimental and it works only on OSX 10.7 or greater. If you experience problems running it please use the Java 6 version.
- Linux: [32 bit](#), [64 bit](#)
- [source](#)

Windows: Installer → 윈도우 OS에서 사용 가능한 설치 형식의 파일.

Windows: ZIP file → 다운로드 후 원하는 디렉터리에 압축 해제 후 사용.

Mac OS X: ZIP File for Java 6 → 애플 OS X에서 사용 가능한 버전입니다.

Mac OS X: Zip file for Java 7 → OSX 10.7 이상의 버전에서 사용 가능한 버전입니다.

Linux: 32 비트, 64 비트 버전

Source: 스케치 IDE 프로그램 소스

Windows: Zip File 사용 방식의 파일을 다운로드 받아서 원하는 디렉터리에 압축 해제 후 사용합니다.

1.5.8 버전 압축 파일 형태의 배포 파일을 받아서 원하는 디렉터리에 압축 해제하면 아래와 같은 형태의 디렉터리가 보입니다.

본 예제에서는 하드 디스크 D:\ 드라이브의 arduino라는 디렉터리를 생성 후 다운로드 받은 파일을 압축 해제 하였습니다.

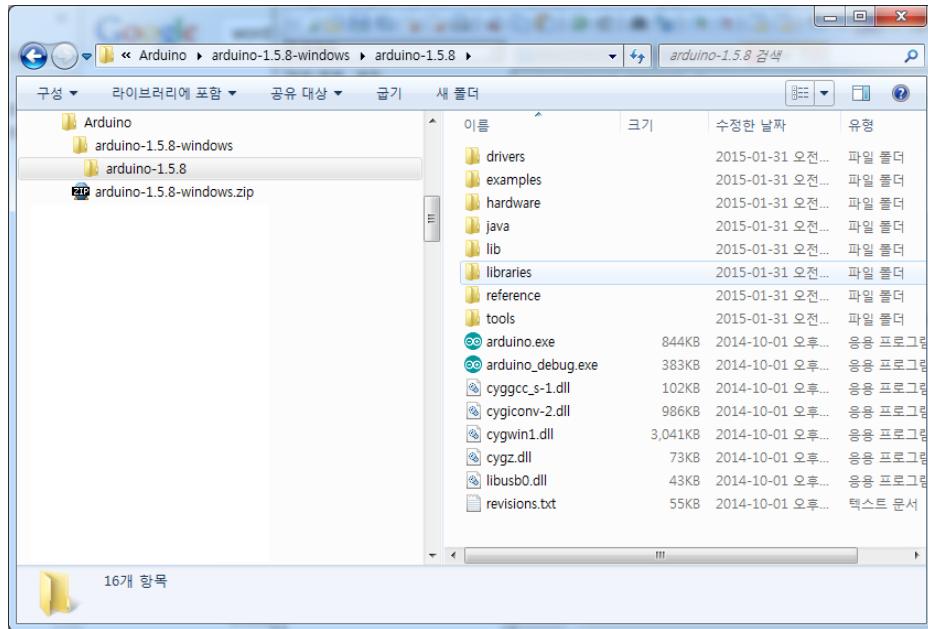
D:\arduino 디렉터리 생성

D:\arduino 디렉터리에 다운로드 받은 arduino-1.5.8-windows.zip 파일 복사  
arduino-1.5.8-windows.zip 파일을 압축 해제합니다.

아래와 같은 디렉터리가 나옵니다.

디렉터리 D:\arduino\arduino-1.5.8-windows\arduino-1.5.8

윈도우 탐색기로 보면 아래와 그림과 같습니다.



| 이름                | 크기      | 수정한 날짜           | 유형         |
|-------------------|---------|------------------|------------|
| drivers           |         | 2015-01-31 오전... | 파일 폴더      |
| examples          |         | 2015-01-31 오전... | 파일 폴더      |
| hardware          |         | 2015-01-31 오전... | 파일 폴더      |
| java              |         | 2015-01-31 오전... | 파일 폴더      |
| lib               |         | 2015-01-31 오전... | 파일 폴더      |
| libraries         |         | 2015-01-31 오전... | 파일 폴더      |
| reference         |         | 2015-01-31 오전... | 파일 폴더      |
| tools             |         | 2015-01-31 오전... | 파일 폴더      |
| arduino.exe       | 844KB   | 2014-10-01 오후... | 응용 프로그램    |
| arduino_debug.exe | 383KB   | 2014-10-01 오후... | 응용 프로그램    |
| cygcc_s-1.dll     | 102KB   | 2014-10-01 오후... | 응용 프로그램 확장 |
| cygconv-2.dll     | 986KB   | 2014-10-01 오후... | 응용 프로그램 확장 |
| cygwin1.dll       | 3,041KB | 2014-10-01 오후... | 응용 프로그램 확장 |
| cygz.dll          | 73KB    | 2014-10-01 오후... | 응용 프로그램 확장 |
| libusb0.dll       | 43KB    | 2014-10-01 오후... | 응용 프로그램 확장 |
| revisions.txt     | 55KB    | 2014-10-01 오후... | 텍스트 문서     |

- Drivers: 모든 아두이노 보드와 FTDI 칩셋 드라이버 파일들이 있습니다.
- Examples: 예제 코드 파일들이 있습니다.
- Hardware: 하드웨어 관련 펌웨어 소스 코드들이 들어 있습니다.
- Java: 아두이노 프로그램 IDE 프로그램에서 사용되는 GUI 포함 관련 파일들입니다.
- Lib: 아두이노 IDE 스케치 프로그램에서 사용되는 기본 라이브러리입니다.
- Libraries: 아두이노 스케치 프로그램에서 사용자가 사용 가능한 C/C++ 라이브러리 모음입니다.
- Reference: 기본 C/C++ 언어 설명들과 스케치 프로그램에서 사용되는 함수들의 설명 문서들입니다. HTML 파일로 되어 있습니다.
- Tools: 아두이노 프로세싱 프로그램에서 사용되는 플러그인 툴 디렉터리입니다.

Arduino → drivers 디렉터리(폴더)에 각종 윈도우 드라이버 파일들이 있습니다.

아두이노 USB 케이블 연결 시 드라이버는 경우 Arduino->drivers 디렉터리의 dpinst-amd64.exe (64 비트 윈도우 드라이버 설치) 실행합니다.

스케치 프로그램 실행 파일은 **arduino.exe** 입니다.

## 9.1 윈도우 탐색기에서 파일 확장자 보이도록 설정 방법

윈도우 XP, 윈도우 7, 윈도우 8 탐색기 설정 도구->폴더 옵션->보기(옵션 탭)"에 "알려진 파일 형식의 파일 확장명 숨기기" 체크 버튼 설정 해제 되어 있어야 파일의 확장 명칭이 보입니다.

확장명 없어도 사용하기 불편함이 없으면 "확장명 숨기기" 옵션으로도 상관 없습니다. 그럼 위의 arduino.exe 라고 탐색기에서 표시되지 않습니다. arduino 라고만 보입니다. 탐색기에서의 파일 리가 아이콘만으로 구분이 되어 충분히 사용할 수 있다면 상관 없지만, 아두이노 IDE 스케치 프로그래밍 시에 빈번히 찾아서 관리 해야 되는 파일들이 많습니다. CPP 파일과 C 파일 H 파일 등등 많이 있습니다.

되도록이면 파일 탐색기에서의 파일 확장명 보이는 상태로 사용하기 바랍니다.

아래의 그림처럼 폴더 옵션 설정ダイ얼로그에서

"알려진 파일 형식의 파일 확장명 숨기기" 체크 해제를 하면 됩니다.

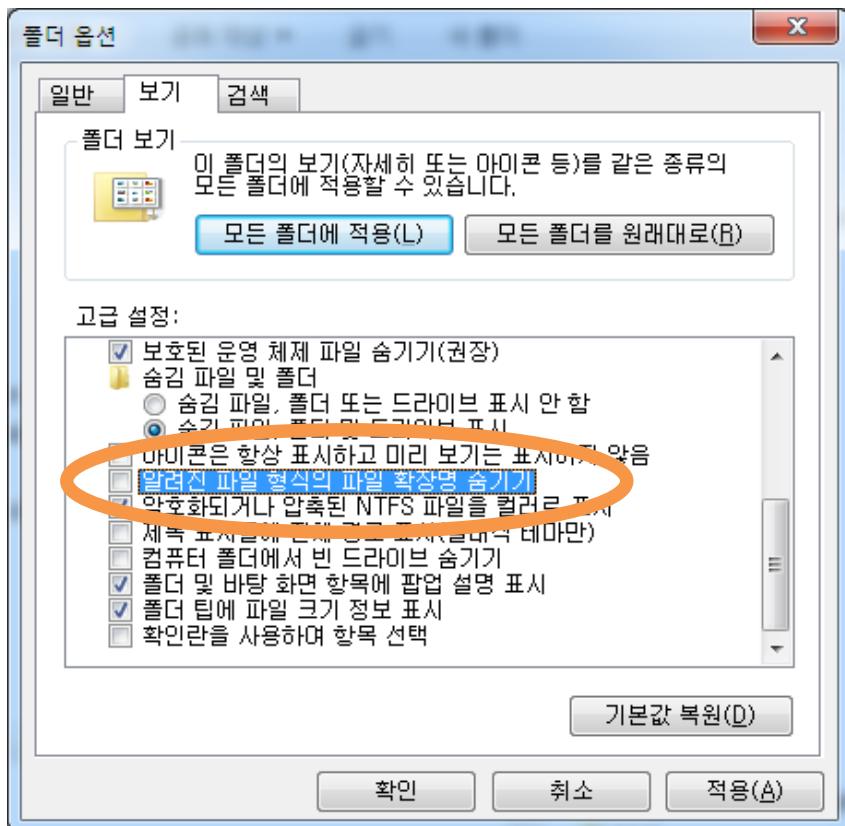
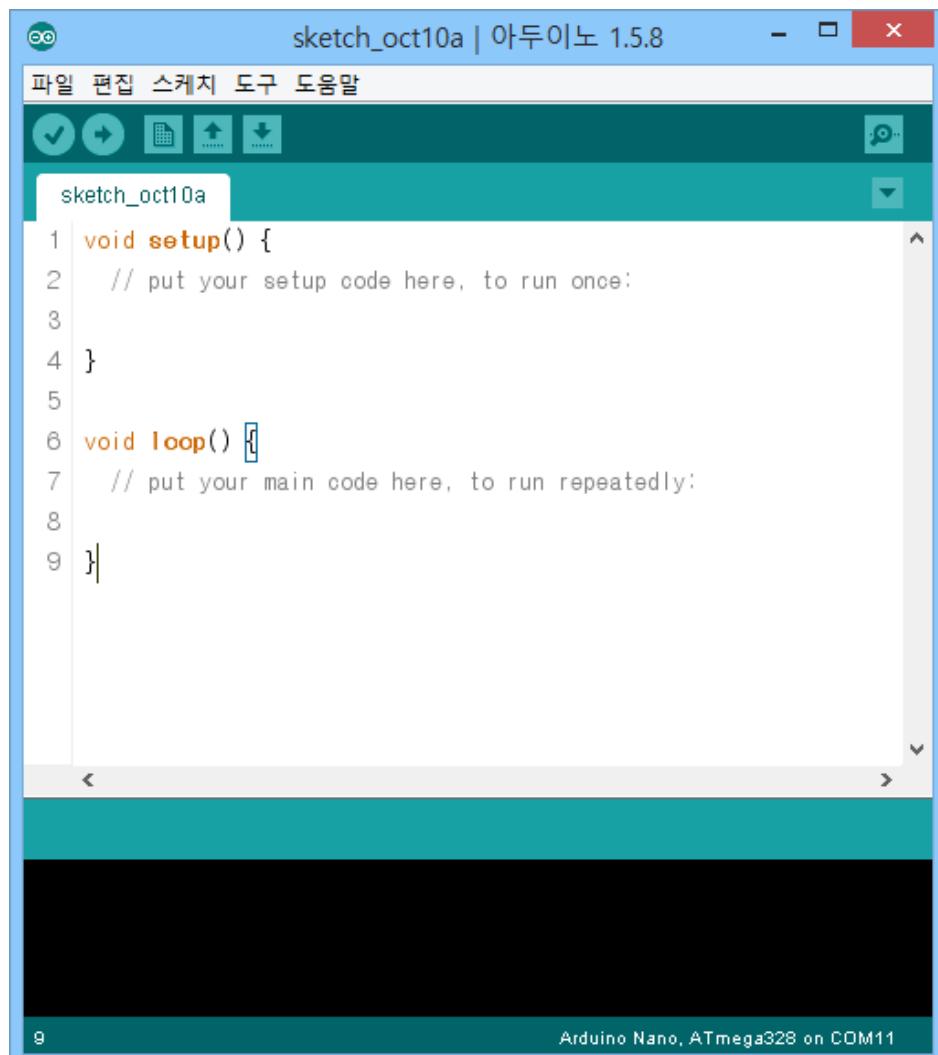


그림 9-1 윈도우 폴더 옵션 대화상자

## 9.2 스케치 기본 함수

스케치 IDE 파일 메뉴 -> ”새 파일” 선택하면 새로운 스케치 IDE 창이 생성되면서 아래의 코드가 나옵니다. 자동으로 입력되어 아래와 같이 코드를 볼 수 있습니다.



The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** sketch\_oct10a | 아두이노 1.5.8
- Menu Bar:** 파일 (File), 편집 (Edit), 스케치 (Sketch), 도구 (Tools), 도움말 (Help)
- Toolbar:** Includes icons for Save, Undo, Redo, Open, Upload, and Download.
- Code Editor:** Displays the following sketch code:

```
sketch_oct10a
1 void setup() {
2     // put your setup code here, to run once:
3
4 }
5
6 void loop() []
7     // put your main code here, to run repeatedly:
8
9 }
```
- Status Bar:** Shows the board as Arduino Nano, ATmega328 on COM11.

그림 9-2 아두이노 IDE 메인 창

## void setup()

아두이노 보드 부팅 시 1 회 호출되는 함수입니다. 보통 장착된 하드웨어 설정 및 각종 변수, 클래스 코드 초기화를 할 수 있습니다.

void 라는 용어는 c/c++에서는 “없다” 또는 NULL, 의미입니다.

함수의 반복 형태로 선언 되는 경우에는 함수 실행 후 반복되는 값이 없다는 것을 의미합니다.

```
void setup(void)
{
}
```

## void loop()

아두이노 보드 실행 시 매번 호출 되는 함수입니다. 아두이노 보드의 전원이 해제 되거나, 이상이 있을 경우를 제외 하고는 무한 계쏙 실행되는 함수입니다.

```
void loop(void)
{
}
```

% C/C++ 언어에서의 “{” “}”

C/C++ 언어에서의 중괄호 { } 의미는 범위 시작과 끝입니다.

한 개의 의미로 표현 할 수 없는 항목 요소에 대한 범위의 시작과 끝을 알립니다. 컴파일러는 중괄호를 기준으로 선언된 요소에 대한 일괄적인 묶음요소로 간주됩니다. 결국 “{” 중괄호 시작 부분과 닫아주는 “}” 표시의 개수는 동일합니다.

- » 함수의 시작과 끝을 정의 할 수 있습니다.
- » 클래스 선언의 시작과 끝을 정의합니다.
- » 각종 enum, structure 정의의 시작과 끝을 정의합니다.
- » 각종 배열 선언 시 내부 원소의 범위를 지정합니다.

```
int arr[] = { 1,2,3 };
» 각종 외부 선언 요소의 범위에 대한 영역 구분도 가능합니다.
extern "C"
{
    #include "MyC.h"
}
```

### 9.3 스케치 프로그램 IDE 기본 기능

스케치 IDE의 상단에는 많이 사용되는 기능의 버튼들이 있습니다.

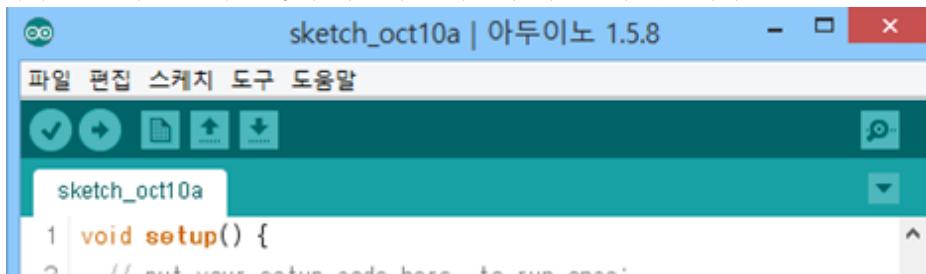


그림 9-3 상단의 메뉴와 툴바



스케치 프로그램 컴파일 버튼. 코드 컴파일 기능.



스케치 프로그램 컴파일과 업로드 버튼.



새 파일



열기 스케치 코드 소스 열기



저장. 현재의 코드 내용을 파일로 저장.



상단 오른쪽에 시리얼 포트 모니터 창 열기 버튼이 있습니다.

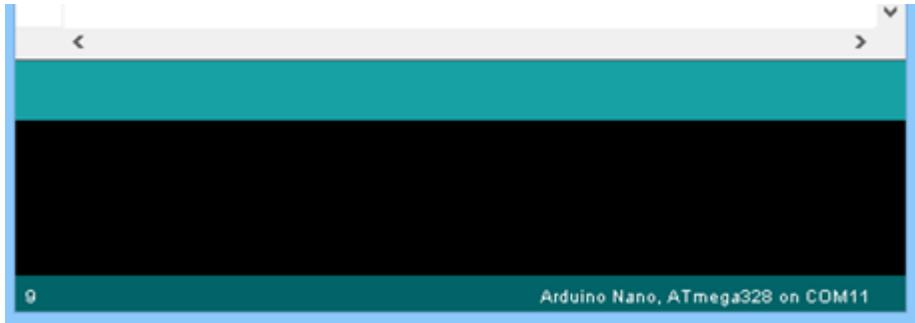


그림 9-4 아두이노 IDE 의 메시지 창 부분

컴파일, 업로드 및 아두이노 IDE 에서의 메시지 출력 창입니다.  
컴파일 과정, 업로드 성공, 실패 등의 메시지를 볼 수 있습니다.

» 코드를 기입하는 행동을 코딩(coding) 이라고 합니다.

코딩(coding)을 하는 사람을 코더(coder)라고도 합니다.  
정식 명칭은 소프트웨어 엔지니어, 소프트웨어 개발자라고 합니다.  
하드웨어 설계 및 개발자들은 하드웨어 엔지니어라고 합니다.  
모두 같은 개념에서의 “개발자” 입니다.

제품을 만드는 사람은 모두 개발자입니다. 제품의 일정 부분을 담당, 할당해서 개발을 하던,  
모두 본인 혼자서 개발을 하던, 제품 개발 프로젝트 진행 관리 업무, 취업하여 개발을 하는  
경우, 시험 또는 과제물, 개인적인 목적에 의해 코드를 기입하고 테스트한다면 보편적으로  
모두 같은 개발자입니다.

일례로, 코더라는 용어가 있습니다. 종이에 코드를 기입하고 설계하고 의사결정 코드 등을  
기입한 내용을 단순히 타자기만 치는 사람에게 컴퓨터에 입력을 하라고 하던 때의 오래 전  
의 직종입니다. 그 당시에는 타자 치는 직업도 고급 업무였습니다. 타자기도 드물뿐더러  
제대로 빠르게 치려면 타자기 치는 자격증을 취득하여 했습니다.

지금은 누구나 컴퓨터/스마트폰 키보드를 다룰 줄 압니다. 키보드는 제 2 의 말하는 도구 라  
고 해도 과언이 아닙니다.

오래 전부터 코더라는 말이 소프트웨어 개발자들을 비하하는 용어로 인식이 되어 있습니다.  
또는 그냥 단순 작업하는 신입 말단 개발자를 말하기도 합니다.

이는 인식의 부재에서 비롯된 개념입니다. 시간과 경험에 따른 업무 분담이 틀릴 뿐입니다.  
일부 IT, ICT 업계의 관리자들은 코더는 단순 컴퓨터 작업자를 말하기도 합니다. 그럼 관리  
자들은 단순 작업자가 아닐까요? 직업에는 귀천이 업듯이 하는 역할이 다를 뿐이라고 보는  
시각으로 정정 하는 것이 정답 같습니다.

기본적인 코딩을 할 줄 알아야 설계를 할 수 있습니다.

컴퓨터 소프트웨어/하드웨어 개발자들의 수명은 개인의 취향, 환경에 따라서 천차만별입니  
다. 어떤 사람은 몇 개월, 1~2년, 또 어떤 사람은 10년, 20~30년이 넘도록 실제 개발을 하  
고 있습니다. 개발 및 코딩을 하고 있습니다. 천재적인 코더들이라고 말할 수 있습니다. 해  
외 유명 개발자 또는 회사 등의 블로그를 보면 다양한 분야의 소프트웨어 개발자, 하드웨어

개발자들이 많다는 걸 알게 됩니다. 실제 개발 경력이 오래된 분들이 많습니다. 개발분야는 위낙 신기술이 많이 나와서 항상 공부를 해야 하는 불편함이 있지만, 이분들 모두 열심히 공부하고 신기술에 대한 소개, 그리고 아두이노를 활용하여 여러 가지 재미있는 제품, 새로운 기술, 제품을 접목하여 소개하고 있습니다.

아두이노를 사용한 소프트웨어/하드웨어 개발도 IC/ICT 등의 한 부분을 담당하게 되었다고 봅니다.

» 아두이노의 큰 장점 중의 하나는 소프트웨어와 하드웨어의 중간자 역할을 충실히 해주고 있다는 겁니다. 단순하게 보일 수는 있지만, 하드웨어와 소프트웨어의 프로그램 개념을 묶어 버렸습니다.

물론 하드웨어를 감싸고 있는 소프트웨어가 있는 기준의 펌웨어 개발 도구 형태를 가지고 있지만, 그 위에 순수 소프트웨어 개념으로 적용하여 최종 결과물을 만들 수 있습니다. 순수 소프트웨어 개발자들도 쉽게 사용할 수 있습니다.

물론 개인적인 취미 용도의 사용자 편하게 가져다 쓰게 되어 있습니다.

결국, 중간자 역할을 넘어 엄청난 파급력을 가지게 된 거 같습니다.

만약 여러분들도 소프트웨어, 하드웨어 개발에 관심이 있다면 꾸준히 공부하면서 창의적인 개발에 노력한다면 남들보다는 좀더 귀중한 시간을 가질 수 있으리라 보여집니다.

현재의 아두이노보다 더 좋은 개념을 가진 창조자 당신이 될 수도 있습니다.

언젠가는 또 다른 누군가에서 새로운 솔루션이 나올 수 있습니다.

아두이노 스케치 IDE 를 사용하여 프로그래밍 하는 경우

펌웨어 소프트웨어 엔지니어라는 명칭이 어울립니다.

하지만 기존 개념의 펌웨어 소프트웨어 개발자라긴 보단

개발자이며 새로운 기술에 만족해야 하는 기술자입니다.

» 스케치 한다?

아두이노에서의 용어는 공식 사이트, 해외 개발자 사이트 등의 포럼을 보면 스케치 한다고 합니다. 프로그래밍, 코딩 한다는 용어를 스케치 한다고 합니다.

새로운 용어이기도 합니다.

세련된 개념의 용어이기도 합니다.

## 9.4 보드 종류 & 포트 설정

프로그램 업로드 하기 전에 보드 종류와 포트를 먼저 확인하도록 합니다.

**보드선택:** 아두이노 IDE 스케치 프로그램은 아두이노의 여러 종류의 MCU 보드들을 사용할 수 있습니다.

우노 R3 / 메가 2560 / 프로미니 / 레오나르도 / 두에 등의 많은 MCU 보드들이 있습니다.

“아두이노 우노 R3” 보드 또한 그 중의 하나입니다.  
아두이노 IDE 스케치 프로그램에서 지정해주고 업로드를 해야 합니다.

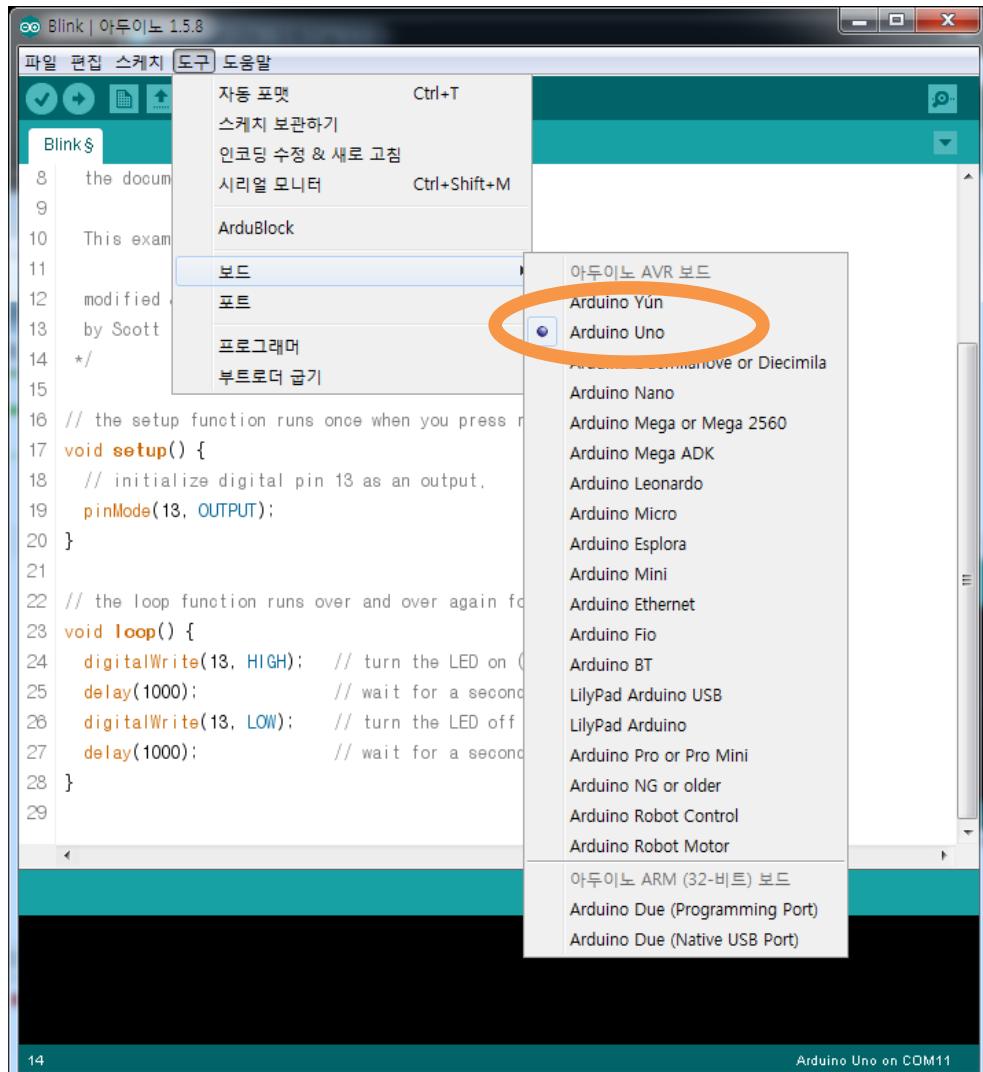


그림 9-5 보드 선택 메뉴

아두이노 IDE에서 사용 가능한 모든 보드의 목록이 보입니다.  
아두이노 우노 R3 보드를 사용하는 경우 아두이노 IDE 메뉴 -> 도구 -> “Arduino Uno”를 선택합니다.

**포트 선택:** 포트 선택을 합니다. 시리얼 통신 포트입니다.

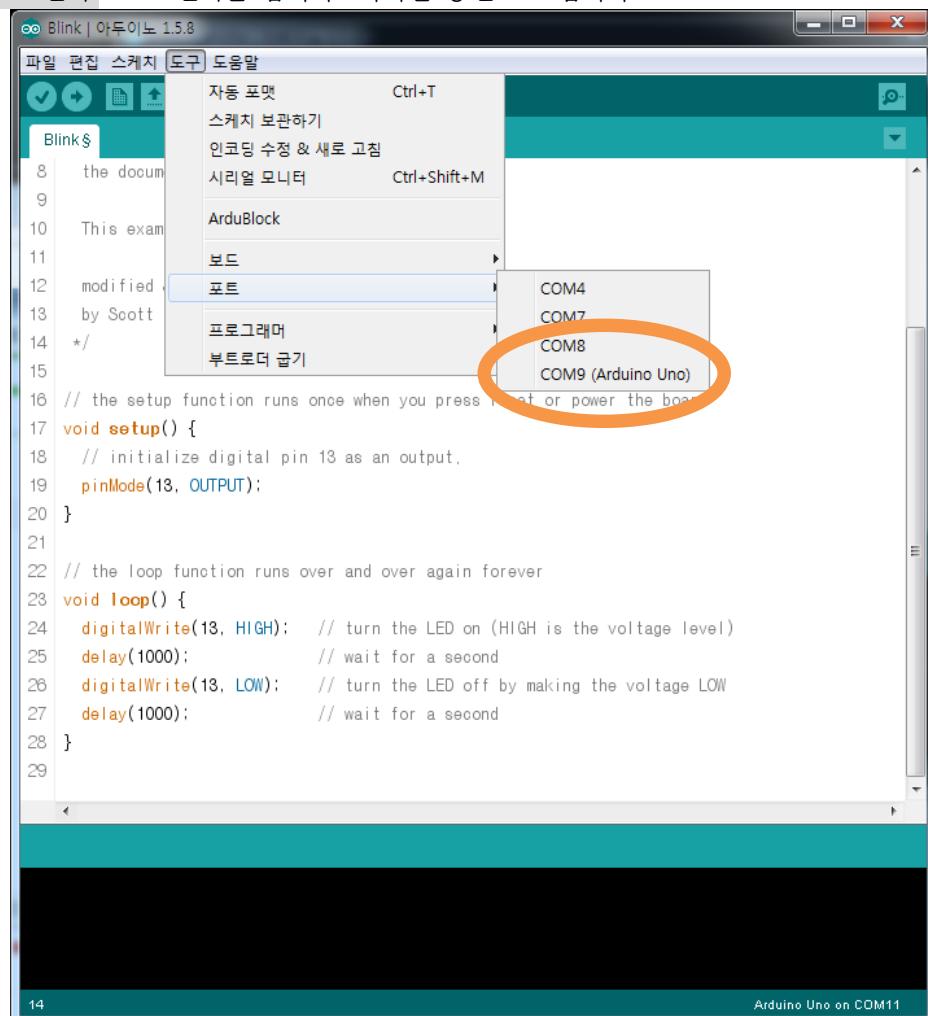


그림 9-6 시리얼 포트 선택

아두이노 우노 R3 보드가 정상적으로 연결되어 있고 드라이버 정상적으로 설치되어 있는 경우에는 위와 같이 COM9 처럼 포트가 할당 되어 있습니다.

포트 번호는 PC에 연결된 환경에 따라 COM1 ~ 부터 COM100 번호도 넘어서 할당 될 수 있습니다.

선택된 포트는 작성된 프로그램 코드가 컴파일&빌드 된 파일 업로드에 사용됩니다.

아두이노 우노 R3 보드에서의 D0, D1 포트가 사용되는 결과와 동일합니다.

업로드 하는 동안에는 아두이노 보드의 D0, D1 포트에는 무언가가 연결되어 있으면 업로드가 안됩니다.

#### 9.4.1 시리얼 포트 확인 방법

아두이노 보드가 있는 시리얼 포트를 모르는 경우 윈도우의 “제어판” 열어 봅니다. 제어판 창은 “시작” 메뉴 → “제어판” 선택하면 바로 볼 수 있습니다. “포트” 아래에 보드 이름에 맞게 “Arduino Uno” 있습니다. 옆에는 포트 번호가 있습니다. “아두이노 나노 V3” 보드는 “USB Serial Port”라고 보입니다. 아두이노 프로 미니 업로드 “FTDI BreakOut Board” 사용 연결도 “USB Serial Port”라고 보입니다.

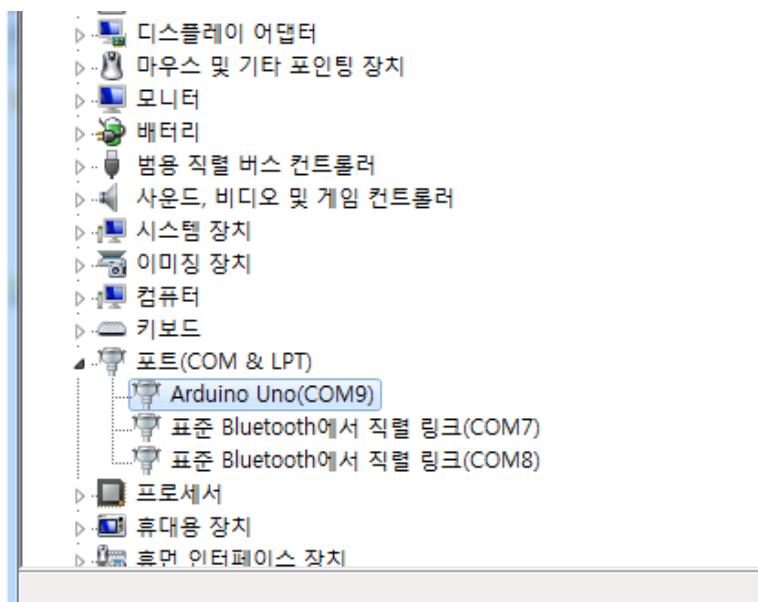


그림 9-7 제어판 포트 정보



그림 9-8 제어판 포트 나노 V3, FTDI BreakOut 보드 연결

## 9.5 스케치 프로그램 업로드

아래의 이미지처럼 기본 코드 상태에서 코드(code)를 기입합니다.

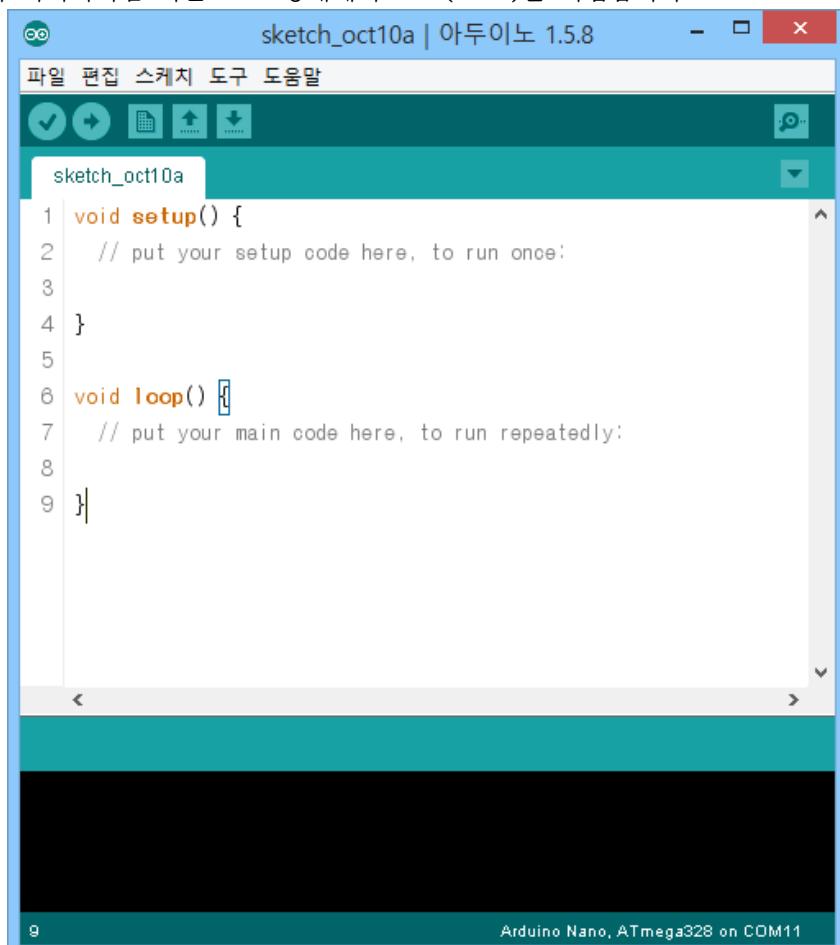


그림 9-9 아두이노 IDE 스케치 프로그램

위 그림처럼 기본 코드 상태에서 아래의 코드를 입력해 봅니다.

1초마다 Running...이라는 문자열을 시리얼 포트로 출력해주는 코드입니다.

```
void setup(void)  
{  
    Serial.begin(9600); // 시리얼 통신 속도 설정 9600 보레이트  
}  
void loop(void)
```

```
{  
    Serial.println("Running..."); // 시리얼 포트로 프린트 “Running”  
    delay(1000);  
}
```

### %% void setup() 과 void setup(void), 함수 파라미터 차이점??

소개되고 있는 예제 코드들의 함수들을 보면 void myFunc() 라고도 선언하고 void myFunc(void) 라고도 선언이 됩니다.

함수 파라미터로서의 void 는 기입하지 않아도 되는 항목입니다. 좀 더 명확히, 또는 일관된 규칙을 선호하는 경우 void myFunc(void) 라고도 기입할 수 있습니다. 같은 의미입니다. void myFunc() 의미, void myFunc(void) 의미는 공통적으로 “함수 파라미터는 사용하지 않는다” 입니다.

현재 스케치에서 작동된 코드가 빌드(컴파일->링크) 과정을 거쳐서 나온 hex 파일이 아두이노에 자동으로 업로드 됩니다. USB를 통한 시리얼 포트로 업로드 됩니다. hex 파일은 윈도우 사용자 임시 디렉터리의 랜덤 번호로 생성된 디렉터리 아래에 있습니다.

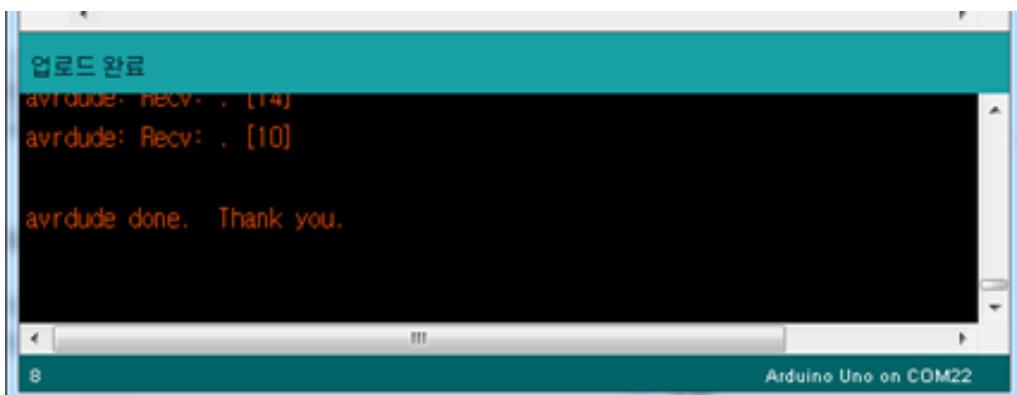


그림 9-10 업로드 진행상황 메시지

성공적인 업로드가 되었다면 아두이노 보드는 1 초마다 “Running...”이라는 문자열을 시리얼 포트 계속 전송하게 됩니다.

확인하기 위해서 시리얼 모니터 창을 열어서 출력되는 문자열을 보도록 합니다. 시리얼 모니터 창은 스케치 IDE의 메뉴 -> 도구 -> “시리얼 모니터” 선택하면 나옵니다. 또는 상단 툴바 영역의 오른쪽의 확대경 아이콘을 클릭해도 됩니다.

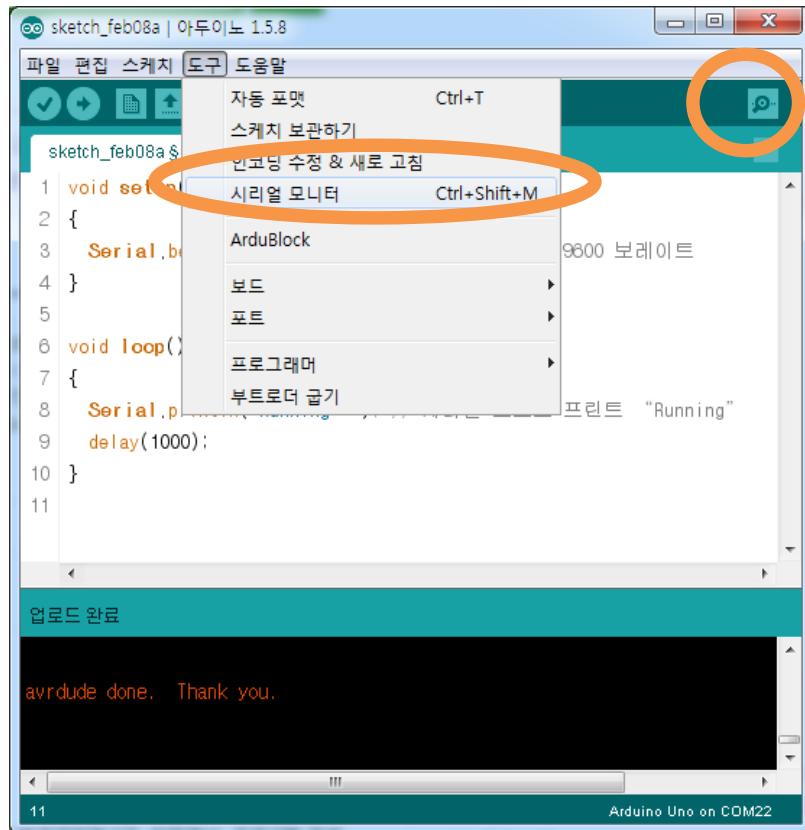


그림 9-11 시리얼 모니터 창 호출 메뉴와 아이콘 위치

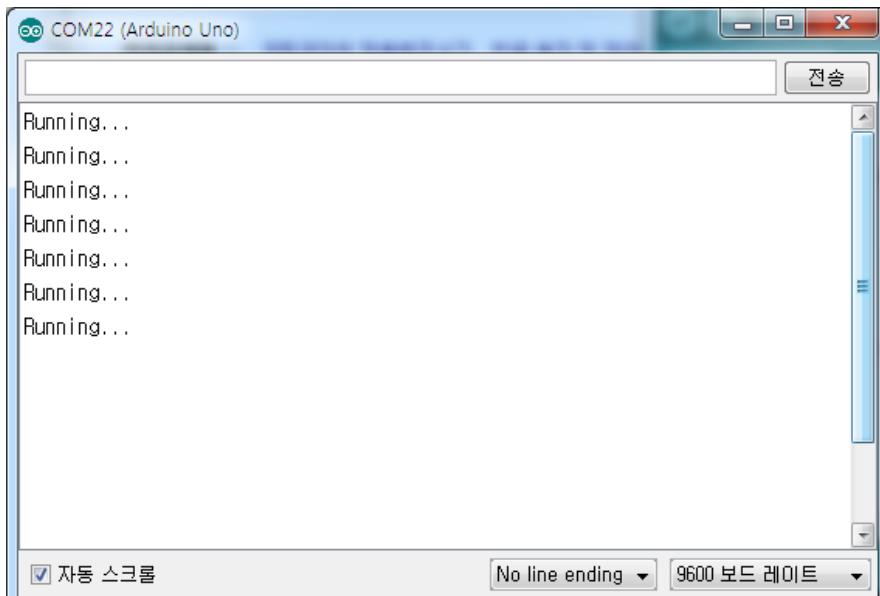


그림 9-12 시리얼 모니터 창에 표시되는 내용

## 10 아두이노 프로그램 업로드 과정에 대한 이해

아두이노 보드는 “부트로더”라는 것이 적용되어 있습니다.

“부트로더”라는 용어는 부트스트랩 로더(bootstrap loader)라는 단어의 약어입니다.

아두이노 보드의 의미 중에 하나는 아두이노 부트로더가 적용된 MCU 보드입니다.

보통 이런 보드는 “아두이노 보드”라고 부릅니다.

아두이노 보드를 PC의 USB에 연결, 또는 배터리에 연결하여 전원을 입력하면, 아두이노 보드는 기동되면서 PC의 개념처럼 부트로더가 제일 먼저 실행됩니다.

물론, 아두이노의 부트로더 또한 소스코드를 볼 수 있고 변경하여 사용할 수도 있습니다. 아두이노 보드에서 사용되는 주 MCU는 AVR 계열의 CPU 칩셋입니다.

그럼 부트로더라는 것은 어떤 기능이 있는지 살펴보도록 합니다.

먼저 MCU로 사용되는 AVR 계열의 간단한 이해를 하도록 합니다.

### 10.1 AVR MCU 란?

AVR(Atmel AVR)은 8비트 RISC 단일칩(OneChip) MCU입니다. MCU라는 용어는 마이콤(Micom)으로 불립니다. 1996년에 미국의 ATmel 회사가 하버드 아키텍쳐(Harvard Architecture)로 수정하여 개발한 구조라고 합니다. 그래서 AVR입니다.

AVR 칩셋은 프로그램을 저장하기 위해 기존의 MCU처럼 ROM, EPROM(or EEPROM)을 사용하지 않고 플래시메모리(Flash memory)를 처음 사용하였습니다. 일종의 약간의 변화입니다.

약간의 변화는 MCU의 보급 및 다양한 분야로의 사용이 가능하게 됩니다. 파급력은 상당히 크다고 보여집니다. 대중화에도 크게 기여한 바가 크다고 보여집니다.

플래시 메모리에는 필요에 의해 데이터를 다시 쓰기가 가능합니다. 플래시 메모리에는 실행 가능한 프로그램이 적재될 수 있습니다. 기존의 대부분의 MCU는 1회, ROM 또는 EEPROM(속도가 느림)에 쓰기 되어 있어 한번 쓰면 고정적인 기능만 할 수밖에 없습니다. 다시 사용하기 위해서는 룸을 제거하거나 버려야 했습니다. 물론, 용도에 의해 ROM에 펌웨어를 업로드하여 사용하는 단일칩도 많이 사용되고 있습니다.

AVR CPU는 Atmel 사에서 “AVR Studio”라는 개발 프로그램을 배포하여 “ISP” 포트를 사용하여 프로그램을 업로드 할 수 있습니다.

물론 “AVR Studio” 프로그램 IDE로 무료로 제공하고 있습니다. 오픈 소스는 아닙니다. AVR Studio의 장점중의 하나는 “C” 언어로 개발할 수 있습니다. 지금은 자바를 다룰 줄 아는 개발인력이 많지만 15년전만 해도 PC, IT, ICT 분야에 C/C++ 언어 소프트웨어 개발자가 상당히 많았습니다.

AVR MCU 소개 사이트:

<http://www.atmel.com/products/microcontrollers/default.aspx>

% 참고로 AVR 상위 개념으로 ARM MCU 가 있습니다. 스마트폰, 소형 PC 개념의 기기 등에 많이 사용되고 있습니다. 우리가 쉽게 들어보고 접할 수 있는 대표적인 ARM 보드로 “라즈베리파이” 있습니다.

## 10.2 AVR MCU 부트로더

대부분의 MCU 는 보통 프로그래머라는 장치에 의해 펌웨어 업로드 가능한 기능을 가지고 있습니다. AVR MCU 에 장착되어 있는 플래시 메모리 같은 비 휘발성 메모리는 부트로더 영역, 응용 프로그램 영역으로 나누어서 사용되고 있습니다.

부트로더 영역에는 부트로더 기능을 가진 프로그램이 적재되어 있습니다. 없는 경우에도 사용자가 부트로더 프로그램을 적재 할 수도 있습니다.

부트로드 영역에 올려진 프로그램은 UART 포트를 항상 감시하는 기능을 가지게 되면 반응이 있을 경우에는 전송된 데이터를 프로그래밍 가능영역에 쓰기를 합니다.

이 부분을 부트로더라고 호칭할 수 있습니다.

사용자는 프로그래밍 가능 영역에 실행 가능 프로그램을 올릴 수 있습니다.

이 기능은 AVR MCU 에서는 내장되어 있다고 볼 수 있습니다.

프로그래머라는 장치는 AVR MK 시리즈, USBTinyIsp, ParallelProgrammer 등이 있습니다.

(<http://www.arduino.cc/en/Hacking/ParallelProgrammer>)

AVR MCU 에서의 부트로더는 바로 언급된 것처럼, 외부에서의 프로그래밍 가능한 기능이 부트로더입니다.

## 10.3 아두이노 보드의 부트로더

아두이노의 부트로더 또한 9.2 내용을 그대로 반영하고 있습니다.

아두이노의 파급력의 한편에는 부트로더가 내장되어 있다는 것입니다.

사용자는 부트로더 같은 항목을 신경 쓸 필요 없이 작성한 프로그램을 PC 에서 USB 연결만 하면 편리하게 업로드를 하여 사용할 수 있습니다.

아두이노 부트로더는 아두이노 IDE 를 이용한 USB 시리얼 통신을 이용한 프로그램 업로드를 위한 기능. 소프트웨어 시리얼을 사용하기 위한 처리 코드가 있습니다.

사용자가 작성하여 업로드 된 프로그램 기동을 위한 기능이 있습니다.

아두이노의 각종 MCU 보드들은 USB 시리얼 통신을 통한 업로드를 방식을 채택하고 있습니다. 그래서 USB 시리얼 변환 MCU 가 한 개 더 사용됩니다.

기존의 ISP 방식의 업로드 방식이 아닌 보편적으로 가장 많이 사용되는 USB 방식에 시리얼 통신을 통하여 업로드를 하고 있습니다. 물론, 기존의 ISP 연결 방식을 통한 프로그램 업로드도 당연히 가능하게 되어 있습니다.

아두이노 부트로더 소스 코드 1:

(<http://code.google.com/p/arduino/source/browse/#svn/trunk/hardware/bootloaders>)

아두이노 부트로더 소스 코드 2:

부트로더 소스 코드는 PC에 아두이노 프로그램의 하위 디렉터리에도 있습니다.

아두이노 우노 R3의 부트로더 코드는 optiboot이라는 디렉터리에 있습니다.

\arduino-1.5.8\hardware\arduino\avr\bootloaders\optiboot

빌드 도구는 CrossPack-AVR-20100115과 WINAVR를 사용하였다고 합니다. 윈도우/리눅스에서 사용할 수 있습니다.

## 11 아두이노와 C/C++ 언어

아두이노 IDE 스케치에서 사용되는 프로그래밍 언어는 C/C++입니다.  
정식 명칭은 C Language & C Plus Plus Language입니다.

최근에는 HTML 친화적인 언어가 대세입니다.

자바, JSP, PHP, Dot Net, C# 등의 컴퓨터 언어를 사용하는 부분이 95% 이상이라고 봐도 됩니다.

C/C++은 이제 교육 교과 과정중의 1 개 과목, 게임엔진 개발 회사, 특정 서버 관련 데몬, 하드웨어 펌웨어 프로그래밍(거의 C 언어 위주), 리눅스 시스템 프로그램, 드라이버 소프트웨어 개발, 사용분야가 좁아지는 추세였습니다.

물론 국내를 비롯한 해외의 ICT 추세는 비슷하다고 보여집니다.

아두이노는 경이롭게도 일반적인 사용자가 사용하는 솔루션인데도 불구하고 C/C++ 언어를 사용하고 있습니다. C 언어와 C++ 언어가 혼용 되어 있지만 쉽게 구성되어 있는 관계로 많은 사람들이 사용하고 있습니다.

필자는 C/C++ 언어를 사용한 실무 개발을 오랫동안 해오고 있습니다.

C/C++ 언어 자체가 좋아서 했다라고 말씀 드릴 수는 없지만, 협업 프로젝트에 계속 몸담고 있다 보니 계속 개발하게 되었습니다. 물론 C/C++ 언어로 개발 할 수 있는 플랫폼은 많습니다. 예전의 DOS 시대의 풀다운 프로그램, 한글 프로그래밍부터, 유닉스(Sun Sparc, X-Window, MOTIF 윈도우 프로그래밍)와

윈도우 NT 출현하면서 MFC 프로그래밍과, OpenGL / DirectX, 3D 프로그래밍, 게임 소프트웨어 개발, 리눅스 시스템 커널과 시스템 프로그램, 드라이버 구축, 리눅스 시스템 프로그래밍 등의 많은 분야를 해온 거 같습니다.

아두이노를 접하면서 느낀 점은 직관적으로 보이는 부분들이 많다는 겁니다.

해당하는 모듈에 대한 많은 리소스들이 존재하여 사용자들은 쉽게 구현하도록 되어 있습니다.

C/C++ 언어의 가장 기초적인 변수 선언과 대입만 알아도 아두이노라는 하드웨어를 사용 할 수 있다는 겁니다. 하드웨어의 복잡한 부분을 몰라도 필요한 부분은 웬만큼 구현하도록 되어 있습니다.

```
void 아두이노기초 1(서적 구입)
```

```
{
```

만약, C/C++ 처음 접하시는 분들은 서점에서 또는 전자서적으로 C/C++ 기초 서적 아무거나 구입하여 약간의 기초 지식만 알고 아두이노 프로그래밍을 분석하거나 구현하는 경우 상당히 쉽게 적용 가능합니다. C/C++ 기초 지식만 알아도 아두이노 관련 프로젝트의 거의 모든 기능을 빠르게 습득 가능하리라 예상됩니다.

```
}
```

## 12 아두이노 IDE 스케치 라이브러리

환경 설정 항목 중 가장 중요한 개념이 라이브러리 설정 개념입니다.

기존 개발자, 제조사에서 미리 작성해 배포하는 라이브러리와 사용자들이 직접 만들어서 배포하는 사용자 라이브러리로 구분할 수 있습니다.

각각의 라이브러리들은 C/C++ 형태로 되어 있습니다.

### 12.1 스케치 IDE 기본 라이브러리 정보

크게 아두이노에서의 기본 제공 라이브러리와 사용자 지정 라이브러리를 사용할 수 있습니다.

아두이노 스케치 IDE 기본 라이브러리

스케치 IDE 실행 디렉터리 아래에 libraries 디렉터리에는 기본 지원 라이브러리가 있습니다.

arduino-1.5.8-windows\arduino-1.5.8\libraries

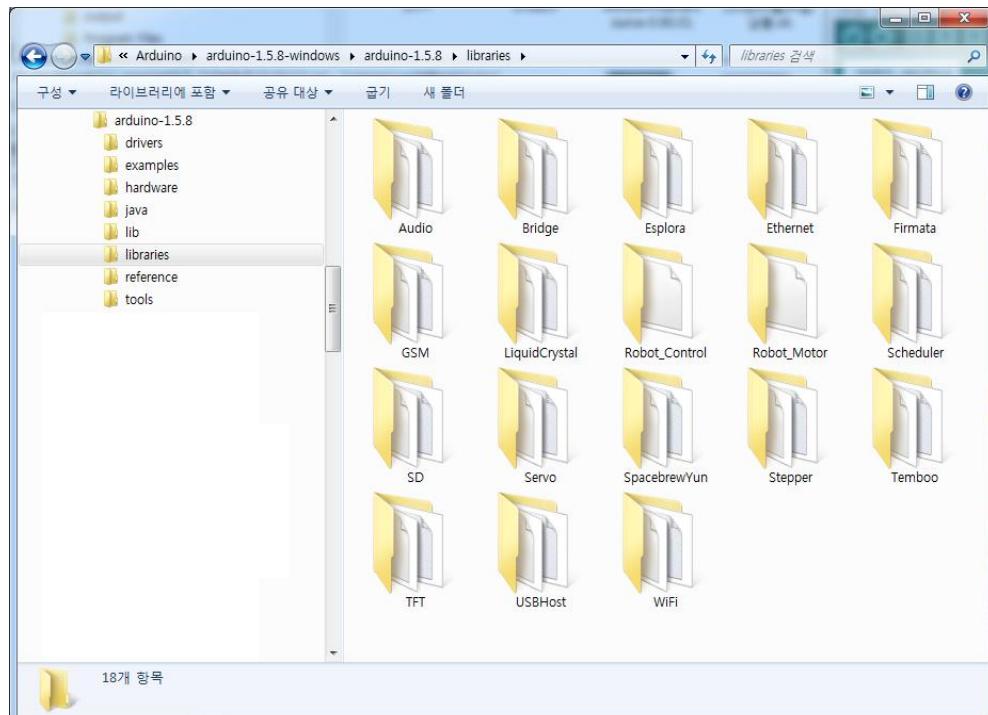


그림 12-1 아두이노 기본 라이브러리 디렉터리

아두이노 사이트에서 배포되는 기본 라이브러리로서 이더넷 네트워크, LCD, 아두이노 로봇 관련, 모터 제어, GPS, WIFI 등등 많은 기본 라이브러리를 제공합니다.

## 12.2 아두이노 사용자 라이브러리 설정

사용자 라이브러리: “스케치북”이라는 디렉터리를 지정하여 공개 라이브러리, 또는 자신만의 라이브러리를 설정 후 사용할 수 있습니다.

환경 설정 창을 열면 사용자 라이브러리 디렉터리의 위치 및 변경이 가능하도록 되어 있습니다.

스케치 IDE 메뉴->파일->”환경 설정” 선택하면 아래의 대화상자(다이얼로그)가 보입니다.

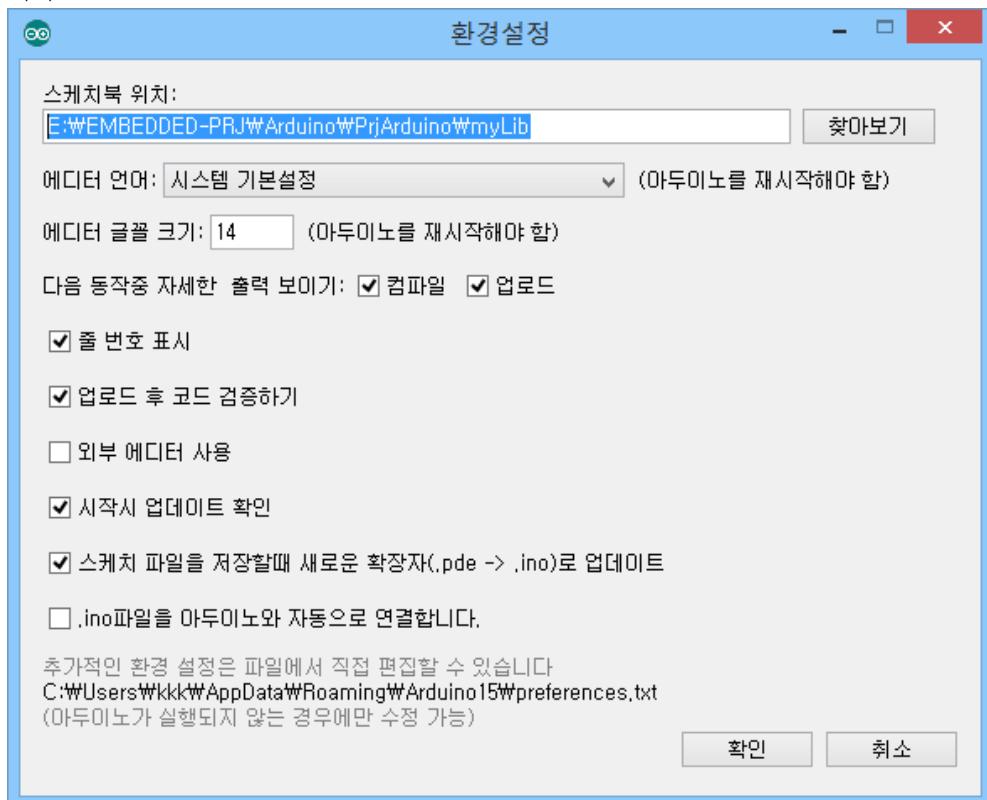


그림 12-2 아두이노 IDE 환경 설정 창

“스케치북 위치:” 기본 위치는 윈도우 사용자 “MyDocuments” 아래의 디렉터리로 설정 되어 있습니다.

기본 설정대로 사용, 또는 임의의 원하는 위치의 디렉터리를 지정하여 사용하도록 합니다.

아두이노는 기본 라이브러리 이외에 공개 라이브러리를 적용하는 경우가 많이 있습니다. 본인이 사용 및 지정된 디렉터리 관리를 꾸준히 하도록 합니다.

본인만의 라이브러리 디렉터리의 코드들을 바탕으로 보다 많은 아두이노 프로젝트 및 개발 및 응용을 할 때 많은 도움이 되리라 봅니다.

블로그, 카페, 포럼, 공식 사이트 등에는 많은 아두이노 사용자 라이브러리가 존재하고 있습니다. 사용자 라이브러리 규칙 적용은 아두이노 대중화(?)의 큰 원동력 중에 하나이기도 합니다. 누구나 원하는 목적에 맞게 코드를 개발하여 라이브러리화하여 블로그, 포럼 등에 게재하면, 누군가는 해당 코드를 다운로드 받아서 사용, 또는 개선하여 다시 올립니다. 소셜(Social) 코딩, 소셜 프로그래밍이라고 명명하기에 부족함이 없습니다.

사용자 라이브러리의 코드를 잘 관리하고 습득하고 공부한다면 보다 많은 응용 프로그램을 만들 수 있을 것입니다.

**“다음 동작 중 자세한 출력 보이기:** 용어 그대로 컴파일과 업로드 시에 아래의 메시지 출력창에 과정들이 텍스트로 모두 보이게 됩니다.

체크를 하고 사용하기 바랍니다. 컴파일과 업로드 과정은 프로그래밍 시에 자주 사용되므로 해당 과정에 대한 이해와 예러 메시지를 참조하여

**“줄 번호 표시:**” 아두이노 IDE 스케치 코드 화면의 왼쪽에 줄 번호가 보이도록 합니다. 개인적인 사용 방식에 따라 다르겠지만, 줄 번호가 보이게 하면 좀 더 편하게 코드 기입이 가능하리라 봅니다.

### 12.3 아두이노 IDE 환경 설정 저장 파일

환경 설정 다이얼로그의 아래 부분에 아두이노 IDE의 모든 설정이 저장된 위치와 파일을 표시해주고 있습니다.

메모장 같은 TXT 파일 편집기를 열어서 환경 설정을 직접 수정하여도 무방합니다. 가끔 아두이노 스케치 IDE가 이상하게 동작하거나 코드 화면의 내용이 잘 안보일 경우 초기화 할 경우에는 설정 저장 파일 TXT를 삭제하면 됩니다.

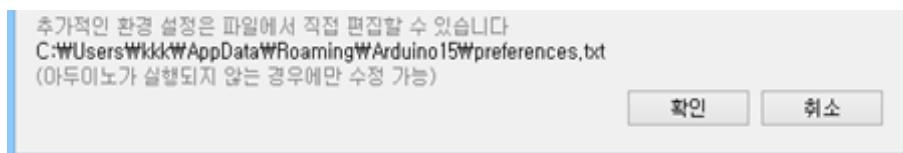


그림 12-3 환경 설정 저장 파일 위치

삭제 후 아두이노 스케치 IDE를 다시 실행하면 모든 환경이 초기 상태로 돌아가면서 새롭게 preferences.txt 파일이 해당 디렉터리 위치에 생성됩니다. 있습니다.

## 13 아두이노 스케치 IDE 예제 업로드 & 테스트

스케치 IDE 프로그램에는 많은 기본 예제들이 있습니다.

### 13.1 예제 열기

LED 깜박이는 예제를 빌드 & 업로드 & 테스트 해봅니다.

스케치 IDE 메뉴 -> 파일 -> 예제 -> 많은 라이브러리 항목이 보입니다.

위에서 정의된 사용자 라이브러리 디렉터리의 예제들도 모두 보입니다.

메뉴->파일->예제->열기에서 사용되는 코드들은 “읽기 전용” 상태로 열리게 됩니다.

코드를 수정 하였다면 다른 디렉터리에 저장하여야 합니다.

제일 처음 보이는 “01. Basics” 의 Blink 예제를 열기 해봅니다.

“Blink” 예제는 1초 주기로 D13 번 포트의 LED On / Off 반복하는 코드입니다.

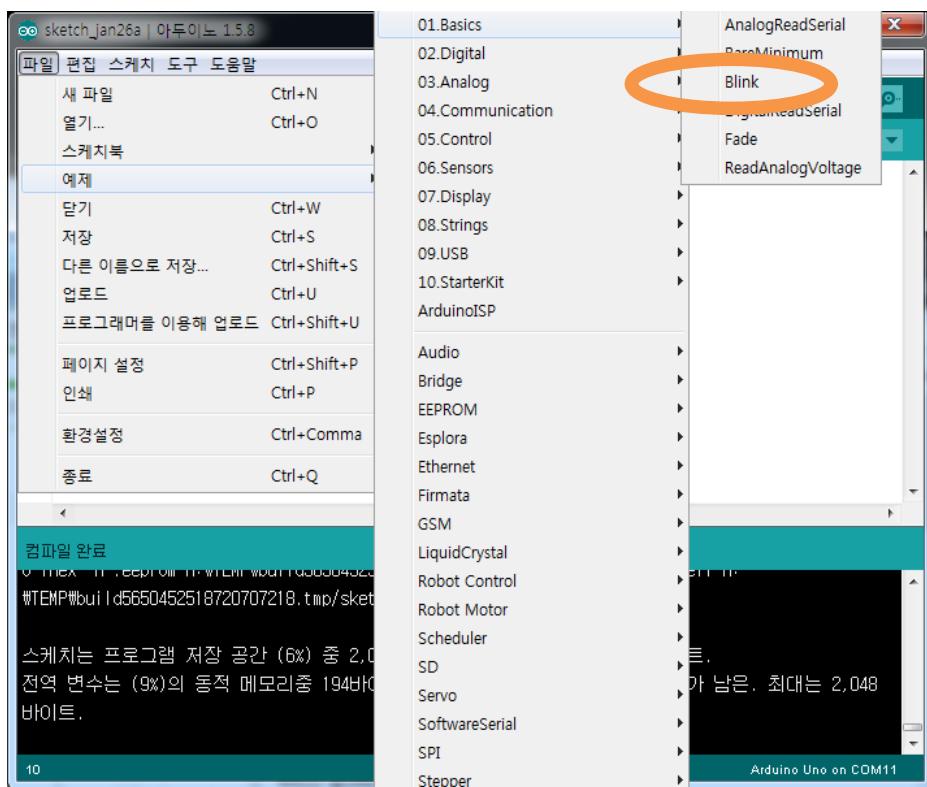


그림 13-1 예제 메뉴 선택 화면

보드의 내장 LED 는 D13 번 포트에 있어서 별도의 LED 부품 장착 없이 코드 실행 확인 가능합니다. 차후에 D13 번 내장 LED 표시는 프로그램 실행 시 디버깅 표시등에 유용하게 사용 할 수 있습니다.



그림 13-1 내장 LED 위치

“Blink” 예제를 가져온 상태의 아두이노 IDE 프로그램 화면입니다.

예제를 열기 할 경우와 열기를 하여 코드 파일을 가져온 경우 새로운 IDE 창이 생성됩니다. 기존 IDE 창에서 열리는 방식이 아닌 새로운 IDE 창이 나오게 됩니다.

기존 IDE 창은 닫거나 필요에 의해 사용해도 무방합니다.

```

Blink | 아두이노 1.5.8
파일 편집 스케치 도구 도움말
Blink §
8   the documentation at http://arduino.cc
9
10 This example code is in the public domain.
11
12 modified 8 May 2014
13 by Scott Fitzgerald
14 */
15
16 // the setup function runs once when you press reset or power the board
17 void setup() {
18   // initialize digital pin 13 as an output,
19   pinMode(13, OUTPUT);
20 }
21
22 // the loop function runs over and over again forever
23 void loop() {
24   digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
25   delay(1000);          // wait for a second
26   digitalWrite(13, LOW); // turn the LED off by making the voltage LOW
27   delay(1000);          // wait for a second
28 }
29

```

14 Arduino Uno on COM11

그림 13-2 “Blink” 예제 코드

“파일”→“예제” 메뉴에서는 아두이노 IDE 의 기본 라이브러리를 포함, 사용자 디렉터리의 라이브러리까지 모두 접근하여 가져올 수 있습니다.

## 14 프로그램 빌드 & 업로드

“Blink” 예제를 업로드 해봅니다.

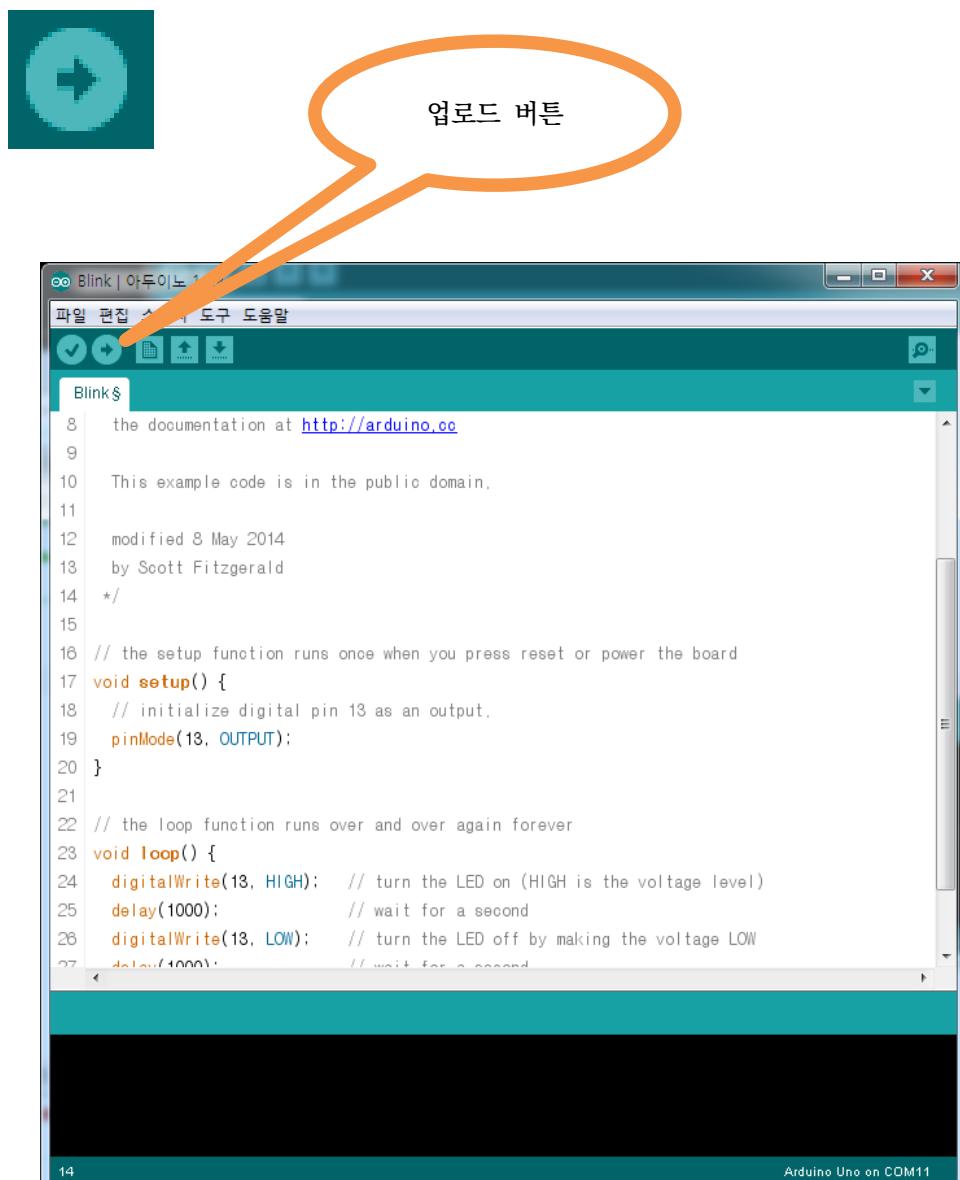


그림 14-1 프로그램 업로드 버튼 위치

“스케치를 컴파일 중...” 메시지가 나오면서 오른쪽에서 진행 상황 막대가 표시됩니다.



The screenshot shows the Arduino IDE interface during compilation. The code editor at the top contains the following code:

```
22 // the loop function runs over and over again forever
23 void loop() {
```

Below the code editor, a green progress bar indicates the status of the compilation. A message box in the center says "스케치를 컴파일 중..." (Compiling sketch...). The status bar at the bottom right shows "Arduino Uno on COM9".

그림 14-2 컴파일 과정 메시지 창

정상 컴파일 빌드 종료된 상태인 경우 “업로드 완료”라는 알림과 함께 아래와 같은 메시지가 나옵니다.



The screenshot shows the Arduino IDE interface after a successful upload. The code editor at the top contains the same code as in the previous screenshot. The message box in the center says "업로드 완료" (Upload complete). Below it, the serial monitor window displays the following text:

```
avrduude: Send: 0 [5]  [20]
avrduude: Recv: . [14]
avrduude: Recv: . [10]

avrduude done. Thank you.
```

The status bar at the bottom right shows "Arduino Uno on COM9".

그림 14-3 업로드 완료 메시지

그림 13-2, 3 번의 출력되는 내용을 보기 위해서는 11-2 의 환경 설정 창에서 체크되어 있어야 합니다.

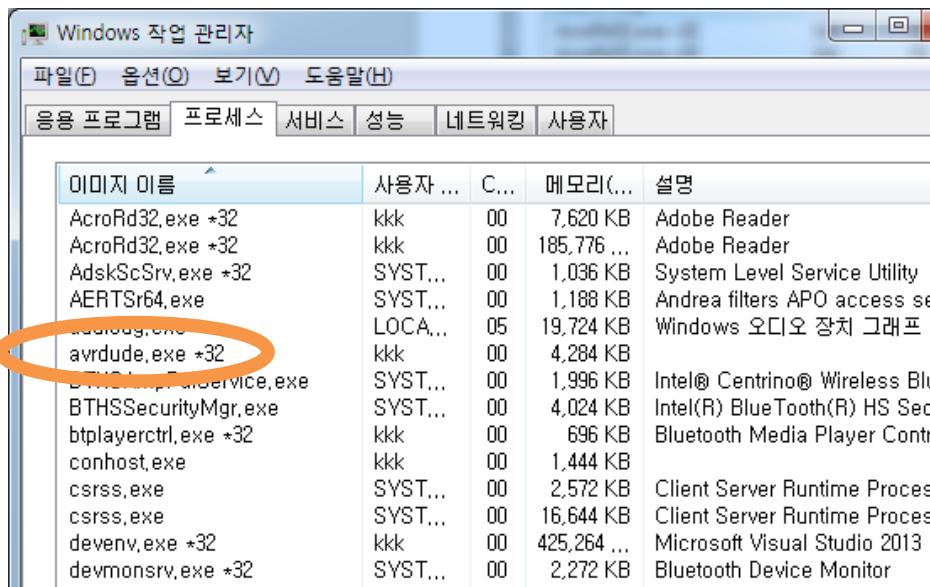
다음 동작중 자세한 출력 보기:  컴파일  업로드

## 15 업로드 안 되는 경우

업로드가 안 되는 경우는 여러 종류입니다.

- 보드 종류 선택 설정이 틀린 경우.
- 포트 선택 안되어 있거나 다른 포트가 지정되어 있음.
- avrdude.exe 계속 실행 중인 경우, 시리얼 포트가 사용 중 상태일 경우가 있습니다.
- 포트 선택은 제대로 되어 있으나 다른 시리얼 통신 프로그램에서 사용하면서 점유되고 있음.
- 아두이노 보드의 D0, D1 포트에 뭔가가 연결되어 작동 되는 경우.
- 드라이버 설치가 제대로 안되어 시리얼 포트가 생기지 않음.
- USB 전원 케이블 불량 또는 허브의 불량으로 전원 공급이 불안.
- 6 번의 이유로 아주 간혹 아두이노의 부트로더가 삭제됨.
- 6 번의 이유로 아주 간혹 아두이노의 ATMEGA16U2 펌웨어가 휘발됨
- 이 외에도 여러 가지 경우가 있습니다.

위의 3 번 증상인 경우 시리얼 포트가 사용 중이라는 메시지가 나오게 되어 있습니다. 작업 관리자를 띠워서 avrdude.exe 강제 종료 후 업로드 하면 정상적으로 됩니다.



| 이미지 이름               | 사용자 ... | C... | 메모리(...)    | 설명                           |
|----------------------|---------|------|-------------|------------------------------|
| AcroRd32.exe *32     | kkk     | 00   | 7,620 KB    | Adobe Reader                 |
| AcroRd32.exe *32     | kkk     | 00   | 185,776 ... | Adobe Reader                 |
| Adsk3ScSrv.exe *32   | SYST... | 00   | 1,036 KB    | System Level Service Utility |
| AERTSR64.exe         | SYST... | 00   | 1,188 KB    | Andrea filters APO access se |
| avrdude.exe          | LOCA... | 05   | 19,724 KB   | Windows 오디오 장치 그래프 :         |
| avrduude.exe *32     | kkk     | 00   | 4,284 KB    |                              |
| BTHSService.exe      | SYST... | 00   | 1,996 KB    | Intel® Centrino® Wireless Bl |
| BTHSSecurityMgr.exe  | SYST... | 00   | 4,024 KB    | Intel(R) BlueTooth(R) HS Sec |
| btplayerctrl.exe *32 | kkk     | 00   | 696 KB      | Bluetooth Media Player Contr |
| conhost.exe          | kkk     | 00   | 1,444 KB    |                              |
| csrss.exe            | SYST... | 00   | 2,572 KB    | Client Server Runtime Proces |
| csrss.exe            | SYST... | 00   | 16,644 KB   | Client Server Runtime Proces |
| devenv.exe *32       | kkk     | 00   | 425,264 ... | Microsoft Visual Studio 2013 |
| devmonsrv.exe *32    | SYST... | 00   | 2,272 KB    | Bluetooth Device Monitor     |

그림 15-1 작업 관리자 화면

7 번, 8 번의 이유로 업로드가 안 되는 경우 부트로더를 다시 적용해 주어야 합니다.

## 16 아두이노 스케치 코드의 MAIN 함수는?

C/C++ 프로그램은 int main(int argc, char\*\* argv); 라는 형식의 메인 함수를 가지게 됩니다. main 이라는 함수는 작성된 시스템 개발 프로그램의 출발지점이 됩니다. 하지만, 스케치 프로그램에서 작성된 펌웨어 프로그램은 setup() 이라는 초기화 함수와 loop() 라는 함수가 사용됩니다.

위에서도 언급되어 있듯이 스케치라는 프로그램은 공개 소스기반 플랫폼입니다. 공개 소스는 과연 어디에 있을까라는, 찾아보고 약간의 분석만 해보아도 좀더 진보적인 프로그래밍이 가능할거라 보여집니다.

### 16.1 스케치 IDE 프로그램 소스 코드 분석

아두이노 IDE 스케치 프로그램에서의 실제 main() 함수를 찾아봅시다.

스케치 IDE 프로그램이 설치된 디렉터리를 탐색기로 열어봅니다.

