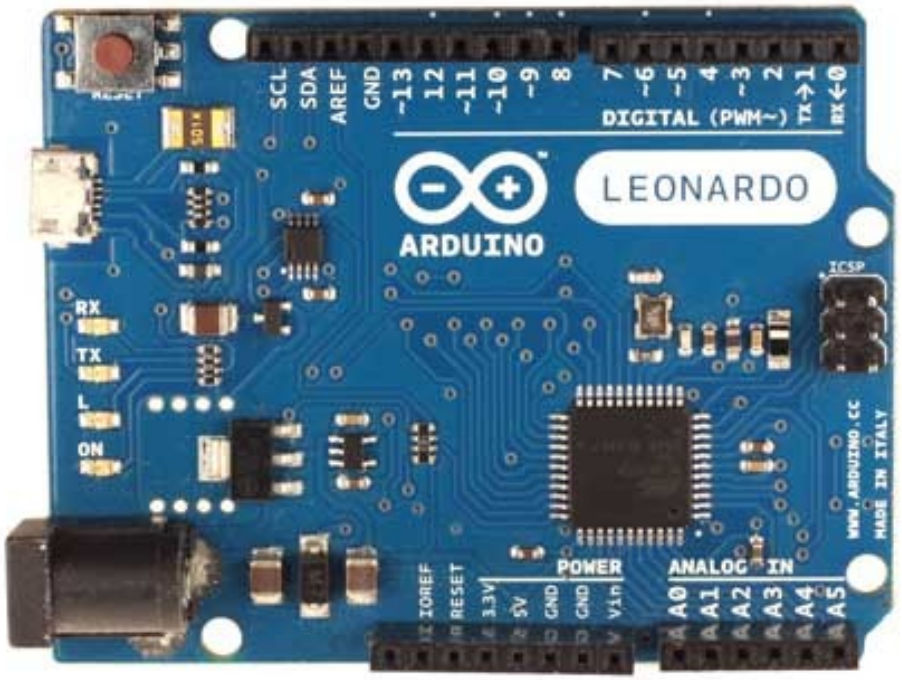
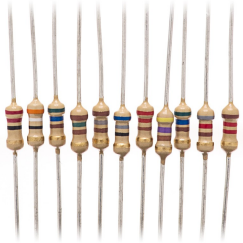
Arduino Electronics Introduction

