



# Arduino

For Beginners

by Rick Suel

# What is Arduino?

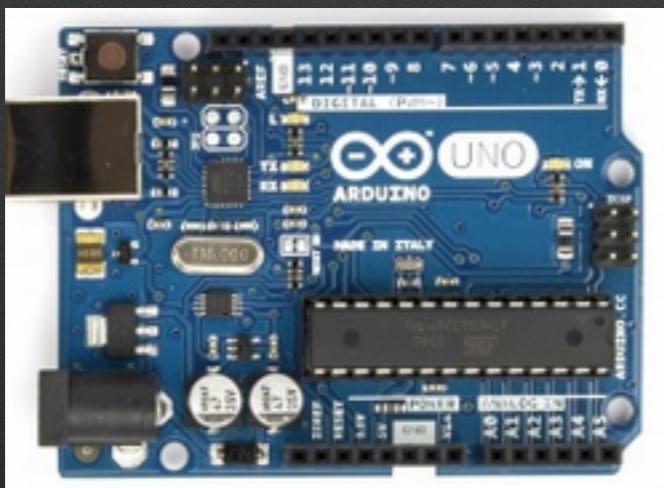
Arduino is an open source hardware and software platform that makes it very simple to experiment with interactive electronics. Hardware is expandable using “shields”.

ENTRY LEVEL	ARDUINO UNO	ARDUINO PRO	ARDUINO PRO MINI	ARDUINO MICRO	ARDUINO NANO
	ARDUINO STARTER KIT	ARDUINO BASIC KIT	ARDUINO MOTOR SHIELD		
ENHANCED FEATURES	ARDUINO MEGA	ARDUINO ZERO	ARDUINO DUE	ARDUINO PROTO SHIELD	
INTERNET OF THINGS	ARDUINO YÚN	ARDUINO ETHERNET SHIELD	ARDUINO GSM SHIELD	ARDUINO WIFI SHIELD 101	
WEARABLE	ARDUINO GEMMA	ARDUINO LILYPAD	ARDUINO LILYPAD SIMPLE	ARDUINO LILYPAD USB	
3D PRINTING	MATERIA 101				
	BOARDS	MODULES	SHIELDS	KITS	ACCESSORIES
					COMING NEXT

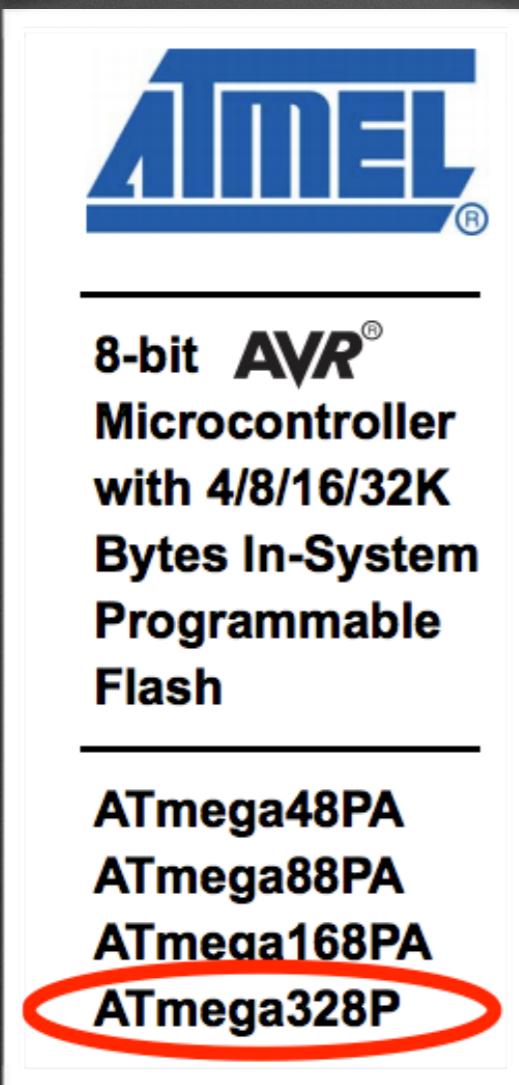
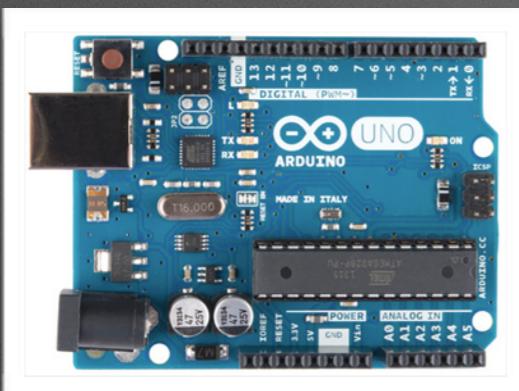


# Arduino vs Raspberry Pi

- Arduino runs on a 16MHz micro controller (\$10 - \$25)
- Runs user-written C programs
- Great for interfacing with sensors
- Very simple to get started
- Pi 2 runs on a 900MHz quad-core CPU (\$35)
- Runs Linux OS (or Windows 10)
- Great for multi-media projects but can also interface with sensors
- Harder to get started



# The Arduino Uno



## Technical specs

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

# The Tools

[www.arduino.cc/en/Main/Software](http://www.arduino.cc/en/Main/Software)

## Download the Arduino Software



### ARDUINO 1.6.5

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.

This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions.