1.5.8 버전을 사용하는 관계로 디렉터리는 아래와 같습니다.

```
\arduino-1.5.8\hardware\arduino\avr\cores\arduino
```

Main.cpp 파일의 내용입니다.

```
/*
 main.cpp - Main loop for Arduino sketches
 Copyright (c) 2005-2013 Arduino Team. All right reserved.
```

This library is free software; you can redistribute it and/or  
modify it under the terms of the GNU Lesser General Public  
License as published by the Free Software Foundation; either  
version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful,  
but WITHOUT ANY WARRANTY; without even the implied warranty of  
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the  
GNU

Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public  
License along with this library; if not, write to the Free Software  
Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

```

*/
#include <Arduino.h>

//Declared weak in Arduino.h to allow user redefinitions.
int atexit(void (*func)()) { return 0; }

// Weak empty variant initialization function.
// May be redefined by variant files.
void initVariant() __attribute__((weak));
void initVariant() {}

int main(void)
{
    init();

    initVariant();

#if defined(USBCON)
    USBDevice.attach();
#endif

    setup();

    for (;;) {
        loop();
        if (serialEventRun) serialEventRun();
    }

    return 0;
}

```

위의 코드의 `setup()` 과 `loop()` 함수가 보입니다.

아두이노 IDE 프로그램의 컴파일과 빌드 과정에서 위에서 찾은 소스 코드도 매번 같이 묶여서 컴파일 됩니다. 위의 코드와 같이 해당 디렉터리의 거의 모든 파일이 “Blink” 예제 같이 작은 프로그램 코드 빌드 시에도 묶여서 같이 빌드 됩니다.

타 회사의 AVR 프로그래밍 IDE 와 비교 시 아두이노 IDE 에서는 약간의 빌드 시간이 더 소모됩니다.

확인 방법은 IDE 화면의 메시지 로그창의 컴파일 내용을 보면 상세히 나와 있습니다.

## 17 아두이노 IDE 하위 디렉터리 정보

아두이노 IDE 는 C/C++ 컴파일러를 사용하고 있습니다. 거의 모든 컴퓨터 언어는 “라이브러리”라는 형식으로 확장 및 소스 코드의 형상 관리를 할 수 있습니다.

아두이노 IDE 는 기본 C/C++ 컴파일러와 많은 라이브러리들의 뮤음이라고 볼 수 있습니다. 아두이노에서 사용되는 라이브러리들은 아래와 같은 디렉터리에 있습니다.

아두이노 1.5.8 버전을 예로 들겠습니다. 물론 아두이노 사이트에서 배포되는 버전마다 IDE 프로그램에 뮤여 있는 라이브러리들은 약간의 차이점은 있지만, 구조는 동일하게 되어 있습니다.

아두이노 IDE 설치된 디렉터리는 아래와 같은 항목들이 존재합니다.

각각의 디렉터리의 파일들은 프로그래밍에 사용되고 있습니다.

### arduino-1.5.8\ 디렉터리.

| 이름                | 수정한 날짜           | 유형         | 크기      |
|-------------------|------------------|------------|---------|
| 파일 폴더 (8)         |                  |            |         |
| drivers           | 2015-01-31 오전... | 파일 폴더      |         |
| examples          | 2015-01-31 오전... | 파일 폴더      |         |
| hardware          | 2015-01-31 오전... | 파일 폴더      |         |
| java              | 2015-01-31 오전... | 파일 폴더      |         |
| lib               | 2015-01-31 오전... | 파일 폴더      |         |
| libraries         | 2015-01-31 오전... | 파일 폴더      |         |
| reference         | 2015-01-31 오전... | 파일 폴더      |         |
| tools             | 2015-01-31 오전... | 파일 폴더      |         |
| 응용 프로그램 확장 (5)    |                  |            |         |
| cygcc_s-1.dll     | 2014-10-01 오후... | 응용 프로그램 확장 | 102KB   |
| cygiconv-2.dll    | 2014-10-01 오후... | 응용 프로그램 확장 | 986KB   |
| cygwin1.dll       | 2014-10-01 오후... | 응용 프로그램 확장 | 3,041KB |
| cygz.dll          | 2014-10-01 오후... | 응용 프로그램 확장 | 73KB    |
| libusb0.dll       | 2014-10-01 오후... | 응용 프로그램 확장 | 43KB    |
| 응용 프로그램 (2)       |                  |            |         |
| arduino.exe       | 2014-10-01 오후... | 응용 프로그램    | 844KB   |
| arduino_debug.exe | 2014-10-01 오후... | 응용 프로그램    | 383KB   |
| TXT 파일 (1)        |                  |            |         |
| revisions.txt     | 2014-10-01 오후... | TXT 파일     | 55KB    |

그림 17-1 아두이노 설치된 디렉터리 하위 파일들

**Drivers:** 아두이노 보드에 대한 모든 드라이버 파일들이 있습니다.

디렉터리 아래의 dpinst-amd64.exe, dpinst-x86.exe 파일은 드라이버 일괄 설치 실행파일입니다. 제어판에서 별도의 드라이버 업데이트 또는 드라이버 찾아보기 기능을 사용하지 않고 실행만 해주면 자동으로 드라이버 설치가 되는 파일입니다. Dpinst-amd64.exe 파일은 64 비트 OS에서, dpinst-x86.exe 실행파일은 32 비트 OS에서 설치 실행 파일입니다.

드라이버 아래의 FTDI USB Drivers 디렉터리는 아두이노 나노 V3 등의 드라이버 설치 파일들이 있습니다. FTDI 제조사(<http://www.ftdichip.com>) 부품이 사용되는 보드의 드라이버 파일들이 있습니다. 아두이노 프로 미니 보드의 경우는 FTDI USB Programmer 를 사용합니다.

**Examples:** 기본 제공되는 예제 코드들이 있습니다.

아두이노에서 기본 제공되는 예제 코드들이 있습니다. 아두이노 프로그래밍에 대한 친절한 예제 코드로 구성되어 있습니다.

**Hardware:** 하드웨어에 대한 정의 파일들이 있습니다.

**Java:** 아두이노 IDE(통합환경 프로그램)는 사용자 인터페이스가 모두 JAVA 언어로 만들어져 있습니다. 아두이노 IDE 가 실행되기 위해서는 자바머신이 있어야 실행됩니다.

**Lib:** 아두이노 IDE에서 사용되는 자바 라이브러리입니다. IDE 프로그램의 사용자 인터페이스, 명령어 처리 등에 사용되는 자바 라이브러리 파일들이 있습니다.

**Libraries:** 아두이노의 실제 기본 라이브러리들이 있습니다. 아두이노 기본 라이브러리들은 가장 많이 사용되는 하드웨어 모듈들이 모두 있습니다. 아두이노를 사용하여 각종 하드웨어 사용을 편하게 하는 기본적인 라이브러리입니다. 가장 많이 사용되는 네트워크 모듈, 저장 장치 모듈, 서보모터, 디스플레이, 와이파이등의 아두이노와 호환되는 장치들의 기본 라이브러리 파일들이 있습니다.

**References:** 아두이노 라이브러리 및 기본 함수들에 대한 설명들이 있습니다.

**Tools:** 아두이노 IDE Tools 항목 작동에 관련되는 파일들이 있습니다.

Tools 항목은 사용자 라이브러리 디렉터리 아래에 Tools 라는 이름으로 디렉터리를 생성한 후 플러그인 파일을 복사하여 실행할 수 있습니다.

플러그인 형태는 자바 프로그램 형태입니다. 대표적인 Tools 프로그램으로는 ArduBlock 이라는 스크래치 프로그래밍 가능한 아두이노 플러그인입니다.

아두블럭 사이트 : <http://blog.ardublock.com/>

아두이노에서는 Tool 이라는 명칭을 사용하는 것으로 보아 도구로 분류가 됩니다.

## 18 아두이노 IDE 라이브러리 추가 및 사용하기

아두이노라는 플랫폼이 전 세계적으로 널리 사용되는 이유중의 하나는 쉽게 사용 가능한 라이브러리가 많아서입니다. 생소한 부품과 PCB 형태의 작은 모듈을 쉽게 사용 가능하게 합니다.

아두이노를 사용한다면 반드시 라이브러리 사용에 대한 작은 이해는 큰 결과를 낳게 해주는 결과가 됩니다. 반드시 숙지하고 사용하기 바랍니다.

아두이노는 하드웨어 모듈 통합 프로그래밍 통합환경(IDE)입니다.

아두이노 보드를 사용하여 프로젝트 진행하는 경우 여러 개의 하드웨어 모듈과 부품들을 사용하는 경우가 대부분입니다.

각각의 하드웨어 모듈들은 아두이노에서 수월하게 사용할 수 있도록 라이브러리 코드를 배포하고 있습니다. 아두이노 공식 사이트와 단체, 개인 블로그, 인터넷에서 많은 자료를 받아 사용할 수 있습니다.

아두이노를 사용하여 프로그래밍 하는 경우에 몇 개 이상의 라이브러리는 기본적으로 사용되고 있습니다.

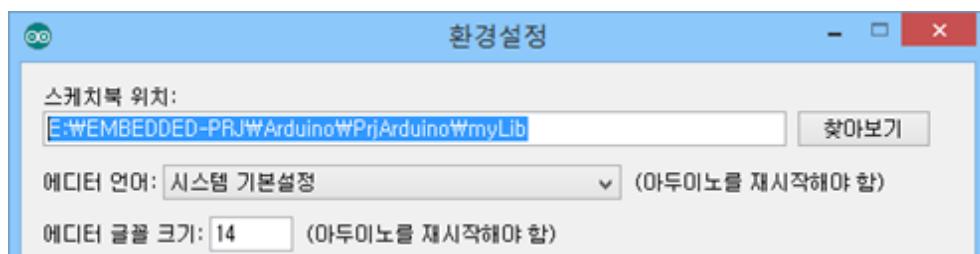
### 18.1 라이브러리 추가하기 전 주의사항

라이브러리 디렉터리의 명칭 정의 할 때 주의해야 할 사항이 있습니다.

라이브러리 디렉터리의 명칭은 영문자 알파벳을 사용해야 합니다.

### 18.2 사용자 라이브러리 디렉터리 (폴더) 확인.

“파일” 메뉴 → “환경 설정”에서 디렉터리 위치를 확인 할 수 있습니다.



MyDocuments 아래에 SketchLib 라는 “스케치북 위치” 기본 디렉터리가 지정되어 있다면 MyDocuments / SketchLib / libraries / MyLib 라는 디렉터리에 나만의 라이브러리 코드를 넣어두고 사용할 수 있습니다.

## Ex) MyLib

하지만 필요에 의해 MyLib 디렉터리 명칭이 아닌 My-Lib 라고 변경하여 사용하고 싶은 경우가 있습니다.

그럼 MyDocuments / SketchLib / libraries / My-Lib 디렉터리가 존재하게 됩니다.

최근 버전(1.5.6 상위버전)은 상관없이 라이브러리에 포함 시켜 사용 할 수 있지만, 필요에 의해 이전 버전의 아두이노 IDE(1.0.xx 버전 ~ 1.5.5 버전)를 사용할 경우에는 인식이 안 되는 디렉터리 명칭입니다.

아두이노 스케치 IDE 실행 시에 경고 메시지가 나옵니다.

이런 경우에는 My\_Lib 라고 이름을 변경하면 아두이노 IDE 하위버전, 상위버전 모두 사용 가능합니다.

### 18.3 라이브러리 추가 방법 1

아두이노 IDE의 메뉴를 이용할 수 있습니다.

“라이브러리 추가” 메뉴 항목은

스케치 IDE 메뉴 → 스케치 → 라이브러리 가져오기 → (서브 메뉴 상단) “라이브러리 추가”

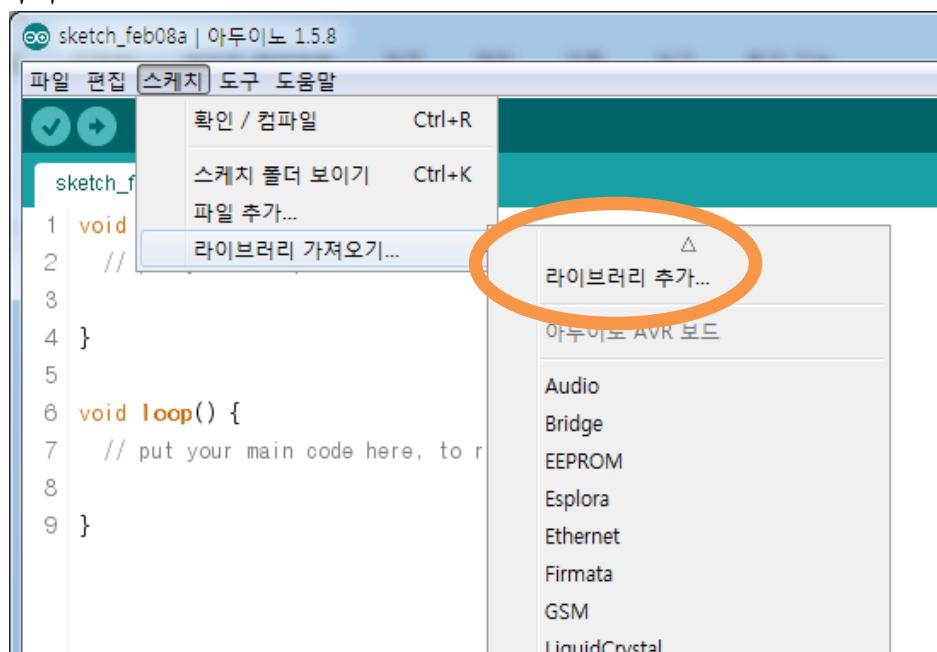


그림 18-2 라이브러리 추가 메뉴 선택

아두이노 1.6.4 버전에서는 상단의 메뉴 “스케치”-> “Include library” -> “Add Zip Library ...” 선택하면 됩니다.

아래와 같은 파일 선택ダイ얼로그가 나옵니다.

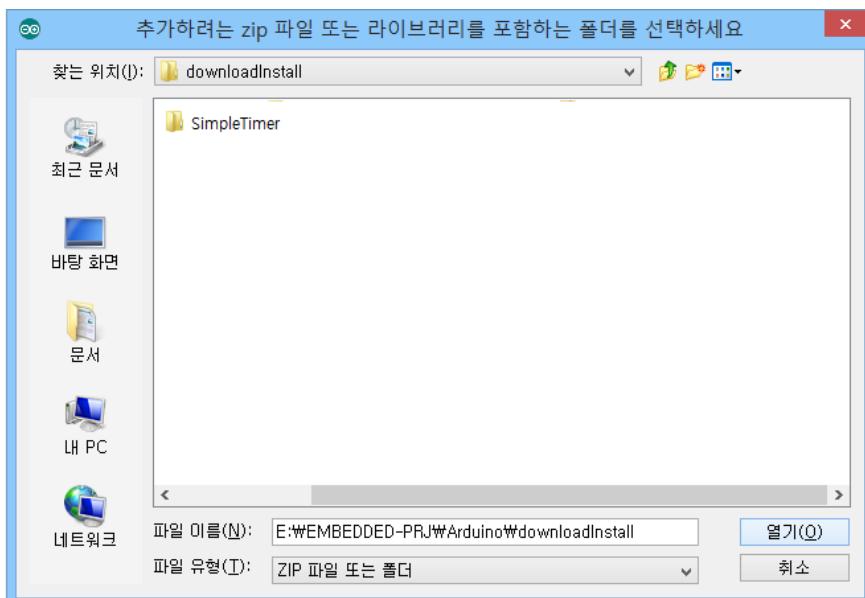


그림 18-3 라이브러리 선택ダイ얼로그

라이브러리 파일 ZIP (압축된 파일 형식) 또는 디렉터리를 지정할 수 있습니다.  
ZIP 파일 또는 폴더를 선택하면 Zip 파일일 경우 압축 해제되어 라이브러리 디렉터리(폴더)에 복사 됩니다.

## 18.4 라이브러리 추가 방법 2

환경 설정에 정의된 “스케치북 위치” 라이브러리 디렉터리 아래에 직접 파일들을 넣어줍니다.



“스케치북 위치” 디렉터리 (스케치 라이브러리 디렉터리 위치)가 “E:\EMBEDDED-PRJ\Arduino\Prj\Arduino\myLib”라고 설정된 상태일 경우  
윈도우 탐색기에서

"E:\EMBEDDED-PRJ\Arduino\PrjArduino\myLib\ libraries" 아래에 넣어 주면 됩니다.

예제로 "RTC1302.ZIP" 파일을 임의의 디렉터리(폴더) 아래에 RTC1302 라는 명칭으로 압축 해제를 하게 되면, 보통 RTC1302 안에는 라이브러리 소스 파일들과 examples라는 예제 코드 디렉터리가 있습니다.

그럼 RTC1302라는 디렉터리를 스케치 라이브러리 디렉터리에 복사(또는 이동)하여 넣어 주도록 합니다.

"E:\EMBEDDED-PRJ\Arduino\PrjArduino\myLib\ libraries" 디렉터리 아래에 복사하여 줍니다.

그럼,

"E:\EMBEDDED-PRJ\Arduino\PrjArduino\myLib\ libraries\ RTC1302" 디렉터리가 존재하게 됩니다.

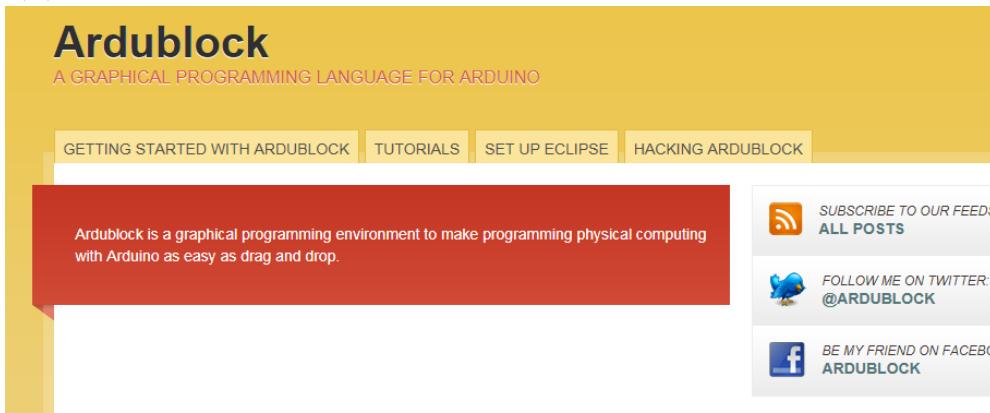
아두이노 IDE를 재실행 해야 위에 적용된 라이브러리가 적용됩니다.

아두이노 IDE의 메뉴의 라이브러리 추가 메뉴를 이용하여 추가 할 경우 재실행하지 않아도 됩니다. 하지만, 깔끔한 적용을 위하여 재실행 권장 합니다.

## 19 아두블럭 간단한 소개 및 사용법

아두이노 IDE 플러그인 방식의 외부 프로그램입니다. 스크래치 형태의 프로그래밍 방식이 가능한 프로그램중의 하나입니다.

아두블럭 도구는 <http://blog.ardublock.com/> 사이트에서 다운로드 받아서 사용합니다.



아두블럭 사이트에 방문 후 왼쪽 상단의 “Getting Started With ArduBlock” 선택하면 다운로드 받을 수 있습니다.

» 아두블럭 보다는 좀 더 발전된 형태의 스크래치 프로그램으로 S4A(Scratch for Arduino)라는 <http://s4a.cat> 프로그램도 있습니다.

별도의 실행 프로그램 형식으로 동일하게 스크래치 방식으로 프로그래밍 하도록 되어 있습니다. 아두블럭과 다른 점은 S4A 고유의 파일 형식으로 저장되어 별도의 C/C++ 코드를 볼 수 없습니다. 스크래치 방식에 익숙해지면 C/C++ 형식으로 변환된 코드는 볼 필요는 없습니다.

### 19.1 아두블럭 설치 방법

아두이노 사용자 라이브러리 디렉터리 아래에 Tools에 복사하여 넣어주면 됩니다.

“tools”라는 디렉터리가 없는 경우 직접 생성하도록 합니다.

» 아두이노 사용자 라이브러리 디렉터리가 “C:\Arduino\PrjArduino\myLib”로 지정된 경우, “C:\Arduino\PrjArduino\myLib\libraries”에는 사용자 라이브러리들이 있습니다. 아두블럭은 “C:\Arduino\PrjArduino\myLib\Tools”라는 디렉터리 아래

에 있어야 됩니다. Tools 디렉터리는 아두이노 IDE 플러그인 프로그램이 위치하고 있습니다.

“C:\Arduino\PrjArduino\myLib\tools\ArduBlockTool” 디렉터리에 정상적으로 설치가 되어 있는 경우 아두이노 IDE 를 실행하면 아래와 같은 도구 메뉴 아래에 “ArduBlock” 이라는 항목이 보입니다.



그림 19-1 아두블럭 메뉴항목

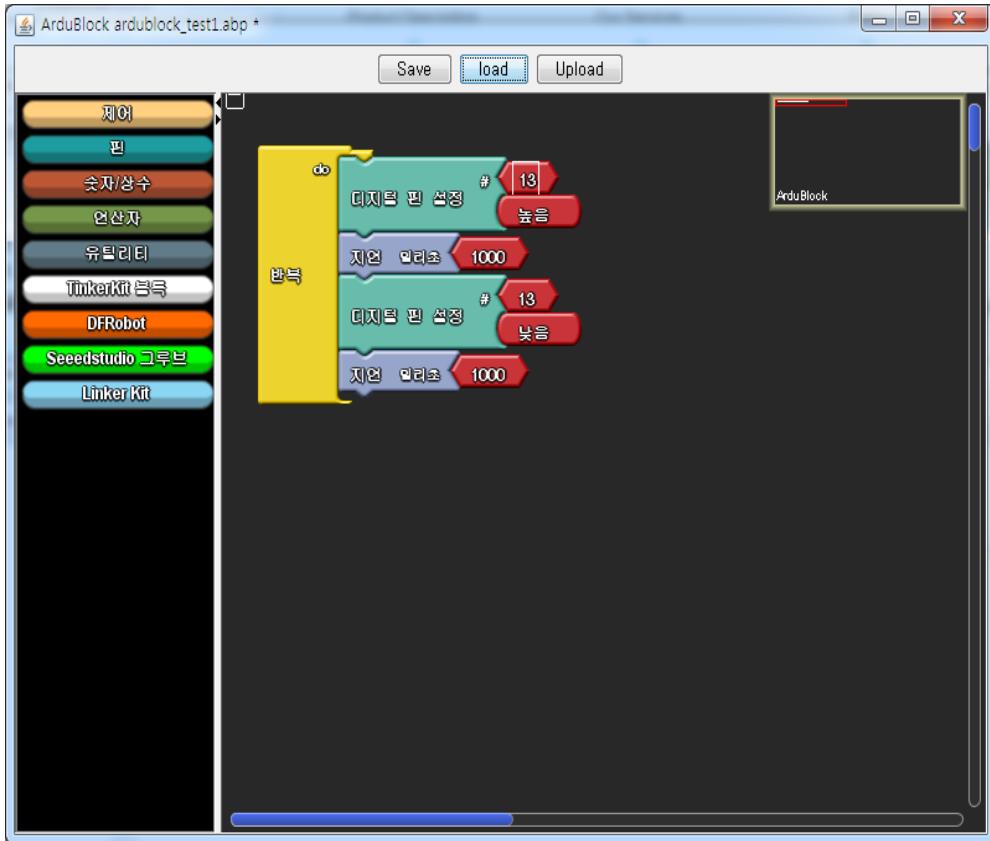


그림 19-2 아두블럭 실행 화면

아두블럭에서 D13 번 포트에 LED 를 1 초 동안 On/Off 시키는 예제입니다.  
상단의 “Save” “Load” 버튼을 누르면 프로그래밍 되는 블록의 내용을 저장 및 가져오기를 할 수 있습니다.  
“Upload” 버튼을 누르면 블록 형태로 프로그래밍된 내용이 코드로 변경되어 업로드가 됩니다.



그림 19-3 스크래치 블럭

sketch\_may05a | 아두이노 1.5.8

파일 편집 스케치 도구 도움말

sketch\_may05a §

```
1 void setup()
2 {
3   pinMode( 13 , OUTPUT );
4 }
5
6 void loop()
7 {
8   digitalWrite( 13 , HIGH );
9   delay( 1000 );
10  digitalWrite( 13 , LOW );
11  delay( 1000 );
12 }
```

업로드 완료

avrduude done. Thank you.

14 Arduino Uno on COM14

그림 19-4 아두블럭에서 생성된 코드 내용

더 많은 정보는 <https://scratch.mit.edu/> 또는 <http://blog.ardublock.com/> 사이트를 방문하면 많은 예제와 정보가 있습니다. 좀 더 친숙한 아두이노 프로그래밍을 원하거나 다양한 응용 프로그래밍까지 배울 수 있습니다.

참고로 아두블럭 사용시 블록을 제거할 때는 블록을 마우스로 잡아서 왼쪽 영역으로 넣으면 사라집니다.

## 20 아두이노 프로세싱

아두이노 프로세싱 “Arduino Processing” 프로그램이 있습니다.

아두이노 프로그래밍을 그대로 사용하면 시각적인 결과물을 표현하기에 적합한 프로그램입니다.

<http://processing.org> 사이트 방문 후 다운로드 하여 설치하면 바로 사용할 수 있습니다. 많은 예제들이 존재하여 아두이노 프로젝트의 결과물을 시각적으로 다양하게 표현 가능합니다.

다운로드: <https://processing.org/download/?processing>

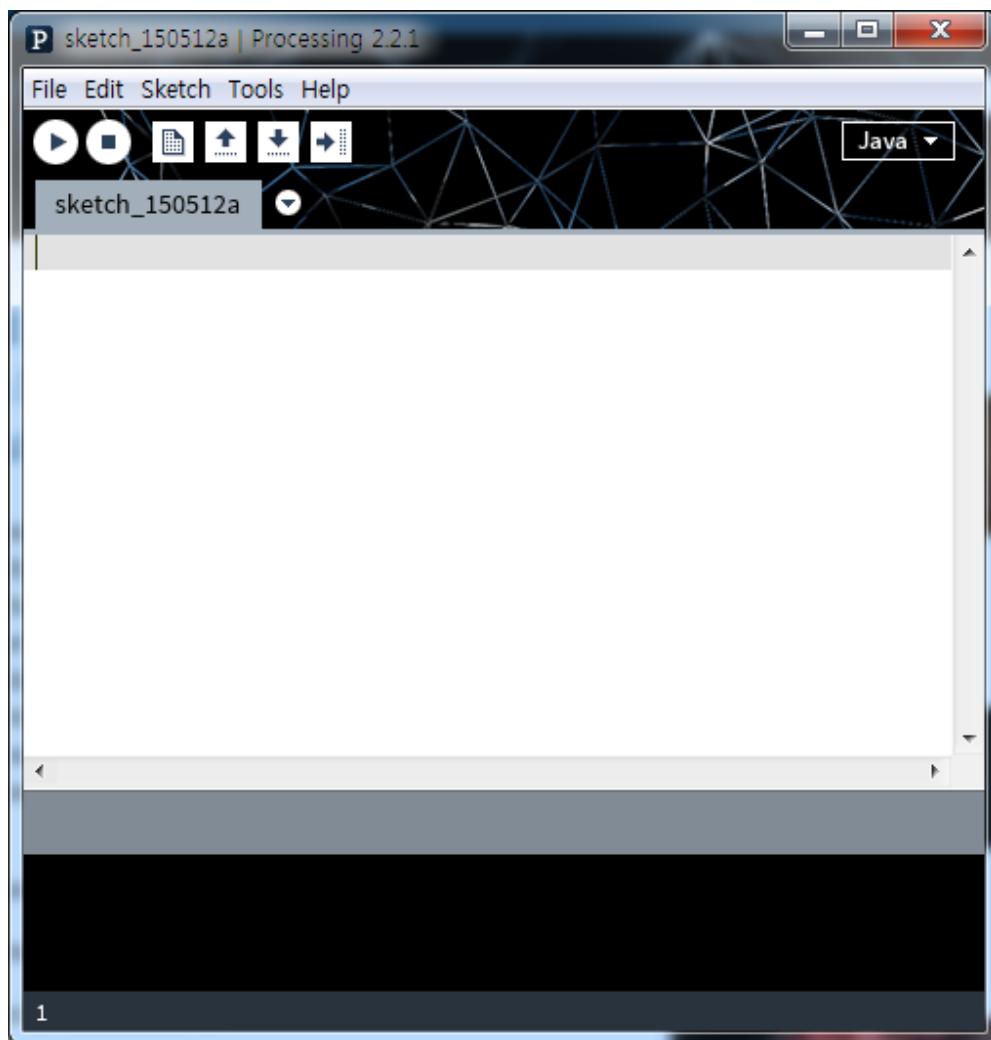


그림 20-1 아두이노 프로세싱 메인 화면

직관적인 사용 예제는 파일 메뉴->Examples 선택하면 프로세싱에서 사용 가능 및 참조 가능한 예제들이 많이 보입니다.

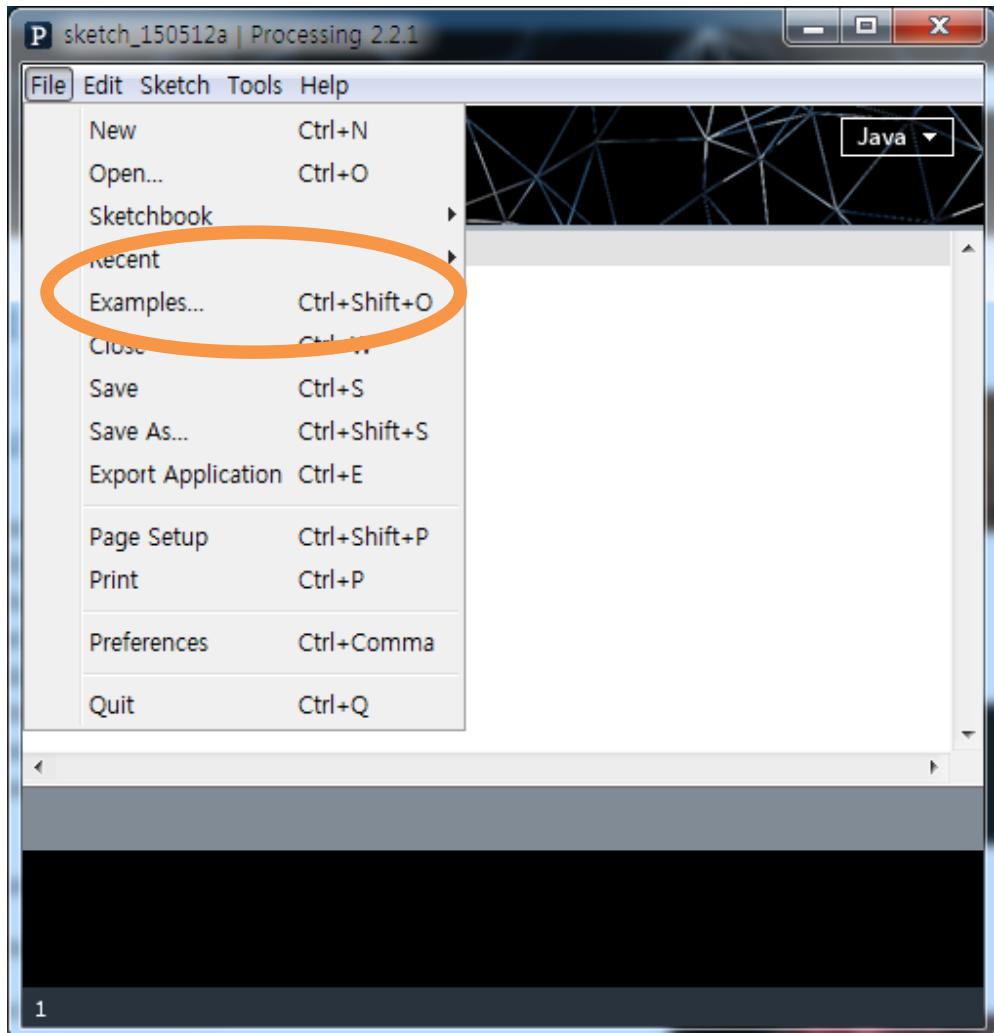


그림 20-2 예제 선택

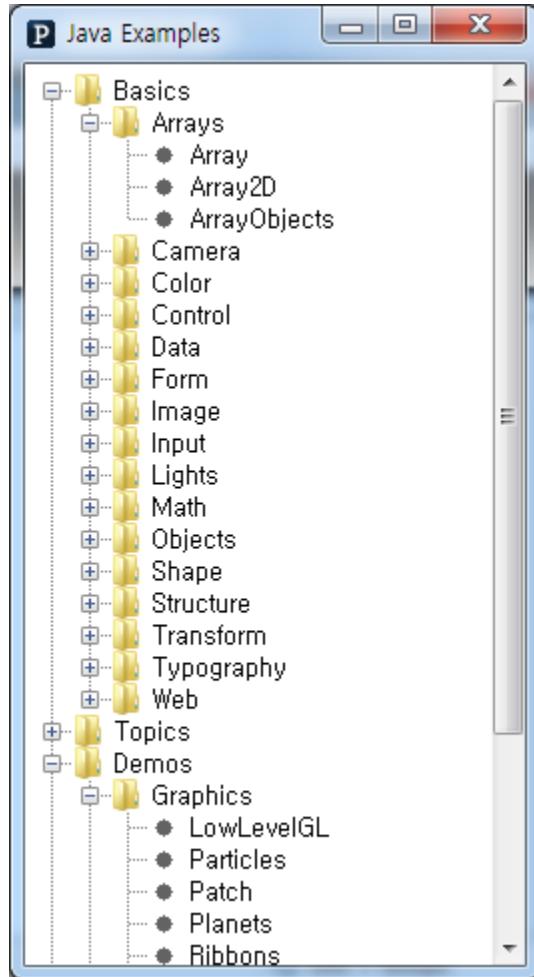


그림 20-3 예제 목록 화면

## 20.1 프로세싱 프로그램과 아두이노 연동 방식

프로세싱 프로그램에서는 아두이노 보드와의 시리얼 통신에 의한 데이터 교환에 의한 시각적인 표시를 할 수 있습니다. 프로세싱과의 시리얼 통신 환경은 아두이노 보드가 정상적으로 USB 연결 되어 있으면 됩니다. 윈도우 제어판의 장치관리자->포트 항목에 반드시 아두이노 보드의 시리얼 포트가 있어야 정상 테스트 가능합니다.

먼저 아래의 아두이노 스케치 코드를 아두이노 보드에 업로드 하여 줍니다.

예제) 아두이노 보드에서는 시리얼 포트로 읽어 들어 들인 값을 LED 의 밝기를 설정하는 예제 코드:

```
//
```

```

// 시리얼 통신으로 데이터를 받아 9 번 포트에 연결된 LED 밝기를 조절합니다.
// 9 번 포트는 PWM 포트입니다.
//
const int led = 9;
int val; // led 밝기 값.

void setup()
{
    Serial.begin(9600); // pinMode(led, OUTPUT); //
}

void loop() {
    if (Serial.available()) { //시리얼 포트에 데이터가 있다면
        val = Serial.read(); //
        analogWrite(led, val); // 받은 값으로 LED 밝기 값 설정.
    }
}

```

프로세싱 IDE에서는 아래와 같은 코드를 실행합니다.

마우스 x, y 좌표를 화면에 출력해주는 간단한 예제 코드입니다.

```

import processing.serial.*; //import serial // 시리얼 포트 사용.

Serial port; // 시리얼 포트 변수.

void setup()
{
    size(200, 150); // 실행 창 크기.
    port = new Serial(this, Serial.list()[1], 9600);
}

void draw()
{
    background(0, 0, 255); // 배경 색상 지정.
    textSize(50); // 화면에 표시되는 문자 크기 지정.
    text(mouseX, width/4, height*2/3); // 문자열 화면에 출력.
    port.write(mouseX); // 시리얼 포트에 mouseX 값 전달.
}

```

프로세싱 예제 코드를 실행하면 파란색 바탕에 흰색 숫자가 나옵니다. 숫자는 코드의 내용과 같이 마우스 X 값입니다.

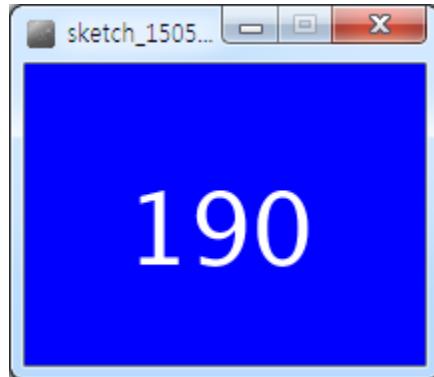


그림 20-4 프로세싱 예제 프로그램 실행 화면

마우스 X 값은 사용자의 PC OS 환경에 따라 0~1024, 0~1280, 0~1600 값이 표시 될 수 있습니다. 아두이노에서의 PWM 포트로의 analogWrite 할 수 있는 값의 범위는 0~255 까지입니다. C/C++ 함수중에 map 이라는 함수로 프로세싱에서 받은 X 값을 0~255 범위의 값으로 변경을 해줍니다.

아래와 같이 코드 변경 후 아두이노에 업로드 해 줍니다.

```
//  
// 시리얼 통신으로 데이터를 받아 9 번 포트에 연결된 LED 밝기를  
조절합니다.  
// 9 번 포트는 PWM 포트입니다.  
//  
const int led = 9;  
int val; // led 밝기 값.  
  
void setup()  
{  
    Serial.begin(9600); //  
    pinMode(led, OUTPUT); //  
}  
  
void loop()  
{  
    if (Serial.available()) { //시리얼 포트에 데이터가 있다면  
        val = Serial.read(); //  
        val = map(0,1280,0,255);  
        analogWrite(led, val); // 받은 값으로 LED 밝기 값 설정.  
    }  
}
```

## 20.2 아두이노 프로세싱의 시리얼 포트 사용시 주의점:

아두이노 프로세싱 프로그램은 자바 언어를 기반으로 작성해야 합니다.  
아래와 같이 시리얼 포트를 사용하기 위해서는 #include 와 같은 개념인

```
import processing.serial.*; // 시리얼 포트 자바 모듈 사용 선언.
```

```
Serial port; // 시리얼 포트 변수.
```

```
port = new Serial(this, Serial.list()[1], 9600);
```

port라는 변수에 Serial 클래스를 할당하면서 포트와 보레이트 설정을 할 수 있습니다.

Serial.list() 호출을 하면 현재 PC의 포트의 목록 정보가 반환됩니다.

Serial.list()[1] 의미는 시리얼 포트의 목록중 2 번째 항목을 사용한다는 의미입니다.

사용하는 윈도우의 장치관리자의 포트를 보면 아래와 같은 정보를 볼 수 있습니다.



그림 20-5 장치 관리자의 포트 목록

Serial.list()[1]을 사용하는 경우 반환되는 포트의 정보는 두 번째 시리얼 포트 항목입니다.

정확하게 사용하기 위해서는 아두이노 보드가 연결된 첫 번째 항목 Serial.list()[0]을 사용해야 합니다.

사용하는 PC 환경에 따라 적절히 수정하면 즉각적인 결과를 볼 수 있습니다.

## 20.3 아두이노와 미니 PC의 미래

아두이노와 프로세싱을 활용한 많은 흥미로운 주제의 프로젝트가 생각보다는 상당히 많습니다.

프로세싱은 자바와 오픈지엘(OpenGL) API를 사용하도록 되어 있어 빠른 하드웨어 가속 렌더링 가능한 구조입니다.

하지만 PC에서 구동되는 프로세싱 프로그램의 구조 때문에 현장 설치, 필드에서의 독립적인 아두이노 프로젝트 활용은 불편하기는 합니다.

하지만, 1~2년 안에 손바닥 PC(지금의 PC 성능) 솔루션이 속속 출현하는 상태이므로 아두이노와, 손바닥 PC의 활용은 무궁무진하리라 보여집니다.

PC 자체의 통신 방식은 모두 사용하면서, 빠른 처리로 인해 수많은 프로그램들을 사용할 수 있게 됩니다.

프로세싱이라는 프로그램이 손바닥 PC의 모니터에서 볼 수 있다는 결과는, 좀 더 복잡하고 다양한 복잡한 작동 및 결과들을 손안에서 처리할 수 있게 됩니다.

## 21 >HIGH QUALITY AND LARGE BREAD BOARD X 1

830 Points Solderless Breadboard. 830 포인트 브레드보드입니다.

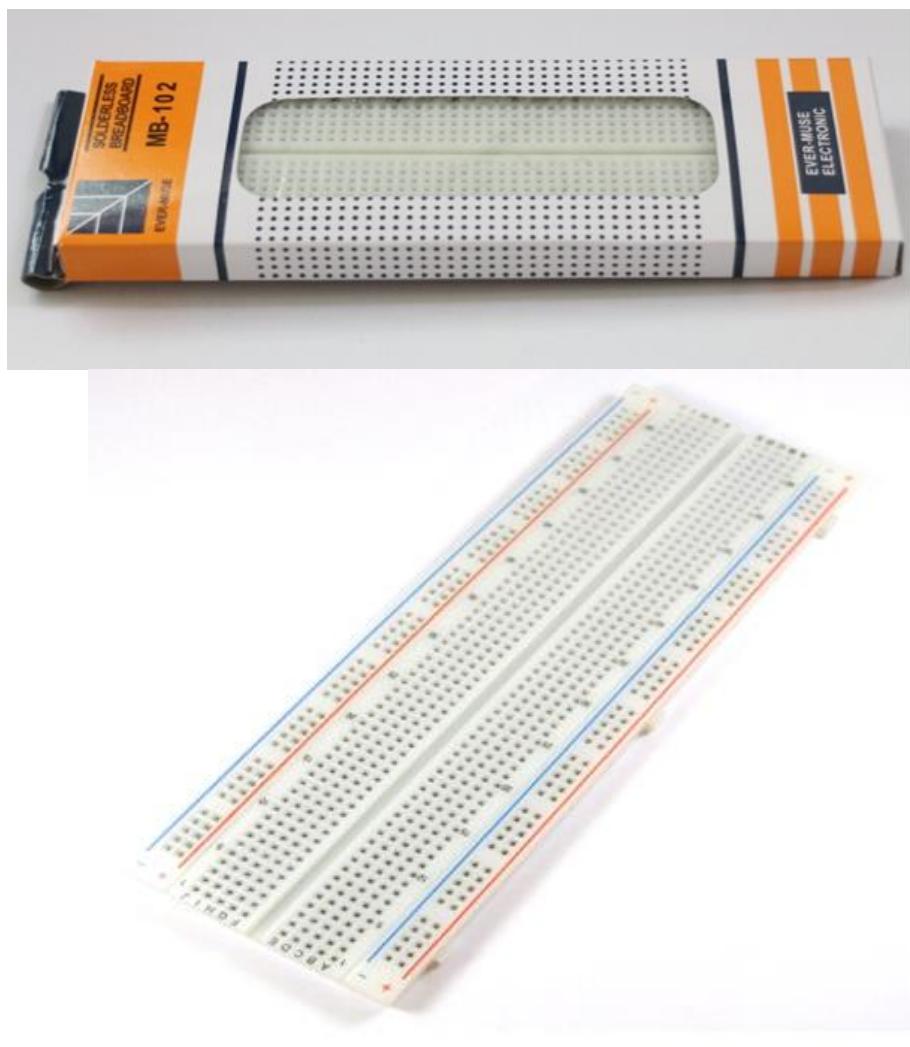


그림 21-1 브레드보드(Breadboard)

브레드보드는 전자 회로 적용 및 이해를 위한 용도로 많이 사용 됩니다.

브레드보드는 납땜하지 않고 전자회로를 구성해볼 수 있도록 해주는 장비입니다.  
빵판이라고 불리우기도 합니다.

## 21.1 버스영역

브래드보드의 (+) 표시부분과 (-) 표시 부분은 전원 공급 공통 부분입니다.

버스영역이라고 합니다.

(+) 부분은 파워(전원) 버스

(-) 부분은 그라운드 버스

## 21.2 IC 영역

중앙의 5 칸씩 되어 있는 부분은 IC 영역이라고 합니다.

부품 및 점퍼선 등을 연결할 수 있습니다.

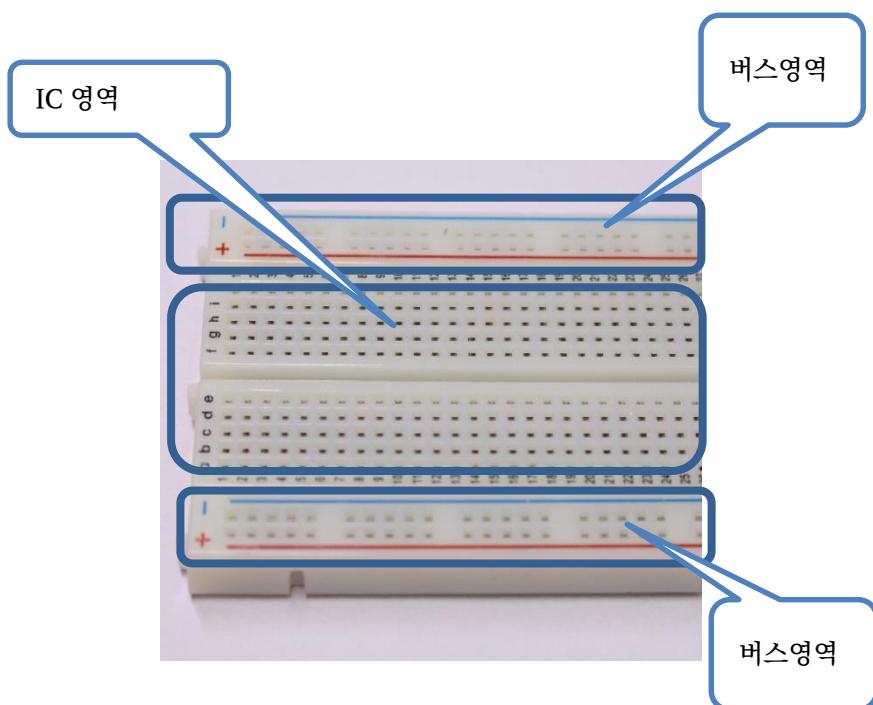


그림 21-2 브래드보드의 구분 영역

### 21.3 내부 구조

브레드보드는 아래의 이미지와 같이 내부적으로 나란히 연결된 부분들이 있습니다. 빨간색과 파란색 표시선은 전원과 그라운드 표시입니다.

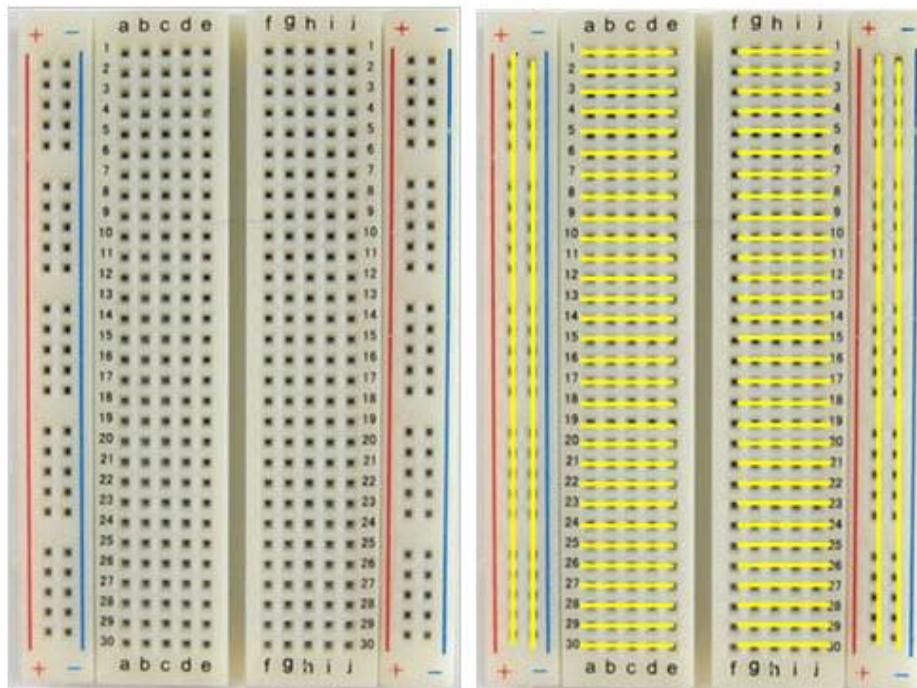


그림 21-3 내부 연결된 구조

노란색 표시된 부분은 내부적으로 전기가 통하는 연결된 구조를 나타냅니다.

공통으로 연결된 부분에는 여러 개의 핀을 연결 가능하게 되어 회로를 구성할 수 있습니다. 즉, 연결된 부분에 점퍼 선을 연결하면 전기가 통하게 되어 동일한 연결 회로 구성이 가능합니다.

## 22 > RFID MODULE KIT

아두이노에서 사용 가능한 RFID 모듈 키트입니다.

아두이노와 여러 MCU 보드에서 사용 가능한 RFID 키트입니다.

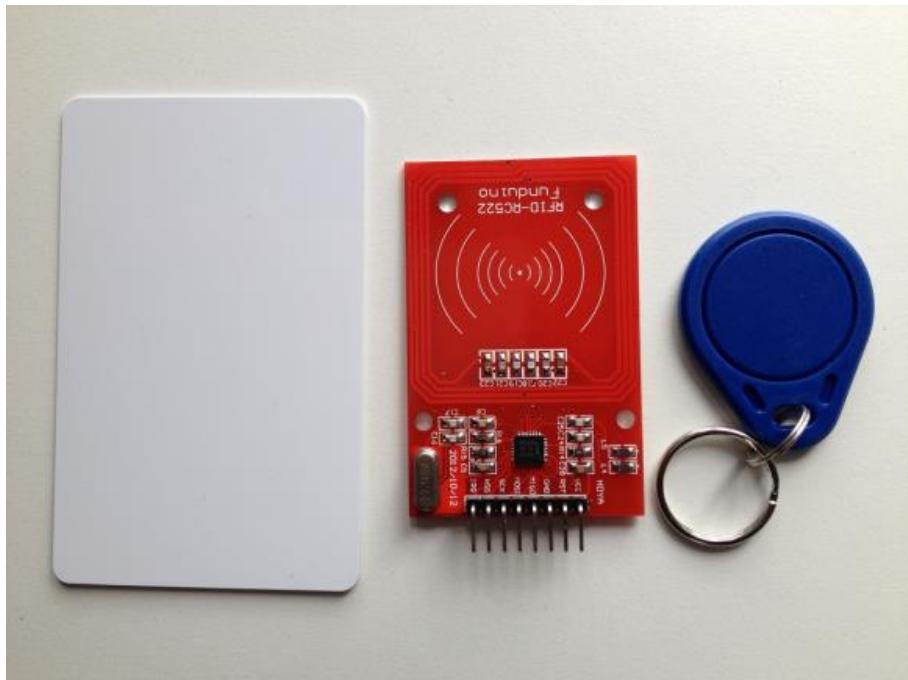


그림 22-1 RFID 구성 항목들

RFID (Radio-Frequency Identification) 는 전파를 이용한 근/원거리 통신 기술입니다.

RFID 기술은 산업/가전 하드웨어 제품에 널리 사용 되는 기능입니다.

RFID 제품 분야는 찾아보면 생각보다 많습니다. 대중교통 버스, 지하철, 아파트 주차장 입구, 출입 통제되는 입구, 공장 자동화 시스템 등에 많이 사용됩니다.

인식 태그 형태는 신용카드 크기가 주로 사용되며 열쇠 모양, 작은 크기의 마스터키, 부착형 스티커 등등 다양합니다.

보통 사용되는 RFID 태그와 판독기의 인식 거리는 5 Cm 정도입니다.

RFID 태그는 내부에 소형의 접적 회로가 있습니다 접적 회로에는 전파 수신이 잘되는 안테나가 물려 있는 형태입니다.

## 22.1 > RFID MODULE X 1

RFID-RC522 모듈 본체 입니다.



그림 22-2 RFID 모듈 본체

모듈의 기본 핀 배열은 SPI 통신 방식으로 사용됩니다.

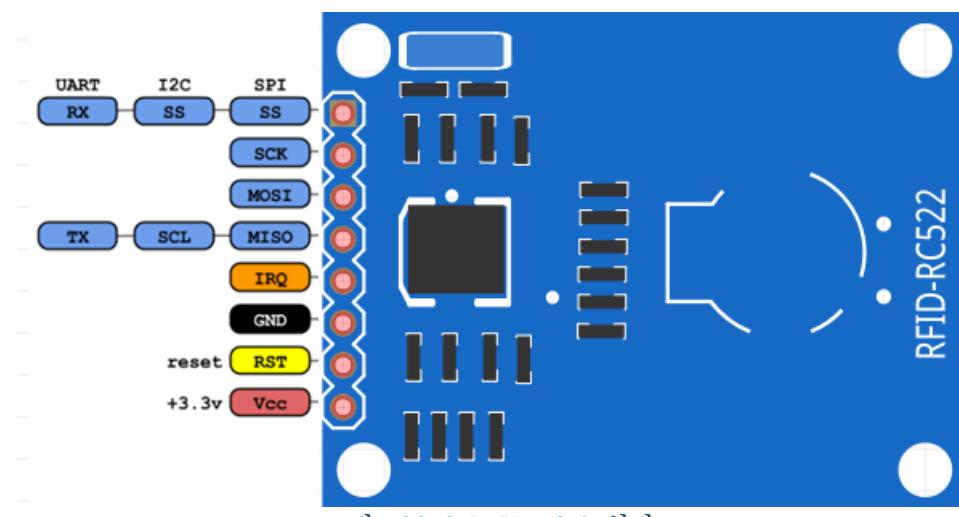


그림 22-3 RFID 포트 설명

## 22.2 > IC KEYCHAIN X 1

열쇠 고리 모양의 RFID 태그입니다.  
내부에는 작은 RFID 판독기에서 인식 가능한 칩이 들어 있습니다.  
보통 마스터 키, 관리자 용도로 많이 사용됩니다.



그림 22-4 열쇠고리 형태의 TAG

## 22.3 > NON-CONTACT TYPE IC CARD X 1

RFID에 사용 가능한 태그는 ISO/IEC 14443-1:2008 또는 ISO/IEC 14443 입니다.  
신용카드 크기입니다. 내부에도 RFID 모듈에서 인식 가능한 작은 칩이 들어 있습니다.



그림 22-5 RFID 카드

## 22.4 아두이노 보드와 연결

SPI 통신 모듈이므로 아두이노의 SPI 통신 포트와 연결을 합니다.

RC522 보드 8 핀 표시 순서대로 나열 합니다.

| RFID_RC522 모듈 | 아두이노 우노 & 프로 미니 & 나노 V3  |      |
|---------------|--------------------------|------|
| SDA           | D10                      | SS   |
| SCK           | D13                      | SCK  |
| MOSI          | D11                      | MOSI |
| MISO          | D12                      | MISO |
| IRQ           | NC (연결 안 함)              |      |
| GND           | GND                      |      |
| RST           | NC or D9 – (예제 라이브러리 참조) |      |
| 3.3V          | 3.3V                     |      |

RFID-RC522 IC DataSheet:

<http://www.gameplusedu.com/pds/gpshop/arduino/pdf/MFRC522.pdf>

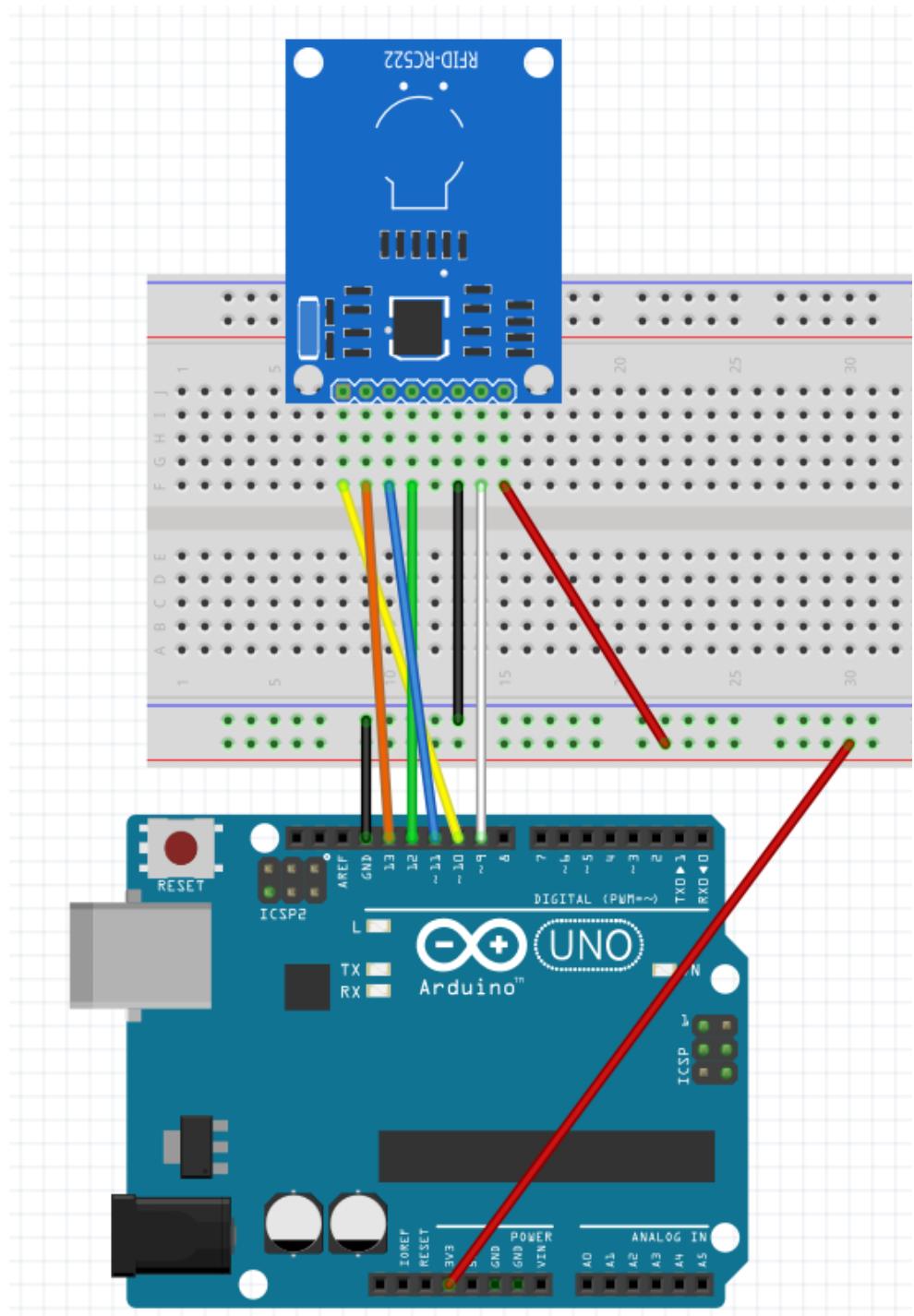
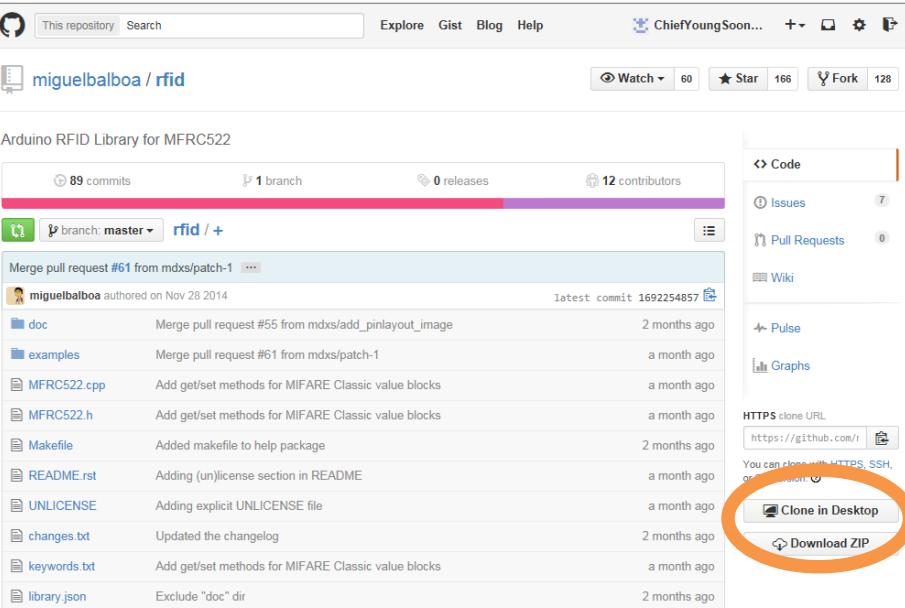


그림 22-6 브레드보드 회로도

아두이노 페모 예제 & 샘플 코드:  
<https://github.com/miguelbalboa/rfid>

위의 주소를 열면 아래와 같은 웹브라우저의 화면이 나옵니다.  
E-BOOK 페이지에서 클릭 후 웹브라우저에서 제대로 안 보이는 경우에는 위의 주소를 웹브라우저 주소 열기에서 입력하여 열기를 합니다.



Arduino RFID Library for MFRC522

89 commits 1 branch 0 releases 12 contributors

branch: master rfid / +

Merge pull request #61 from mdxs/patch-1

miguelbalboa authored on Nov 28 2014

latest commit 1692254857

| File         | Description  | Time Ago     |
|--------------|--|--------------|
| doc          | Merge pull request #55 from mdxs/add_pinlayout_image | 2 months ago |
| examples     | Merge pull request #61 from mdxs/patch-1             | a month ago  |
| MFRC522.cpp  | Add get/set methods for MIFARE Classic value blocks  | a month ago  |
| MFRC522.h    | Add get/set methods for MIFARE Classic value blocks  | a month ago  |
| Makefile     | Added makefile to help package                       | 2 months ago |
| README.rst   | Adding (un)license section in README                 | a month ago  |
| UNLICENSE    | Adding explicit UNLICENSE file                       | a month ago  |
| changes.txt  | Updated the changelog                                | 2 months ago |
| keywords.txt | Add get/set methods for MIFARE Classic value blocks  | a month ago  |
| library.json | Exclude "doc" dir                                    | 2 months ago |

Code Issues Pull Requests Wiki Pulse Graphs

HTTPS clone URL  
https://github.com/miguelbalboa/rfid

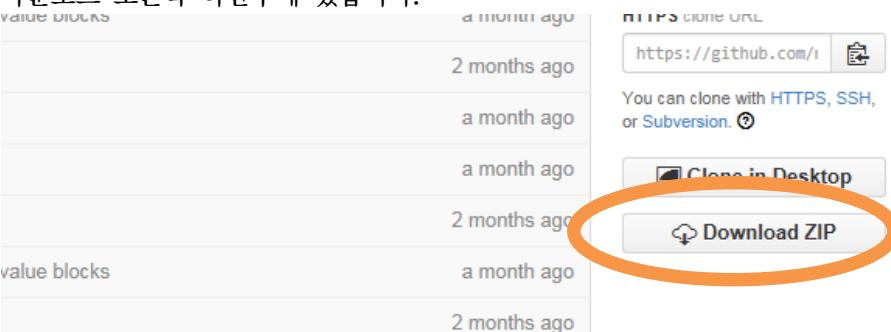
You can clone with HTTPS, SSH, or Subversion.

Clone in Desktop Download ZIP

그림 22-7 github rfid 리포지터리

위 주소에서의 예제 코드에는 우노 & 메가 보드에서의 사용 설명이 포함 되어 있습니다.

예제코드를 다운로드 받습니다.  
다운로드 오른쪽 하단부에 있습니다.



value blocks

2 months ago

a month ago

a month ago

2 months ago

a month ago

2 months ago

value blocks

a month ago

2 months ago

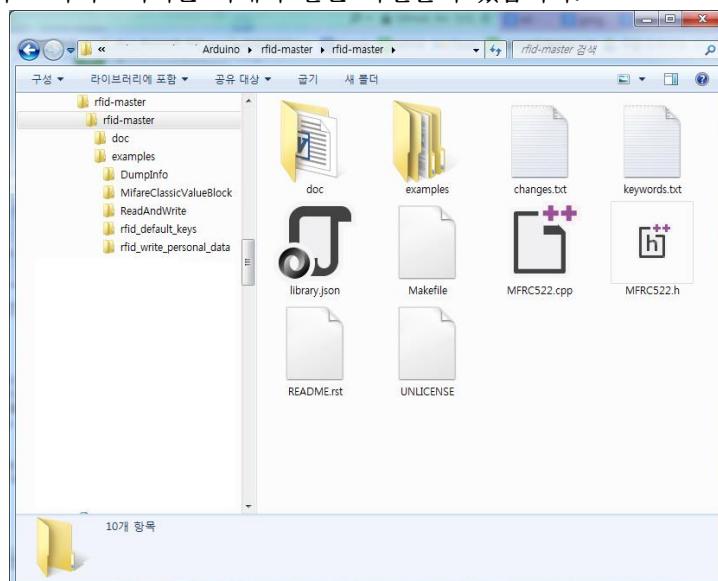
“Download ZIP” 클릭하면 현재까지의 배포되는 버전 코드의 압축된 묶음을 받게 됩니다.

다운로드 하면 “rfid-master.zip”이라는 파일을 임의의 디렉터리에 저장을 합니다.

rfid-master.zip은 아두이노 IDE에 설치를 해줍니다.

Rfid-master.zip 파일을 압축 해제 후 아두이노 IDE 라이브러리 설치하는 경우 주의 할 점은 rfid-master라는 라이브러리 디렉터리 명칭이 들어가면 제대로 로딩이 안 될 경우가 있습니다. rfid\_master라는 명칭처럼 변경 후 라이브러리 디렉터리 복사해주면 됩니다. “-“ 하이픈 기호를 “\_” 언더 바 기호로 변경합니다.

Rfid 아두이노 라이브러리는 아래와 같은 파일들이 있습니다.



Examples 디렉터리의 아래에 여러 가지 예제가 있습니다.

DumpInfo라는 예제를 사용해보도록 합니다.

DumpInfo 예제는 읽어 들인 카드의 데이터를 시리얼 포트로 출력해주는 예제입니다.

예제코드)

```
/*
 * -----
 * This is a MFRC522 library example; see
https://github.com/miguelbalboa/rfid
 * for further details and other examples.
 *
 * NOTE: The library file MFRC522.h has a lot of useful info. Please read it.
 *
 * Released into the public domain.
 * -----
 */
```

\* Example sketch/program showing how to read data from a PICC (that is:  
 a RFID  
 \* Tag or Card) using a MFRC522 based RFID Reader on the Arduino SPI  
 interface.  
 \*  
 \* When the Arduino and the MFRC522 module are connected (see the pin  
 layout  
 \* below), load this sketch into Arduino IDE then verify/compile and upload  
 it.  
 \* To see the output: use Tools, Serial Monitor of the IDE (hit Ctrl+Shift+M).  
 \* When you present a PICC (that is: a RFID Tag or Card) at reading distance  
 \* of the MFRC522 Reader/PCD, the serial output will show the ID/UID, type  
 and  
 \* any data blocks it can read. Note: you may see "Timeout in  
 communication"  
 \* messages when removing the PICC from reading distance too early.  
 \*  
 \* If your reader supports it, this sketch/program will read all the PICCs  
 \* presented (that is: multiple tag reading). So if you stack two or more  
 \* PICCs on top of each other and present them to the reader, it will first  
 \* output all details of the first and then the next PICC. Note that this  
 \* may take some time as all data blocks are dumped, so keep the PICCs at  
 \* reading distance until complete.  
 \*  
 \* Typical pin layout used:  
 \* -----
 

| Signal      | MFRC522 Reader/PCD Pin | Arduino Uno Pin | Arduino Mega Pin | Arduino Nano v3 Pin |
|-------------|------------------------|-----------------|------------------|---------------------|
| * RST/Reset | RST                    | 9               | 5                | D9                  |
| * SPI SS    | SDA(SS)                | 10              | 53               | D10                 |
| * SPI MOSI  | MOSI                   | 11 / ICSP-4     | 51               | D11                 |
| * SPI MISO  | MISO                   | 12 / ICSP-1     | 50               | D12                 |
| * SPI SCK   | SCK                    | 13 / ICSP-3     | 52               | D13                 |

 \*/

// 위의 내용은 SPI 포트에 연결하는 방법을 설명하고 있습니다.

```

#include <SPI.h>
#include <MFRC522.h>

#define RST_PIN      9      //
#define SS_PIN       10     //

MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 instance

void setup() {
    Serial.begin(9600); // Initialize serial communications with the PC
    SPI.begin();         // Init SPI bus
    mfrc522.PCD_Init(); // Init MFRC522

    ShowReaderDetails(); // Show details of PCD - MFRC522 Card
    Reader details

    Serial.println("Scan PICC to see UID, type, and data blocks...");
}

void loop() {
    // Look for new cards
    if (!mfrc522.PICC_IsNewCardPresent()) {
        return;
    }

    // Select one of the cards
    if (!mfrc522.PICC_ReadCardSerial()) {
        return;
    }

    // Dump debug info about the card; PICC_HaltA() is automatically called
    mfrc522.PICC_DumpToSerial(&(mfrc522.uid));
}

void ShowReaderDetails() {
    // Get the MFRC522 software version
    byte v = mfrc522.PCD_ReadRegister(mfrc522.VersionReg);
    Serial.print("MFRC522 Software Version: 0x");
    Serial.print(v, HEX);
    if (v == 0x91)

```

```
Serial.print(" = v1.0");
else if (v == 0x92)
    Serial.print(" = v2.0");
else
    Serial.print(" (unknown)");
Serial.println();
// When 0x00 or 0xFF is returned, communication probably failed
if ((v == 0x00) || (v == 0xFF)) {
    Serial.println("WARNING: Communication failure, is the MFRC522
properly connected?");
}
}
```

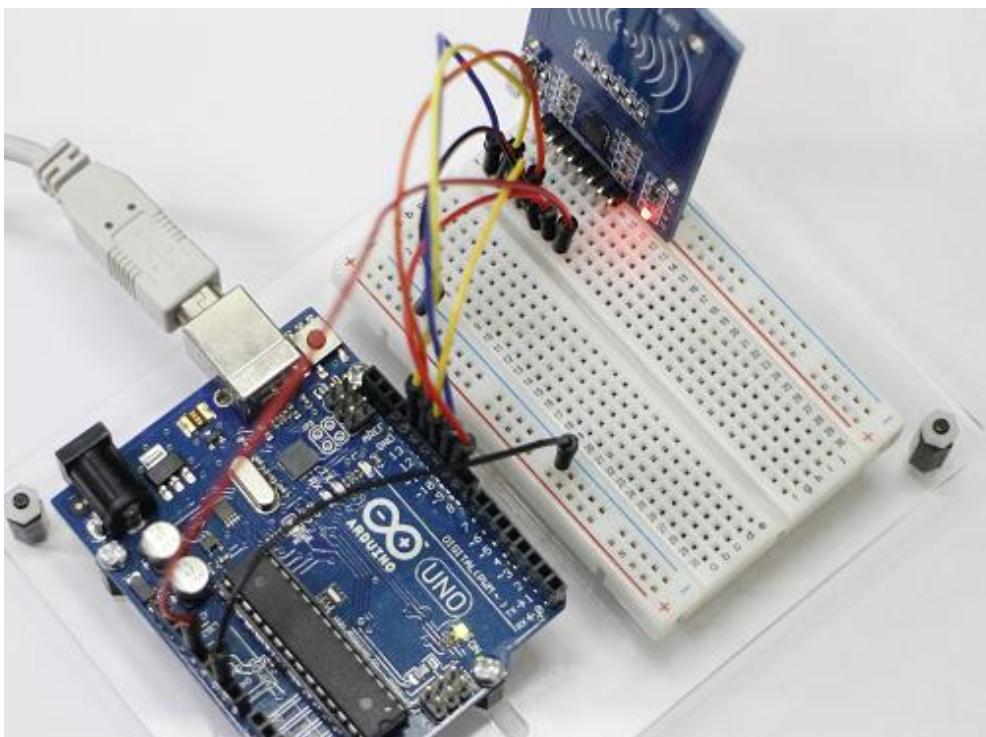


그림 22-8 RFID 브레드보드 회로 구성 사용

프로그램 업로드 후 시리얼 모니터 창을 열면 다음과 같은 화면이 나옵니다.  
정상 연결된 상태인 경우 MFRC52 모듈의 버전이 보입니다.

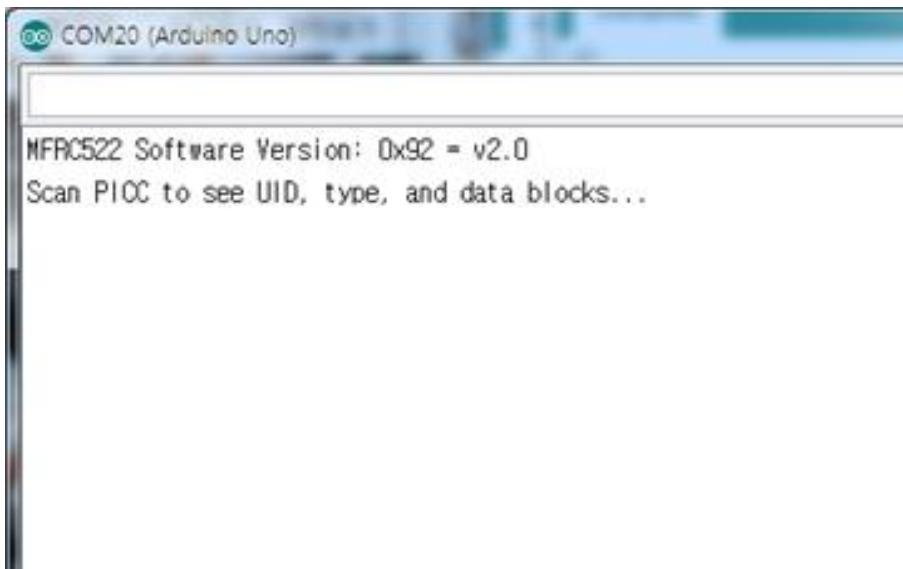


그림 22-9 실행 후 시리얼 모니터 화면

카드 형태 또는 열쇠 고리 모양의 RFID 접촉하여 아래의 그림과 같이 테스트 해봅니다. 버전은 v1.0 또는 v2.0으로 나올 수 있습니다.

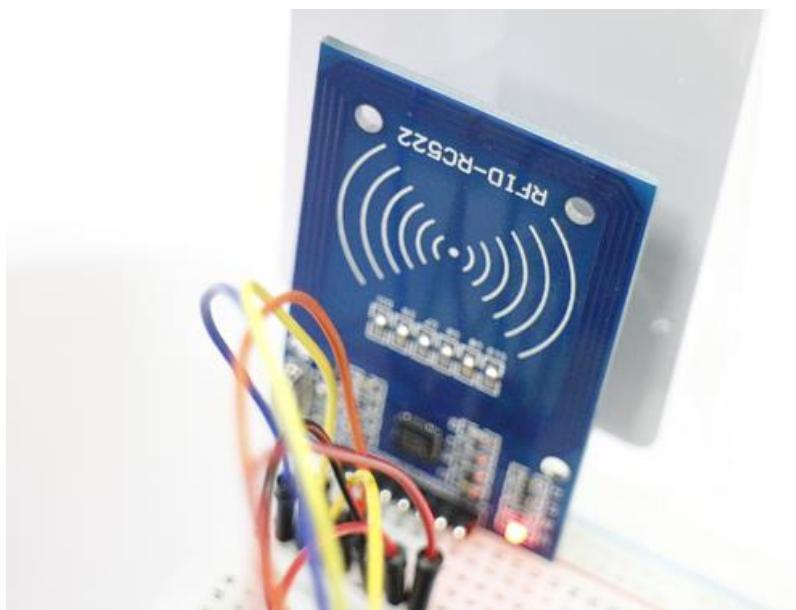


그림 22-10 카드 근거리 접근

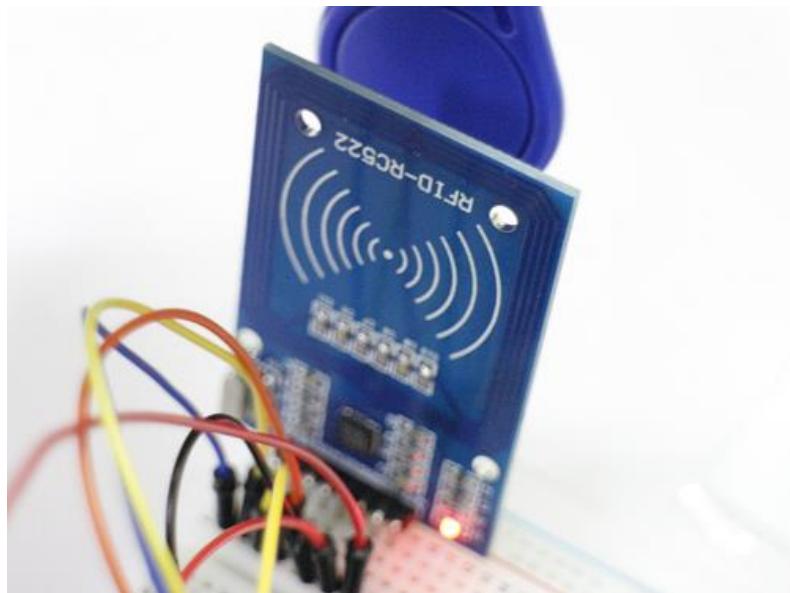


그림 22-11 열쇠 고리 형태 접근

```
MFRC522 Software Version: 0x92 = v2.0
Scan PICC to see UID, type, and data blocks...
Card UID: 94 D5 DF 5F
PICC type: MIFARE 1KB

Sector Block 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 AccessBits
15   63  00 00 00 00  00 00 FF 07  80 BC FF FF  FF FF FF FF [ 0 0 1 ]
      62  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00 [ 0 0 0 ]
      61  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00 [ 0 0 0 ]
      60  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00 [ 0 0 0 ]
14   59  00 00 00 00  00 00 FF 07  80 69 FF FF  FF FF FF FF [ 0 0 1 ]
      58  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00 [ 0 0 0 ]
      57  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00 [ 0 0 0 ]
      56  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00 [ 0 0 0 ]
13   55  00 00 00 00  00 00 FF 07  80 69 FF FF  FF FF FF FF [ 0 0 1 ]
      54  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00 [ 0 0 0 ]
      53  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00 [ 0 0 0 ]
      52  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00 [ 0 0 0 ]
```

그림 22-12 시리얼 모니터 로그 화면

DumpInfo 예제는 RFID TAG IC의 모든 섹터 정보를 가져오면서 시리얼 모니터 창에 프린트 되는 시간 때문에 2초정도 TAG를 모듈 가까이 유지해야 합니다.