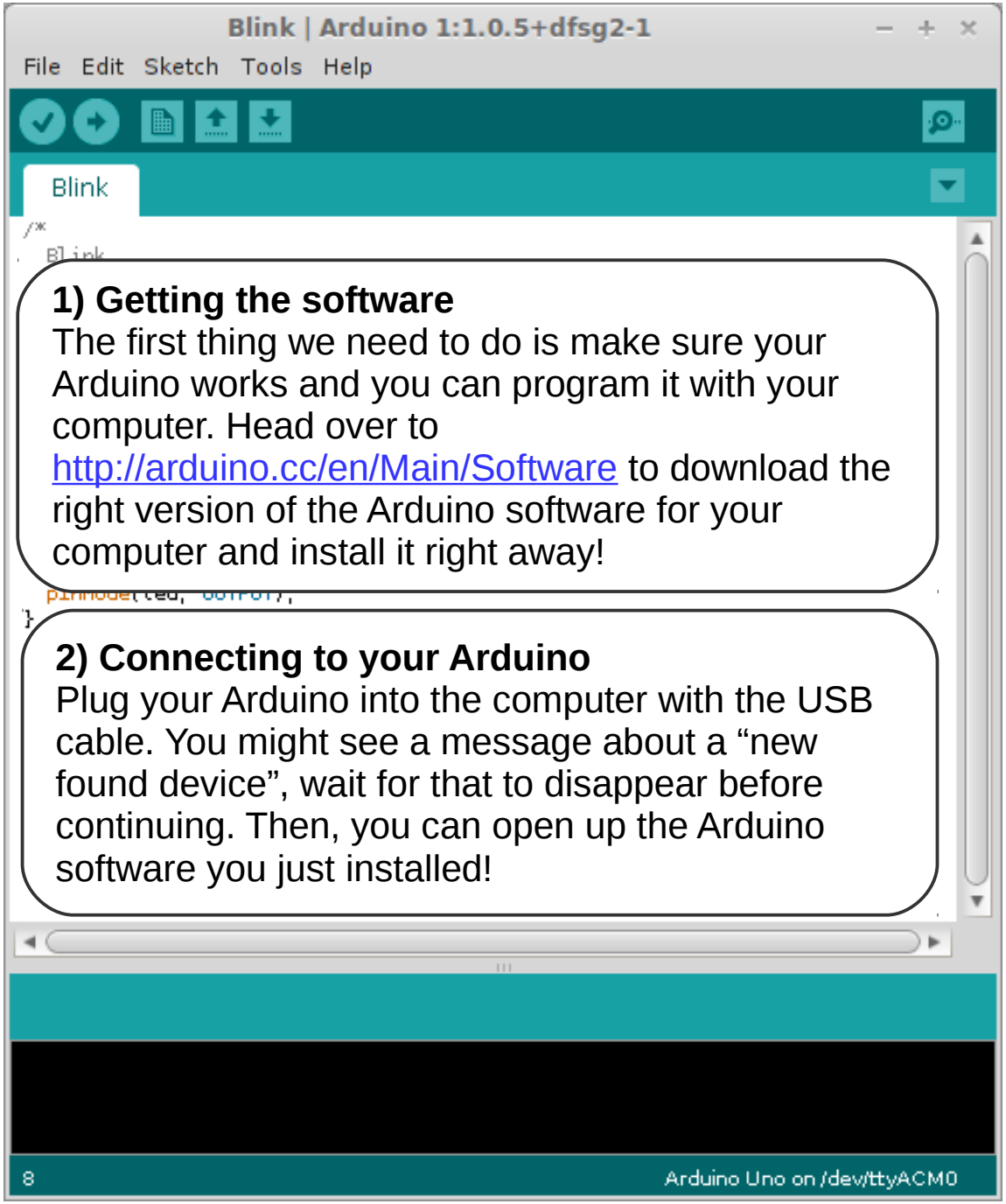
Welcome! This tutorial is designed to teach you the basics of making your own electronic circuits using the Arduino microcontroller, the blue thing on the left. You will learn how to safely connect different kinds of components to the Arduino and how to program it to interact with the outside world.





We will be using several different electronic components with the Arduino. Can you name the ones pictured on the page?

Don't worry if you have never programmed an Arduino before, it's the easiest thing in the world!
Have a look at: http://arduino.cc/en/Tutorial



1) Getting the software
The first thing we need to do is make sure your Arduino works and you can program it with your computer. Head over to http://arduino.cc/en/Main/Software to download the right version of the Arduino software for your computer and install it right away!

2) Connecting to your Arduino
Plug your Arduino into the computer with the USB cable. You might see a message about a "new found device", wait for that to disappear before continuing. Then, you can open up the Arduino software you just installed!

3) Load An Example Program
When the software opens, click on: File -> Examples -> 01 Basics -> Blink. You will see another window open with some example code. Don't worry if you don't understand yet, we'll get to that!

4) Configure your board
We are using an "Arduino Leonardo". So under Tools -> Board Select the Arduino Leonardo! You will also need to select the right "port" under Tools -> Port. It is usually the one with the biggest number at the end!

5) Program the board!
Click on the button with the arrow pointing right under "Edit". This will compile your code and "upload" it to the Arduino! When it finishes, you should see a single LED on the board flashing on and off. Congratulations, you now know how to program and Arduino!

