



Arduino training 1-Basic

Dr. DANG Hoang-Anh



What is Arduino?

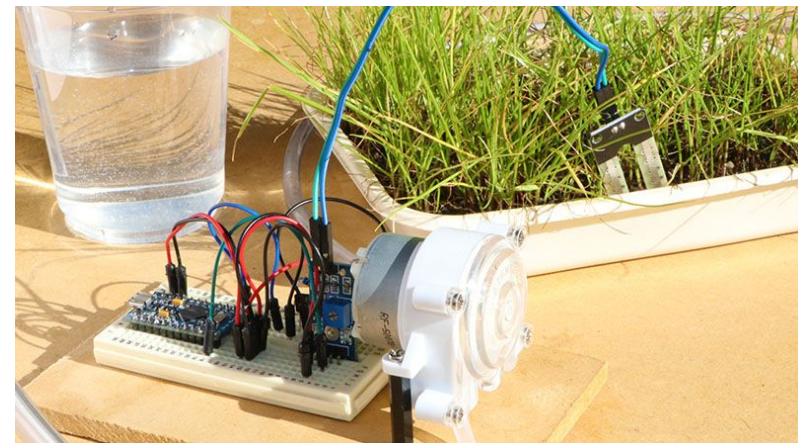
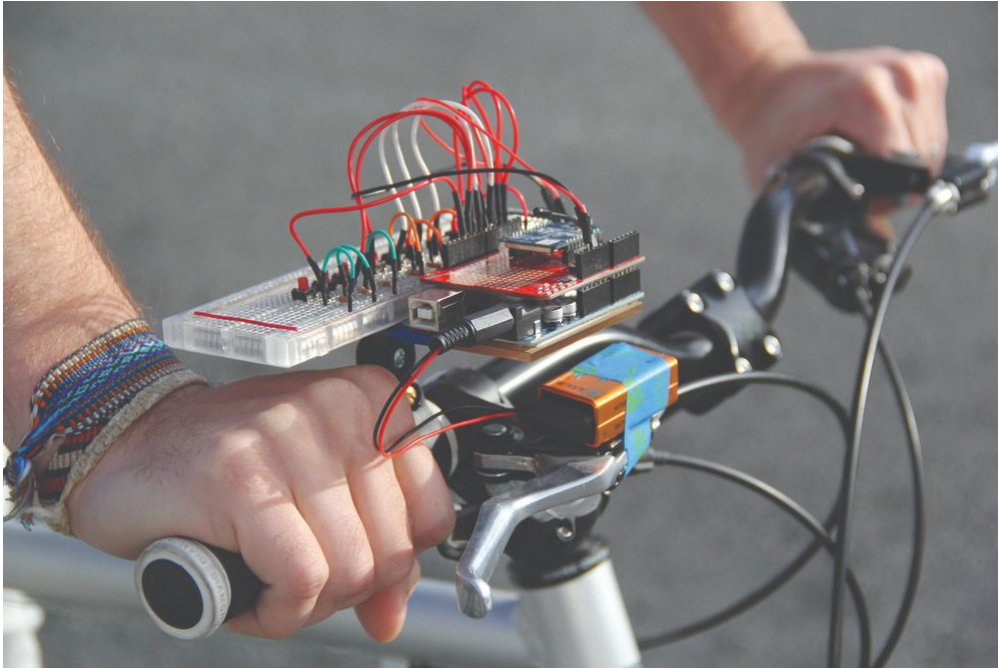
Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to

- read inputs - light on a sensor, a finger on a button, or a Twitter message
- turn it into an output - activating a motor, turning on an LED, publishing something online

You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the [Arduino programming language](#) (based on Wiring), and [the Arduino Software \(IDE\)](#), based on Processing