RC522 모듈의 버전은 V1.0 또는 V2.0 표시가 됩니다. 버전의 차이점은 TAG 종류의 호환성이므로 큰 의미는 없습니다. V1.0으로 표시되는 경우 호환되는 RFID TAG의 종류가 더 많습니다.

## 23 > LCD1602 I2C + IIC INTERFACE MODULE X 1

LCD 1602 모듈과 I2C 인터페이스 모듈입니다.

VCC 는 5V 입니다. 또는 3V3 전원 연결도 작동 되리라 봅니다.

LCD 1602 (16 x 2) 16 행 2 열의 문자 표시 모듈입니다.



그림 23-1 LCD1602 모듈

LCD1602 16 핀 I2C 4 핀 인터페이스 모듈입니다.



그림 23-2 LCD1602 I2C 변환 모듈

위의 2 가지 모듈을 아래의 이미지처럼 연결해서 사용합니다.  
(현재 키트에 포함된 LCD1602 모듈은 미리 납땜으로 합체된 모듈입니다)

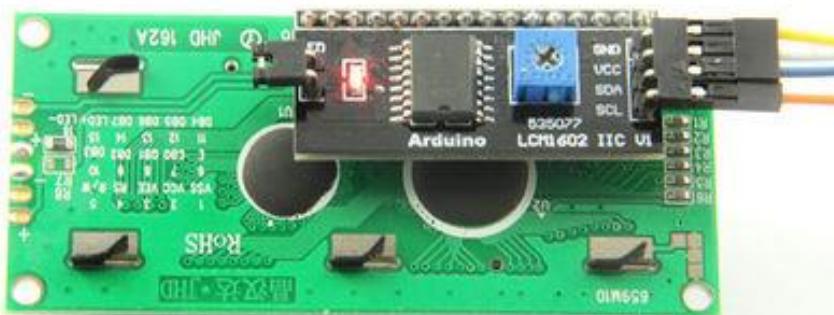


그림 23-3 LCD1602 모듈에 I2C 변환 모듈 장착된 모습

아두이노 우노 보드의 I2C 포트에 연결합니다.

I2C Address: 0x27

Pin Definition: VCC, GND, SDA, SCL

| 1602 i2c 모듈 | 아두이노 우노 |
|-------------|---------|
| VCC         | 5V      |
| GND         | GND     |
| SDA         | A4      |
| SCL         | A5      |

I2C 연결은 아두이노 우노 보드의 A4, A5 에 연결하도록 합니다.  
또는 아두이노 우노 보드의 AREF 옆의 2 개의 포트에 연결하면 됩니다.

## 23.1 I2C LCD1602 예제

I2C LCD1602 DataSheet

<http://www.igameplus.com/pds/gpshop/arduino/pdf/LCD1602-I2C.pdf>

라이브러리를 다운로드 후 스케치 myLibrary 디렉터리에 적절히 복사 해 줍니다.

[http://www.gameplusedu.com/pds/gpshop/arduino/shield\\_module/LiquidCrystal\\_I2Cv1-1.rar](http://www.gameplusedu.com/pds/gpshop/arduino/shield_module/LiquidCrystal_I2Cv1-1.rar)

I2C LCD1602 라이브러리 & 예제 코드 첫 번째

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
// set the LCD address to 0x27 for a 16 chars and 2 line display
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup()
{
    lcd.init(); // initialize the lcd
    // Print a message to the LCD.
    lcd.backlight();
    lcd.print("Hello, world!");
}

void loop()
{}
```

## 23.2 시리얼로 받은 문자열을 LCD1602 I2C 모듈로 출력

아래의 예제 코드를 업로드 합니다.

코드의 내용은 시리얼로 입력 받은 문자열을 LCD1602 I2C 인터페이스 모듈로 출력하는 코드입니다.

예제 코드를 테스트하기 위해서는 업로드 후 스케치 IDE 의 시리얼 모니터 창을 열고 임의의 문자/숫자 들을 입력해 봅니다.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2); // set the LCD address to 0x27 for a 16
chars and 2 line display

void setup()
{
    lcd.init(); // initialize the lcd
    lcd.backlight();
    Serial.begin(9600);
```

```

}

void loop()
{
    // when characters arrive over the serial port...
    if (Serial.available()) {
        // wait a bit for the entire message to arrive
        delay(100);
        // clear the screen
        lcd.clear();
        // read all the available characters
        while (Serial.available() > 0) {
            // display each character to the LCD
            lcd.write(Serial.read());
        }
    }
}

```

위의 코드 적용 시 깨진 문자처럼 나옵니다.

lcd.write() 함수는 BYTE 값 출력이라 의도하지 않게 깨진 문자로 보입니다.

아래의 예제코드는 시리얼로 받은 BYTE 값을 프린트 가능하게 변환하는 부분 추가된 코드입니다.

아래의 예제 코드와 위의 예제 코드를 비교해 보시기 바랍니다.

시리얼로 입력된 전체 문자열들을 LCD1602로 출력해주는 코드입니다.

```

#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2); // set the LCD address to 0x27 for a 16
chars and 2 line display

void setup()
{
    lcd.init();           // initialize the lcd
    lcd.backlight();
    Serial.begin(9600);
}

void loop()
{
    // when characters arrive over the serial port...
}

```

```
if (Serial.available()) {  
    // wait a bit for the entire message to arrive  
    delay(100);  
  
    // clear the screen  
    lcd.clear();  
  
    String szTemp;  
    // read all the available characters  
    while (Serial.available() > 0) {  
        // display each character to the LCD  
        char cRead=Serial.read();  
        szTemp+=cRead;  
    }  
    if( szTemp )  
    {  
        lcd.print(szTemp);  
    }  
}  
}
```

## 24 LCD1602 디렉트 와이어링

LCD1602 모듈을 위의 I2C 변환 모듈 보드를 사용하지 않고 LCD1602 본체에 바로 연결을 해서도 사용할 수 있습니다. 다만, 연결 핀 수가 많아 지는 단점은 있지만, 다른 용도로의 사용시 용이하게 적용 할 수 있습니다. LCD1602 본 모듈만을 사용하기 위해서는 별도로 모듈을 준비하도록 합니다.

| LCD1602 모듈 | 아두이노 우노          |
|------------|------------------|
| VSS        | GND              |
| VDD        | 5V               |
| V0         | 가변 저항 10K OUT 연결 |
| RS         | 12               |
| RW         | 11               |
| E          | 10               |
| D0         | NC               |
| D1         | NC               |
| D2         | NC               |
| D3         | NC               |
| D4         | 5                |
| D5         | 4                |
| D6         | 3                |
| D7         | 2                |
| A          | 5V               |
| K          | GND              |

LCD1602 직접 연결 사용시에는 별도의 라이브러리 설치 없이 사용 가능합니다.  
아두이노 기본 라이브러리에 포함되어 있습니다.

디렉터리 위치는 “arduino-1.6.4\libraries\LiquidCrystal\src”

아래의 코드도 아두이노 라이브러리 LiquidCrystal 사용합니다.

별도의 라이브러리 설치 없이 사용 가능합니다.

### 24.1 가변저항으로 대비(CONTRAST) 조절 방법

10K 가변 저항으로 밝기 조절을 합니다. 가변 저항은 우노 보드의 13번 또는 원하는 핀에 연결 후 코드에 적용 합니다. 가변저항에 입력되는 전원은 D13 번 포트에서 받아서 사용됩니다. 또는 필요에 따라 우노의 5V 출력에서 나오는 전원에 가변저항을 연결하여 사용할 수도 있습니다.

LCD 모듈의 V0 과 우노 보드의 13 번 핀에 가변 저항 연결의 이해는 가변 저항으로 LED 밝기 조절하는 개념과 동일합니다.

13 번핀에 5V (HIGH) 출력->가변저항에서 5V 받아서 전압조절로 밝기 조절됩니다.

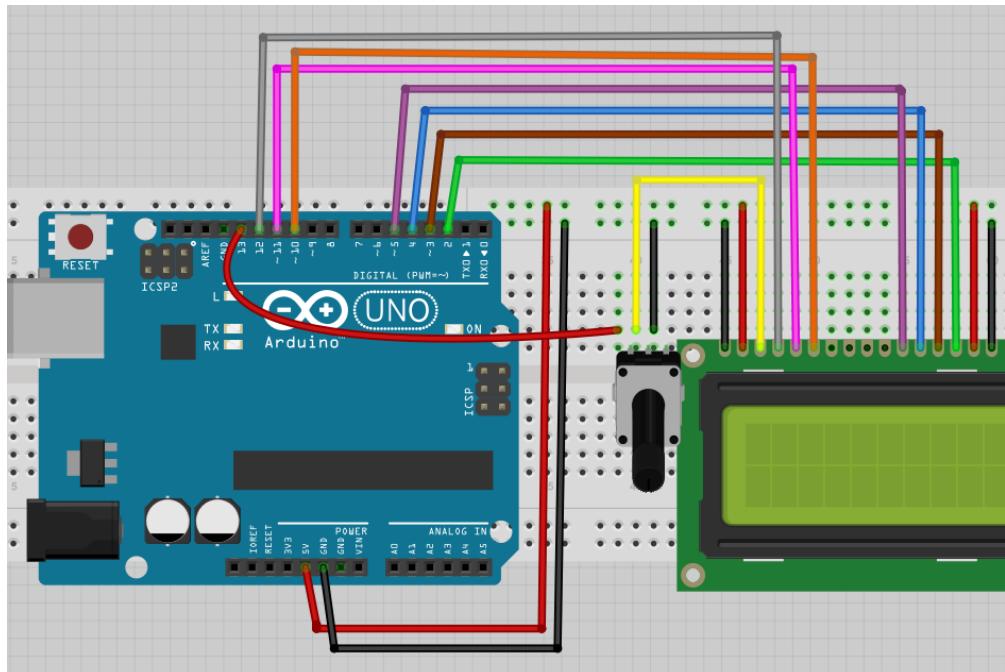


그림 24-1 LCD, 가변저항 구성 브레드보드 회로도

## 24.2 PWM 포트로 대비 조절방법

PWM 포트와 LCD1602 모듈의 V0 포트와 연결하여 PWM 값을 조절하여 대비 (Contrast) 설정 가능합니다.

| LCD1602 모듈 | 아두이노 우노 |
|------------|---------|
| V0         | PWM 포트  |

D6 번 포트와 LCD1602 의 V0 포트를 연결하고 아래와 같이 setup() 함수에서 포트 방향 설정 후 analogWrite() 함수를 사용하여 조절을 하면 됩니다.

```
#define BACK_LIGHT 6 // 포트 지정

void setup()
{
    pinMode(BACK_LIGHT, OUTPUT);
```

```
    analogWrite(BACK_LIGHT, 60); // 적절하게 값을 조절하여 업로드합니다.  
}
```

## 24.3 LCD1602 HELLO 표시 예제 코드

[http://www.gameplusedu.com/pds/gpshop/arduino/kit/a\\_example\\_kit\\_strongest/lcd1602\\_direct\\_hello.ino](http://www.gameplusedu.com/pds/gpshop/arduino/kit/a_example_kit_strongest/lcd1602_direct_hello.ino)

```
#include <LiquidCrystal.h>  
  
// Connections:  
// RS (LCD pin 4) to Arduino pin 12  
// RW (LCD pin 5) to Arduino pin 11  
// Enable (LCD pin 6) to Arduino pin 10  
// LCD pin 15 to Arduino pin 13  
// LCD pins d0,d1,d2,d3 사용 안 함.  
// LCD pins d4, d5, d6, d7 to Arduino pins 5, 4, 3, 2  
//  
// 아두이노 13 번 핀에 10 K 가변 저항을 연결합니다.  
const int BACK_LIGHT = 13; // Pin 13 will control the backlight  
  
LiquidCrystal g_lcd(12, 11, 10, 5, 4, 3, 2);  
  
void setup()  
{  
    pinMode(BACK_LIGHT, OUTPUT);  
    digitalWrite(BACK_LIGHT, HIGH);      // Turn backlight on.  
                                         // Replace 'HIGH' with 'LOW' to turn it off.  
    g_lcd.clear();                      // Start with a blank screen  
    g_lcd.setCursor(0, 0);              // Set the cursor to the beginning  
    g_lcd.print("Hello,");  
    g_lcd.setCursor(0, 1);              // Set the cursor to the next row  
    g_lcd.print("Have a nice day");  
}  
void loop()  
{  
}
```

## 25 > OPTO-COUPLE 1 CHANNEL 5V RELAY X 1

5V 1 채널 릴레이 모듈 입니다.



그림 25-1 1 채널 5V 릴레이 모듈

5V 신호로 AC 전원 연결 1 개를 온/오프 합니다.  
릴레이 작동 시 약간의 톡, 톡 소음이 발생 되는 건 정상입니다.

모듈의 기능은 아래와 같습니다.

Dimensions: 19mm x 15.5mm x 20mm

Coil Voltage: 5V DC

Operate Voltage: 3.75V DC

Release Voltage: 0.5V DC

Coil resistance: 70Ω (+/- 7Ω)

Rated Coil Power: 0.36W (48VDC 0.51W)

Insulation Resistance:  $\geq 100M\Omega$

Dielectric Strength: Between coil and contacts  $\geq 1500VAC$  1min, between open contacts  $\geq 750VAC$  1min

Operate/Release Time:  $\leq 10ms/5ms$

Terminal Type: PCB mounting

Contact Form: 1H/1Z

Contact resistance:  $\leq 100m\Omega$  (1A 6VDC)

Contact Material: AgCdO, AgSnO<sub>2</sub>

Contact Capacity: 5A 240VAC/30VDC, 10A 240VAC/28VDC



그림 25-2 연결 포트 위치

NO - Normal Open 입니다. 평상시에는 비 접속(오픈) 상태입니다.

NC - Normal Close(d) 입니다. 평상시에 접속된 상태입니다.

COM - AC 전원 케이블 2 개중 1 선을 연결하고 NO (or NC) 에 연결합니다.

그림 23-2에서의 오른쪽 헤더 핀은 아두이노 보드에서 연결 됩니다.

아두이노 연결:

| 릴레이 모듈 | 아두이노      |
|--------|-----------|
| VCC    | 5V        |
| IN1    | 디지털 가능 포트 |
| GND    | GND       |

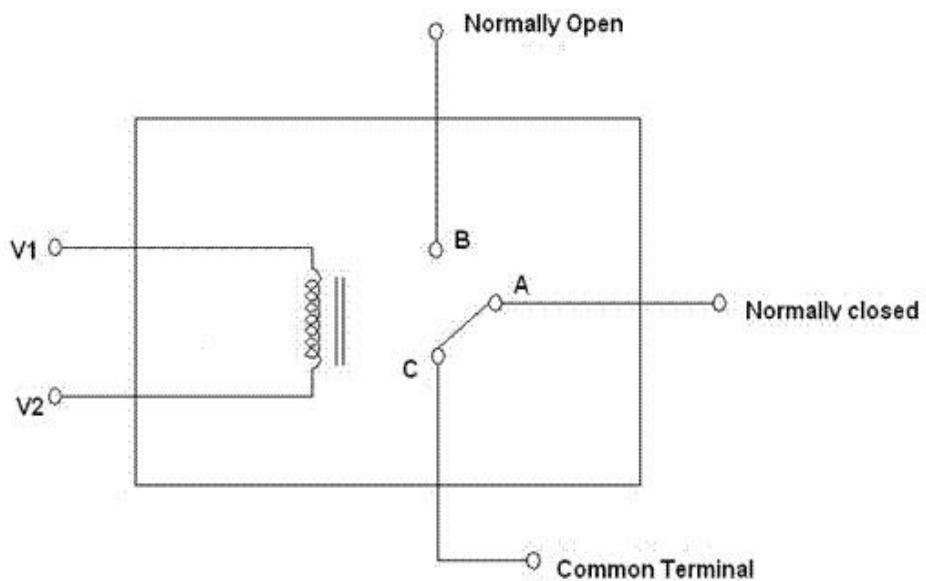


그림 25-3 릴레이 회로도

**5V-Relay 1 channel DataSheet:**

<http://www.gameplusedu.com/pds/gpshop/arduino/pdf/5v-relay-JQC-3F-T73.pdf>

OR

<http://www.gameplusedu.com/pds/gpshop/arduino/pdf/pcb-relay-T73.pdf>

## 25.1 AC 전원선과 릴레이 모듈 연결 방법

먼저 AC 전원 연결 플러그를 빼서 안전하게 합니다.

사용할 AC 전원 연결선 커트합니다. 아래의 그림처럼 분리 합니다.

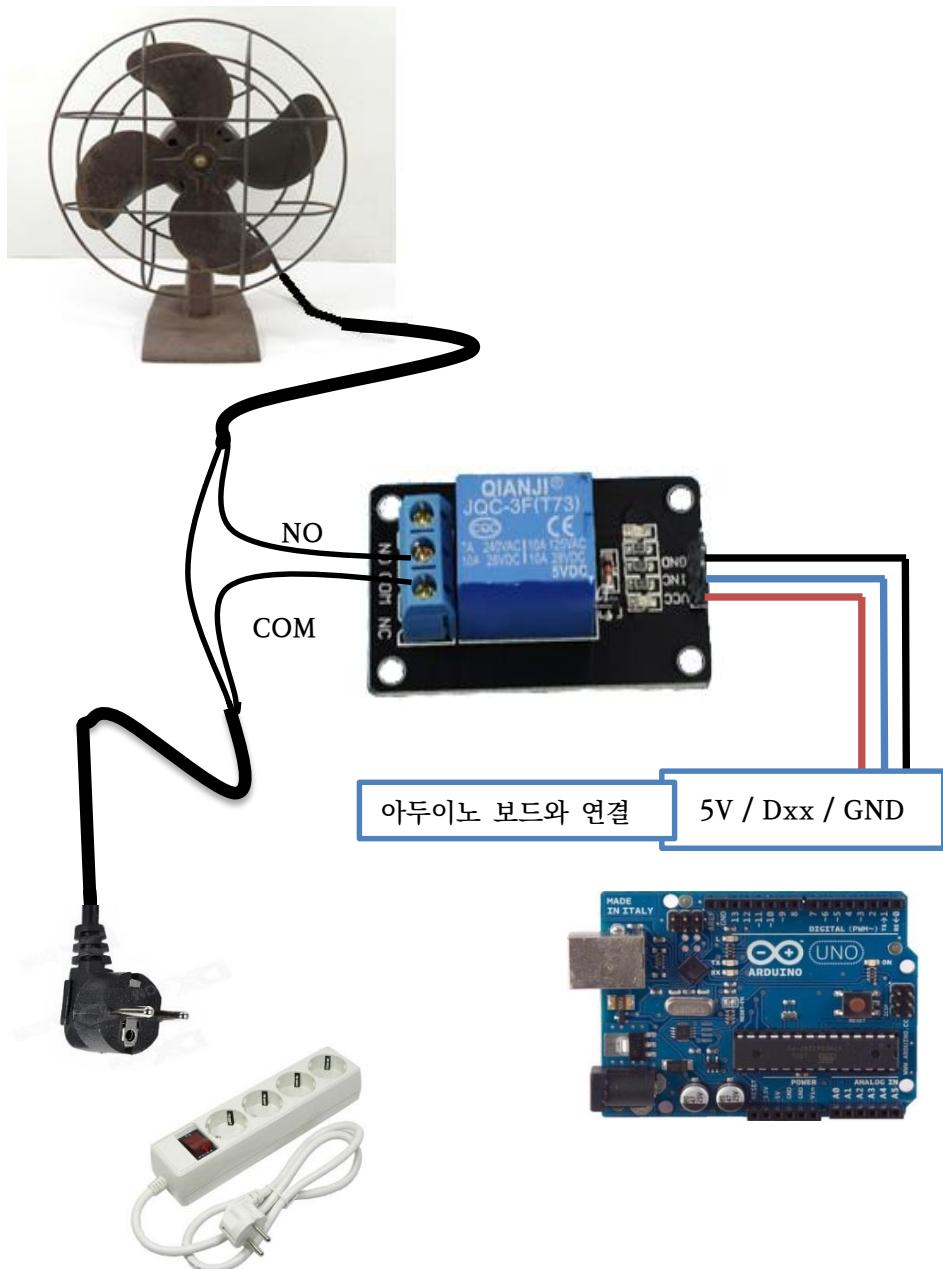


그림 25-4 전선 연결 방법

연결된 모습입니다. 테스트 용도일 경우 AC 220V 전원선을 연결하지 않고 낮은 전압의 LED 연결도 괜찮습니다.



선풍기 5V 릴레이 장착 아두이노 보드와 연결 참조 그림



## 25.2. 5V 릴레이 테스트 예제

5V 릴레이 테스트 하기 위해 스케치 프로그램의 기본 버튼 예제를 이용합니다.  
버튼 예제는 10K 저항이 필요합니다. 아래의 이미지처럼 와이어링 합니다.  
버튼을 누르면 LED ON, 안 눌린 버튼 상태이면 LED OFF 상태를 정확히 확인 및 테스트 합니다.

아두이노 연결:

| 왼쪽 AC 전원 연결 | 오른쪽 헤더 핀 | 아두이노 보드 |
|-------------|----------|---------|
| NO          | GNC      | GND     |
| COM         | IN1      | 디지털 포트  |
| NC          | VCC      | 5V 연결   |

릴레이 모듈의 5V & GND 아두이노 보드와 연결합니다.  
릴레이 IN 포트와 아두이노 보드 13 번 포트와 연결합니다.  
13 번 포트와 릴레이 IN 포트 연결하는 이유는 HIGH, LOW 신호를 보기 위한 용도입니다. 원하는 다른 디지털 포트 사용하여도 무방합니다.

릴레이 모듈 보드의 초록색(모듈에 따라 색상이 다른 색상일 경우도 있음)은 모듈 가동 상태입니다. 가동 중이면 ON입니다.  
릴레이 모듈 보드의 빨강색 LED 는 모듈에 따라 NC/NO 에 물려진 표시입니다.  
HIGH or LOW 일 경우 ON 됩니다.

»> 1 채널 이상의 릴레이 사용하기

1 채널 5V 릴레이 사용을 이해하는 경우 2 채널, 4 채널, 8 채널 릴레이도 동일하게 사용됩니다.

모듈 자체에 제어포트와 전원연결부만 추가되는 형식입니다.

## 25.2.1 릴레이 테스트 코드 – 5 초마다 켜고, 끄기

5 초 반복으로 릴레이 켜고 끄기 테스트 코드입니다.

아두이노 디지털포트 13 번에 릴레이 모듈의 in1 or INC 포트와 연결합니다.

코드의 내용은 아두이노 보드의 디지털포트 13 번에 HIGH, LOW 신호 5 초마다 반복합니다.

예제코드:

```
#define RELAY_ON 0
#define RELAY_OFF 1

#define Relay_1 10 // Digital Pin D 10

void setup()
{
    Serial.begin(9600); // 시리얼 포트로 메시지 보기 위해
    digitalWrite(Relay_1, RELAY_OFF);

    pinMode(Relay_1, OUTPUT);
    delay(5000); // 부팅시 5 초 지연.
}

void loop()
{
    // 릴레이 켜기 신호.
    digitalWrite(Relay_1, RELAY_ON);
    Serial.println("Relay On");
    delay(5000); // wait for 5 second

    // 릴레이 끄기 신호.
    digitalWrite(Relay_1, RELAY_OFF);
    Serial.println("Relay Off");
    delay(5000);
}
```

## 25.2.2 버튼 누르면 LED 켜지게 하기

버튼을 누르면 아두이노 보드 10 번 포트의 LED On 됩니다.

릴레이 모듈 보드의 초록색 LED 는 Off 됩니다.

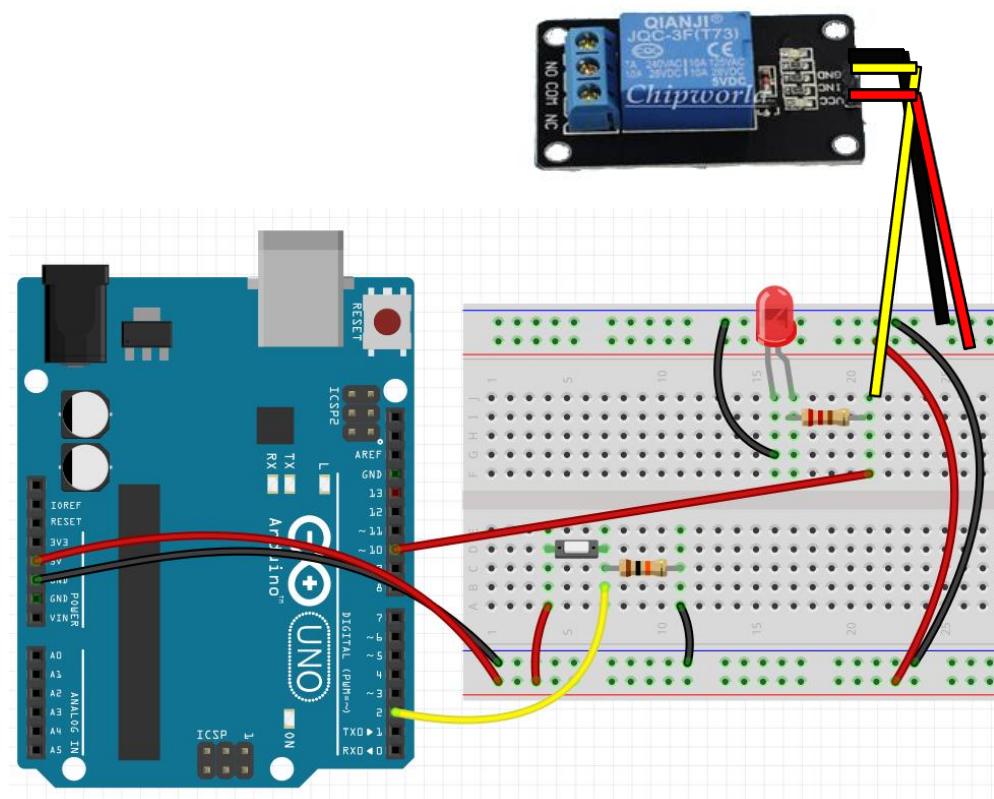
버튼 포트는 PullDown 설정입니다.

AC 전원 220V 를 사용하는 만큼 정확한 사용법 숙지 후 안전하게 테스트 하시기 바랍니다.

5V 릴레이의 VCC → 우노 보드의 5V 와 연결

5V 릴레이의 GND → 우노 보드의 GND 와 연결

5V 릴레이의 IN1 → 우노 보드의 D10 와 연결.



위의 버튼 예제 브레드보드에 릴레이 연결된 브레드보드 회로 구성 이미지입니다.

버튼 예제 코드입니다.

```
// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 2;      // the number of the pushbutton pin
const int ledPin = 10;        // the number of the LED pin

// variables will change:
int buttonState = 0;          // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop(){
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  //

  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  }
  else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

먼저 위의 버튼 테스트 코드 및 회로도 구성 완성 후 5V 릴레이 모듈과 와이어링을 합니다.

5V 릴레이 모듈의 GND, VCC 와 아두이노 보드의 GND, 5V 연결 합니다.  
아두이노 보드의 13 번 핀과 릴레이 모듈의 IN1 과 연결 합니다.

#### 릴레이 모듈 테스트 시 주의사항:

아두이노에 릴레이 GND, VCC 연결 후 제어포트로 D13 번 포트는 가급적 사용 하지 않도록 합니다. D13 번 포트는 아두이노 보드의 리셋, 파워 등의 표시(Indicator)역할도 있습니다. 코드 업로드 시, 또는 리셋 버튼을 누를 시 깜박거리게 되어 있습니다. D13 에 제어되는 포트의 신호 레벨이 연결된 모듈에게 그대로 반영되게 되어 있습니다. 릴레이, 또는 다른 전기적인 제어 모듈의 제어포트로는 D13 번 사용하지 않기 권장 드립니다.

## 25.3 릴레이 활용

조도센서, 온도센서, 인체감지 센서 모듈과 연동하여 다양한 용도로의 사용 및 테스트가 가능합니다. A0~A5 에는 주로 아날로그 센서모듈들이 많이 사용됩니다.

인체감지 센서를 이용하여 자동으로 등 켜고/끄기 등은 쉽게 구현하실 수 있습니다.  
조도센서를 이용하는 경우에는 어두울 경우 스위치를 켜고, 반대일 경우 스위치를 닫기 할 수 있습니다. 릴레이의 사용용도는 대부분 전자기기를 켜고 끄는 기능을 하게 됩니다.

응용하기에 따라 많은 용도로 활용할 수 있습니다.

## 26 > DS1302 CLOCK MODULE X 1



그림 26-1 RTC1302 모듈

RTC 모듈입니다. Real Time Clock 모듈입니다.

시간 저장 및 저장된 시간 가져오기를 할 수 있습니다.

작동 전압 3V~5V 모듈입니다.

DS1302 RTC 모듈에는 외부 전원 공급이 있는 상태에서 시간 저장이 됩니다. RTC 모듈들의 기본역할은 전원 공급이 없어도 시간을 저장해야 한다는 겁니다. DS1302 모듈과 같이 배터리 내장 모듈들은 포장시에 배터리 방지를 위해 배터리가 뒷면으로 되어 있는 경우 또는 분리되어 있을 수 있습니다. 사용시 다시 앞면으로 해주시거나 위의 형태로 사용합니다.

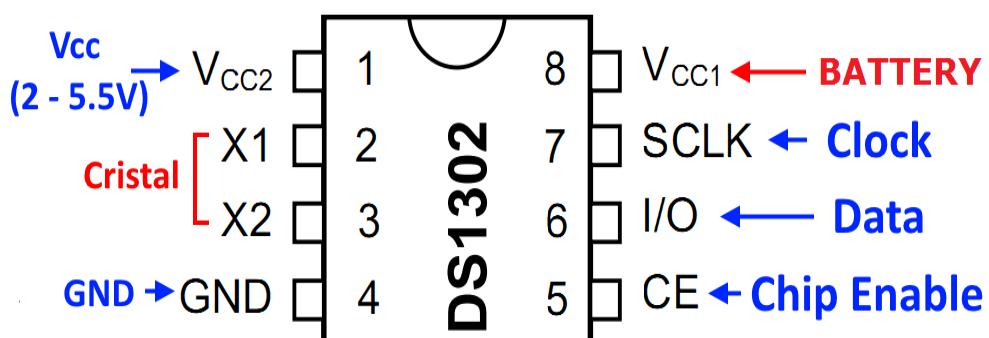


그림 26-1 DS1302 펈맵

DS1302 IC 부품만을 사용하여 브레드보드, 만능기판에 제작하는 경우 X1,X2 포트에 진동소자(크리스탈)가 연결되어야 합니다. Cristal 부품은 32768 KHz 사용합니다.

아두이노 연결:

| DS1302 모듈 | 아두이노 |
|-----------|------|
| VCC       | 5V   |
| GND       | GND  |
| CLK       | D4   |
| DAT       | D3   |
| RST       | D2   |

아두이노 예제 코드 및 설명

<http://playground.arduino.cc/Main/DS1302> (모듈 형태만 다릅니다. 사용방법은 동일합니다. )

DS1302 DataSheet:

[http://www.gameplusedu.com/pds/gpshop/arduino/pdf/DS1302\\_to\\_DS1302ZN.pdf](http://www.gameplusedu.com/pds/gpshop/arduino/pdf/DS1302_to_DS1302ZN.pdf)

위의 아두이노 사이트의 예제가 복잡할 경우 아래의 주소에서 다운로드 받아서 사용하시면 됩니다.

[http://www.gameplusedu.com/pds/gpshop/arduino/shield\\_module/RTC-DS1302-A-TYPE/DS1302.zip](http://www.gameplusedu.com/pds/gpshop/arduino/shield_module/RTC-DS1302-A-TYPE/DS1302.zip)

DS1302.zip 을 다운로드 후 압축 해제 합니다.

아두이노 스케치 IDE 의 myLibrary 디렉터리로 적절하게 복사 합니다.

또는 스케치 IDE 메뉴 -> 스케치 -> 라이브러리 가져오기 -> “라이브러리 추가” 선택 합니다. 파일 선택 압축파일(ZIP), 파일 또는 폴더를 선택하면 자동으로 스케치 라이브러리에 추가 됩니다.

아래의 예제 코드를 적용 후 테스트 해 봅니다.

코드의 내용은 1 초마다 RTC DS1302 모듈에서 가져온 시간을 시리얼 포트로 출력 합니다.

```
/* Define the DIO pins used for the RTC module */
#define SCK_PIN 4
#define IO_PIN 3
#define RST_PIN 2

/* Include the DS1302 library */
#include <DS1302.h>
```

```

/* Initialise the DS1302 library */
DS1302 rtc(RST_PIN, IO_PIN, SCK_PIN);

void setup()
{
    /* Clear the 1302's halt flag */
    rtc.halt(false);
    /* And disable write protection */
    rtc.writeProtect(false);

    /* Initialise the serial port */
    Serial.begin(9600);
}

/* Main program */
void loop()
{
    /* Set the time and date to 16:30 on the 3rd of September 2013 */
    rtc.setDOW(MONDAY); // 월요일로 설정합니다.
    rtc.setTime(16,30,0); // 24 시간 형태로 16 시 30 분 0 초 설정.
    rtc.setDate(3, 9, 2015); // 2015년 9월 3일로 지정합니다.

    while(1) // 무한루프
    {
        Serial.print("It is ");
        Serial.print(rtc.getDOWStr());
        Serial.print(" ");
        Serial.print(rtc.getDateStr());
        Serial.print(" ");
        Serial.print("and the time is: ");
        Serial.println(rtc.getTimeStr());
        /* Wait before reading again */
        delay (1000);
    }
}

```

## 27 › SOUND DETECTION MODULE X 1

LM393 Sound Detection Sensor Module  
사운드 감지 센서 모듈입니다.

### 27.1 사운드 센서 모듈 소개

키트의 구성에 따라 아래와 같은 A, B 중에 1 개 포함 되어 있습니다.

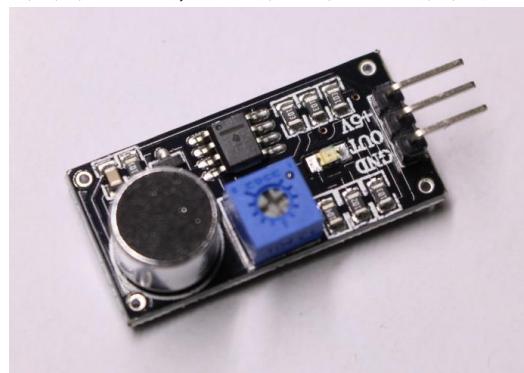


그림 27-1 A 모듈 3 포트

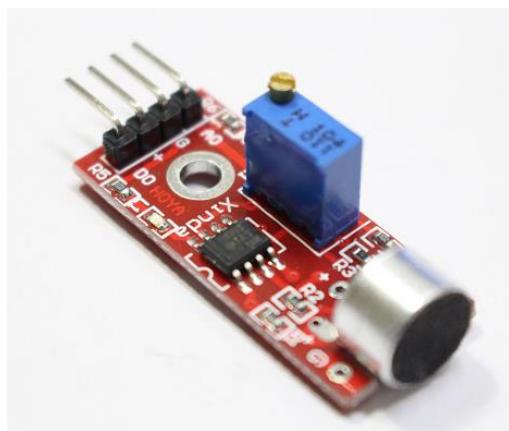


그림 27-2 B 모듈 4 포트 – 디지털 포트 추가된 형태)

모양은 틀리지만 소리 센서 기능은 동일합니다.

사용 되는 IC 칩은 LM393 입니다. LM393 IC 는 비교기(Comparator)입니다. 2 가지의 신호를 비교하는 용도로 사용하고 있습니다. 원하는 수치의 입력 전압을 비교하여 해당 출력 포트로 전압을 출력합니다. 사운드 센서를 비롯하여 각종 센서 모듈에서 가장 많이 사용되고 있는 IC 입니다. 사운드 센서 모듈에서의 LM393 IC 역할은 MIC 으로부터의 신호 전압을 비교하여 일정 수치 이상인 경우 해당 포트로 신호를 출력합니다.

데이터시트 LM393

<http://www.gameplusedu.com/pds/gpshop/sensor/lm393-n.pdf>

Chip: LM393, Electric condenser microphone

Operation voltage: DC 4-6V

1-Way signal output, low level output

사운드 센서 모듈은 보통 2 가지 형태가 있습니다. 3 핀과 4 핀 모듈입니다.

3 핀 (GND, VCC, 아날로그 출력)

4 핀(GND, VCC, 아날로그 출력, 디지털 출력)

본 모듈의 연결 헤더 핀은 일반적으로 3 개입니다. 아날로그 값만 가져와서 사용하는 경우가 대부분입니다. 물론 디지털 포트 (0, 1) 추가된 4 개의 포트 모듈도 있습니다.

| 사운드 모듈    | 아두이노                 |
|-----------|----------------------|
| GND       | GND                  |
| OUT or A0 | A0                   |
| VCC       | 5V                   |
| D0        | D2 or Any Input Port |

연결 방법은 위의 핀맵을 참조하여 연결하도록 합니다.

GND      우노 보드의 GND 와 연결합니다.

OUT      아두이노의 아날로그 포트에 (A0) 연결합니다.

VCC      아두이노의 5V 에 연결합니다.

D0      아두이노의 디지털 포트 연결합니다.

사운드 센서 모듈은 기준값을 맞추고 소리의 반응값을 확인 후 사용해야 합니다.

## 27.2 기준값 설정

기준값 확인 및 지정은 사운드 센서 모듈에 가변 저항 조절 부분을 돌려서 맞추어야 합니다.

아래와 같은 코드를 작성 & 업로드 후 사운드 센서 아날로그 값을 출력 해봅니다. 아날로그 값을 출력하는 간단한 코드 작성 후, 스케치의 시리얼 모니터 창 열기 후 시리얼 출력되는 텍스트로 연속해서 읽어 들인 값을 볼 수 있습니다.

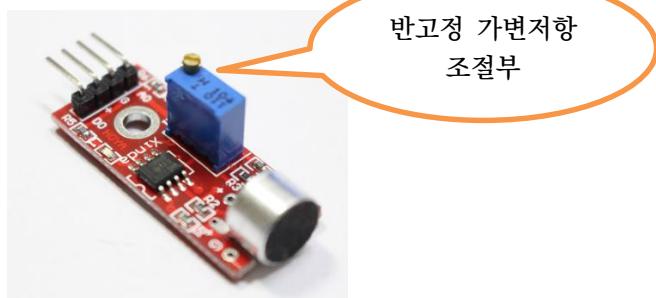
```
// 사운드 모듈 아날로그 출력 기준 가변저항 값 확인 코드.  
// for Analog input from sound breakout board.  
int soundSensorPinA=A0;  
  
void setup()  
{  
    Serial.begin(9600);  
}  
void loop()  
{  
    int sndValue = analogRead(soundSensorPinA);  
    Serial.println(sndValue); // 아날로그 입력 값 출력.  
    delay(20); //  
}
```

» A 모듈



A 모듈의 가변저항 조절부는 최소/최대 회전 지정된 형태입니다.  
왼쪽으로 최대 회전 후 오른쪽으로 돌려가면 값을 조절하도록 합니다.  
34~35 정도의 값이 보이도록 조절합니다.

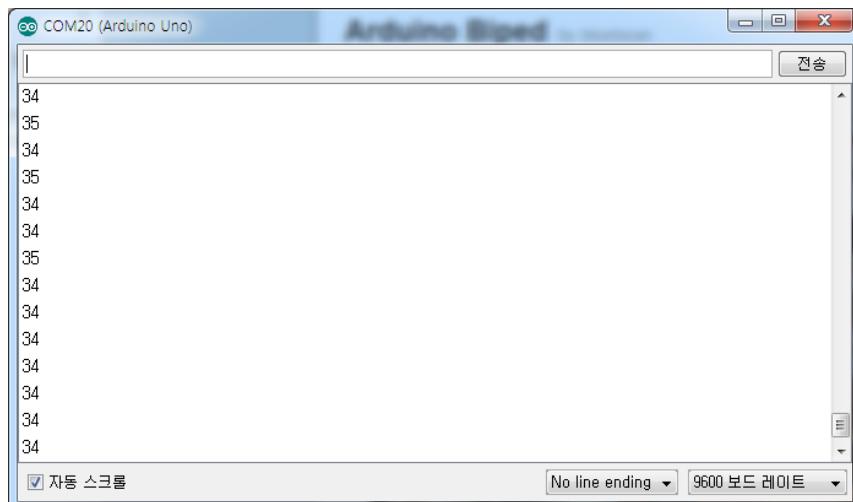
» B 모듈



B 모듈의 가변저항 조절부는 최소/최대 회전이 없는 연속 회전 부품입니다. 값 조절될 때까지 돌려주면 됩니다. 490 정도 내외로 맞추도록 합니다. 또는 필요에 따라 원하는 값으로 지정해 주면 됩니다.

» A 사운드 모듈 가변저항을 조절하여 아래와 비슷한 값이 출력되면 정상입니다.

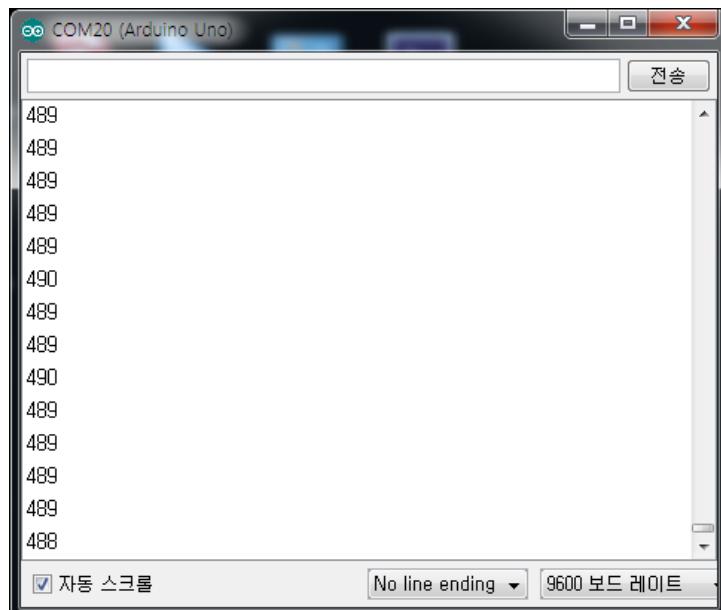
34~



The screenshot shows the Arduino Serial Monitor window titled "COM20 (Arduino Uno)" with the subtitle "Arduino Sived". The main text area displays the value "34" repeated approximately 15 times. The bottom status bar shows "자동 스크롤" (Auto Scroll) is checked, and the baud rate is set to "9600 보드 레이트" (9600 Board Rate).

그림 27-3 A 사운드 센서 출력 값

» B 사운드 모듈 가변저항을 조절하여 아래와 비슷한 값이 출력되면 정상입니다.  
490~ 값으로 조절하면 됩니다.



The screenshot shows the Arduino Serial Monitor window titled "COM20 (Arduino Uno)". The main text area displays a series of values starting at 489 and ending at 488, with one instance of 490. The bottom status bar shows "자동 스크롤" (Auto Scroll) is checked, and the baud rate is set to "9600 보드 레이트" (9600 Board Rate).

그림 27-4 B 사운드 센서 출력 값

시리얼 모니터 창으로의 출력 값을 보면서 가변 저항 조절부를 왼쪽/오른쪽으로 돌려보면서 기본 측정값을 지정 합니다.

위의 A 모듈은 기본 값을 34~ 범위 정도 맞춰 봅니다.

위의 B 모듈은 기본 값을 500~ 범위 정도 맞춰 봅니다.

소리를 내어서 값을 확인하여, 용도에 맞게 최소, 최대 범위의 적절한 값을 찾아야 합니다.

키트에 포함된 사운드 센서는 2가지 모듈 이외에도 여러 사운드 모듈이 사용될 수 있습니다. 각각의 PCB 구성에 따라 소리 반응에 대한 기본 감도가 약간씩 다릅니다.

위의 코드를 기초로 아래의 코드처럼 몇 줄 추가해봅니다.

그럼 사운드 모듈의 성격 및 방법이 이해되리라 보여집니다.

### 27.3 소리 반응 기초 프로그래밍하기

대략적인 기본 수치 값을 확인 후 센싱 프로그래밍에서 필요한 수치를 기준으로 간단히 아래와 같이 코드를 작성하여 소리 반응을 프로그래밍 할 수 있습니다.

사운드 모듈에서의 아날로그 입력값이 500 넘었을 경우에만 메시지를 출력해주는 간단한 코드입니다.

사운드 모듈의 아날로그 출력 범위 모니터링

(예제 코드: st\_8\_3) A, B 모듈 공통 테스트 코드입니다.

```
// 사운드 모듈 아날로그 출력 모니터링 확인 코드.  
// for Analog input from sound breakout board.  
int soundSensorPinA=A0;  
int soundSensorPinD=2;  
  
void setup()  
{  
    Serial.begin(9600);  
}  
  
void loop()  
{  
    int sndValue = analogRead(soundSensorPinA);  
    //    Serial.println(sndValue);
```

```

//    delay(20); //
if( sndValue > 500 )
{
    Serial.print("Detected Sound.==> ");
    Serial.println(sndValue);
}
}

```

위의 코드를 실행하면서 시리얼 모니터 창을 열어서 값을 확인합니다.  
사운드 센서의 목적 및 용도에 따라 회로 구성이 다양하게 나와 있으므로, 차후에 용도에 맞는 사운드 센서를 선택하는데 도움이 되리라 봅니다.

## 27.4 소리 반응 LED 켜기와 끄기

사운드 센서에서 받은 값을 기준으로 일정 값을 보다 클 경우 단순히 LED를 켜고 끄는 코드를 작성해봅니다.

LED는 3,5,6 번 포트에 연결하여 줍니다.

```

// 사운드 모듈 아날로그 출력 모니터링 확인 코드.
// for Analog input from sound breakout board.
int soundSensorPinA=A0;
int soundSensorPinD=2;

int led_pwm_pin[] = {3,5,6};

void setup()
{
    Serial.begin(9600);
    pinMode(led_pwm_pin[0],OUTPUT);
    pinMode(led_pwm_pin[1],OUTPUT);
    pinMode(led_pwm_pin[2],OUTPUT);
}

void loop()
{
    int sndValue = analogRead(soundSensorPinA);
}

```

```

//      Serial.println(sndValue); // 아날로그 입력
값 출력.
//      delay(20); //
if( sndValue > 500 )
{
    Serial.print("Detected Sound.==> ");
    Serial.println(sndValue);

    digitalWrite(led_pwm_pin[0],HIGH);
    digitalWrite(led_pwm_pin[1],HIGH);
    digitalWrite(led_pwm_pin[2],HIGH);
}
else
{
    digitalWrite(led_pwm_pin[0],LOW);
    digitalWrite(led_pwm_pin[1],LOW);
    digitalWrite(led_pwm_pin[2],LOW);
}
}

```

## 27.5 소리 반응 LED 반응 PWM 사용하기

아래의 예제 코드는 사운드 발생시 PWM 포트에 연결된 LED를 순차적으로 켜고, 부드럽게 OFF 하는 예제입니다.

st\_code\_ex\_8)

[http://www.gameplusedu.com/pds/gpshop/kit\\_code/strongest/sound\\_sensor.ino](http://www.gameplusedu.com/pds/gpshop/kit_code/strongest/sound_sensor.ino)

```

//
// LED 는 3,5,6 번에 연결 후 테스트 해야 정확한
// 코드 이해가 됩니다.
// 센서 모듈 <<--->> 아두이노 보드
// A0 <<----->> A0
// GND <<----->> GND
// VCC <<----->> 5V
// D0 <<----->> D2 // 4 핀 센서 모듈//

```

```

// 센서 모듈의 D0 핀 설명
// D0 -- HIGH
// D1 -- LOW (일정 강도의 소리가 전달될 때 LOW )
// 박수 소리 또는 단발성 소리가 나면 LED 가 순차적으로
// 점멸되는 예제입니다.
//
// sensing programming
// sound demo
//
// caution : base soundThreshold adjust 490 or you want.
// 사운드 센서 모듈의 가변저항을 왼쪽으로 계속 돌려서 490 정도로 맞추거나
// 센서 회로 구성에 따라 적절히 맞추어야 합니다.
//

//
int soundSensorPin=A0; // for Analog input from sound breakout board.
int soundSensorDigitalPin=2; // for Digital input from sound breakout
board.

int soundReading=0;
// 기준값을 적절히 수정해서 사용합니다.
int soundThreshold=500; // 3 핀 모듈, 4 핀 모듈
int intensity[3]={0,0,0};
int LEDPins[3] = {3,5,6}; // PWM 포트
int numberOfPins=3;
int currentPin=0;
int fadeCounter=0;
int fadeDelay=50;

boolean switcher = true;

void setup()
{
    Serial.begin(9600);
    pinMode(soundSensorPin, INPUT);

    pinMode(soundSensorDigitalPin, INPUT);

    for(int i=0; i<numberOfPins;i++)
    {

```

```

        pinMode(LED Pins[i], OUTPUT);
    }
}

void loop(){
    // 4 핀 디지털 입력핀 사용시 코드.
    // if(digitalRead(soundSensorDigitalPin)==LOW)
    // {
    //     Serial.println("LOW Digital input");
    // }

    soundReading=analogRead(soundSensorPin);
    if(soundReading>soundThreshold)
    {
        Serial.println(soundReading);
        if(switcher)
        {
            aboveThreshold(currentPin);
            switcher=true;
        }
    }
    else
    {
        if(switcher)
        {
            belowThreshold();
            switcher=true;
        }
    }
}

void aboveThreshold(int cPin)
{
    switcher=false;
    if(intensity[cPin]<10)
    {
        intensity[cPin]=255;
        delay(50);
        currentPin=currentPin+1;
    }
}

```

```

if(currentPin==numberOfPins)
{
    currentPin=0;
}
}

void belowThreshold()
{
    switcher=false;
    fadeCounter++;
    if(fadeCounter==fadeDelay)
    {
        fadeCounter=0;
        for(int i=0; i<numberOfPins;i++)
        {
            analogWrite(LED Pins[i],intensity[i]);
        }
        for(int i=0; i<numberOfPins;i++)
        {
            intensity[i]--;
            if(intensity[i]<0)
            {
                intensity[i]=0;
            }
        }
    }
}

```

아래는 위의 코드 구현된 예시 이미지입니다.

LED 사용시 LED 의 (+) 방향에 저항 220 R 사용하면 됩니다. 저항을 사용하지 않아도 되긴 합니다만, 지속적으로 사용하는 경우에는 반드시 저항을 사용하여 LED 부품을 보호하도록 합니다.

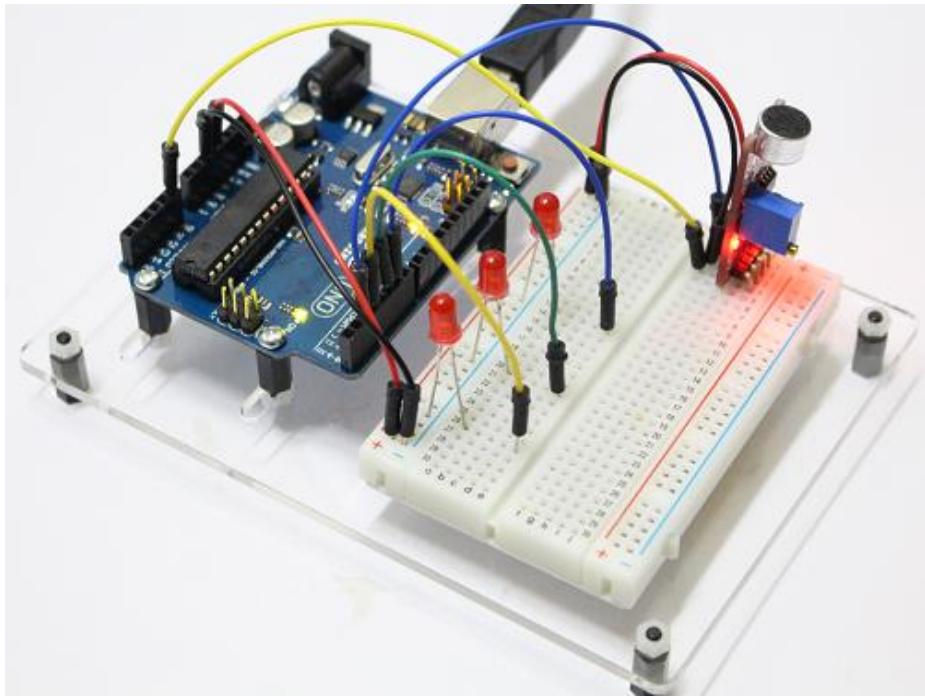


그림 27-5 박수소리 LED 점등 브레드보드 회로

## 28 › TEMPERATURE AND HUMIDITY MODULE X 1

디지털 온/습도 측정 센서입니다.

센서 모듈에서의 디지털 신호를 받아서 온/습도 측정을 할 수 있습니다.

### 28.1 DHT11 온도, 습도 센서 모듈

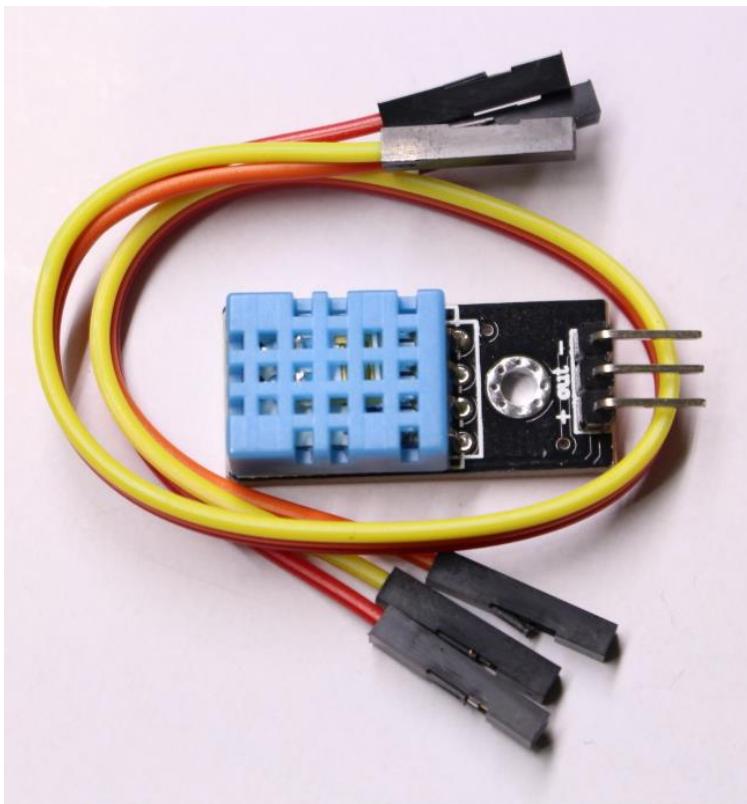


그림 28-1 DHT11 온습도 센서 모듈 & 뒤풋 케이블

온도, 습도 센서입니다. DHT11 온/습도 측정하는 센서 모듈입니다.

온도센서는 형태와 소자만 조금 다를 뿐 너무나 많은 전자기기에 사용되고 있습니다.

산업용/가전용 거의 모든 제품에 위와 비슷한 계열의 온도 센서가 사용 됩니다. 의학/정밀 기기/소형 디지털 온도계의 칩셋들도 기본 온도 센서는 비슷하게 스위치 구성되어 있으나 오차 범위 측정(정밀도 오차 범위)로직이 보강된 센서 모듈들은 고부가가치 센서 제품입니다)

DHT11 온도센서는 대부분 실내 온도 측정 용도로 많이 사용되는 온도센서 모듈입니다. 온도 측정 범위는  $0\text{~}50^{\circ}\text{C}$  오차범위  $2^{\circ}\text{C}$ ,  $\pm 2^{\circ}\text{C}$ , 습도 측정 범위는  $20\%\text{~}90\%$ , 오차범위는  $\pm 5\%$ 입니다. 실내 온/습도 측정에 유용한 모듈입니다.

끓는 물, 기름 또는 영하의 온도를 측정하기 위해서는 다른 온도 센서 모듈을 사용해야 합니다.

DHT11 센서 PCB 형태의 모듈 작동 전원은 3.3~5.5V 모듈입니다.  
5V 사용하도록 합니다.

아두이노 연결:

| DHT11 모듈 | 아두이노 우노 |
|----------|---------|
| -        | GND     |
| Out      | Any 포트  |
| +        | 5V      |

DH11 DataSheet:

[http://www.gameplusedu.com/pds/gpshop/arduino/pdf/Temperature\\_And\\_Humidity\\_Module\\_DHT11.pdf](http://www.gameplusedu.com/pds/gpshop/arduino/pdf/Temperature_And_Humidity_Module_DHT11.pdf)

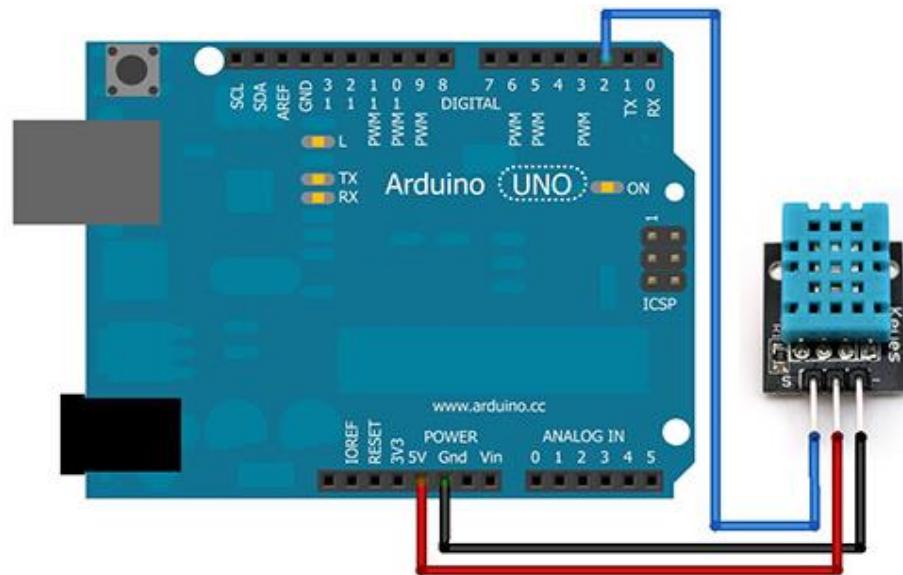


그림 28-2 아두이노 연결 회로 그림

DHT11 온/습도 센서 모듈의 형태는 아래의 형태와 같이 Power LED 표시 가능 PCB 형태도 있습니다. 작동 표시 LED On 표시 되는 모듈입니다. 모듈 사용법은 위의 모듈과 동일하게 사용합니다.



그림 28-3 DHT11 Module with Power Indicator

동일한 모델이지만 작동 표시 램프가 달린 PCB 형태도 있습니다. Power LED indicator 모듈은 장시간 사용시 작동 유/무 가시적으로 즉시 확인 가능하다는 장점이 있습니다.

DHT11 모듈을 사용하기 위한 아두이노 라이브러리가 있습니다.

아두이노 예제 및 라이브러리 참조:

아두이노 사이트에서의 설명 <http://playground.arduino.cc/main/DHT11Lib>

DHT11 Library 아래의 적용되는 라이브러리 파일.

<http://www.gameplusedu.com/pds/gpshop/arduino/pdf/DHTlib.zip>

라이브러리 예제 코드) DHTlib 예제 코드 설명

```
//  
// dht11_test.ino  
  
#include <dht.h>  
  
dht DHT;  
  
#define DHT11_PIN 2 // 연결된 포트 지정.  
  
void setup()  
{  
    Serial.begin(9600);  
    Serial.println("DHT TEST PROGRAM ");  
    Serial.print("LIBRARY VERSION: ");  
    Serial.println(DHT_LIB_VERSION);  
    Serial.println();
```

```

    Serial.println("Type,\tstatus,\tHumidity
    (%),\tTemperature (C)");
}

void loop()
{
    // READ DATA
    Serial.print("DHT11, \t");
    int chk = DHT.read11(DHT11_PIN);
    switch (chk)
    {
        case DHTLIB_OK:
            Serial.print("OK,\t");
            break;
        case DHTLIB_ERROR_CHECKSUM:
            Serial.print("Checksum error,\t");
            break;
        case DHTLIB_ERROR_TIMEOUT:
            Serial.print("Time out error,\t");
            break;
        case DHTLIB_ERROR_CONNECT:
            Serial.print("Connect error,\t");
            break;
        case DHTLIB_ERROR_ACK_L:
            Serial.print("Ack Low error,\t");
            break;
        case DHTLIB_ERROR_ACK_H:
            Serial.print("Ack High error,\t");
            break;
        default:
            Serial.print("Unknown error,\t");
            break;
    }
    // DISPLAY DATA
    Serial.print(DHT.humidity, 1);
    Serial.print(",\t");
    Serial.println(DHT.temperature, 1);

    delay(2000);
}
// 
// END OF FILE
//

```

## 28.2 DHT11 센서

DHT11 온/습도 센서 본체입니다. 센서 모듈 본체와 저항을 사용하여 회로를 만들어서 온/습도 측정 가능합니다.

| DHT11 핀 |      |
|---------|------|
| 1       | VCC  |
| 2       | DATA |
| 3       | NC   |
| 4       | GND  |



DHT11 센서 모듈 설명서에는 2번 데이터 핀과 통신을 하기 위해서는 Pull-Up 회로를 구성해야 합니다. 풀업 저항은

센서 본체를 바로 사용하는 경우에는 5K 이하의 저항 (OR 4.7K 저항)을 사용하여 회로를 구성해야 합니다. 아두이노 보드에 사용하는 경우에는 10K 저항을 사용하여 회로를 구성하여도 무방합니다. 제조사에 따라 5K 정도의 저항, 또는 10K 저항을 사용하는 경우가 있습니다.

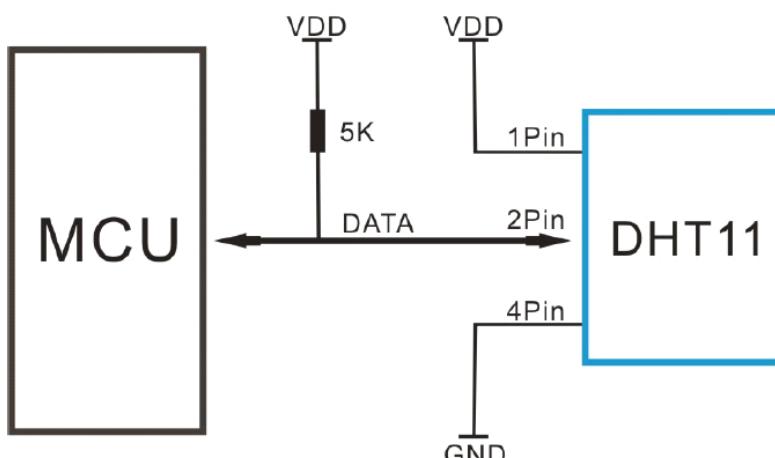


그림 28-1 연결 회로도

온/습도 표시를 하기 위한 LCD1602 모듈 라이브러리입니다.

LCD1602 I2C Library.

[http://www.gameplusedu/pds/gpshop/arduino/pdf/LiquidCrystal\\_I2C.zip](http://www.gameplusedu/pds/gpshop/arduino/pdf/LiquidCrystal_I2C.zip)

온도센서 모듈을 연결합니다. 예제에 사용되는 모듈은 이미 저항 구성이 되어있는 관계로 브레드보드에 점퍼선 연결하여 회로를 구성하도록 합니다.



그림 28-2 브레드보드 장착 참조 그림

키트에 포함되어 있는 LCD1602 I2C 모듈과 온/습도 센서 모듈을 사용하여 연결하도록 합니다.

LCD1602 I2C 모듈에 현재 측정된 온도와 습도를 표시합니다.

모든 센서 측정은 지정된 시간에 의한 평균값을 측정해야 정확한 값이지만, 온/습도 자체는 급격하게 변하는 형태의 데이터가 아니므로, 센서 모듈로부터 받은 1 회 측정 값을 그대로 출력하여도 무방합니다.

물론, 프로젝트의 목적에 따라 1 분, 10 분, 1 시간등의 시간편차에 대한 데이터가 필요한 경우에는 타이머 카운트를 하여 데이터를 내보내도록 프로그래밍 해야 합니다.

아래의 예제 코드는 온/습도 센서 모듈로부터 받은 데이터를 디스플레이 장치에 표시 해주는 예제 코드입니다.

사용되는 LCD1602 I2C 모듈은 아두이노 I2C 포트에 연결하도록 합니다.

| LCD1602 I2C 모듈 | 아두이노 우노 |
|----------------|---------|
| GND            | GND     |
| VCC            | 5V      |
| SDA            | A4      |
| SCL            | A5      |

예제코드)

[http://www.gameplusedu.com/pds/gpshop/kit\\_code/strongest/dht11\\_ex\\_1.ino](http://www.gameplusedu.com/pds/gpshop/kit_code/strongest/dht11_ex_1.ino)

```
#include <dht.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

#define DHT11_PIN 8

dht DHT;
//LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
LiquidCrystal_I2C lcd(0x27,16,2);

void setup() {
  Serial.begin(9600);
  lcd.init();
  lcd.backlight();

}

int getDHT(int& t, int &h)
{
  int chk = DHT.read11(DHT11_PIN);
  switch (chk)
  {
    case DHTLIB_OK:
      Serial.print("OK,\t");
      break;
    case DHTLIB_ERROR_CHECKSUM:
      Serial.print("Checksum error,\t");
      break;
    case DHTLIB_ERROR_TIMEOUT:
      Serial.print("Time out error,\t");
      break;
  }
}
```

```

case DHTLIB_ERROR_CONNECT:
    Serial.print("Connect error,\t");
    break;
case DHTLIB_ERROR_ACK_L:
    Serial.print("Ack Low error,\t");
    break;
case DHTLIB_ERROR_ACK_H:
    Serial.print("Ack High error,\t");
    break;
default:
    Serial.print("Unknown error,\t");
    break;
}
if(chk==DHTLIB_OK)
{
    t=DHT.temperature;
    h=DHT.humidity;
}
}

void loop() {
    int temp,humidity;
    int ret=getDHT(temp,humidity);
    if(ret!=DHTLIB_OK)
    {
        lcd.setCursor(0,0);
        lcd.print("Temp=error");
        lcd.setCursor(0,1);
        lcd.print("Humidity=error");
    }
    else
    {
        lcd.setCursor(0,0);
        lcd.print("Temp=");
        lcd.print(temp);
        lcd.print(" *C");
        lcd.setCursor(0,1);
        lcd.print("Humidity=");
        lcd.print(humidity);
        lcd.print("% ");
        delay(500);
    }
}

```

## 29 › LEVEL DETECTION MODULE X 1

물높이 측정 센서입니다.

(Raindrop Water Level / Height Depth Detection Sensor Module)



그림 29-1 물높이 측정 센서 모듈

기술 사양:

Product name: water level sensor, Liquid Level Sensor

Operating voltage: DC 3.3V – 5.5V

Operating current: less than 20mA

Sensor type: analog

Detection area: 40 x 16mm (측정 영역)

Production process: FR4 double-sided HASL

Operating temperature: 10°C – 30°C

Humidity: 10% - 90% non-condensing

Product dimensions: 62 x 20 x 8mm

센서 모듈의 빗금 부분을 물속에 넣고 물높이를 체크합니다.

액체의 표면 높이에 따라 출력 저항이 변화하게 구성된 센서 모듈입니다.

적용 가능 용도는 낮은 물 수위 체크, 강수량 체크, 물 탱크 등의 일정 수위 넘침 체크 등에 사용할 수 있습니다. 또는 물이 있지 않아야 하는 장소의 물높이 체크도 할 수 있습니다.

센서 측정의 원리는 커패시터(컨덴서) 구성 재료의 특성을 응용한 것입니다. 전해질이 많은 경우 전류가 잘 통하고, 반대일 경우 전류가 잘 안 통합니다.

### 아두이노 연결

| Level 모듈 | 아두이노 우노 |
|----------|---------|
| S        | A0      |
| +        | 5V      |
| -        | GND     |

예제코드:

아날로그 포트 0 번에서 값을 읽어 시리얼 프린트 합니다.

```
void setup() {
  Serial.begin(9600);
}

void loop()
{
  // 아날로그 포트 0 번 읽기.
  int sensorValue=analogRead(A0);
  Serial.print("Water Level Sensor Value:");
  Serial.println(sensorValue);
  //
  delay(200);
}
```

예제코드: 일정 수치 값에 도달하면 부저 울리기

[http://www.gameplusedu.com/pds/gpshop/kit\\_code/strongest/water\\_level\\_ex\\_1.ino](http://www.gameplusedu.com/pds/gpshop/kit_code/strongest/water_level_ex_1.ino)

```
/*macro definition of water sensor and the buzzer*/
#define WATER_SENSOR 2
#define BUZZER 3
void setup()
{
  pins_init();
}
void loop()
{
  if(isExposedToWater())
    soundAlarm();
}
void pins_init()
{
  pinMode(WATER_SENSOR, INPUT);
```

```

        pinMode(BUZZER, OUTPUT);
    }
    *****/
/*Function: When the sensor is exposed to the water, the buzzer sounds */
/*           for 2 seconds.
*/
void soundAlarm()
{
    for(uint8_t i = 0;i < 20;i++)
    {
        digitalWrite(BUZZER, HIGH);
        delay(50);
        digitalWrite(BUZZER, LOW);
        delay(50);
    }
}
 *****/
/*Function: Determine whether the sensor is exposed to the water */
/*Parameter:-void
*/
/*Return: -boolean,if it is exposed to the water,it will return true. */
boolean isExposedToWater()
{
    if(digitalRead(WATER_SENSOR) == LOW)
        return true;
    else return false;
}

```

모듈 부분(커넥터 하부)에만 물이 닿게 하고, 다른 부위(보드, 커넥터 포함)에는 물이 닿지 않게 합니다. 만약 물 높이 센서 측정 길이 이상으로 여러 개를 높이 단위에 따라(저수지 등등) 측정 하신다면 다른 여러 모듈 제품들도 있지만 방수 글리세린을 케이블 포함되게 충분히 덮어주는 식으로 가공하시면 충분히 Waterproof 제품을 구입하지 않고서도 사용 가능합니다.

다만 방수 글리세린은 환경에 따라 다르겠지만 영구적인 방수가 아닙니다.  
환경에 따라 3~4 일 정도 사용 가능하다고 합니다.

## 29.1 여러 개의 아날로그 센서 사용시 주의사항.

A0 포트에는 아날로그 온도센서, A1 에는 사운드 센서, A2 에는 다른 아날로그 센서 등등 사용 할 경우에는 값을 제대로 읽어 오지 못하거나 이상한 값이 넘어오는 경우가 있습니다. 센서 자체의 Fetch 반환 기능이 늦거나, 아두이노의 ADC 처리 기능에도 최소한의 지연 시간이 발생됩니다.

위와 같은 문제 해결 방법은 여러 방법이 있겠지만, 간단하게 지연 함수를 사용 해봅니다.

이런 경우는 아래의 예시 코드처럼 지연 함수를 사용해야 합니다.

```
int temp=analogRead(A0); // 온도 센서  
delay(100); // 적절한 값 테스트 요망  
int snd_value = analogRead(A1); // 사운드 센서  
delay(100); // 적절한 값 테스트 요망  
int temp=analogRead(A2); // 물높이 센서  
delay(100); // 적절한 값 테스트 요망
```

물론 위의 지연 함수(delay)를 사용하지 않고 제대로 값이 넘어온다면 delay 함수 사용 안 합니다.

여러 개의 아날로그 입력 핀 연결은 개별적으로 분리 되어 있으나 ADC(아날로그 디지털 변환기)라는 MCU 에서의 처리 부분이 여러 개의 아날로그 처리 시 제대로 처리되지 않을 수 있습니다.

## 30> 4\*4 KEYPAD MODULE X 1

4x4 버튼 키패드입니다. 16 개 버튼 입력 테스트 가능합니다.



그림 30-1 4x4 버튼 키패드 PCB

동일한 기능을 가진 멤브레인 방식의 키패드입니다.



그림 30-2 4x4 멤브레인 키패드

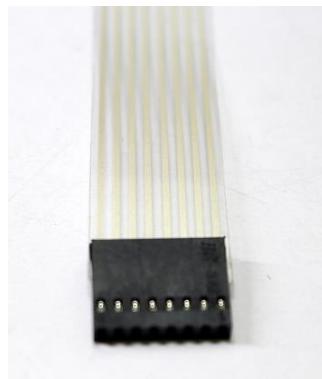


그림 30-3 멤브레인 키패드 연결 부분

아두이노 보드에 4x4 등의 KeyPad 모듈 사용시에는 KeyPad 라이브러리를 사용하도록 합니다. KeyPad 라이브러리는 아래의 사이트에서 상세히 설명 되어 있습니다.  
키패드 라이브러리 참조 사이트:

<http://playground.arduino.cc/Code/Keypad>

참조 사이트 본문에는 공개 라이브러리 다운로드 주소가 있습니다.

<http://playground.arduino.cc/uploads/Code/keypad.zip>

또는 다양한 키패드를 사용하고자 하는 경우에는

[https://github.com/joeyoung/arduino\\_keypads](https://github.com/joeyoung/arduino_keypads) 깃헙 키패드 라이브러리 주소에서 다운로드 받아서 사용하도록 합니다. 키패드 I2C 방식 등의 여러가지 라이브러리들이 있습니다.

keypad.zip 다운로드 받아 임의의 디렉터리에 압축 해제 하였을 경우, 아두이노 스케치 IDE 의 “스케치북 위치:” 아래의 libraries 하위 디렉터리에 keypad 디렉터리를 복사 해주어야 합니다. 이런 경우에는 아두이노 스케치 IDE 를 재 실행해주어야 라이브러리 목록, 정보가 갱신 되어 사용 가능 합니다.

또는 아두이노 스케치 IDE 메뉴의 “스케치” -> “include Library” -> “Add Zip Library” 메뉴 선택을 하면, ZIP 파일 선택ダイ얼로그 창이 나오면 다운로드 받은 keypad.zip 을 선택하여 주면 자동으로 설치가 됩니다.(아두이노 스케치 IDE 버전 1.6.4)

아래의 예제 코드처럼 헤더 파일을 선언하여 사용합니다.

