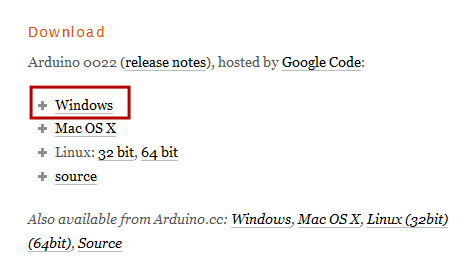
## [아두이노 가이드. 설치하고 프로그래밍 해보기 RT1](http://robobob.co.kr/51)

로보밥 아두이노 튜토리얼(Robobob Arduino Tutorial) 첫번째 이야기  
  
본글과 동일한 내용의 좀더 정리된 문서를 참고하시기 바랍니다. [[아트로봇 관련 튜토리얼 링크 바로가기](http://artrobot.co.kr/front/php/b/board_read_new.php?board_no=7&no=89&number=13&offset=0&page=1&search_key=&search=)]  
  
**RT1. 아두이노 설치하고 LED Blink 깜빡이 프로그래밍 하기**  
  
다루는 내용  
. 아두이노 프로그래밍 환경(스케치) 설치하기  
. PC에 아두이노 보드 인식시키기  
. 프로그래밍한 뒤 아두이노에 전송하여 작동시키기  
  
개요:   
본글은 아두이노를 처음 사용하는분들을 위한 안내문입니다.  
모든 아두이노 보드의 설치작업은 유사하며, 사용되는 USB시리얼 통신용 칩에 따라 드라이버 파일이나 inf설정 파일만 다릅니다. 본 글을 통해 아두이노 프로그래밍 환경인 스케치(Sketch)를 설치하고, 아두이노 보드를 PC에 인식시킨 후 아두이노에 기본 장착된 LED를 제어하는 프로그램을 전송하여 실행시키는 과정까지 다루고 있습니다.  
아마도, 마이크로콘트롤러와 프로그래밍에 대해 아무것도 모르시던 분들도 1시간내에 아두이노에 장착된 LED를 제어해보고 그 가능성을 경험해보실 수 있을실 겁니다.   
  
순서  
.아두이노 스케치를 다운로드 받아서 설치하기  
.아두이노를 PC에 인식시키기  
.스케치(아두이노 개발환경)의 실행  
.예제 소스 불러오기(Blink; LED깜빡이)  
.예제 컴파일 및 아두이노에 전송  
.작동 상태 확인  
  
자, 그럼 차근 차근 하나 둘 순서대로 진행해 보겠습니다  
  
  
**1단계. 준비물 확인**  
  
**준비물**  
  
**CASE 1 ::**아두이노 보드 자체에 USB시리얼 변환기능이 포함된 제품의 경우  
 .아두이노 UNO[[제품링크](http://robobob.cafe24.com/front/php/product.php?product_no=159&main_cate_no=7&display_group=1)], Mega2560 [[제품링크](http://robobob.cafe24.com/front/php/product.php?product_no=160&main_cate_no=7&display_group=1)]  
 .USB 케이블 (A to B 타입단자)  [[제품링크](http://robobob.cafe24.com/front/php/product.php?product_no=165&main_cate_no=7&display_group=1)]  
  
**CASE 2 ::**아두이노 FIO, Pro, Pro mini, LilyPad  등(내장 USB시리얼변환기가 없는 경우)  
 .아두이노 보드 [[아두이노 리스트](http://robobob.cafe24.com/front/php/category.php?cate_no=7)]  
 .FTDI USB시리얼 변환기 [[제품선택 가이드](http://robobob.cafe24.com/front/php/b/board_read_new.php?board_no=7&no=33&number=6&offset=0&page=1&search_key=&search=)]  
 .A to mini-B 타입 USB케이블 [[제품링크](http://robobob.cafe24.com/front/php/product.php?product_no=236&main_cate_no=7&display_group=1)]  
  
아두이노는 Linux , Mac OS, Windows 모두를 지원하며 본 글은 Windows 환경의 경우를 기본으로 소개합니다.  
  
  
**2단계. 아두이노 개발환경 설치하기**  
  
그림을 그리듯 프로그래밍도 Sketch 하세요!  
  
아두이노 공식 홈페이지인 [Arduino.cc](http://arduino.cc/)에 가보시면, 아두이노(Arduino) 소개문이 있습니다.   
이를 한 줄로 요약하면, 아두이노란 오픈소스 전자기기 프로토타입 플래폼이라는 얘기인데요, 간단히 말하자면, 전자기기 개발에 사용하는 소프트웨어와 하드웨어가 공개되어 있다는 겁니다.   
  
바로 지금 설치하려는 Sketch(스케치) 프로그램이 아두이노 개발에 사용되는 공개형 개발환경입니다.  
무료일 뿐만 아니라 소스코드도 공개되어 있습니다. 아래의 링크에서 최신버전을 다운로드하시기 바랍니다.

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



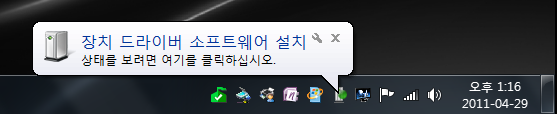
사용하시는 OS용 파일을 선택하여 다운로드 합니다. 본 예제에서는 Windows 를 선택합니다.

다운로드 받은 파일은 zip압축파일이며 이를 원하는 위치에 압축 해제합니다.  
(\*가령 윈도우의 경우 파일명은 arduino-0022.zip(87MB) 이며 버전이 업그레이드되면 제목의 숫자가 증가됩니다.)  
  
압축을 해제하면 arduino-0022 같은 폴더가 보이며 그안에 arduino.exe 실행 파일이 보입니다.  
아두이노 개발환경(스케치)은 설치과정 없이 곧바로 실행할 수 있게 배포되므로 이것으로 설치과정이 완료되었습니다. ^^.  
arduino.exe 를 실행하면 아두이노 개발환경이 열립니다. 일단 종료해 둡니다.  
  
**3단계. PC에 아두이노 연결하기**  
  
UNO와 대다수의 아두이노 보드들은 PC와 연결시 별도의 전원이 없이 작동이 가능합니다.  
즉, USB케이블을 통해 시리얼통신 신호를 주고 받는것과 더불어 전원도 공급받게 됩니다.  
  
호환되는 USB케이블로 PC와 아두이노를 연결합니다.  
FIO나 Pro mini같은 아두이노의 경우 FTDI USB시리얼 변환보드를 경유해 PC와 연결합니다.  
전원 ON상태 표시등(초록색 LED)에 불이 들어옵니다.  
  
  
**4단계. 드라이버 설치하기**  
  
아두이노(가령 UNO)와 PC를 케이블로 연결하면, 잠시 후 새장치를 발견하여 드라이버를 설치한다는 메시지가 나오게됩니다.  
그리고 몇 초 동안 짱구를 돌리며 열심히 노력하던 우리의 윈도우OS는 결국 자동인식에 실패했다는 메시지를 남기곤 숨어버립니다. (단, 윈도우 환경과 보드 종류에 따라 드라이버를 자동인식하여 설치하는 경우도 있습니다.)  
  
결국 여러분이 직접 드라이버를 골라서 설치해 주셔야하는데요,  많은 분들이 이미 이 작업에 익숙해져 있으실 겁니다. 가령 아래의 순서대로 하시면 됩니다.  (다른방법을 사용하셔도 되며, 드라이버 위치만 참고하시면 됩니다.)  
  
.윈도우 시작 > 제어판 > 장치 관리자("장치 및 프린터" 그룹)  를 엽니다.  
.장치리스트에서 "포트"를 선택하면, "Arduino UNO (COMxx)"라는 장치명이 보입니다.  
.해당장치를 우측버튼으로 클릭한 뒤 "드라이버 업데이트"를 선택합니다.  
  
.직접 드라이버 검색위치지정을 선택합니다.  
**UNO와 Mega2560 의 경우:**  
  > 2단계에서 다운로드 후 압축해제한 arduino-00xx 폴더내에 있는 drivers 폴더를 선택  
**Pro, Pro mini, FIO, LilyPad등의 FTDI시리얼 보드 사용제품의 경우**:   
  > drivers폴더안에 있는 FTDI USB Drivers 폴더를 선택  
  
.위도우가 UNO 장치 인식을 완료하게됩니다.  
  
장치 설치가 완료된 이후엔 장치관리자 "포트" 장치 리스트에서 인식된 아두이노 보드의 COM번호를 알아두는게 중요합니다.  
  
 **4단계 순서대로 다시 보기**(드라이버 설치 과정 캡쳐이미지, WINDOWS 7 기준)

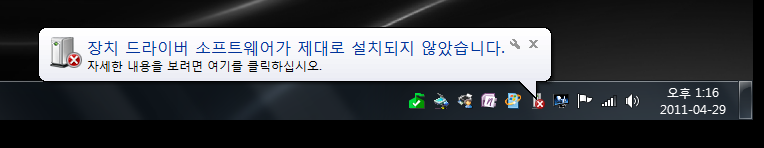
드라이버 설치과정을 아래의 캡쳐된 이미지 순서대로 다시한번 살펴봅니다.

아두이노와 PC를 연결합니다.

가령, UNO + USB Cable + PC USB 포트  
가령, Pro mini + FTDI USB 시리얼 변환기 + USB mini Cable + PC USB 포트



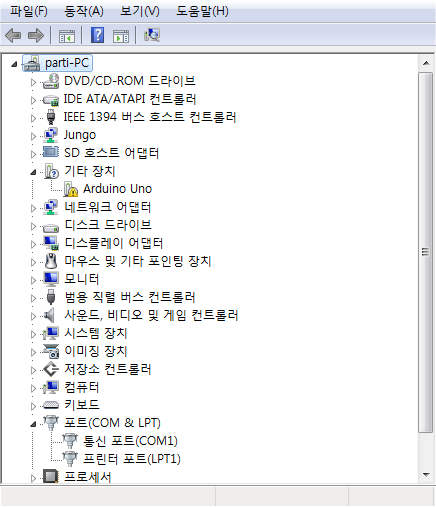
윈도우가 장치를 발견하고 드라이버 자동설치를 시도합니다.