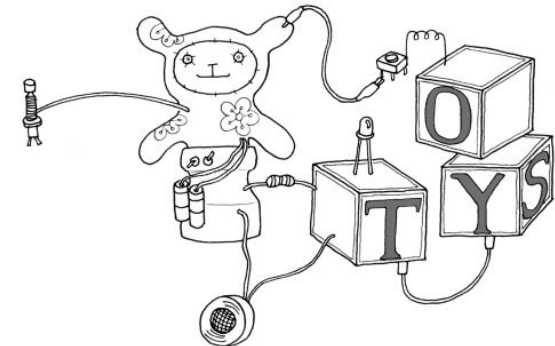
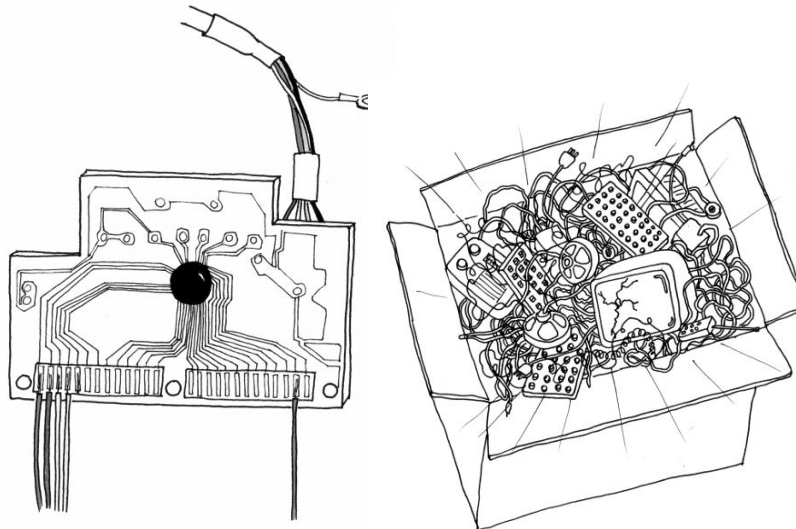
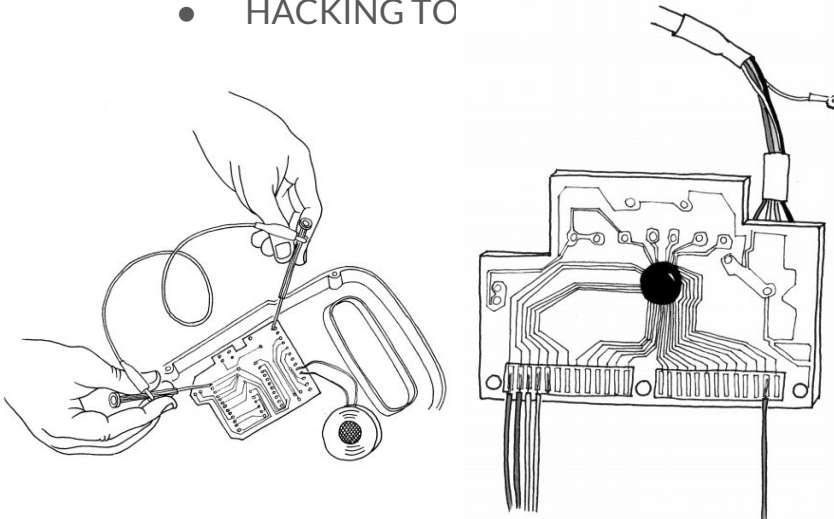
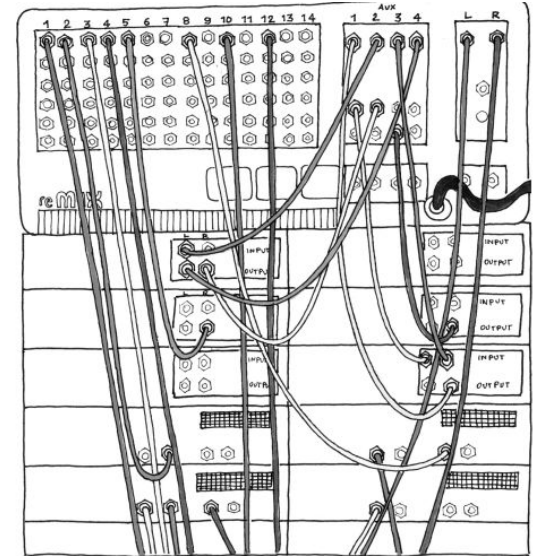
The Arduino Way