File Edit Sketch Tools Help

basics

```
/*
  This is a very basic Arduino Program to show you how to structure your
  code. Use it for reference when writing your own programs!
*/

// This is how you declare a variable. These store which pins we
// will access.
int button_pin = 5;
int light_pin  = 6;

// The setup function runs once every time you restart the Arduino.
// Use it to Set up all of your variables
void setup() {
  // This is how we set a pin to an input we can "read"
  pinMode(button_pin, INPUT);
  // You can probably guess what this line does!
  pinMode(light_pin, OUTPUT);
}

// The loop function runs over and over again forever. This is where most
// code will go! You can also declare your own functions like below.
void loop() {
  // Store if the button is pressed or not.
  int is_button_pressed = digitalRead(button_pin);

  if(is_button_pressed == 1){
    digitalWrite(light_pin, HIGH); // if it is pressed turn the light on.
  }
  else {
    digitalWrite(light_pin, LOW); // Otherwise turn it off!
  }
}
```

Done Saving.

3 Arduino Uno on COM1

code_statements | Arduino 1:1.0.5+dfsg2-2

File Edit Sketch Tools Help

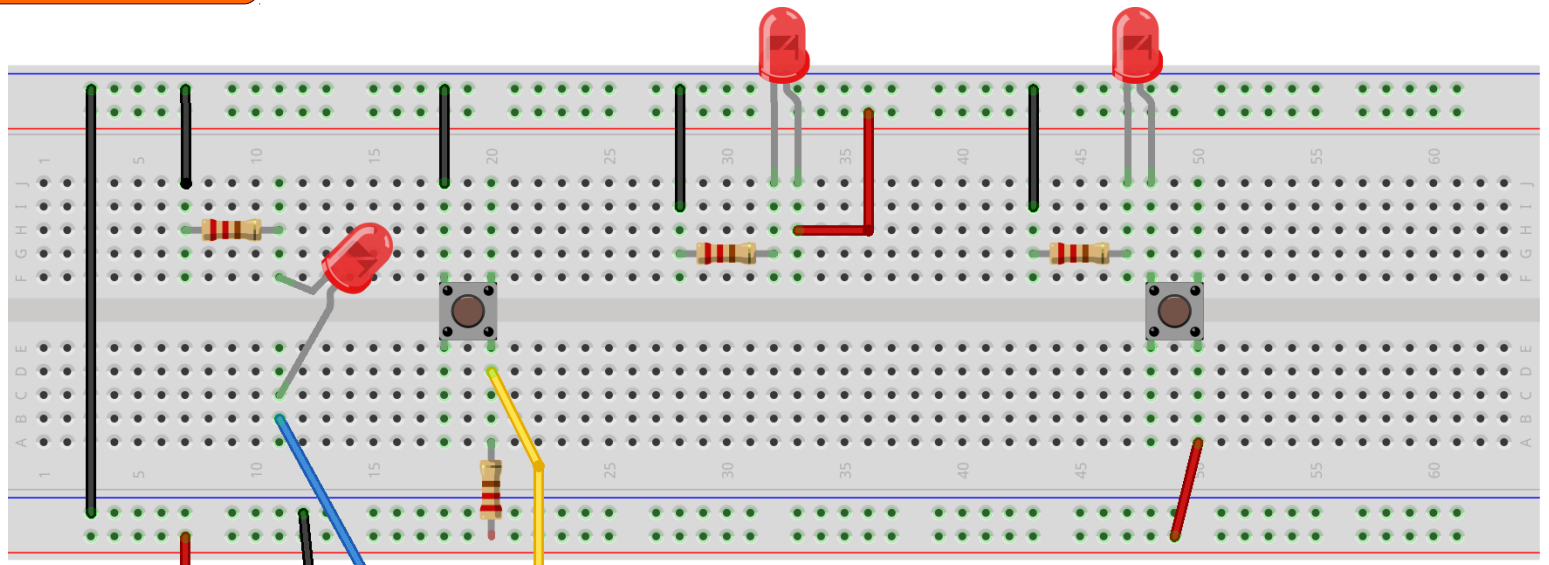
code_statements

```
//  Here you can see how to do basic programming tasks like loops, functions and
//  printing messages from the arduino back to your computer.

int button_pin = 5;
int light_pin  = 6;
int press_count = 0;
int is_button_pressed = 0;

void setup() {
  pinMode(button_pin, INPUT);
  pinMode(light_pin, OUTPUT);
  // Open a "Serial connection" to your computer. More on this later!
  Serial.begin(9600);
  Serial.println("Hello World!");
}

void loop() {
  is_button_pressed = digitalRead(button_pin);

  while(is_button_pressed == 1){
    is_button_pressed = digitalRead(button_pin); // Update this with every loop!
    // How is this different to how we turned it on before?
    digitalWrite(button_pin, is_button_pressed);
  }

  press_count += 1; // Add 1 to the button press counter.
  send_message(press_count);
}

void send_message(int button_count) { // Here we declare our own function.
  Serial.println("Button Pressed %d times", button_count);
}
```