```
#include <Keypad.h> // keypad 라이브러리 사용
```

예제코드)

[http://www.gameplusedu.com/pds/gpshop/kit\\_code/strongest/keypad\\_4x4\\_ex\\_1.ino](http://www.gameplusedu.com/pds/gpshop/kit_code/strongest/keypad_4x4_ex_1.ino)

```

#include <Keypad.h>

const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
    {'1','2','3','A'},
    {'4','5','6','B'},
    {'7','8','9','C'},
    {'*','0','#','D'}
};

/* 아두이노 우노 보드 와이어링 편입니다.*/
byte rowPins[ROWS] = {2,3,4,5}; //connect to row pinouts
byte colPins[COLS] = {6,7,8,9}; //connect to column pinouts

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS,
COLS );

void setup()
{
    Serial.begin(9600);
}

void loop()
{
    char key = keypad.getKey();

    if (key != NO_KEY)
    {
        Serial.println(key);
    }
}

```

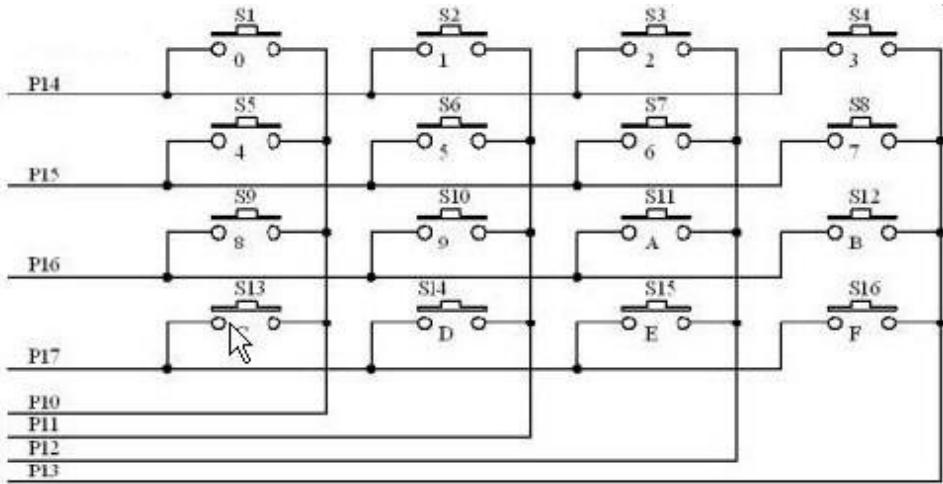


그림 30-4 4x4 키패드 회로도

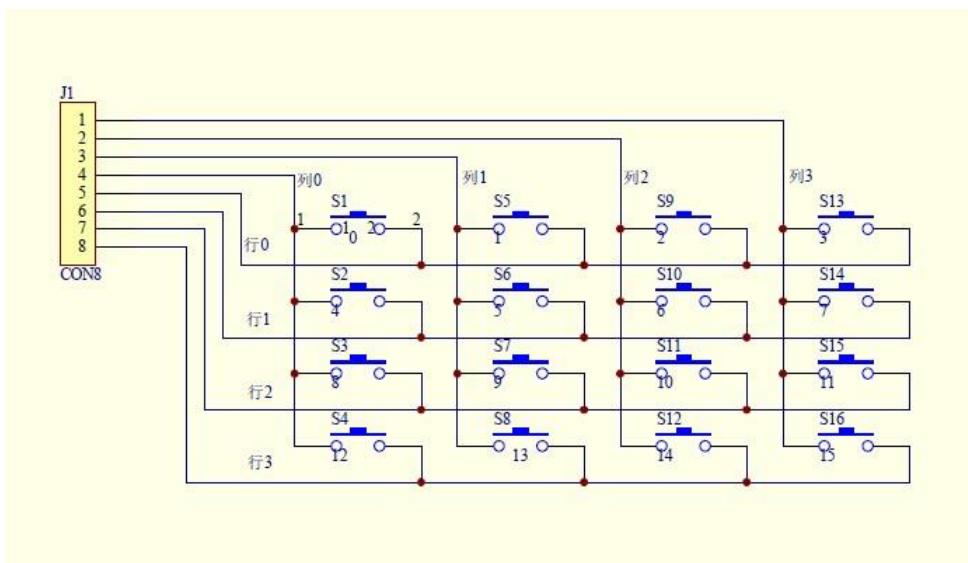


그림 30-5 좀 더 보기 쉬운 회로도

아두이노 예제코드 & 참조

<http://playground.arduino.cc/Main/KeypadTutorial>

## 31 > THREE-COLOR RGB MODULE X 1

RGB Color LED 모듈입니다.

RGB 값에 의한 LED 색상 변경 가능합니다. 5파이 (5 Pi)크기의 RGB LED 가 적용된 모듈입니다.



그림 31-1 RGB LED 모듈

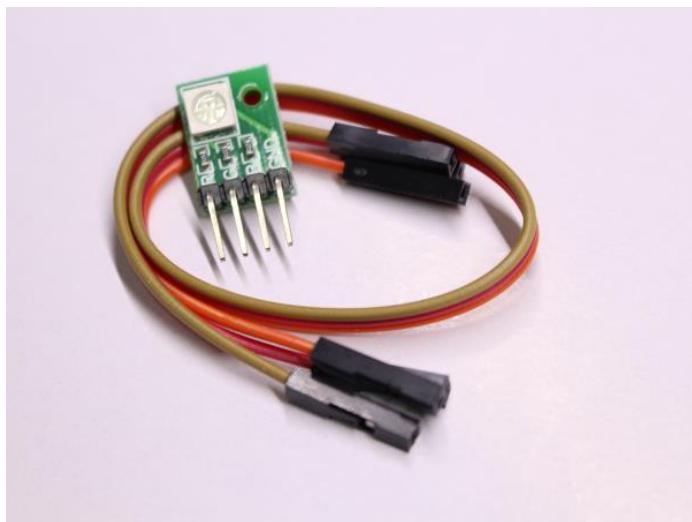


그림 31-2 RGB LED SMD 모듈

2개의 모듈 동일한 기능입니다. 크기와 형태만 다릅니다.

아두이노 연결:

| RGB LED Module | Arduino Uno |
|----------------|-------------|
| GND            | GND         |
| R              | 11          |
| G              | 10          |
| B              | 9           |

아두이노 예제 코드입니다. 1 초마다 색상 변경해주는 코드입니다.

아두이노에 연결 시 9,10,11 주의하실 점은 PWM 포트에 연결합니다.

포트에 `analogWrite()` 함수를 사용하기 위해서는 PWM 지원 포트를 지정 해 주어야 합니다. `analogWrite()` 함수를 사용하여 PWM 포트에 각각의 RGB 값을 출력하여 줍니다.

## 31.1 RGB LED 모듈 형태가 아닌 부품 사용

### 31.1.1 CATHODE (캐소드) Type

캐소드 타입의 RGB LED 는 왼쪽에서 2 번째 GND 핀을 공통으로 사용하게 되어 있습니다.

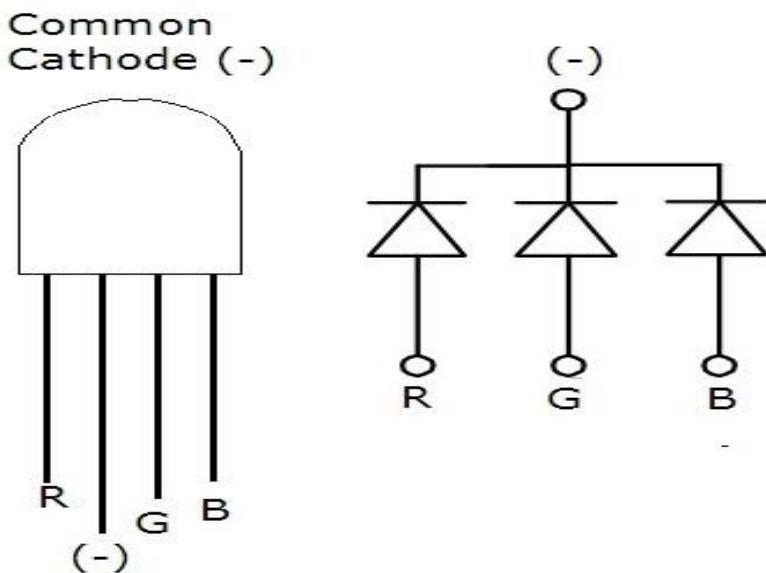


그림 31-3 RGB LED CATHODE 핀 순서

RGB LED 부품을 사용 하는 경우에는 GND 를 제외한 나머지 RGB 핀에는 220R 정도의 저항을 연결하여 사용합니다.

아두이노 연결:

| RGB LED Module | Arduino Uno |
|----------------|-------------|
| GND            | GND         |
| R              | 9           |
| G              | 10          |
| B              | 11          |

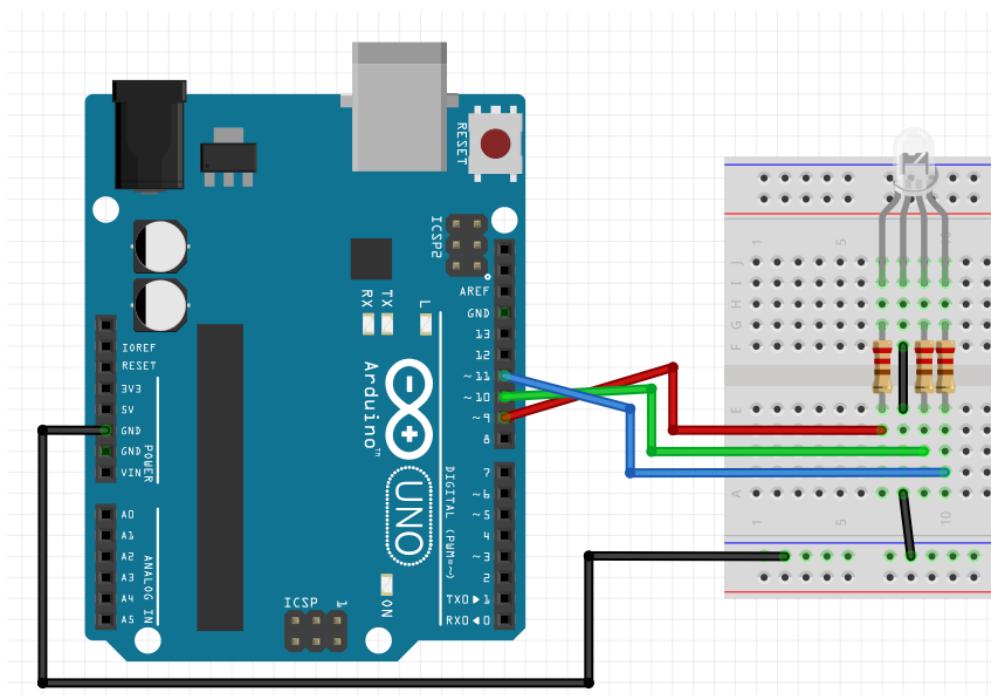


그림 31-4 RGB LED 브레드보드 회로도

무대 조명, 빛 조명에서의 컬러 조절 및 배합은 상당히 예술적인 감각을 요하는 부분이기도 합니다. 위의 모듈로 기본적인 개념을 이해 하시고, 24 비트, RGB 의 색상 조절 개념을 숙지 하신다면 차후 전문가 or 취미 용도로도 활용 가능한 범위입니다.

아래의 예시 코드 적용 후 색상이 정확하지 않다면 와이어링 다시 체크 해 주세요.

코드 적용 후 1초마다 색상 변경 됩니다.

빨간색/초록색/파란색/노란색/보라색 순서로 반복됩니다.

예제코드)

[http://www.gameplusedu.com/pds/gpshop/kit\\_code/strongest/led\\_rgb\\_ex\\_1.ino](http://www.gameplusedu.com/pds/gpshop/kit_code/strongest/led_rgb_ex_1.ino)

```
/*
Arduino - RGB LED
*/

int redPin = 11;
int greenPin = 10;
int bluePin = 9;

void setup()
{
    Serial.begin(9600);
    pinMode(redPin, OUTPUT);
    pinMode(greenPin, OUTPUT);
    pinMode(bluePin, OUTPUT);
}

void loop()
{
    Serial.println("red");
    setColor(255, 0, 0); // red
    delay(1000);
    Serial.println("green");
    setColor(0, 255, 0); // green
    delay(1000);
    Serial.println("blue");
    setColor(0, 0, 255); // blue
    delay(1000);
    Serial.println("yellow");
    setColor(255, 255, 0); // yellow
    delay(1000);
    Serial.println("purple");
    setColor(80, 0, 80); // purple
    delay(1000);
    Serial.println("aqua");
    setColor(0, 255, 255); // aqua
    delay(1000);
}
```

```

void setColor(int red, int green, int blue)
{
    analogWrite(redPin, red);
    analogWrite(greenPin, green);
    analogWrite(bluePin, blue);
}

```

### 31.1.2 ANODE (애노드) TYPE

애노드 타입의 RGB LED 는 왼쪽에서 2 번째 VCC 핀을 공통으로 사용하게 되어 있습니다.

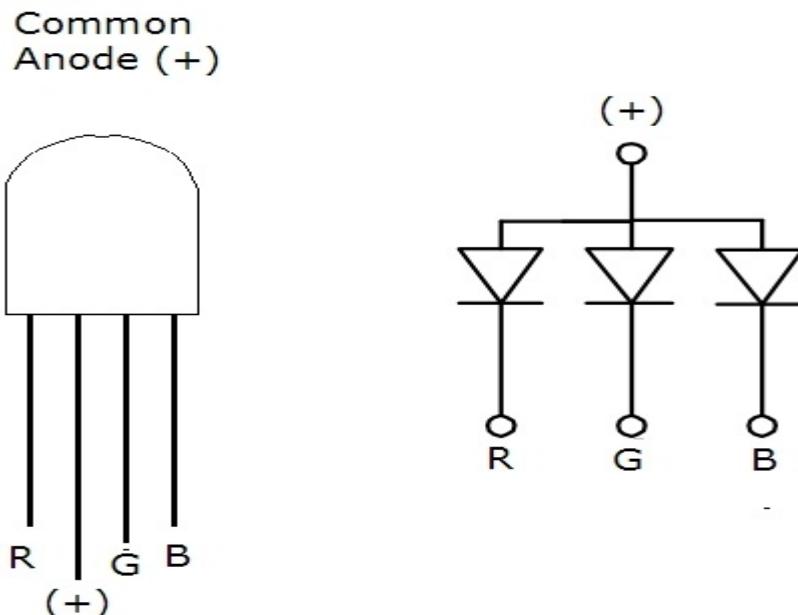
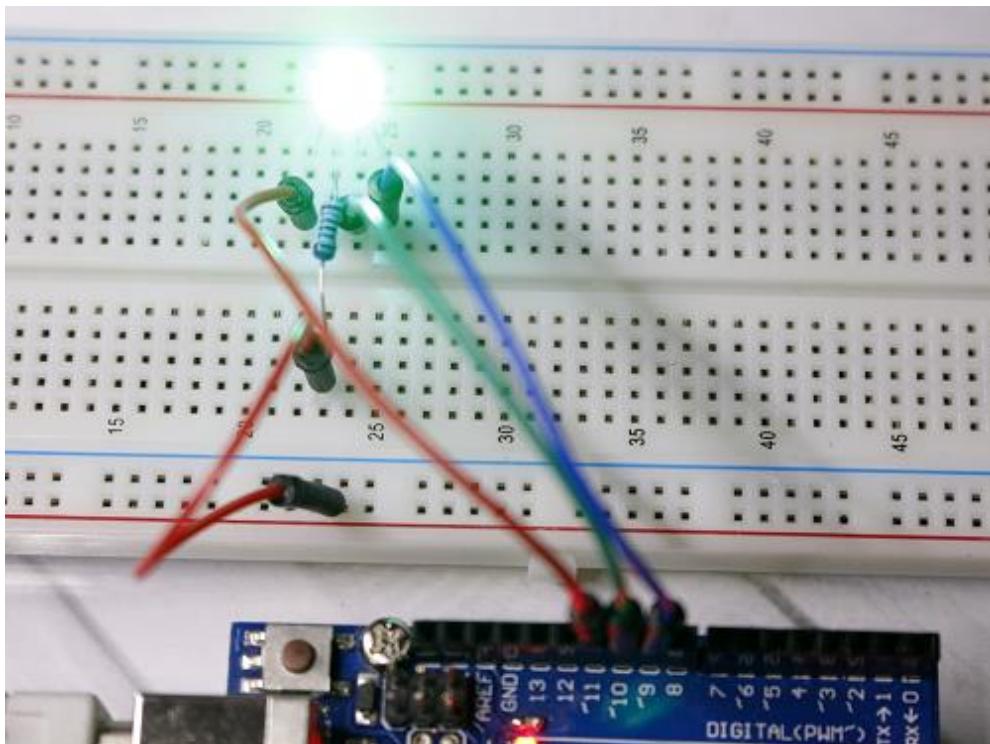


그림 31-5 RGB LED ANODE 핀 순서

RGB LED 2 번 핀에만 220R, 300R 정도의 저항만 연결하여 주면 됩니다.

### 아두이노 연결:

| RGB LED Module | Arduino Uno |
|----------------|-------------|
| (+)            | 5V          |
| R              | 9           |
| G              | 10          |
| B              | 11          |



예제코드: 캐소드 적용 코드와는 반대로 0~255 순서가 아닌 255부터 0 입니다.

[http://www.gameplusedu.com/pds/gpshop/kit\\_code/strongest/led\\_rgb\\_ex\\_2.ino](http://www.gameplusedu.com/pds/gpshop/kit_code/strongest/led_rgb_ex_2.ino)

```
/*
 *
int redPin = 11;
int greenPin = 10;
int bluePin = 9;

//uncomment this line if using a Common Anode LED
#define COMMON_ANODE
```

```
void setup()
{
    pinMode(redPin, OUTPUT);
    pinMode(greenPin, OUTPUT);
    pinMode(bluePin, OUTPUT);
}

void loop()
{
    setColor(255, 0, 0); // red
    delay(1000);
    setColor(0, 255, 0); // green
    delay(1000);
    setColor(0, 0, 255); // blue
    delay(1000);
    setColor(255, 255, 0); // yellow
    delay(1000);
    setColor(80, 0, 80); // purple
    delay(1000);
    setColor(0, 255, 255); // aqua
    delay(1000);
}

void setColor(int red, int green, int blue)
{
    #ifdef COMMON_ANODE
        red = 255 - red;
        green = 255 - green;
        blue = 255 - blue;
    #endif
    analogWrite(redPin, red);
    analogWrite(greenPin, green);
    analogWrite(bluePin, blue);
}
```

## 32 › XY JOYSTICK X 1

XY joystick 입니다.(Joystick Module Dual-axis XY for Arduino)  
PS2 Game Joystick Module For Arduino



그림 32-1 PS2 조이스틱 모듈

X 축, Y 축, 1 버튼 기능을 가진 조이스틱 모듈입니다. X, Y 축의 움직임을 2 개의 아날로그 신호로 받을 수 있습니다. 일반적인 조이스틱처럼 조종이 가능하며 조이스틱을 꾹 누르면(Z 축 기능) 버튼 기능이 됩니다.

조이스틱 모듈은 PS2 게임기에 사용되던 조이스틱과 동일한 모듈입니다. RC 컨트롤러와 여러 산업용 기기 등에 많이 사용되고 있습니다. 위의 버튼 캡 형태만 조금 다를 뿐, 내부의 모듈은 동일한 것으로 많이 사용되고 있습니다.

아두이노 연결:

| XY Joystick Module | Description | Arduino           |
|--------------------|-------------|-------------------|
| VCC                | 전원          | 5V                |
| GND                | 그라운드        | GND               |
| VRx                | 가로 방향       | 아날로그 포트 A0        |
| VRy                | 세로 방향       | 아날로그 포트 A1        |
| SW(switch)         | 버튼          | 디지털 포트 2 번 or Any |

아두이노 보드와 연결 후 아래의 예제코드를 사용하여 시리얼 모니터 창으로 출력되는 값들을 살펴봅니다.

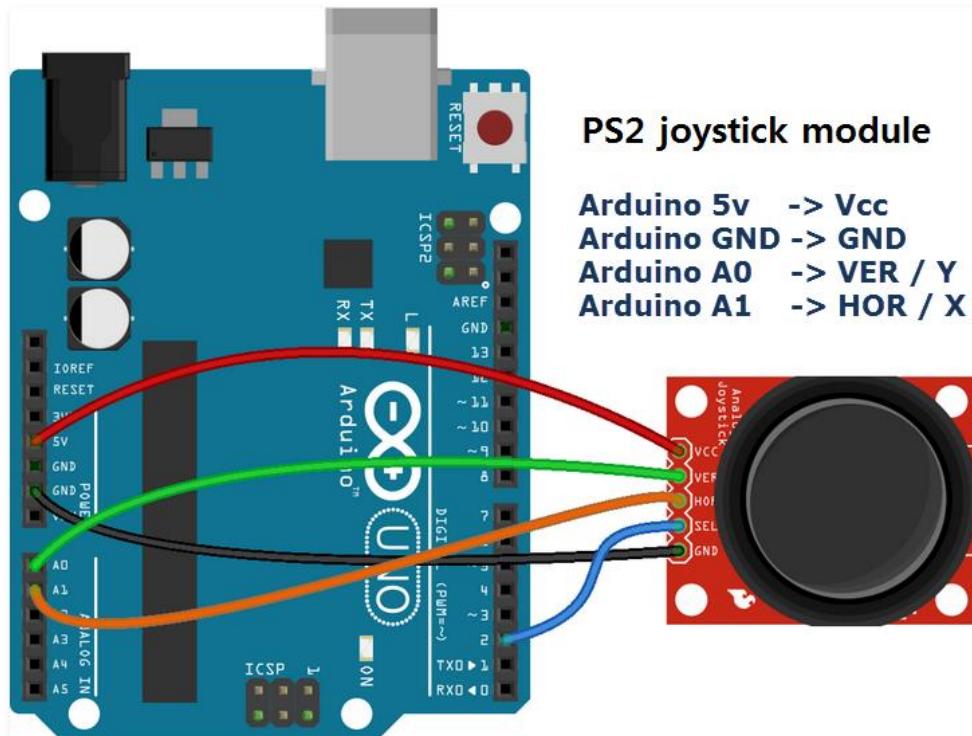


그림 32-2 우노 R3 보드와 연결

예제코드: 아두이노 보드에 바로 연결 후 입력되는 값을 보기 위한 코드입니다.  
[http://www.gameplusedu.com/pds/gpshop/kit\\_code/strongest/ps2\\_joystick\\_ex\\_1.ino](http://www.gameplusedu.com/pds/gpshop/kit_code/strongest/ps2_joystick_ex_1.ino)

```

int xPin = A1;
int yPin = A0;
int buttonPin = 2;

int xPosition = 0;
int yPosition = 0;
int buttonState = 0;

void setup() {
    // initialize serial communications at 9600 bps:
    Serial.begin(9600);

    pinMode(xPin, INPUT);
    pinMode(yPin, INPUT);

    //activate pull-up resistor on the push-button pin
    pinMode(buttonPin, INPUT_PULLUP);
}

```

```

// For versions prior to Arduino 1.0.1
// pinMode(buttonPin, INPUT);
// digitalWrite(buttonPin, HIGH);

}

void loop() {
    xPosition = analogRead(xPin);
    yPosition = analogRead(yPin);
    buttonState = digitalRead(buttonPin);

    Serial.print("X: ");
    Serial.print(xPosition);
    Serial.print(" | Y: ");
    Serial.print(yPosition);
    Serial.print(" | Button: ");
    Serial.println(buttonState);

    delay(100); // add some delay between reads
}

```

예제코드: 아두이노 사이트 튜토리얼 참조 (조이스틱 형태가 조금 다르게 보이지만 개념은 같습니다)

<http://www.arduino.cc/en/Tutorial/JoyStick>

```

// # Description:
// # Modify the Sample code for the Joystick Module
// # Connection:
// #      X-Axis  -> Analog pin 0
// #      Y-Axis  -> Analog pin 1
// #      Z-Axis  -> Digital pin 3
// #

int JoyStick_X = 0; //x // A0
int JoyStick_Y = 1; //y // A1
int JoyStick_Z = 3; //key // D3

void setup()
{

```

```
pinMode(JoyStick_Z, INPUT);
Serial.begin(9600); // 9600 bps
}
void loop()
{
    int x,y,z;
    x=analogRead(JoyStick_X);
    y=analogRead(JoyStick_Y);
    z=digitalRead(JoyStick_Z);
    Serial.print(x ,DEC);
    Serial.print(",");
    Serial.print(y ,DEC);
    Serial.print(",");
    Serial.println(z ,DEC);
    delay(100);
}
```

## 33 > SERVO X 1

SG90 모듈입니다.

서보 (서보 메커니즘) 모터입니다. 또는 Servo 단어 의미에는 Servant라는 단어와 연관되어 주인 명령에 충실한 의미도 있다고 합니다.



그림 33-1 서보 모터 SG90 모듈

위 모듈은 180 DEGREE (180 도 회전)입니다.

실제 사용시에는 180 도 적용 되긴 합니다. 데이터 쉬트와 실제 모듈과의 오류인지, 제조사 데이터 쉬트를 정확히 확인 하시기 바랍니다.

스케치 기본 라이브러리에서의 각도는 180 도로 구현되어 있습니다.

### Specifications

Weight: 9 g

Dimension: 22.2 x 11.8 x 31 mm approx.

Stall torque: 1.8 kgf·cm

Operating speed: 0.1 s/60 degree

Operating voltage: 4.8 V (~5V)

Dead band width: 10  $\mu$ s

Temperature range: 0 °C – 55 °C

모듈의 연결선은 3 개가 있습니다.

| 서보 모터 모듈            | Arduino Uno         |
|---------------------|---------------------|
| Black wire          | GND                 |
| Red wire            | 5V                  |
| Blue or Yellow wire | 9 or other PWM port |

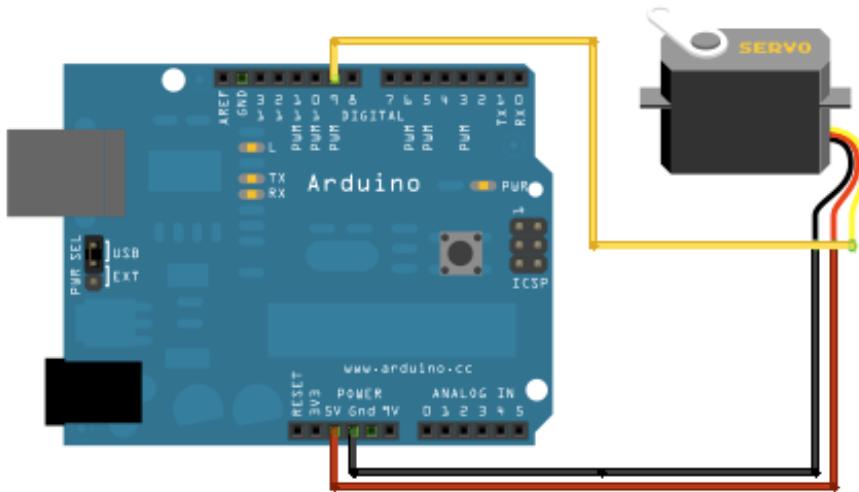


그림 33-2 아두이노 우노 R3 보드와 연결

아두이노 사이트에 서보 모터 예제 및 설명이 있습니다.

1 번째 예제는 <http://arduino.cc/en/Tutorial/Sweep>

예제는 0 ~ 180 도 회전 후 다시 0 위치로 복구 되는 소스입니다.

```
// Sweep
// by BARRAGAN <http://barraganstudio.com>
// This example code is in the public domain.

#include <Servo.h>

Servo myservo; // create servo object to control a servo
                // a maximum of eight servo objects can be created

int pos = 0;    // variable to store the servo position

void setup()
```

```

{
    myservo.attach(9); // attaches the servo on pin 9 to the servo object
}

void loop()
{
    for(pos = 0; pos < 180; pos += 1) // goes from 0 degrees to 180 degrees
    {
        myservo.write(pos);           // tell servo to go to position in
variable 'pos'
        delay(15);                  // waits 15ms for the servo to reach the
position
    }
    for(pos = 180; pos>=1; pos-=1) // goes from 180 degrees to 0 degrees
    {
        myservo.write(pos);           // tell servo to go to position in
variable 'pos'
        delay(15);                  // waits 15ms for the servo to reach the
position
    }
}

// 2 번째 예제는 http://arduino.cc/en/Tutorial/Knob
// 포텐시미터(가변저항)의 아날로그 값을 받아 모터 컨트롤 하는 예제입니다.
// Controlling a servo position using a potentiometer (variable resistor)
// by Michal Rinott <http://people.interaction-ivrea.it/m.rinott>

#include <Servo.h>

Servo myservo; // create servo object to control a servo

int potpin = 0; // analog pin used to connect the potentiometer
int val; // variable to read the value from the analog pin

void setup()
{
    myservo.attach(9); // attaches the servo on pin 9 to the servo object
}

void loop()
{

```

```

// reads the value of the potentiometer (value between 0 and 1023)
val = analogRead(potpin);
// scale it to use it with the servo (value between 0 and 180)
// 아날로그 입력된 0~1023 의 정수를 0~179 범위로 변경.
val = map(val, 0, 1023, 0, 179);
// sets the servo position according to the scaled value
myservo.write(val);
// waits for the servo to get there
delay(15);
}

```

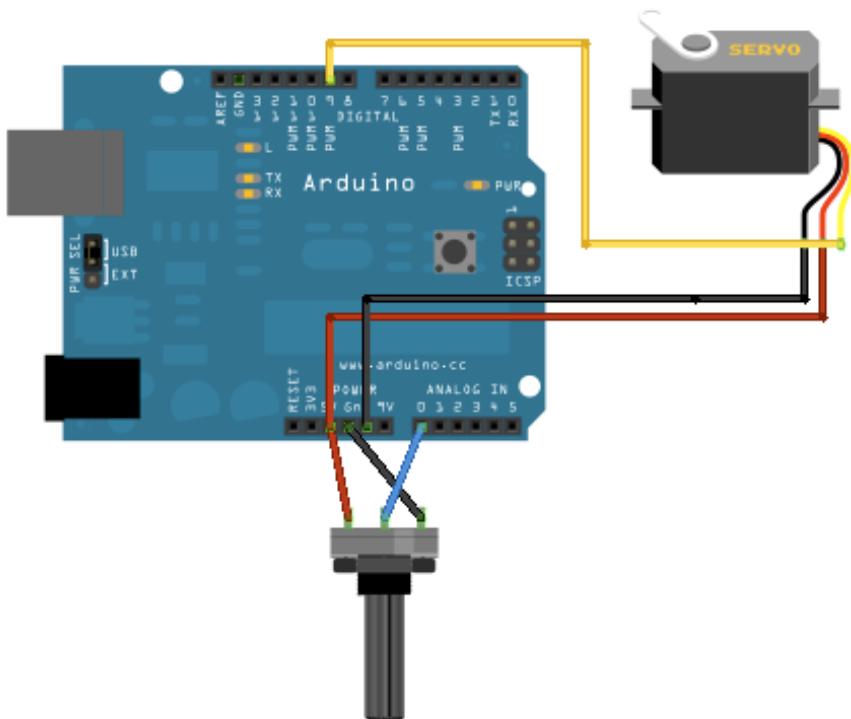


그림 33-3 가변저항 연결 후 서보 모터 작동 회로도

위의 예제코드에서 사용된 `map()` 함수는 지정된 범위내의 입력된 값을 지정된 범위의 비율로 변환하여 반환하는 함수입니다.

### » map 함수 사용법

지정된 비율의 입력된 값을 다른 비율의 값으로 구할 때 사용되는 함수입니다.

## map(value, fromLow, fromHigh, toLow, toHigh)

value: 변환에 사용되는 변수

fromLow: 입력 되는 value 최소값 범위 지정

fromHigh: 입력 되는 value 최대값 범위 지정.

toLow: 변환되는 최소 범위

toHigh: 변환되는 최대 범위

아래의 예제는 analogRead(0) 0 번 포트의 값을 가져와서 0~255 까지의 비율로 변환하는 예제코드입니다.

```
void setup() {}  
  
void loop()  
{  
    int val = analogRead(0);  
    val = map(val, 0, 1023, 0, 255);  
    analogWrite(9, val);  
}
```

스케치에서 사용되는 map 함수 본체입니다.

```
long map(long x, long in_min, long in_max, long out_min, long  
out_max)  
{  
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) +  
    out_min;  
}
```

## » constrain 정의 함수 사용법

주어진 값이 지정된 범위의 값을 벗어나지 않도록 만들어주는 함수입니다. 지정된 범위의 값으로 제한하는 기능의 함수입니다.

constrain(x, a, b)

x: 입력 변수.

a: 최소값

b: 최대값

```
int aVal = constrain(sensVal, 10, 150);
```

정수형 변수 sensVal 을 10, 150 범위의 값으로 제한하는 예제입니다.

만약 sensVal 값이 1 이 입력되는 경우 최소로 지정된 10 이 반환됩니다.  
150 넘어가는 경우에는 150 반환됩니다.

constrain 정의된 코드는 아래와 같습니다.

```
#define constrain(amt,low,high) ((amt)<(low)?(low):((amt)>(high)?(high):(amt)))
```

### 33.1 서보 모터 전원 공급 참조.

서보모터 사용시 sg90 등의 소량의 전류를 사용하는 모터는 우노 R3 보드 등의 5V 공급으로 2 개~3 개 정도는 충분히 작동 가능합니다. 만약, 고성능의 출력이 좋은, 서보모터를 사용하는 경우에는 우노 보드의 5V 연결로는 작동 불가입니다. 연결 시 우노 R3 보드 전원부에 무리가 가서 망가질 수도 있습니다.

이런 경우에는 서보 모터에 전원을 별도로 공급해주어야 합니다.

서보 모터, DC 모터, 스템페 모터 등의 일반적인 전원 공급 개념입니다.

전원 공급의 소스가 다른 경우에는 GND 를 공통 연결 해 주어야 합니다. GND 값이 동일해야 VCC 레벨이 정확하게 나올 수 있습니다.

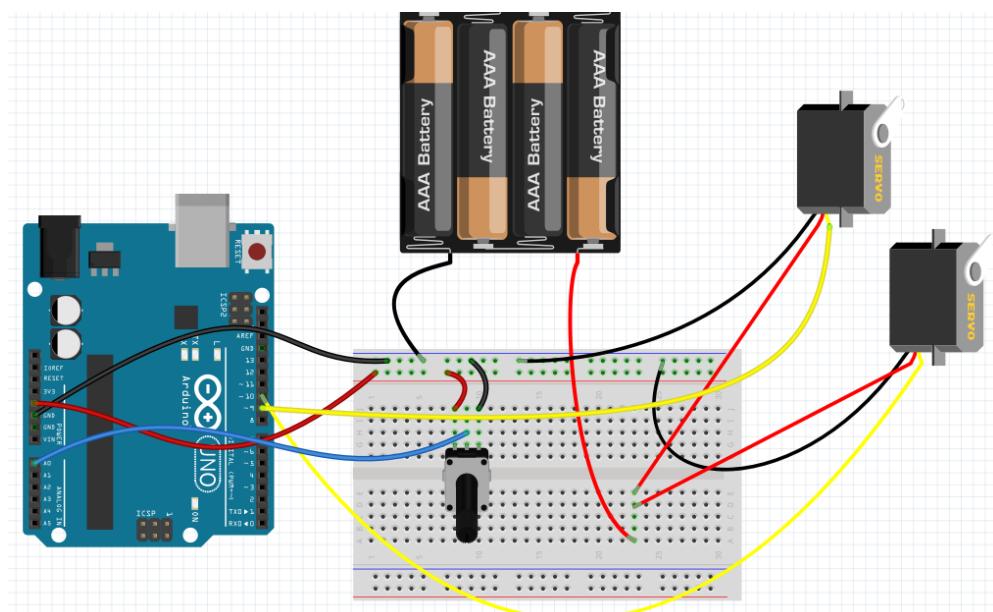


그림 33-4 서보모터 외부전원 공급 브레드보드 회로도

## 34 > STEPPER MOTOR & DRIVER BOARD X 1

스테퍼 모터입니다. 스테퍼 모터는 펄스 입력으로 컨트롤합니다. 스테퍼 모터 또 다른 용어로는 펄스 모터라고도 합니다.



그림 34-1 스텝 모터 & 제어 드라이버



그림 34-2 스텝 모터

스테퍼 모터의 용도는 모터의 회전수와 회전량 등을 정밀하게 제어해야 하는 경우에 많이 사용됩니다. 또는 경우에 따라 RC 회전바퀴에도 사용되기도 합니다.

28BYJ-48 모델 스테퍼 모터입니다. ULN2003 스테퍼 모터 컨트롤 보드를 사용합니다. ULN2003 컨트롤 IC 보드, 28BYJ-48 스테퍼 모터는 4상(Phase 4) 모터입니다. 가동 전원은 5V 사용합니다.

스테퍼 모터의 사양은 아래와 같습니다.

Rated voltage : 5VDC

Number of Phase 4

Speed Variation Ratio 1/64

Stride Angle  $5.625^\circ$  (360/64)

Frequency 100Hz

DC resistance  $50\Omega \pm 7\%$ ( $25^\circ\text{C}$ )

Idle In-traction Frequency > 600Hz

Idle Out-traction Frequency > 1000Hz

In-traction Torque > 34.3mN.m (120Hz)

Self-positioning Torque > 34.3mN.m

Friction torque 600-1200 gf.cm

Pull in torque 300 gf.cm

Insulated resistance >  $10M\Omega$  (500V)

Insulated electricity power 600VAC/1mA/1s

Insulation grade A

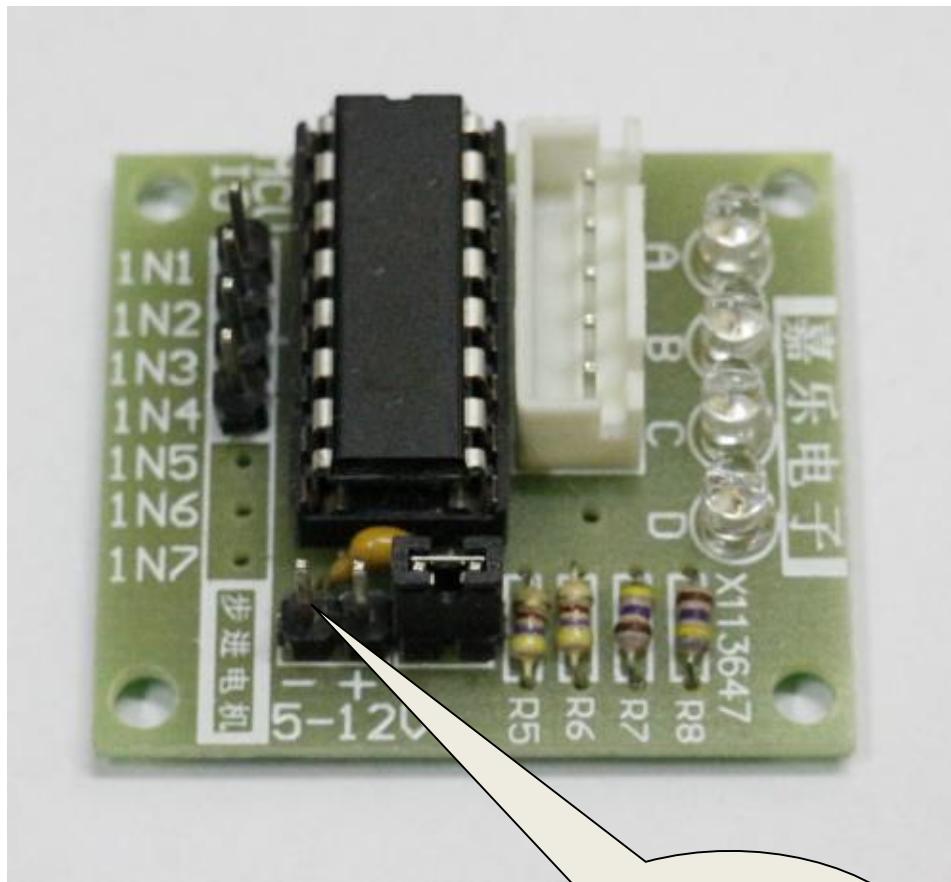
Rise in Temperature < 40K (120Hz)

Noise < 35dB (120Hz, No load, 10cm)

Model 28BYJ-48 – 5V

스테퍼 모터를 제어하는 ULN2003 제어 보드입니다.

핀 위치 참조 이미지입니다.



아두이노 연결:

| ULN2003 모듈 | 아두이노 우노 |
|------------|---------|
| IN1        | 8       |
| IN2        | 9       |
| IN3        | 10      |
| IN4        | 11      |
| +          | 5V 연결   |
| -          | GND     |

(-) GND  
(+) 5V  
우노와 연결합니다.

스테퍼 모터 사양은 보통 360 도(또는 제한된 각도) 회전에 대한 몇 회 분할을 할 수 있는 점이 중요합니다.

28BYJ-48 스테퍼모터의 사양으로 스피드와 스텝을 계산해 봅니다.

Speed Variation Ratio (속도 변화 비율): 1/64

Stride Angle 5.625°

Steps=Number of steps in One Revolution \* Gear ratio

Steps=1 회전 단계 수 \* 속도 변화 비율  
Steps=  $(360^\circ / 5.625^\circ) \times 64 = 64 \times 64 = 4096$  스텝입니다.

### 34.1 스케치 예제 코드 1

ULN2003 제어 드라이버와 바로 연결하여 (다이렉트 제어) 사용하는 예제 코드입니다.

[http://www.gameplusedu.com/pds/gpshop/kit\\_code/strongest/uln2003\\_stepper\\_ex\\_1.ino](http://www.gameplusedu.com/pds/gpshop/kit_code/strongest/uln2003_stepper_ex_1.ino)

```
/*
BYJ48 Stepper motor code
Connect :
IN1 -> D8
IN2 -> D9
IN3 -> D10
IN4 -> D11
VCC . 5V Prefer to use external 5V Source
Gnd
*/
#define IN1 8
#define IN2 9
#define IN3 10
#define IN4 11

int Steps = 0;
boolean Direction = true; //
unsigned long last_time;
unsigned long currentMillis ;
int steps_left=4095;
long time;
void setup()
{
  Serial.begin(115200);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
```

```

pinMode(IN4, OUTPUT);
// delay(1000);
}

void loop()
{
    while(steps_left>0)
    {
        currentMillis = micros();
        if(currentMillis-last_time)>=1000
        {
            stepper(1);
            time=time+micros()-last_time;
            last_time=micros();
            steps_left--;
        }
    } // while

    Serial.println(time);
    Serial.println("Wait...!");
    delay(2000);
    Direction=!Direction; // 반전.
    steps_left=4095;
}

void stepper(int xw)
{
    for (int x=0;x<xw;x++)
    {
        switch(Steps)
        {
            case 0:
                digitalWrite(IN1, LOW);
                digitalWrite(IN2, LOW);
                digitalWrite(IN3, LOW);
                digitalWrite(IN4, HIGH);
                break;
            case 1:
                digitalWrite(IN1, LOW);
                digitalWrite(IN2, LOW);

```

```
digitalWrite(IN3, HIGH);
digitalWrite(IN4, HIGH);
break;
case 2:
digitalWrite(IN1, LOW);
digitalWrite(IN2, LOW);
digitalWrite(IN3, HIGH);
digitalWrite(IN4, LOW);
break;
case 3:
digitalWrite(IN1, LOW);
digitalWrite(IN2, HIGH);
digitalWrite(IN3, HIGH);
digitalWrite(IN4, LOW);
break;
case 4:
digitalWrite(IN1, LOW);
digitalWrite(IN2, HIGH);
digitalWrite(IN3, LOW);
digitalWrite(IN4, LOW);
break;
case 5:
digitalWrite(IN1, HIGH);
digitalWrite(IN2, HIGH);
digitalWrite(IN3, LOW);
digitalWrite(IN4, LOW);
break;
case 6:
digitalWrite(IN1, HIGH);
digitalWrite(IN2, LOW);
digitalWrite(IN3, LOW);
digitalWrite(IN4, LOW);
break;
case 7:
digitalWrite(IN1, HIGH);
digitalWrite(IN2, LOW);
digitalWrite(IN3, LOW);
digitalWrite(IN4, HIGH);
break;
default:
```

```

        digitalWrite(IN1, LOW);
        digitalWrite(IN2, LOW);
        digitalWrite(IN3, LOW);
        digitalWrite(IN4, LOW);
        break;
    }

    SetDirection();
}
}

void SetDirection()
{
    if(Direction==1) { Steps++; }
    if(Direction==0) { Steps--; }
    if(Steps>7) { Steps=0; }
    if(Steps<0) { Steps=7; }
}

```

## 34.2 스케치 예제 코드 2

스케치 라이브러리 Stepper Motor Library 제어 코드입니다

```

/*
스케치 라이브러리 Stepper 를 사용합니다.
*/
#include <Stepper.h>

int in1Pin = 8;
int in2Pin = 9;
int in3Pin = 10;
int in4Pin = 11;

Stepper motor(64, in1Pin, in2Pin, in3Pin, in4Pin);

void setup()
{

```

```

pinMode(in1Pin, OUTPUT);
pinMode(in2Pin, OUTPUT);
pinMode(in3Pin, OUTPUT);
pinMode(in4Pin, OUTPUT);

// this line is for Leonardo's, it delays the serial interface
// until the terminal window is opened
// while (!Serial);

Serial.begin(9600);
motor.setSpeed(30); // 30 rpm.
}

void loop()
{
    if (Serial.available())
    {
        int steps = Serial.parseInt();
        motor.step(steps);
    }
}

```

위의 코드는 시리얼로부터 정수를 받아 클래스 변수 motor 의 함수 step() 호출 해 주는 예제입니다.

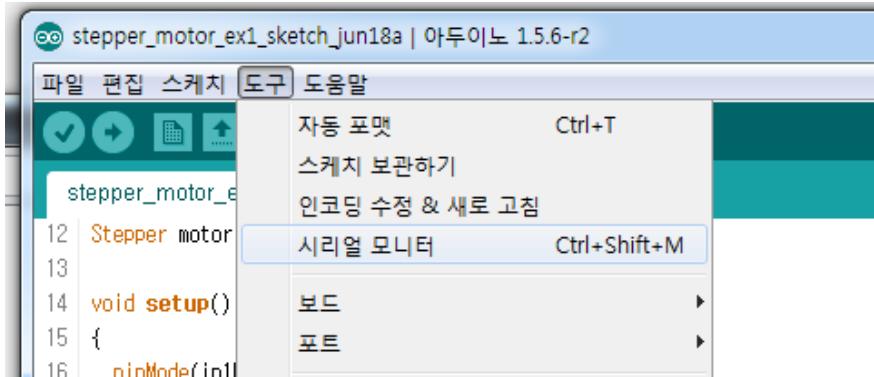
위의 코드를 스케치에서 업로드 후 시리얼 모니터 창을 열고 64, 128, 256 등의 숫자를 입력해 봅니다. 입력 후 스텝 모터 회전 확인 바랍니다.

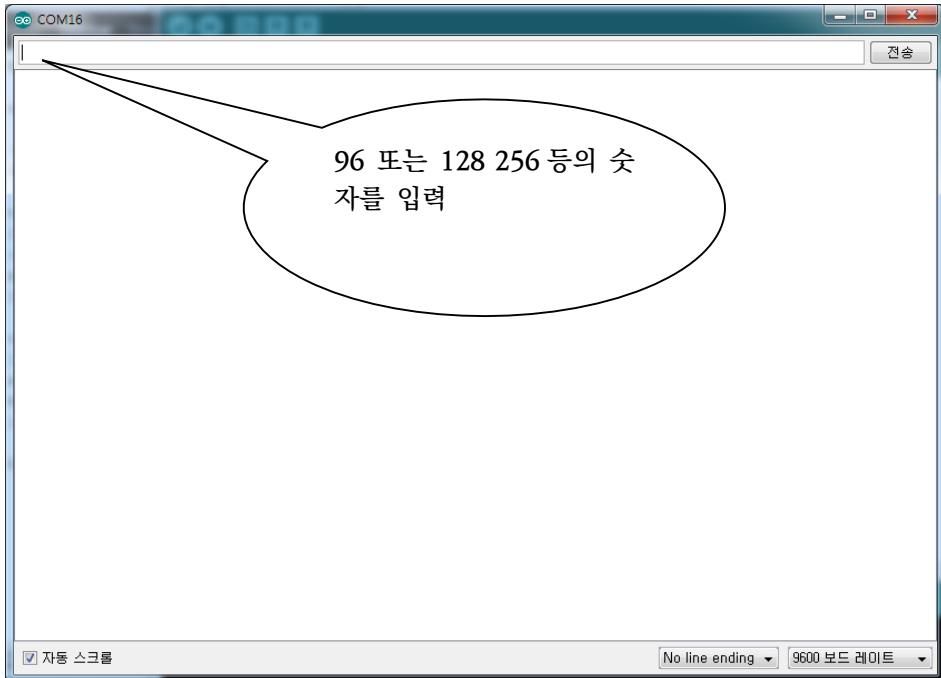
시리얼 입력 후, 다시 2초 3초 후에 적당한 입력 값을 입력 하도록 합니다.

```
stepper_motor_ex1_sketch_jun18a | 아두이노 1.5.6-r2
파일 편집 스케치 도구 도움말
stepper_motor_ex1_sketch_jun18a §
12 Stepper motor(512, in1Pin, in2Pin, in3Pin, in4Pin);
13
14 void setup()
15 {
16   pinMode(in1Pin, OUTPUT);
17   pinMode(in2Pin, OUTPUT);
18   pinMode(in3Pin, OUTPUT);
19   pinMode(in4Pin, OUTPUT);
20
21 // this line is for Leonardo's, it delays the serial interface
22 // until the terminal window is opened
23 while (!Serial);
24
25 Serial.begin(9600);
26 motor.setSpeed(20);
27 }
28
29 void loop()
30 {
31   if (Serial.available())
32   {
33     int steps = Serial.parseInt();
34     motor.step(steps);
35   }
}
업로드 완료
avrdude done. Thank you.

26
Arduino Uno on COM18
```

시리얼 모니터 창을 엽니다.





스테퍼 모터가 지정된 스텝만큼 회전합니다.

스테퍼 모터 스피드 지정과 스텝 회전에 대한 이해가 필요합니다.

모터 스피드는 위의 함수 `motor.setSpeed(20);` 사용 할 경우 함수의 파라미터 20이라는 의미는 RPM입니다. Revolutions Per Minute (1 분당 회전수)입니다. 즉 1분에 20 번 회전할 수 있는 속도를 의미합니다.

## 35 > RED, GREEN, YELLOW LED

LED는 전자/전기에서 가장 많이 사용되는 부품중의 하나입니다.

작동 표시, 상태 표시, 기타 디버깅 용도 등의 많은 목적으로 사용되고 있습니다. 그에 따라 LED는 여러 형태와 크기를 가지고 있습니다.

여기서 사용되는 LED는 5mm 직경을 가진 LED입니다.

빨간색, 노란색, 초록색, 용도에 따라 다른 색상을 구하여 사용할 수 있습니다.

아두이노 보드의 5V 전류를 사용하는 경우 LED (+) 리드선에 220~300 ohm 정도의 저항을 연결합니다.

## 36 LED 종류와 전압 사용 예제

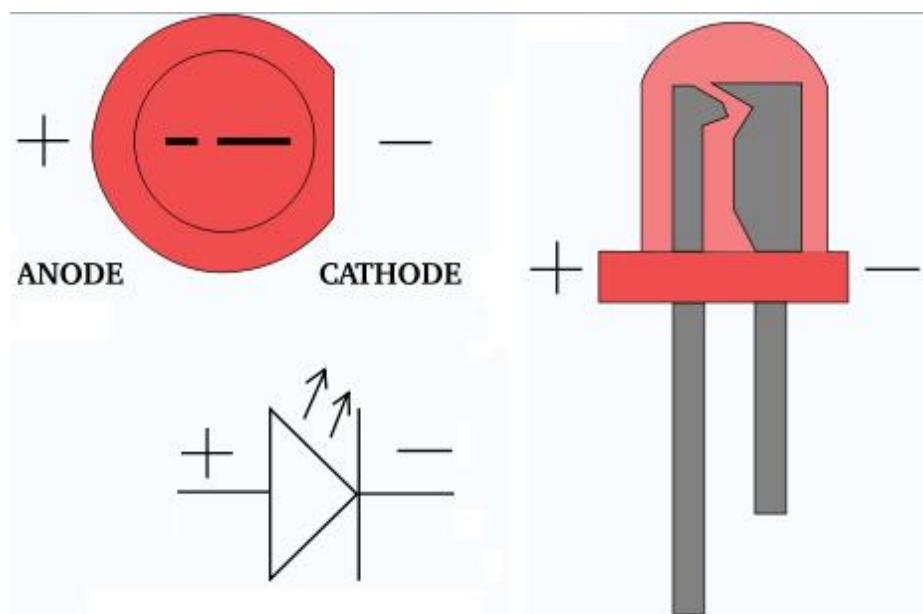


그림 36-1 LED 부품 설명 단면도

5파이 크기의 LED는 보통 아래와 같은 전압, 전류를 사용합니다.

| 색상     | 구 분           | 최소전압 | 최대전압 | 전류(일반) | 전류(최대) |
|--------|---------------|------|------|--------|--------|
| 적색●    | Red           | 1.8V | 2.3V | 20 mA  | 50 mA  |
| 오렌지●   | Orange        | 2.0V | 2.3V | 30 mA  | 50 mA  |
| 황색●    | Real Yellow   | 2.0V | 2.8V | 20 mA  | 50 mA  |
| 진한초록●  | Emerald Green | 1.8V | 2.3V | 20 mA  | 50 mA  |
| 초록●    | Real Green    | 3.0V | 3.6V | 20 mA  | 50 mA  |
| 밝은 청색● | sky Blue      | 3.4V | 3.8V | 20 mA  | 50 mA  |
| 청색●    | Real Blue     | 3.4V | 3.8V | 20 mA  | 50 mA  |
| 핑크●    | Pink          | 3.4V | 3.8V | 20 mA  | 50 mA  |
| 백색○    | White         | 3.4V | 4.0V | 20 mA  | 50 mA  |

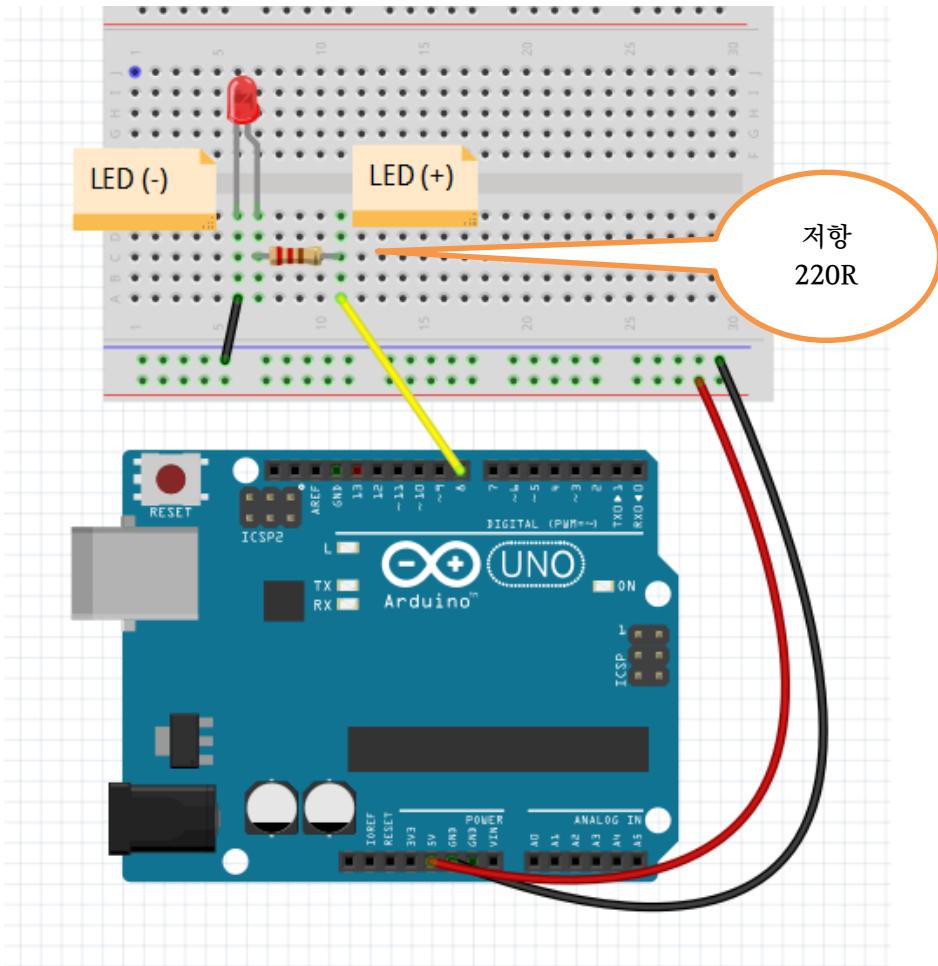


그림 36-2 LED 연결 브레드보드 회로도

예제 코드:

```

int ledPin=8; //set IO pin of LED in control

void setup()
{
    pinMode(ledPin,OUTPUT); //set digital pin IO is OUTPUT
}

void loop()
{
    digitalWrite(ledPin,HIGH); //set PIN8 is HIGH , about 5V
    delay(1000); //delay 1000ms, 1000ms = 1s
}

```

```

digitalWrite(ledPin,LOW); //set PIN8 is LOW, 0V
delay(1000); //delay 1000ms, 1000ms = 1s
}

```

## 37 저항 기초 정보

저항은 전류의 세기를 제어하는 소자입니다.

상식적으로 저항의 크기가 클수록, 저항 값이 커질수록 전류를 그 크기만큼 억제하는 역할을 합니다.

간단한 부품이지만, 전자, 전기에서는 항상 사용되고 있습니다. 각종 부품들의 전류의 흐름을 제어하기 위해서는 필수입니다.

| 색명<br>(컬러 코드) | 제 1 색대 | 제 2 색대 | 제 3 색대     | 제 4 색대             |
|---------------|--------|--------|------------|--------------------|
|               | 제 1 숫자 | 제 2 숫자 | 승수         | 공칭 저항값 허용<br>치(오차) |
| 검은색           | 0      | 0      | 1          |                    |
| 갈색            | 1      | 1      | 10         | 1%                 |
| 적색            | 2      | 2      | 100        | 2%                 |
| 주황색           | 3      | 3      | 1000       |                    |
| 노랑색           | 4      | 4      | 10000      |                    |
| 녹색            | 5      | 5      | 100000     | 0.5% *             |
| 청색            | 6      | 6      | 1000000    |                    |
| 보라색           | 7      | 7      | 10000000   |                    |
| 회색            | 8      | 8      | 100000000  |                    |
| 흰색            | 9      | 9      | 1000000000 |                    |
| 금색            |        |        | 0.1        | 5%                 |
| 은색            |        |        | 0.01       | 10%                |
| 무색            |        |        |            | 20%                |

› 1k Ohm resistor x 10

## 38 1K 저항 10 개

› 1K ohm resistor x 10

## 39 10K 저항 10 개

› 10K ohm resistor x 10

## 40 220 OHM 저항 10 개

› 220 ohm resistor x 10

## 41 부저의 종류

부저(Buzzer, 부저, 베저)는 음향을 출력하는 부품입니다. 전자기기에서의 경고음, 알람, 멜로디 등의 소리를 내기 위해 주로 사용되고 있습니다. 부저의 형태 및 성능만 조금 다를 뿐 수많은 전자기기에 사용되고 있습니다.

부저의 제작 방식에 따라 아래와 같은 2 가지의 항목이 있습니다.

### » 마그네틱 부저 (MAGNETIC BUZZER)

원리는 마그네틱 스피커와 같으나 작은 진동판을 붙인 아주 간단한 모양으로 만들기 때문에 음성 출력용으로는 쓰이지 않고 전자기기 내부에서 발생하는 단속음 등 음향 신호 출력용으로 주로 사용 됩니다.

내부에는 이어폰 전자코일과 자석, 진동판 등으로 구성되어 있습니다. 이어폰은 작은 스피커와 같습니다. 마그네틱 부저도 같은 구성 원리로 되어 있습니다.

### » 피에조 부저 (PIEZOELECTRIC BUZZER)

Piezoelectric Buzzer. 압박 부저입니다.

압전기 효과를 이용한 것으로 작은 전기신호로 큰 음량을 낼 수 있기 때문에 전자시계의 알람 멜로디나 전화기의 벨소리 등을 출력하는데 사용 됩니다.

작동 방식에 따라 위의 2 종류의 부저를 다른 용어로도 표시됩니다.

### 41.1 패시브 부저

Passive Buzzer, 수동 부저. 마그네틱 부저

패시브 부저는 전기가 항상 흐르는 상태입니다. 전류의 흐름으로 주파수 파형을 만들어 소리를 내도록 되어 있습니다. 일반적인 스피커와 같은 작동 원리입니다.

### 41.2 액티브 부저

Active Buzzer, Piezo Buzzer.

액티브 부저는 전기가 흐르는 경우에만 소리가 납니다. 전류가 흐르게 되면 압전 효과가 발생되어서 소리가 나도록 되어 있습니다. 전류가 흐르는 경우에만 소리가 나서 액티브 부저라고도 합니다.

보통 액티브 부저는 위에 스티커로 양극/음극 표시가 되어 있습니다.

그리고 부저의 하부는 고무로 덮여 있는 모양입니다.

패시브 부저는 PCB 형태의 기판이 보이게 되어 있습니다.

## 42 > PASSIVE BUZZER X 1

수동 부저입니다. 마그네틱 부저입니다.

경고, 간단한 멜로디 등의 소리를 낼 수 있는 모듈입니다. 5V 사용됩니다.



그림 42-1 Passive Buzzer, 수동 부저

부저 하단부의 2 개의 리드선 극성을 구분하여 연결하여 줍니다.

(+) 표시된 리드선과 나머지는 (-) 리드선 입니다.

내부 구성된 원리는 작은 이어폰과 거의 동일합니다. 전자코일과 자석, 진동판의 구조로 되어 있습니다. 용도에 따라 다양하게 제조, 생산되는 부품중의 하나입니다.

전류의 흐름으로 주파수 파형을 만들어서 진동하면서 소리를 내도록 되어 있습니다.

아두이노 우노 보드/MCU 보드 등에서 사용하는 경우에는 PWM 가능한 포트에 연결해야 정확히 원하는 소리를 낼 수 있습니다.

정확히 음계가 구분되는 멜로디, 알람 소리 등에 많이 사용되고 있습니다.

부저의 리드선이 연결된 부분은 PCB 형태를 가지고 있습니다.



그림 42-2 패시브 부저의 밑면

아두이노 연결:

| 부저  | 아두이노 보드 |
|-----|---------|
| (-) | GND     |
| (+) | PWM 포트  |

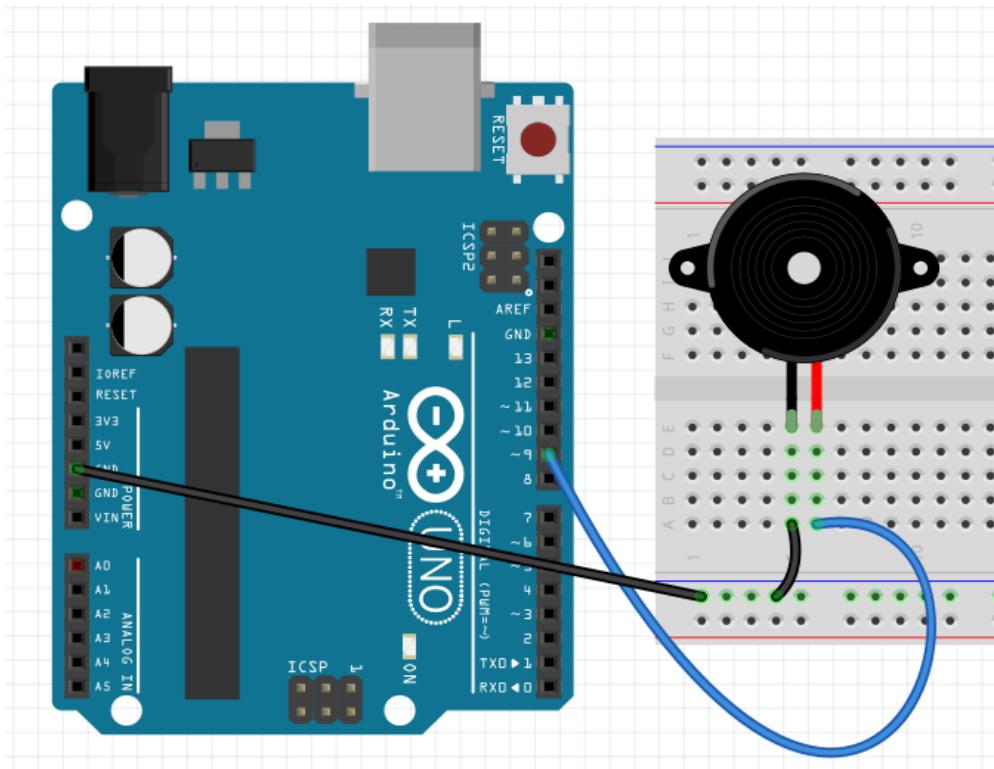


그림 42-3 브레드보드 회로 구성도

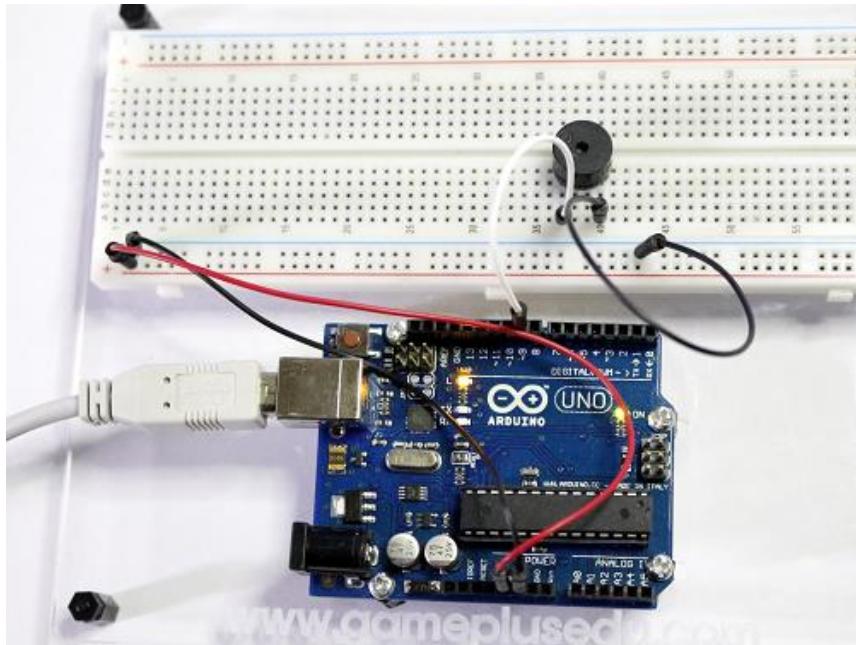


그림 42-4 우노 R3 거치대와 구성된 모습

(예제코드: 사이렌 소리)

[http://www.gameplusedu.com/pds/gpshop/kit\\_code/strongest/passive\\_siren\\_ex\\_1.ino](http://www.gameplusedu.com/pds/gpshop/kit_code/strongest/passive_siren_ex_1.ino)

```
/* 사이렌 소리 예제 코드입니다. */
void setup()
{
    pinMode(9,OUTPUT);
}

void loop()
{
    //Dynamic to set the frequency from 200HZ to 800HZ in a cycle
    for(int i=200;i<=800;i++)
    {
        tone(9,i);      //Output the frequency in port 4
        delay(5);       //Keep this frequency 5ms
    }
    delay(4000);      //Keep 4mins in the Highest frequency
    for(int i=800;i>=200;i--)
    {
        tone(9,i);
        delay(10); //Keep this frequency 10ms
    }
}
```

(예제코드: 아두이노 PlayTone)

[http://www.gameplusedu.com/pds/gpshop/kit\\_code/strongest/passive\\_abc\\_song\\_ex\\_1.ino](http://www.gameplusedu.com/pds/gpshop/kit_code/strongest/passive_abc_song_ex_1.ino)

ABC 알파벳 송입니다.

```
int speakerPin = 9;

int length = 15; // the number of notes
char notes[] = "ccggaagffeeddc "; // a space represents a rest
int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 4 };
int tempo = 300;

void playTone(int tone, int duration)
{
    for (long i = 0; i < duration * 1000L; i += tone * 2)
    {
        digitalWrite(speakerPin, HIGH);
        delayMicroseconds(tone);
        digitalWrite(speakerPin, LOW);
        delayMicroseconds(tone);
    }
}

void playNote(char note, int duration)
{
    char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C' };
    int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136, 1014, 956 };

    // play the tone corresponding to the note name
    for (int i = 0; i < 8; i++)
    {
        if (names[i] == note)
        {
            playTone(tones[i], duration);
        }
    }
}

void setup()
{
```

```
pinMode(speakerPin, OUTPUT);
}

void loop()
{
    for (int i = 0; i < length; i++)
    {
        if (notes[i] == ' ')
        {
            delay(beats[i] * tempo); // rest
        }
        else
        {
            playNote(notes[i], beats[i] * tempo);
        }
        // pause between notes
        delay(tempo / 2);
    } // end for loop
}
```

## 43 › PIEZO BUZZER X 1

피에조 부저입니다. 압박 부저, 액티브 부저라고도 합니다..



그림 43-1 피에조 부저

전류가 통하는 상태에서만 소리가 나서 액티브 부저입니다.

피에조 부저입니다. 5V 사용됩니다. 피에조 뜻은 압박이라는 의미입니다.

부저의 밑면이 딱딱한 고무 재질로 덮여 있으면 피에조 부저입니다.

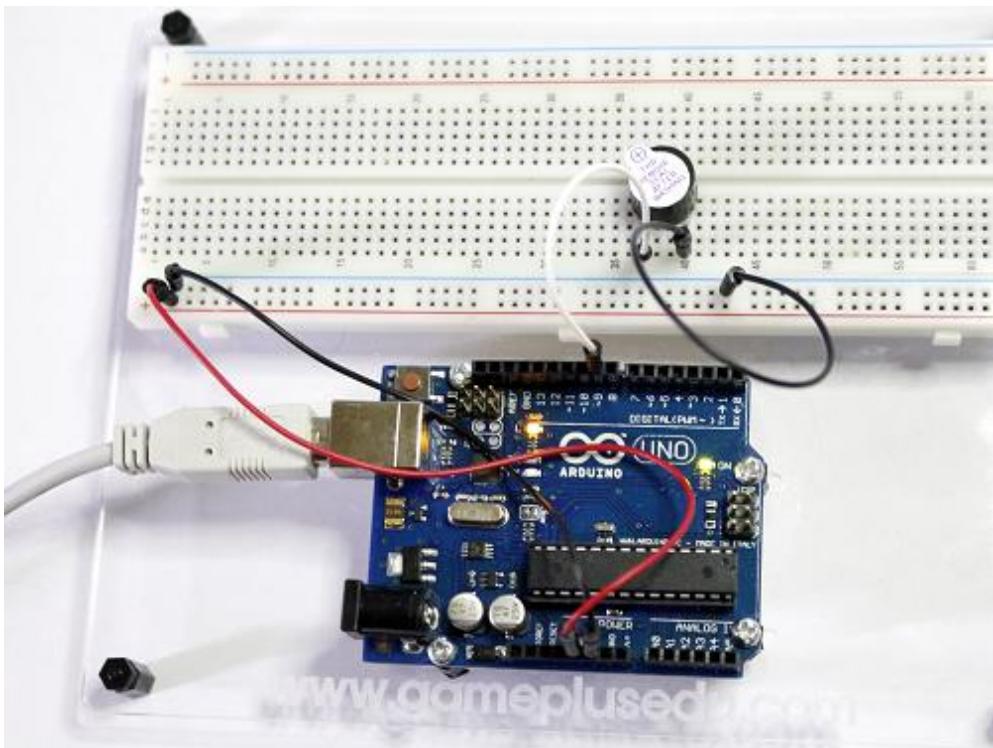
부저 하단부의 2 개의 리드선 극성을 구분하여 연결하여 줍니다.

(+) 표시된 리드선과 나머지는 (-) 리드선입니다.

작동 방법은 + 리드선 High 일 경우에는 빽~ 하는 소리, Low 일 경우에는 소리가 안 납니다. 즉, 전류가 흐를 때만 소리가 나게 되어 있습니다.

아두이노 연결:

| 부저  | 아두이노 보드 |
|-----|---------|
| (-) | GND     |
| (+) | D9      |



아두이노 예제코드:

```
int buzzer=9; //  
  
void setup() {  
    pinMode(buzzer, OUTPUT);  
}  
  
void loop() {  
    digitalWrite(buzzer,HIGH); // 소리 켜기  
    delay(500);  
    digitalWrite(buzzer,LOW); // 소리 끄기  
    delay(500);  
}
```

## 44 > KEY MODULE (WITH HAT) X 4

Tact Switch 12x12 mm 입니다.

Tact 는 Tactile 의 약자입니다. 접촉 스위치, 접촉 버튼으로 번역할 수 있습니다..

노란색 캡이 있어서 버튼 키 위에 덮어 주면 완성 됩니다.

물론 노란색 캡이 없어도 버튼 기능은 됩니다.

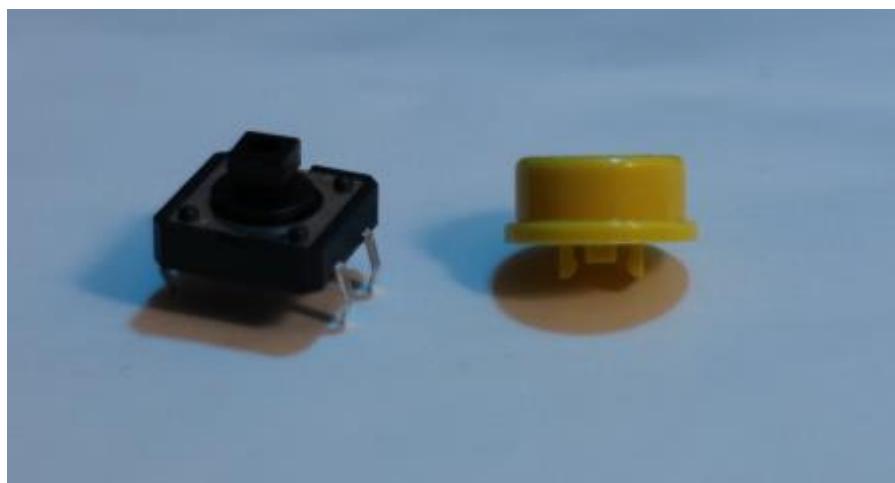


그림 44-1 택 스위치 12x12mm & 캡

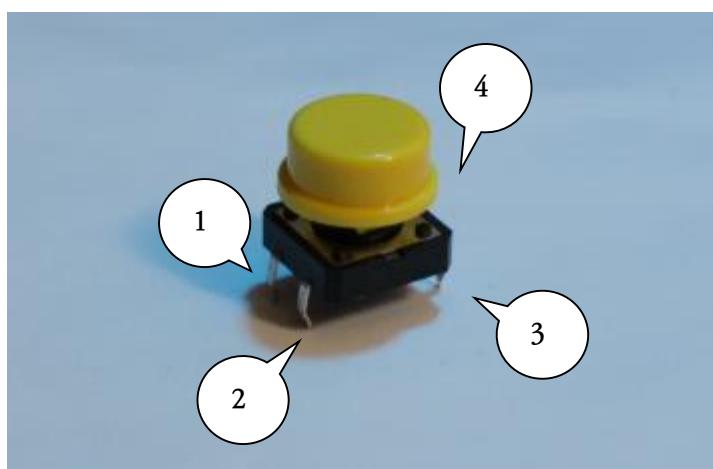


그림 44-2 택 스위치 핀 번호

Tact Switch (or Tact Button)은 여러 용도로 사용 되고 있습니다.

기초 전기/전자 부품이면서, 크기, 형태만 약간 다를 수많은 가전/산업분야의 전자 제품, 전자 기기에 많이 사용됩니다.

버튼의 내부는 아래와 같은 구조입니다.

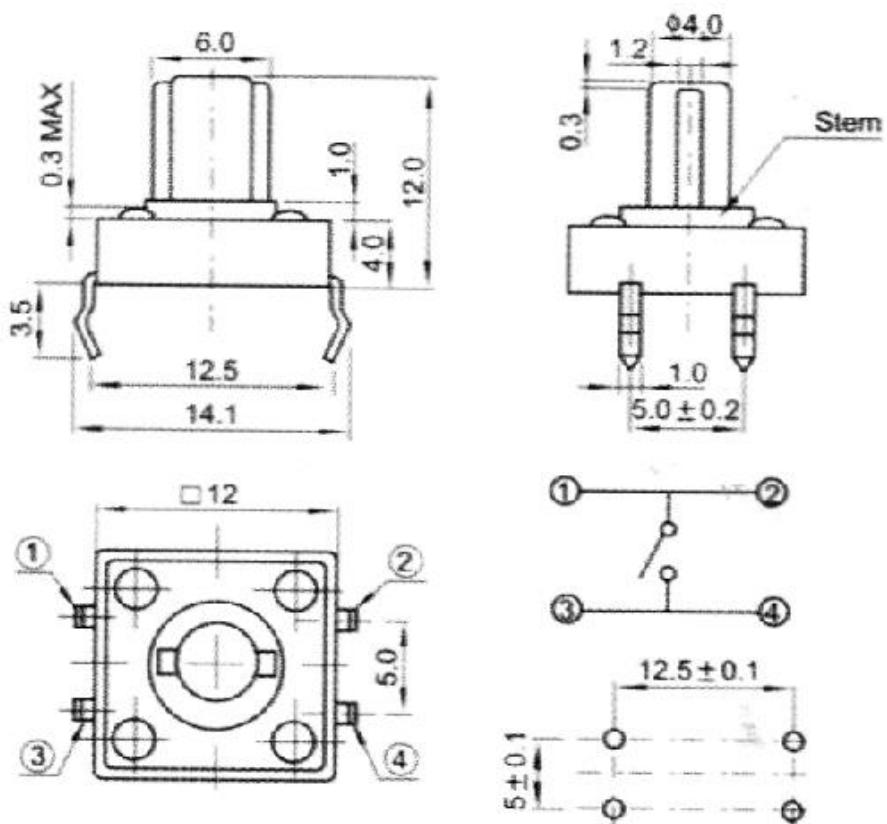


그림 44-3 택 스위치 회로도

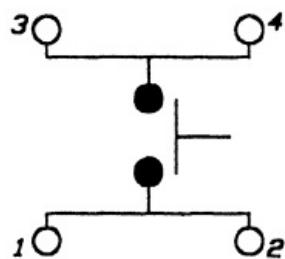


그림 44-4 택 스위치 핀 번호

아래의 예제는 버튼이 눌리면 LED 켜는 예제입니다.

Pull Down 예제입니다.

예제코드는 버튼이 눌려진 상태인 경우에만 13 번 포트에 연결된 LED 켜는 예제입니다.

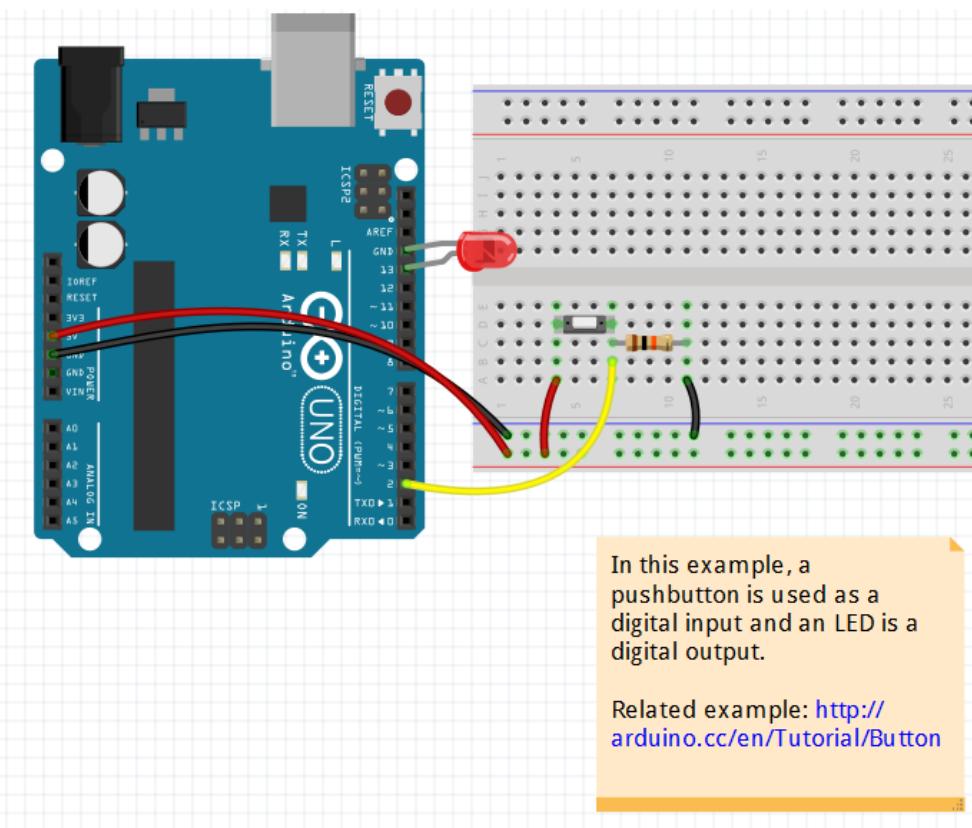


그림 44-5 브레드보드 회로도

```
// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 2; // the number of the pushbutton pin
const int ledPin = 13; // the number of the LED pin

// variables will change:
int buttonState = 0; // variable for reading the pushbutton status

void setup() {
    // initialize the LED pin as an output:
    pinMode(ledPin, OUTPUT);
    // initialize the pushbutton pin as an input:
    pinMode(buttonPin, INPUT);
}
```

```
void loop(){
    // read the state of the pushbutton value:
    buttonState = digitalRead(buttonPin);

    // check if the pushbutton is pressed.
    // if it is, the buttonState is HIGH:
    if (buttonState == HIGH) {
        // turn LED on:
        digitalWrite(ledPin, HIGH);
    }
    else {
        // turn LED off:
        digitalWrite(ledPin, LOW);
    }
}
```

## 45 PULL-UP 풀업 저항, PULL-DOWN 풀다운 저항

### 45.1 플로팅(FLOATING)

아두이노 보드의 디지털/아날로그 핀에 입력을 5V 또는 0V를 입력 할 수 있습니다. 모든 MCU는 포트를 가지고 있습니다. 디지털/아날로그 입/출력 되는 포트입니다. MCU의 핀 방향을 인풋으로 설정한 후, 입력을 하지 않는 경우는 MCU 포트 자체에서는 5V를 입력하였는지 1V, 0V를 입력하였는지 구분이 안됩니다.

이런 문제에 의하여 오작동이 발생할 수 있습니다. 이런 상태를 플로팅 현상이라고 합니다. 일종의 잡음과 비슷합니다. Floating의 단어적 의미도 “허공에 뜨다”입니다. 아두이노 보드에 아무것도 연결하지 않은 상태에서 digitalRead, analogRead 함수를 사용하여 읽어 들인 값을 Serial 포트로 print() 해보면 증상 확인 가능합니다. 신호를 주지 않아도 HIGH, LOW, 아날로그인 포트인 경우 의미 없는 값들이 보입니다. 이런 증상을 없애기 위해 풀업/풀다운 저항을 사용하여 0V 또는 5V(정확히는 HIGH MCU AREF 기준전압)로 묶어 놓게 됩니다.

즉, 풀업/풀다운 저항 회로는 마이크로프로세서 핀의 상태(State)를 명확히 하는 목적으로 활용합니다. 보통 저항의 크기는 4.7k~100K Ω(Ohm)이 많이 사용됩니다.

아두이노 보드에 아무것도 연결하지 않은 상태에서

아래의 간단한 코드를 입력, 업로드 해봅니다.

이상한 값이 넘어오게 되어 있습니다. 미세한 전류의 변화에도 입력되는 값은 민감하게 변하게 되어 있습니다.