- PROTOTYPING



The Arduino Way

- TINKERING
- PATCHING
- CIRCUIT BENDING
- KEYBOARD HACKS
- WE LOVE JUNK
- HACKING TO

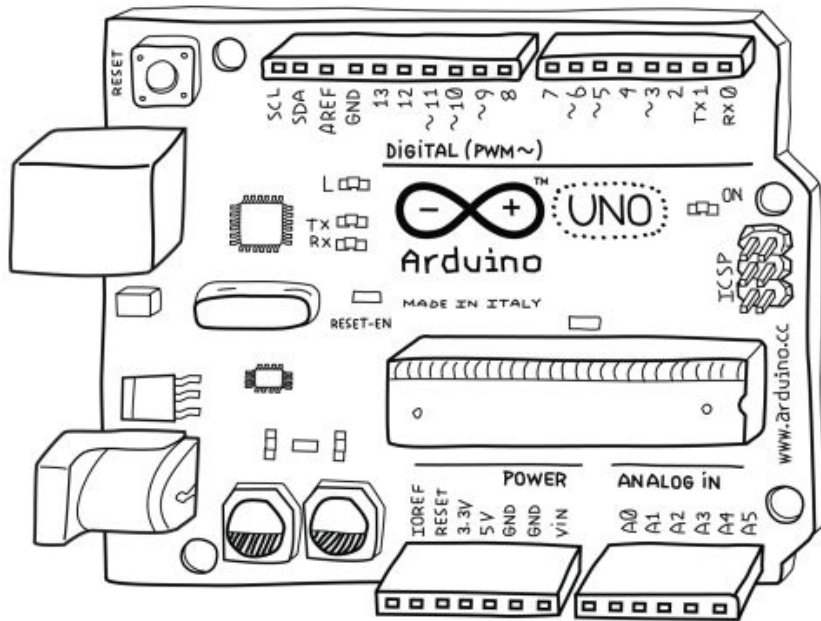




The Arduino Hardware

ENTRY LEVEL	<div>UNO</div> <div>LEONARDO</div> <div>101</div> <div>ESPLORA</div> <div>MICRO</div> <div>NANO</div> <div>MINI</div> <div>MKR2UNO ADAPTER</div> <div>STARTER KIT</div> <div>LCD SCREEN</div>
ENHANCED FEATURES	<div>MEGA</div> <div>ZERO</div> <div>DUE</div> <div>MEGA ADK</div> <div>MO</div> <div>MO PRO</div> <div>MKR ZERO</div> <div>MOTOR SHIELD</div> <div>USB HOST SHIELD</div> <div>PROTO SHIELD</div> <div>MKR PROTO SHIELD</div> <div>4 RELAYS SHIELD</div> <div>MEGA PROTO SHIELD</div> <div>MKR RELAY PROTO SHIELD</div> <div>ISP</div> <div>USB2SERIAL MICRO</div> <div>USB2SERIAL CONVERTER</div>
INTERNET OF THINGS	<div>YÚN</div> <div>ETHERNET</div> <div>TIAN</div> <div>INDUSTRIAL 101</div> <div>LEONARDO ETH</div> <div>MKR FOX 1200</div> <div>MKR WAN 1300</div> <div>MKR GSM 1400</div> <div>MKR1000</div> <div>YUN MINI</div> <div>YÚN SHIELD</div> <div>WIRELESS SD SHIELD</div> <div>WIRELESS PROTO SHIELD</div> <div>ETHERNET SHIELD V2</div> <div>GSM SHIELD V2</div> <div>MKR IoT BUNDLE</div>
EDUCATION	<div>CTC 101</div>
WEARABLE	<div>GEMMA</div> <div>LILYPAD ARDUINO USB</div> <div>LILYPAD ARDUINO MAIN BOARD</div> <div>LILYPAD ARDUINO SIMPLE</div> <div>LILYPAD ARDUINO SIMPLE SNAP</div>
3D PRINTING	<div>MATERIA 101</div>

The Arduino UNO



- ATmega328P (AVR) - 16MHz
- Flash memory - 32 KB
- SRAM - 2 KB
- EEPROM - 1KB

14 Digital I/O pins (pins 0–13)

These pins can be either *inputs* or *outputs*. Inputs are used to read information from sensors, while outputs are used to control actuators. You will specify the direction (in or out) in the sketch you create in the IDE. Digital inputs can only read one of two values, and digital outputs can only output one of two values (HIGH and LOW).