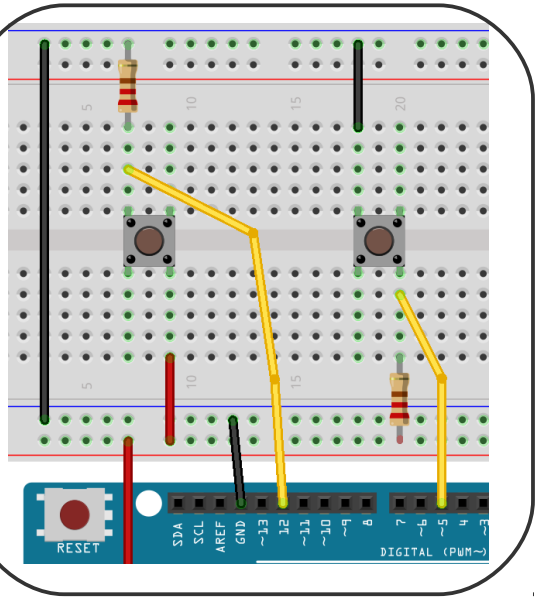
1 Arduino Uno on COM1

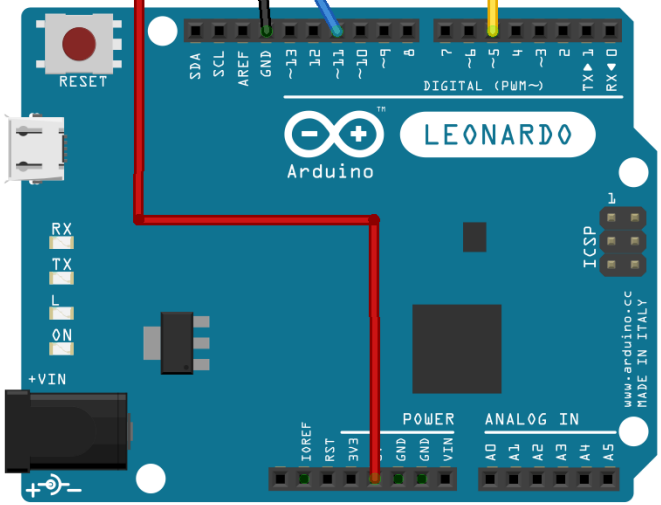
LED's and Resistors

These are the two most basic components we can use. Try copying this circuit and controlling the LED with the Button and the Arduino. The long leg of the LED always connects to power.



We **always use a resistor with an LED.** This makes sure it doesn't draw too much power and burn out!





Try building the circuit on the left. What do you notice is different about the values the buttons normally take? Print them out so you can see them.

Here is the code you will need, can you work out where to put it all?