드라이버 자동설치에 실패합니다.

(아두이노 종류와 OS에 따라 자동설치 되는 경우도 있습니다.)

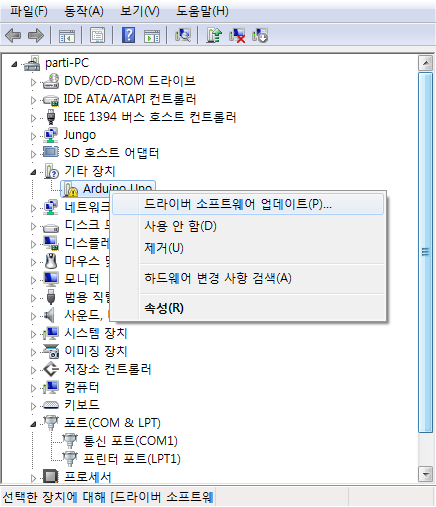
참고로, UNO와 Mega2560의 경우 MAC OS와 Linux에서 자동 인식된다고 합니다.



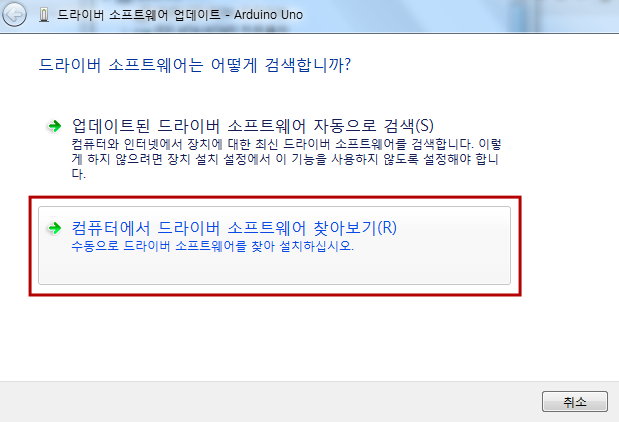
정상적으로 장치설치가 완료되면 포트(COM & LPT) 리스트에 등록되게 되지만,

정상인식이 되지 않아 장치관리자 "기타장치"에 Arduino UNO란 이름으로 등록되어있습니다.

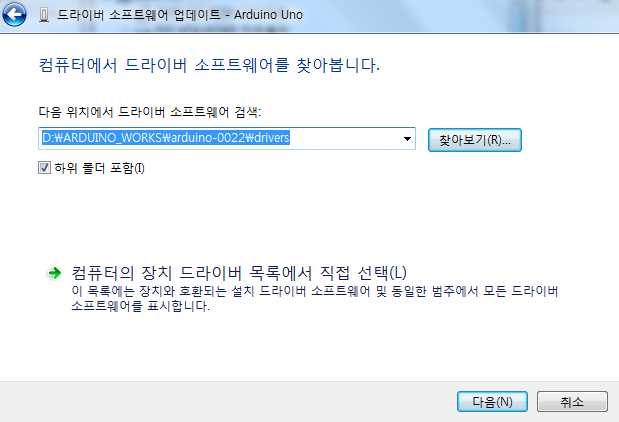
**다음의 절차를 통해 장치를 정상 인식시킵니다.**



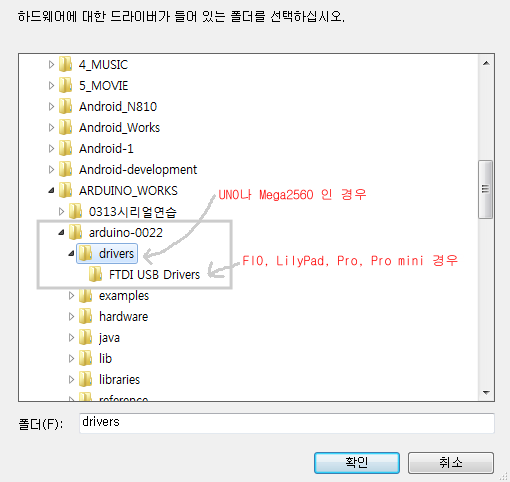
장치인식을위해 해당 장치(가령 Arduino UNO)를 우측버튼으로 클릭 후, 드라이버 소프트웨어 업데이트를 선택합니다.



자동검색을 하지말고,  수동으로 컴퓨터에있는 드라이버 찾아보기를 선택합니다.



찾아보기 버튼을 누르고, 해당 장치드라이버가 있는 폴더를 찾아 지정해줍니다.

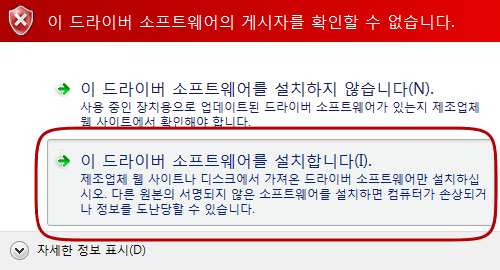


UNO와 Mega2560의 경우, 2단계에서 설치한 아두이노 프로그램 폴더(arduino-00xx)내에 있는 drivers 폴더를 선택해줍니다.

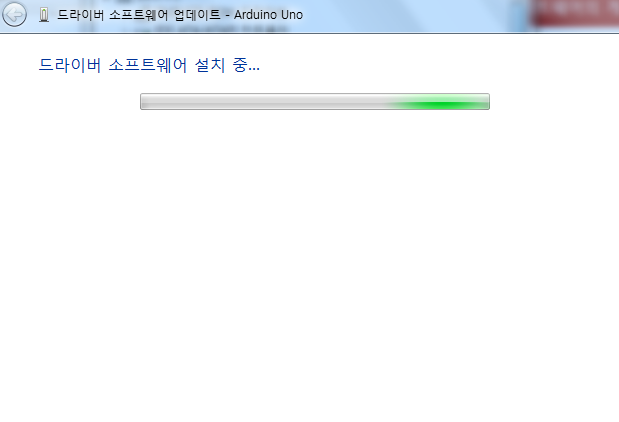
FIO, LilyPad, Pro, Pro mini등은 drivers폴더안에 있는 FTDI USB Drivers 폴더를 선택해줍니다.

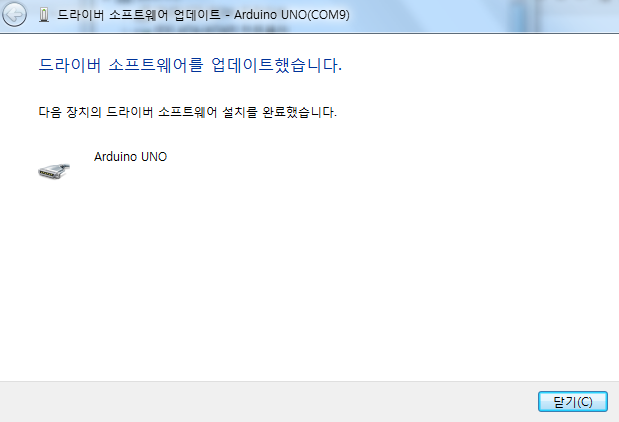
\*참고사항: UNO와 Mega2560은 dirvers폴더에 들어있는 inf(설정)파일 한개만 있으면 됩니다.

 기타 구형 아두이노들은 FTDI칩을 사용하므로 FTDI칩 인식용 장치드라이버 파일들이 필요합니다.

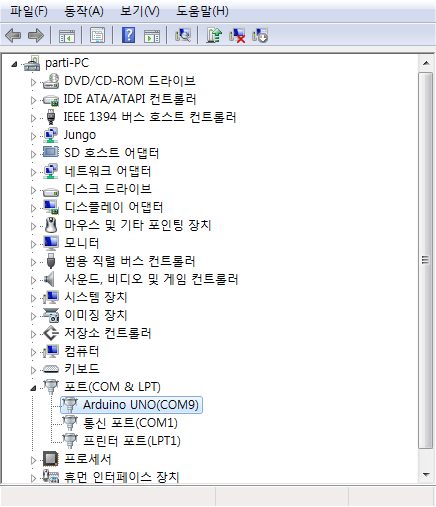


보안경고가 나오면 설치 허용을 선택합니다.





장치 인식이 완료되었습니다.

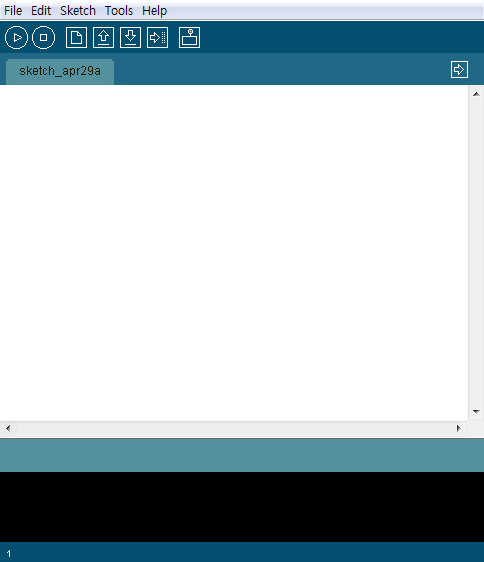


장치관리자 > 포트 정보를 보면  Arduino UNO(COMxx)와 같이 새로운 COM포트로 등록된 것을 확인할 수 있습니다.

컴퓨터 환경에 따라 COM번호는 다른 번호로 할당될 수 있습니다.