```
void setup(void)
{
    Serial.begin(9600);
    pinMode(2,INPUT);
}

void loop(void)
{
    Serial.println(digitalRead(2)); // or Serial.println(analogRead(0))
}
```

시리얼 모니터 로그를 살펴보면 아무것도 연결되지 않은 상태면 0, LOW 값이어야 하지만, 변동이 있습니다. 동일하게 아날로그 포트의 입력 값을 읽어 보면 1023 숫자 까지의 랜덤하고 규칙이 없는 형태로 나옵니다.

스위치 버튼을 회로 구성없이 D2 포트에 연결하면 스위치가 On (닫힌) 상태인 경우 원하는 HIGH, LOW 등의 값을 읽을 수 있습니다. 스위치가 Off(열린) 상태인 경우 아무것도 연결되지 않은 상태와 동일합니다. 그럼 스위치가 On 상태인지, Off 상태인

지 구분을 못하게 됩니다. 그래서 풀업, 풀다운 회로 구성이 필요하게 됩니다. 그리고 On 상태의 높은 전류는 입력 포트에 블랙 현상(망가짐) 있을 수 있습니다.

## 45.2 풀업(PULL-UP) 저항

풀업은 입력 핀을(포트를) VCC(5V) 입력 상태로 묶어두는 것이 pull-up입니다. 즉, 풀업 저항으로 구성된 상태는 기본값은 HIGH입니다. 해당 포트에 5V를 항상 입력해주고 작동이 일어날 경우 0V로 만들어주게 됩니다.

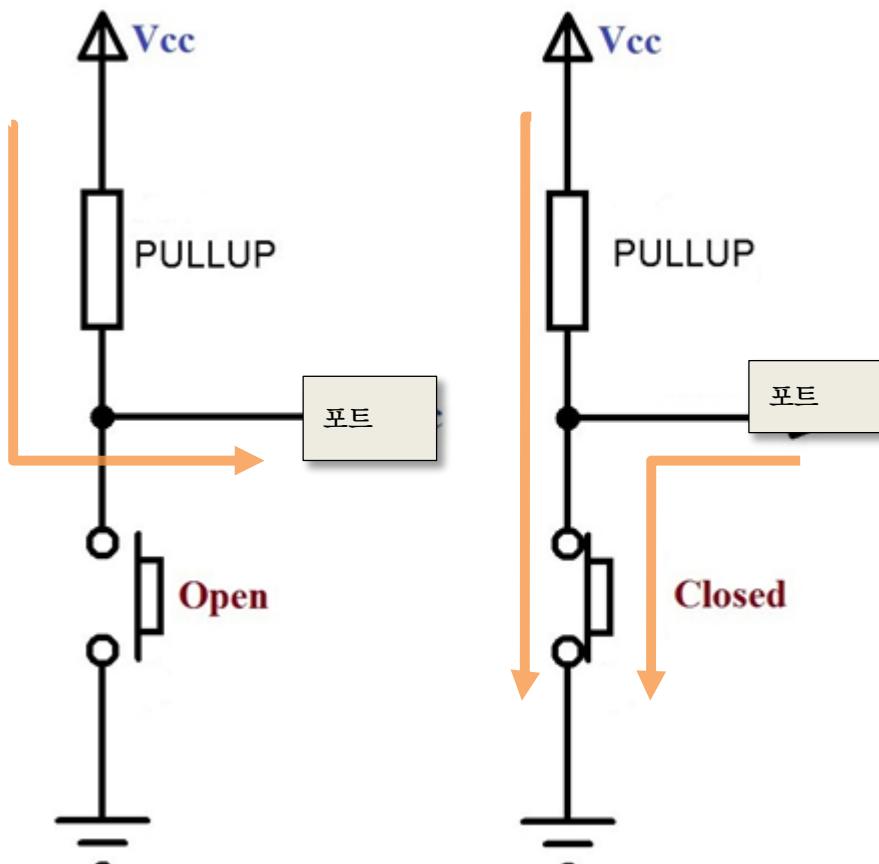


그림 45-1 풀업 회로 구성도

풀업 상태에서는 스위치가 열려 있는 경우 VCC 5V 전압이 들어갑니다. 스위치를 닫는 경우에는 그라운드 쪽으로 전류가 흐르게 되며 회로의 전압은 0V가 됩니다. 그라운드 방향에는 온/오프 스위치, (+) 방향에는 풀업 저항을 연결 하도록 구성되어 있습니다. 풀업이라는 말은 HIGH 신호로 묶어 놓는 의미도 있지만 (+) 양극 방향의 연결에 저항을 연결한다는 의미도 내포하고 있습니다.

대부분 의문 사항은 왜 스위치를 닫았는데(연결된 상태) MCU 입력포트에서는 0V 가 되는 건가 의아해 할 수 있습니다.

논리는 간단합니다.

전류의 흐름은 물의 흐름과 동일 하다고 이해하면 됩니다.

전류의 흐름은 높은 곳에서 낮은 곳으로 흘르게 되어 있습니다. 압력이 높은 곳에서 낮은 곳으로 흘르게 되어 있습니다.

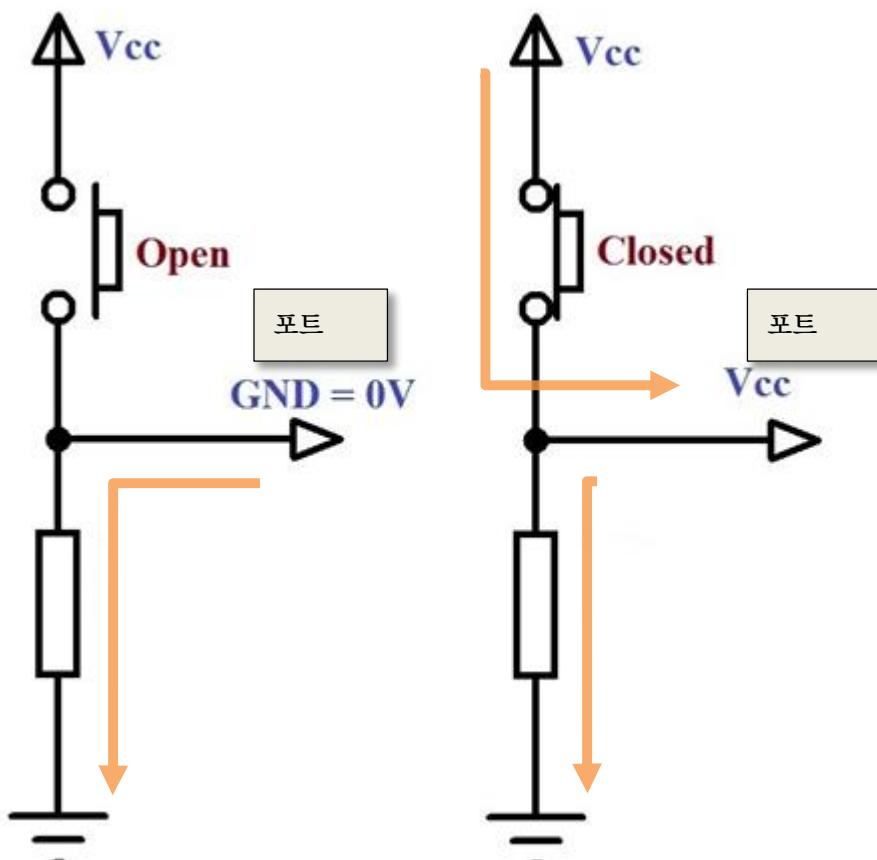
### 45.3 풀다운(PULL-DOWN) 저항

풀다운 GND (0V) 입력 상태로 묶어두는 것이 pull-down 저항입니다.

풀다운 상태의 포트는 0V, LOW 상태에서 5V 입력을 받으면 1, HIGH 상태로 됩니다.

스위치가 열려 있는 상태이므로 그라운드와 연결 되어 항상 0V 입니다.

저항의 연결 위치는 GND 방향에 연결해야 합니다.



## 46 풀업, 풀다운 저항 계산 논리.

풀업, 풀다운 회로 구성하는 경우 반드시 들어가야 할 부품이 저항입니다.

풀-업(Pull Up)은 HIGH 신호를 포트로 넣어주는 겁니다.

풀-다운(Pull Down)은 LOW 신호를 포트로 넣어주는 겁니다.

주의사항은 회로를 구성하기 전에 반드시 사용하는 전압과 전류의 크기를 알아야 저항을 사용할 수 있습니다.

사용되는 저항의 크기는 회로의 구성 부품 및 사용 전원에 따라 되도록 정확한 계산 하에 사용되어야 합니다.

풀업 저항 구성 시 저항 계산은 옴의 법칙(Ohm's law) 공식을 사용하면 사용 가능한 저항의 크기를 구할 수 있습니다.

$$V = I \times R$$

(전압 = 전류 x 저항)

아두이노 보드에서의 풀업 저항은 5K, 4.9K, 10K 크기 등을 사용할 수 있습니다.

가장 많이 소개되는 5K 저항(4.9K 저항 포함)입니다.

5K 저항을 쓰면 옴의 법칙에 의해 1mA 정도의 전류가 흐를 수 있습니다.

$$R=V/I$$

5K = 5 / 0.001 mA 간단한 식에 의해 5K (5,000) 됩니다.

참고로 1A 는 1,000 mA 입니다.

또는 4.9K 사용하여도 무방합니다.

여기서 의문 사항은 왜 1mA 흘려 보내주어야 하는 것입니다. 2mA 정도의 전류를 흘려 보내주어도 풀업 저항 회로도를 구성할 수 있습니다. 그럼 저항의 크기는 변합니다. 500 R 저항을 사용해야 됩니다. 그래도 풀업 회로는 구성될 수 있습니다.

보통 MCU 포트의 논리 입력에서의 HIGH 판단은 1mA, 0.1mA 의 전류가 흐르던 HIGH로 인식합니다. 미량의 전류가 입력되면 HIGH로 인식될 수 있습니다.

물론 10mA, 20mA 입력되어도 HIGH 신호로 인식하게 되어 있습니다. 너무 높은 전류를 입력해주면 당연히 MCU 입력 포트는 망가질 수 있습니다.

아두이노 우노 R3 보드는 디지털, 아날로그 포트의 출력이 최대 5V 40mA 입니다.

즉, D13 번 포트를 OUTPUT 방향 설정 후 digitalWrite(13, HIGH); 코드 되어 업로드 되었을 경우 D13 번 포트의 전압과 전류는 5V, 40mA 정도 나오게 되어 있습니다.

아두이노 보드에서 사용되는 외부 입출력 가능 포트의 전체 개수는 14 + 6 = 20 개 정도입니다. 그래서 보통 20개의 입출력을 사용 하는 경우 40 mA 나누기 20을 해주면 2mA 의 전류의 크기가 무난하다고 간주 할 수 있습니다.

즉, 40mA / 20 = 2 mA, 포트당 2mA 정도, MCU 포트에 대한 신호 처리를 해주면 안정적인 신호 입/출력이 가능하다고 보여집니다.

아두이노 보드의 해당 포트에 풀업 저항 10K 를 쓰는 경우에는 해당 포트에 0.0005mA 정도의 전류가 흘러 들어가게 됩니다. 너무 큰 풀업 저항을 사용하는 경우에는 HIGH로 인식은 되겠지만, 중간에 연결되는 전선, 회로 소재 또한 미세하나마 저항으로 작동되고 있습니다. 그럼 불안정한 신호로 오차가 생길 수도 있습니다.

그래서 5K 저항을 사용하여 1mA 전류를 흘려 보냅니다.

그럼 풀다운 저항의 크기 계산도 동일한 원칙입니다. 10K 정도를 사용하여 확실히 0V로 인식하게 하도록 합니다. 풀다운은 그라운드 방향에 저항을 사용합니다.

풀다운 VCC 신호 입력에서의 전압, 전류의 흐름이 열린 상태에서는 그라운드 방향과 연결된 상태로 됩니다. MCU 의 입력 포트에는 0V 입니다. 저항을 연결하지 안아도 물론 0V 입니다. 그럼 풀다운 회로에서 입력 부분의 스위치를 연결합니다.

스위치 ON 상태입니다. 그럼 VCC 입력 되면서 전류가 흐르게 됩니다.

그럼 전류의 흐름은 높은 곳에서 낮은 곳으로 흐르게 되어 있습니다.

MCU 의 입력 포트, GND 방향이 있습니다. 전류는 GND 방향으로 흘러 들어가는 결과가 됩니다. MCU 의 입력 포트에는 전류가 흘러 들어가지 않습니다. 그럼 그라운드 방향에 저항을 연결하는 경우 전류는 저항의 방해를 받아 MCU 의 입력 포트로 흘러 가게 됩니다.

그럼 이제 전압과 전류, 저항 공식을 적용하여 무슨 저항을 사용할지 계산하여 보도록 합니다.

먼저, 전압(V: Voltage)과 암페어(A: Amp) 공식을 이해하면 됩니다.

### 저항 사용:

전압: 전압은 V 기호를 사용합니다. 단위는 V (Voltage, 볼티지) 입니다.

I는 전류입니다. 임피던스, 전류입니다. 단위는 A (Ampere, 암페어) 입니다.

저항의 단위는 옴(Ohm), 기호로  $\Omega$  표시합니다.

위의 표시된 계산공식은 “옴의 법칙”이라고 합니다.  $V=IR$ ,  $I=V/R$ ,  $R=V/I$  입니다.

전류는 전압에 비례하고 저항에 반비례합니다. 전압은 저항에 반비례합니다.

그럼 간단하게 1V 인 경우의 전류(암페어)와 저항 옴 값을 알아봅니다.

$1V = 1A \times 1\Omega$  입니다. 1V 는 1A 와 1(Ohm)  $\Omega$  저항을 사용합니다.

그럼  $1V = ?A \times 2R$  인 경우는  $A=V/R$  ( $0.5=1/2$ ),  $1V = 0.5A \times 2R$  이 됩니다.

1V 에 2 Ohm 의 저항을 사용하면 0.5 암페어가 흐르게 됩니다.

$V = I \times R$  공식에서 V 와 R 값을 아는 경우 암페어 값을 구한다면 아래와 같은 식이 유도됩니다.

$$I = V/R \text{ (전류=전압/저항값)}$$

반대로,  $1V = 0.5A \times ?R$  인 경우에는, 즉 전압과 암페어 값을 알고 있을 경우 사용되는 저항의 크기는 아래와 같이 유도 됩니다.

$$R = V / I \quad (2\text{Ohm} = 1V / 0.5A)$$

$$R = V / I$$

(저항 = 전압 / 전류)

$5/0.04=125$  값이 나오게 됩니다. 그럼, 저항 125 Ohm 정도 사용하는 경우 5V 전체가 다 걸려 버립니다.

풀업에는 1mA 정도의 전류를 흘려 보내주기 위해 5K 를 사용합니다.

$$5K = 5/0.001 A$$

풀다운 회로 구성에는 1mA 이하의 전류를 사용하기 위해 10K 정도의 저항을 사용합니다.

$$10K = 5/0.0005 A$$

## 46.1 풀다운 버튼 예제 (PULL DOWN BUTTON)

풀다운 회로의 버튼을 구성하여 평소에는 0, 버튼을 누르면 1 로 만들어 보도록 합니다. 풀다운 회로에 필요한 저항의 크기도 계산하여 회로를 구성하도록 합니다. 아두이노의 공식 사이트 또는 여러 블로그에 소개되는 내용들은 풀다운 버튼 예제들에는 10K 저항이 사용되고 있습니다.

풀다운, 풀업 공통 사용 예제 코드:

디지털 D2 번 포트의 신호가 HIGH인 경우 D13 번 포트에 연결된 LED에 HIGH 신호를 줍니다. 반대인 경우 LOW 신호를 주도록 되어 있습니다.

```
// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 2;      // the number of the pushbutton pin
const int ledPin = 13;        // the number of the LED pin

// variables will change:
int buttonState = 0;          // variable for reading the pushbutton
status

void setup() {
```

```
// initialize the LED pin as an output:  
pinMode(ledPin, OUTPUT);  
// initialize the pushbutton pin as an input:  
pinMode(buttonPin, INPUT);  
}  
  
void loop(){  
    // read the state of the pushbutton value:  
    buttonState = digitalRead(buttonPin);  
  
    // check if the pushbutton is pressed.  
    // if it is, the buttonState is HIGH:  
    if (buttonState == HIGH) {  
        // turn LED on:  
        digitalWrite(ledPin, HIGH);  
    }  
    else {  
        // turn LED off:  
        digitalWrite(ledPin, LOW);  
    }  
}
```

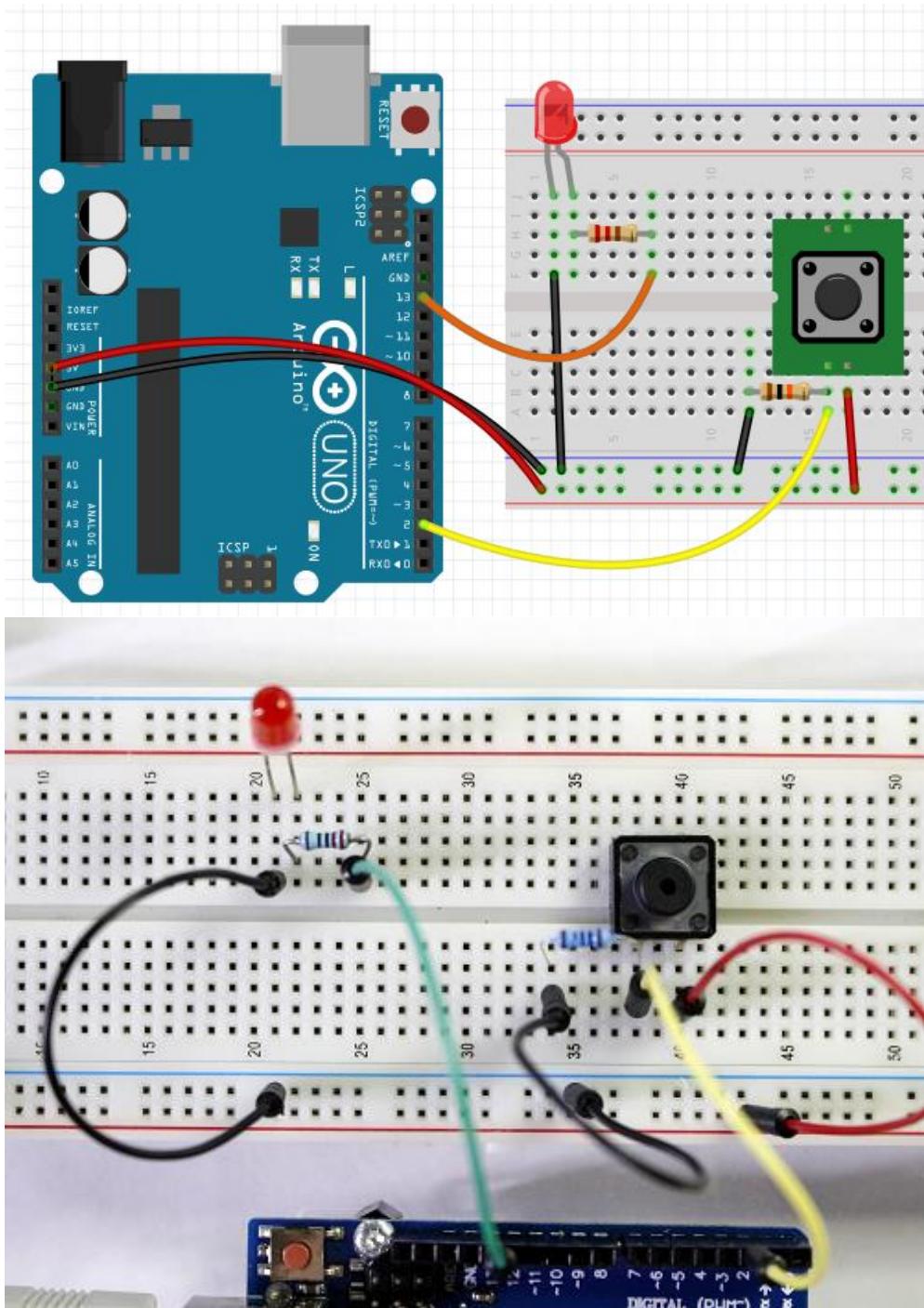
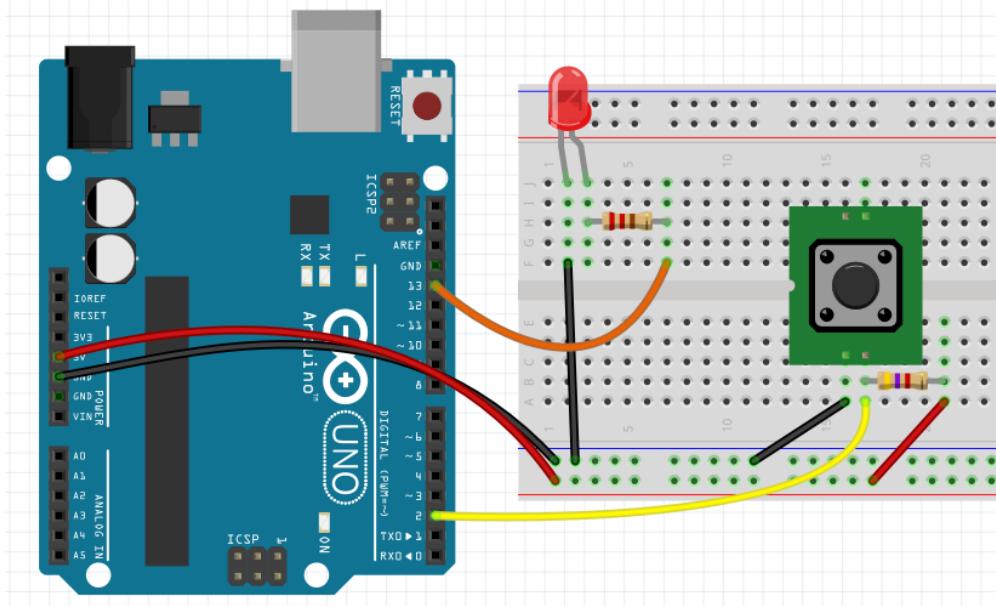


그림 46-1 풀다운 버튼 브레드보드

## 46.2 풀 업 버튼 예제 (PULL UP BUTTON)

Pull Up 회로도를 보면 (+) 방향에 저항이 연결되어 있습니다.  
5V 회로에서 1mA 전류를 사용하기 위한 저항값 계산입니다.

예제코드는 풀 다운 버튼 예제의 코드를 사용하면 됩니다. 회로 구성 후 D13 번 포트에 연결된 LED는 ON 상태입니다. 버튼을 누르면 LED는 OFF입니다.



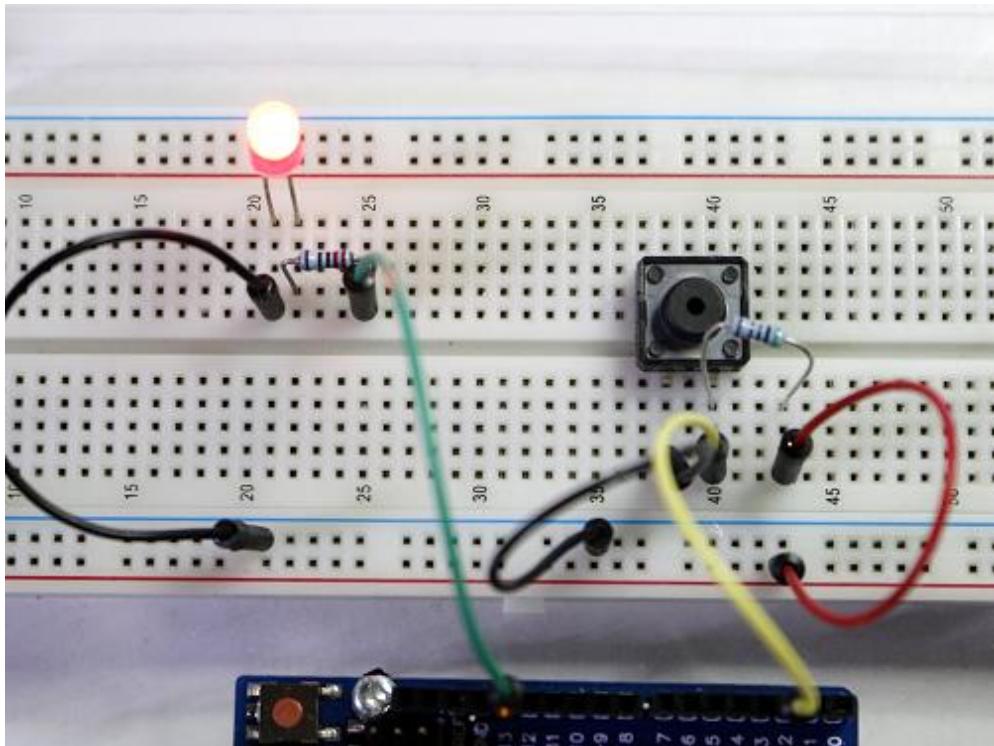


그림 46-2 풀업 저항 브레드보드 구성

## 47 INPUT\_PULLUP, 풀업 버튼 회로 구성

아두이노 우노 R3 보드에 사용되는 ATMEGA328P (아트메가 328p) MCU 들에는 거의 모두 내부풀업저항이 존재하고 있습니다. 외부에서 굳이 풀업 회로를 만들지 않아도 MCU 포트 상태를 풀업 상태로 설정 후 사용할 수 있도록 되어 있습니다. MCU를 구매하여 사용하는 입장에서는 비용, 시간이 이득이 되는 부분입니다.

풀업 구성은 전류를 흘려 보내어 지정된 기준 HIGH 신호를 유지하는 목적 회로입니다. MCU 자체에서는 내부 풀업 사용시에는 당연히 전류 소모가 더 발생됩니다. 여러 가지 이유에서 내부 풀업 회로에 대한 많은 장점, 단점에 대한 토론은 아직도 일어나고 있는 듯 합니다. 풀업 구성이 필요한 경우는 MCU 측면에서는 자기보호적인 측면도 있습니다. 외부 풀다운 회로로 구성되어 있고 적절한 HIGH 신호 레벨을 넘겨받는 것이 MCU 측면에서는 안전하게 작동됩니다. 하지만 MCU의 기능이 TTL, ADC, DAC, ROM, RAM, I2C, SPI 등의 다양해짐에 따라 MCU 자체에서도 초기화, 또는 특정 기능 수행 시 내부 풀업이 필요하다고 보여집니다.

내부 풀업 사용시 버튼 신호를 받아들이기 위해서는 아래와 같이 회로가 간단해집니다.

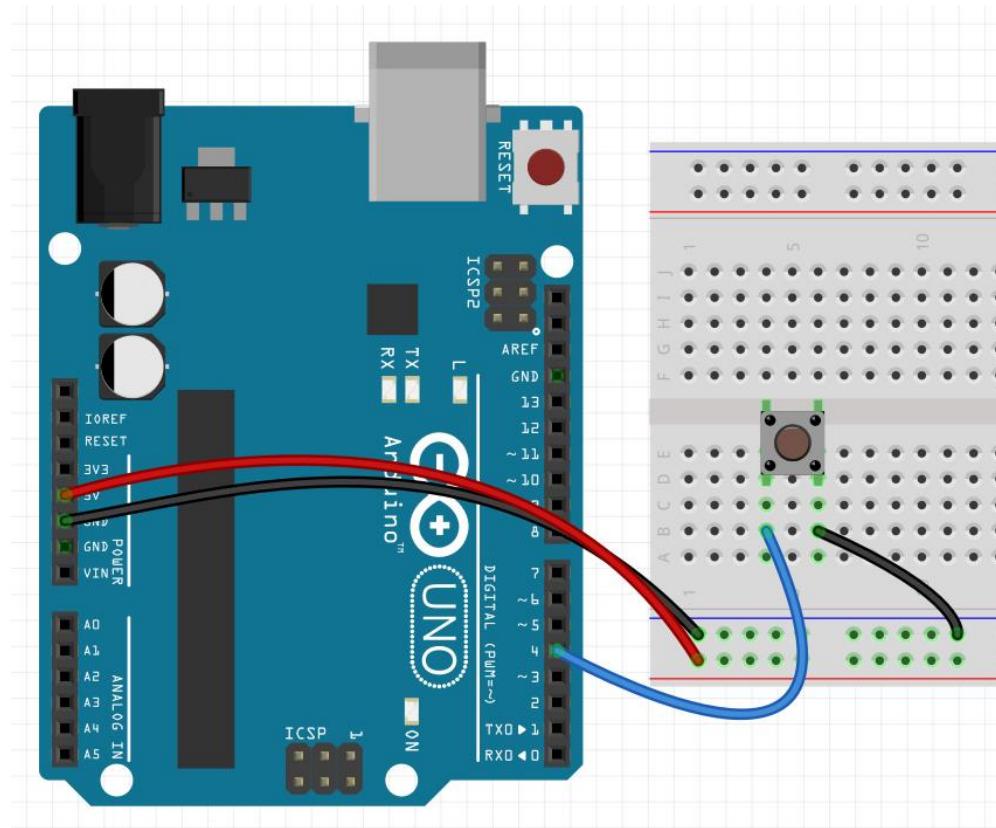


그림 47-1 내부풀업 사용 시 버튼 회로

풀업 버튼 예제 코드: 내부 풀업이므로 항상 HIGH 신호입니다. 버튼을 누르는 경우 LOW 신호로 변경 되게 됩니다.

```
const int buttonPin = 2;      // 버튼 신호 포트.
const int ledPin = 13;        // LED 포트.

int buttonState = 0; // 버튼 상태 전역 변수.

void setup() {
    pinMode(ledPin, OUTPUT);
    pinMode(buttonPin, INPUT_PULLUP);
}

void loop(){
    buttonState = digitalRead(buttonPin);

    // 버튼이 눌린 상태.
    if (buttonState == LOW) {
        // turn LED on:
        digitalWrite(ledPin, HIGH);
    }
    else {
        // turn LED off:
        digitalWrite(ledPin, LOW);
    }
}
```

## 48 › TILT SWITCH X 2

기울기 센서 스위치 SW-520D 모듈입니다.



그림 48-1 기울기 센서

기울기 센서입니다. 볼 스위치라고도 합니다.

부품을 살짝 흔들어 보면 안쪽에서 “딸깍” 거리는 소리가 들립니다.

기울기를 체크하기 위한 용도로 많이 사용됩니다. 또는 진동을 체크하기 위한 용도로도 사용됩니다.

520D 부품에는 2개의 리드선이 있습니다. 안쪽에는 조그만 형태의 전기가 통하는 도체의 공이 들어 있습니다.

2 개의 리드선에 볼(도체)이 걸쳐 있는 경우 전류가 흐르고, 그 외에는 전류가 끊어지게 됩니다.

2 개의 볼이 있어 위, 아래 또는 비규칙적인 진동이 있을 경우 2 개의 볼이 서로 충돌이 생겨 전류 또한 비규칙적으로 흐르게 됩니다.

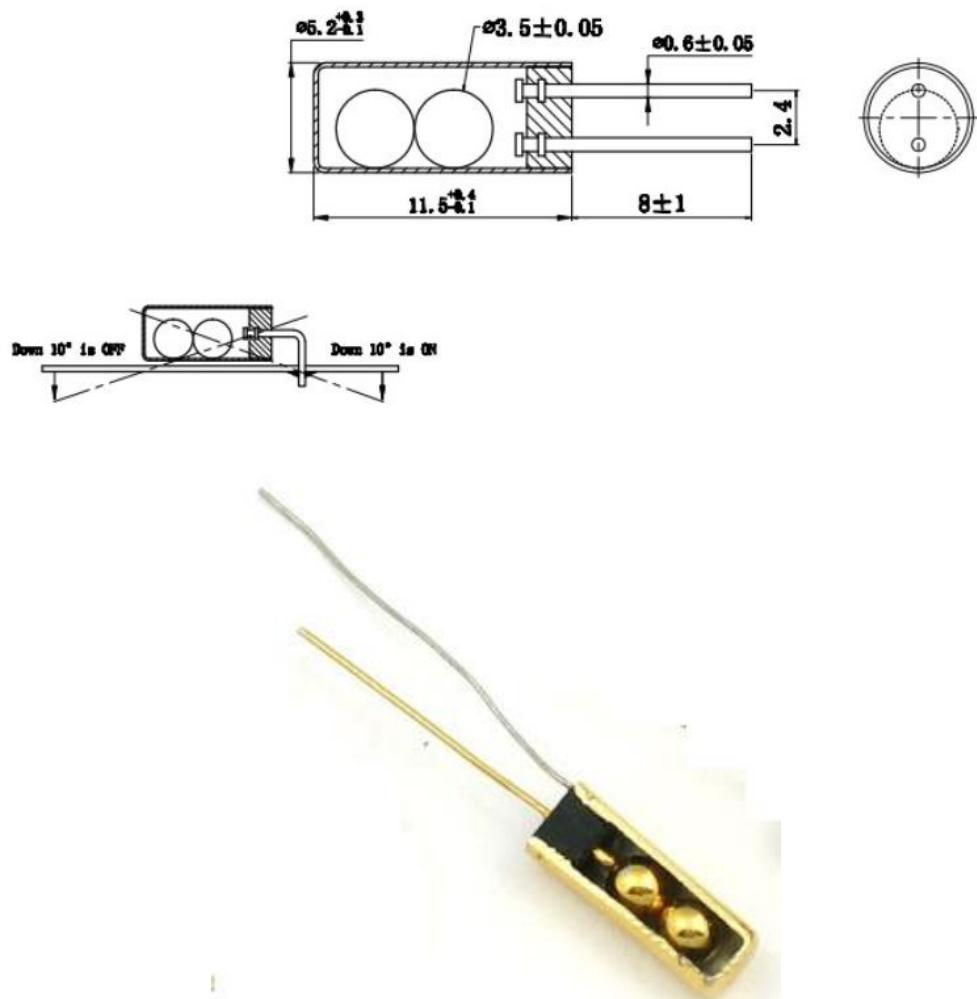
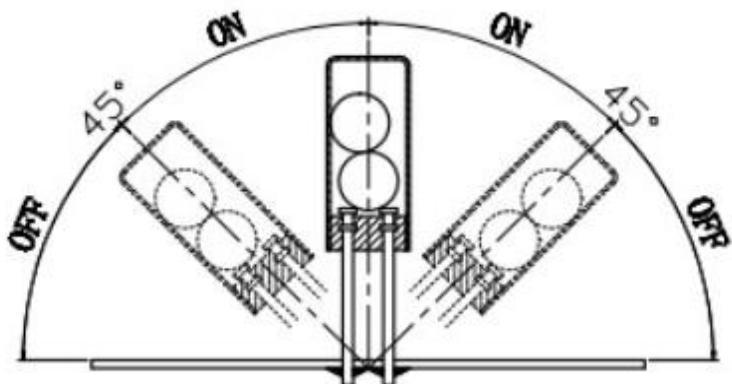


그림 48-2 SW-520D 내부 단면 이해도

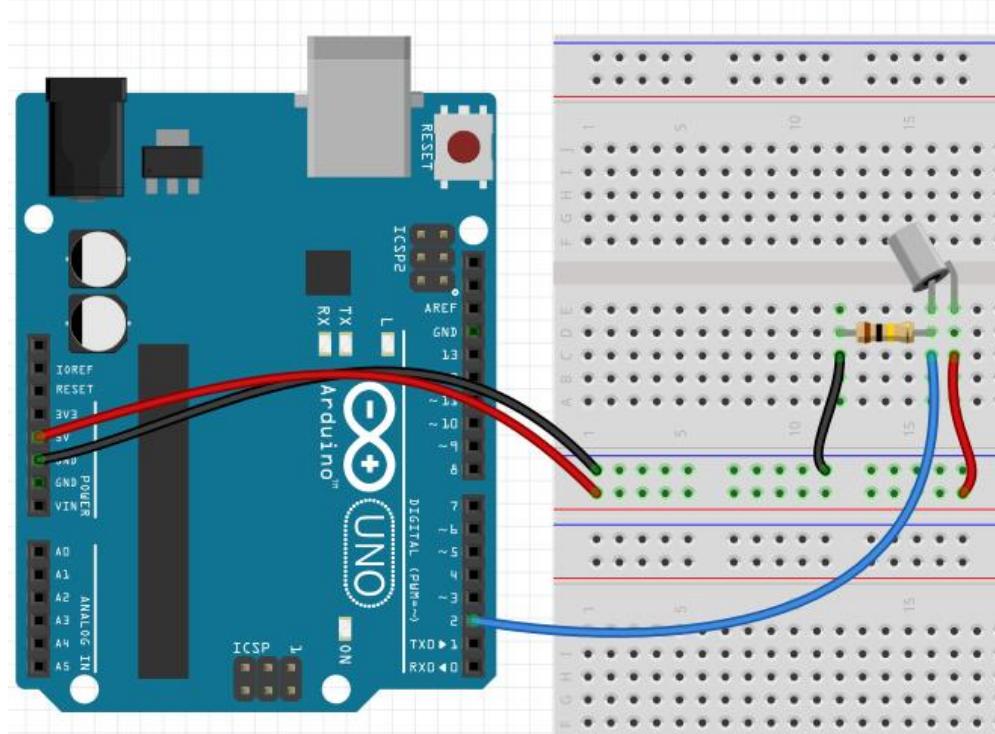
위의 예시 이미지를 보시면 내부에 2 개의 볼이 들어 있습니다. 기울기 또는 진동 등에 의한 스위치 역할을 합니다.

본 예제에서는 기울기 값에 의한 단순한 스위치 On / Off 합니다.



SW-520D 센서는 2 개의 핀에 GND, SIGNAL 구분은 없습니다. 버튼과 동일한 기능입니다. 신호를 받을 핀에는 10K 저항을 연결 해 아두이노 보드 디지털 2 번 포트에 연결합니다.

아두이노 보드로의 연결은 아래 이미지를 참조 합니다.



스케치 예제 코드는 Digital->Button 예제를 그대로 사용 합니다.

```

// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 2;      // the number of the
pushbutton pin
const int ledPin = 13;        // the number of the
LED pin

// variables will change:
int buttonState = 0;          // variable for reading
the pushbutton status

void setup() {
    // initialize the LED pin as an output:
    pinMode(ledPin, OUTPUT);
    // initialize the pushbutton pin as an input:
    pinMode(buttonPin, INPUT);
}

void loop() {
    // read the state of the pushbutton value:
    buttonState = digitalRead(buttonPin);

    // check if the pushbutton is pressed.
    // if it is, the buttonState is HIGH:
    if (buttonState == HIGH) {
        // turn LED on:
        digitalWrite(ledPin, HIGH);
    }
    else {
        // turn LED off:
        digitalWrite(ledPin, LOW);
    }
}

```

아두이노에 버튼 예제 프로그램 업로드 후 센서의 기울기를 조절 해 봅니다.  
 또는 센서를 툭툭 쳐서 건드려 봅니다.  
 LED 13 번 온/오프 되는 것을 확인 합니다.  
 실제 제품에 적용 시에는 순간의 기울기나 진동을 체크하긴 보단 타이머 인터럽트로  
 일정 시간 센서 값 평균을 구해서, 변동이 있는 경우만 처리하는 것도 좋습니다.

특정한 목적, 특수한 경우가 아닌 경우, 거의 모든 아날로그 센서 값 기준 하드웨어 동작은 지정 시간 대비 또는 지정 클럭 수 대비 평균을 구하여 프로그래밍 하도록 합니다.

## 49 > LM35 SENSOR MODULE X 1

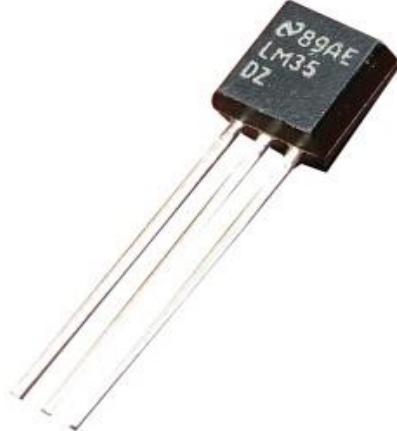


그림 49-1 LM35 ( DZ ) 온도 센서

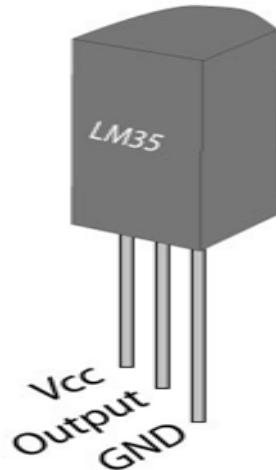


그림 49-2 핀맵

온도 센서입니다. LM35 DZ 센서입니다.

섭씨 1 도에 10 mA 의 변화 값을 구할 수 있는 온도 센서입니다.

측정 온도의 범위는 섭씨온도 0~100 도입니다. 비접촉식 온도 센서 부품으로 많이 사용되고 있습니다. 물론, 손으로 감싸 쥐거나 하여 온도를 측정할 수도 있습니다. 하지만 대부분 공기중의 온도를 측정하는데 사용됩니다.

LM35 소자는 온도값을 입력된 전압에 대한 비율로 OUTPUT 핀으로의 출력 전압을 받아서 측정하는 방식입니다. 측정하는 MCU 의 레퍼런스 기준값에 의해 온도 측정된 값이 다르게 넘어오므로 데이터시트를 참조하여 사용합니다.

LM35DZ 페이터쉬트:

<http://www.gameplusedu.com/pds/gpshop/arduino/pdf/LM35DZ.pdf>

아두이노 연결:

| Pin | Function                         | Name   |
|-----|----------------------------------|--------|
| 1   | Supply voltage; 5V (+35V to -2V) | VCC    |
| 2   | Output voltage (+6V to -1V)      | Output |
| 3   | Ground (0V)                      | Ground |

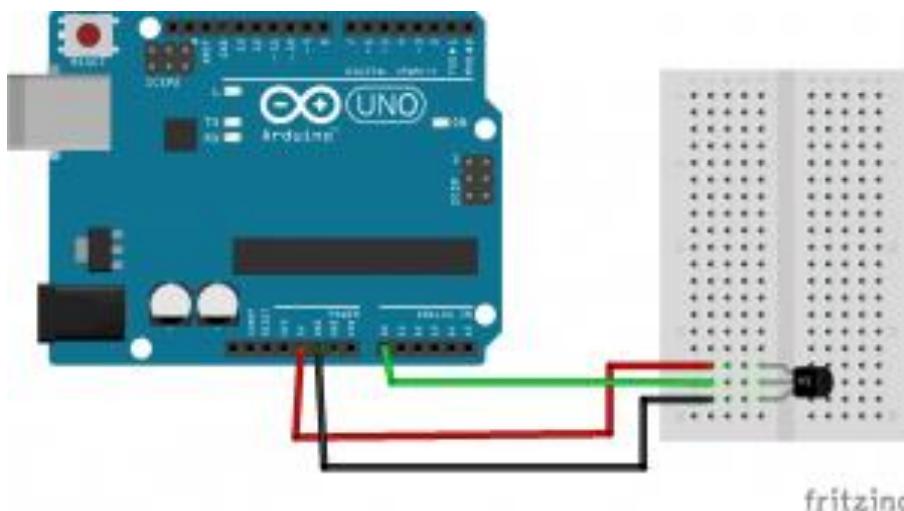


그림 49-3 아두이노 연결 이해도

온도 값 계산시 입력 전원 5V, 3V3 참조 기준 연산 값으로 계산 해주면 됩니다.  
(예제코드: LM35 code)

```
// 1 초마다 온도를 구합니다.
const unsigned int TEMP_SENSOR_PIN = 0; // 온도 센서 입력 아날로그 포트
const float SUPPLY_VOLTAGE = 5.0; // 5V 전류 연결 정의 합니다.
const unsigned int BAUD_RATE = 9600; //

const float get_temperature() {
    const int sensor_voltage = analogRead(TEMP_SENSOR_PIN);
    const float voltage =
        sensor_voltage * SUPPLY_VOLTAGE / 1024;
    return (voltage * 100);
}
```

```
void setup() {
    Serial.begin(BAUD_RATE); // 보 레이트 설정.
}

void loop() {
    Serial.print(get_temperature());
    Serial.println(" C");
    delay(1000);
}
```

## 50 › PHOTO RESISTOR X 3

광 센서 입니다. 조도 센서 / 라이트 센서 라는 명칭도 있습니다.



그림 50-1 Photo Resistor

다른 용어로는 LDR 입니다. Light Dependent Resistor. 빛 의존적 저항이라고 풀이  
가 됩니다. 광(Photo) + Resistor (저항) 합성어입니다.

광소자, 광도전 센서라고도 합니다. 포토셀(Photo Cell)이라는 용어도 있습니다.

외부 빛의 세기에 따라 내부 저항 값이 감소합니다. 즉, 가변 저항인데 빛의 세기에 따라 자동으로 변화한다고 보면 됩니다. 광센서는 저항 타입이라 양극(+)과 음극(-) 구분하지 않습니다. 외부 빛이 적거나 차단되면 저항의 크기가 커집니다. 반대로 외부 빛이 많다면 광센서의 저항 매우 작아 집니다.

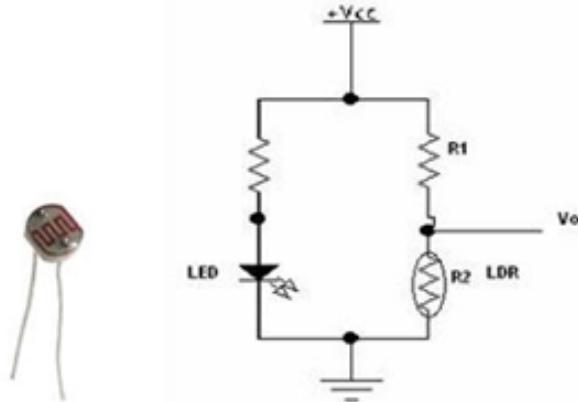


그림 50-1 조도센서 구성 회로도

아두이노에 와이어링은 아날로그 포트, +5V에 10K 저항을 적용하여 테스트 해봅니다. 회로 구성은 풀 다운 회로 구성입니다.

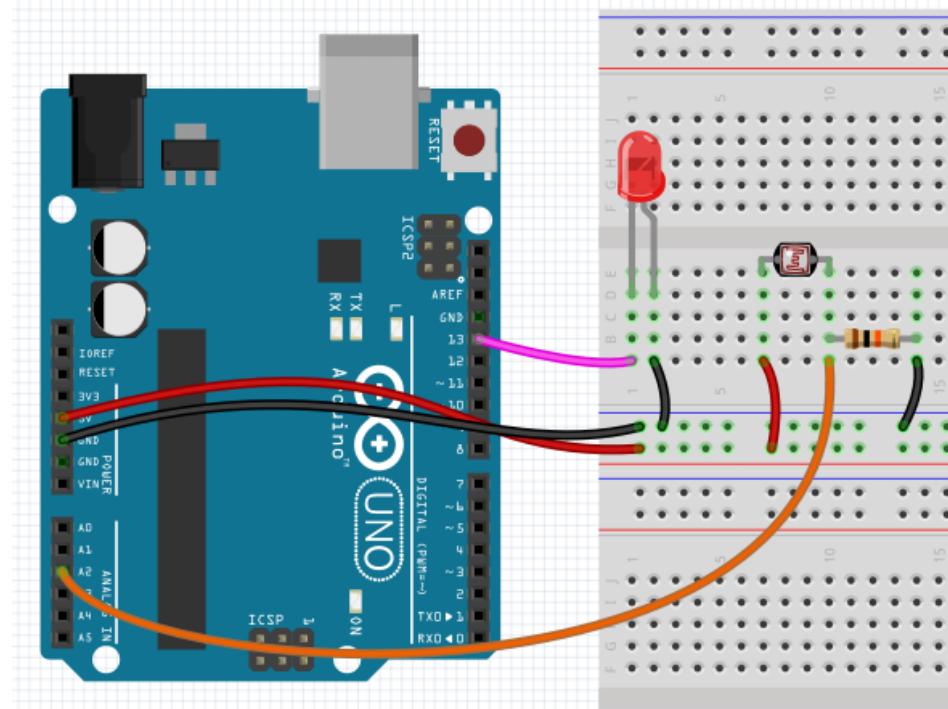


그림 50-2 브레드보드 회로도

키트에 3 개 센서 포함된 이유는 빛 자체가 방향성 데이터이므로, 가장 이상적인 빛 센서 체크는 삼각형 기준 밖으로(또는 상향) 센서를 배치 후, 각 3 개의 센서 입력의 평균값으로 조도 값을 측정하면 매우 정확한 결과치를 얻을 수 있습니다.

물론 1 개로도 충분합니다.

회로 C&R 구성에 따라 가장 밝은 빛에서의 체크 용도, 보통 날씨의 조도, 빛이 약간 있는 어두운 곳에서의 사용 등등 많이 구분됩니다.

```
int lightPin = 3;           // 아날로그 포트 A3
int threshold = 250;        // 조도값 250 기준으로 LED 온/오프

void setup()
{
    Serial.begin(9600);    //Begin serial communication
    pinMode(13, OUTPUT);
}

void loop()
{
    Serial.println(analogRead(lightPin));

    if(analogRead(lightPin) > threshold )
    {
        digitalWrite(13, HIGH);
        Serial.println("high");
    }else{
        digitalWrite(13, LOW);
        Serial.println("low");
    }

    delay(100);
}
```

## 51 > FIRE X 1 (FLAME SENSOR)



그림 51-1 적외선 불꽃감지 센서

Flame Sensor 입니다. 화염, 불꽃 감지 센서입니다. 포토트랜지스터 수신용 부품입니다. “광트랜지스터” 라고도 합니다.

포토트랜지스터의 외부는 Infrared LED 의 외부는 Visible light filter(가시광선필터) 감싸져 있습니다. 트랜지스터 NPN 방식의 부품입니다.

주로 화염의 고유 760nm~1100nm 파장의 빛을 검출 하기 위해 사용 됩니다.

Flame Sensor DataSheet:

<http://www.igameplus.com/pds/gpshop/arduino/pdf/flamesensor.pdf>

»» 적외선, 가시광선이란?

우리의 주위에는 전자기파라는 것이 있습니다. 전자기파는 고유의 “파장”으로 구분하고 있습니다. 그 중 시각적으로 보이는 부분, “빛(Light)”이라고 명명된 부분은 “가시광선”이라고 말합니다. “빨주노초파남보” 무지개 색상입니다.

빛은 우리가 볼 수 있는 영역의 전자기파, 파장입니다. 파장이 제일 큰 부분, 긴 부분이 빨강색으로 보여지며, 제일 짧은 파장은 보라색으로 보여집니다. 보라색보다 더 짧은 파장은 “자외선”이라고 합니다.

적외선은 빨강색보다 큰 파장입니다. 파장의 크기로 보았을 때 빨간색의 바깥쪽에 있다고 하여 적외선입니다. Infrared ( Infra + red ) 입니다.



## 51.1 FLAME SENSOR MODULE TYPE

일반적으로 사용되는 불꽃감지 센서 모듈은 PCB에 회로로 구성되어 있어 저항 등의 다른 부품 연결 없이 간편하게 사용 가능합니다.



그림 51-3 Flame Sensor Module

불꽃 감지 센서 모듈을 사용하는 경우에는 아래와 같은 아두이노 코드를 사용합니다. 모듈로부터의 신호 변경 (HIGH → LOW, 또는 LOW→HIGH)이 있는 경우에 부저를 울리면 정확한 체크를 할 수 있습니다. 반고정 가변 저항을 미세하게 조절 하면서 체크를 할 수 있습니다. 불꽃 감지 센서의 제조사에 따라 아날로그 값이 출력되는 모듈도 있습니다.

아두이노 예제코드: 불꽃, 화염 감지 반응이 있을 경우 부저를 울려 경고를 하도록 하는 코드입니다.

```

int buzzer_port = 8; // 피에조 부저 또는 패시브 부저 포트.
int flame_port = 2; //

void setup()
{
    Serial.begin(9600);
    pinMode(flame_port,INPUT);
}

```

```
}

void loop()
{
    int sensorValue = digitalRead(2);
    // 감지 센서의 기본 값이 LOW 인 경우는 LOW 변경
    if(sensorValue == HIGH)
    {
        Serial.println(sensorValue);
        digitalWrite(buzzer_port, HIGH);
    }
    else
    {
        digitalWrite(buzzer_port, LOW);
    }
}
```

## 51.2 FLAME SENSOR 직접 회로 구성

적외선 수신 센서 부품과 10K 저항을 사용하여 불꽃 감지를 하도록 합니다.

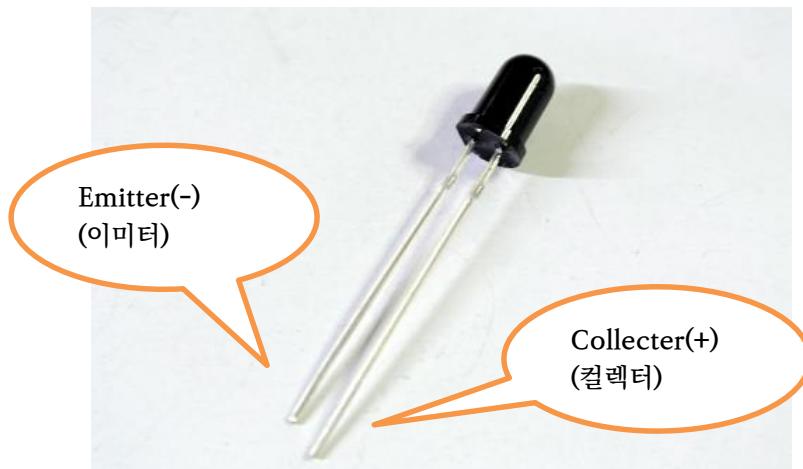


그림 51-4 불꽃 감지 센서, 적외선 수신 센서

연결 회로도는 아래와 같습니다.

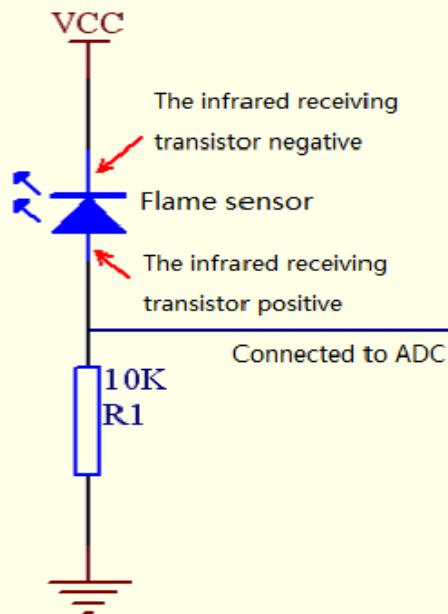


그림 51-5 적외선 센서 연결 설명 그림

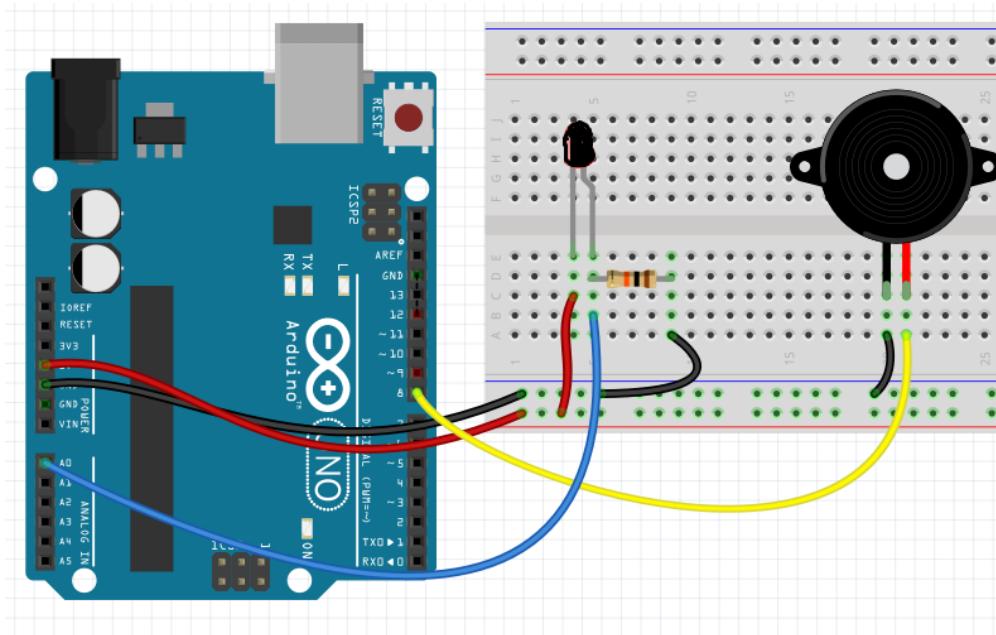


그림 51-6 브레드보드 회로 구성

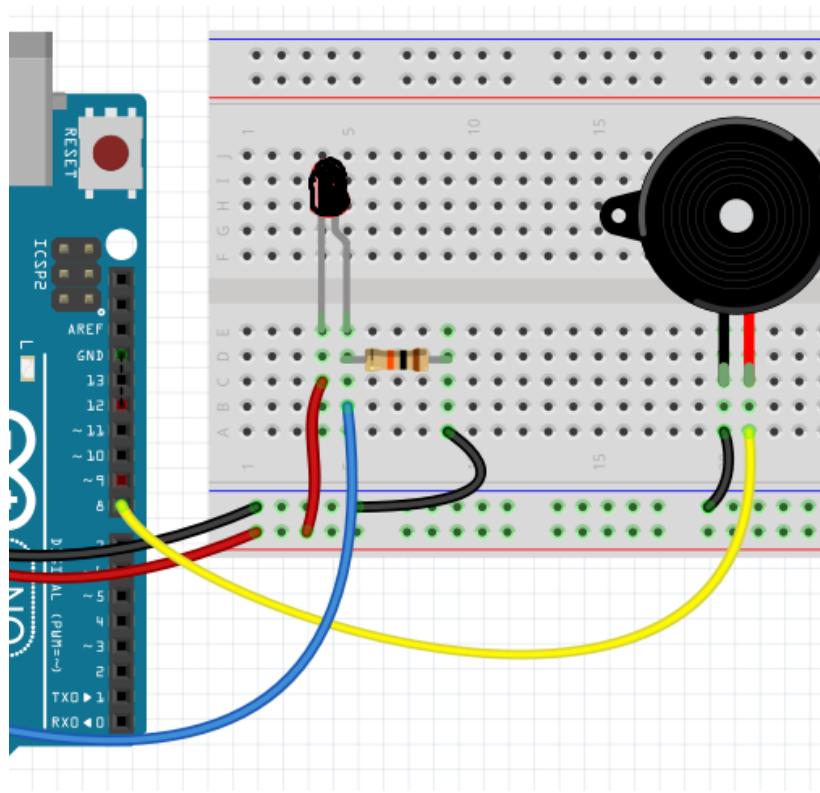


그림 51-7 브레드보드 상세 연결 그림

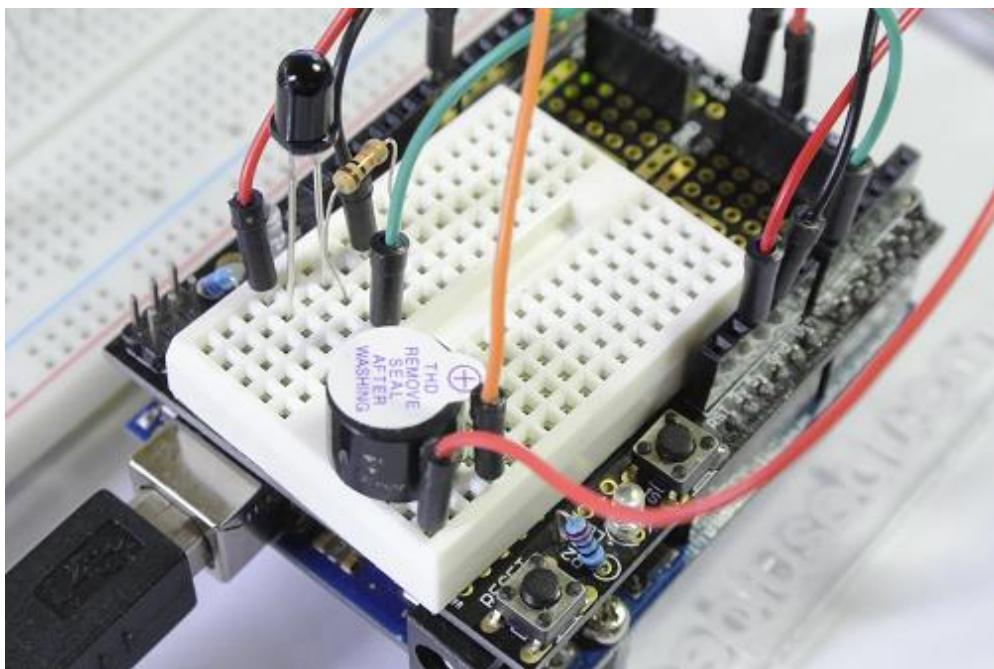


그림 51-8 불꽃감지 센서 구성 그림

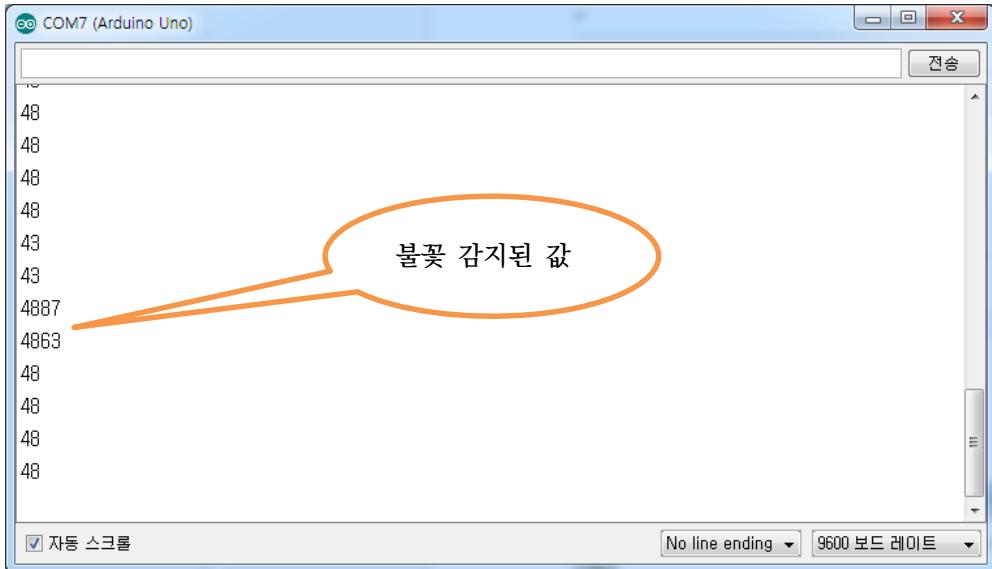
예제코드: 불꽃, 화염 감지 반응이 있을 경우 부저를 울려 경고를 하도록 하는 코드입니다.

예제 코드 업로드 후 보호자, 책임자의 안전한 관리 상태에서 라이터, 또는 양초를 사용하여 적외선 센서 방향의 60도 내외, 10 ~ 15센치 거리에서 테스트를 해보도록 합니다.

```
int flame = A0 ;// define the flame interface analog 0
interface
int Beep = 8 ;// buzzer interface defines the interface
number 7
int val = 0 ;// define numeric variables

void setup ()
{
  pinMode (Beep, OUTPUT) ;// define LED as output interface
  pinMode (flame, INPUT) ;// define the buzzer as the input
interface
  Serial.begin (9600) ;// set the baud rate to 9600
}

void loop ()
{
  val = analogRead (flame) ;// read the analog value flame
sensor
  // Serial.println (val) ;// output analog values, and
print them out
  // convert to voltage.or check in 0~1023 range.
  float sensorV = val*5.0/1024.0;
  int sensormV = sensorV*1000;
  //Serial.println(sensormV);
  if(sensormV > 4000 ) //
  {
    digitalWrite(Beep,HIGH);
  }
  else
  {
    digitalWrite(Beep,LOW);
  }
  delay(500);
}
```



48  
48  
48  
48  
43  
43  
4887  
4863  
48  
48  
48  
48

자동 스크롤 No line ending ▾ 9600 보드 레이트 ▾

## 52 > INFRARED RECEIVER X 1

적외선 수신기입니다.

적외선 리모컨 컨트롤러 또는 적외선 송신기에서 보내는 신호를 받는 모듈입니다.

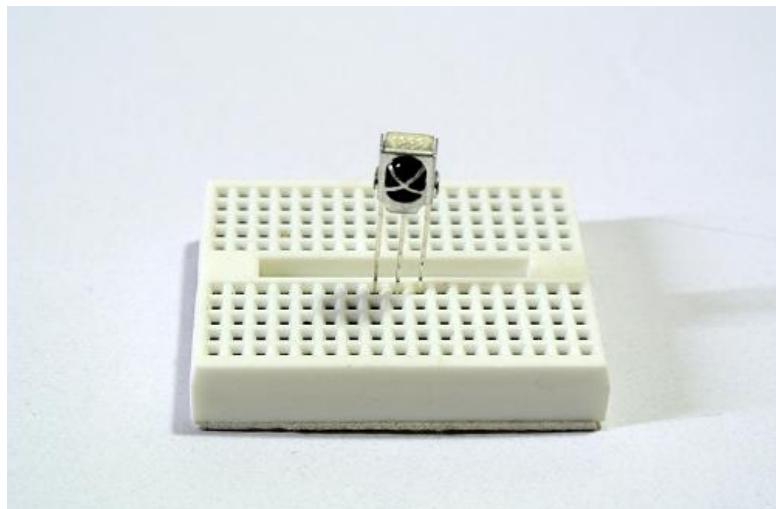


그림 52-1 적외선 수신기

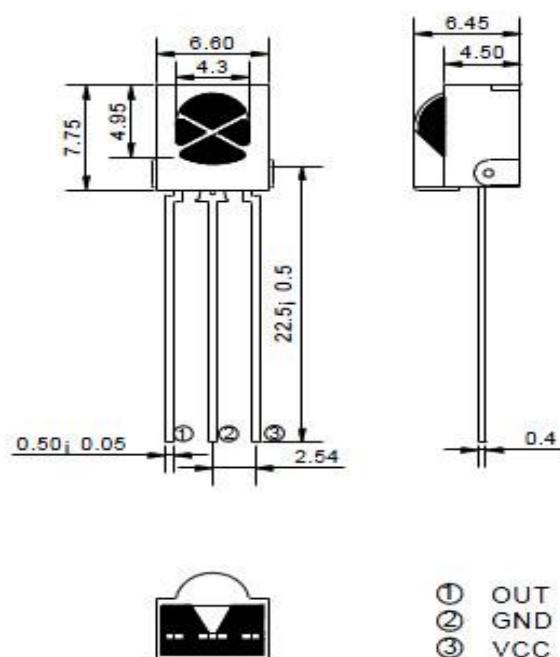


그림 52-2 부품 크기와 핀 맵



그림 52-3 리모트 컨트롤러



그림 52-4 리모트 컨트롤러의 배터리 장착

IR Receiver 핀맵입니다.

| 적외선 리시버 | 아두이노 우노 |
|---------|---------|
| OUT     | 11      |
| GND     | GND     |
| VCC     | 5V      |

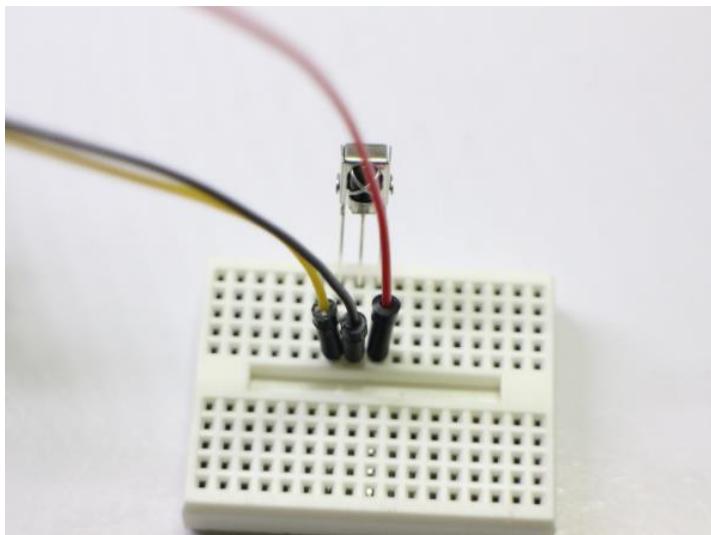


그림 52-5 미니브레드보드에 설치된 모습

리모트 컨트롤 라이브러리 & 예제 코드입니다.

적외선 컨트롤러 모든 참조를 할 수 있습니다.

<https://github.com/shirriff/Arduino-IRremote>

또는 다운로드 [Arduino-IRremote-master.zip](#)

[\(http://www.gameplusedu.com/pds/gpshop/arduino/comms/ir-receiver-module/Arduino-IRremote-master.zip\)](http://www.gameplusedu.com/pds/gpshop/arduino/comms/ir-receiver-module/Arduino-IRremote-master.zip)

Arduino-IRremote 라이브러리 설치 위치는 아두이노 사용자 라이브러리 위치 아래에 압축 해제하여 넣도록 합니다.

## 52.1 IR 수신 라이브러리 사용시 주의점

적외선 리모트 컨트롤러 라이브러리 사용시 코드에는 아래와 같은 헤더 파일 선언이 필요합니다.

```
#include <IRremote.h>
```

## » 아두이노 라이브러리 파일 위치 컴파일 찾기 순서

아두이노는 내부적으로 C/C++ 컴파일러를 사용하고 있습니다.

작성되는 코드에서 `#include <IRremote.h>` 라고 선언하여 사용하는 경우 아두이노 컴파일러는 우선적으로 기본 라이브러리의 디렉터리의 파일부터 찾게 되어 있습니다. 만약 찾는 경우 나머지 사용자 라이브러리의 디렉터리는 찾지 않게 됩니다. 우연히 IRremote.h 파일의 내용이 같을 수는 있지만, 대부분 용도에 맞게 변경, 추가된 사항들이라 다른 목적으로 사용되고 있습니다.

여기에서 작성되는 코드에서 필요한 라이브러리는 아두이노 IRremote 라이브러리의 IRremote.h 파일이지만, 기본 라이브러리 IRremote.h 사용으로 인식되어 컴파일 에러 등의 문제가 있습니다. 우연히 컴파일, 빌드, 업로드가 되더라도 목적하는 결과와 전혀 다르게 나올 수 있습니다.

## » 예제 빌드 시 아래와 비슷한 컴파일 에러 메시지가 나올 수 있습니다.



The screenshot shows the Arduino IDE interface with a terminal window displaying a compilation error. The error message is as follows:

```
컴파일 오류 발생.
E:\EMBEDDED-PRJ\Arduino\arduino-1.5.6-r2-windows\arduino-1.5.6-r2\hardware\tools\avr\bin\avr-g++ -c -g -Os -fno-exceptions -ffunction-sections -fdata-sections -MD -mmcu=atmega328p -DF_CPU=1600000UL -DARDUINO=156 -DARDUINO_AVR_UNO -DARDUINO_ARCH_AVR -IE: E:\EMBEDDED-PRJ\Arduino\arduino-1.5.6-r2-windows\arduino-1.5.6-r2\hardware\tools\avr\cores\arduino -IE: E:\EMBEDDED-PRJ\Arduino\arduino-1.5.6-r2-windows\arduino-1.5.6-r2\hardware\tools\avr\variants\standard -IE: E:\EMBEDDED-PRJ\Arduino\arduino-1.5.6-r2-windows\arduino-1.5.6-r2\libraries\Robot\IRremote\src\IRremote.cpp -o R:\Temp\buildd4440870335709870051.tmp\Robot\IRRemote\IRRemote.cpp.o
E:\EMBEDDED-PRJ\Arduino\arduino-1.5.6-r2-windows\arduino-1.5.6-r2\hardware\tools\avr\bin\avr-g++ -c -g -Os -fno-exceptions -ffunction-sections -fdata-sections -MD -mmcu=atmega328p -DF_CPU=1600000UL -DARDUINO=156 -DARDUINO_AVR_UNO -DARDUINO_ARCH_AVR -IE: E:\EMBEDDED-PRJ\Arduino\arduino-1.5.6-r2-windows\arduino-1.5.6-r2\hardware\tools\avr\cores\arduino -IE: E:\EMBEDDED-PRJ\Arduino\arduino-1.5.6-r2-windows\arduino-1.5.6-r2\hardware\tools\avr\variants\standard -IE: E:\EMBEDDED-PRJ\Arduino\arduino-1.5.6-r2-windows\arduino-1.5.6-r2\libraries\Robot\IRremote\src\IRremoteTools.cpp -o R:\Temp\buildd4440870335709870051.tmp\Robot\IRRemote\IRRemoteTools.cpp.o
E:\EMBEDDED-PRJ\Arduino\arduino-1.5.6-r2-windows\arduino-1.5.6-r2\libraries\Robot\IRremote\src\IRremoteTools.cpp:5: error: 'TKD2' was not declared in this scope
```

메시지의 내용은

“.....Arduino\arduino-1.5.6-r2-windows\arduino-1.5.6-r2\libraries\Robot\IRremote\src\IRremoteTools.cpp:5: error: 'TKD2' was not declared in this scope”

또는

In function 'void dump(decode\_results\*)':

\_4.30.ino:100:35: error: 'LG' was not declared in this scope

IRRemote.h라는 라이브러리 헤더 파일 명칭이 아두이노의 로봇 IR 파일들과 중복되어 RobotIRRemote 스케치 기본 라이브러리가 빌드 시 나오는 현상입니다.

RobotIRRemote 라이브러리의 헤더파일 중복을 피하기 위해 RobotIRRemote 디렉터리를 옮기거나 삭제를 합니다. 윈도우 탐색기를 열어서 아두이노 스케치 프로그램이 있는 디렉터리로 갑니다.

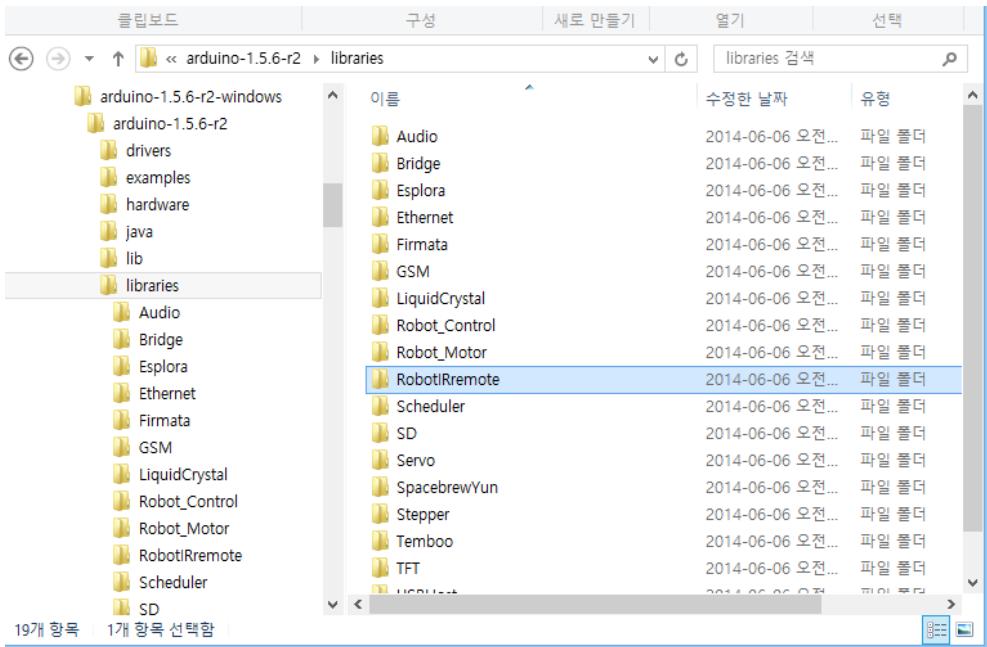


그림 52-6 아두이노 스케치 IDE 설치된 디렉터리

아두이노의 기본 라이브러리 디렉터리 → RobotIRremote라는 디렉터리를 다른 곳으로 옮겨 놓습니다. 삭제하거나, C:\TEMP 등의 별도의 디렉터리로 옮겨 놓기 권장합니다.

## 52.2 IR 수신 모듈 와이어링

IR 수신 모듈 IR 수신기의 신호 포트와 아두이노 우노의 포트와 연결하면 됩니다. 수신기 본체와 브레이크아웃보드의 연결 순서가 조금 틀립니다.

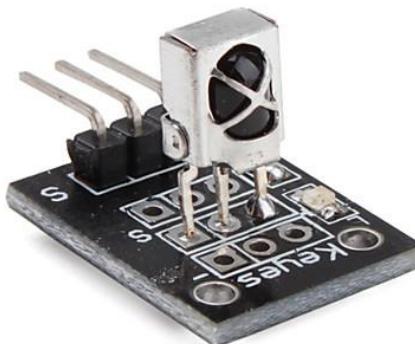


그림 52-7 적외선 수신 모듈

브레드보드에 와이어링 이미지입니다. 연결 와이어링의 색상을 참조하기 바랍니다.

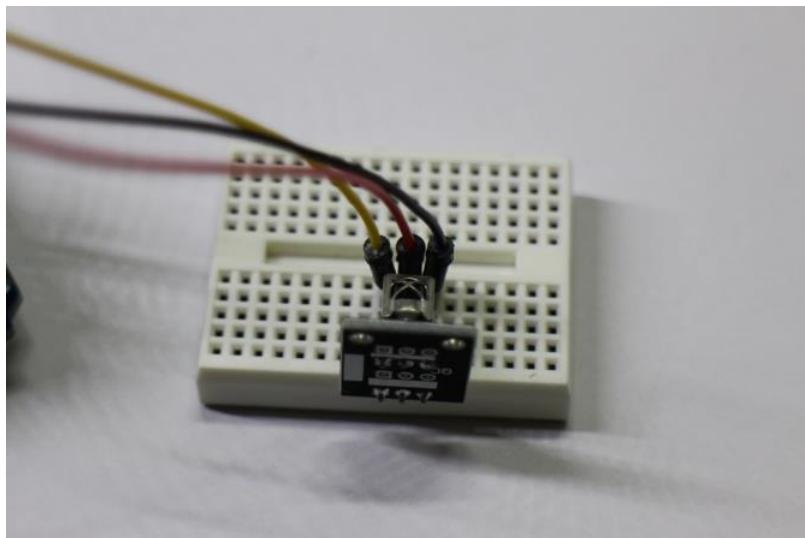


그림 52-8 미니브레드보드에 장착된 모습

아두이노 연결:

| IR 수신 모듈 | 아두이노 |
|----------|------|
| -        | GND  |
| +        | 5V   |
| S        | 11   |

» 아래의 예제코드는 IR 수신이 있을 경우에 LED 13 On/Off Toggle 되는 예제입니다.

IR 수신 예제 코드.