[Windows Installer](#)

[Windows ZIP file for non admin install](#)

[Mac OS X 10.7 Lion or newer](#)

[Linux 32 bits](#)

[Linux 64 bits](#)

[Release Notes](#)

[Source Code](#)

[Checksums](#)

# The ‘Sketch’

```
sketch_aug14b | Arduino 1.6.4
sketch_aug14b
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
```

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

**Executed 1st:**

- Initialization Code
- Runs once

**Executed 2nd:**

- Application code
- Runs “forever”

# Language Reference

<https://www.arduino.cc/en/Reference/HomePage>

## Language Reference

Arduino programs can be divided in three main parts: *structure*, *values* (variables and constants), and *functions*.

### Structure

- `setup()`
- `loop()`

### Control Structures

- `if`
- `if...else`
- `for`
- `switch case`
- `while`
- `do... while`
- `break`
- `continue`
- `return`
- `goto`

### Variables

#### Constants

- `HIGH | LOW`
- `INPUT | OUTPUT | INPUT_PULLUP`
- `LED_BUILTIN`
- `true | false`
- `integer constants`
- `floating point constants`

#### Data Types

- `void`
- `boolean`
- `char`
- `unsigned char`
- `byte`
- `int`

### Functions

#### Digital I/O

- `pinMode()`
- `digitalWrite()`
- `digitalRead()`

#### Analog I/O

- `analogReference()`
- `analogRead()`
- `analogWrite() - PWM`

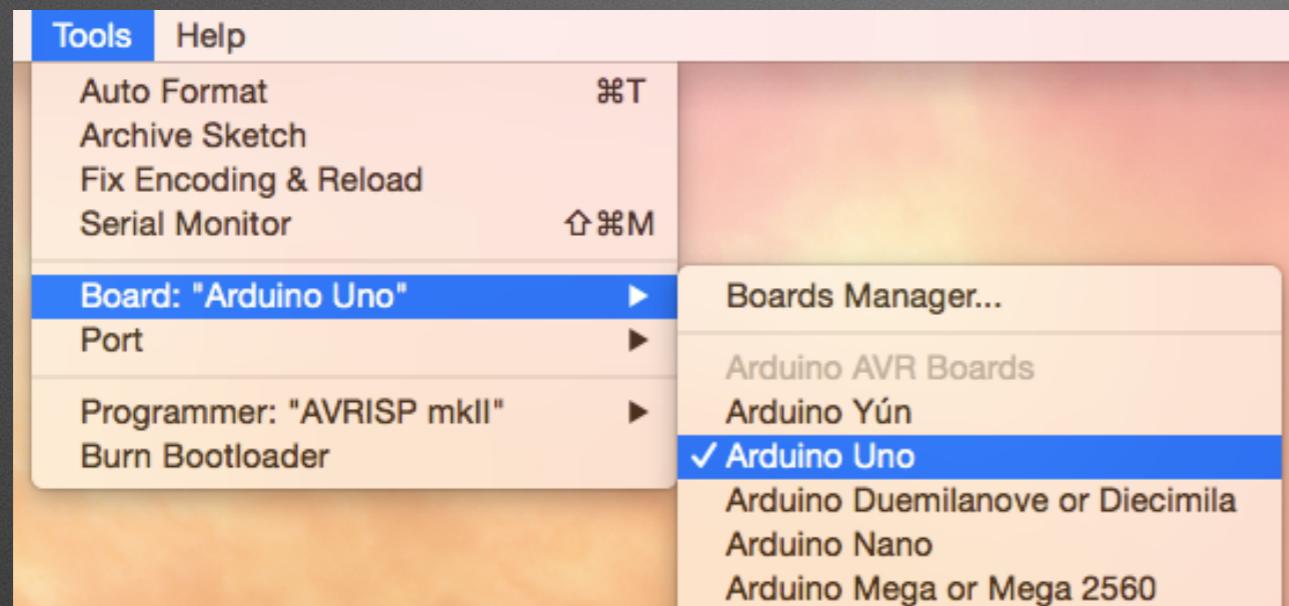
#### Due & Zero only

- `analogReadResolution()`
- `analogWriteResolution()`

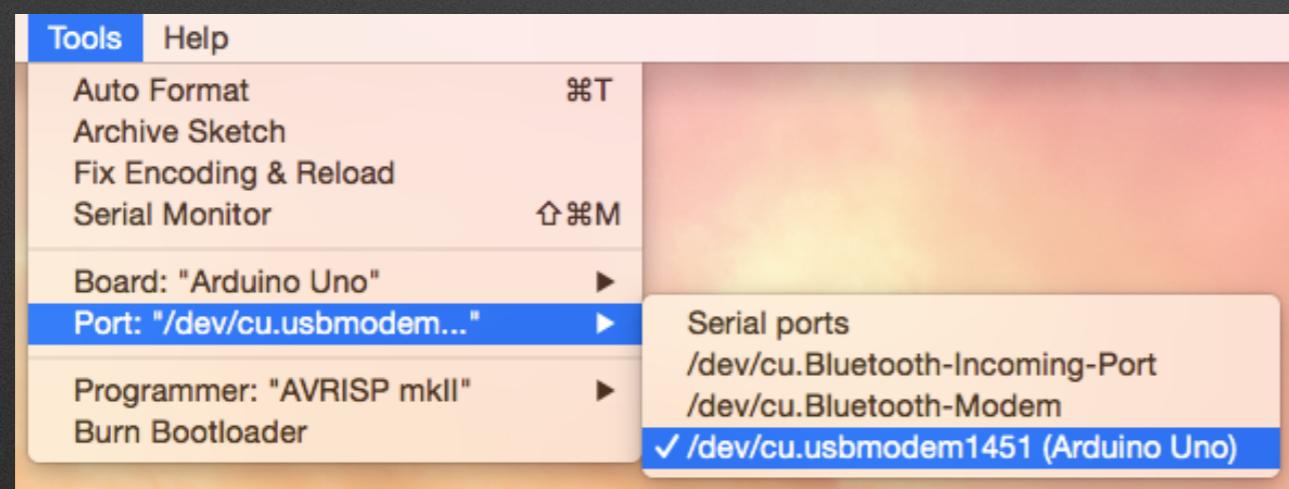
#### Advanced I/O

# Connecting the Arduino

1. Connect the USB cable from the Arduino to your PC
2. Select Tools->Board->Arduino Uno



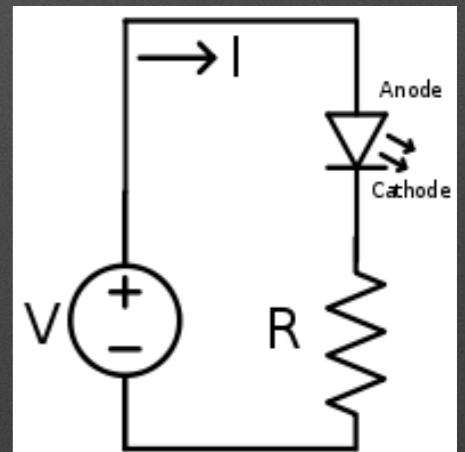
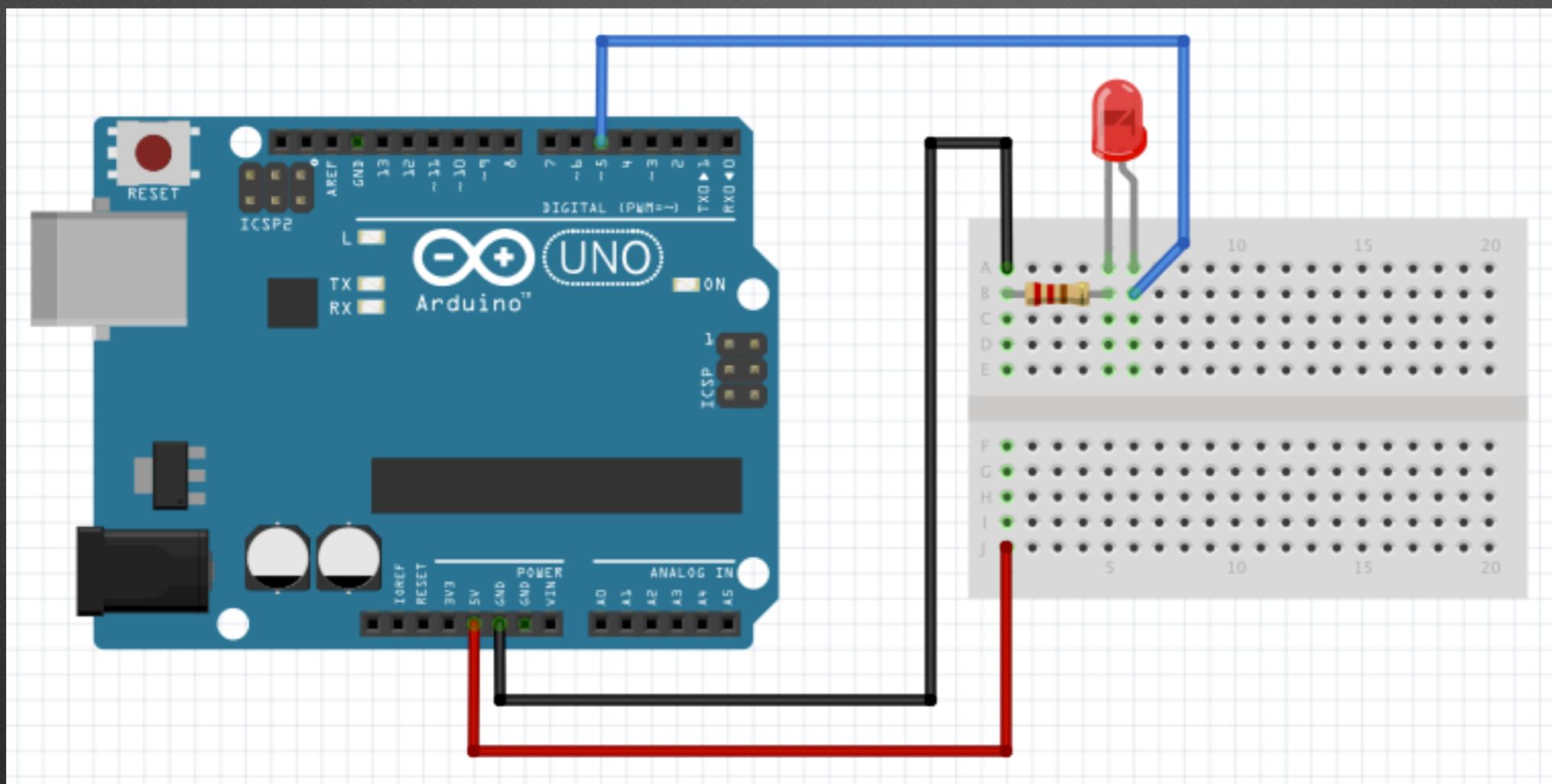
3. Select Tools->Port->....(Arduino Uno)