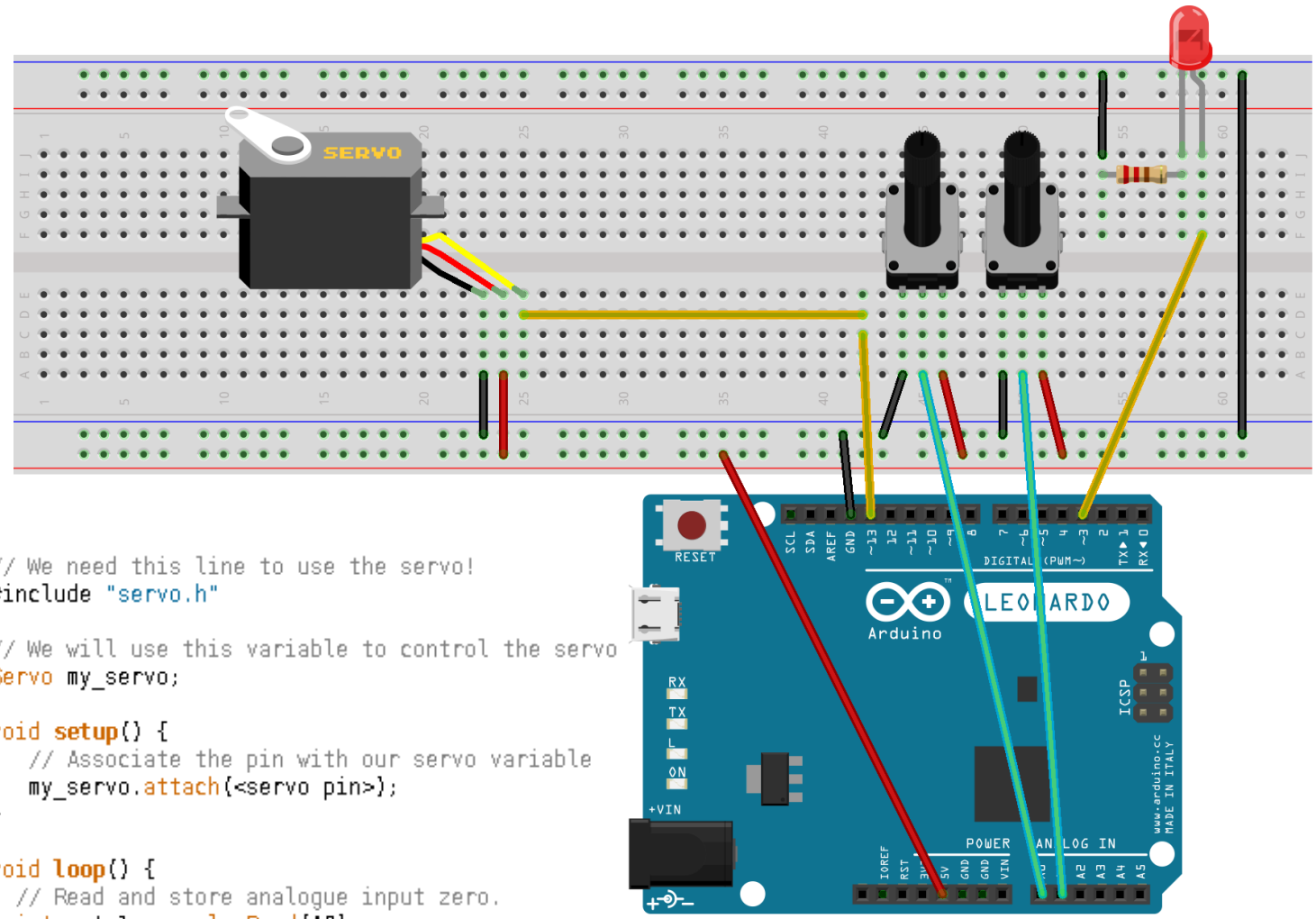
digital_read_write

```
// Declare a variable to store the button
int button_pressed;
// Read the value of the pin with the button
button_pressed = digitalRead(<pin number>);

// Turn on an output pint
digitalWrite(<pin number>, HIGH);
// Or turn off an output pin
digitalWrite(<pin number>, LOW);

// if statement
if(<my variable> == <some value>){
   // DO something!
}
else {
   // DO something ELSE!
}
```

Potentiometers
These are *variable* resistors, but we can change how much "resistance" they have. Think of them as sliding scales we can use to control things. In this circuit, we can control the brightness of an LED or the position of a servo arm. They have three pins. Left for ground, right for power and the middle to the Arduino analogue inputs.