```
/*
IR_remote_tester_and_detector
Connect the output pin of Infrared remote to DIG 2 or 11
Connect an LED to pin 13.
*/
#include <IRremote.h>

const int irReceiverPin = 11; // PWM or you want.
const int ledPin = 13;
IRrecv irrecv(irReceiverPin); //create an IRrecv object
decode_results decodedSignal; //stores results from IR sensor

void setup()
```

```

{
    pinMode(ledPin, OUTPUT);
    irrecv.enableIRIn(); // Start the receiver object
}

boolean lightState = false; //keep track of whether the LED is on
unsigned long last = millis(); //remember when we last received an
IRmessage

void loop()
{
    //this is true if a message has been received
    if (irrecv.decode(&decodedSignal) == true)
    {
        if (millis() - last > 250)
        {
            //has it been 1/4 sec since last message
            lightState = !lightState; //toggle the LED
            digitalWrite(ledPin, lightState);
        }

        last = millis();
        irrecv.resume(); // watch out for another message
    }
}

```

» 아래의 예제 코드는 수신되는 코드를 시리얼 포트로 모니터링 하는 예제 코드입니다.

리모트 수신 코드 라이브러리에 포함되어 있습니다. IRrecvDump 예제코드입니다.  
IR 수신 코드 출력 예제코드)

```

/*
 * IRremote: IRrecvDump - dump details of IR codes with IRrecv
 * An IR detector/demodulator must be connected to the input RECV_PIN.
 * Version 0.1 July, 2009
 * Copyright 2009 Ken Shirriff
 * http://arcfn.com
 * JVC and Panasonic protocol added by Kristian Lauszus (Thanks to zenwheel
and other people at the original blog post)
 * LG added by Darryl Smith (based on the JVC protocol)
 */

```

```
#include <IRremote.h>
```

```

int RECV_PIN = 11;

IRrecv irrecv(RECV_PIN);

decode_results results;

void setup()
{
    Serial.begin(9600);
    irrecv.enableIRIn(); // Start the receiver
}

// Dumps out the decode_results structure.
// Call this after IRrecv::decode()
// void * to work around compiler issue
//void dump(void *v) {
//    decode_results *results = (decode_results *)v
void dump(decode_results *results) {
    int count = results->rawlen;
    if (results->decode_type == UNKNOWN) {
        Serial.print("Unknown encoding: ");
    }
    else if (results->decode_type == NEC) {
        Serial.print("Decoded NEC: ");
    }
    else if (results->decode_type == SONY) {
        Serial.print("Decoded SONY: ");
    }
    else if (results->decode_type == RC5) {
        Serial.print("Decoded RC5: ");
    }
    else if (results->decode_type == RC6) {
        Serial.print("Decoded RC6: ");
    }
    else if (results->decode_type == PANASONIC) {
        Serial.print("Decoded PANASONIC - Address: ");
        Serial.print(results->panasonicAddress,HEX);
        Serial.print(" Value: ");
    }
    else if (results->decode_type == LG) {
        Serial.print("Decoded LG: ");
    }
    else if (results->decode_type == JVC) {
        Serial.print("Decoded JVC: ");
    }
}

```

```

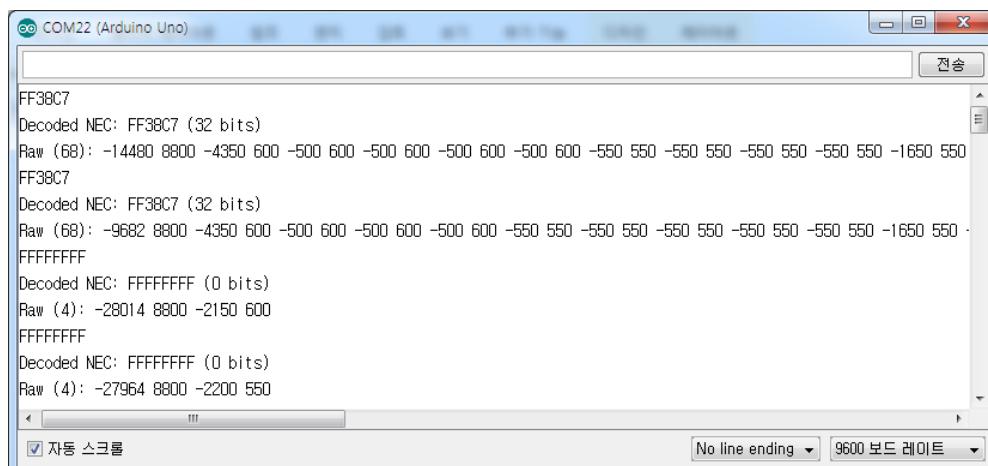
Serial.print(results->value, HEX);
Serial.print(" (");
Serial.print(results->bits, DEC);
Serial.println(" bits)");
Serial.print("Raw (");
Serial.print(count, DEC);
Serial.print("): ");

for (int i = 0; i < count; i++) {
    if ((i % 2) == 1) {
        Serial.print(results->rawbuf[i]*USECPERTICK, DEC);
    }
    else {
        Serial.print(-(int)results->rawbuf[i]*USECPERTICK, DEC);
    }
    Serial.print(" ");
}
Serial.println("");
}

void loop() {
    if (irrecv.decode(&results)) {
        Serial.println(results.value, HEX);
        dump(&results);
        irrecv.resume(); // Receive the next value
    }
}

```

위의 코드 적용 시리얼 모니터 창을 열어서 출력되는 코드를 살펴보도록 합니다.

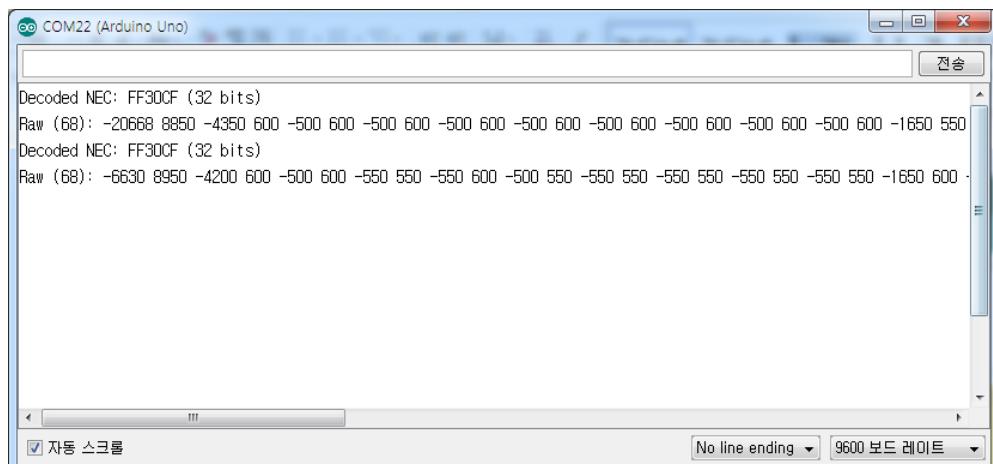


예제 코드는 리모트 컨트롤러 버튼을 누르고 있으면 연속해서 값이 넘어오게 되어 있습니다. 리모트 컨트롤러 버튼을 계속 누르고 있으면 “Decoded NEC: FFFFFFFF (0 bits)”라는 메시지가 보입니다.

리모트 컨트롤러 버튼을 눌렀을 경우의 메시지는 아래의 문자열이 출력됩니다. “Decoded NEC: FF18E7 (32 bits)” 32Bits라는 항목이 보입니다.

위의 코드에서 loop 함수 내부의 코드를 아래와 같이 변경하여 리모트 컨트롤러 버튼을 누르는 경우에만 해당 키 값을 체크하도록 변경합니다.

```
void loop()
{
    //this is true if a message has been received
    if (irrecv.decode(&results) == true)
    {
        if (results.bits > 0) // bit > 0 인 경우에만 처리.
        {
            dump(&results);
        }
        irrecv.resume(); // watch out for another message
    }
}
```



시리얼 모니터 창을 열고, 리모트 컨트롤러의 버튼을 눌러봅니다.  
계속 누른 상태여도 리모트 컨트롤러의 버튼 값만 볼 수 있도록 변경됩니다.

## 53 > 인체 감지 센서 모듈 x 1

인체 감지 센서 모듈입니다. (Pyroelectric (Passive) Infrared motion sensor)  
HC-SR501 PIR Motion Sensor Module 입니다.

PIR 센서입니다. 실생활에 많이 사용되고 있는 센서중의 하나이기도 합니다.



그림 53-1 인체감지 센서(PIR 모듈)

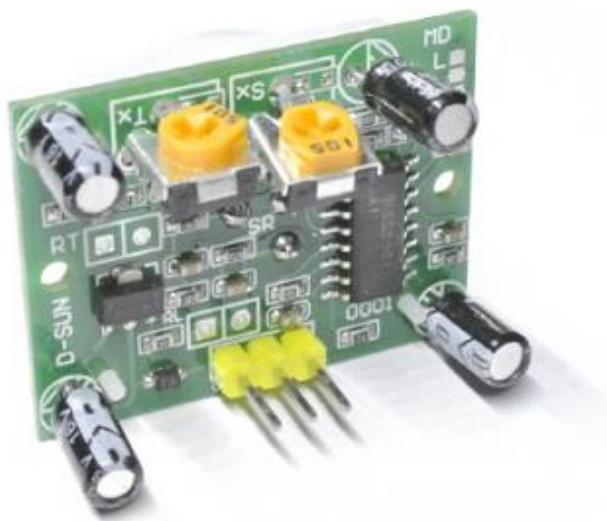


그림 53-2 센서 모듈 뒷면

인간의 몸이나, 동물에서 나오는 적외선은 약  $10 \mu\text{m}$  의 파장을 가지고 있습니다.  
PIR 센서는  $10 \mu\text{m}$ 의 적외선 파장 감지 하였을 때 센서가 반응하고 있습니다.

반응 신호는 전압의 형태로 받아서 처리하면 됩니다.

#### 인체감지 센서 기술사양:

Working voltage range: DC 4.5-20V

Static current: < 50uA

High level output: 3.3 V / low 0V

Trigger mode: L unrepeatable trigger /H repeated trigger (Default Repeating trigger)

Delay time: 0.5-200S (adjustable) can be produced range 0.1 seconds - a few minutes

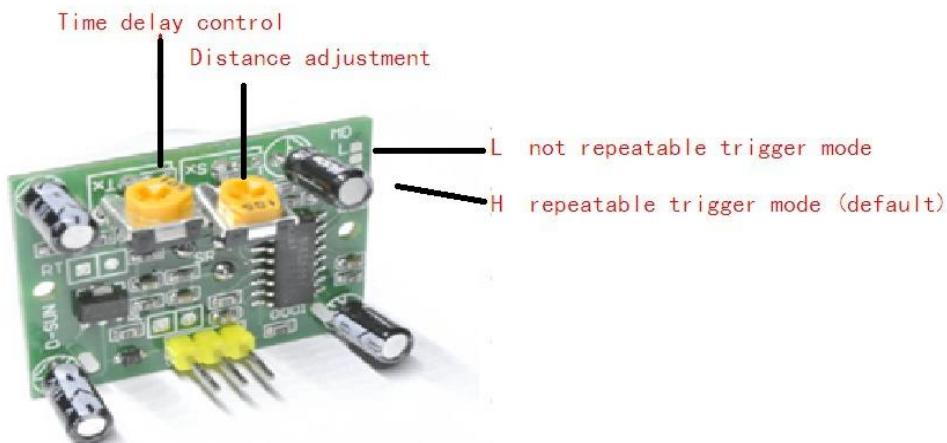
Blocking time: 2.5S (the default) can be produced range 0.1 seconds, tens of seconds

Size: 32mm\*24mm circuit board

Induction angle: <100 degree cone angle

Working temperature: -15~+70 degrees

Induction of lens size: diameter: 23mm (default)



1. Adjust distance potentiometer clockwise rotation, response distance increases (about 7 m); On the other hand, response distance decrease (about 3 m).

2. Adjust delay potentiometer clockwise rotation, response time delay increase (about 300 s); On the other hand, response time delay reduced (about 0.5 s)

적외선은 가시광선의 적색부분 바깥쪽에 해당하는 전자기파의 일종입니다.

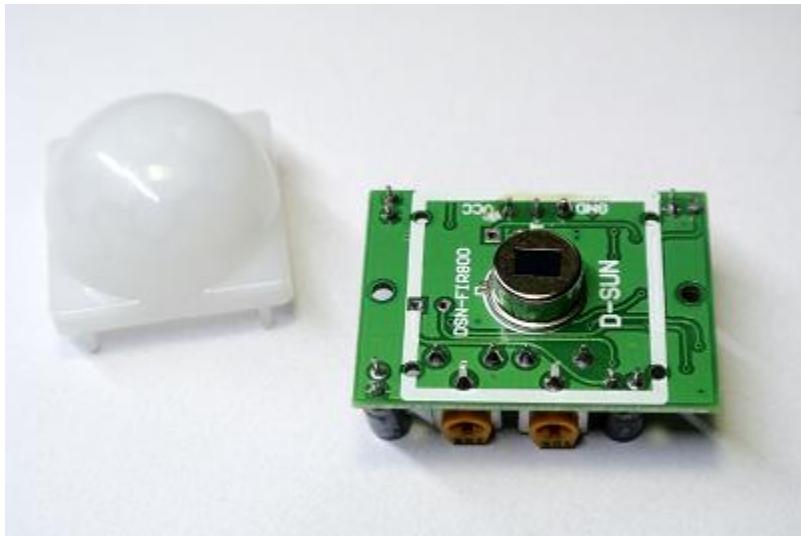


그림 53-1 센서 내부 참조 그림

#### 아두이노 연결:

| 인체감지센서 | 아두이노 |
|--------|------|
| OUT    | D2   |
| GND    | GND  |
| VS     | 5V   |

위의 가변저항을 적절히 돌려서 사용하면 됩니다. 감지 되었을 경우 계속적인 신호 지연 시간 설정과 감지 거리 조정을 할 수 있습니다.

예제코드: 센서 반응 아날로그 값을 지연시간으로 변경하여 LED 켜고 끄기. 아래의 코드는 가변저항의 조절, LED On/Off 동작 지연 시간 설정을 할 수 있습니다.

```
#define potPin A2 // 아날로그 포트 2 번
#define LED 13

int val=0; // 읽어들이 아날로그 포트의 값 저장 변수. 전역 변수.
void setup()
{
    pinMode(LED, OUTPUT); //LED 포트 출력 방향 설정.
    pinMode(potPin, INPUT);
}
void loop()
{
    val=analogRead(potPin); //
    delay(100);
    digitalWrite(LED, HIGH); //
    delay(val); //
```

```
    digitalWrite(LED, LOW); //  
    delay(val); //  
}
```

피에조 부저와 LED, 인체감지 센서를 이용하여 인체, 동물 등이 접근하면 소리와 LED 작동 예제입니다.  
(예제코드)

[http://www.gameplusedu.com/pds/gpshop/kit\\_code/strongest/pir\\_sensor\\_ex\\_1.ino](http://www.gameplusedu.com/pds/gpshop/kit_code/strongest/pir_sensor_ex_1.ino)

```
// Uses a PIR sensor to detect movement, buzzes a buzzer  
// more info here: http://blog.makezine.com/projects/pir-sensor-arduino-alarm/  
// email me, John Park, at jp@jpxl.net  
// based upon:  
// PIR sensor tester by Limor Fried of Adafruit  
// tone code by michael@thegrebs.com  
int ledPin = 13; // choose the pin for the LED  
int inputPin = 2; // choose the input pin (for PIR sensor)  
int pirState = LOW; // we start, assuming no motion detected  
int val = 0; // variable for reading the pin status  
int pinSpeaker = 10; //Set up a speaker on a PWM pin (digital 9, 10,  
or 11)  
  
void setup() {  
    pinMode(ledPin, OUTPUT); // declare LED as output  
    pinMode(inputPin, INPUT); // declare sensor as input  
    pinMode(pinSpeaker, OUTPUT);  
    Serial.begin(9600);  
}  
  
void loop(){  
    val = digitalRead(inputPin); // read input value  
    if (val == HIGH) { // check if the input is HIGH  
        digitalWrite(ledPin, HIGH); // turn LED ON  
        playTone(300, 160);  
        delay(150);  
  
        if (pirState == LOW) {  
            // we have just turned on  
            Serial.println("Motion detected!");  
            // We only want to print on the output change, not state  
            pirState = HIGH;  
        }  
    } else {
```

```
digitalWrite(ledPin, LOW); // turn LED OFF
playTone(0, 0);
delay(300);
if (pirState == HIGH){
    // we have just turned off
    Serial.println("Motion ended!");
    // We only want to print on the output change, not state
    pirState = LOW;
}
}

// duration in mSecs, frequency in hertz
void playTone(long duration, int freq) {
    duration *= 1000;
    int period = (1.0 / freq) * 1000000;
    long elapsed_time = 0;
    while (elapsed_time < duration) {
        digitalWrite(pinSpeaker,HIGH);
        delayMicroseconds(period / 2);
        digitalWrite(pinSpeaker, LOW);
        delayMicroseconds(period / 2);
        elapsed_time += (period);
    }
}
```

## 54 > ADJUSTABLE POTENTIOMETER X 1

가변저항 10K 입니다.

어저스터를 포텐시미터 10K / 조절 가능한 전위차계 부품입니다.



그림 54-1 10K 가변저항

가변저항으로 LED 밝기 조절하는 예제입니다.

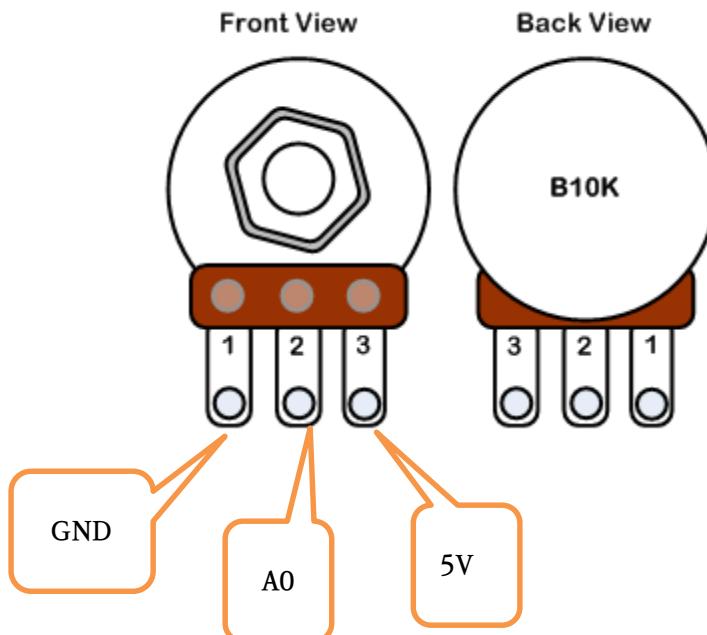


그림 54-1 10K 가변저항 �ин맵

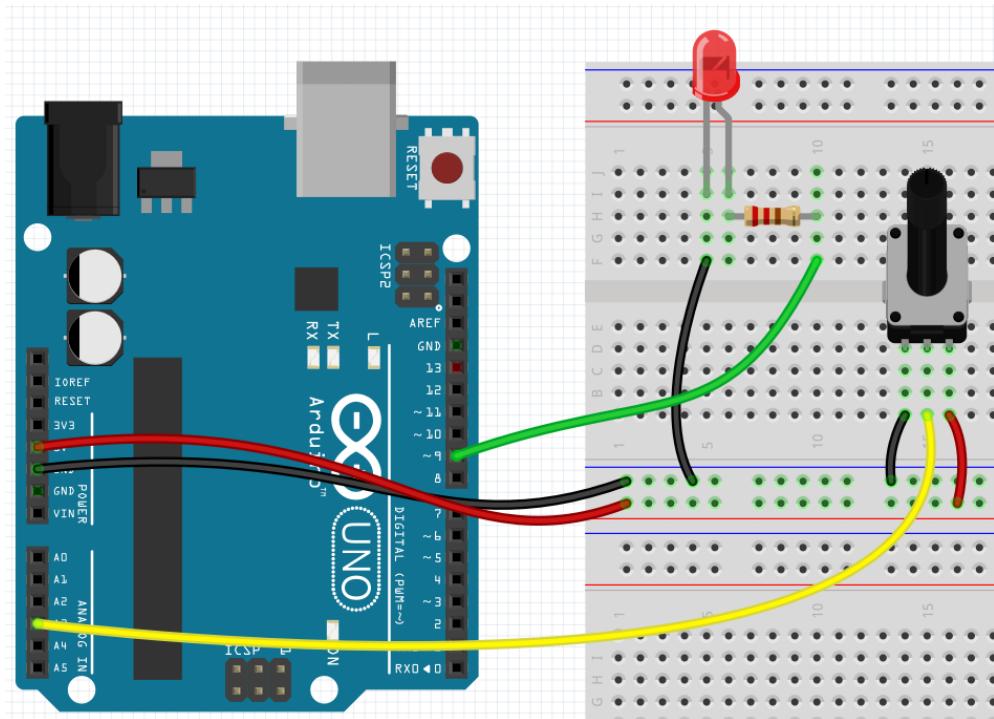


그림 54-2 가변저항 예제코드 브레드보드 회로도

아래의 코드는 `analogRead()`, `analogWrite()` 함수의 사용 예제입니다.

예제코드: 가변저항 10K 의 손잡이를 돌려서 9 번 포트에 연결된 LED 의 밝기 조절하는 예제 코드입니다.

```
// analogRead values go from 0 to 1023,
// LED connected to digital pin 9
// LED 를 아두이노 보드 13 번에 연결.
int ledPin = 9;

// potentiometer connected to analog pin 3
// 가변저항 10K 의 신호선을 아두이노 보드 3 번 핀에 연결
int analogPin = A3;

int val = 0;           // variable to store the read value

void setup()
{
    pinMode(ledPin, OUTPUT); // sets the pin as output
}
```

```
void loop()
{
    // read the input pin
    val = analogRead(analogPin);

    // analogRead values go from 0 to 1023, analogWrite values from 0 to
    255
    analogWrite(ledPin, val / 4);
}
```

## 55 > A DIGITAL CONTROL X 1



그림 55-1 1-Digit 7-Segment

1-Digit 7 Segment 모듈입니다.

숫자 하나당, 7 개 내부 LED 를 On/Off 시켜 표시하는 장치 또는 모듈입니다.

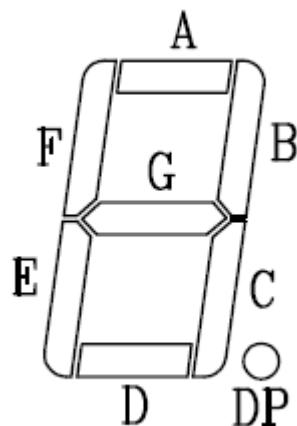


그림 55-2 내부 LED 인덱스

1-Digit 7 Segment 의 내부 LED 구성은 8 개의 작은 LED 가 있습니다.

추가 1 개의 LED 는 DOT 표시로 사용 됩니다. 결국 LED 8 개로 구성 되어 있지만 숫자 표시는 7 개 LED 사용 되므로 보통 7-segment 라고 부릅니다.

FND (Flexible Numeric Display) 라고 부릅니다.

마이크로 컨트롤러 교재, 문서 또는 최근엔 아두이노를 배울 때 꼭 거쳐야 되는 부분이 FND 모듈입니다.

FND 데이터시트 또는 설명 다이어그램을 보면 항상 COM 이라는 부분이 보입니다. COM 이라는 것은 COMMON, 즉 공통 핀 개념으로 내부 LED의 구조에 따라 GND, 또는 VCC 가 될 수 도 있습니다.

사용시 데이터 시트를 참조해야 하는 부분입니다.

COMMON 핀이 VCC 인 경우에는 커먼 애노드 타입, COMMON 핀이 GND 경우에 커먼 캐소드라고 합니다.

애노드, 캐소드는 꼭 알아야 할 단어이기도 합니다.

이미 아시는 단어이겠지만, Anode 를 (+), Cathode 는 (-) 입니다.

즉 단어 뜻을 이해하신다면 FND 이해가 쉽게 되리라 봅니다.

캐소드 - com7-Segment 는 양극 입력 시 발광하는 캐소드(Cathode) 타입과 음극 입력 시 발광하는 애노드(Anode), 본 실습에는 캐소드 타입을 이용 합니다

애노드 타입: 공통 핀 + 이므로 나머지 LED 는 - 연결 후 On/Off

캐소드 타입: 공통 핀 - 이므로 나머지 LED 는 + 연결 후 On/Off

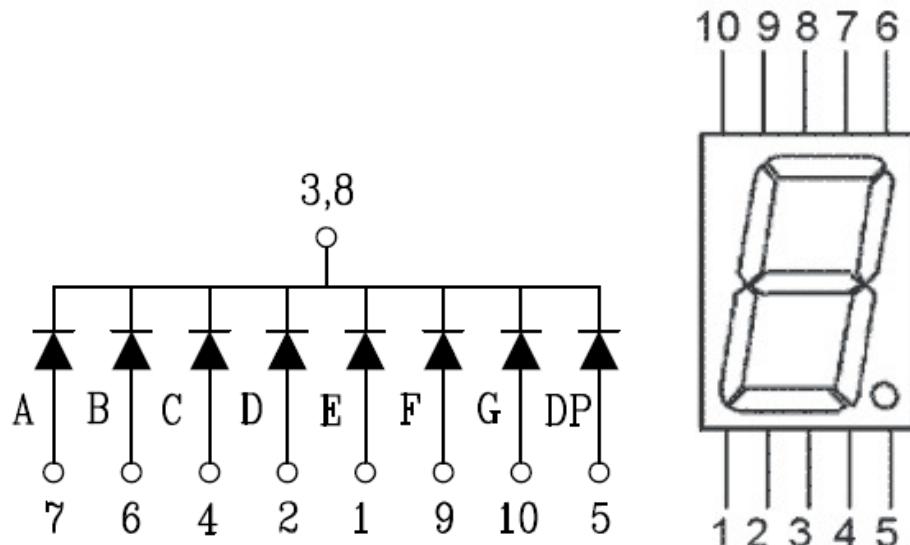
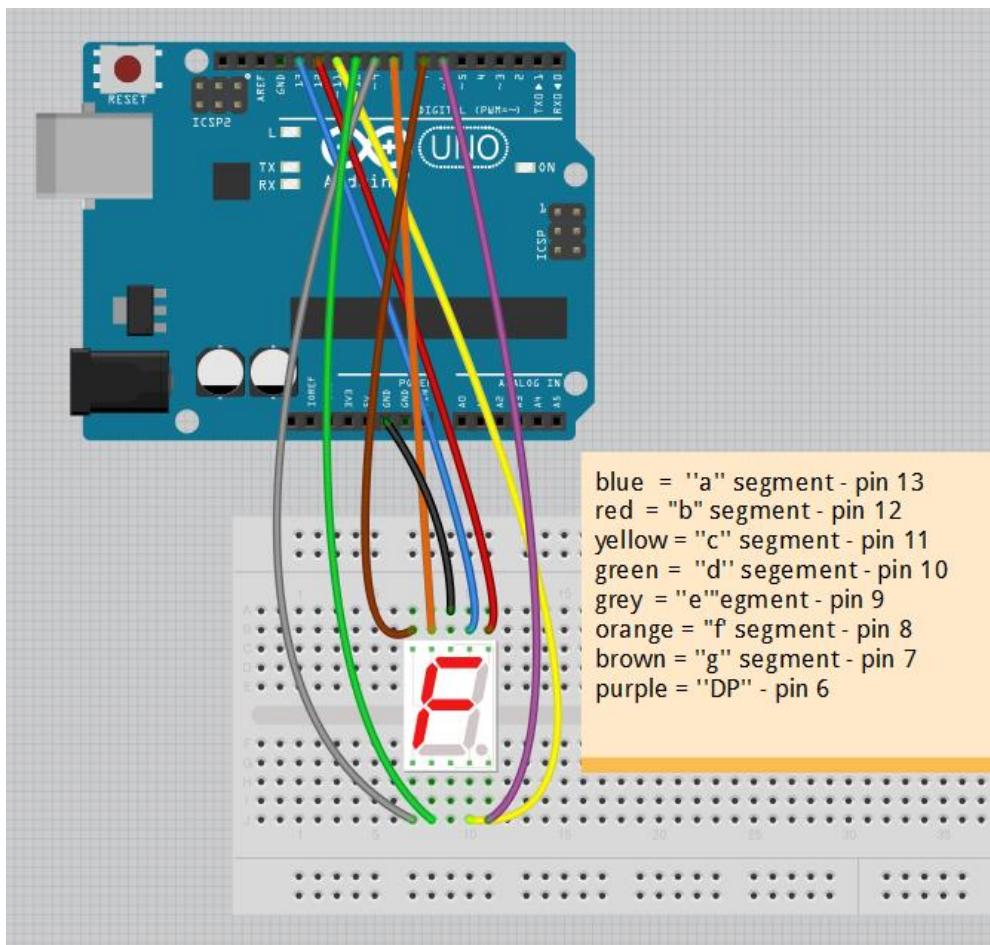


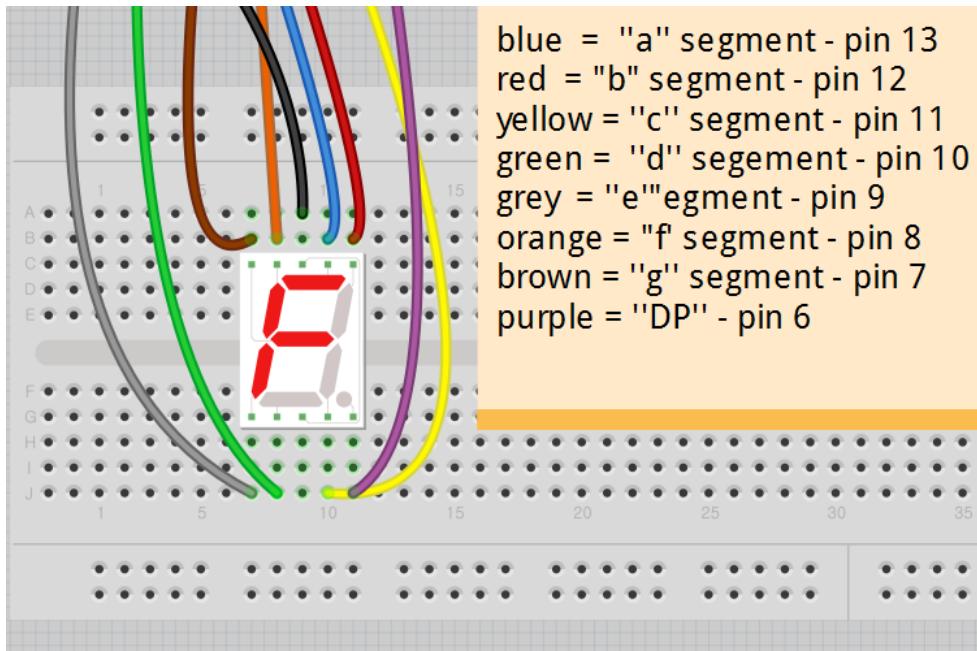
그림 55-3 7-Segment 회로도

DataSheet:

<http://www.gameplusedu.com/pds/gpshop/arduino/pdf/7-segment-2I17KS85.pdf>

아래는 아두이노 우노에서 커먼 캐소드 FND 모듈 와이어링 설명 그림입니다.





예제코드)

[http://www.gameplusedu.com/pds/gpshop/kit\\_code/strongest/7segment\\_1digit.ino](http://www.gameplusedu.com/pds/gpshop/kit_code/strongest/7segment_1digit.ino)

```
//  
// author : http://www.gameplusedu.com  
// contact point : gameplusedu@gmail.com  
  
// 시리얼 포트로 번호를 입력 받아 FND에 숫자를 표시하는 예제입니다.  
// 연결 펀을 정의합니다.  
  
int a = 13;  
int b = 12;  
int c = 11;  
int d = 10;  
int e = 9;  
int f = 8;  
int g = 7;  
int dp = 6;  
  
void setup() {  
    Serial.begin(9600); // uart0 속도 지정.
```

```
// 사용되는 포트 방향을 모두 OUTPUT 으로 합니다.
pinMode(a , OUTPUT);
pinMode(b , OUTPUT);
pinMode(c , OUTPUT);
pinMode(d , OUTPUT);
pinMode(e , OUTPUT);
pinMode(f , OUTPUT);
pinMode(g , OUTPUT);
pinMode(dp , OUTPUT);
}

void loop()
{
    int val = Serial.read() - '0';
    if (val == 0){
        digitalWrite(a , HIGH);
        digitalWrite(b , HIGH);
        digitalWrite(c , HIGH);
        digitalWrite(d , HIGH);
        digitalWrite(e , HIGH);
        digitalWrite(f , HIGH);
        digitalWrite(dp , HIGH);
        delay(1000);
        digitalWrite(a , LOW);
        digitalWrite(b , LOW);
        digitalWrite(c , LOW);
        digitalWrite(d , LOW);
        digitalWrite(e , LOW);
        digitalWrite(f , LOW);
        digitalWrite(dp , LOW);
    }
    if (val == 1){
        digitalWrite(b , HIGH);
        digitalWrite(c , HIGH);
        digitalWrite(dp , HIGH);
        delay(1000);
        digitalWrite(b , LOW);
        digitalWrite(c , LOW);
        digitalWrite(dp , LOW);
    }
}
```

```
}

if (val == 2){
    digitalWrite(a , HIGH);
    digitalWrite(b , HIGH);
    digitalWrite(d , HIGH);
    digitalWrite(e , HIGH);
    digitalWrite(g , HIGH);
    digitalWrite(dp , HIGH);
    delay(1000);
    digitalWrite(a , LOW);
    digitalWrite(b , LOW);
    digitalWrite(d , LOW);
    digitalWrite(e , LOW);
    digitalWrite(g , LOW);
    digitalWrite(dp , LOW);
}

if (val == 3){
    digitalWrite(a , HIGH);
    digitalWrite(b , HIGH);
    digitalWrite(g , HIGH);
    digitalWrite(c , HIGH);
    digitalWrite(d , HIGH);
    digitalWrite(dp , HIGH);
    delay(1000);
    digitalWrite(a , LOW);
    digitalWrite(b , LOW);
    digitalWrite(g , LOW);
    digitalWrite(c , LOW);
    digitalWrite(d , LOW);
    digitalWrite(dp , LOW);
}

if (val == 4){
    digitalWrite(b , HIGH);
    digitalWrite(c , HIGH);
    digitalWrite(f , HIGH);
    digitalWrite(g , HIGH);
    digitalWrite(dp , HIGH);
    delay(1000);
    digitalWrite(b , LOW);
    digitalWrite(c , LOW);
}
```

```
digitalWrite(f , LOW);
digitalWrite(g ,LOW);
digitalWrite(dp , LOW);
}
if (val == 5) {
    digitalWrite(a , HIGH);
    digitalWrite(c , HIGH);
    digitalWrite(d , HIGH);
    digitalWrite(f , HIGH);
    digitalWrite(g , HIGH);
    digitalWrite(dp , HIGH);
    delay(1000);
    digitalWrite(a , LOW);
    digitalWrite(c , LOW);
    digitalWrite(d , LOW);
    digitalWrite(f , LOW);
    digitalWrite(g ,LOW);
    digitalWrite(dp , LOW);
}
if (val == 6) {
    digitalWrite(a , HIGH);
    digitalWrite(c , HIGH);
    digitalWrite(d , HIGH);
    digitalWrite(e , HIGH);
    digitalWrite(f , HIGH);
    digitalWrite(g , HIGH);
    digitalWrite(dp , HIGH);
    delay(1000);
    digitalWrite(a , LOW);
    digitalWrite(c , LOW);
    digitalWrite(d , LOW);
    digitalWrite(e , LOW);
    digitalWrite(f , LOW);
    digitalWrite(g ,LOW);
    digitalWrite(dp , LOW);
}
if (val == 7){
    digitalWrite(a , HIGH);
    digitalWrite(b , HIGH);
    digitalWrite(c , HIGH);
```

```
digitalWrite(dp , HIGH);
delay(1000);
digitalWrite(a , LOW);
digitalWrite(b , LOW);
digitalWrite(c , LOW);
digitalWrite(dp , LOW);
}
if (val == 8){
    digitalWrite(a , HIGH);
    digitalWrite(b , HIGH);
    digitalWrite(c , HIGH);
    digitalWrite(d , HIGH);
    digitalWrite(e , HIGH);
    digitalWrite(f , HIGH);
    digitalWrite(g , HIGH);
    digitalWrite(dp , HIGH);
    delay(1000);
    digitalWrite(a , LOW);
    digitalWrite(b , LOW);
    digitalWrite(c , LOW);
    digitalWrite(d , LOW);
    digitalWrite(e , LOW);
    digitalWrite(f , LOW);
    digitalWrite(g , LOW);
    digitalWrite(dp , LOW);
}
if (val == 9){
    digitalWrite(a , HIGH);
    digitalWrite(b , HIGH);
    digitalWrite(c , HIGH);
    digitalWrite(d , HIGH);
    digitalWrite(f , HIGH);
    digitalWrite(g , HIGH);
    digitalWrite(dp , HIGH);
    delay(1000);
    digitalWrite(a , LOW);
    digitalWrite(b , LOW);
    digitalWrite(c , LOW);
    digitalWrite(d , LOW);
    digitalWrite(f , LOW);
```

```
    digitalWrite(g ,LOW);
    digitalWrite(dp , LOW);
}
}
```

## 56 > 4 DIGITAL TUBE X 1, 4-DIGIT 7-SEGMENT

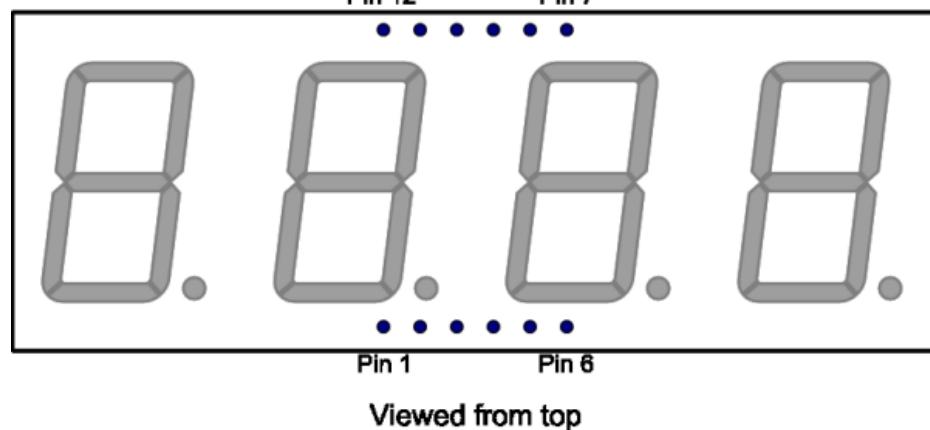
4-Digit 7-Segment 입니다.

4 개의 숫자 & DOT 표시 가능한 모듈입니다.



캐소드 타입 4 자리 LED 모듈입니다.

핀 번호는 아래와 같습니다.



4-Digit 모듈 모델 넘버 인쇄된 부분 (SH5461AS 찍혀 있는 부분) 아래쪽으로 향하게 하고 핀 번호는 위의 이미지를 참조합니다.

데이터시트는 아래의 링크를 참조합니다.(기술지원 게시판에도 있습니다)

[http://www.igameplus.com/pds/gpshop/arduino/shield\\_module/4digit-7segment/4digit-7segment-datasheet.pdf](http://www.igameplus.com/pds/gpshop/arduino/shield_module/4digit-7segment/4digit-7segment-datasheet.pdf)

데이터쉬트 내용 중 Circuit Diagram 을 확인합니다.

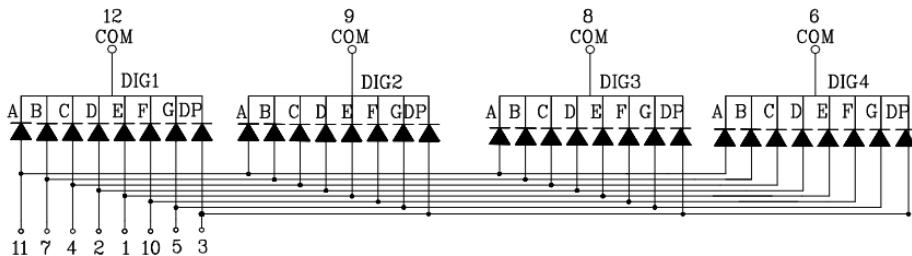


그림 56-1 4-Digit 7-Segment Circuit

1-Digit 7-Segment 모듈이 4 개 12,9,8,6 핀이 Common 핀이면서 그라운드 표시입니다. 캐소드 방식의 7-Segment 입니다. 그라운드 표시가 12,9,8,6 번입니다.  
참고사항으로 아래의 이미지는 1-Digit 7-Segment 의 회로도 입니다.

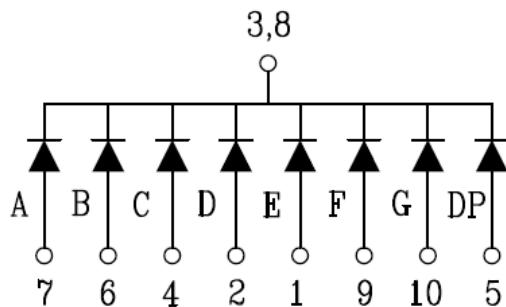


그림 56-2 1-Digit 7-Segment Circuit

4-Digit 표시 모듈은 1-Digit 4 개를 병렬 연결한 것을 알게 됩니다.  
그리고 12,9,8,6 번 핀은 1-Digit 의 3,8 번 핀의 역할을 하게 됩니다.

이제 아두이노 보드와 연결을 합니다.

4 Digit 7 Segment 1 번부터 12 번까지의 핀이 있습니다.

**주의 사항:** 7-Segment 모듈의 각각의 핀에 아두이노 우노 보드의 GND 포트 옆에 있는 5V 출력을 바로 연결하면 안됩니다. 높은 전압/전류 들어가는 경우, 해당 연결된 LED 부분이 변(Burn)됩니다. 안전하게 장시간 사용하기 위해서는 7-Segment 의 각각의 (+) 연결 포트에는 220 Ohm 저항을 연결해야 합니다.

아두이노 우노 보드의 D0~D13, A0~A5 포트의 전압 출력은 신호 용도의 전압과 전류가 사용됩니다. 7-Segment (+) 연결 포트에 연결하여도 무방합니다.

하지만 장시간 설치 및 사용시에는 반드시 220Ohm 정도의 저항을 사용하도록 합니다.

각각의 핀 기능은 아래의 기능 아두이노와의 연결은 위의 이미지를 참조하여 아래의 표와 같이 합니다.

| 모듈 핀 | 아두이노 | Segment |
|------|------|---------|
| 1    | D10  | segE    |
| 2    | D9   | segD    |
| 3    | D13  | segDP   |
| 4    | D8   | segC    |
| 5    | D12  | segG    |
| 6    | D5   |         |
| 7    | D7   | segB    |
| 8    | D4   |         |
| 9    | D3   |         |
| 10   | D11  | segF    |
| 11   | D6   | segA    |
| 12   | D2   |         |

아두이노 공식 사이트의 링크 SevSeg 라이브러리를 사용 하도록 합니다.

7-Segment 를 편리하게 사용하도록 지원되는 클래스 라이브러리입니다.

7-Segment 라이브러리:

[http://www.gameplusedu.com/pds/gpshop/kit\\_code/strongest/SevSeg.zip](http://www.gameplusedu.com/pds/gpshop/kit_code/strongest/SevSeg.zip)

라이브러리 적용 후 스케치 코드에서는

#include <SevSeg.h> 선언하고 사용 하게 됩니다.

아두이노&스케치 프로그램의 장점은 기 구현된 많은 라이브러리들이 존재 한다는 것입니다. 최소한의 C/C++ 언어 이해도만으로도 많은 것들을 활용 해볼 수도 있습니다.

스케치 예제 코드입니다.

[http://www.gameplusedu.com/pds/gpshop/kit\\_code/strongest/7segment\\_4digit.ino](http://www.gameplusedu.com/pds/gpshop/kit_code/strongest/7segment_4digit.ino)

```
// 7-Segment 라이브러리 사용, 헤더파일 선언.  
#include <SevSeg.h>  
  
//Create an instance of the object.  
// 글로벌 클래스 변수입니다. (전역 클래스 변수)  
SevSeg myDisplay;
```

```

//Create global variables
unsigned long timer;
int decisecond = 0;

void setup()
{
    int displayType = COMMON_CATHODE; //캐소드 모듈 지정.

    //This pinout is for a regular display
    int digit1 = 2; //Pin 12 on my 4 digit display
    int digit2 = 3; //Pin 9 on my 4 digit display
    int digit3 = 4; //Pin 8 on my 4 digit display
    int digit4 = 5; //Pin 6 on my 4 digit display

    //Declare what pins are connected to the segments
    int segA = 6; //Pin 11 on my 4 digit display
    int segB = 7; //Pin 7 on my 4 digit display
    int segC = 8; //Pin 4 on my 4 digit display
    int segD = 9; //Pin 2 on my 4 digit display
    int segE = 10; //Pin 1 on my 4 digit display
    int segF = 11; //Pin 10 on my 4 digit display
    int segG = 12; //Pin 5 on my 4 digit display
    int segDP= 13; //Pin 3 on my 4 digit display

    int numberOfDigits = 4; //Do you have a 1, 2 or 4 digit display?

    myDisplay.Begin(displayType, numberOfDigits, digit1, digit2, digit3,
    digit4, segA, segB, segC, segD, segE, segF, segG, segDP);

    myDisplay.SetBrightness(100); //Set the display to 100% brightness level

    timer = millis();
}

void loop()
{
    //Example ways of displaying a decimal number
    char tempString[10]; //Used for sprintf
}

```

```

sprintf(tempString, "%04d", deciSecond); //Convert deciSecond into a
string that is right adjusted
    //sprintf(tempString, "%d", deciSecond); //Convert deciSecond into a
string that is left adjusted
    //sprintf(tempString, "%04d", deciSecond); //Convert deciSecond into a
string with leading zeros
    //sprintf(tempString, "%4d", deciSecond * -1); //Shows a negative sign
infront of right adjusted number
    //sprintf(tempString, "%4X", deciSecond); //Count in HEX, right adjusted

    //Produce an output on the display
myDisplay.DisplayString(tempString, 8); //(numberToDisplay, decimal
point location)

    //Other examples
    //myDisplay.DisplayString(tempString, 0); //Display string, no decimal
point
    //myDisplay.DisplayString("-23b", 3); //Display string, decimal point in
third position

    //Check if 100 ms has elapsed // 100ms (10 분 1 초) 경과시간 체크.
    // 100 ms, 0.1 초
    if (millis() - timer >= 100)
    {
        timer = millis();
        deciSecond++;
    }

    delay(5);
}

```

와이어링 및 업로드 정상 상태인 경우 숫자 증가 하는 것을 볼 수 있습니다.

위의 코드중 myDisplay.DisplayString(tempString, 8);

DisplayString 이라는 함수가 있습니다.

SevSeg.h 파일을 열어보면

void SevSeg::DisplayString(char\* toDisplay, byte DecAposColon); 라고 정의 되어 있습니다. 2 번째 파라메터는 DOT 표시입니다.  
지정된 위치에 점(DOT)을 표시하게 됩니다.

```
myDisplay.DisplayString(tempString, 0); // DOT 표시 안함.  
myDisplay.DisplayString(tempString, 1); // DOT 표시 1 번째 위치.  
myDisplay.DisplayString(tempString, 2); // DOT 표시 2 번째 위치.  
myDisplay.DisplayString(tempString, 4); // DOT 표시 3 번째 위치.  
myDisplay.DisplayString(tempString, 8); // DOT 표시 4 번째 위치.
```

2 번째와 3 번째 위치에 DOT 을 표시하고 싶다면

```
myDisplay.DisplayString(tempString, 2 | 4); // OR 연산자 or 6
```

## 57 > 8 \* 8 DOT MATRIX MODULE X 1



그림 57-1 8 x 8 DOT MATRIX

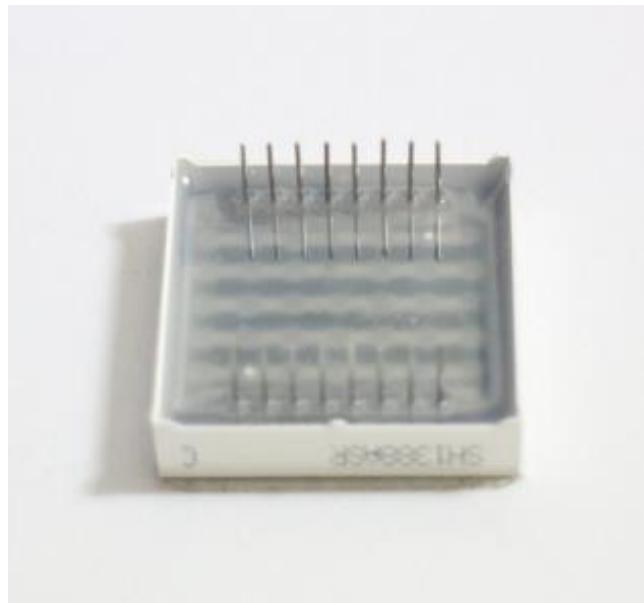


그림 57-2 DOT MATRIX 아랫면

8 x 8 사이즈의 DOT MATRIX 모듈입니다.  
가로 8 칸 세로 8 칸으로 LED 가 교차 연결된 모듈입니다.

Dot: 3mm

색상: RED

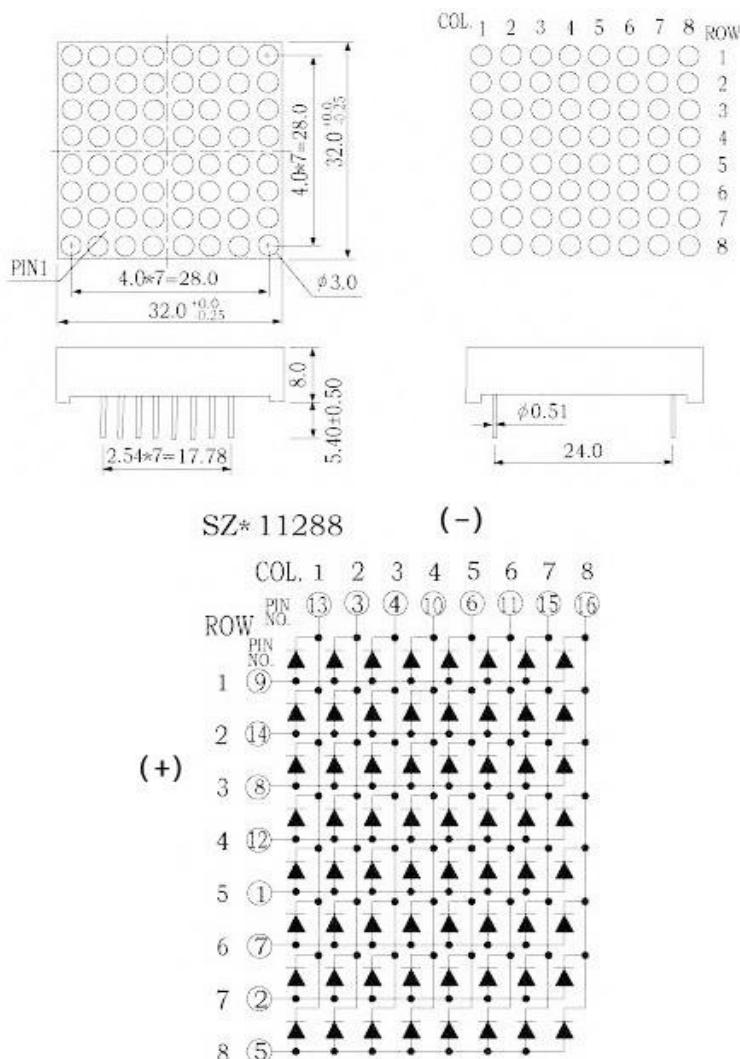
크기: 32mm x 32 mm x 8 mm(height)

8x8 pixel로 표현 가능한 모든 것을 디스플레이 할 수 있습니다.

64 개의 교차 방식의 LED로 구성된 모듈입니다.

고급 디지털시계, 시스템 상황 표시(설명한), 경고 표시 등등 여러 가지 용도로 사용되고 있습니다. 실제 설치 및 제품화될 때의 LED 모듈은 LED 사이즈 대, 중, 소 등 의 크기가 용도별로 다를 뿐 구현 내용이나 모듈 연결 방식은 비슷합니다.

아래의 참조 회로도를 봅니다.



전체 크기와 핀 간격, 그리고 핀 번호, 직병렬 회로도가 있습니다. 세모꼴 모양은 LED 다이오드 표시입니다.

“LED”, 정확한 의미는 “Light Emitting Diode”

즉, 빛 발산 2극관, 발광 다이오드입니다. 즉 (+), (-)소자입니다. 회로도의 세모꼴 모양을 보시면 (+)와 (-) 정확히 구분된 것을 보실 수 있습니다.

LED 는 아시다시피 (+),(-) 연결 해야 켜집니다.

아래의 회로도를 보면 ROW 는 (+), COL 은 (-) 되어 있습니다.

COL 3 번째, ROW 5 번째의 LED 를 켜고 싶다면, ROW 5 번째 (+)를, COL 3 번째를 (-) 연결 해주면 됩니다. 아두이노 보드에 연결 되었다면 ROW 5 번째 핀을 HIGH, COL 3 번째 핀을 LOW 로 하면 됩니다.

이제 아두이노 우노 R3 에서 컨트롤 하기 위해서 16 핀 모두 사용 합니다.

그런데, 아두이노 우노 R3 는 디지털핀은 0~13, 14 개 입니다. 2 개 모자랍니다.

결국 아날로그 핀도 사용합니다.

대부분 사용되는 핀수를 줄이기 위해서 74HC595 종류의 칩을 사용 합니다.

74HC595 칩은 3 개의 핀으로 8 개의 OUTPUT 신호를 컨트롤 합니다.

그래도 6 핀을 사용하게 됩니다. 결국, 2 개의 74HC595 필요하게 됩니다.

본 예제에서는 직접(다이렉트로) 모두 연결 해보도록 합니다. 아날로그 핀도 디지털 핀처럼 사용할 수 있습니다. 아래의 그림처럼 보통 핀 번호는 마크, 모델 명칭이 있는 부분의 맨 왼쪽부터 1 번째 번호입니다.

1 번부터 끝 번호까지는 반시계 방향 (CCW)으로 순차적으로 부여 해서 사용합니다.

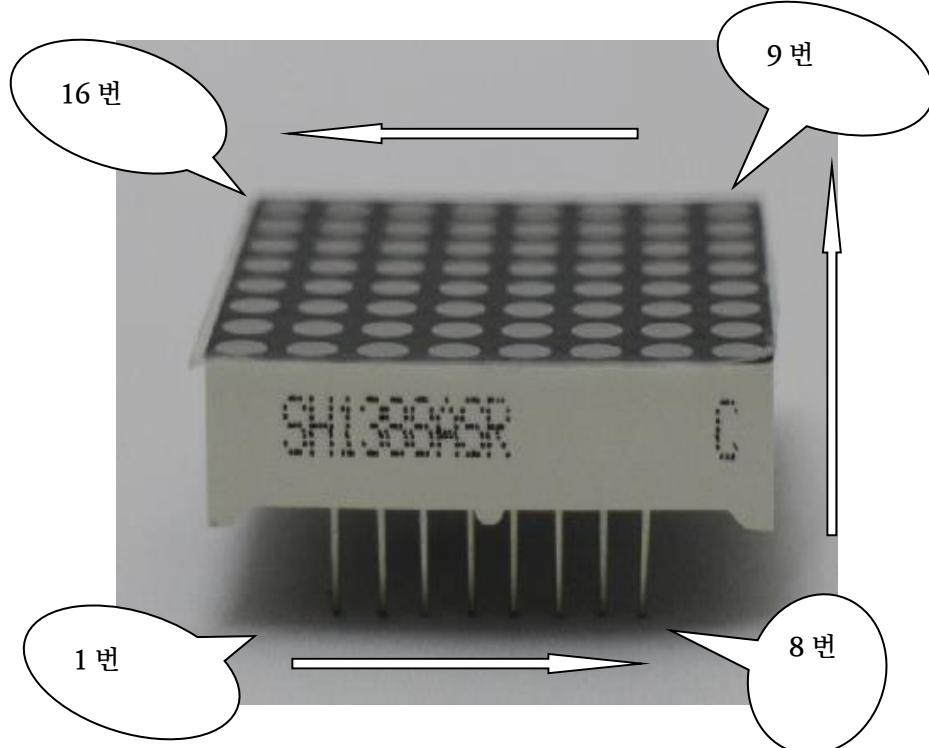
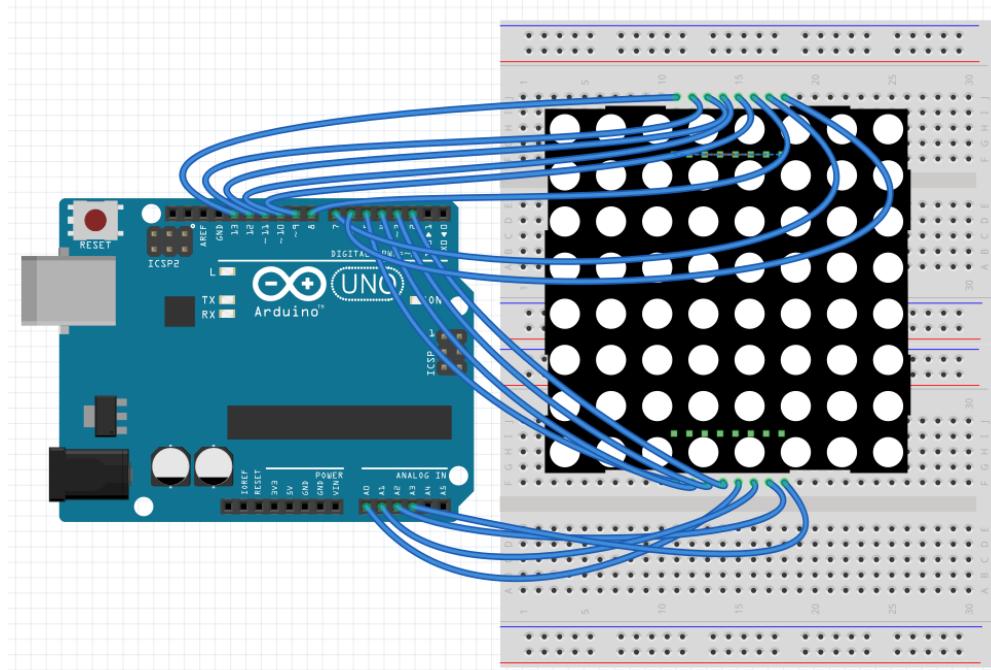


그림 57-3 핀 번호 순서

아두이노 보드와의 연결은 아래의 표와 그림을 참조합니다.

5, 4, 3, 2, 14, 15, 16, 17, 13, 12, 11, 10, 9, 8, 7, 6

| 모듈 편 번호 | 아두이노    |
|---------|---------|
| 1       | 5       |
| 2       | 4       |
| 3       | 3       |
| 4       | 2       |
| 5       | A0 (14) |
| 6       | A1 (15) |
| 7       | A2 (16) |
| 8       | A3 (17) |
| 9       | 13      |
| 10      | 12      |
| 11      | 11      |
| 12      | 10      |
| 13      | 9       |
| 14      | 8       |
| 15      | 7       |
| 16      | 6       |



## 그림 57-4 브레드보드 연결 그림

200~220 Ohm 저항을 사용해야 합니다. 사용하는 이유는 아두이노에서 HIGH 신호 전류가 5V 가량 나옵니다. LED 는 거의 대부분 2V~3V 동작하기 때문에 높은 전압으로 사용하게 되면 오동작 및 에러가 발생될 수 있습니다. 정확하고 안전한 동작을 위해서는 전압을 조절 해 주기 위해 저항을 사용합니다. 그럼 220 R Ohm 또는 300 Ohm 저항을 사용하는지 알아 봅니다.

저항 계산식은 아래와 같습니다.

기호 범례: V: 전압, I: 전류(A), R: 저항(옴), P: 전력(w)

$$P=VI \text{ (전력=전압} \times \text{전류)}$$

$$V=IR \text{ (전압=전류} \times \text{저항)}$$

$$I=V/R \text{ (전류=전압/저항)}$$

$$R=V/I \text{ (저항=전압/전류)}$$

아두이노 보드의 핀 HIGH 신호는 5V 출력 됩니다.

5V 전압에 2V 10mA LED 를 사용 한다면

$5V - 2V = 3V$  즉, 5V 전력에 LED 에 적용되는 전력은 2V 이고 나머지 3V 저항에서 처리해 주도록 합니다.

LED 모듈은 2V 10mA 동작 전원이라면

$$R=V/I \rightarrow 3V / 0.01A = 200 \text{ Ohm} \text{ 계산됩니다.}$$

8x8 DOT 매트릭스 예제 코드입니다.

KIT 예제 중 가장 어려운 HELLO 예제 이기도 합니다.

HELLO 라는 각각의 구성 문자를 사이드 스크롤 하는 예제입니다.

보통 옥외 LED 간판에 많이 사용되는(오른쪽->왼쪽) 스크롤 방식입니다.

예제코드:

[http://www.gameplusedu.com/pds/gpshop/kit\\_code/strongest/led\\_8by8.ino](http://www.gameplusedu.com/pds/gpshop/kit_code/strongest/led_8by8.ino)

```
#include <FrequencyTimer2.h>

#define SPACE { \
    {0, 0, 0, 0, 0, 0, 0, 0}, \
    {0, 0, 0, 0, 0, 0, 0, 0}, \
    {0, 0, 0, 0, 0, 0, 0, 0}, \
    {0, 0, 0, 0, 0, 0, 0, 0}, \
    {0, 0, 0, 0, 0, 0, 0, 0}, \
    {0, 0, 0, 0, 0, 0, 0, 0}, \
    {0, 0, 0, 0, 0, 0, 0, 0}, \
    {0, 0, 0, 0, 0, 0, 0, 0} \
}
```

```

#define H { \
    {0, 1, 0, 0, 0, 0, 1, 0}, \
    {0, 1, 0, 0, 0, 0, 1, 0}, \
    {0, 1, 0, 0, 0, 0, 1, 0}, \
    {0, 1, 1, 1, 1, 1, 1, 0}, \
    {0, 1, 0, 0, 0, 0, 1, 0}, \
    {0, 1, 0, 0, 0, 0, 1, 0}, \
    {0, 1, 0, 0, 0, 0, 1, 0} \
}

#define E { \
    {0, 1, 1, 1, 1, 1, 1, 0}, \
    {0, 1, 0, 0, 0, 0, 0, 0}, \
    {0, 1, 0, 0, 0, 0, 0, 0}, \
    {0, 1, 1, 1, 1, 1, 1, 0}, \
    {0, 1, 0, 0, 0, 0, 0, 0}, \
    {0, 1, 0, 0, 0, 0, 0, 0}, \
    {0, 1, 0, 0, 0, 0, 0, 0}, \
    {0, 1, 1, 1, 1, 1, 1, 0} \
}

#define L { \
    {0, 1, 0, 0, 0, 0, 0, 0, 0}, \
    {0, 1, 0, 0, 0, 0, 0, 0, 0}, \
    {0, 1, 0, 0, 0, 0, 0, 0, 0}, \
    {0, 1, 0, 0, 0, 0, 0, 0, 0}, \
    {0, 1, 0, 0, 0, 0, 0, 0, 0}, \
    {0, 1, 0, 0, 0, 0, 0, 0, 0}, \
    {0, 1, 0, 0, 0, 0, 0, 0, 0}, \
    {0, 1, 1, 1, 1, 1, 1, 1, 0} \
}

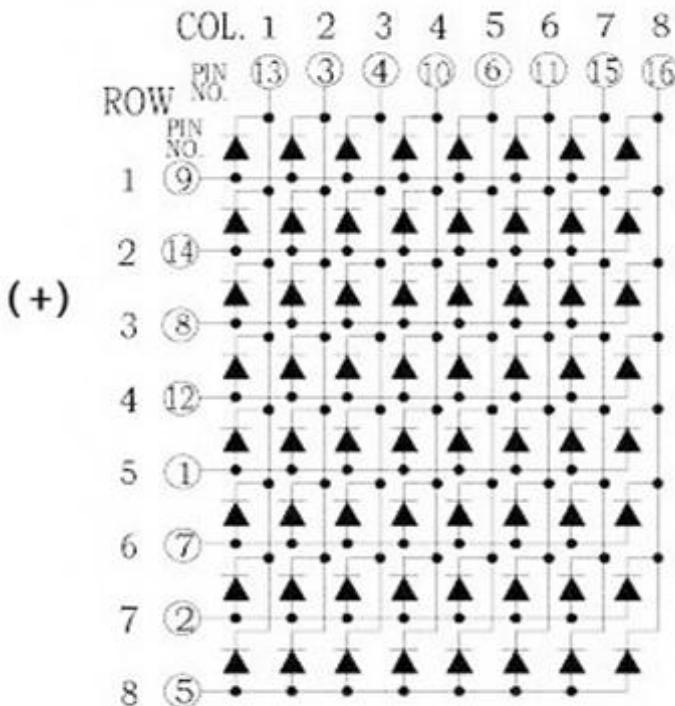
#define O { \
    {0, 0, 0, 1, 1, 0, 0, 0, 0}, \
    {0, 0, 1, 0, 0, 1, 0, 0, 0}, \
    {0, 1, 0, 0, 0, 0, 1, 0, 0}, \
    {0, 1, 0, 0, 0, 0, 1, 0, 0}, \
    {0, 1, 0, 0, 0, 0, 1, 0, 0}, \
    {0, 1, 0, 0, 0, 0, 1, 0, 0}, \
    {0, 0, 1, 0, 0, 1, 0, 0, 0}, \
    {0, 0, 0, 1, 1, 0, 0, 0, 0} \
}

byte col = 0;
byte leds[8][8];

```

```
// pin[xx] on led matrix connected to nn on Arduino (-1 is dummy to make array start at pos 1)
int pins[17]={-1, 5, 4, 3, 2, 14, 15, 16, 17, 13, 12, 11, 10, 9, 8, 7, 6};
```

// 아래의 코드는 위에서 보았던 내부 핀 순서를 참조합니다.



```
// col[xx] of leds = pin yy on led matrix
// 행 8 개 순서대로 지정합니다.
//
int cols[8] = {pins[13], pins[3], pins[4], pins[10], pins[6], pins[11], pins[15], pins[16]};

// row[xx] of leds = pin yy on led matrix
// 열 8 개 순서대로 지정해줍니다.
int rows[8] = {pins[9], pins[14], pins[8], pins[12], pins[1], pins[7], pins[2], pins[5]};

const int numPatterns = 6;
byte patterns[numPatterns][8][8] = {
    H,E,L,L,O,SPACE
};
```

```

int pattern = 0;

void setup() {
    // sets the pins as output
    for (int i = 1; i <= 16; i++) {
        pinMode(pins[i], OUTPUT);
    }

    // set up cols and rows
    for (int i = 1; i <= 8; i++) {
        digitalWrite(cols[i - 1], !LOW);
    }

    for (int i = 1; i <= 8; i++) {
        digitalWrite(rows[i - 1], !LOW);
    }

    clearLeds();
}

// Turn off toggling of pin 11
FrequencyTimer2::disable();
// Set refresh rate (interrupt timeout period)
FrequencyTimer2::setPeriod(2000);
// Set interrupt routine to be called
FrequencyTimer2::setOnOverflow(display);

setPattern(pattern);
}

void loop() {
    pattern = ++pattern % numPatterns;
    slidePattern(pattern, 100);
}

void clearLeds() {
    // Clear display array
    for (int i = 0; i < 8; i++) {
        for (int j = 0; j < 8; j++) {
            leds[i][j] = 0;
        }
    }
}

void setPattern(int pattern) {
    for (int i = 0; i < 8; i++) {

```

```

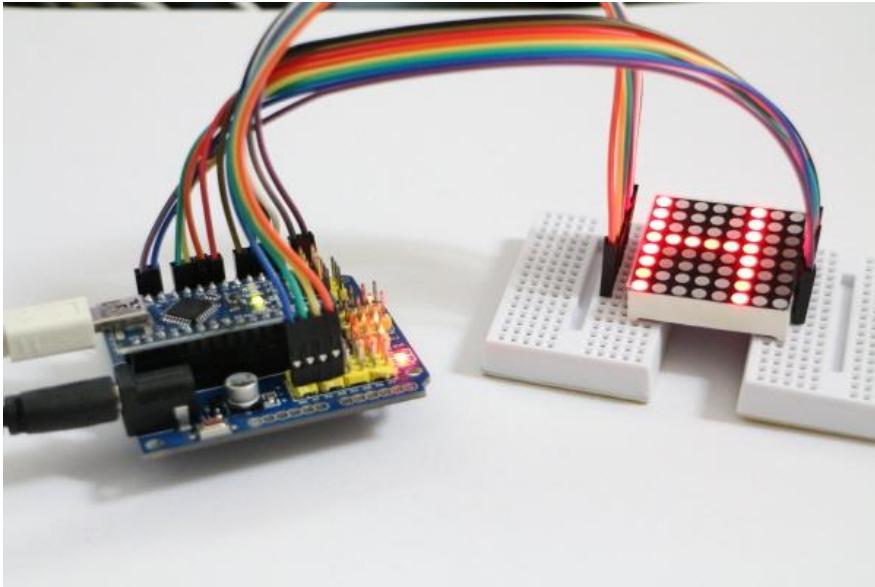
        for (int j = 0; j < 8; j++) {
            leds[i][j] = patterns[pattern][i][j];
        }
    }

void slidePattern(int pattern, int del) {
    for (int l = 0; l < 8; l++) {
        for (int i = 0; i < 7; i++) {
            for (int j = 0; j < 8; j++) {
                leds[j][i] = leds[j][i+1];
            }
        }
        for (int j = 0; j < 8; j++) {
            leds[j][7] = patterns[pattern][j][0 + l];
        }
        delay(del);
    }
}

// Interrupt routine
void display() {
    digitalWrite(cols[col], !LOW); // Turn whole previous column off
    col++;
    if (col == 8) {
        col = 0;
    }
    for (int row = 0; row < 8; row++) {
        if (leds[col][7 - row] == 1) {
            digitalWrite(rows[row], !LOW); // Turn on this led
        }
        else {
            digitalWrite(rows[row], !HIGH); // Turn off this led
        }
    }
    digitalWrite(cols[col], !HIGH); // Turn whole column on at once (for equal
lighting times)
}

```

위의 코드와 이미지는 아두이노 나노(확장보드)에서 테스트된 코드입니다.  
물론 아두이노 우노에서의 와이어링은 동일합니다.



제대로 와이어링 & 코드를 적용한 경우에는 아래와 같은 동영상처럼 실행됩니다.

Right To Left “HELLO” 스크롤입니다.

(기술지원 게시판에도 바로가기 링크 있습니다)

<http://www.youtube.com/watch?v=8nmkUlYU1PA>

## 58 > 74HC595N IC x 1

8 비트 쉬프트 레지스터 IC 칩입니다.

8-Bit Serial-Parallel Shift Register 3-State

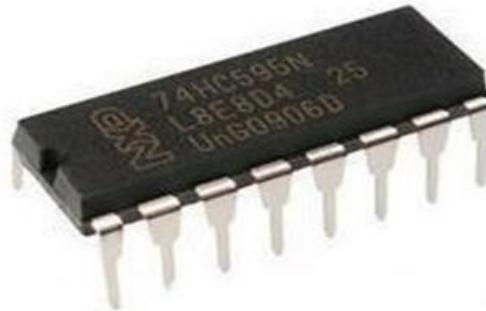


그림 58-1 74HC595N

여러 개의 OUTPUT 방향 신호 처리할 때 많이 사용되는 칩입니다.

시리얼로 데이터를 받아서 병렬로 출력 해주는 칩입니다.

Only Direction Output 신호 아웃풋 전용으로만 사용되는 특성을 가진 칩입니다.

아두이노에서 “10101010” 의 1 BYTE 넘겨주면, 74HC595 IC 핀에서는 Q0~Q7 핀으로 동시에 “10101010” 출력하도록 되어 있습니다. 참고로 1 바이트(BYTE)는 8 비트(BIT)입니다.

즉, Q0 은 처음 비트, Q1 은 그 다음 자리, Q7 까지 8 개의 출력입니다.

3 개의 핀을 사용하여 8 개의 출력 전용(OUTPUT Only) 컨트롤을 가능하게 합니다.

아두이노 연결에는 디지털 핀에 연결하면 됩니다.

8 비트 쉬프트 칩셋 가용 전원은 대부분 2V ~ 6V 이므로 아두이노에서 쉽게 사용 가능합니다. 쉬프트 레지스터 IC 는 보통 출력 용도로의 많은 포트가 필요할 경우에 사용하게 됩니다. 8 개의 포트로 출력이 가능하므로 많은 수의 LED 제어에 가장 많이 사용됩니다.

일반적으로 적용 가능한 용도로는 LED, 7-Segment (FND 종류), DOT MATRIX LED 등에 많이 사용 됩니다.

| 핀 번호    | Pin 명칭, 위치 핀 명칭 |       | 설명                                 |
|---------|-----------------|-------|------------------------------------|
| 1-7, 15 | Q0~Q7           | QA~QH | Parallel Data Output(병렬 데이터 출력 포트) |
| 8       | GND             |       | Ground. 그라운드.                      |

|    |         |       |  |
|----|---------|-------|--|
| 9  | Q7'     | QH'   | Serial data output(다음 쉬프트 레지스터로 데이터를 넘김)           |
| 10 | —<br>MR | SRCLR | master reclear (active LOW, 5V로 연결한다)              |
| 11 | SH_CP   | SRCLK | shift register clock input (clock pin, 아두이노와 연결)   |
| 12 | ST_CP   | RCLK  | storage register clock input (latch pin, 아두이노와 연결) |
| 13 | —<br>OE |       | output enable (active LOW, GND에 연결)                |
| 14 | DS      | SER   | serial data input (data pin, 아두이노와 연결)             |
| 16 | Vcc     |       | positive supply voltage                            |

아두이노와의 연결은 11,12,14 번 핀을 연결합니다.

### 74HC595

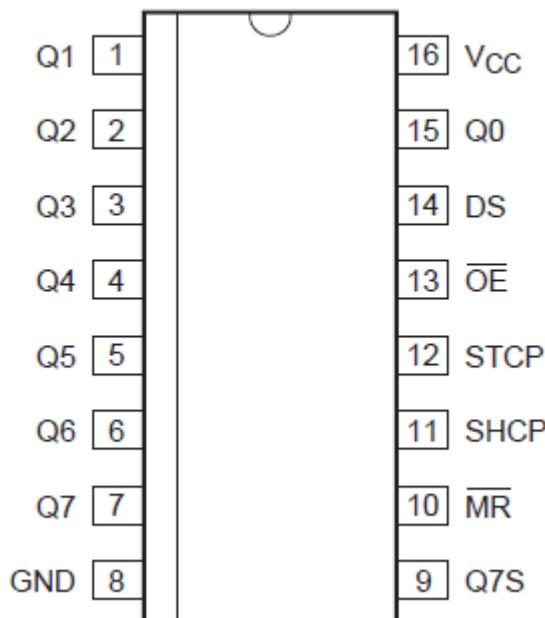


그림 58-2 핀 설명

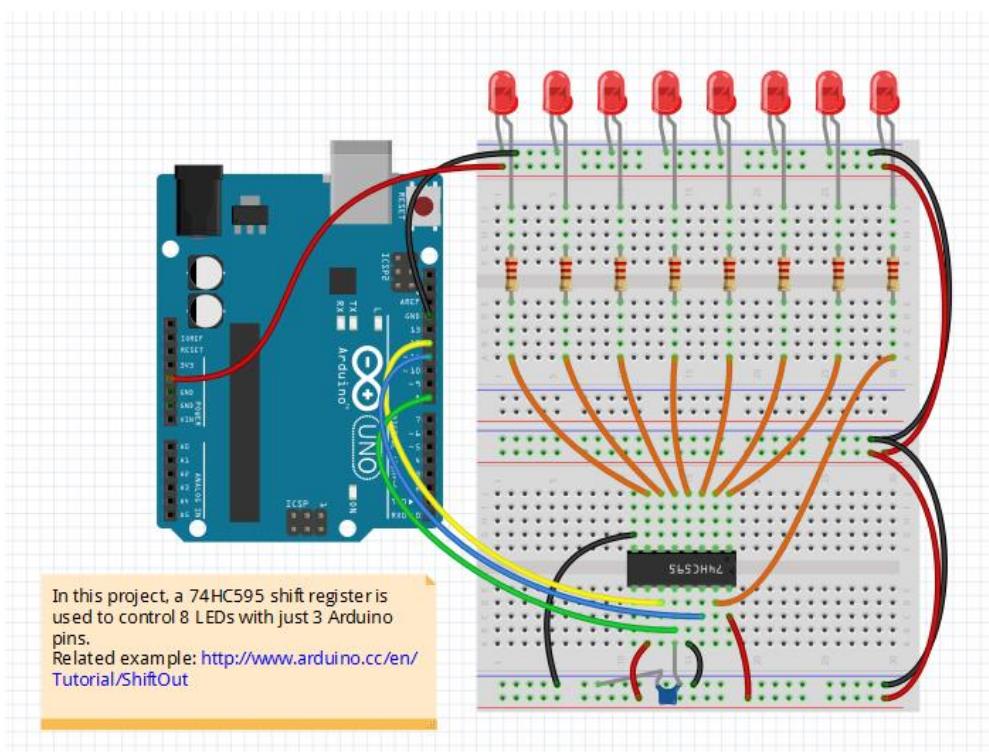


그림 58-3 브레드보드 회로도

수프트 레지스터 사용시, 아두이노 스케치 기본 라이브러리에서 bitWrite, bitRead, shiftOut, shiftIn 함수가 지원됩니다.