# Ex 1: Blink an LED

## Requirements:

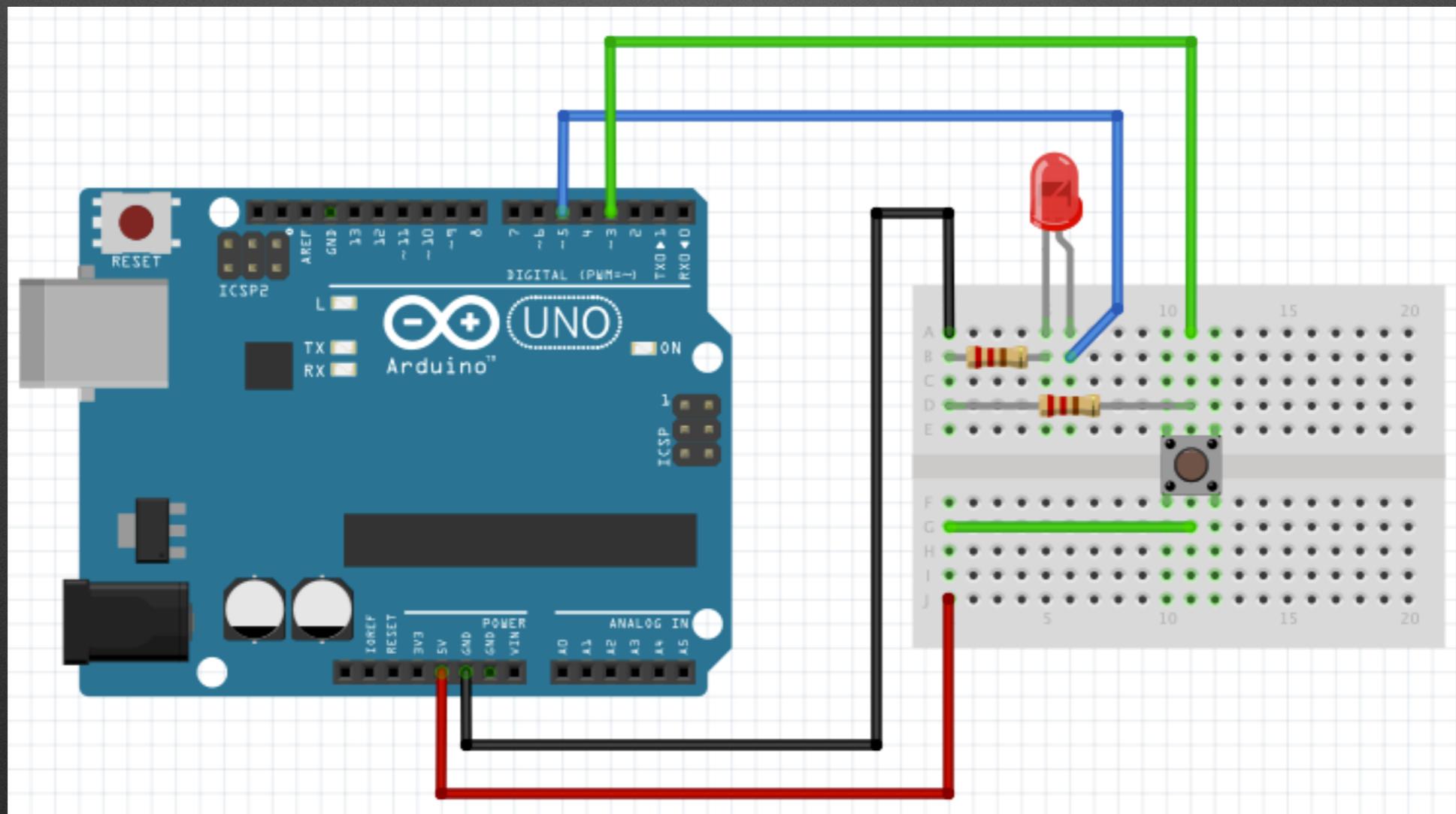
- Connect anode (+) of LED to pin 5 of Arduino
- Connect cathode (-) of LED through 200 Ohm (blue) resistor to ground
- Blink the LED at 1Hz (500ms on, 500ms off)



# Ex 2: Add a button

## Requirements:

- Connect one pin of push button to pin 3 or Arduino.
- Connect other pin of push button to ground through 10K “pulldown” resistor.
- When button is pressed, turn LED on and print “press” in serial window.
- When button is released, turn LED off and print “release” in serial window.



# Ex 3: External Interrupts

## Requirements:

- Keep the same circuit from example 3.
- Use Interrupt 1 (pin 3) to achieve the same button/LED functionality.
- Simplifies the loop function!