Servo's
Our first mechanical output. They have an "arm" that moves back and forth depending on how we control it. They have three wires. One for power (red), one for ground (black) and the yellow one we connect to the Arduino to control it with

The way we read the value of the potentiometer is the same for all *analogue* inputs. So, the same could work for microphones or light sensors. Can you make an LED that gets brighter as the room darkens?




```
// We need this line to use the servo!
#include "servo.h"

// We will use this variable to control the servo
Servo my_servo;

void setup() {
   // Associate the pin with our servo variable
   my_servo.attach(<servo pin>);
}

void loop() {
  // Read and store analogue input zero.
  int pot_1 = analogRead(A0);
  // For more info on the "map" function, look at the arduino reference page.
  int toWrite = map(pot_1, 0, 1023, 0, 180);
  my_servo.write(toWrite);          // Move the servo arm!
  analogWrite(<LED pin>, toWrite); // Set the brightness of the LED
}
```

Arduino Electronics Introduction

This page has some reference code to help you with writing the programs for the Arduino. Please do ask if something doesn't make sense! You can also look on http://arduino.cc/en/Reference for more

reference_1

```
// BASIC CODE CONSTRUCTS

if (variable_1 == variable_2){
  // Some Code.
} else if(variable_1 > variable_2) {
  // Some more code.
} else {
  // EVEN MORE CODE! :O
}

int counter;
for(counter = 0; counter <= 10; counter++) {
   Serial.println(counter);
}

int counter = 10
while(counter >= 0) {
   Serial.println(counter);
   counter = counter - 1;
}

int add_numbers(int num_1, int num_2) {
  return num_1 + num_2;
}

void print_text(String to_print){
   Serial.print("Arduino says: ");
   Serial.println(to_print);
}
```

Done Saving.

reference_2 §

```
// HOW TO STRUCTURE YOUR PROGRAMS

// include statements always go first!
#include "Servo.h"

// declare all of your "global" variables at the top.
// Things like pin numbers for components or Servo's
int output_pin_1 = 3;
int output_pin_2 = 4;
int input_pin_1 = 5;
int servo_pin = 13;
Servo my_servo;

// Anything in this function will only happen once.
// Set up your input and output pins here.
void setup() {
  pinMode(output_pin_1, OUTPUT);
  pinMode(output_pin_2, OUTPUT);
  pinMode(input_pin_1,  INPUT);
  my_servo.attach(servo_pin);
}

// This code will run again and again. Here is the
// "main" part of the program.
void loop(){
  // You can work out what this bit should do
  // for yourself ;)
  // It is your program after all!
}
```

Done Saving.