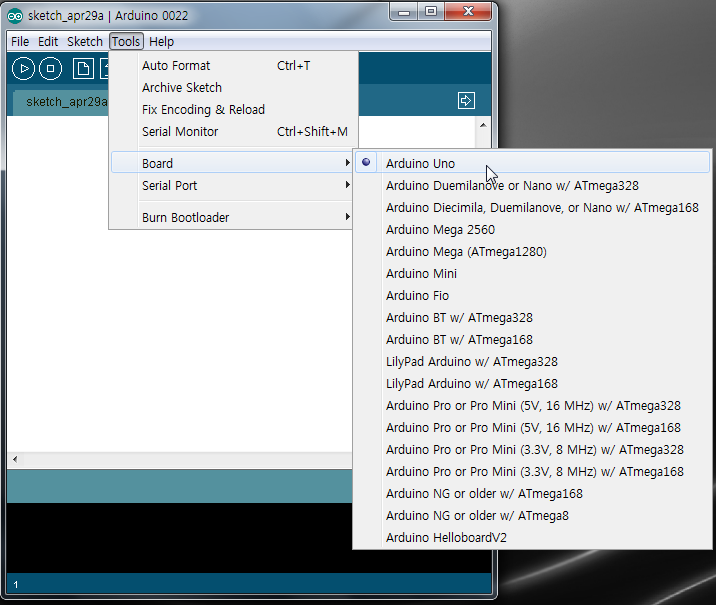
위 과정은, Windows OS 버젼별로 약간의 차이가 있지만 거의 비슷합니다.

자, 이제 아두이노 프로그램 설치와 장치인식이 모두 완료 되었습니다.   
이제 본격적으로 프로그래밍을 해보고 아두이노에 전송하여 작동시켜 보도록 합시다!  
  
  
**5단계. 아두이노 개발환경(스케치)을 실행합니다.**  
  
2단계에서 설치된 arduino.exe 를 실행합니다.  
앞으로 자주 실행을 해야 하므로 단축아이콘을 만들어두면 편리합니다.

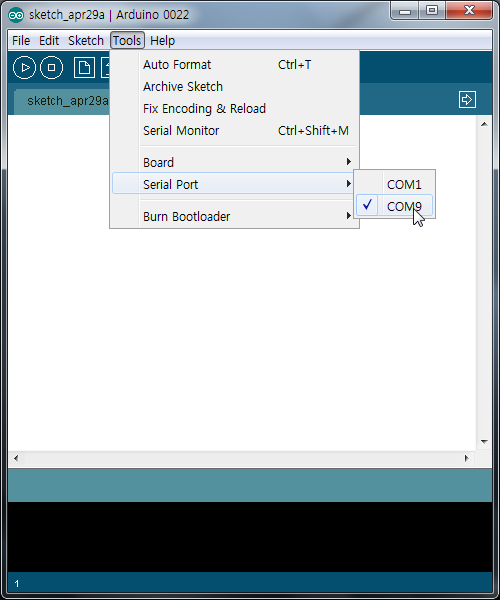


심플 담백한 스케치화면

**6단계. 아두이노 보드종류 선택하기**  
  
'단순 무식한 컴파일러에게 아두이노의 종류를 알려주세요!'  
  
아두이노 보드들이 많은 부분에서 호환성을 갖고 있지만, 구동속도( 8MHz, 16MHz),  전압레벨( 3.3V , 5V), 포트의 수, 프로그래밍 용량등의 차이가 있으므로 개발을 할때 이를 고려해 줘야 합니다. 스케치 개발환경에서 여러분이 사용하는 아두이노의 종류가 무엇인지 자동인식하지 못하므로 직접 보드 종류를 지정해 주는 과정을 꼭 하셔야합니다.  
  
스케치 메뉴에서 **Tools > Board**를 선택한 뒤 목록에서 자신의 보드명을 찾아서 선택해 줍니다.( 가령, Arduino UNO)



**7단계. 시리얼 포트 선택하기**  
  
'스케치에게 아두이노와 통신할 COM번호를 알려줍니다'  
  
PC와 아두이노간의 프로그램 전송 및 데이타통신을 위해서는, 4단계에서 아두이노 통신용으로 등록된 COM번호(위 경우 COM9번)를 지정해줘야합니다.   
  
스케치 메뉴에서**Tools > Serial Port**를 선택한 뒤 연결된 아두이노 포트번호를 지정합니다.



연결된 시리얼 장치가 여러개일때 아두이노의 COM번호 식별이 안될경우, 아두이노를 케이블에서 제거할때 목록에서 사라지는 COM번호가 아두이노 할당 COM번호입니다. (4단계에서와 같이 장치관리자에서 포트 리스트를 확인해도 됩니다.)  
  
  
  
  
**8단계. Blink 예제 소스코드 불러오기**  
  
스케치 메뉴에서 File > Examples > 1.Basics > Blink를 선택하여 불러옵니다.  
새로운 창이 뜨면서 아래와 같은 간단한 예제소스코드가 불러들여지게 됩니다.  
  
아래의 소스는 아두이노 13번핀을 1초마다 ON, OFF를 반복하게 합니다.  
  
/\*  
  Blink  
  Turns on an LED on for one second, then off for one second, repeatedly.  
   
  This example code is in the public domain.  
 \*/  
  
void setup() {                  
  // initialize the digital pin as an output.  
  // Pin 13 has an LED connected on most Arduino boards:  
  pinMode(13, OUTPUT);       
}  
  
void loop() {  
  digitalWrite(13, HIGH);   // set the LED on  
  delay(1000);              // wait for a second  
  digitalWrite(13, LOW);    // set the LED off  
  delay(1000);              // wait for a second  
}  
  
  
**9단계. 컴파일 및 아두이노에 전송하기**

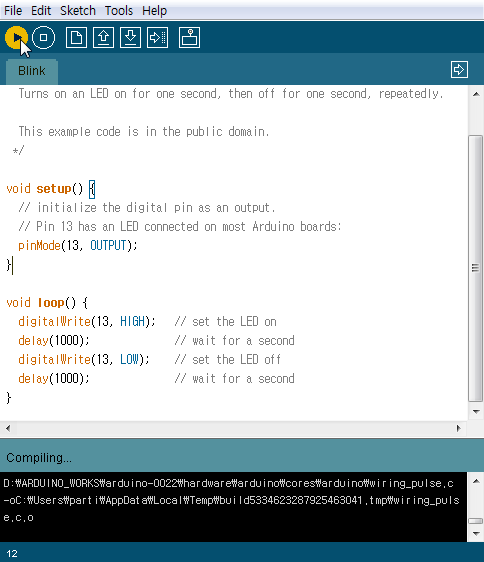
**verify & compile 하기** (생략 가능)

소스코드를 검증하고 컴파일하기 위해  재생버튼같이 생긴 verify & compile 버튼을 눌러줍니다.

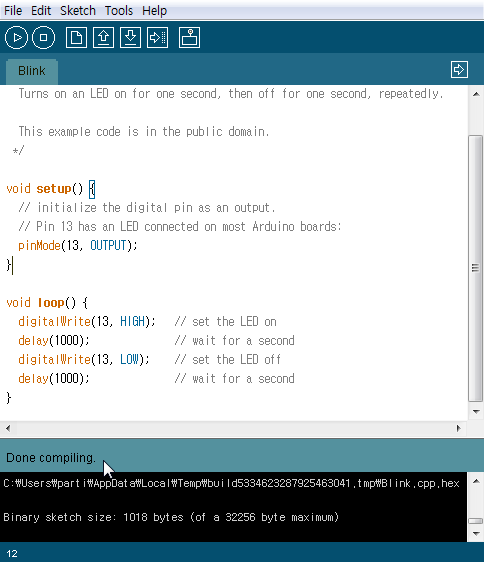
http://cfile1.uf.tistory.com/image/167CB44A4DBACBA3024BDD

Verify/Compile

참고로, 컴파일이란 사람이 이해가능한 소스코드를 기계어로 번역하는 과정입니다.  
컴파일을 하기전에는 아두이노의 종류에따라 일부 설정을 달리하여 컴파일하므로 자신이 사용중인 아두이노 보드의 종류를 잘 설정(6단계 참고)해줘야합니다. 보드 종류가 틀리면,  컴파일 결과를 전송하거나 전송 후 작동시 문제가 될 수 있습니다.  
앞으로 소스코드를 수정한 뒤 검증이 필요할때마다 이 버튼을 눌러주면 소스코드 검증이 이뤄지고 문제시 오류메시지를 확인할 수 있습니다.



컴파일 과정



컴파일이 완료되면 Done compiling 메시지가 뜨고 하단 메시지창에 프로그램 용량이 표시됩니다.

UNO가 약 30KB 프로그램 용량을 지원하므로 1/30 정도 크기입니다.

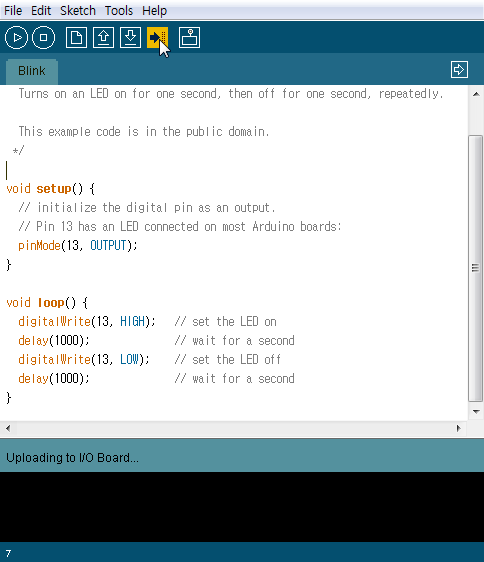
**upload 하기**

이제 컴파일된 정보를 아두이노로 전송하기위해 upoad 버튼을 눌러줍니다.  ctrl-U 단축키를 눌러도 됩니다.

http://cfile8.uf.tistory.com/image/157CB44A4DBACBA301E6D7

Upload to I/O Board

참고로, verify & compile 버튼을 누르지 않고 곧바로 upload버튼을 눌러도 됩니다.  
(이경우, 자동으로 컴파일 과정이 수행된 후 업로드가 이뤄지게 됩니다. 그리고, 소스코드 수정 후 곧바로 upload 버튼을 눌러주는 것 보다는 verify버튼을 눌러서 코드검증을 한 뒤에 문제가 없는경우 upload하는 것을 추천드립니다. )



**10단계. LED Blink - 발광다이오드의 깜빡임 확인하기**  
  
upload버튼을 누르면 아두이노 시리얼통신 관련 Tx RX  LED들이 빠르게 점등되는것 을 볼 수 있습니다.  
아두이노와 PC가 서로 정보를 주고(Tx, Transmit) 받기(Rx, Receive) 하면서 컴파일된 정보를 전달하는 과정입니다.   
이과정은 UNO의 경우 5초도 안걸립니다.  
  