## attachInterrupt()

### Description

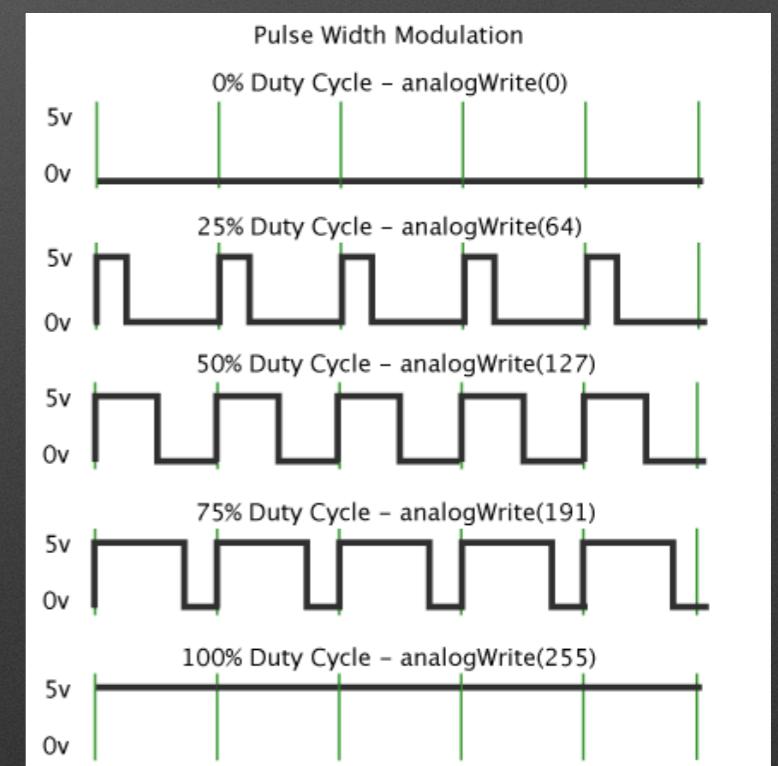
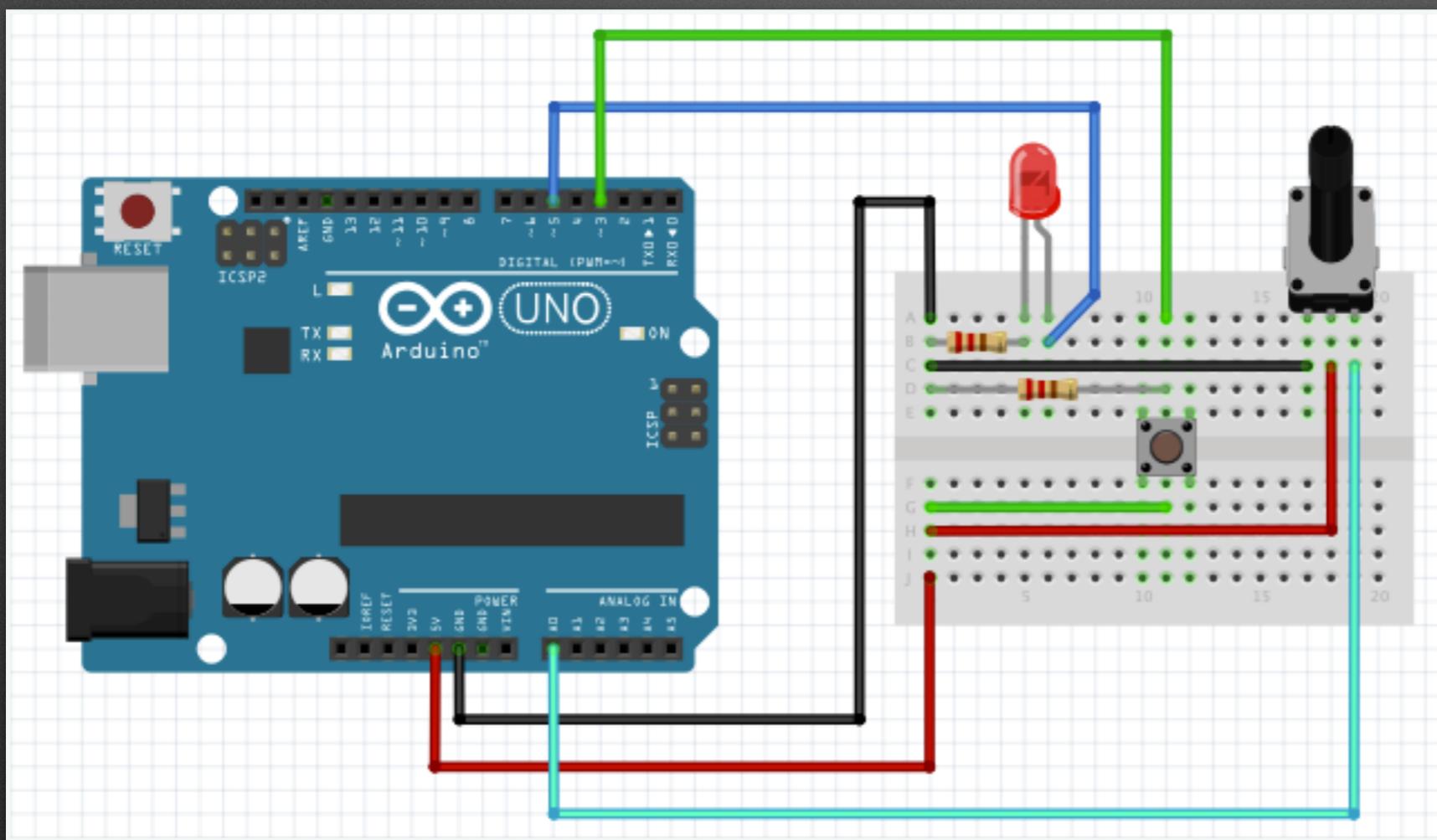
Specifies a named Interrupt Service Routine (ISR) to call when an interrupt occurs. Replaces any previous function that was attached to the interrupt. Most Arduino boards have two external interrupts: numbers 0 (on digital pin 2) and 1 (on digital pin 3). The table below shows the available interrupt pins on various boards.

Board	int.0	int.1	int.2	int.3	int.4	int.5
Uno, Ethernet	2	3				
Mega2560	2	3	21	20	19	18
Leonardo	3	2	0	1	7	
Due			(see below)			