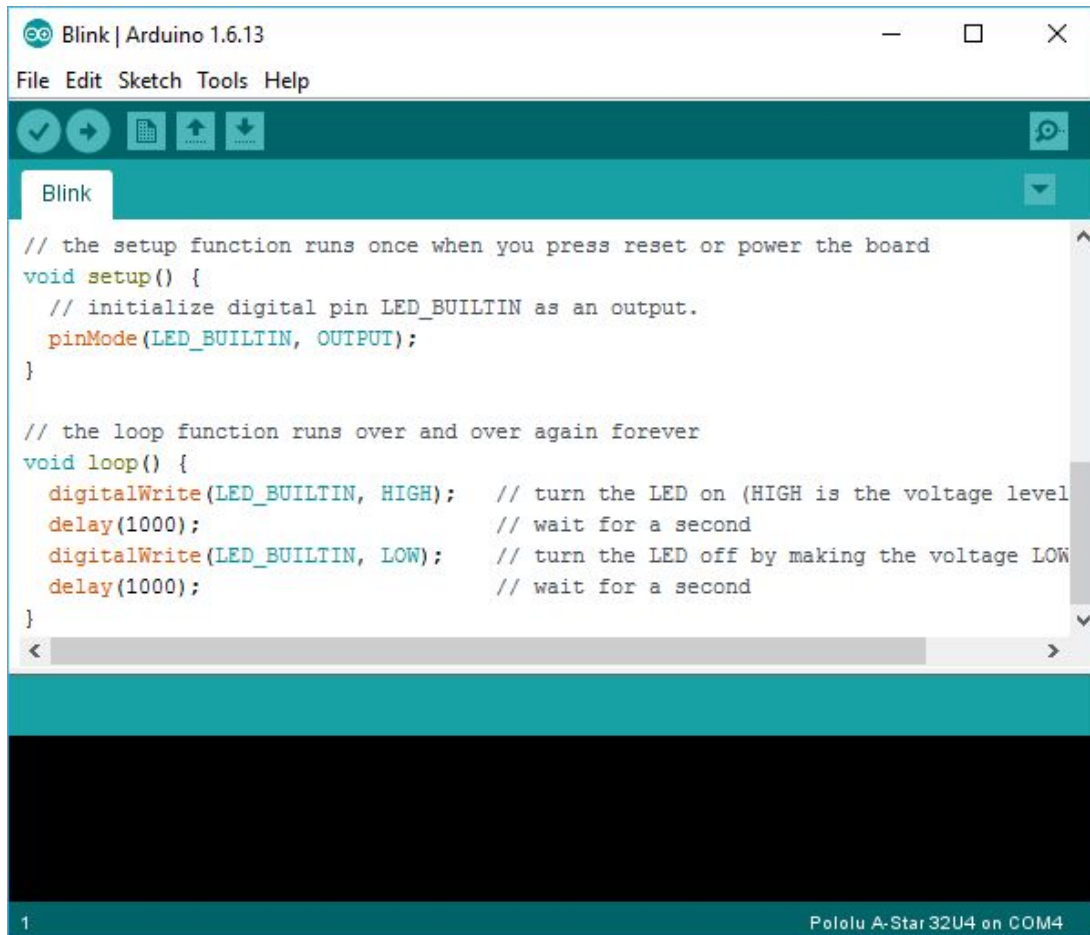
6 Analogue In pins (pins 0–5)

The analogue input pins are used for reading voltage measurements from analogue sensors. In contrast to digital inputs, which can distinguish between only two different levels (HIGH and LOW), analogue inputs can measure 1,024 different levels of voltage.

6 Analogue Out pins (pins 3, 5, 6, 9, 10, and 11)

These are actually six of the digital pins that can perform a third function: they can provide analogue output. As with the digital I/O pins, you specify what the pin should do in your sketch.