소스코드에서 정의한대로 아두이노 보드를 보면 LED 하나가 1초 주기로 점등하는 것을 확인 할 수 있습니다.  
UNO를 비롯한 최근의 모든 공식 아두이노 보드들은 디지탈 13번 핀에 LED와 저항이 달려있으므로 별도로 LED를 장착하지 않아도 간단한 LED 제어 테스트를 해볼 수 있습니다.

**http://arduino.cc/en/Tutorial/HomePage**

**Learning**   [Examples](http://arduino.cc/en/Tutorial/HomePage) | [Foundations](http://arduino.cc/en/Tutorial/Foundations) | [Hacking](http://arduino.cc/en/Hacking/HomePage) | [Links](http://arduino.cc/en/Tutorial/Links)

*Examples > Basics*

http://arduino.cc/en/Tutorial/DigitalReadSerial

**Digital Read Serial**

This example shows you how to monitor the state of a switch by establishing [serial communication](http://arduino.cc/en/Reference/Serial) between your Arduino and your computer over USB.

**Hardware Required**

* Arduino Board
* A momentary switch, button, or toggle switch
* 10k ohm resistor
* breadboard
* hook-up wire

**Circuit**

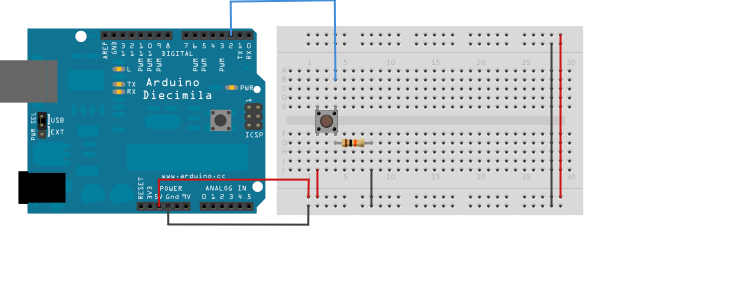


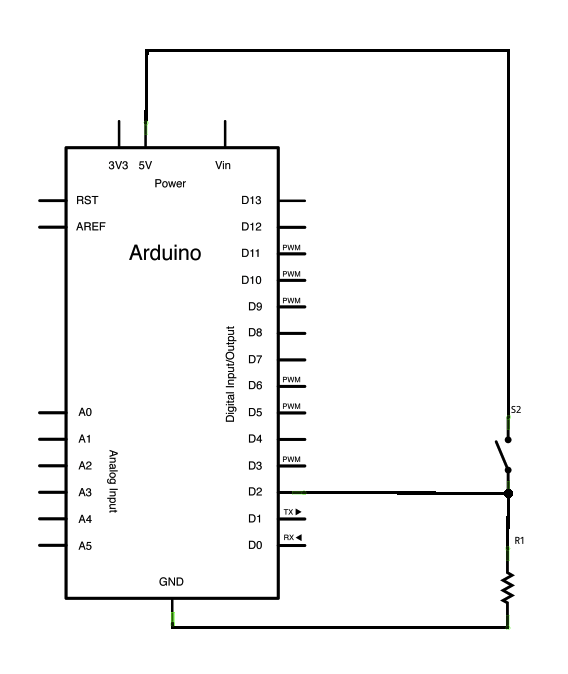
image developed using [Fritzing](http://www.fritzing.org/). For more circuit examples, see the [Fritzing project page](http://fritzing.org/projects/)

Connect three wires to the Arduino board. The first two, red and black, connect to the two long vertical rows on the side of the breadboard to provide access to the 5 volt supply and ground. The third wire goes from digital pin 2 to one leg of the pushbutton. That same leg of the button connects through a pull-down resistor (here 10 KOhms) to ground. The other leg of the button connects to the 5 volt supply.

Pushbuttons or switches connect two points in a circuit when you press them. When the pushbutton is open (unpressed) there is no connection between the two legs of the pushbutton, so the pin is connected to ground (through the pull-down resistor) and reads as LOW, or 0. When the button is closed (pressed), it makes a connection between its two legs, connecting the pin to 5 volts, so that the pin reads as HIGH, or 1.

If you disconnect the digital i/o pin from everything, the LED may blink erratically. This is because the input is "floating" - that is, it doesn't have a solid connection to voltage or ground, and it will randomly return either HIGH or LOW. That's why you need a pull-down resistor in the circuit.

**Schematic**

[](http://arduino.cc/en/uploads/Tutorial/button_sch.png)

**Code**

In the program below, the very first thing that you do will in the setup function is to begin serial communications, at 9600 bits of data per second, between your Arduino and your computer with the line:

Serial.begin(9600);

Next, initialize digital pin 2, the pin that will read the output from your button, as an input:

pinMode(2,INPUT);

Now that your setup has been completed, move into the main loop of your code. When your button is pressed, 5 volts will freely flow through your circuit, and when it is not pressed, the input pin will be connected to ground through the 10-kilohm resistor. This is a digital input, meaning that the switch can only be in either an on state (seen by your Arduino as a "1", or HIGH) or an off state (seen by your Arduino as a "0", or LOW), with nothing in between.

The first thing you need to do in the main loop of your program is to establish a variable to hold the information coming in from your switch. Since the information coming in from the switch will be either a "1" or a "0", you can use an[int datatype](http://arduino.cc/en/Reference/Int). Call this variable sensorValue, and set it to equal whatever is being read on digital pin 2. You can accomplish all this with just one line of code:

int sensorValue = digitalRead(2);

Once the Arduino has read the input, make it print this information back to the computer as a decimal value. You can do this with the command [Serial.println](http://arduino.cc/en/Serial/Println)() in our last line of code:

Serial.println(sensorValue);

Now, when you open your Serial Monitor in the Arduino environment, you will see a stream of "0"s if your switch is open, or "1"s if your switch is closed.

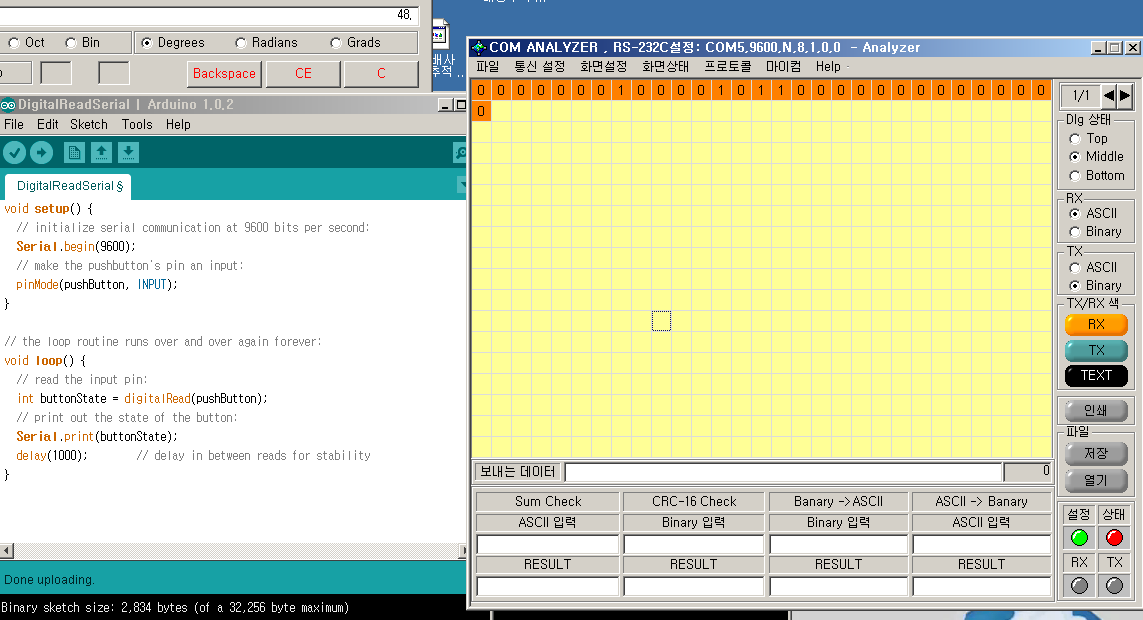
*/\*  
  DigitalReadSerial  
 Reads a digital input on pin 2, prints the result to the serial monitor   
   
 This example code is in the public domain.  
 \*/*  
  
*// digital pin 2 has a pushbutton attached to it. Give it a name:*  
int pushButton = 2;  
  
*// the setup routine runs once when you press reset:*  
void **setup**() {  
  *// initialize serial communication at 9600 bits per second:*  
  Serial.begin(9600);  
  *// make the pushbutton's pin an input:*  
  pinMode(pushButton, INPUT);  
}  
  
*// the loop routine runs over and over again forever:*  
void **loop**() {  
  *// read the input pin:*  
  int buttonState = digitalRead(pushButton);  
  *// print out the state of the button:*  
  Serial.println(buttonState);  
  delay(1);        *// delay in between reads for stability*  
}