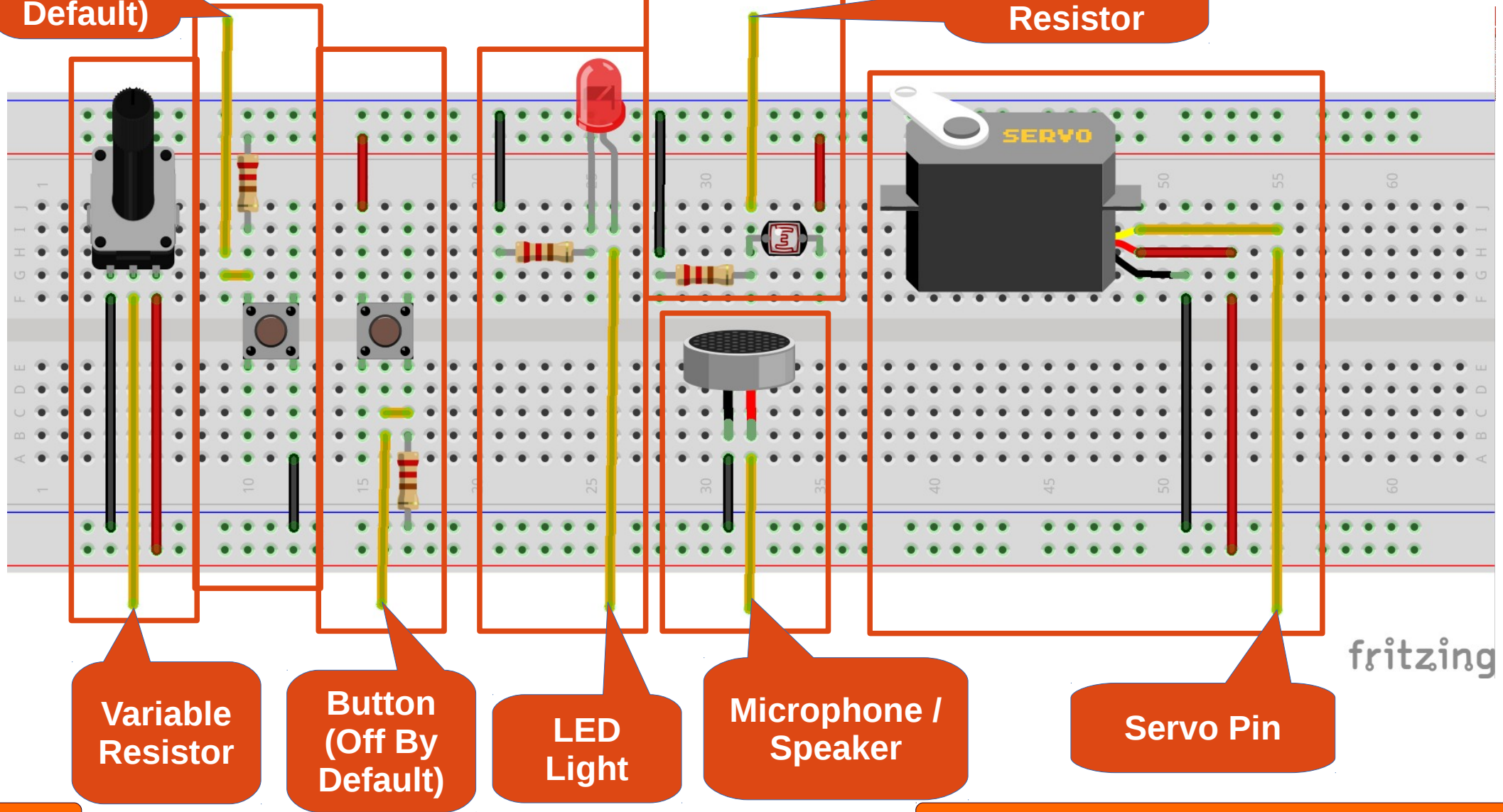
reference_3

```
// Controlling Pins.

void setup(){
   // Set the modes of pins in the setup function
   pinMode(<pin number>, INPUT);
   pinMode(<pin number>, OUTPUT);
}

void loop(){
  // Read the value of a digital input
  int pin_value = digitalRead(<pin_number>);
  // Write the value of a digital output
  digitalWrite(<pin_number>, HIGH); // ON
  digitalWrite(<pin_number>, LOW);  // OFF

  // Read the value of an analog input. This will
  // be between 0 and 1023. Do you know why?
  int analog_val = analogRead(A0);

  // The MAP function. It scales the value of a
  // variable from one range of numbers to another.
  // We use this alot for analog outputs!
  int mapped = map(analog_val, from_min, from_max,
                   to_min, to_max);

  // Write to an analog output. You can only write
  // analog values between 0 and 255. So we use "map"
  analogWrite(<pin_number>, analog_val);
}
```

Done Saving.

Arduino Electronics Introduction

This page has example diagrams for how to connect different types of component to the Arduino. You can use several of these circuits on the breadboard with the Arduino to make almost anything!



TEMPLATE PAGE