# Ex 4: Analog In / PWM Out

## Requirements:

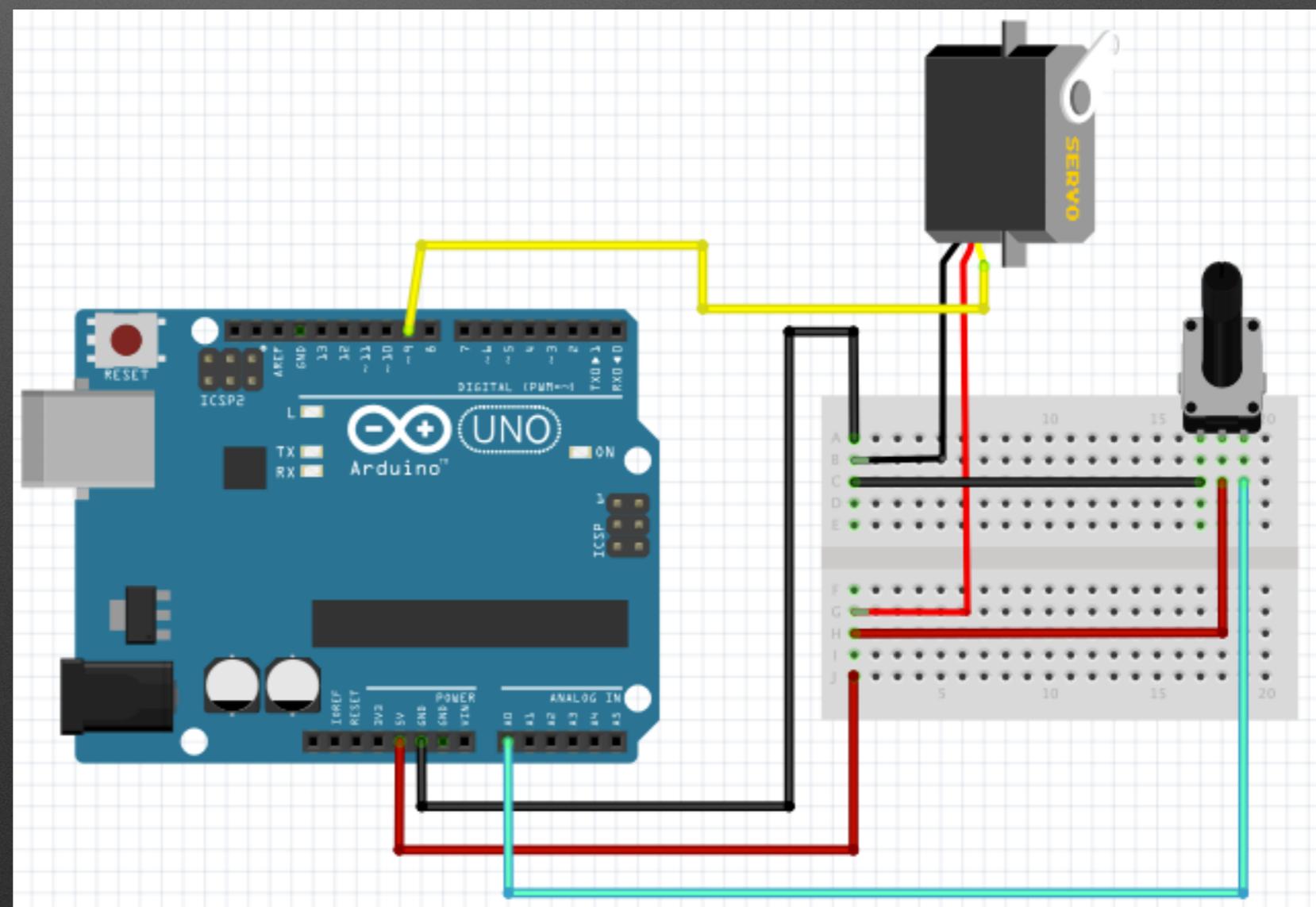
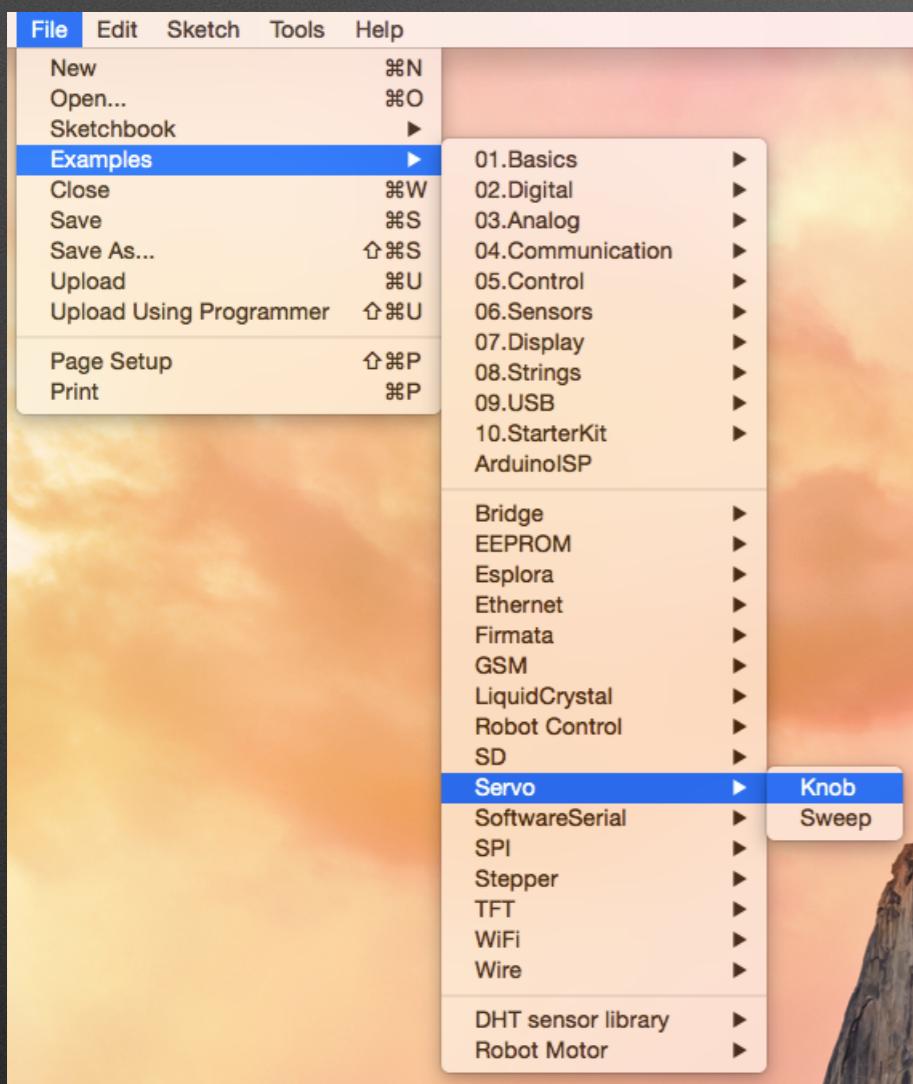
- Connect the potentiometer to pin A0 as shown below.
- Drive the LED as a PWM output using the analog input to adjust brightness.
- Press/hold the button and you should be able to adjust the brightness.
- Print brightness to the serial window.