[[Get Code]](http://arduino.cc/en/Tutorial/DigitalReadSerial?action=sourceblock&num=1)

**See Also:**

* [setup](http://arduino.cc/en/Reference/Setup)()
* [loop](http://arduino.cc/en/Reference/Loop)()
* [pinMode](http://arduino.cc/en/Reference/PinMode)()
* [digitalRead](http://arduino.cc/en/Reference/DigitalRead)()
* [delay](http://arduino.cc/en/Reference/Delay)()
* [int](http://arduino.cc/en/Reference/Int)
* [serial](http://arduino.cc/en/Reference/Serial)
* [DigitalPins](http://arduino.cc/en/Tutorial/DigitalPins)
* [BareMinimum](http://arduino.cc/en/Tutorial/BareMinimum): The bare minimum of code needed to start an Arduino sketch.
* [Blink](http://arduino.cc/en/Tutorial/Blink): Turn an LED on and off.
* [DigitalReadSerial](http://arduino.cc/en/Tutorial/DigitalReadSerial): Read a switch, print the state out to the Arduino Serial Monitor.
* [AnalogReadSerial](http://arduino.cc/en/Tutorial/AnalogReadSerial): Read a potentiometer, print it's state out to the Arduino Serial Monitor.
* [Fade](http://arduino.cc/en/Tutorial/Fade): Demonstrates the use of analog output to fade an LED.
* [ReadAnalogVoltage](http://arduino.cc/en/Tutorial/ReadAnalogVoltage) : Reads an analog input and prints the voltage to the serial monitor

[More Sharing ServicesShare](http://www.addthis.com/bookmark.php?v=250&username=arduinoteam)|[Share on emailShare on favoritesShare on printShare on facebookShare on twitter](http://arduino.cc/en/Tutorial/DigitalReadSerial)



아스키값으로 넘어옴-\_- 인트형이긴한데

**받아서 시리얼쏘고 LED로 본다.**

// digital pin 2 has a pushbutton attached to it. Give it a name:

int pushButton = 2;

int led=13;

// the setup routine runs once when you press reset:

void setup() {

// initialize serial communication at 9600 bits per second:

Serial.begin(9600);

// make the pushbutton's pin an input:

pinMode(pushButton, INPUT);

**pinMode(led, OUTPUT);**

}

// the loop routine runs over and over again forever:

void loop() {

// read the input pin:

int buttonState = digitalRead(pushButton);

// print out the state of the button:

Serial.println(buttonState);

if(buttonState==1){

**digitalWrite(led, HIGH);**

Serial.println("133311");

}else{

**digitalWrite(led, LOW);**

}

delay(10); // delay in between reads for stability

}

**페이드인 페이드아웃**

int led = 9; // the pin that the LED is attached to

int brightness = 0; // how bright the LED is

int fadeAmount = 5; // how many points to fade the LED by

// the setup routine runs once when you press reset:

void setup() {

// declare pin 9 to be an output:

**pinMode(led, OUTPUT);**

}

// the loop routine runs over and over again forever:

void loop() {

// set the brightness of pin 9:

**analogWrite(led, brightness);**

// change the brightness for next time through the loop:

brightness = brightness + fadeAmount;

// reverse the direction of the fading at the ends of the fade:

if (brightness == 0 || brightness == **255**) {

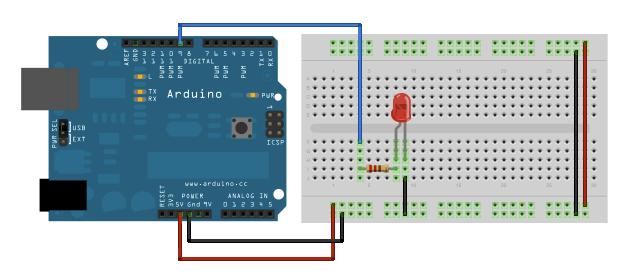
fadeAmount = -fadeAmount ;

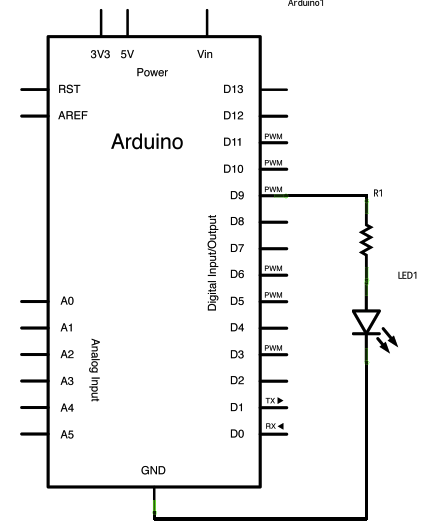
}

// wait for 30 milliseconds to see the dimming effect

delay(30);

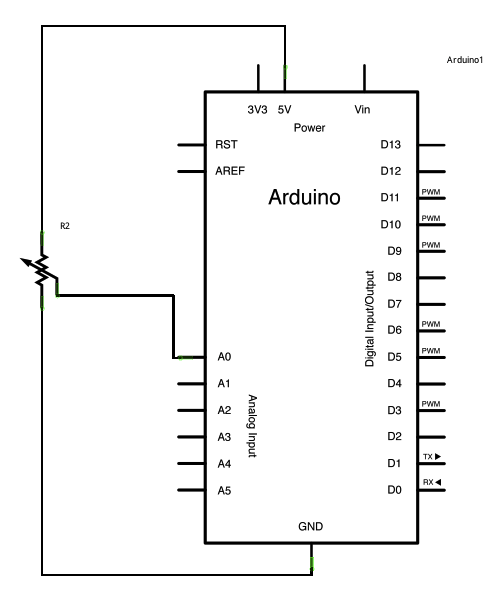
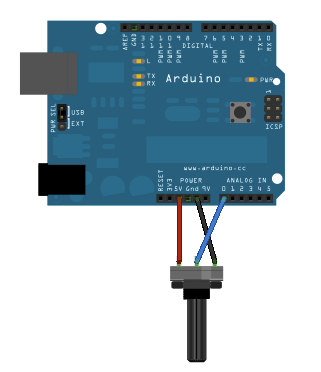
}





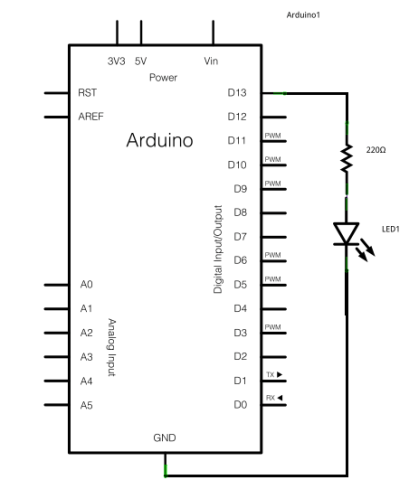
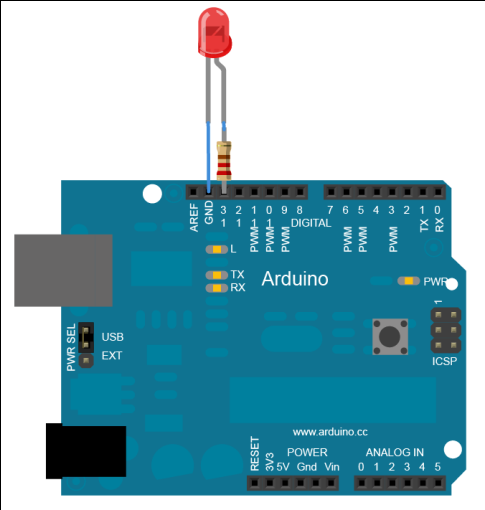
아날로그 입력은 255까지인듯.

**아날로그 입력값 받기**

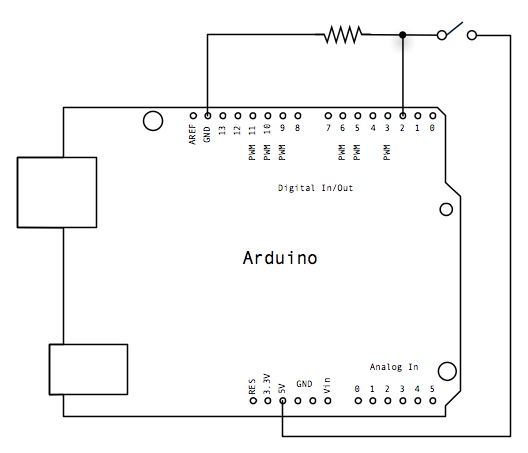
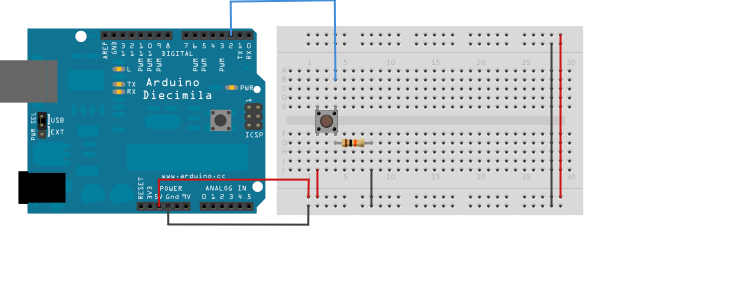
void **setup**() {  
  *// initialize serial communication at 9600 bits per second:*  
  Serial.begin(9600);  
}  
  
*// the loop routine runs over and over again forever:*  
void **loop**() {  
  *// read the input on analog pin 0:*  
  int sensorValue = analogRead(A0);  
  *// Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):*  
  float voltage = sensorValue \* (5.0 / 1023.0);  
  *// print out the value you read:*  
  Serial.println(voltage);  
}