```
#define bitRead(value, bit) (((value) >> (bit)) & 0x01)
#define bitSet(value, bit) ((value) |= (1UL << (bit)))
#define bitClear(value, bit) ((value) &= ~(1UL << (bit)))
#define bitWrite(value, bit, bitvalue) (bitvalue ? bitSet(value, bit) :
bitClear(value, bit))
```

위의 정의(define)된 선언은 Defined from <#include <Arduino.h>>

위의 비트 관련 함수는 비트 연산 함수(매크로 정의)로서, 하드웨어 개발, 소프트웨어 개발에도 많이 사용 됩니다. 가능하면 이해를 하시는게 좋습니다.

Shift Out에 관한 내용은 아두이노 공식 사이트에도 상세히 설명 되어 있습니다.  
<http://www.arduino.cc/en/Tutorial/ShiftOut>

위 링크 주소에 설명 되어 있는 내용들은 가능하면 이해하고 넘어가시면 차후에 여러 가지 용도에 맞는 하드웨어 개발 및 응용 개발에 많은 도움이 되시리라 봅니다. 특히 여러 개의 745HC595 사용에 대한 설명도 주의 깊게 살펴 보시기 바랍니다.

그리고 74HC595 와 비슷한 칩으로는 74LS164, 74C299 등도 있습니다.

8 개의 OUTPUT 신호 HIGH/LOW 기능이 되므로 8 개의 LED 컨트롤 해보도록 합니다.

아두이노 사이트에서 친절하게 설명 & 예제 있습니다.

<http://www.arduino.cc/en/Tutorial/ShiftOut>

사용되는 부품은 74HC595N x 1 개, 220 Ohm R x 8 개, 100nF 커패시터 1 개, 점퍼 선 필요합니다. 실습시에는 100nF 커패시터는 사용 안해도 무방합니다.

위의 코드는 아래와 같습니다.

시리얼 인풋으로 ‘0’ ~‘9’ 입력을 받아 숫자로 변환합니다. ASCII 코드 ‘0’ ~ ‘9’를 정수로 변환하는 방법은 많이 있지만, 아래의 예제 코드는 입력 받은 값에 마이너스 48 을 해주고 있습니다.

```
int bitToSet = Serial.read() - 48;
```

Ascii 코드 테이블에 대한 내용은 <http://en.wikipedia.org/wiki/ASCII> 에도 상세히 설명 되어 있습니다. 해당 되는 ‘0’ ~ ‘9’ 아스키 코드값을 찾아 보기 바랍니다.

대입된 숫자 인덱스에 의해 LED ON/OFF 예제입니다.

[http://www.gameplusedu.com/pds/gpshop/kit\\_code/strongest/74hc595n\\_ex\\_1.ino](http://www.gameplusedu.com/pds/gpshop/kit_code/strongest/74hc595n_ex_1.ino)

```
/*
Shift Register Example
for 74HC595 shift register
```

This sketch turns reads serial input and uses it to set the pins  
of a 74HC595 shift register.

Hardware:

- \* 74HC595 shift register attached to pins 2, 3, and 4 of the Arduino,  
as detailed below.
- \* LEDs attached to each of the outputs of the shift register

Created 22 May 2009

Created 23 Mar 2010

by Tom Igoe

```
*/  
  
//Pin connected to latch pin (ST_CP) of 74HC595  
const int latchPin = 8;  
//Pin connected to clock pin (SH_CP) of 74HC595  
const int clockPin = 12;  
///Pin connected to Data in (DS) of 74HC595  
const int dataPin = 11;  
  
void setup() {  
    //set pins to output because they are addressed in the main loop  
    pinMode(latchPin, OUTPUT);  
    pinMode(dataPin, OUTPUT);  
    pinMode(clockPin, OUTPUT);  
  
    Serial.begin(9600);  
    Serial.println("reset");  
}  
  
void loop()  
{  
    if (Serial.available() > 0)  
    {  
        // ASCII '0' through '9' characters are  
        // represented by the values 48 through 57.  
        // so if the user types a number from 0 through 9 in ASCII,  
        // you can subtract 48 to get the actual value:  
        int bitToSet = Serial.read() - 48;  
  
        // write to the shift register with the correct bit set high:  
        registerWrite(bitToSet, HIGH);  
    }  
}  
  
// This method sends bits to the shift register:  
void registerWrite(int whichPin, int whichState)  
{  
    // the bits you want to send
```

```
byte bitsToSend = 0;

// turn off the output so the pins don't light up
// while you're shifting bits:
digitalWrite(latchPin, LOW);

// turn on the next highest bit in bitsToSend:
bitWrite(bitsToSend, whichPin, whichState);

// shift the bits out:
shiftOut(dataPin, clockPin, MSBFIRST, bitsToSend);

// turn on the output so the LEDs can light up:
digitalWrite(latchPin, HIGH);
}
```

## 59 > INFRARED REMOTE CONTROL X 1

IR 리모트 컨트롤러. 적외선 리모트 컨트롤러입니다.



그림 59-1 적외선 리모트 컨트롤러

Infrared Receiver 설명 되어 있습니다.

## 60 > BREADBOARD JUMPER X 65

브레드보드 점퍼 와이어 65 개

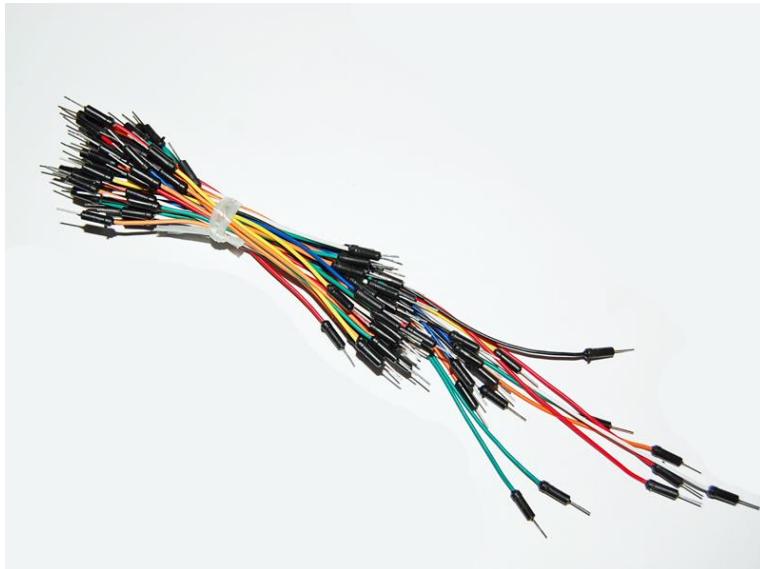


그림 60-1 브레드보드 점퍼선 65 개

아두이노 보드와 각종 브레드보드 연결에 필요한 점퍼 케이블입니다.

## 61 > FEMALE TO MALE DUPONT LINES X 10

30cm 30pcs Female-Male Dupont Breadboard Jumper Cable Wire Line  
2.54mm pitch.



뒤풍 (듀폰) 케이블입니다. 한쪽은 헤더핀에 연결 할 수 있고 한쪽은 아두이노 보드 핀에 꽂아서 사용 할 수 있습니다.

2.54 피치 간격이 유지되는 케이블입니다. 즉, 아두이노 보드의 Female 헤더 훌더 간격과 정확히 일치되는 케이블입니다. 동시에 여러 개를 이어서 홀딩 할 경우에도 밀리지 않고 정확히 연결 할 수 있습니다.

10 개 미만의 포트 연결이 필요한 경우 뒤풍 케이블의 특성상 떼어서 사용할 수 있습니다.

## 62 > 9 VOLT BATTERY SNAP X 1

9 볼트 전용 배터리를 사용 할 경우 필요합니다. 대부분 9V 전용 배터리는 스냅 케이블에 연결하여 아두이노 우노 R3 보드의 전원 공급으로 사용됩니다. 용도에 맞게 스냅 부분과 DC 연결 부분을 잘라서 사용할 수도 있습니다.



## 63 > USB CABLE X 1

B 타입 USB 케이블입니다.

아두이노 보드와 PC USB 포트와 연결 후 스케치 프로그램 업로드 시에 사용되는 케이블입니다.



아두이노는 프로그래밍 & 디버깅을 위해 가상 시리얼 포트를 할당하여 사용됩니다.

아두이노 우노 R3 보드는 USB 시리얼 통신을 하기 위해 ATMEGA16U2 IC 있습니다. 내부적으로 ATMEGA16U2 는 ATMEGA328P-PU 와 SPI 방식으로 연결되어 있습니다.

## 64 > HC-SR04 x 1 초음파 센서

초음파 거리 측정 센서 입니다.



그림 64-1 HC-SR04 초음파 센서 모듈

초음파 거리 측정 센서의 기본 원리는 음파를 쏘아서 반향 되어 수집되는 음파까지의 시간차로 거리를 계산할 수 있습니다.

음속이  $340\text{m/s}$  ( $1\text{초에 } 340\text{ 미터}$ ) 센서를 통해 응답이 오는 시간만 알면 초음파 센서 앞에 있는 사물까지의 거리를 챌 수 있습니다.

측정거리:  $2\text{cm} \sim 5\text{m}$  ( $4\text{m}$ ), 측정각도:  $15$  도, 측정 해상도:  $3\text{mm}$

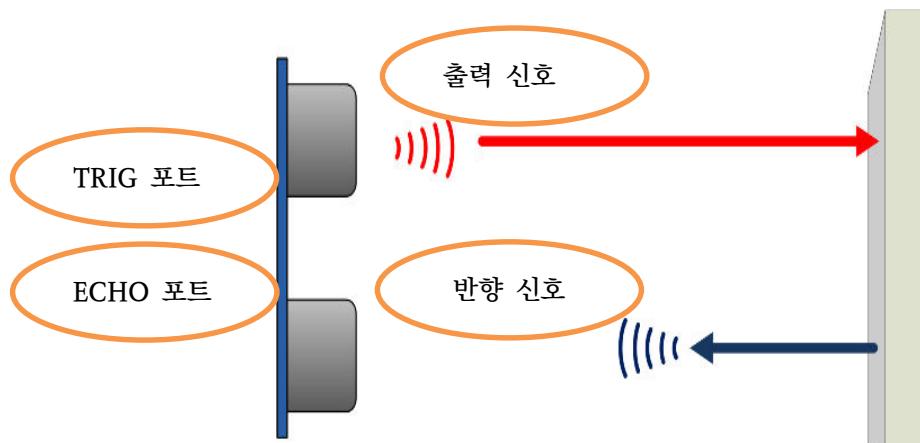


그림 64-2 초음파 센서 작동 원리

초음파 센서 모듈에는 4 개의 핀(포트)이 있습니다. VCC, Trig, Echo, GND 입니다.  
아두이노에 다음과 같이 연결합니다.

| 초음파 센서 모듈 | 아두이노 |
|-----------|------|
| VCC       | 5V   |
| Trig      | D12  |
| Echo      | D13  |
| GND       | GND  |

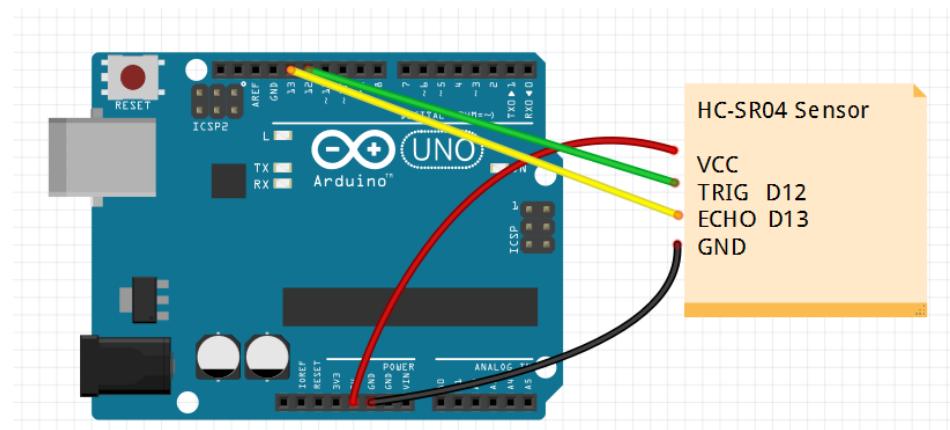


그림 64-3 브레드보드 회로도

아두이노에서 Trigger(트리거) 핀으로 HIGH 를 입력하면 초음파 모듈에서 40KHz 음파를 발사합니다. (10us 이상 HIGH 유지) 이때부터 Echo 핀은 High 상태가 되고, 음파가 되돌아와 수신되면 echo 핀이 다시 Low 상태가 됩니다. 이 간격에서 거리를 구하고 다시 2로 나누면 됩니다.

음파속도가 340m/s 이고 1cm 가는데 29us 가 걸립니다. 거리를 구하는 공식은  
 $Distance = time / 29 / 2;$

4 미터 정도의 거리까지 측정 가능합니다.

데이터시트:<http://www.igameplus.com/pds/gpshop/arduino/pdf/HCSR04.pdf>

아두이노 연결:

| 센서 핀 | 아두이노 핀 |
|------|--------|
| VCC  | 5V     |
| TRIG | D12    |
| ECHO | D13    |
| GND  | GND    |

## 64.1 DIRECT TRIG, ECHO 사용

기본 함수만을 사용하여 코드를 작성하면 아래와 같습니다.

예제코드:

[http://www.gameplusedu.com/pds/gpshop/kit\\_code/strongest/hcsr04\\_direct\\_connection.ino](http://www.gameplusedu.com/pds/gpshop/kit_code/strongest/hcsr04_direct_connection.ino)

```
#define trigPin 12
#define echoPin 13

void setup() {
    // 시리얼 포트 9600 속도 초기화.
    Serial.begin (9600);
    // 센서 Trig 연결 포트를 출력으로 설정합니다.
    pinMode (trigPin, OUTPUT);
    // 센서 에코 연결핀은 입력 모드로 설정.
    pinMode (echoPin, INPUT);
}

void loop() {
    // 지속시간 , 거리 변수 정의
    int duration, distance;

    //
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(1000);
    digitalWrite(trigPin, LOW);

    duration = pulseIn(echoPin, HIGH);
    distance = (duration/2) / 29.1;
    if (distance >= 200 || distance <= 0) {
        Serial.println("Out of range");
    } else {
        Serial.print(distance);
        Serial.println(" cm");
    }
    delay(500);
}
```

## 64.2 PULSEIN 함수 사용 설명

위의 코드에는 duration = pulseIn(echoPin, HIGH); 라는 함수를 사용합니다. 직관적인 이해는 echoPin 으로 지정된 포트의 값이 HIGH 신호로 변경되기까지의 시간을 반환하는 함수입니다. 변경되기까지의 시간은 pulseIn 함수 호출 시간부터입니다.

함수 선언 모체는 아래와 같습니다.

```
unsigned long pulseIn(uint8_t pin, uint8_t state);
unsigned long pulseIn(uint8_t pin, uint8_t state, unsigned long timeout);
```

timeout 이 지정되지 않으면 기본값은 1,000,000 usec 입니다.

usec 설명

1 sec 는 모두가 아는 1 초입니다.

1 msec 는 1 초의 1/1,000 입니다.

## 64.3 NEWPING 라이브러리 사용 예제

아두이노 사이트에서 NewPing 이라는 라이브러리 배포를 합니다. 초음파 센서 사용 시 간단히 몇 줄로 구현 할 수 있습니다.

<http://playground.arduino.cc/Code/NewPing>

```
#include <NewPing.h>
#define TRIGGER_PIN 12
#define ECHO_PIN    13
#define MAX_DISTANCE 200

NewPing sonar(TRIGGER_PIN,ECHO_PIN,MAX_DISTANCE);
void setup() {
  Serial.begin(115200);
}
void loop() {
  delay(50);
  int uS = sonar.ping();
  Serial.print("Ping: ");
  Serial.print(uS / US_ROUNDTRIP_CM);
  Serial.println("cm");
}
```

## 65 > 블루투스 HC-06 슬레이브 모듈

### 65.1 블루투스 소개

블루투스(Bluetooth)는 휴대폰, 노트북, 헤드폰, 스피커, GPS, 등의 휴대기기를 서로 연결해 정보를 교환하는 근거리 무선 기술입니다.

대부분 10 미터 안팎의 근거리에서 저전력 무선 연결이 필요할 때 사용 됩니다.

블루투스 통신 기술을 사용한 제품들은 실생활에 무수히 많습니다.

가령, 블루투스 헤드폰을 사용하면 거추장스러운 케이블 없이도 주머니 속의 MP3 플레이어의 음악을 들을 수 있습니다.

블루투스 통신기술은 1994년 휴대폰 공급업체인 에릭슨(Ericsson)이 시작한 무선 기술 연구를 바탕으로, 1998년 에릭슨, 노키아, IBM, 도시바, 인텔 등으로 구성된 ‘블루투스 SIG(Special Interest Group)’를 통해 본격적으로 개발, 이후 블루투스 SIG 회원은 급속도로 늘어나 2010년 말 통계 자료에 의하면 전세계 회원사가 13,000개가 넘는다고 합니다.

### 65.2 블루투스 모듈 아두이노

블루투스 통신 모듈 HC-06 입니다.

HC-06 모듈은 슬레이브 전용 모듈입니다.

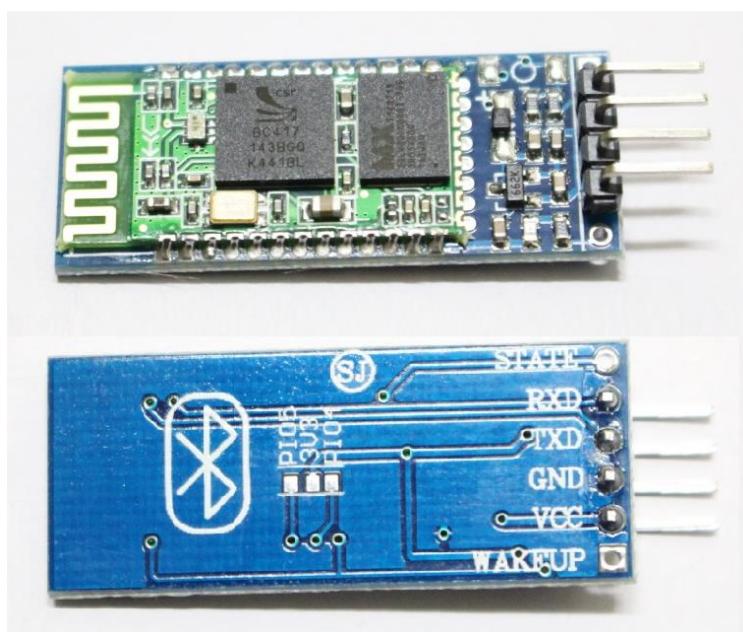


그림 65-1 HC-06 모듈 앞면, 뒷면

블루투스 마스터 모듈은 주변의 슬레이브 기능이 되는 모듈을 검색하여 페어링 접속을 요구하는 기능이 있습니다. 블루투스 슬레이브 모듈은 마스터 모듈에서의 페어링에 응답하는 구조로 되어 있습니다.

블루투스 HC-06 (HC-05 포함) 페어링 연결 기본 암호는 1234 입니다. 또는 0000입니다.

추가 정보가 필요할 시에는 모듈의 STATE / WAKEUP 사용 가능합니다.

- 2.4GHz 안테나 내장
- EDR 블루투스 2.0, 2Mbps - 3Mbps
- 외부 8Mbit FLASH
- 3.3V ~5V 사용 가능. (최대 7V)
- 표준 HCI 포트 (UART)
- 옵션 PIO 제어
- SMD 배치 프로세스로 모듈
- 디지털 2.4GHz 무선 송신
- CSR BC04 블루투스 칩 기술
- 블루투스 클래스 2 전력 레벨
- RoHS 규제 절차
- 보관 온도: -40 +85 도, 작동 온도: -25 으로 75 도
- 외형 (27mm × 13mm × 2mm)

블루투스의 가장 큰 장점은 근거리(최근에는 원거리 가능)에서의 안정된 전원 작동 상태에서는 유선 시리얼 통신의 안정성에 준하는 기능이 보장된다는 것입니다.

간단한 연결 방식과, 직관적인 시리얼 통신 사용은 단점보다는 장점이 많습니다.

## 65.3 블루투스 연결 사용 방법

블루투스 HC-06, HC-05, 기타 UART 통신(시리얼 통신과 개념은 동일)이 지원되는 모듈들은 대부분 아두이노에 바로 연결하여 사용 가능합니다.

### 65.3.1 블루투스 페어링(Pairing)

블루투스 기기는 모두 페어링을 먼저 해주어야 합니다. 즉, 사용하고자 하는 기기와의 등록을 먼저 해주어야 합니다. 페어링이 안되면 사용 불가입니다.

HC-06 이라는 모듈은 슬레이브(Slave) 기능이 됩니다. 블루투스 슬레이브 기능은 다른 기기에서의 요청이 들어오면 연결이 가능한 모듈입니다. 반대로 HC-05 모듈은 슬레이브 기능과, 마스터 기능이 지원되고 있습니다. HC-05 모듈에서의 마스터 기능은

주변의 블루투스 슬레이브 기기 검색 및 연결 요청 기능이 있습니다. 단, 1 대의 슬레이브만 사용 할 수 있습니다. 즉, 1:1 페어링 및 통신만 지원되고 있습니다.

스마트폰의 블루투스 검색, 또는 PC, 노트북 등에서 “장치 추가” 등의 기능을 이용하면 HC-06 모듈이 작동 중인 경우 등록을 하여 암호 설정 후 바로 사용 가능합니다.

### 65.3.2 하드웨어 시리얼 포트 사용

블루투스 HC-06 모듈을 아두이노 보드의 하드웨어 시리얼 포트에 연결하여 사용하는 방식입니다.

아두이노 보드들에는 모두 시리얼 통신 가능 포트가 존재하고 있습니다. 즉, 아두이노 보드에 사용되는 MCU의 종류에 따라 하드웨어 시리얼 통신 가능 포트는 1 개 또는 4 개의 포트까지 사용 가능합니다.

아두이노 우노 R3 보드에는 1 개의 하드웨어 시리얼 통신 포트가 있습니다.

아두이노 포트의 하드웨어 UART 포트 RX (D0), TX (D1) 연결하여 바로 사용 가능합니다. 블루투스 모듈만 연결 하면 Serial.print(), Serial.Read() 등의 함수 사용시 시리얼 모니터 창의 내용을 동일하게 사용 가능합니다.

아두이노 연결:

| HC-06 | 아두이노 |
|-------|------|
| VCC   | 5V   |
| GND   | GND  |
| TX    | D0   |
| RX    | D1   |

D0, RX 수신  
D1, TX 송신

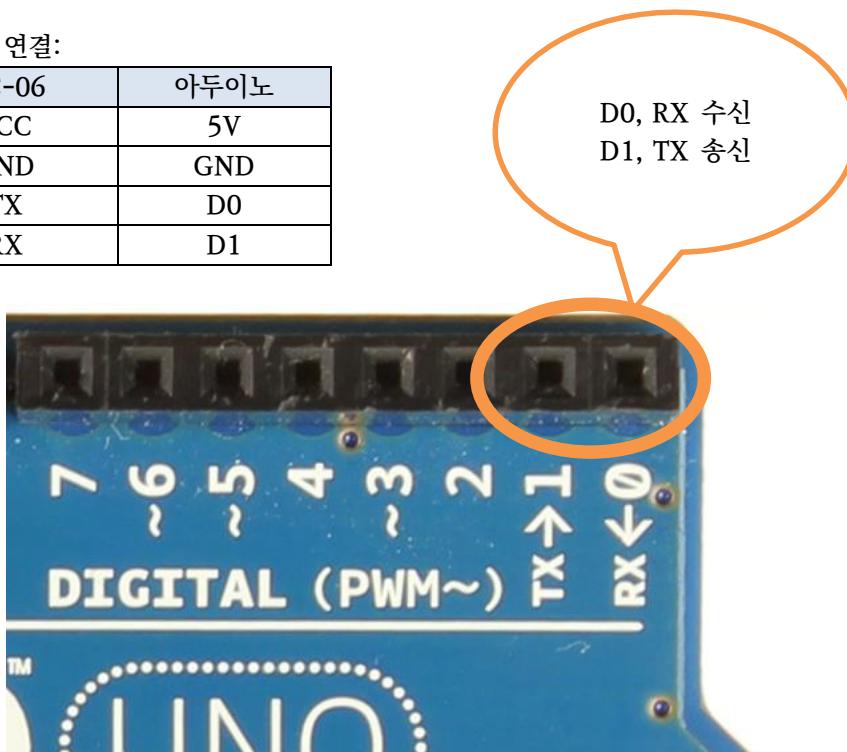


그림 65-2 시리얼 통신 포트, UART 포트

하드웨어 시리얼 포트 사용시 장점은 안정된 통신이 보장됩니다. 별도의 문제가 없는 경우, 외부 환경에 의한 통신 방해는 발생되지 않습니다. 아두이노에서의 여러 종류의 하드웨어 모듈 장착 시, 충돌 현상에서 제외되어 안정된 통신을 할 수 있습니다. 하드웨어, 소프트웨어에서의 안정된 통신 보장은 상당히 중요한 부분이기도 합니다.

#### 하드웨어 시리얼 포트 사용시 주의점:

아두이노 보드는 하드웨어 시리얼 포트(D0, D1)를 스케치 IDE에서 작성된 소스코드의 업로드 용도로 사용하게 되어 있습니다. D0, D1 포트에 외부의 시리얼 통신 모듈(UART 통신 모듈)이 연결되어 있는 경우에는 업로드가 안됩니다.

업로드 시에 아래와 같은 에러 메시지가 반복 출력되면서 업로드가 중단됩니다.

```
avrdude: stk500_getsync(): not in sync: resp=0x00
```

아두이노 스케치 프로그램 업로드 시에는 D0, D1에 연결된 블루투스 모듈의 연결을 제거 후 사용해야 합니다. 업로드 완료 후 다시 블루투스 모듈을 D0, D1에 연결하면서 사용합니다.

아두이노 프로그램 업로드 시에 약간의 번거로움이 있습니다. 용도에 따라 부족한 시리얼 포트의 대체방법으로 일반 디지털 포트를 시리얼 포트로 사용하는 방법이 소프트웨어 시리얼 통신입니다.

아두이노 우노 기본 라이브러리 중에 소프트웨어 시리얼 통신 라이브러리가 있습니다.

### 65.3.3 소프트웨어 시리얼 블루투스 통신

아두이노에서는 디지털 포트, 아날로그 포트를 시리얼 통신 포트로 사용할 수 있습니다. “Software Serial Library” 사용하여 적용 가능 합니다. 기존의 MCU 펌웨어 프로그래밍 시 소프트웨어 시리얼 통신 구현은 쉽지 않은 개념이자, 어려운 개발 작업이었습니다. 하지만, 아두이노 스케치 IDE 사용하는 경우에는 상당히 안정된 성능을 보장하는 소프트웨어 시리얼 라이브러리를 사용할 수 있습니다.

우선 사용 방식은 원하는 디지털, 아날로그 포트에 RX, TX 정의 후 HC-06 모듈과 연결만 해주면 됩니다.

블루투스 모듈의 GND 와 VCC 5V(또는 3.3V) 연결해 줍니다.

아래의 설명은 D2, D3 포트에 연결하여 사용하는 방법입니다.

블루투스 모듈의 RX / TX 와 아두이노 보드의 D3, D2 핀에 연결 해 줍니다.

아두이노 연결:

| HC-06 | 아두이노 |
|-------|------|
| VCC   | 5V   |
| GND   | GND  |
| TX    | D2   |
| RX    | D3   |

스케치 IDE에서 아래의 예제 코드를 업로드 후 시리얼 모니터 창을 엽니다.  
제대로 연결된 상태라면 스마트폰에서의 입력 그대로 시리얼 모니터 창에 나옵니다.  
반대로 시리얼 모니터 창에서의 입력을 하면 스마트폰의 BlueTerm 화면에 문자가 표시됩니다.

예제코드) 소프트웨어 시리얼 통신 예제입니다.

[http://www.gameplusedu.com/pds/gpshop/kit\\_code/strongest/bt\\_hc\\_06\\_ex\\_1.ino](http://www.gameplusedu.com/pds/gpshop/kit_code/strongest/bt_hc_06_ex_1.ino)

```
/*
블루투스를 사용하기 위한 준비를 합니다.
2 번은 RX, 3 번은 TX 입니다.
D2 번은 블루투스 모듈의 TX 연결합니다.
D3 번은 블루투스 모듈의 RX 연결합니다.
데이터를 주고 받을 핀 번호입니다.
*/
#include <SoftwareSerial.h>

SoftwareSerial bt(2, 3);

void setup()
{
    Serial.begin(9600); // 시리얼 통신 초기화
    bt.begin(9600); // 블루투스를 사용하기 위해 초기화
    Serial.println("ready");
}

void loop()
{
    if(bt.available()) //블루투스를 통해 데이터가 들어오면
    {
        Serial.write("PHONE: ");
        Serial.write(bt.read()); //시리얼 모니터로 받은 내용 출력
        Serial.println();
    }
    if(Serial.available()) //시리얼 모니터에서 데이터를 보내면
    {
        bt.write("PC : ");
        bt.write(Serial.read()); //스마트폰으로 받은 데이터 출력
        bt.println();
    }
}
```

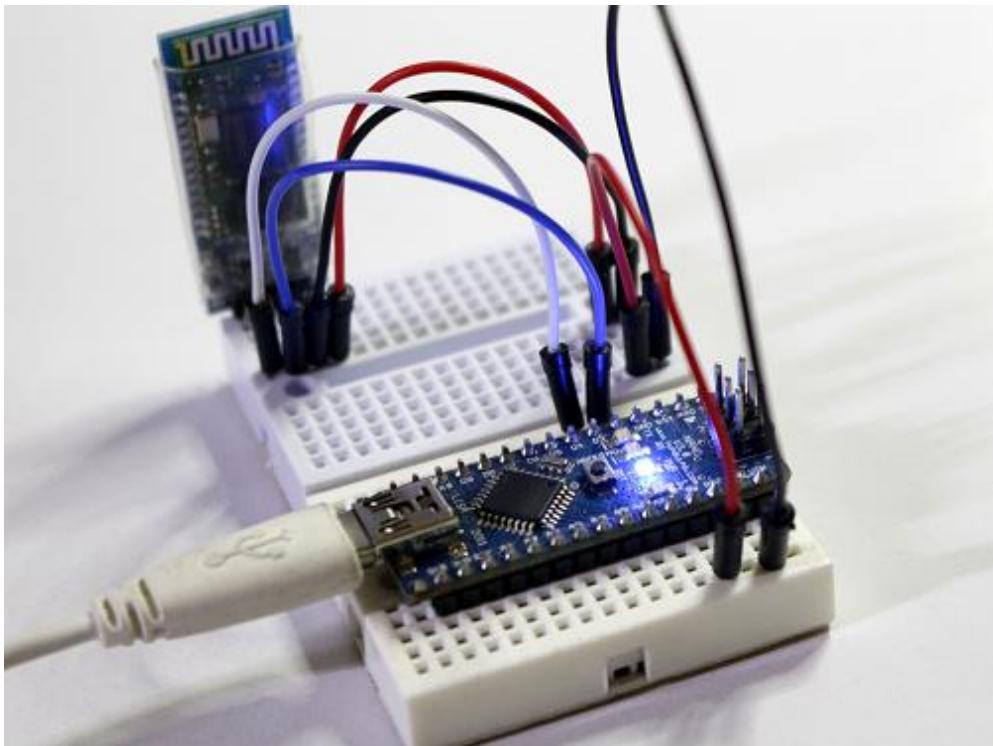


그림 65-3 아두이노 나노 & 블루투스 모듈 연결 참조 그림

## 65.4 스마트폰 연동

스마트폰에서 구글 앱 스토어 접속 후 공개 프로그램 BlueTerm 검색, 설치 해줍니다.

블루텀 사이트:

<https://play.google.com/store/apps/details?id=es.pymasde.blueterm>

블루텀 앱을 안드로이드 스마트폰에 설치 후 블루투스 주변기기 검색을 해봅니다.

블루텀 소스코드는 공개입니다. (출처: <http://pymasde.es/blueterm/>)

블루텀 앱의 용도는 시리얼 통신 터미널입니다. 전송되는 내용들을 볼 수 있고, 반대로 화면상의 키보드를 열어서 데이터를 보낼 수도 있습니다.

아두이노 보드에 연결된 HC-06 장치가 검색되면 페어링을 해줍니다. 페어링시의 암호는 1234 입니다.

블루투스 HC-06 모듈의 LED 깜박일 경우는 연결이 안된 상태입니다.

연결이 되었을 경우는 모듈 제조사에 따라 LED 항상 On 또는 통신 데이터 전송일 경우에만 점등됩니다.

연결이 되면 일반 시리얼 통신처럼 양방향 시리얼 통신이 가능합니다.  
블루투스에서 입력한 문자는 바로 아두이노 보드에서 볼 수 있습니다.  
아두이노에서의 시리얼 모니터 창에서 보기 위해서는 아래의 예제 코드가 아두이노  
보드에 업로드 되어 있어야 합니다.

[http://www.gameplusedu.com/pds/gpshop/kit\\_code/strongest/bt\\_hc\\_06\\_ex\\_1.ino](http://www.gameplusedu.com/pds/gpshop/kit_code/strongest/bt_hc_06_ex_1.ino)



그림 65-1 앱스토어 블루텀

## 65.5 PC 연동

PC에서의 블루투스 HC-06과 페어링을 하기 위해서는 사용하는 PC의 블루투스의 드라이버를 가능하면 제조사에서 배포하는 최신 버전으로 업데이트 해줍니다.

물론 윈7, 윈8 OS에서 설정 되는 기본 블루투스 드라이버에서도 제대로 되는 경우도 있습니다. 만약 HC-06 등의 블루투스 모듈이 장치 검색에서 안 보이는 경우 PC에서 사용되는 블루투스 칩셋 드라이버 버전을 최신으로 설치해 주시기 바랍니다.

윈도우 7에서 블루투스 사용하는 경우에는 “시작” 메뉴→“장치 및 프린터” 선택합니다.



장치 추가 창에서 왼쪽 상단의 “장치 추가” 버튼을 눌러줍니다.

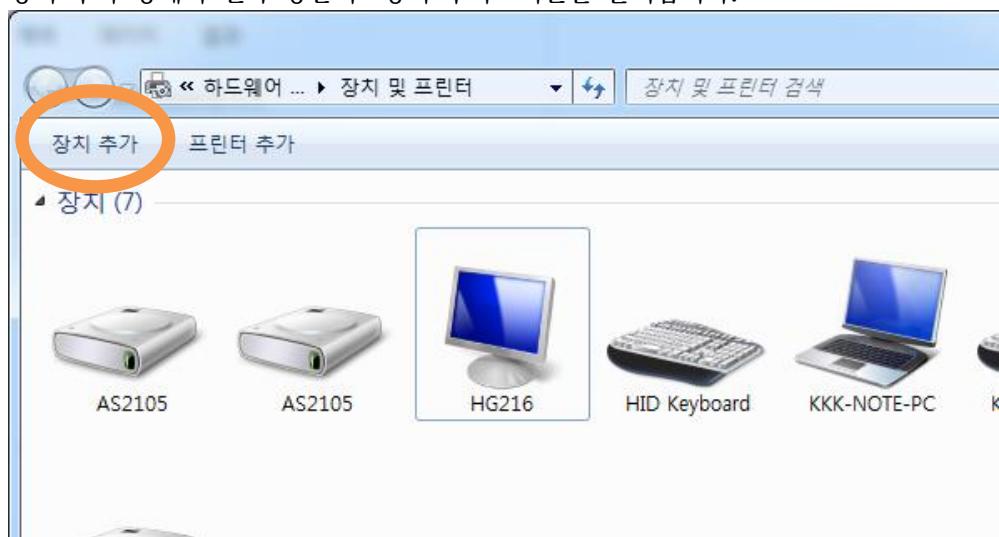


그림 65-4 윈7 장치 목록 창

장치 추가 버튼을 누르면 자동으로 장치 검색이 발생되어 검색된 장치들의 목록이 나오게 됩니다. 검색된 장치의 아이콘을 눌러 간단한 속성 정보를 미리 볼 수도 있습니다.

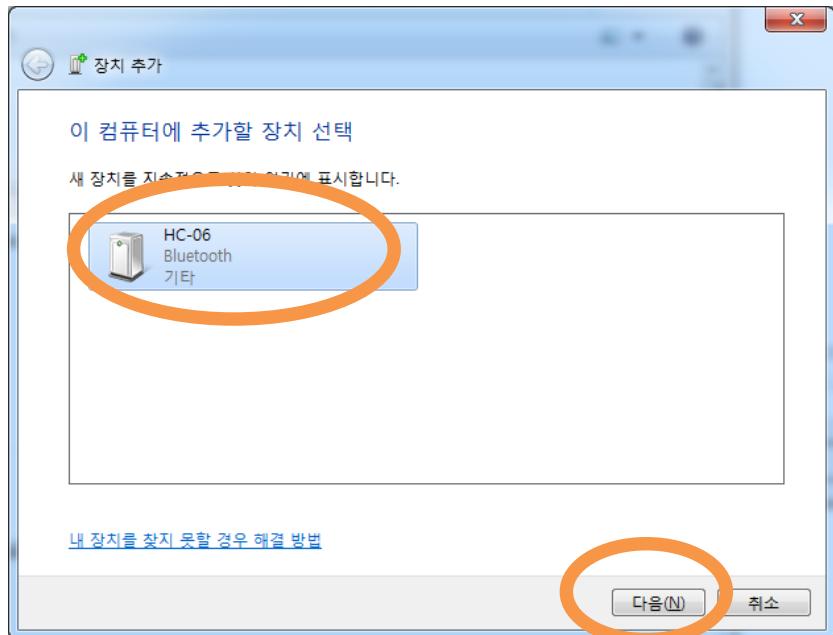


그림 65-5 장치 추가에 HC-06 블루투스 검색

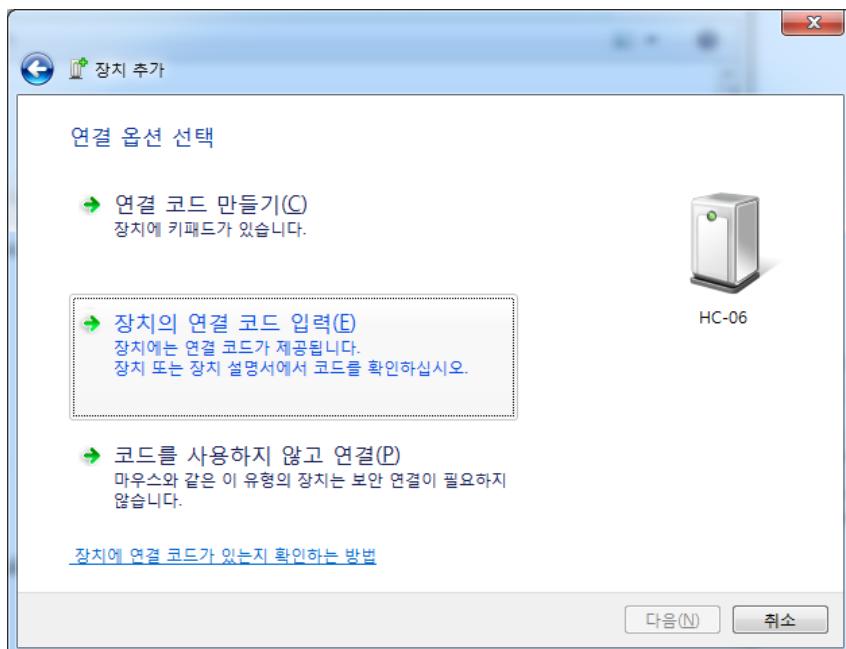


그림 65-6 연결 코드 설정

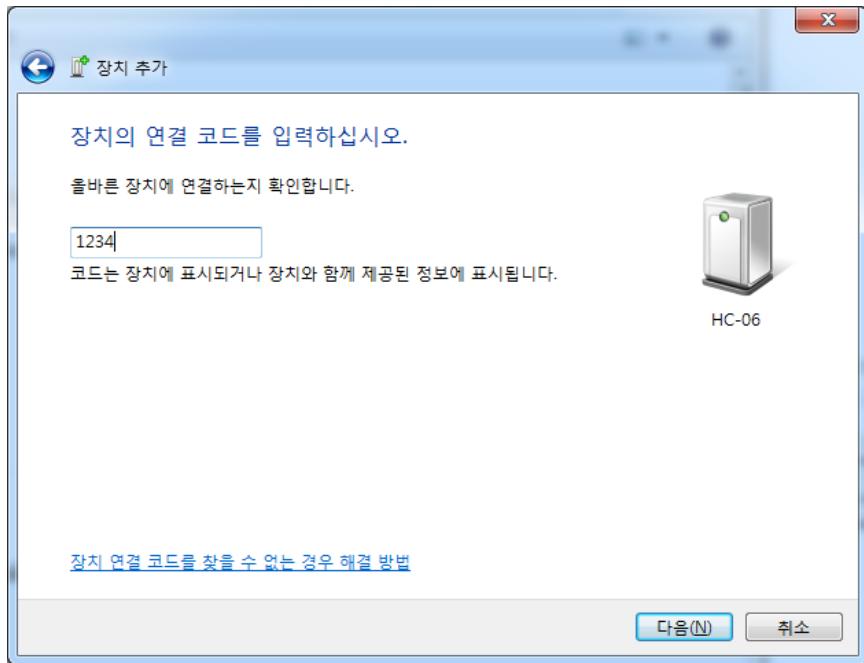


그림 65-7 연결코드 입력 화면

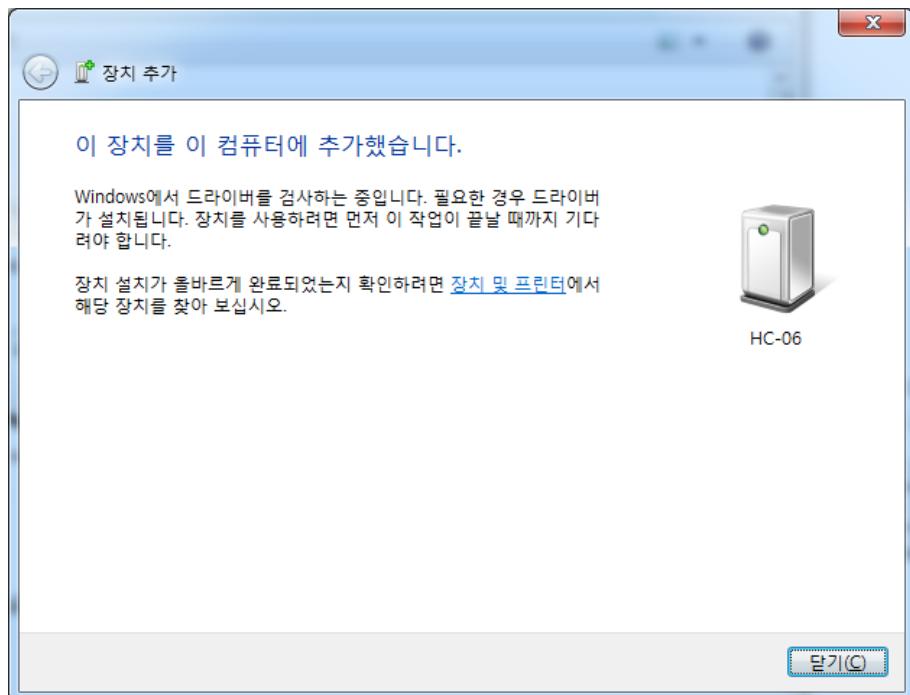


그림 65-8 추가 완료

블루투스 설정 도움말 사이트:

<http://windows.microsoft.com/ko-kr/windows7/add-a-bluetooth-enabled-device-to-your-computer>

## 65.6 블루투스 모듈 이름 변경

HC-06 모듈을 사용하는 목적에 부합 되도록 모듈의 이름을 변경하거나 기타 이유로 설정을 변경해야 하는 경우가 생깁니다.

HC-06 모듈의 경우 PC 또는 스마트폰에서 블루투스 장치 검색을 하면 “HC-06”이라는 명칭으로 설정되어 있습니다. 기본 연결 코드도 “1234”로 되어 있습니다. 이름이나 암호를 변경하려면 AT Command 모드로 들어가야 합니다.

블루투스 HC-06 슬레이브 모듈에 먼저 아래와 같은 코드를 업로드 합니다.

예제코드:

[http://www.gameplusedu.com/pds/gpshop/kit\\_code/strongest/bt\\_hc\\_06\\_rename\\_ex\\_1.ino](http://www.gameplusedu.com/pds/gpshop/kit_code/strongest/bt_hc_06_rename_ex_1.ino)

```
/*
블루투스 HC-06 명칭 변경.
소프트웨어 시리얼 통신.
*/
#include <SoftwareSerial.h>

#define rxPin 2
#define txPin 3

SoftwareSerial SwSerial(rxPin, txPin);

void setup() {
    Serial.begin(9600);
    SwSerial.begin(9600);
    Serial.println("Ready..");
}

void loop()
{
    char data;
    Serial.flush();
    Serial.print("command: ");
    while (!Serial.available());
```

```
while (Serial.available() {
    data = Serial.read();
    if (data == -1) break;
    SwSerial.write(data);
    Serial.write(data);
    delay(1);
}
Serial.println();
delay(1000);
Serial.print(" return: ");
while (SwSerial.available() {
    data = SwSerial.read();
    if (data == -1) break;
    Serial.write(data);
    delay(1);
}
Serial.write("\n\n");
}
```

시리얼 모니터 창을 열면 아래와 같은 메시지가 먼저 보이게 됩니다.

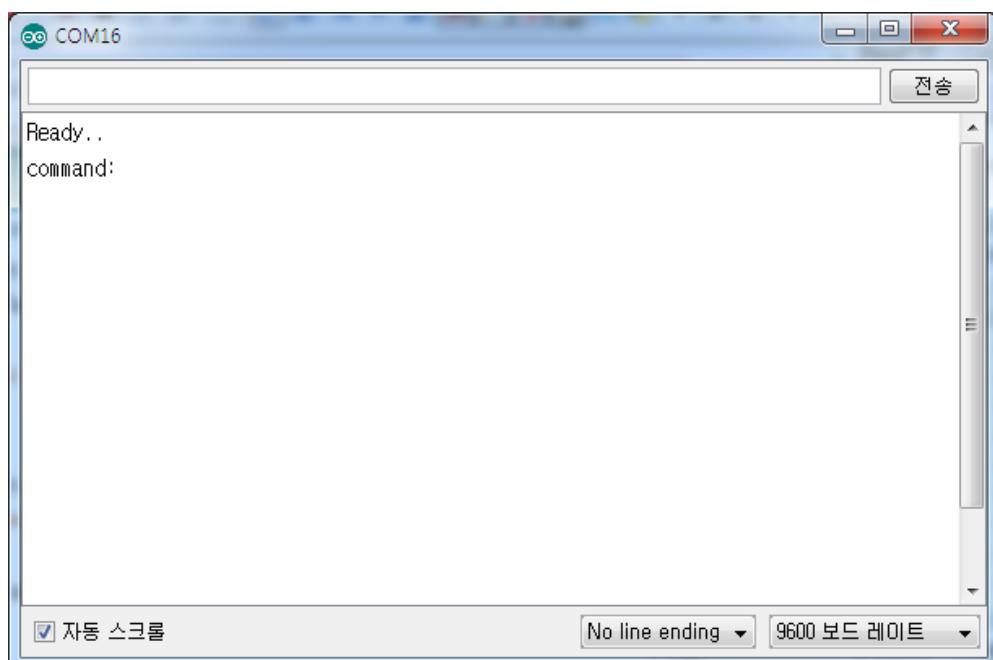


그림 65-9 시리얼 모니터 창

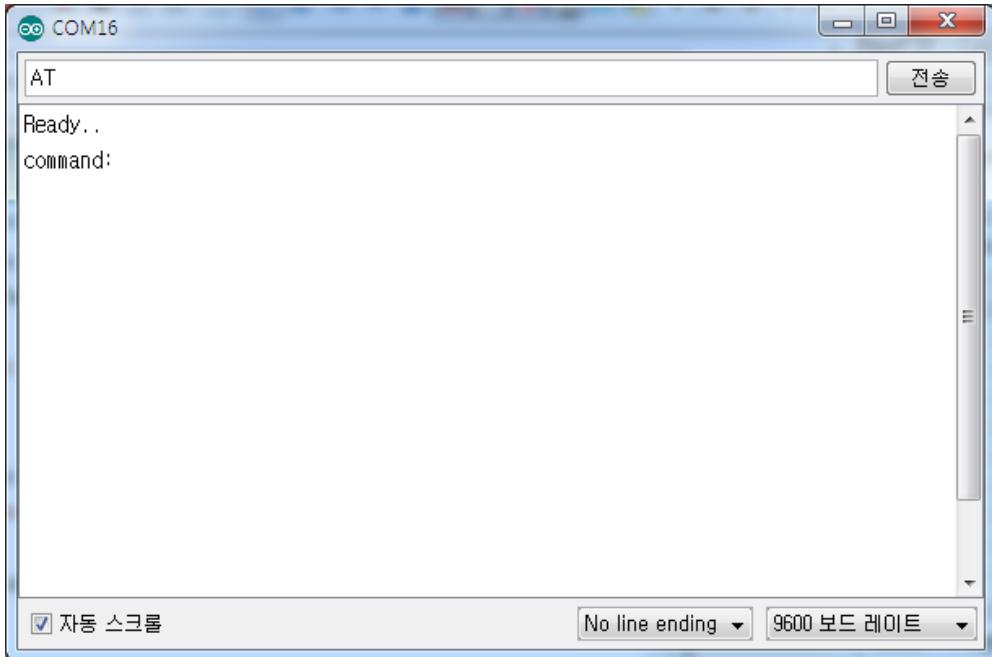


그림 65-10 AT 명령어 입력

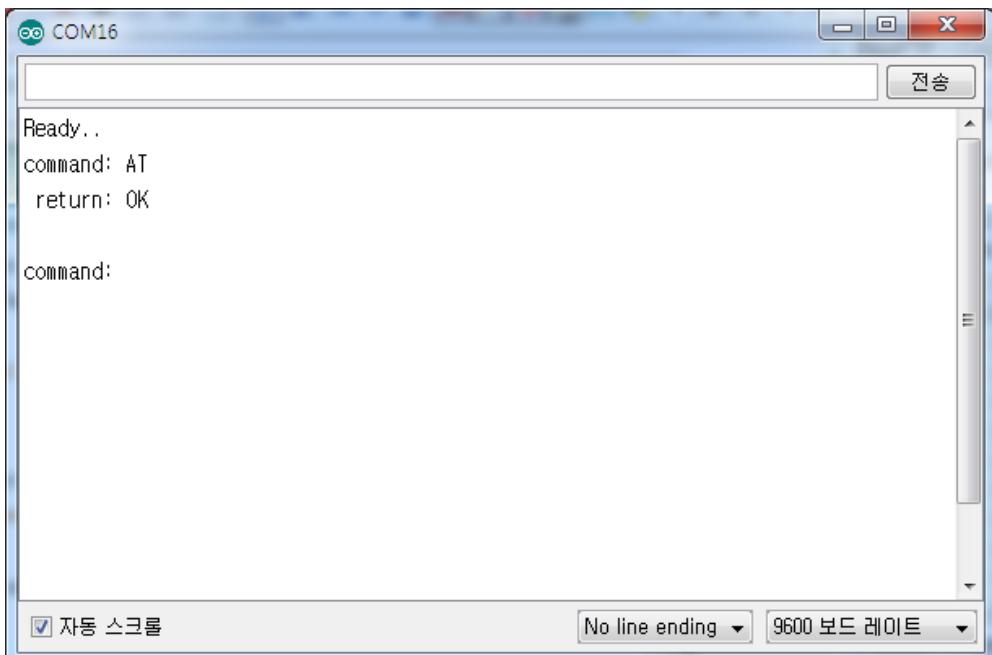


그림 65-11 AT 명령어 입력 후 잠시 뒤에 OK 메시지

위와 같은 “OK” 메시지가 나오면 정상적으로 AT 명령어 모드 상태입니다.

블루투스 HC-06 슬레이브(slave) 모듈은 마스터 기능을 제외한 슬레이브 기능에서만 사용 가능한 AT 명령어들이 있습니다.

명령어는 아래의 목록을 참조합니다.

크게 AT 명령어 모드 체크, 버전 체크, 이름 변경, 연결코드 변경, 통신 속도 변경이 가능하게 되어 있습니다.

다음과 같은 명령어들을 사용할 수 있습니다.

| 명령어               | 응답 문자열       | 비고                                    |
|-------------------|--------------|---------------------------------------|
| AT                | OK           | Communications test                   |
| AT+VERSION        | OKlinvorV1.8 | Firmware version.                     |
| AT+NAMEmyBTmodule | OKsetname    | Sets the modules name to “myBTmodule” |
| AT+PIN6789        | OKsetPIN     | Set the PIN to 6789                   |
| AT+BAUD1          | OK1200       | Sets the baud rate to 1200            |
| AT+BAUD2          | OK2400       | Sets the baud rate to 2400            |
| AT+BAUD3          | OK4800       | Sets the baud rate to 4800            |
| AT+BAUD4          | OK9600       | Sets the baud rate to 9600            |
| AT+BAUD5          | OK19200      | Sets the baud rate to 19200           |
| AT+BAUD6          | OK38400      | Sets the baud rate to 38400           |
| AT+BAUD7          | OK57600      | Sets the baud rate to 57600           |
| AT+BAUD8          | OK115200     | Sets the baud rate to 115200          |
| AT+BAUD9          | OK230400     | Sets the baud rate to 230400          |
| AT+BAUDA          | OK460800     | Sets the baud rate to 460800          |
| AT+BAUDB          | OK921600     | Sets the baud rate to 921600          |
| AT+BAUDC          | OK1382400    | Sets the baud rate to 1382400         |

블루투스 모듈의 이름을 “myBluetoothModule”이라고 변경하는 경우에는 아래와 같이 입력을 하면 됩니다.

“AT+NAMEmyBluetoothModule”

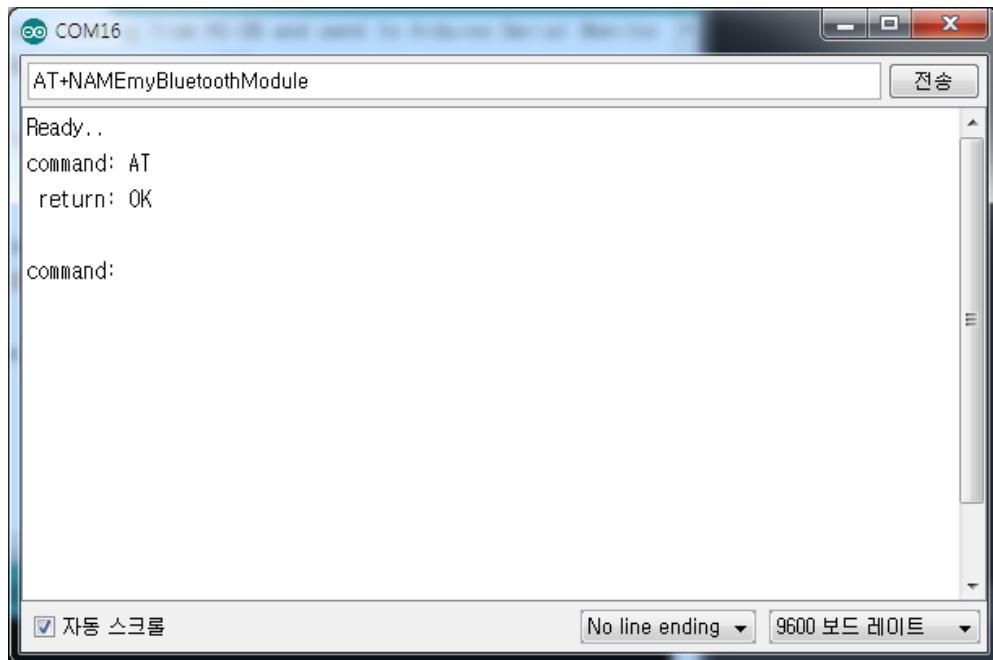


그림 65-12 이름 변경 명령

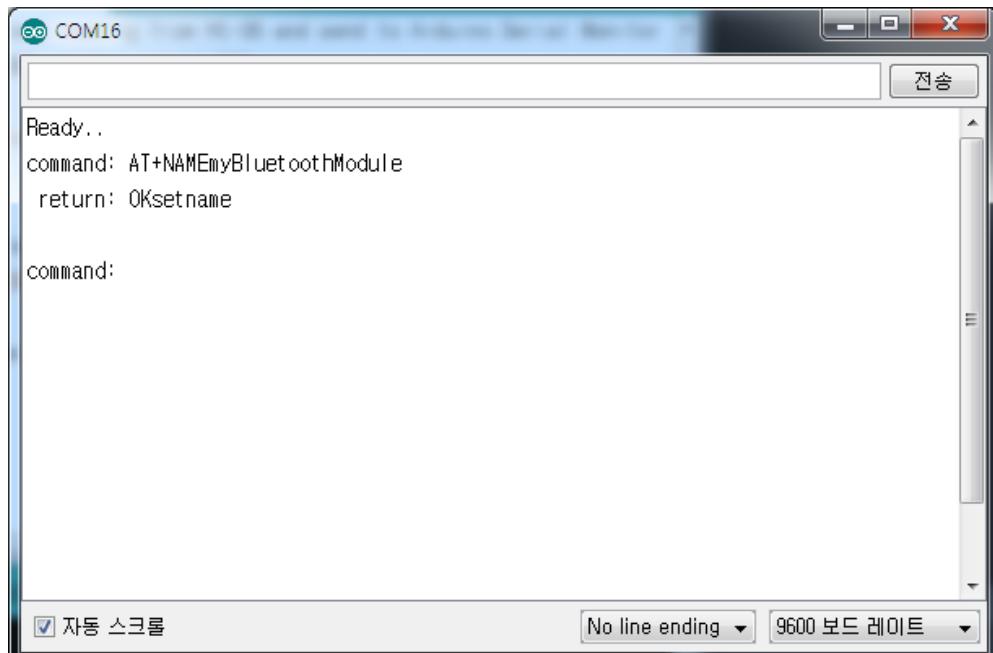


그림 65-13 이름 변경 완료

## 66 무선통신 RF 433/315 MHz 송/수신 장치

무선통신이 가능한 433 MHZ Receiver & Transmitter 모듈입니다.

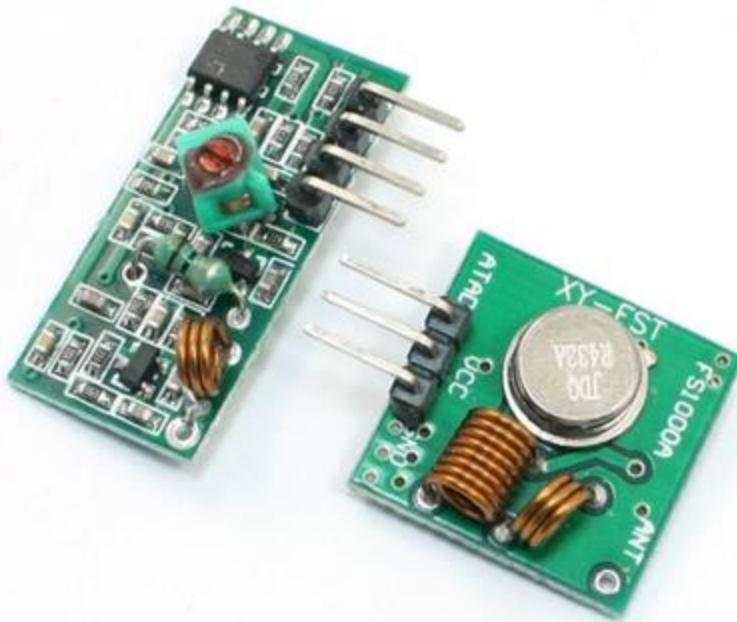


그림 66-1 RF 433/315 MHz 송/수신 모듈

원격으로 송신 모듈과 수신 모듈 사이에 통신을 할 수 있습니다.

즉, 데이터 전송이 가능합니다. 데이터의 형태는 BYTE 이므로 문자열을 비롯한 많은

형태가 가능합니다. 모듈의 송/수신 거리는 장애물이 없을 경우 90m 이상입니다.

저렴한 비용으로 먼 거리 통신이 가능한 모듈 중에 하나입니다.

필요한 준비물은 아두이노 우노 보드 2 개 필요합니다. 송신기를 장착한 MCU 보드와

수신기를 장착한 MCU 보드 각각의 송/수신 코드가 업로드 되어 테스트 됩니다.

물론, 코드 수정하여 송신, 수신 코드를 병합하여 테스트도 가능합니다.

(2 개의 MCU 보드(아두이노 보드)가 없을 경우 1 개의 MCU 보드로 테스트 하는 경우

에는 송신 장치의 연결 포트와, 수신 장치의 연결 포트를 다르게 하여 송/수신 테스  
트 가능합니다)

넓은 장소(학교 운동장 끝에서 끝, 공원 등)에서 테스트시 몇백미터 이상은 충분히 통  
신 가능합니다.

송/수신 모듈 기술 사양

Operating voltage: 3~12V

Operating frequency: 433.92MHz; 315Mhz

Standby current: 0mA

Operating current: 20-28mA

Transmission distance: > 500m (open to receiving plate sensitivity at -103dBm distance of more than)

Output Power: 16dBm (40mW)

Transfer rate: <10Kbps

Modulation mode: OOK (Amplitude Modulation)

Operating Temperature: -10 °C ~ +70 °C

수신 모듈(Receiver module)



모듈 핀 순서 VCC/DATA/DATA/GND

송신모듈 (Transmitter module)

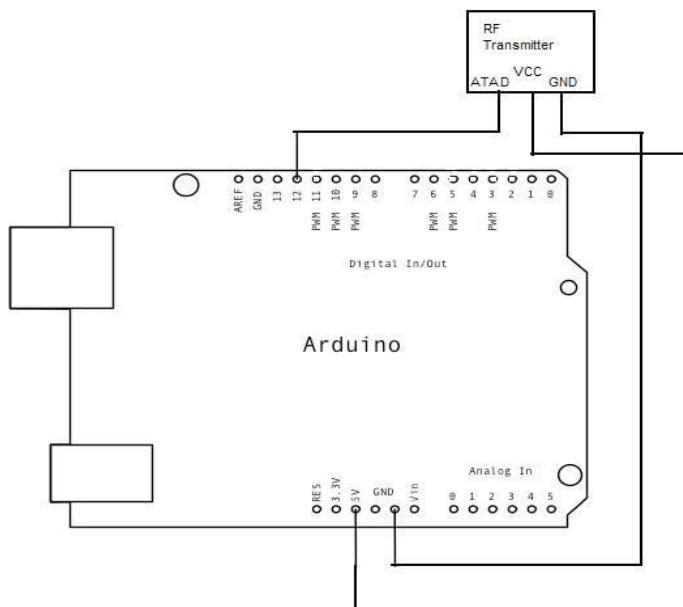


모듈 핀 순서 ATAD/VCC/GND

송/수신 가동 전압은 5V 입력하면 됩니다.

수신모듈은 DATA 핀 중 1 개만 아두이노 디지털 포트에 연결하면 됩니다.

송신모듈은 DATA 핀을 아두이노 디지털 포트에 연결하면 됩니다.



### 그림 66-2 송신 모듈 연결

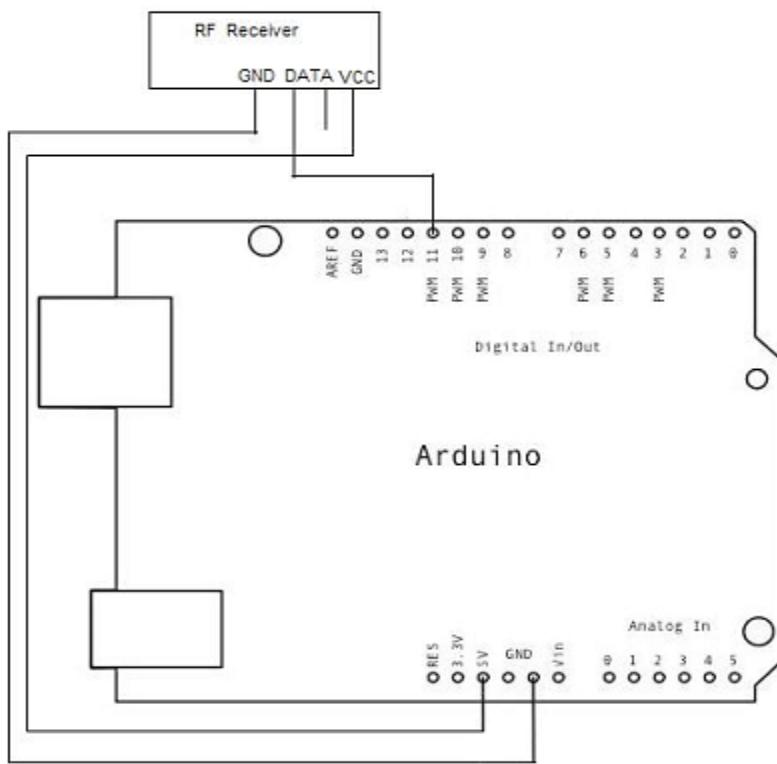


그림 66-3 수신 모듈 연결

테스트 환경에 따라 두개의 아두이노 보드, 또는 1 개의 아두이노 보드에 연결되는 포트를 지정하여 사용할 수 있습니다.

무선 송신 코드입니다.  
예제코드)

```
#include <VirtualWire.h>

char *controller;
void setup() {
    pinMode(13,OUTPUT);
    vw_set_ptt_inverted(true); //
    vw_set_tx_pin(12);
    vw_setup(4000); // speed of data transfer Kbps
}

void loop() {
    controller="1" ;
    vw_send((uint8_t *)controller, strlen(controller));
    vw_wait_tx(); // Wait until the whole message is gone
}
```

```

digitalWrite(13,1);
delay(2000);
controller="0" ;
vw_send((uint8_t *)controller, strlen(controller));
vw_wait_tx(); // Wait until the whole message is gone
digitalWrite(13,0);
delay(2000);
}

```

The D13 LED On the arduino board must be turned ON when received character '1' and Turned Off when received character '0'

무선 수신 코드입니다.

(예제코드)

```

#include <VirtualWire.h>

void setup()
{
    vw_set_ptt_inverted(true); // Required for DR3100
    vw_set_rx_pin(11);
    vw_setup(4000); // Bits per sec
    pinMode(13, OUTPUT);
    vw_rx_start(); // Start the receiver PLL running
}

void loop()
{
    uint8_t buf[VW_MAX_MESSAGE_LEN];
    uint8_t buflen = VW_MAX_MESSAGE_LEN;

    if (vw_get_message(buf, &buflen)) // Non-blocking
    {
        if(buf[0]=='1')
        {
            digitalWrite(13,1);
        }
        if(buf[0]=='0')
        {
            digitalWrite(13,0);
        }
    }
}

```

## 67 아두이노 네트워크 프로그래밍

아두이노에서 네트워크 프로그래밍을 가능하게 하는 이더넷 실드 입니다.  
실드 형태 또는 모듈들이 있습니다.

키트에서 사용될 장치는 Arduino Ethernet Shield W5100 입니다.  
W5100 이더넷 칩셋은 100 Mbps 속도입니다.



그림 67-1 아두이노 이더넷 W5100 실드 R2

아두이노 스케치 IDE 프로그램에는 Ethernet 사용 가능한 라이브러리가 포함 되어 있습니다. 별도의 사용자 라이브러리를 사용하지 않아도 됩니다.

구현되는 코드상에서 아래와 같이 2 줄의 include 선언만 해주면 사용 가능합니다.

```
#include <SPI.h>
#include <Ethernet.h>
```

위의 이미지와 같은 W5100 이더넷 실드를 사용하면 그대로 예제 코드를 가져와서 사용 할 수 있습니다.

본 모듈은 마이크로 SD 카드 와 이더넷 칩셋이 가능한 실드입니다.

스케치 IDE 의 메뉴→예제→Ethernet→ 다수의 이더넷 예제가 포함되어 있습니다.

이더넷 상세 라이브러리의 기능을 몰라도 네트워크 접속/해제, 읽기/쓰기(송신/수신)를 할 수 있습니다.

이더넷 예제 코드는 스케치 라이브러리에서 아래의 이미지처럼 많은 예제가 있습니다.

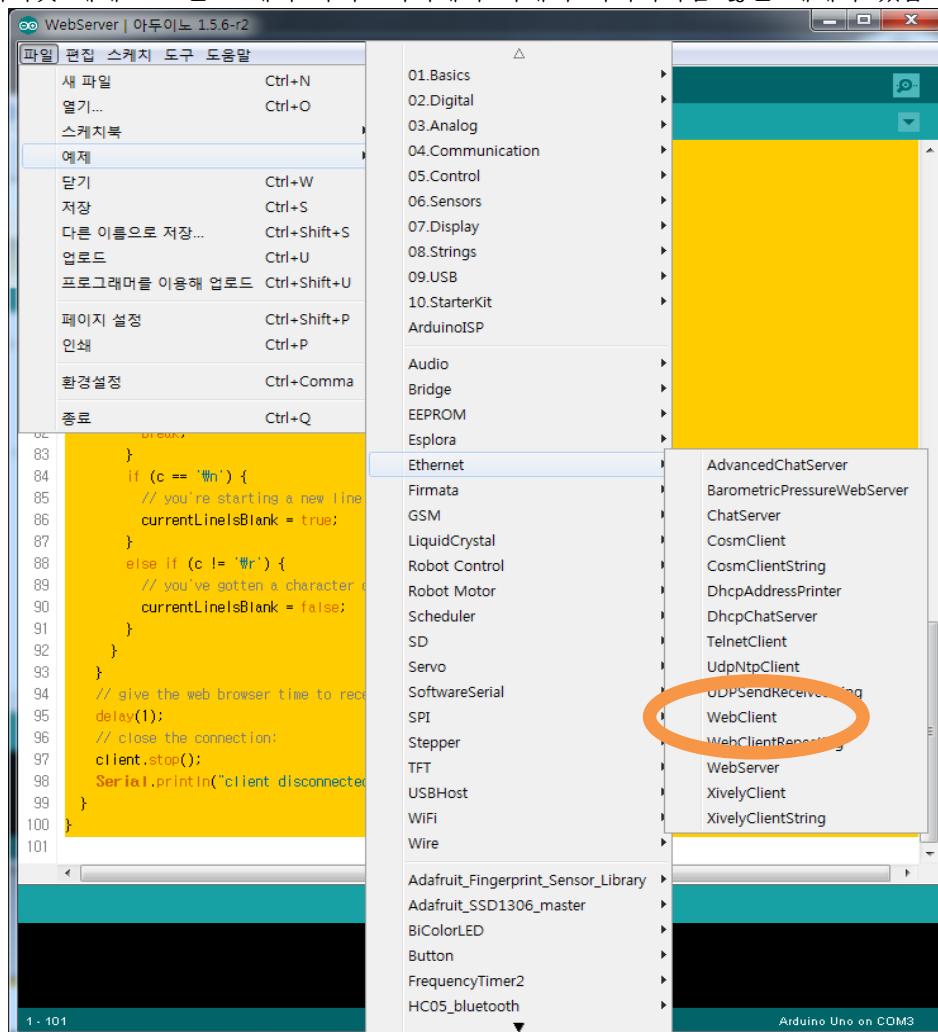


그림 67-2 이더넷 예제 선택 메뉴

아두이노 이더넷 네트워크 예제 중에 “WebClient” 항목을 열기 합니다.

이더넷 실드 W5100 사용하는 경우는 실드 장착만 하면 아두이노 스케치의 기본 라이브러리 코드를 그대로 사용 가능합니다. 실드는 내부 SPI 연결 되어 있는 상태입니다.

이더넷 실드의 RJ45 케이블(네트워크 케이블)이 인터넷 공유기에 물려 있는 상태로 테스트합니다.

코드의 작동은 인터넷 연결이 되어 있는 경우에는 구글 홈페이지의 내용을 시리얼 포트로 출력해 줍니다.

DHCP 방식 공유기 사용중인 경우 아래의 코드 중 기본으로 이해해야 될 부분은 IP 넘버 할당 받는 부분입니다.

Ethernet.begin(mac); // DHCP 할당을 받아 사용하기 위한 코드입니다.

Ethernet.begin(mac,ip); // ip 지정을 하여 사용하기 위한 코드입니다.

// 이더넷 시작

// DHCP 방식으로 ip 할당을 받습니다.

```
if (Ethernet.begin(mac) == 0)
```

```
{
```

```
    Serial.println("Failed to configure Ethernet using DHCP");
```

```
    // no point in carrying on, so do nothing forevermore:
```

```
    // try to conigure using IP address instead of DHCP:
```

```
    // 만약 DHCP ip 할당이 실패할 경우 ip 변수를 사용.
```

```
    Ethernet.begin(mac, ip);
```

```
}
```

```
/*
```

```
Web client
```

```
This sketch connects to a website (http://www.google.com)
using an Arduino Wiznet Ethernet shield.
```

```
Circuit:
```

```
* Ethernet shield attached to pins 10, 11, 12, 13
```

```
created 18 Dec 2009
```

```
by David A. Mellis
```

```
modified 9 Apr 2012
```

```
by Tom Igoe, based on work by Adrian McEwen
```

```
*/
```

```
#include <SPI.h>
#include <Ethernet.h>
```

```

// Enter a MAC address for your controller below.
// Newer Ethernet shields have a MAC address printed on a sticker on the
shield
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
// if you don't want to use DNS (and reduce your sketch size)
// use the numeric IP instead of the name for the server:
//IPAddress server(74,125,232,128); // numeric IP for Google (no DNS)
char server[] = "www.google.com"; // name address for Google (using
DNS)

// Set the static IP address to use if the DHCP fails to assign
IPAddress ip(192, 168, 0, 177);

// Initialize the Ethernet client library
// with the IP address and port of the server
// that you want to connect to (port 80 is default for HTTP):
EthernetClient client;

void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);

  // while (!Serial) {
  //   // wait for serial port to connect. Needed for Leonardo only
  // }

  // start the Ethernet connection:
  if (Ethernet.begin(mac) == 0) {
    Serial.println("Failed to configure Ethernet using DHCP");
    // no point in carrying on, so do nothing furthermore:
    // try to conigure using IP address instead of DHCP:
    Ethernet.begin(mac, ip);
  }
  // give the Ethernet shield a second to initialize:
  delay(1000);
  Serial.println("connecting...");

  // if you get a connection, report back via serial:
  if (client.connect(server, 80)) {

```

```

Serial.println("connected");
// Make a HTTP request:
client.println("GET /search?q=arduino HTTP/1.1");
client.println("Host: www.google.com");
client.println("Connection: close");
client.println();
}
else {
    // kf you didn't get a connection to the server:
    Serial.println("connection failed");
}
}

// loop 함수의 내용은 네트워크 포트에 데이터가 있을 경우 시리얼로 출력해 줍니다.
void loop()
{
    // if there are incoming bytes available
    // from the server, read them and print them:
    if (client.available()) {
        char c = client.read();
        Serial.print(c);
    }

    // if the server's disconnected, stop the client:
    if (!client.connected()) {
        Serial.println();
        Serial.println("disconnecting.");
        client.stop();
    }

    // do nothing forevermore:
    while (true);
}
}

```

## 68 웹브라우저에서 아두이노 LED 제어하기

아두이노 보드에 이더넷 기능 적용되는 경우 네트워크를 활용한 많은 프로젝트들이 가능합니다.

아두이노 보드에 간단한 웹 서버 기능을 추가하여 외부 인터넷과 웹브라우저에서 접속 후 아두이노의 포트를 제어하여 LED 점등을 할 수 있습니다.

예제코드:

[http://www.gameplusedu.com/pds/gpshop/kit\\_code/strongest/arduino\\_web\\_server\\_ex\\_1.ino](http://www.gameplusedu.com/pds/gpshop/kit_code/strongest/arduino_web_server_ex_1.ino)

```
// W5100 사용할 경우 헤더 선언입니다.  
#include <SPI.h>  
#include <Ethernet.h>  
  
// MAC address from Ethernet shield sticker under board  
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };  
IPAddress ip(192,168,1,123);  
EthernetServer server(80); // 8080 or you want  
  
String HTTP_req; // stores the HTTP request  
boolean LED_status = 0; // state of LED, off by default  
  
void setup()  
{  
    Serial.begin(9600);  
  
    // Ethernet.begin(mac, ip); //  
    Ethernet.begin(mac); //, ip); // DHCP  
    //  
    Serial.print("localIP: ");  
    Serial.println(Ethernet.localIP());  
    Serial.print("subnetMask: ");  
    Serial.println(Ethernet.subnetMask());  
    Serial.print("gatewayIP: ");  
    Serial.println(Ethernet.gatewayIP());  
    Serial.print("dnsServerIP: ");  
    Serial.println(Ethernet.dnsServerIP());  
    //
```

```

server.begin();           // start to listen for clients
pinMode(2, OUTPUT);      // LED on pin 2

}

void loop()
{
    EthernetClient client = server.available(); // try to get client

    if (client) { // got client?
        boolean currentLineIsBlank = true;
        while (client.connected()) {
            if (client.available()) { // client data available to read
                char c = client.read(); // read 1 byte (character) from client
                HTTP_req += c; // save the HTTP request 1 char at a time
                // last line of client request is blank and ends with \n
                // respond to client only after last line received
                if (c == '\n' && currentLineIsBlank) {
                    // send a standard http response header
                    client.println("HTTP/1.1 200 OK");
                    client.println("Content-Type: text/html");
                    client.println("Connection: close");
                    client.println();
                    // send web page
                    client.println("<!DOCTYPE html>");
                    client.println("<html>");
                    client.println("<head>");
                    client.println("<title>Arduino LED Control</title>");
                    client.println("</head>");
                    client.println("<body>");
                    client.println("<h1>LED</h1>");
                    client.println("<p>Click to switch LED on and");
                    off.</p>");
                    client.println("<form method=\"get\">");
                    ProcessCheckbox(client);
                    client.println("</form>");
                    client.println("</body>");
                    client.println("</html>");
                    Serial.print(HTTP_req);
                    HTTP_req = ""; // finished with request, empty
                    string
                    break;
                }
                // every line of text received from the client ends with \r\n
                if (c == '\n') {

```

```

        // last character on line of received text
        // starting new line with next character read
        currentLineIsBlank = true;
    }
    else if (c != '\r') {
        // a text character was received from client
        currentLineIsBlank = false;
    }
} // end if (client.available())
} // end while (client.connected())
delay(1);           // give the web browser time to receive the data
client.stop(); // close the connection
} // end if (client)
}

// switch LED and send back HTML for LED checkbox
void ProcessCheckbox(EthernetClient cl)
{
    if (HTTP_req.indexOf("LED2=2") > -1) { // see if checkbox was clicked
        // the checkbox was clicked, toggle the LED
        if (LED_status) {
            LED_status = 0;
        }
        else {
            LED_status = 1;
        }
    }

    if (LED_status) { // switch LED on
        digitalWrite(2, HIGH);
        // checkbox is checked
        cl.println("<input type=\"checkbox\" name=\"LED2\" value=\"2\" \n" +
                  "onclick=\"submit()\" checked>LED2");
    }
    else { // switch LED off
        digitalWrite(2, LOW);
        // checkbox is unchecked
        cl.println("<input type=\"checkbox\" name=\"LED2\" value=\"2\" \n" +
                  "onclick=\"submit()\">LED2");
    }
}

```

>>> 모든 네트워크 통신 기기는 주소가 있어야 합니다. 유선, 무선 포함 모두 고유의 주소가 있어야 통신이 가능합니다.

모든 이더넷 기기는 통신을 하기 위해서는 IP 주소(Internet Protocol address)가 있어야 합니다. 로컬 네트워크상에서도 ip 넘버는 지정 되어 있어야 합니다.

ip 설정: 대부분 ip 공유기에 연결하여 사용하므로 DHCP 설정으로 ip 할당을 받도록 합니다.