# Ex. 5: Drive a Servo (Library)

# Requirements:

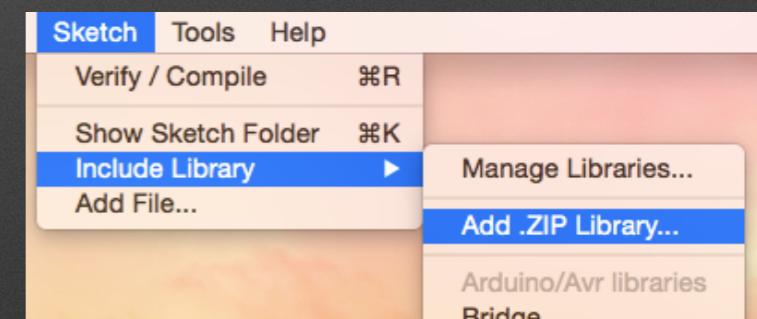
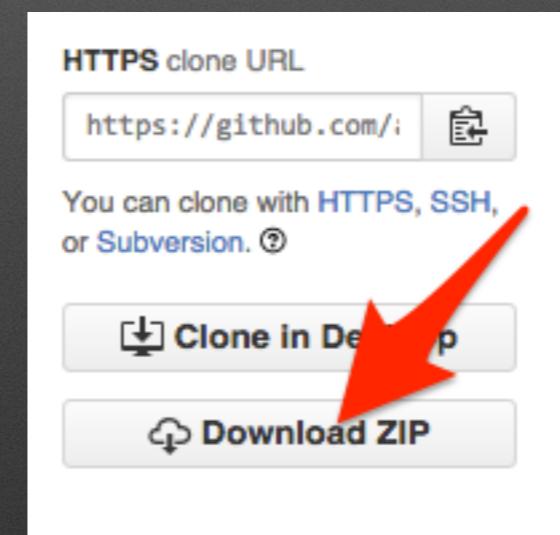
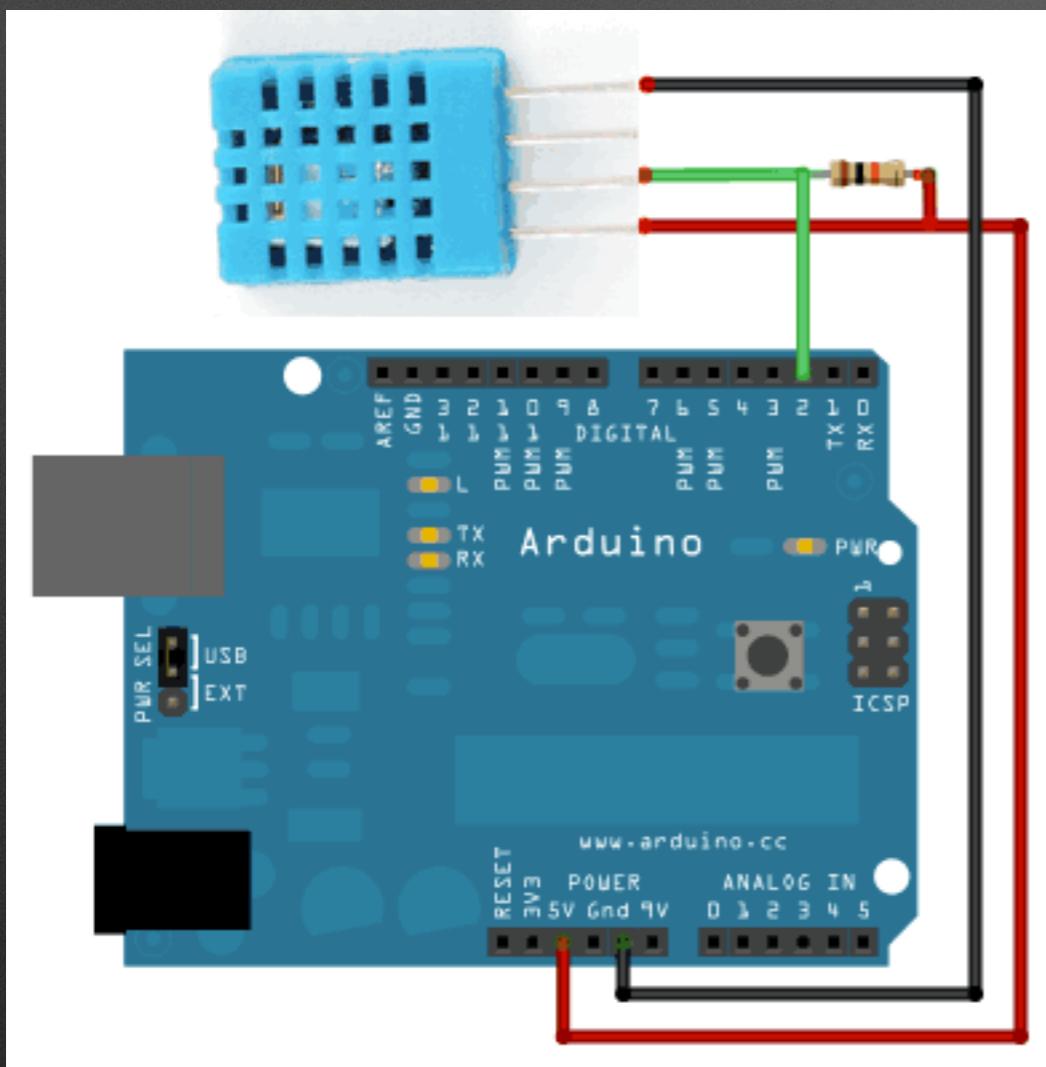
- Connect the servo and potentiometer as shown below (Brn=Gnd, Red=5V, Org=Sig)
  - Launch the built in servo example
  - Download and run!