**깜빡깜빡 Digital**

const int ledPin =  13;      *// the number of the LED pin*  
  
*// Variables will change:*  
int ledState = LOW;             *// ledState used to set the LED*  
long previousMillis = 0;        *// will store last time LED was updated*  
  
*// the follow variables is a long because the time, measured in miliseconds,*  
*// will quickly become a bigger number than can be stored in an int.*  
long interval = 1000;           *// interval at which to blink (milliseconds)*  
  
void **setup**() {  
  *// set the digital pin as output:*  
  pinMode(ledPin, OUTPUT);        
}  
  
void **loop**()  
{  
  *// here is where you'd put code that needs to be running all the time.*  
  
  *// check to see if it's time to blink the LED; that is, if the*  
  *// difference between the current time and last time you blinked*  
  *// the LED is bigger than the interval at which you want to*  
  *// blink the LED.*  
  unsigned long currentMillis = millis();  
   
  if(currentMillis - previousMillis > interval) {  
    *// save the last time you blinked the LED*  
    previousMillis = currentMillis;     
  
    *// if the LED is off turn it on and vice-versa:*  
    if (ledState == LOW)  
      ledState = HIGH;  
    else  
      ledState = LOW;  
  
    *// set the LED with the ledState of the variable:*  
    digitalWrite(ledPin, ledState);  
  }  
}



**버튼누르면 LED켜짐**

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);   
  }  
}

**노래 딴다다단 따다!**

int melody[] = {  
  NOTE\_C4, NOTE\_G3,NOTE\_G3, NOTE\_A3, NOTE\_G3,0, NOTE\_B3, NOTE\_C4};  
  
*// note durations: 4 = quarter note, 8 = eighth note, etc.:*  
int noteDurations[] = {  
  4, 8, 8, 4,4,4,4,4 };  
  
void **setup**() {  
  *// iterate over the notes of the melody:*  
  for (int thisNote = 0; thisNote < 8; thisNote++) {  
  
    *// to calculate the note duration, take one second*  
    *// divided by the note type.*  
    *//e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.*  
    int noteDuration = 1000/noteDurations[thisNote];  
    tone(8, melody[thisNote],noteDuration);  
  
    *// to distinguish the notes, set a minimum time between them.*  
    *// the note's duration + 30% seems to work well:*  
    int pauseBetweenNotes = noteDuration \* 1.30;  
    delay(pauseBetweenNotes);  
    *// stop the tone playing:*  
    noTone(8);  
  }  
}  
  
void **loop**() {  
  *// no need to repeat the melody.*  
}

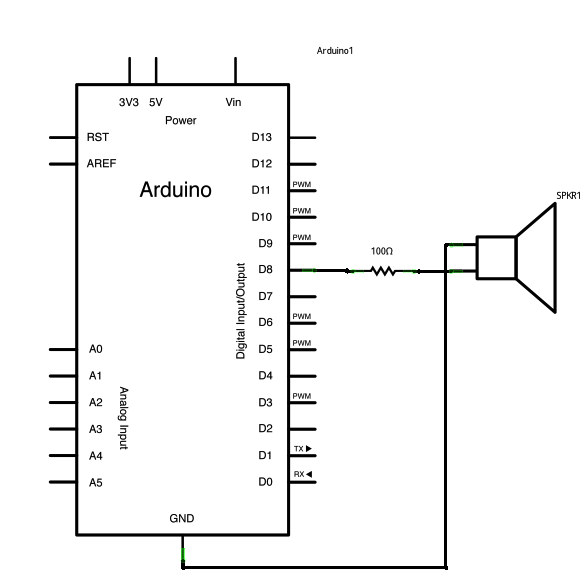
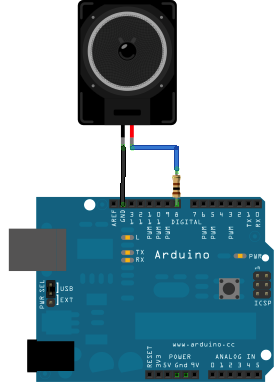
[[Get Code]](http://arduino.cc/en/Tutorial/Tone?action=sourceblock&num=1)

To make the pitches.h file, click on the "new Tab" button in the upper right hand corner of the window. It looks like this:

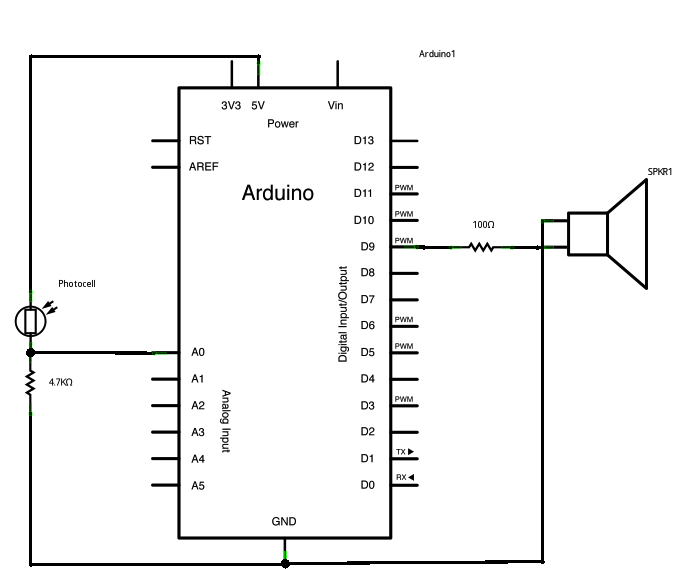
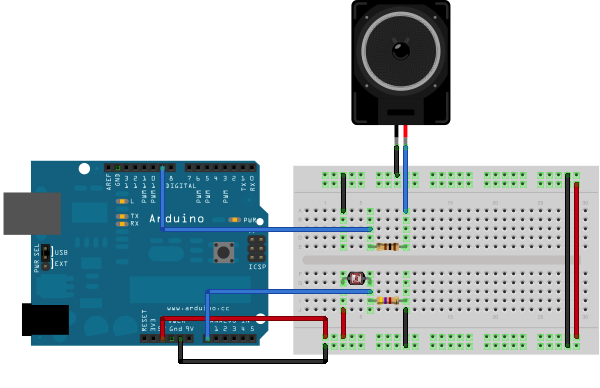
http://arduino.cc/en/uploads/Tutorial/newTab.png

The paste in the following code:

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* Public Constants  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  
  
#define NOTE\_B0  31  
#define NOTE\_C1  33  
#define NOTE\_CS1 35  
#define NOTE\_D1  37  
#define NOTE\_DS1 39  
#define NOTE\_E1  41  
#define NOTE\_F1  44  
#define NOTE\_FS1 46  
#define NOTE\_G1  49  
#define NOTE\_GS1 52  
#define NOTE\_A1  55  
#define NOTE\_AS1 58  
#define NOTE\_B1  62  
#define NOTE\_C2  65  
#define NOTE\_CS2 69  
#define NOTE\_D2  73  
#define NOTE\_DS2 78  
#define NOTE\_E2  82  
#define NOTE\_F2  87  
#define NOTE\_FS2 93  
#define NOTE\_G2  98  
#define NOTE\_GS2 104  
#define NOTE\_A2  110  
#define NOTE\_AS2 117  
#define NOTE\_B2  123  
#define NOTE\_C3  131  
#define NOTE\_CS3 139  
#define NOTE\_D3  147  
#define NOTE\_DS3 156  
#define NOTE\_E3  165  
#define NOTE\_F3  175  
#define NOTE\_FS3 185  
#define NOTE\_G3  196  
#define NOTE\_GS3 208  
#define NOTE\_A3  220  
#define NOTE\_AS3 233  
#define NOTE\_B3  247  
#define NOTE\_C4  262  
#define NOTE\_CS4 277  
#define NOTE\_D4  294  
#define NOTE\_DS4 311  
#define NOTE\_E4  330  
#define NOTE\_F4  349  
#define NOTE\_FS4 370  
#define NOTE\_G4  392  
#define NOTE\_GS4 415  
#define NOTE\_A4  440  
#define NOTE\_AS4 466  
#define NOTE\_B4  494  
#define NOTE\_C5  523  
#define NOTE\_CS5 554  
#define NOTE\_D5  587  
#define NOTE\_DS5 622  
#define NOTE\_E5  659  
#define NOTE\_F5  698  
#define NOTE\_FS5 740  
#define NOTE\_G5  784  
#define NOTE\_GS5 831  
#define NOTE\_A5  880  
#define NOTE\_AS5 932  
#define NOTE\_B5  988  
#define NOTE\_C6  1047  
#define NOTE\_CS6 1109  
#define NOTE\_D6  1175  
#define NOTE\_DS6 1245  
#define NOTE\_E6  1319  
#define NOTE\_F6  1397  
#define NOTE\_FS6 1480  
#define NOTE\_G6  1568  
#define NOTE\_GS6 1661  
#define NOTE\_A6  1760  
#define NOTE\_AS6 1865  
#define NOTE\_B6  1976  
#define NOTE\_C7  2093  
#define NOTE\_CS7 2217  
#define NOTE\_D7  2349  
#define NOTE\_DS7 2489  
#define NOTE\_E7  2637  
#define NOTE\_F7  2794  
#define NOTE\_FS7 2960  
#define NOTE\_G7  3136  
#define NOTE\_GS7 3322  
#define NOTE\_A7  3520  
#define NOTE\_AS7 3729  
#define NOTE\_B7  3951  
#define NOTE\_C8  4186  
#define NOTE\_CS8 4435  
#define NOTE\_D8  4699  
#define NOTE\_DS8 4978



**빛 저항으로인한 소리 변화**

void **setup**() {  
  *// initialize serial communications (for debugging only):*  
  Serial.begin(9600);  
}  
  
void **loop**() {  
  *// read the sensor:*  
  int sensorReading = analogRead(A0);  
  *// print the sensor reading so you know its range*  
  Serial.println(sensorReading);  
  *// map the analog input range (in this case, 400 - 1000 from the photoresistor)*  
  *// to the output pitch range (120 - 1500Hz)*  
  *// change the minimum and maximum input numbers below*  
  *// depending on the range your sensor's giving:*  
  int thisPitch = map(sensorReading, 400, 1000, 120, 1500);  
  
  *// play the pitch:*  
  tone(9, thisPitch, 10);  
  delay(1);        *// delay in between reads for stability*  
}

## Examples

See the[***foundations page***](http://arduino.cc/en/Tutorial/Foundations)for in-depth description of core concepts of the Arduino hardware and software; the[***hacking page***](http://arduino.cc/en/Hacking/HomePage)for information on extending and modifying the Arduino hardware and software; and the[***links page***](http://arduino.cc/en/Tutorial/Links)for other documentation.