```
// 지정된 ip 넘버 사용하는 경우  
Ethernet.begin(mac, ip); // use ip number.
```

```
// DHCP 자동으로 IP 주소 할당 받아서 사용하기.  
Ethernet.begin(mac); //, ip); // DHCP 사용  
//
```

>>>

ip 주소를 할당 받기 위해서는 MAC 주소(media access control address)가 있어야 합니다. MAC 주소는 이더넷 통신 칩의 고유 주소입니다. 각각의 집에는 주소가 있는 것처럼 통신기기들은 모두 주소가 있습니다.

```
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
```

%% byte 의 의미는 unsigned char (부호 없는 문자열 타입)입니다. %%

아두이노에서의 이더넷 모듈(또는 네트워크 모듈)은 하드웨어 단계의 제어 개념이라 마음대로 MAC 주소의 변경이 가능합니다. 위에 표시되는 MAC 주소는 원하는 주소 값으로 변경하여 주어도 무방합니다.

>>>

```
// IP 주소 선언 변수.  
IPAddress ip(192.168.1.123);
```

```
// 아두이노 웹 서버 포트 지정. // 8080, 또는 원하는 포트 지정 가능.  
EthernetServer server(80); // 8080 or you want
```

192.168.1.123 이라는 ip 주소를 사용하겠다는 변수 선언입니다. 공유기 ip 대역에서 사용 중이 아닌 ip 넘버를 지정하여 주면 됩니다.

EthernetServer 클래스 변수를 선언하여 줍니다. server(80); 은 포트번호 80을 지정하여 사용한다는 의미입니다. 다른 주소를 사용하고 싶은 경우 변경하여 주면 됩니다. 포트 번호는 0 ~ 65535 까지입니다.

만약 9999 라는 포트를 사용하고 싶다면

EthernetServer server(9999); 기입하여 주면 됩니다.

그럼, PC의 인터넷 익스플로러 등에서 접속하기 위해서는 주소 입력 줄에서

<http://192.168.1.123:9999> 입력하여 주면 정상 작동되는 경우 웹 페이지가 보입니다.

EthernetServer server(80); 이라고 선언하여 주는 경우에는 <http://192.168.1.123> 까지만 주소 입력을 해주어도 무방합니다. 기본 포트 지정은 80 포트로 되어 있어서 포트 번호까지는 입력 해주지 않아도 됩니다. 물론, <http://192.168.1.123:80> 입력과 동일하게 작동 됩니다.

## » IP 번호 사용 및 확인

필요에 의해 지정된 ip 넘버를 사용하는 경우 공유기의 ip 클래스 넘버를 알아야 합니다. 즉, 대부분의 내부 로컬 ip 번호 대역은 192.168.XXX.XXX 지정되어 사용되고 있습니다.

192.168.0.XXX 또는 192.168.1.XXX, 192.168.10.xxx 사용 되므로 공유기 설정을 참조하여 ip 지정을 해주면 됩니다. 만약 192.168.1.xxx 를 사용하는 공유기인 경우 웹브라우저에서 <http://192.168.1.1> 주소를 열면 공유기 설정 화면이 보입니다. 공유기 제조사에 따라 공유기 설정 주소는 다를 수 있으므로 제조사 홈페이지, 설명서 등을 참조 하시기 바랍니다.

사무실에서 집에 있는 아두이노에 접속하기 위해서는 포트 포워드 설정이 필요합니다. 즉, 외부 지역에서 접속을 하는 경우 로컬 이더넷에 연결된 아두이노의 ip 넘버의 포트 포워딩을 해주어야 합니다.

가정, 회사 등에서 사용하는 통신사의 모뎀은 고유 인터넷 ip 를 가지고 있습니다. SXB, KXX, LXT 등의 여러 인터넷 통신사는 고유의 ip 넘버를 서로 할당 되어 관리되고 있습니다. 물론 해외의 여러 나라들도 마찬가지입니다. 국가, 지역 등의 ip 넘버 대역은 모두 다릅니다. 보통 가정, 회사 등에서 사용하는 인터넷 모뎀은 전원을 키면 통신사에 접속되어 고정된 ip 는 아니지만 매번 ip를 할당 받아서 사용되고 있습니다.

## 69 SD 카드 읽기/쓰기

SD 카드는 PC, 노트북을 비롯하여 디지털 카메라, 스마트폰 등에 많이 사용되는 저장 장치입니다.

SD 카드의 대부분의 통신 방식은 SPI 통신 방식을 사용합니다.

아두이노와 같은 MCU 보드에 연결하여 사용 가능한 여러 모듈들이 있습니다.



그림 69-1 SD 카드 모듈과 마이크로 SD 카드 모듈



그림 69-2 SD 카드 실드

아두이노에서의 SD 카드는 주로 데이터 로거(Logger) (데이터 기록 저장 장치) 프로젝트에 많이 사용됩니다.

아두이노 스케치 IDE 기본 라이브러리에는 SD 카드 읽기/쓰기 라이브러리가 있습니다.

SD 카드 라이브러리를 사용하기 위해서는 반드시 헤더 파일을 선언해 주어야 합니다.

```
#include <SPI.h>
#include <SD.h>
```

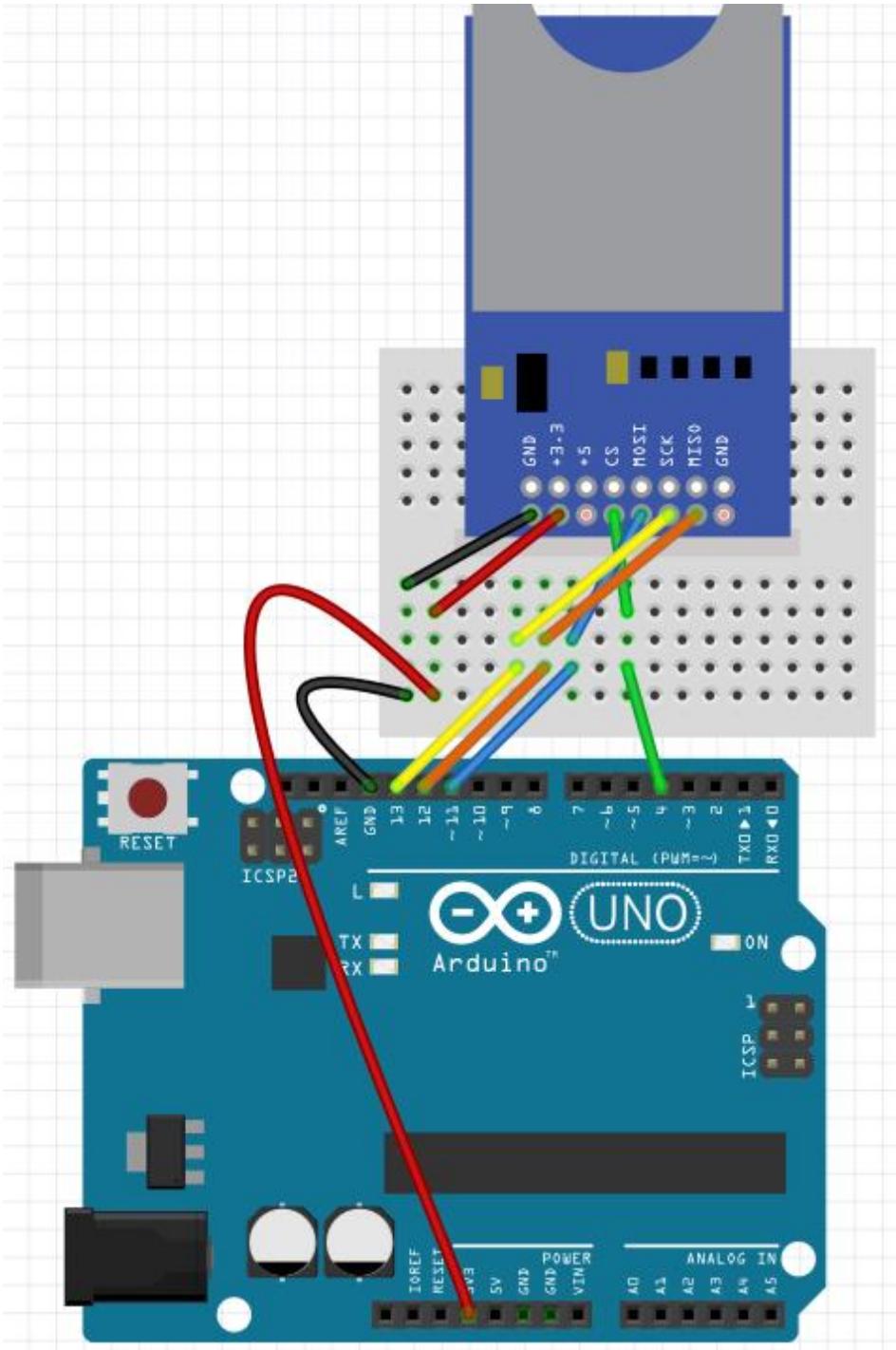


그림 69-3 브레드보드 회로도

스케치 IDE 메뉴 예제 -> SD-> listFiles 열기 합니다.

예제 코드는 SD 카드의 루트 디렉터리부터 하위 디렉터리 포함 모든 파일 목록을 시리얼 포트로 출력해 줍니다.

```
/*
Listfiles

This example shows how print out the files in a
directory on a SD card

The circuit:
* SD card attached to SPI bus as follows:
** MOSI - pin 11
** MISO - pin 12
** CLK - pin 13
** CS - pin 4

created Nov 2010
by David A. Mellis
modified 9 Apr 2012
by Tom Igoe
modified 2 Feb 2014
by Scott Fitzgerald

This example code is in the public domain.
*/



#include <SPI.h>
#include <SD.h>

// 파일 클래스 글로벌 변수 선언.
File root;

void setup()
{
    // Open serial communications and wait for port to open:
    Serial.begin(9600);
    // while (!Serial) {
    //     ; // wait for serial port to connect. Needed for Leonardo only
    // }
```

```

Serial.print("Initializing SD card...");
// On the Ethernet Shield, CS is pin 4. It's set as an output by default.
// Note that even if it's not used as the CS pin, the hardware SS pin
// (10 on most Arduino boards, 53 on the Mega) must be left as an output
// or the SD library functions will not work.
pinMode(10, OUTPUT);

if (!SD.begin(4)) {
    Serial.println("initialization failed!");
    return;
}
Serial.println("initialization done.");

root = SD.open("/");
printDirectory(root, 0);

Serial.println("done!");
}

void loop()
{
    // nothing happens after setup finishes.
}

void printDirectory(File dir, int numTabs) {
    while(true) {

        File entry = dir.openNextFile();
        if (!entry) {
            // no more files
            break;
        }
        for (uint8_t i=0; i<numTabs; i++) {
            Serial.print('\t');
        }
        Serial.print(entry.name());
        if (entry.isDirectory()) {
            Serial.println("/");
        }
    }
}

```

```
    printDirectory(entry, numTabs+1);
} else {
    // files have sizes, directories do not
    Serial.print("\t\t");
    Serial.println(entry.size(), DEC);
}
entry.close();
}
}
```

## 70 BREADBOARD POWER BOARD 2 LOAD 3V3, 5V

브레드보드 전원 공급장치 입니다. 입고 순서에 따라 검정색 PCB 형태와 약간의 PCB 형태가 다를 수도 있습니다.

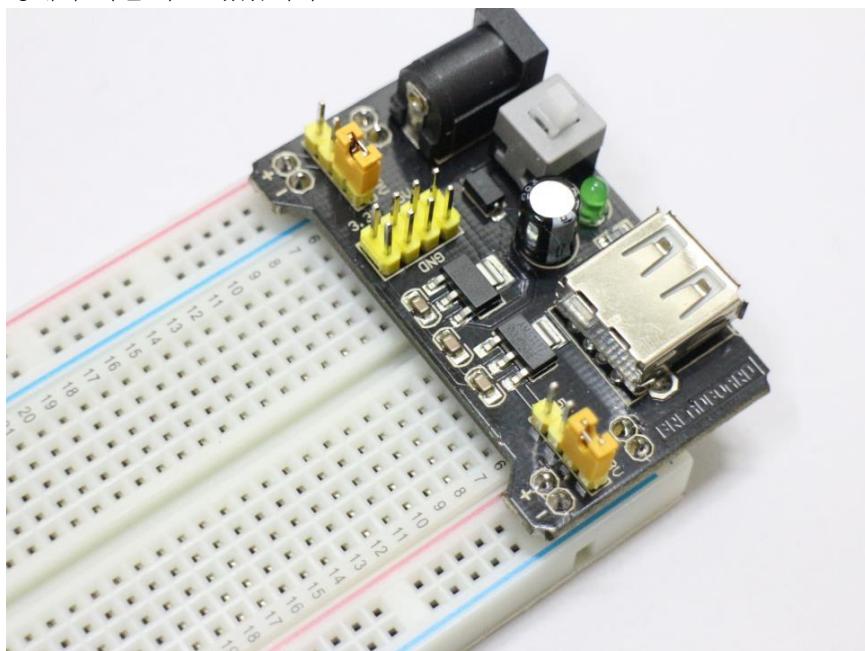


그림 70-1 브레드보드에 장착된 모습

USB 및 DC 어댑터의 전원을 받아 브레드보드에 전원을 공급하여 주는 장치입니다. 브레드보드에 전원공급 모듈로서 각각의 전원버스에 5V/3.3V 를 동시에 사용 가능한 제품입니다. 출력은 5V/3.3V 전원 선택 가능하며, 개별 전원 공급 가능한 제품입니다. 각각의 오른쪽, 왼쪽 상단에 3V3, 5V 전원 설정 가능한 스위치가 있습니다.

전원 연결은 USB A Type & DC 입니다. DC 입력 범위는 7V~12V 사용할 수 있습니다. 브레드보드의 앞 번호 부분에 장착해야 +, - 방향이 맞습니다. 주의하시기 바랍니다.



그림 70-2 장착시 주의 사항 부분

반드시 +, -

방향 주의하여 부착

## 71 아두이노 지그비 S2 통신

### 71.1 XBEE 소개

XBEE (지그비, 직비, 엑스비)

“지그비 통신은 근거리 무선 통신의 대표적으로 사용되는 통신 규약입니다.”

기존의 음성통화, 비디오, 유/무선 네트워크 등의 데이터에 대한 표준은 많이 있습니다. 하지만 MCU 기반의 센서, 컨트롤 장치 등의 많은 데이터 전송이 필요로 하지 않고, 장시간의 사용을 위해 낮은 에너지 소비를 필요로 하는 곳에 적절한 무선 통신의 필요에 의해 확립된 무선 통신 표준중의 하나입니다.

현재의 지그비는 저비용, 저전력의 무선 망사형(mesh) 네트워크의 표준으로 원거리, 근거리까지 널리 사용되고 있습니다.

지그비(ZigBee, 지그비, 직비)의 어원은 zig-zag 와 Bee 의 합성어로 벌(bee)이 이리저리 날아다니면서(zig-zag) 다른 벌들과 의사소통을 하는 것에서 착안하여 벌처럼 단순하지만 효율적인 통신을 가능하게 한다는 의미에서 붙여진 이름이라고 합니다.

지그비의 저비용 특성은 지그비가 무선 제어 및 모니터링 분야에 널리 설치되는 것을 가능하게 하였고, 저전력 특성은, 작은 배터리로, 장치를 오래 사용하는 것을 가능하게 하였다. 그리고, 망사형 네트워크는 넓은 통신 범위와 높은 안전성을 제공해 줍니다.

지그비는 산업, 과학, 의료용 기기에서 사용 가능한 주파수 대역인, ISM 무선 대역에서 동작합니다.

유럽에서는 868 MHz 대역을, 미국 및 오스트레일리아에서는 915 MHz 대역을, 나머지 지역에서는 2.4 GHz 주파수 대역을 사용합니다.

지그비 기술은, 블루투스 같은 다른 WPANs 에 비해서 간단하고, 저렴하게 구성 가능합니다.

지그비는 15 ms 이내에 활성화(슬립 모드에서 액티브 모드로) 될 수 있으며, 대기 시간이 매우 짧아서, 장치의 반응성이 매우 좋습니다.

그러므로, 지그비는 대부분의 시간을 슬립 모드 상태로 있을 수 있고, 평균 전력 사용이 매우 적다. 지그비의 긴 배터리 시간은 이렇게 해서 가능한 것이다.

지그비 프로토콜은, 저전력을 사용하고, 낮은 데이터 전송속도를 요구하는 어플리케이션에 맞도록 설계 되어 있습니다. 지그비의 현재 목표는 범용이고, 저렴하면서, 스스로 구성하는 망사형 네트워크를 만드는 것이다. 이런 망사형 네트워크는 산업용 제어, 내장형 센서, 의료 데이터 수집, 연기 혹은 침입자 경고, 빌딩 자동화, 홈 오토메이션 등에 사용될 수 있습니다.

현재 지그비 기기 인증을 통과하기 위해서는 각각의 장치가 최소 2 년 이상 지속되는 배터리를 가져야 한다고 합니다.

## 71.2 지그비 무선 통신 규약

지그비는 IEEE 802.15.4-2003 을 기반으로 한 작고, 저전력의 디지털 라디오를 사용하는 하이 레벨 통신 프로토콜입니다. IEEE 802.15.4-2003 는 단거리 라디오 주파수를 사용하는 램프, 전자계량기, 소비자용 전자제품과 같은 근거리 개인 무선통신망의 기준이다. 지그비는 낮은 데이터와 적은 배터리 소모, 네트워크의 안전성을 요구하는 RF 어플리케이션에 주로 사용된다.

지그비 통신을 하고자 하는 MCU 보드 또는 PC 기기는 각각의 지그비 모듈이 장착되어야 합니다. 각각의 부착된 지그비 모듈은シリ얼 통신(UART 통신) 방식으로 부착 기기와의 통신 및 설정을 하게 됩니다.

## 71.3 지그비 디바이스 설정

지그비 디바이스에는 3 가지 종류가 있습니다.

개별적인 지그비 모듈에 대한 네트워크상의 역할 설정을 해주어야 합니다.

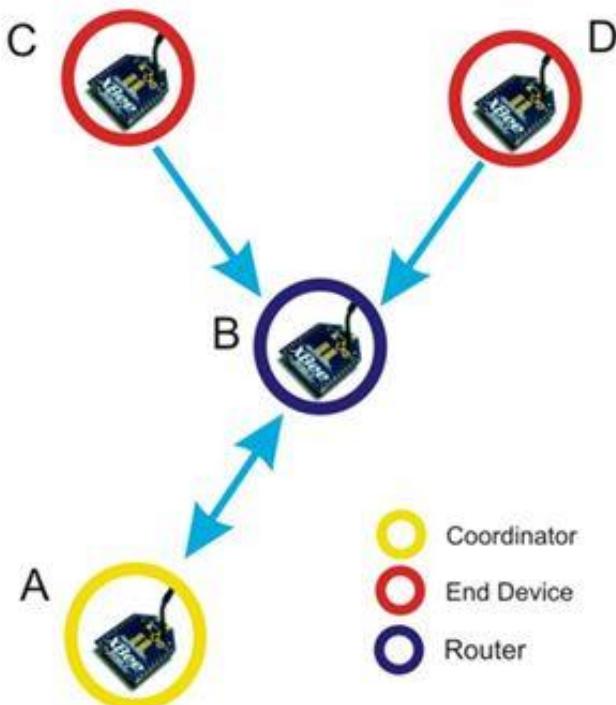


그림 71-1 지그비 S1 통신 구성도

아래의 그림처럼 여러 개의 라우터 노드에 엔드 노드가 설정되는 경우는 아래와 같습니다. 메시(Mesh) 네트워크 구성입니다.

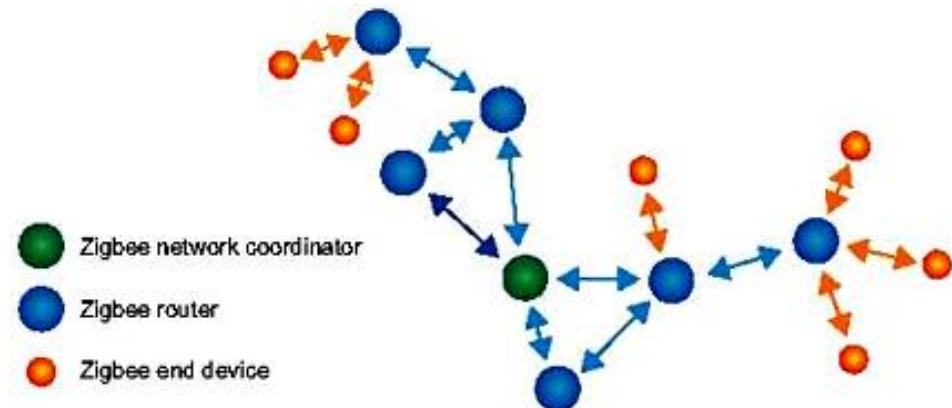


그림 71-2 지그비 S2 통신 구성도

Mesh Network(메시 네트워크), 또는 방사형 네트워크는 기존의 중계기 또는 무선 공유기를 통해 신호를 연결해가는 방식이지만

#### 지그비 코디네이터

가장 중요한 디바이스로 네트워크를 형성하고 다른 네트워크들과 연결 시킵니다. 각각의 네트워크에는 단 한 개의 코디네이터가 존재하게 됩니다.

지그비 코디네이터는 네트워크에 관한 정보를 저장할 수 있고, trust center 또는 보안 키를 위한 저장소로서의 역할도 수행 할 수 있습니다.

#### 지그비 라우터

라우터는 애플리케이션 기능뿐만 아니라, 다른 디바이스로부터의 데이터를 전달할 수 있는 라우터로서의 기능도 할 수 있습니다.

#### 지그비 엔드 디바이스

지그비 엔드 디바이스는 부모 노드와 통신할 수 있는 기능을 포함한다.

엔드 디바이스로 설정되어 있는 경우 오랜 시간을 대기할 수 있도록 하여 배터리 수명을 더욱 길게 연장할 수 있다. S2 같은 모듈은 대기모드와 작동모드의 전환 반응 시간이 빠르므로 장시간의 배터리 사용을 가능하게 합니다.

## 71.4 XBEE S2 소개



그림 71-3 XBee S2 모듈

지그비(직비) S2 모듈은 S1(PRO) 버전을 보완한 제품이라고 보면 됩니다.

S1은 1:1 통신만 가능한 구성이었으나 S2는 1:N 통신이 가능한 모듈입니다.

지그비 모듈들은 서로 통신하기 위해, 동일한 ID를 먼저 설정하고 사용해야 합니다.

동일한 ID를 설정한 후에는 일반적인 시리얼 통신처럼 사용하면 됩니다.

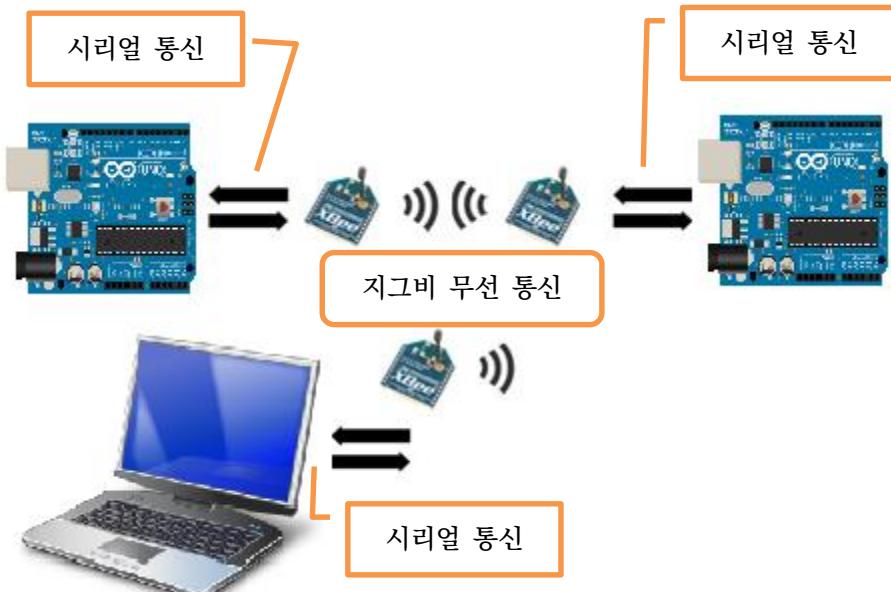


그림 71-4 지그비 통신 개념도

기본적으로 지그비 S1, S2 는 통신 작동 방식에 대한 지정을 해야 사용 가능합니다.

지그비 S1 과 지그비 S2 비교

| XBee 802.15.4   | XBee ZigBee  |
|---|--|
| 시리즈 1<br>series 1 or S1 표기  | 시리즈 2<br>series 2 or S2 표기   |
| Point to point (1:1 통신)   | Point to point, Star, Mesh, 1:N, N:1   |
| up to 300 ft. (100m)  | up to 400 ft. (120m)   |
|  XBee Series1 |  XBee Series2 |
| 간단한 초기 설정 후 사용.   | 간단한 초기 설정 후 사용.  |

## 71.5 지그비 S2 설정

### 71.5.1 지그비 S2 USB Adaptor 연결

XBee USB Adaptor 를 사용하여 PC 에서 설정 할 수 있습니다.



그림 71-5 XBee USB 어댑터 모듈

먼저 지그비 USB 어댑터 보드와 지그비 모듈 S2 를 장착을 합니다.  
USB 케이블은 Mini B 타입입니다.



그림 71-1 XBEE 모듈 장착된 모습



그림 71-6 지그비 모듈 장착된 모습

X-CTU 프로그램과, 일반 시리얼 통신 프로그램을 사용하여 설정할 수 있습니다. USB 시리얼 변환기를 비롯하여 설정에 관련된 여러 가지의 장치들이 있습니다. 지그비 모듈의 펌웨어 업로드 기능까지 사용하는 경우 위와 같은 USB 시리얼 변환기 모듈이 필요합니다. 대부분의 MCU 펌웨어 업로드 기능에는 리셋 기능이 필요합니다. PC의 USB 포트에 지그비 USB 어댑터를 연결하면 FTDI USB 범용 시리얼 칩이므로 윈도우 7,8 등의 최신 OS에서는 자동으로 드라이버 설정이 됩니다. 만약 USB 시리얼 포트로 인식 안 되는 경우에는 <http://www.ftdichip.com>에서 VCP 드라이버를 다운로드 받아 설치를 하도록 합니다.

아래의 그림과 같이 장치 관리자의 화면에서 “USB Serial Port(COM13)”으로 보입니다. 사용자 환경에 따라 시리얼 포트 번호는 다릅니다. 지그비 설정 할 때 시리얼 포트에 접근되어 설정되므로 포트 번호를 기억하도록 합니다.

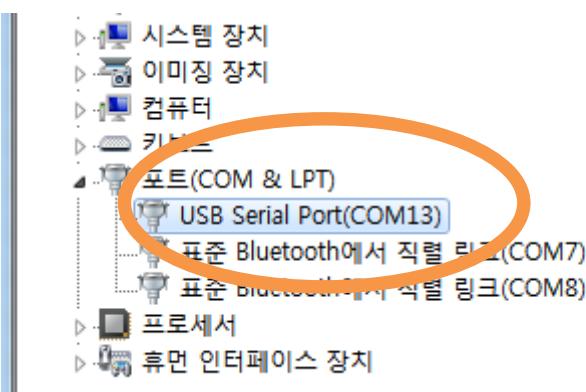


그림 71-7 USB 시리얼 포트 정보

## 71.6 지그비 S2 설정 방법 1 X-CTU 사용

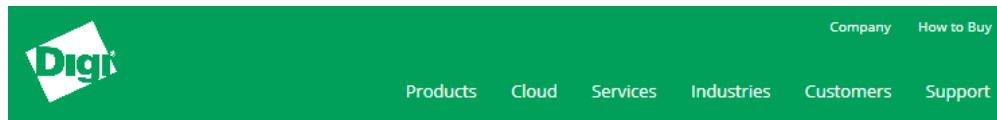
제조사에서 배포하는 XBee 설정 및 관리 프로그램 X-CTU 사용합니다.

X-CTU 프로그램 사용하여 설정하는 방법.

다운로드 사이트

<http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/xctu>

X-CTU 프로그램을 다운로드 받아 설치를 합니다.

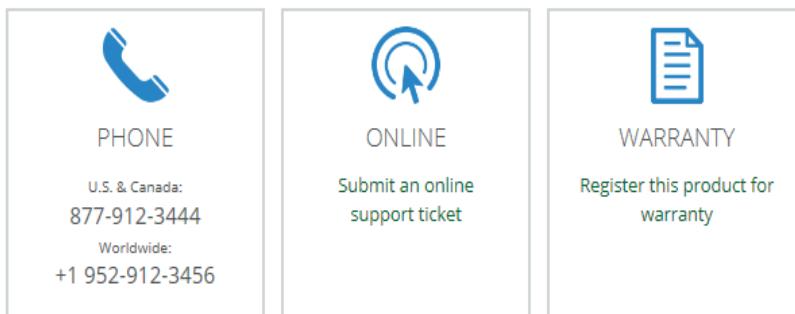


- X-CTU is a **free, multi-platform** application compatible with Windows and MacOS
- **Graphical Network View** for simple wireless network configuration and architecture
- **API Frame Builder** is a simple development tool for quickly building XBee API frames
- **Device Cloud** integrated, allowing configuration and management of XBee devices anywhere in the world

**DOWNLOAD**



# XCTU Software



Product Status: Active

Support Status: Web, Email, Phone

[Toggle all](#)

[RSS Feed](#)

**Diagnostics, Utilities and MIBs**

**Next Generation XCTU**

- [XCTU Next Gen Installer, Windows x32/x64](#)
- [XCTU Next Gen Installer, MacOS X](#)
- [XCTU Next Gen Installer, License Agreement](#)

**Legacy XCTU**

**XCTU ver. 5.2.8.6 installer** A solid version of XCTU. Contains features from previous versions, plus adds support for XBee Wi-Fi modules, required for XTend firmware updates. Compatible with Windows 2000, XP, 2003, Vista, 7.

- [XCTU 32-bit ver. 5.2.8.6 installer release notes](#)
- [XCTU ver. 5.1.0.0 installer](#)

최근 버전을 설치하면 아래의 UI 인터페이스를 가진 프로그램이 실행됩니다.  
설치 후 실행하면 아래와 같은 프로그램이 실행됩니다.



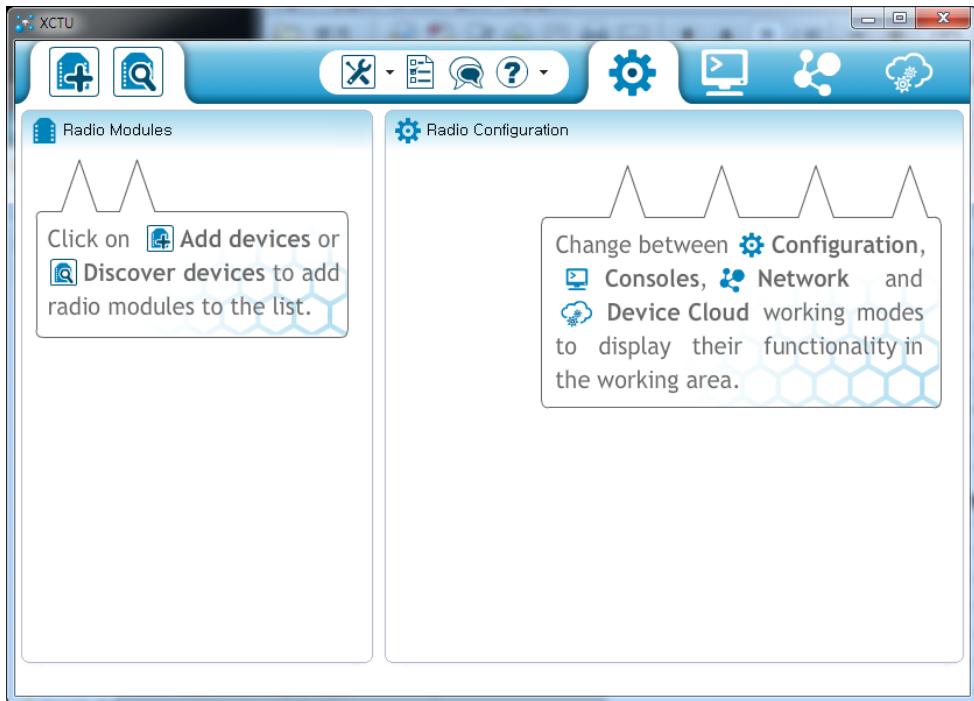


그림 71-8 XCTU 설정 프로그램 메인 화면

지그비 모듈 추가 버튼을 누릅니다.

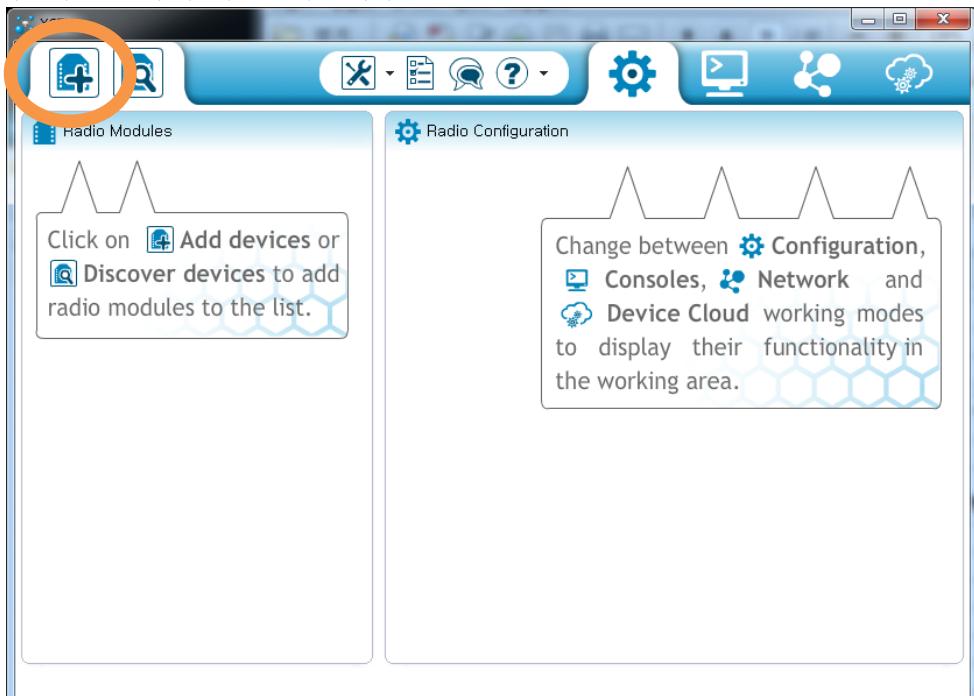
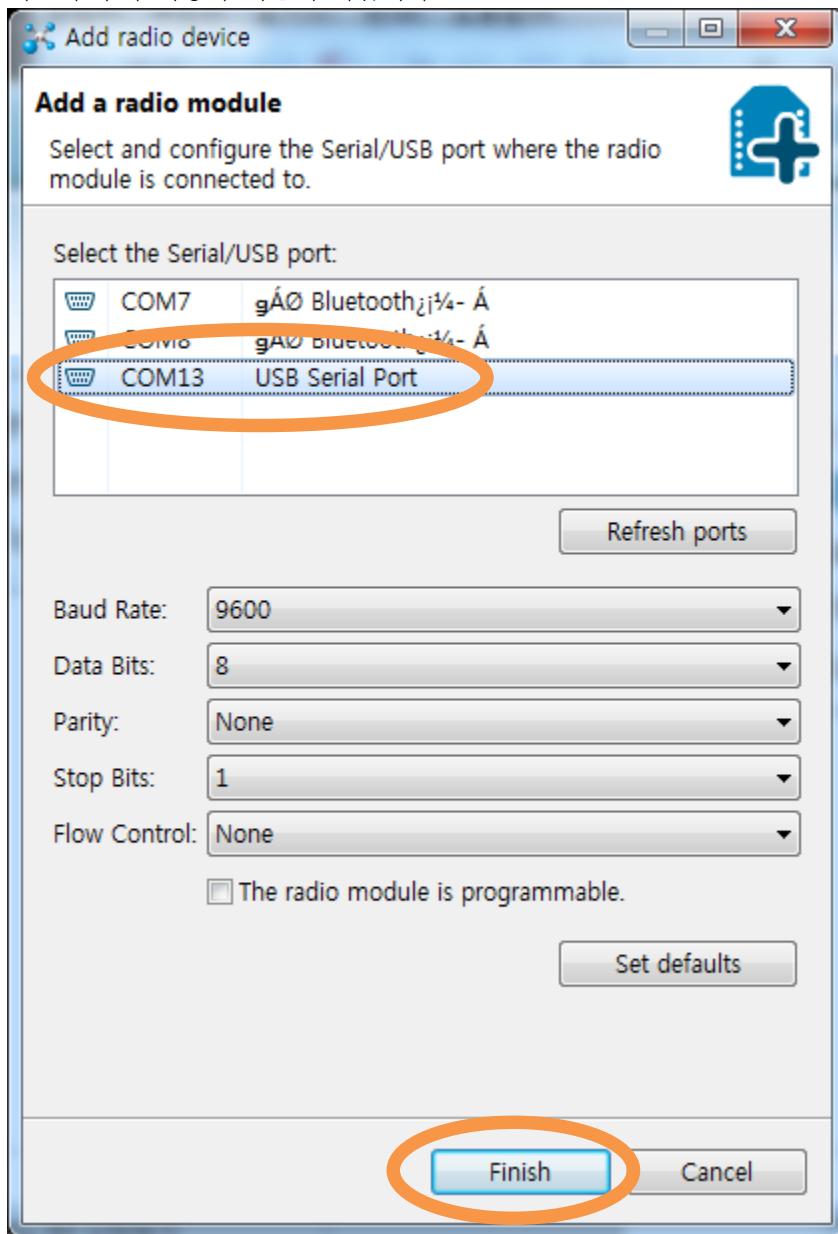


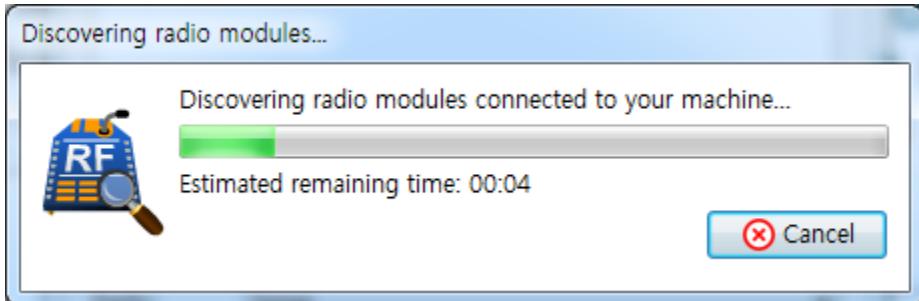
그림 71-9 XCTU 추가 버튼

지그비 추가 사용자 화면이 나옵니다.

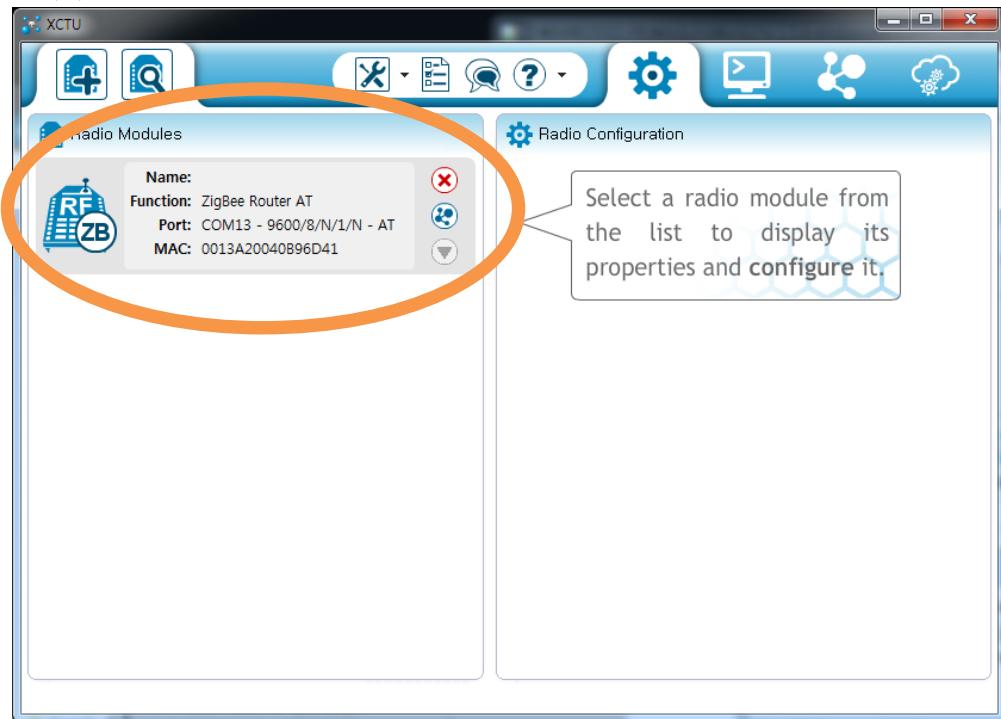


지그비 USB 어댑터가 설정되어 있는 포트를 선택 후 “Finish” 버튼을 눌러 줍니다.

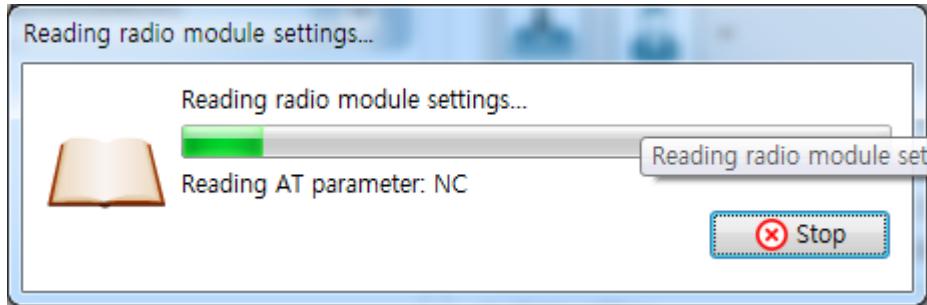
지그비 모듈을 찾는 진행 상황ダイヤログ가 표시됩니다.



정상적으로 찾게 되면 아래와 같은 화면이 보입니다. 왼쪽에 찾은 지그비 모듈이 보입니다.



마우스로 선택하면 검색된 지그비 모듈의 실제 세부 정보를 가져오게 됩니다.

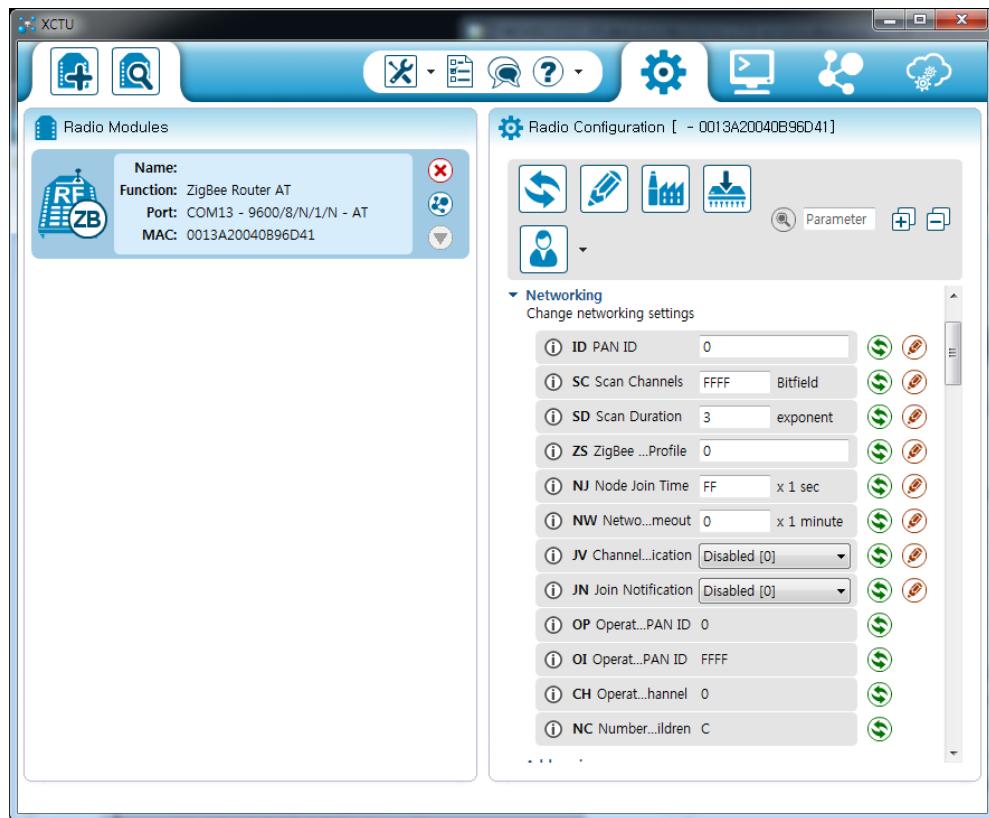


먼저 ID 를 지정하여 줍니다.

사용되는 지그비 모듈의 ID 는 반드시 동일 해야 됩니다.

기본값은 “0” 입니다. 본인이 사용할 지그비 모듈들의 ID 를 설정하여 줍니다.

ID 값의 범위는 0 - 0xFFFFFFFFFFFFFF. 입니다.



원하는 ID 를 입력 후 저장 버튼(연필 모양 아이콘 버튼)을 누릅니다.

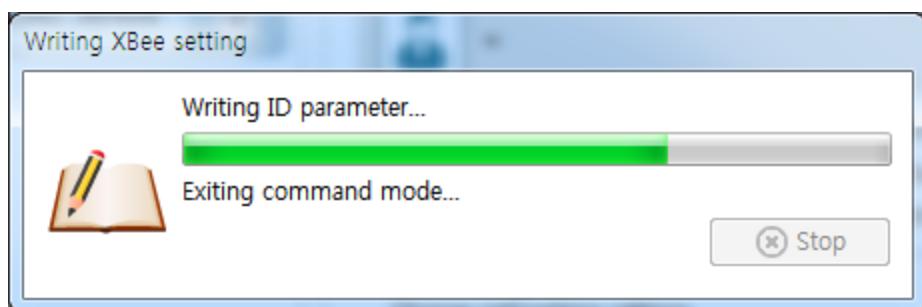
2 개의 모듈을 사용하는 경우, 3 개 모듈, 모두 ID 를 같은 번호로 사용하도록 해야 합니다.

▼ Networking  
Change networking settings

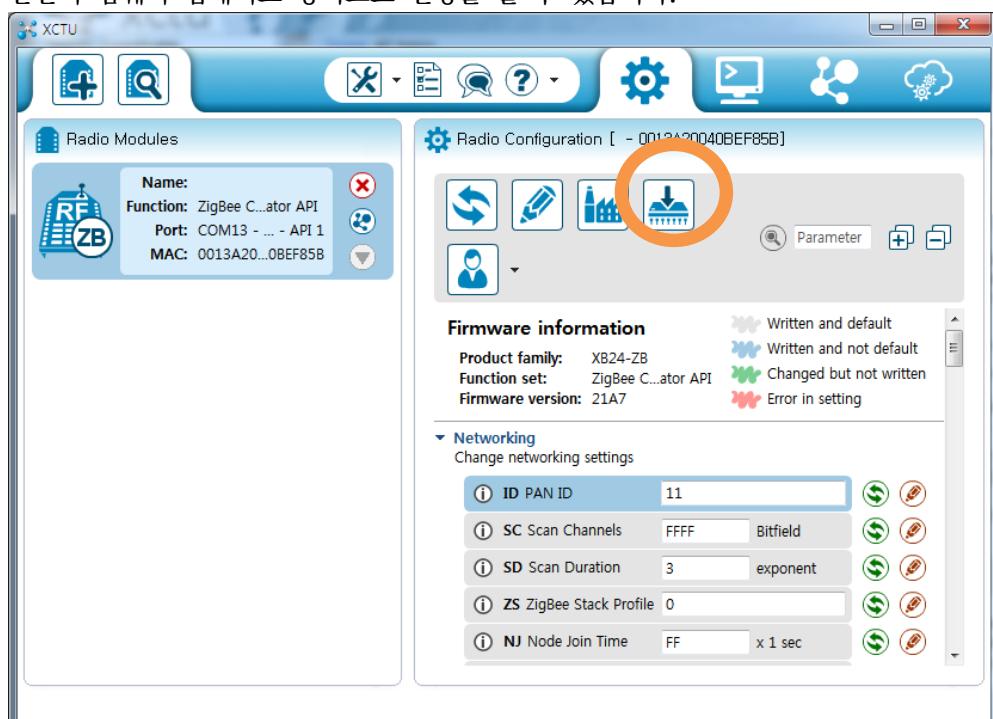
|                           |      |          |
|---------------------------|------|----------|
| ① ID PAN ID               | 11   |          |
| ① SC Scan Channels        | FFFF | Bitfield |
| ① SD Scan Duration        | 3    | exponent |
| ① ZS ZigBee Stack Profile | 0    |          |
| ① NJ Node Join Time       | FF   | x 1 sec  |

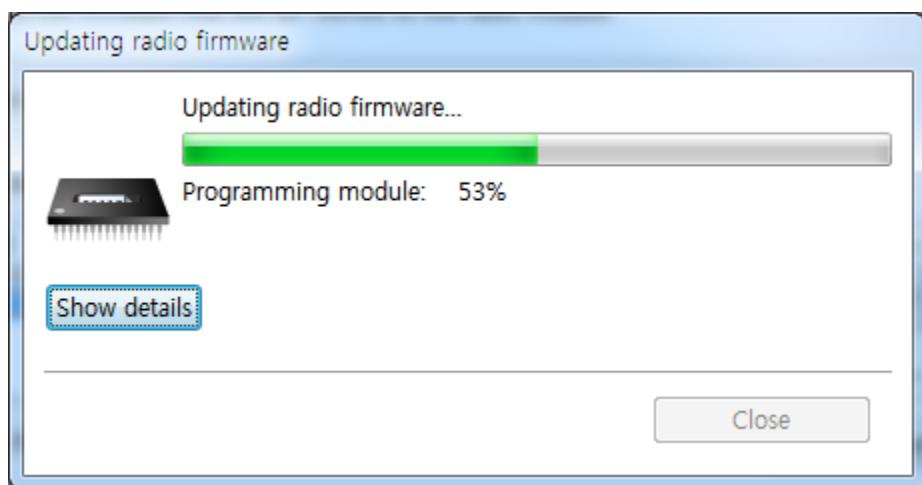
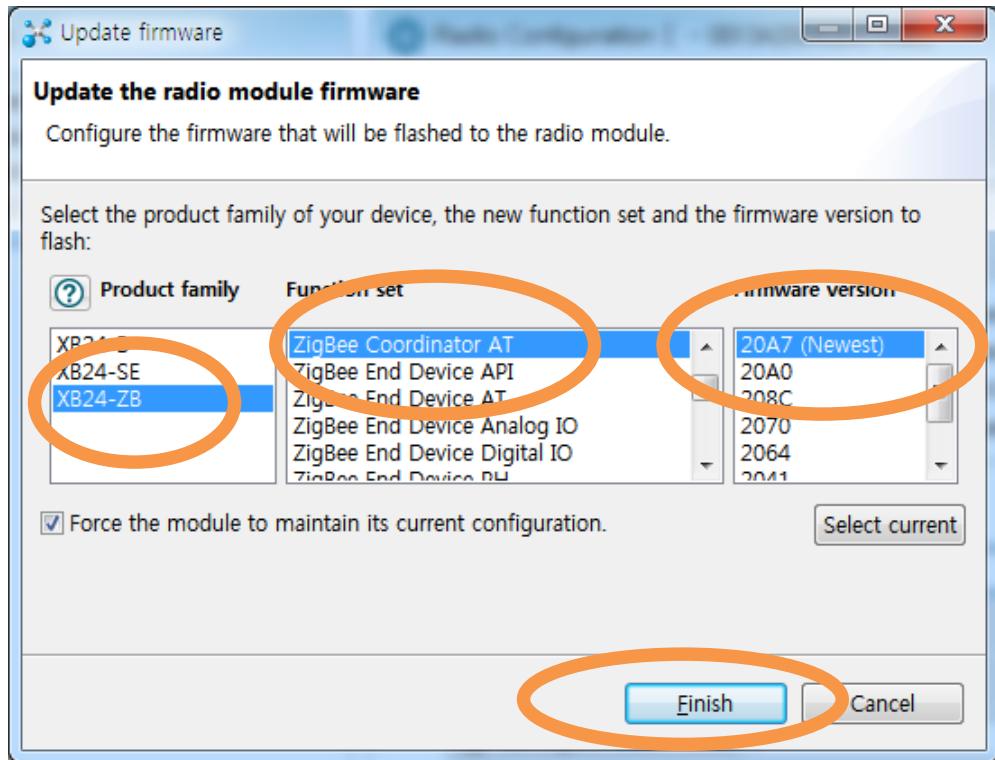


ID 저장 진행ダイアログ가 보이면서 저장이 됩니다.

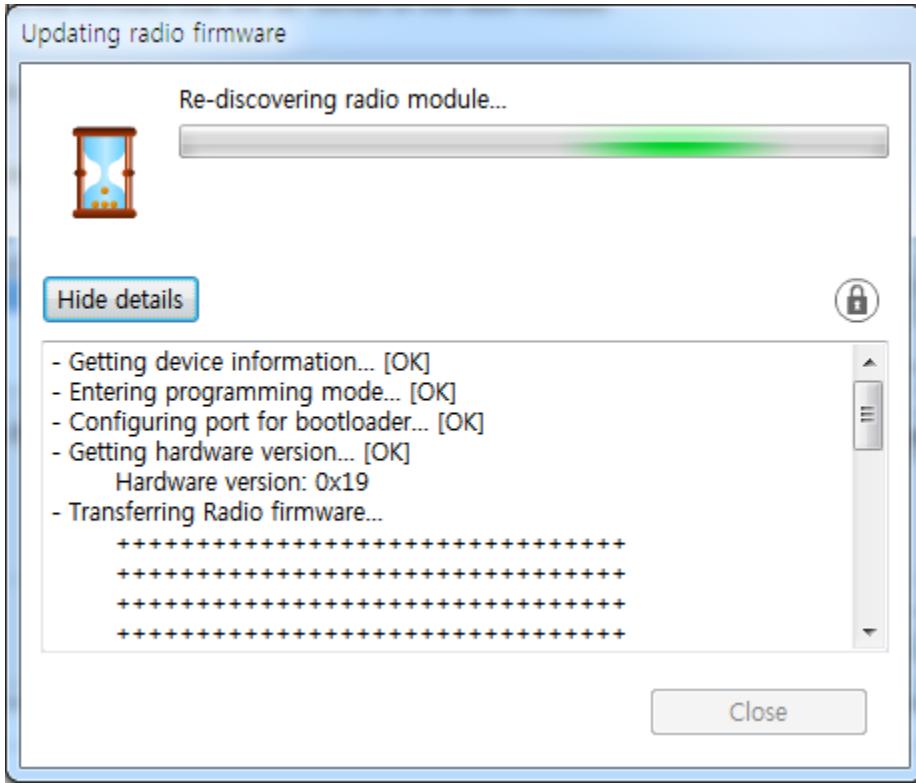


지그비 코디네이터와 라우터 앤드 노드를 설정해보도록 합니다.  
간단히 펌웨어 업데이트 방식으로 변경을 할 수 있습니다.





Update Radio firmware 진행 상황 다이얼로그의 “Show details” 버튼을 누르면 아래와 같은 상세 로그 화면을 볼 수 있습니다.



완료 후 아래와 같은 화면을 볼수 있습니다.

Parameter

Written and default

Written and not default

Changed but not written

Error in setting

Firmware information

Product family: XB24-ZB

Function set: ZigBee Coordinator AT

Firmware version: 20A7

Networking

Change networking settings

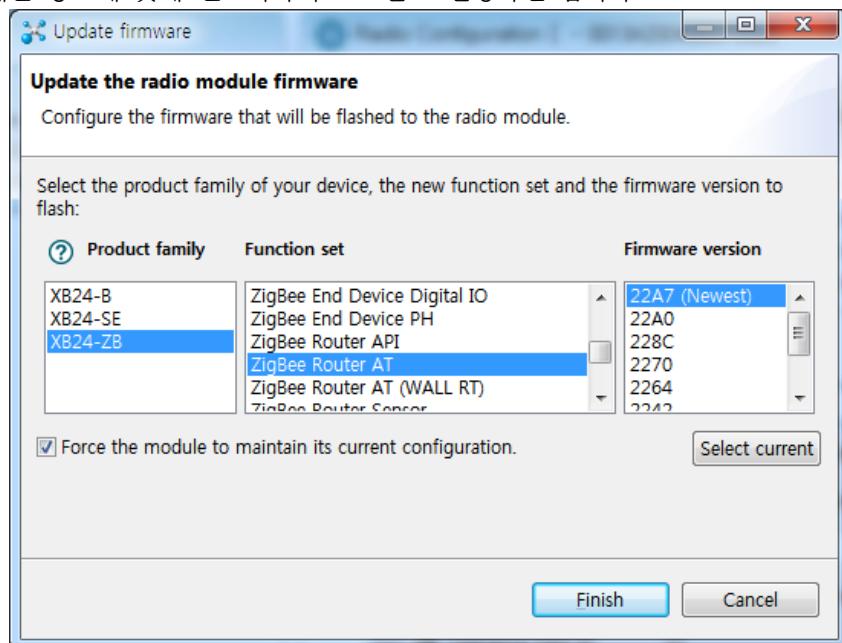
ID PAN ID: 11

SC Scan Channels: FFFF Bitfield

SD Scan Duration: 3 exponent

위의 방식으로 2 개의 지그비 모듈을 설정하도록 합니다.

코디네이터와 라우터 기능으로 설정을 하도록 합니다. 2 개 이상, 여러 개를 사용하는 경우에는 용도에 맞게 앤드디바이스 모듈로 설정하면 됩니다.



(라우터 디바이스로 펌웨어 설정)

펌웨어 설정하더라도 ID 값은 유지되므로 바로 사용 하면 됩니다.

지그비 설정에 관한 많은 도움말들이 있으므로 아래의 화면으로 진입하여 더 많은 정보를 볼 수 있습니다.



## 71.7 지그비 S2 설정 방법 2

일반 시리얼 통신 프로그램 사용하여 설정을 해보도록 합니다.

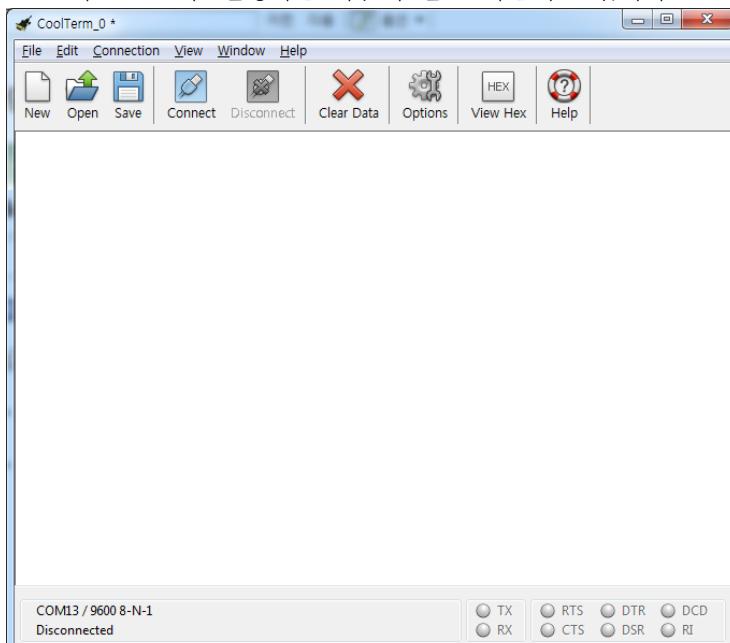
Win/Linux/Mac에 간편하면서도 쓸만한 프로그램이 있습니다.

<http://freeware.the-meiers.org/>

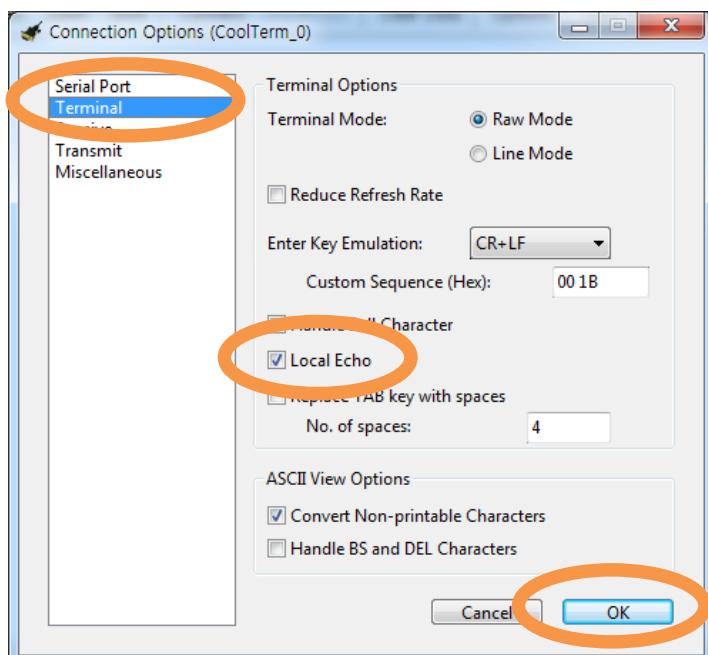
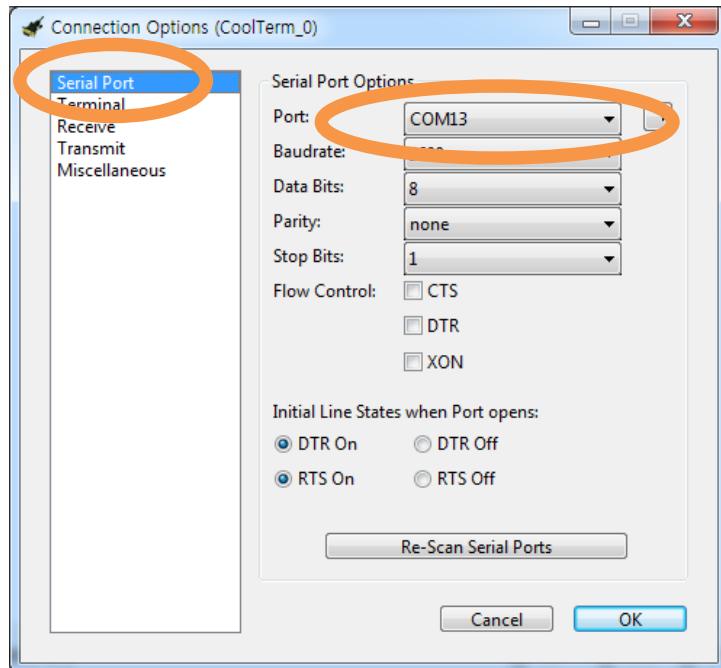
위의 주소로 방문하면 많은 프로그램들이 있습니다. 아래의 그림을 참조하여 다운로드 받으면 됩니다. 설치 없이 바로 실행되는 프로그램입니다.

The screenshot shows a software download page for 'Very useful stuff'. It lists several applications, with 'CoolTerm' being the primary focus. The 'CoolTerm' entry includes a thumbnail image of a serial port adapter, the version '1.4.4', and compatibility information for Mac, Win, and Linux. A large orange circle highlights the 'CoolTerm' row. To the right of the application details, there are sections for 'Description', 'Reviews / Awards', and links to MacUpdate, Rocky Bytes, FindMySoft, SoftSea, and SOFOTEX reviews. Below the main table, there is another row for 'BatchTouch'.

다운로드 후 실행하면 다음과 같은 화면이 보입니다.



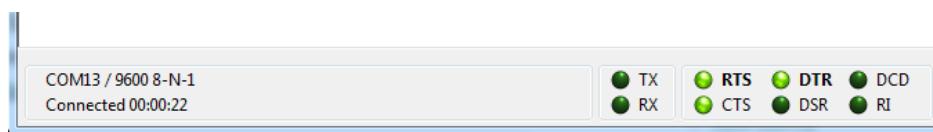
“Options” 버튼을 누르면 시리얼 포트 설정 대화상자가 나옵니다.  
포트는 사용자 PC 마다 다르므로 확인 후 선택해줍니다.  
나머지 설정은 아래 대화상자의 값처럼 동일하게 사용하면 됩니다.  
9600,N81, 그리고 Local Echo 설정해야 합니다.



“Local Echo” 옵션은 시리얼 연결 상태에서 입력되는 키보드의 문자 내용을 보여주는 기능입니다. 지역반향(반송)기능입니다.

“OK” 버튼을 눌러 설정을 마친 후 메인 프로그램의 상단 버튼의 “Connect” 버튼을 누릅니다.

정상 연결되면 하단에 아래와 같은 메시지가 보입니다.



이제 XBee S2 모듈의 기본 설정을 해주어야 합니다.

지그비 모듈 명령어 모드로 전환하도록 합니다.

“+++” 입력하면 명령어 모드로 전환됩니다.

“+++” 입력 후 일정시간 입력이 없으면 명령어 모드에서 나오게 됩니다.

“+++” 입력 시 “+++” 만 쳐야 합니다. Enter Key(엔터 키)까지 안쳐도 됩니다.  
엔터 키까지 입력하면 “OK” 메시지가 출력되지 않습니다.

정상 작동일 경우 “+++” 입력 상태에서 1 초 내로 옆에 “OK” 문자가 나옵니다.

명령어 모드 진입 후 아무런 입력이나 작동이 없이 10 초 경과 시 명령어 모드에서 빠져나옵니다.

다시 명령어 모드로 진입하려면 언제든지 “+++” 입력하면 됩니다.

“OK” 표시가 나오면 명령어 모드로 진입된 상태입니다.

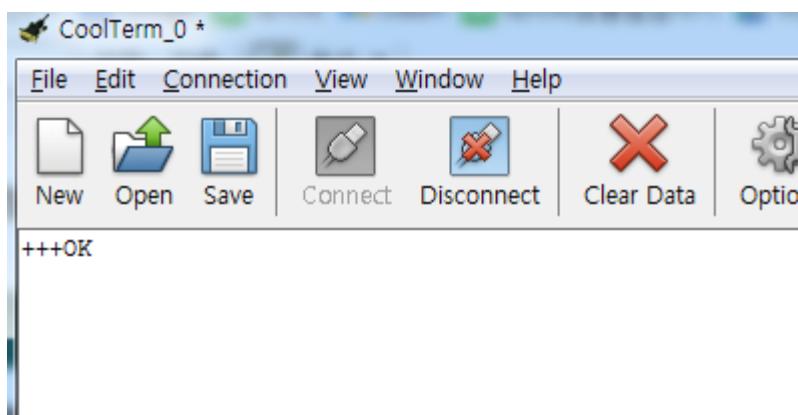


그림 71-10 명령어 모드 진입 상태

이제 기본 주소 설정을 하도록 합니다.

#### 기능

| 기능                       | 명령   | 입력 값 설명                                   |
|--------------------------|------|---|
| PAN ID                   | ATID | 2001 (any address from 0 to FFFE will do) |
| Destination address high | ATDH | 0013A200                                  |
| Destination address low  | ATDL | [see your recorded Router Address]        |

PAN ID 설정을 하기 위해 ATID 를 입력합니다.

PAN(Personal Area Network) ID 는 사용하려는 (묶어서 사용하려는) XBEE 모듈들은 같은 넘버를 지정해주어야 합니다.

이번에는 엔터까지 쳐야 명령어로 인식됩니다.

기본값은 0 으로 되어 있습니다.

ATID (엔터키) -- 현재 설정된 ATID 값을 보여줍니다.

0

ATID 2001 (엔터키) 입력하여 2001 이라는 PAN ID 설정해줍니다.

OK ( 정상일 경우 OK 표시가 됩니다.)

위와 같은 명령어 작동방식으로 ATDH 와 ATDL 까지 입력해줍니다.

+++OK

ATDH 13A300

OK

ATDL

403B9E21

ATWR

OK

최종 지그비 설정 저장은 “ATWR” 입력해 줍니다.

ATWR (엔터키)

OK

## 71.8 아두이노 지그비 실드 & 블루투스 실드

아두이노 지그비 실드입니다.

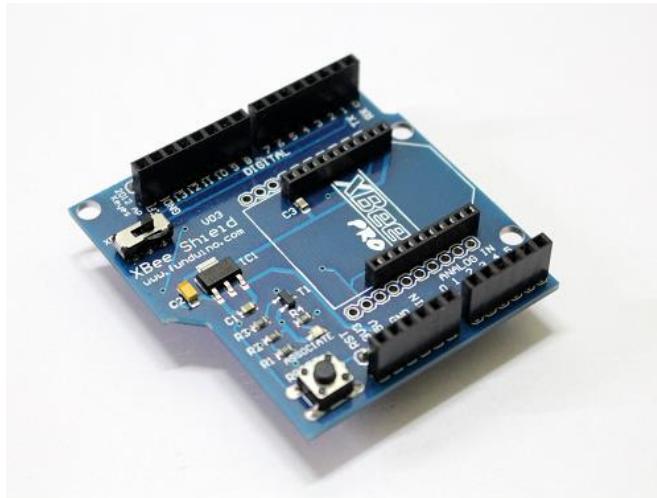


그림 71-11 XBEE 실드

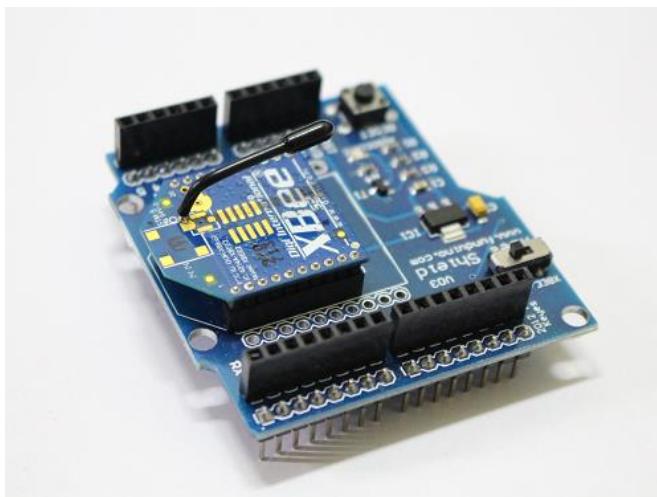


그림 71-12 XBee S2 모듈 장착된 모습

아두이노에서 지그비 모듈을 탑재하여 사용할 수 있는 적층형 실드입니다.

PC에서 각각의 지그비 모듈들을 USB 아답터에서 코디네이터 기기, 라우터 기기 등으로 설정 후 아두이노에 적용하여 사용 합니다. 물론, 아두이노에서 프로그래밍으로 지그비 모듈을 설정 할 수도 있습니다.

동일한 크기의 블루투스 기능이 되는 블루비(블루투스+BEE)라는 모듈을 장착하여 사용도 가능하게 되어 있습니다.



그림 71-2 아두이노 블루투스 블루비 HC-06 모듈



그림 71-13 XBEE / USB 통신 선택

실드를 아두이노에 적층 후 펌웨어 업로드 시 USB 방향으로 스위치를 옮기고 해야 합니다. USB 방향으로 스위치가 되어 있을 경우 지그비 모듈의 통신 부분과 차단된 상태가 됩니다. XBEE 방향으로 스위치가 되어 있을 경우에는 지그비 모듈의 통신 부분과 연결된 상태가 되어 지그비와의 통신이 가능합니다.

아래의 코드를 지그비 실드 적층 되어 있는 아두이노 우노 R3 에 업로드를 합니다. 시리얼 포트로 받은 내용을 프린트 하는 코드입니다.

```

void setup() {
    Serial.begin(9600);
}

void loop() {
    if (Serial.available() > 0)
    {
        Serial.println(Serial.read());
    }
}

```

우선 PC에서 지그비 USB 어댑터 연결된 상태에서 CoolTerm 등의 프로그램으로 연결 후 키보드로 문자를 입력하면 아두이노의 시리얼 모니터 창을 열어서 위의 코드처럼 받은 문자열이 그래도 출력이 되면 정상 작동입니다.

작동 확인 후 아두이노 2개를 연결하여 원격지에서의 통신 데이터를 주고 받을 수 있습니다. 원격 통신이 필요한 프로젝트에 여러 가지 용도로 사용 가능합니다.

PC와 아두이노와의 통신 테스트 완료 후 2개의 아두이노 보드에서 지그비 실드 각각 적층 후 송신/수신 및 데이터 교환 프로그래밍이 가능합니다.

## 72 최강 키트 구성 항목입니다.

### 72.1 최강 키트.

| 항 목                                  | 개 수 |
|--------------------------------------|-----|
| ➤ Arduino UNO R3 Board               | 1   |
| ➤ High quality and large bread board | 1   |
| ➤ RFID module                        | 1   |
| ➤ RFID IC Keychain                   | 1   |
| ➤ RFID Non-contact type IC card      | 1   |
| ➤ I2C LCD 1602 module                | 1   |
| ➤ 5v Relay 1 Channel                 | 1   |
| ➤ DS1302 clock module                | 1   |
| ➤ Voice detection module             | 1   |
| ➤ Temperature and humidity module    | 1   |
| ➤ Level detection module             | 1   |

|                                      |    |
|--------------------------------------|----|
| ➤ 4 * 4 keypad module                | 1  |
| ➤ Three-color RGB module             | 1  |
| ➤ XY joystick                        | 1  |
| ➤ Servo motor                        | 1  |
| ➤ Stepper motor driver board         | 1  |
| ➤ Green LED                          | 5  |
| ➤ Yellow LED                         | 5  |
| ➤ Red LED                            | 5  |
| ➤ 1 K ohm resistor                   | 10 |
| ➤ 10 K ohm resistor                  | 10 |
| ➤ 220 ohm resistor                   | 10 |
| ➤ Buzzer                             | 2  |
| ➤ Key module (with hat) 12x12mm      | 4  |
| ➤ Tilt Sensor/Switch                 | 2  |
| ➤ LM35 sensor module                 | 1  |
| ➤ Photo resistor                     | 3  |
| ➤ Fire sensor, Flame Sensor          | 1  |
| ➤ Infrared receiver                  | 1  |
| ➤ Adjustable potentiometer           | 1  |
| ➤ A digital control                  | 1  |
| ➤ 4 digital tube 1 Digit 7-Segment   | 1  |
| ➤ 8 * 8 dot matrix module            | 1  |
| ➤ 74HC595N chip                      | 1  |
| ➤ Infrared remote control            | 1  |
| ➤ Breadboard Jumper Cable            | 65 |
| ➤ Male to Female DuPont lines        | 10 |
| ➤ 9 volt battery snap                | 1  |
| ➤ USB Cable B-Type                   | 1  |
| ➤ HC-SR04 Ultrasonic Sensor          | 1  |
| ➤ Plastic Box – Bi level & Partition | 1  |

## 72.2) 최강 키트 부품 보관 상자

Plastic Box – Bi level & Partition. 키트 구성 항목을 보관 할 수 있는 플라스틱 상자입니다. 내부는 2 단으로 구성되어 있습니다. 파티션 가능 구조물이 있어 분류 별로 보관 할 수 있습니다.



그림 72-1 보관 상자



그림 72-2 보관 상자 내부 참조

## 73 최강 키트 플러스 구성 항목입니다.

이더넷 네트워크 & 블루투스 통신 기능이 추가됩니다.

| 항목                                     | 개수 |
|--|----|
| ➤ Arduino Ethernet Shield W5100        | 1  |
| ➤ Arduino Uno Prototype Plate Shield   | 1  |
| ➤ Arduino Uno Sensor Shield V4         | 1  |
| ➤ Bluetooth HC-06 4 pin Slave          | 1  |
| ➤ Breadboard Power Board 2 Load 3V3,5V | 1  |
| ➤ Breadboard Standard 400 points       | 1  |
| ➤ Breadboard Mini 170 points           | 2  |
| ➤ DuPont Male to Female Cable 40 pin   | 1  |
| ➤ DuPont Female to Female Cable 40 pin | 1  |
| ➤ Plastic Kit Large Box                | 1  |

## 74 최강 키트 얼티밋 플러스 구성 항목입니다.

XBee (지그비) 통신 테스트를 위해 2 개 모듈을 기반으로 합니다.

XBee 통신에 필요한 XBee 모듈 포함 관련 부품들과 테스트하기 위한 여러 종류의 MCU 보드가 포함된 형태입니다.

사용되는 기본 통신 모듈은 “XBee S2” 사용합니다.

MCU 보드를 사용한 원격지 통신 구성 항목입니다.

| 항목                             | 개수 |
|--------------------------------|----|
| ➤ XBee S2 Module               | 2  |
| ➤ XBee USB Adaptor & USB Cable | 2  |
| ➤ XBee Shield V3               | 2  |
| ➤ 아두이노 우노 R3 & USB Cable       | 1  |
| ➤ 아두이노 메가 2560 & USB Cable     | 1  |
| ➤ 아두이노 메가 2560 프로토타입 실드        | 1  |
| ➤ 아두이노 나노 V3 & USB Cable       | 1  |
| ➤ 아두이노 나노 V3 확장 실드             | 1  |
| ➤ 자이로 3 축 센서 GY-50 모듈          | 1  |
| ➤ DC 9V 1A 어댑터                 | 2  |

# 감사합니다

<http://www.gameplusedu.com>