# Ex. 6: Temp Sensor (Library)

## Requirements:

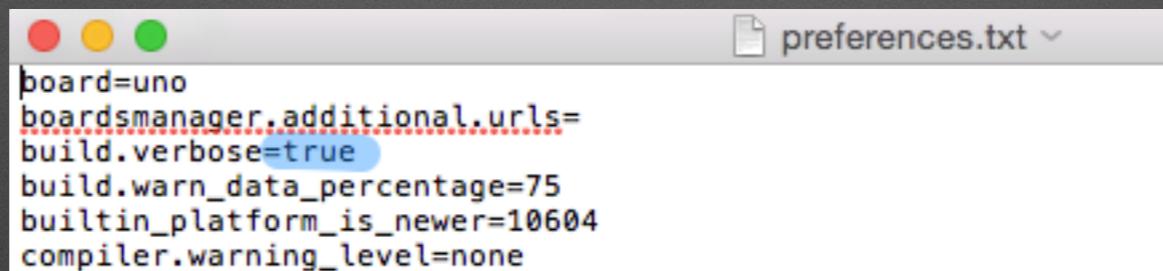
- Follow this tutorial: <https://learn.adafruit.com/dht/connecting-to-a-dhtxx-sensor>
- Download the library zip from here: <https://github.com/adafruit/DHT-sensor-library>
- Import the library as a zip file
- Launch the example, configure as DHT



# Advanced: Verbose Build

## Turn On Verbose Build:

- Arduino->Preferences - then click the link to the preferences near the bottom
- Close Arduino software
- Open the preferences.txt file and change the following option to “true”



```
board=uno
boardsmanager.additional.urls=
build.verbose=true
build.warn_data_percentage=75
builtin_platform_is_newer=10604
compiler.warning_level=none
```

- Restart the Arduino software and build your project.
- Now you can see everything that goes into building that sketch!

# Advanced: Eclipse IDE/TDD

## Github Project

- [https://github.apl.ge.com/NelsonTanquero/arduino\\_tdd](https://github.apl.ge.com/NelsonTanquero/arduino_tdd)

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

- Project Explorer:** Shows the project structure for 'arduinoTdd' under 'C/C++ Projects'. It includes 'lib', 'Release', and 'src' folders containing 'Application', 'Events', 'HardwareInterfaces', 'Time', and 'Utilities' subfolders with their respective source and header files.
- Editor View:** Displays the content of 'main.c'. The code initializes a timer module and runs it in a loop.

```
16 TimeSource_Interrupt_t oneMsecTickTimeSource;
17 I_Interrupt_t *oneMsecTickInterrupt;
18 TimerModule_t appTimerModule;
19 Timer_t periodicLedBlinkTimer;
20 Application_t ledBlinkApplication;
21
22 void setup()
23 {
24     oneMsecTickInterrupt = ArduinoTimer2_Init();
25     arduinoGpioGroup = GpioGroup_Arduino_Init();
26
27     TimeSource_Interrupt_Init(&oneMsecTickTimeSource, oneMsecTickInterrupt);
28     TimerModule_Init(&appTimerModule, &oneMsecTickTimeSource.timeSource);
29
30     Application_Init(
31         &ledBlinkApplication,
32         &arduinoGpioGroup->interface,
33         &appTimerModule,
34         OneSecInMsec,
35         GpioChannel_13);
36 }
37
38 void loop()
39 {
40     TimerModule_Run(&appTimerModule);
41 }
```

- Console View:** Shows the terminal output of the build process and a test run.

```
<terminated> arduino_tdd_build [Program] /usr/bin/make
compiling AllTests.cpp
compiling Application.c
compiling Constants_Binary.c
compiling EventSubscription.c
compiling Event_Synchronous.c
compiling LinkedList.c
compiling TimeSource_Interrupt.c
compiling Timer.c
Building archive Testing/Build/Lib//libname_this_in_the_makefile.a
a - Testing/Build//src/Application/Application.o
a - Testing/Build//src/Application/Constants_Binary.o
a - Testing/Build//src/Application/Events/EventSubscription.o
a - Testing/Build//src/Application/Events/Event_Synchronous.o
a - Testing/Build//src/Application/Time/LinkedList.o
a - Testing/Build//src/Application/Time/TimeSource_Interrupt.o
a - Testing/Build//src/Application/Time/Timer.o
Linking Testing/Build/arduino_cppUTest_tests
Running ./Testing/Build/arduino_cppUTest_tests
OK (101 tests, 101 ran, 239 checks, 0 ignored, 0 filtered out, 5 ms)
```

# Go Make Something!