The Arduino IDE



The screenshot shows the Arduino IDE interface. The title bar reads "Blink | Arduino 1.6.13". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". The toolbar contains icons for opening, saving, and uploading. The sketch name "Blink" is displayed in the top left of the editor. The code is as follows:

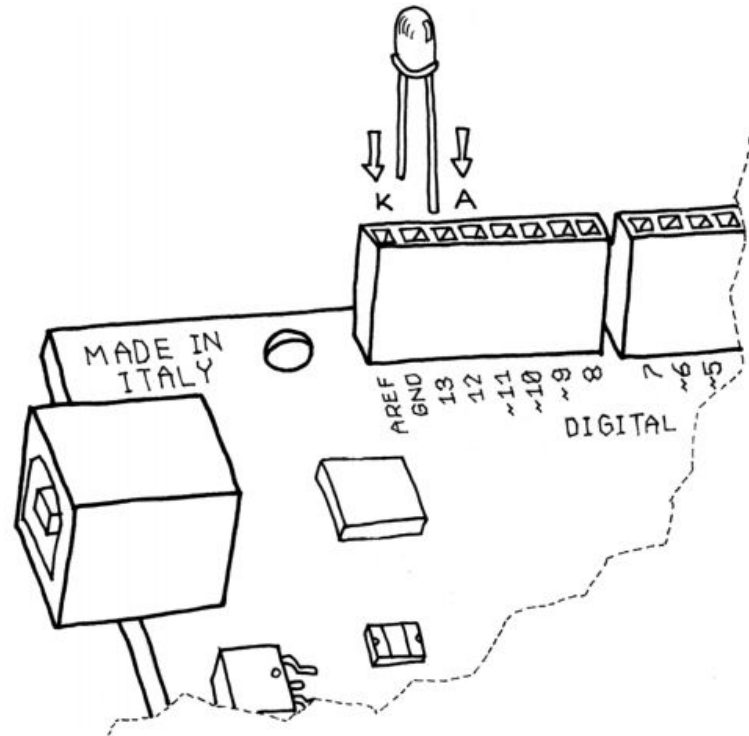
```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

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

The status bar at the bottom indicates "1" on the left and "Pololu A-Star 32U4 on COM4" on the right.

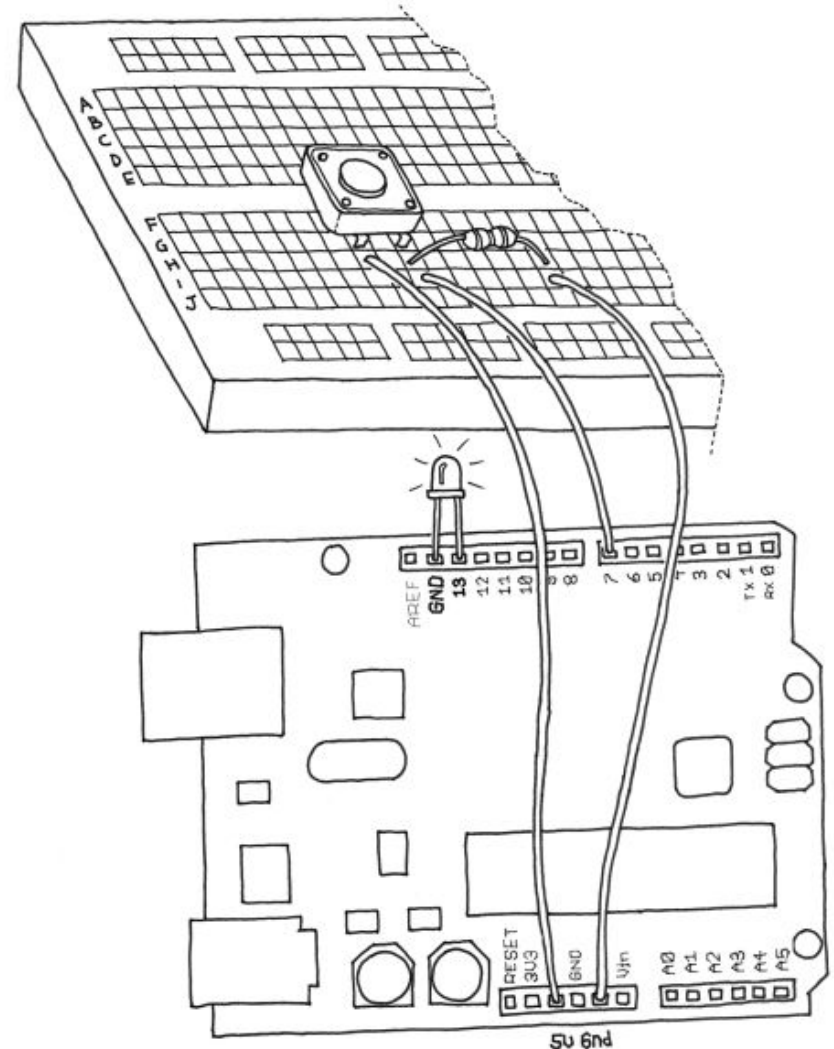
Blinking an LED