**Note: these examples are written for Arduino 1.0 and later.** Certain functions may not work in earlier versions. For best results, [download the latest version](http://arduino.cc/en/Main/Software).

Here's a [style guide](http://arduino.cc/en/Reference/StyleGuide) that helps with writing examples for beginners.

|  |  |  |
| --- | --- | --- |
| Core Functions Simple programs that demonstrate basic Arduino commands. These are included with the Arduino environment; to open them, click the Open button on the toolbar and look in the **examples** folder. 1.Basics  * [BareMinimum](http://arduino.cc/en/Tutorial/BareMinimum): The bare minimum of code needed to start an Arduino sketch. * [Blink](http://arduino.cc/en/Tutorial/Blink): Turn an LED on and off. * [DigitalReadSerial](http://arduino.cc/en/Tutorial/DigitalReadSerial): Read a switch, print the state out to the Arduino Serial Monitor. * [AnalogReadSerial](http://arduino.cc/en/Tutorial/AnalogReadSerial): Read a potentiometer, print it's state out to the Arduino Serial Monitor. * [Fade](http://arduino.cc/en/Tutorial/Fade): Demonstrates the use of analog output to fade an LED. * [ReadAnalogVoltage](http://arduino.cc/en/Tutorial/ReadAnalogVoltage) : Reads an analog input and prints the voltage to the serial monitor  2.Digital  * [Blink Without Delay](http://arduino.cc/en/Tutorial/BlinkWithoutDelay): blinking an LED without using the delay() function. * [Button](http://arduino.cc/en/Tutorial/Button): use a pushbutton to control an LED. * [Debounce](http://arduino.cc/en/Tutorial/Debounce): read a pushbutton, filtering noise. * [Button State Change](http://arduino.cc/en/Tutorial/ButtonStateChange): counting the number of button pushes. * [Input Pullup Serial](http://arduino.cc/en/Tutorial/InputPullupSerial): Demonstrates the use of INPUT\_PULLUP with pinMode(). * [Tone](http://arduino.cc/en/Tutorial/Tone): play a melody with a Piezo speaker. * [Pitch follower](http://arduino.cc/en/Tutorial/Tone2): play a pitch on a piezo speaker depending on an analog input. * [Simple Keyboard](http://arduino.cc/en/Tutorial/Tone3): a three-key musical keyboard using force sensors and a piezo speaker. * [Tone4](http://arduino.cc/en/Tutorial/Tone4): play tones on multiple speakers sequentially using the tone() command.  3.Analog  * [AnalogInOutSerial](http://arduino.cc/en/Tutorial/AnalogInOutSerial): read an analog input pin, map the result, and then use that data to dim or brighten an LED. * [Analog Input](http://arduino.cc/en/Tutorial/AnalogInput): use a potentiometer to control the blinking of an LED. * [AnalogWriteMega](http://arduino.cc/en/Tutorial/AnalogWriteMega): fade 12 LEDs on and off, one by one, using an Arduino Mega board. * [Calibration](http://arduino.cc/en/Tutorial/Calibration): define a maximum and minimum for expected analog sensor values. * [Fading](http://arduino.cc/en/Tutorial/Fading): use an analog output (PWM pin) to fade an LED. * [Smoothing](http://arduino.cc/en/Tutorial/Smoothing): smooth multiple readings of an analog input.  4.Communication These examples include code that allows the Arduino to talk to Processing sketches running on the computer. For more information or to download Processing, see[*processing.org*](http://processing.org/). There are also Max/MSP patches that can communicate with each Arduino sketch as well. For more on Max/MSP see [Cycling 74](http://www.cycling74.com/). For Pd patches that can communicate with these sketches, see [Scott Fitzgerald's examples](http://www.ennuigo.com/?p=409).   * [ReadASCIIString](http://arduino.cc/en/Tutorial/ReadASCIIString): parse a comma-separated string of ints to fade an LED * [ASCII Table](http://arduino.cc/en/Tutorial/ASCIITable): demonstrates Arduino's advanced serial output functions. * [Dimmer](http://arduino.cc/en/Tutorial/Dimmer): move the mouse to change the brightness of an LED. * [Graph](http://arduino.cc/en/Tutorial/Graph): send data to the computer and graph it in Processing. * [Physical Pixel](http://arduino.cc/en/Tutorial/PhysicalPixel): turn a LED on and off by sending data to your Arduino from Processing or Max/MSP. * [Virtual Color Mixer](http://arduino.cc/en/Tutorial/VirtualColorMixer): send multiple variables from Arduino to your computer and read them in Processing or Max/MSP. * [Serial Call Response](http://arduino.cc/en/Tutorial/SerialCallResponse): send multiple vairables using a call-and-response (handshaking) method. * [Serial Call Response ASCII](http://arduino.cc/en/Tutorial/SerialCallResponseASCII): send multiple variables using a call-and-response (handshaking) method, and ASCII-encode the values before sending. * [SerialEvent](http://arduino.cc/en/Tutorial/SerialEvent): Demonstrates the use of SerialEvent(). * [Serial input (Switch (case) Statement)](http://arduino.cc/en/Tutorial/SwitchCase2): how to take different actions based on characters received by the serial port. * [MIDI](http://arduino.cc/en/Tutorial/Midi): send MIDI note messages serially. * [MultiSerialMega](http://arduino.cc/en/Tutorial/MultiSerialMega): use two of the serial ports available on the Arduino Mega.  5.Control Structures  * [If Statement](http://arduino.cc/en/Tutorial/IfStatement) (Conditional): how to use an if statement to change output conditions based on changing input conditions. * [For Loop](http://arduino.cc/en/Tutorial/ForLoop): controlling multiple LEDs with a for loop and. * [Array](http://arduino.cc/en/Tutorial/Array): a variation on the For Loop example that demonstrates how to use an array. * [While Loop](http://arduino.cc/en/Tutorial/WhileLoop): how to use a while loop to calibrate a sensor while a button is being read. * [Switch Case](http://arduino.cc/en/Tutorial/SwitchCase): how to choose between a discrete number of values. Equivalent to multiple If statements. This example shows how to divide a sensor's range into a set of four bands and to take four different actions depending on which band the result is in. * [Switch Case 2](http://arduino.cc/en/Tutorial/SwitchCase2): a second switch-case example, showing how to take different actions based in characters received in the serial port.  6.Sensors  * [ADXL3xx](http://arduino.cc/en/Tutorial/ADXL3xx): read an ADXL3xx accelerometer. * [Knock](http://arduino.cc/en/Tutorial/Knock): detect knocks with a piezo element. * [Memsic2125](http://arduino.cc/en/Tutorial/Memsic2125): two-axis acceleromoter. * [Ping](http://arduino.cc/en/Tutorial/Ping): detecting objects with an ultrasonic range finder.  7.Display Examples of basic display control   * [LED Bar Graph](http://arduino.cc/en/Tutorial/BarGraph): how to make an LED bar graph. * [Row Column Scanning](http://arduino.cc/en/Tutorial/RowColumnScanning): how to control an 8x8 matrix of LEDs.  8.Strings  * [StringAdditionOperator](http://arduino.cc/en/Tutorial/StringAdditionOperator): add strings together in a variety of ways. * [StringAppendOperator](http://arduino.cc/en/Tutorial/StringAppendOperator): append data to strings. * [StringCaseChanges](http://arduino.cc/en/Tutorial/StringCaseChanges): change the case of a string. * [StringCharacters](http://arduino.cc/en/Tutorial/StringCharacters): get/set the value of a specific character in a string. * [StringComparisonOperators](http://arduino.cc/en/Tutorial/StringComparisonOperators): compare strings alphabetically. * [StringConstructors](http://arduino.cc/en/Tutorial/StringConstructors): how to initialize string objects. * [StringIndexOf](http://arduino.cc/en/Tutorial/StringIndexOf): look for the first/last instance of a character in a string. * [StringLength & StringLengthTrim](http://arduino.cc/en/Tutorial/StringLengthTrim): get and trim the length of a string. * [StringReplace](http://arduino.cc/en/Tutorial/StringReplace): replace individual characters in a string. * [StringStartsWithEndsWith](http://arduino.cc/en/Tutorial/StringStartsWithEndsWith): check which characters/substrings a given string starts or ends with. * [StringSubstring](http://arduino.cc/en/Tutorial/StringSubstring): look for "phrases" within a given string.  9.USB (Leonardo, Micro, and Due specific examples) The Keyboard and Mouse examples are unique to the Leonardo, Micro and Due. They demonstrate the use of libraries that are unique to the board.   * [KeyboardAndMouseControl](http://arduino.cc/en/Tutorial/KeyboardAndMouseControl): Demonstrates the Mouse and Keyboard commands in one program.  Keyboard  * [KeyboardMessage](http://arduino.cc/en/Tutorial/KeyboardMessage): Sends a text string when a button is pressed. * [KeyboardLogout](http://arduino.cc/en/Tutorial/KeyboardLogout) : Logs out the current user with key commands * [KeyboardSerial](http://arduino.cc/en/Tutorial/KeyboardSerial): Reads a byte from the serial port, and sends back a keystroke. * [KeyboardReprogram](http://arduino.cc/en/Tutorial/KeyboardReprogram) : opens a new window in the Arduino IDE and reprograms the Leonardo with a simple blink program  Mouse  * [ButtonMouseControl](http://arduino.cc/en/Tutorial/ButtonMouseControl): Control cursor movement with 5 pushbuttons. * [JoystickMouseControl](http://arduino.cc/en/Tutorial/JoystickMouseControl): Controls a computer's cursor movement with a Joystick when a button is pressed. |  | [Libraries](http://arduino.cc/en/Reference/Libraries) Examples from the libraries that are included in the Arduino software. EEPROM Library  * [EEPROM Clear](http://arduino.cc/en/Tutorial/EEPROMClear): clear the bytes in the EEPROM. * [EEPROM Read](http://arduino.cc/en/Tutorial/EEPROMRead): read the EEPROM and send its values to the computer. * [EEPROM Write](http://arduino.cc/en/Tutorial/EEPROMWrite): stores values from an analog input to the EEPROM.  Ethernet Library  * [ChatServer](http://arduino.cc/en/Tutorial/ChatServer): set up a simple chat server. * [WebClient](http://arduino.cc/en/Tutorial/WebClient): make a HTTP request. * [WebClientRepeating](http://arduino.cc/en/Tutorial/WebClientRepeating): Make repeated HTTP requests. * [WebServer](http://arduino.cc/en/Tutorial/WebServer): host a simple HTML page that displays analog sensor values. * [PachubeClient](http://arduino.cc/en/Tutorial/PachubeClient): connect to pachube.com, a free datalogging site. * [PachubeClientString](http://arduino.cc/en/Tutorial/PachubeClientString): send strings to pachube.com. * [BarometricPressureWebServer](http://arduino.cc/en/Tutorial/BarometricPressureWebServer): outputs the values from a barometric pressure sensor as a web page. * [UDPSendReceiveString](http://arduino.cc/en/Tutorial/UDPSendReceiveString): Send and receive text strings via UDP. * [UdpNtpClient](http://arduino.cc/en/Tutorial/UdpNtpClient): Query a Network Time Protocol (NTP) server using UDP. * [DnsWebClient](http://arduino.cc/en/Tutorial/DnsWebClient): DNS and DHCP-based Web client. * [DhcpChatServer](http://arduino.cc/en/Tutorial/DhcpChatServer): A simple DHCP Chat Server * [DhcpAddressPrinter](http://arduino.cc/en/Tutorial/DhcpAddressPrinter): Get an IP address via DHCP and print it out * [TwitterClient](http://arduino.cc/en/Tutorial/TwitterClient): A Twitter client with Strings * [TelnetClient](http://arduino.cc/en/Tutorial/TelnetClient): A simple Telnet client  Firmata Libraries  * [Guide to the Standard Firmata Library](http://arduino.cc/en/Reference/Firmata)  LiquidCrystal Library  * [Hello World](http://arduino.cc/en/Tutorial/LiquidCrystal): displays "hello world!" and the seconds since reset. * [Blink](http://arduino.cc/en/Tutorial/LiquidCrystalBlink): control of the block-style cursor. * [Cursor](http://arduino.cc/en/Tutorial/LiquidCrystalCursor): control of the underscore-style cursor. * [Display](http://arduino.cc/en/Tutorial/LiquidCrystalDisplay): quickly blank the display without losing what's on it. * [TextDirection](http://arduino.cc/en/Tutorial/LiquidCrystalTextDirection): control which way text flows from the cursor. * [Scroll](http://arduino.cc/en/Tutorial/LiquidCrystalScroll): scroll text left and right. * [Serial input](http://arduino.cc/en/Tutorial/LiquidCrystalSerial): accepts serial input, displays it. * [SetCursor](http://arduino.cc/en/Tutorial/LiquidCrystalSetCursor): set the cursor position. * [Autoscroll](http://arduino.cc/en/Tutorial/LiquidCrystalAutoscroll): shift text right and left.  SPI Library  * [BarometricPressureSensor](http://arduino.cc/en/Tutorial/BarometricPressureSensor): read air pressure and temperature from a sensor using the SPI protocol. * [SPIDigitalPot](http://arduino.cc/en/Tutorial/SPIDigitalPot): control a AD5206 digital potentiometer using the SPI protocol.  Servo Library  * [Knob](http://arduino.cc/en/Tutorial/Knob): control the shaft of a servo motor by turning a potentiometer. * [Sweep](http://arduino.cc/en/Tutorial/Sweep): sweeps the shaft of a servo motor back and forth.  Software Serial Library  * [Software Serial Example](http://arduino.cc/en/Tutorial/SoftwareSerialExample): how to use theSoftwareSerial Library...Because sometimes one serial port just isn't enough! * [Two Port Receive](http://arduino.cc/en/Tutorial/TwoPortReceive): how to work with multiple software serial ports.  Stepper Library  * [Motor Knob](http://arduino.cc/en/Tutorial/MotorKnob): control a highly accurate stepper motor using a potentiometer.  Wire Library  * [SFRRanger\_reader](http://arduino.cc/en/Tutorial/SFRRangerReader): read a Devantech SRFxx ultra-sonic range finder using I2C communication. * [digital\_potentiometer](http://arduino.cc/en/Tutorial/DigitalPotentiometer): control a AD5171 digital pot using the Wire Library. * [master reader/slave sender](http://arduino.cc/en/Tutorial/MasterReader): set up two (or more) arduino boards to share information via a master reader/slave sender configuration. * [master writer/slave reader](http://arduino.cc/en/Tutorial/MasterWriter): allow two (or more) arduino boards to share information using a master writer/slave reader set up.  WiFi Library  * [ConnectNoEncryption](http://arduino.cc/en/Tutorial/ConnectNoEncryption) : Demonstrates how to connect to an open network * [ConnectWithWEP](http://arduino.cc/en/Tutorial/ConnectWithWEP) : Demonstrates how to connect to a network that is encrypted with WEP * [ConnectWithWPA](http://arduino.cc/en/Tutorial/ConnectWithWPA) : Demonstrates how to connect to a network that is encrypted with WPA2 Personal * [ScanNetworks](http://arduino.cc/en/Tutorial/ScanNetworks) : Displays all WiFi networks in range * [WiFiChatServer](http://arduino.cc/en/Tutorial/WiFiChatServer) : Set up a simple chat server * [WiFiPachubeClient](http://arduino.cc/en/Tutorial/WiFiPachubeClient) : connect to pachube.com, a free datalogging site * [WiFiPachubeClientString](http://arduino.cc/en/Tutorial/WiFiPachubeClientString): send strings to pachube.com * [WiFiTwitterClient](http://arduino.cc/en/Tutorial/WiFiTwitterClient) : A Twitter client with Strings * [WiFiWebClient](http://arduino.cc/en/Tutorial/WiFiWebClient) : Connect to a remote webserver * [WiFiWebClientRepeating](http://arduino.cc/en/Tutorial/WiFiWebClientRepeating): Repeatedly make HTTP calls to a server * [WiFiWebServer](http://arduino.cc/en/Tutorial/WiFiWebServer) : Serve a webpage from the WiFishield  Esplora LibraryBeginners  * [EsploraBlink](http://arduino.cc/en/Tutorial/EsploraBlink) : Blink the Esplora's RGB LED * [EsploraAccelerometer](http://arduino.cc/en/Tutorial/EsploraAccelerometer) : Read the values from the accelerometer * [EsploraJoystickMouse](http://arduino.cc/en/Tutorial/EsploraJoystickMouse) : Use the Esplora's joystick to control the cursor on your computer * [EsploraLedShow](http://arduino.cc/en/Tutorial/EsploraLedShow) : Use the Joystick and slider to create a light show with the LED * [EsploraLedShow2](http://arduino.cc/en/Tutorial/EsploraLedShow2) : Use the Esplora's microphone, linear potentiometer, and light sensor to change the color of the onboard LED. * [EsploraLightCalibrator](http://arduino.cc/en/Tutorial/EsploraLightCalibrator) : Read the values from the accelerometer * [EsploraMusic](http://arduino.cc/en/Tutorial/EsploraMusic) : Make some music with the Esplora * [EsploraSoundSensor](http://arduino.cc/en/Tutorial/EsploraSoundSensor) : Read the values from the Esplora's microphone * [EsploraTemperatureSensor](http://arduino.cc/en/Tutorial/EsploraTemperatureSensor) : Read the temperature sensor and get the temperature in in Farhenheit or Celsius.  Experts  * [EsploraKart](http://arduino.cc/en/Tutorial/EsploraKart) : Use the Esplora as a controller to play a kart racing game. * [EsploraTable](http://arduino.cc/en/Tutorial/EsploraTable) : Print the Esplora sensor information to a table format. * [EsploraRemote](http://arduino.cc/en/Tutorial/EsploraRemote) : Connect the Esplora to Processing and control the outputs. * [EsploraPong](http://arduino.cc/en/Tutorial/EsploraPong) : Play Pong with the Esplora using Processing.  Arduino as ISP Programmer [ArduinoISP](http://arduino.cc/en/Tutorial/ArduinoISP) turns your Arduino into an in-circuit programmer to re-program Atmega chips. Useful when you need to re-load the bootloader on an Arduino, if you're [going from Arduino to an Atmega on a breadboard](http://arduino.cc/en/Tutorial/ArduinoToBreadboard), or if you're making your own [Arduino-compatible circuit on a breadboard](http://arduino.cc/en/Main/Standalone). More For a huge list of examples from the Arduino community, see the [interfacing with hardware](http://www.arduino.cc/playground/Main/InterfacingWithHardware) page on the playground wiki. Also see the list of [old examples](http://arduino.cc/en/Tutorial/Old). |