```
// Blinking LED
const int LED = 13; // LED connected to
// digital pin 13
void setup()
{
  pinMode(LED, OUTPUT); // sets the digital
  // pin as output
}
void loop()
{
  digitalWrite(LED, HIGH); // turns the LED on
  delay(1000); // waits for a second
  digitalWrite(LED, LOW); // turns the LED off
  delay(1000); // waits for a second
}
```



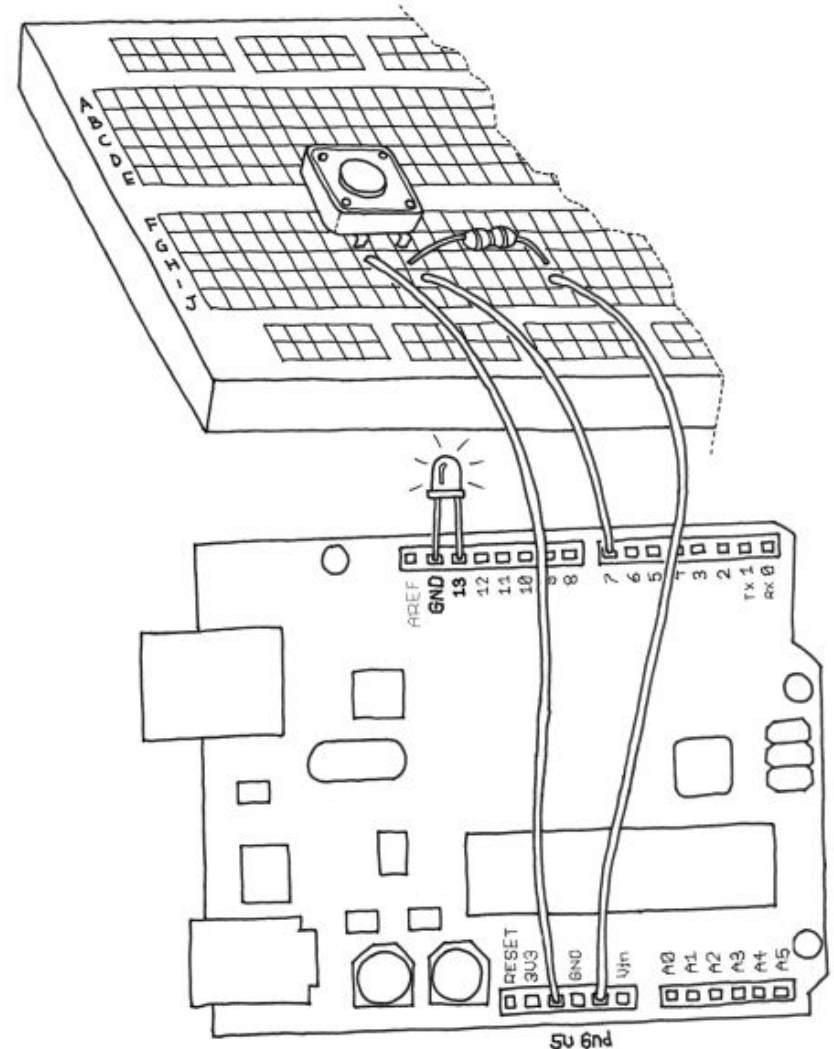
Turn on LED when the button is pressed and keep it on after it is released

```
const int LED = 13; // the pin for the LED
const int BUTTON = 7; // the input pin where the
// pushbutton is connected
int val = 0; // val will be used to store the state
// of the input pin
int state = 0; // 0 = LED off while 1 = LED on
void setup() {
  pinMode(LED, OUTPUT); // tell Arduino LED is an output
  pinMode(BUTTON, INPUT); // and BUTTON is an input
}
void loop() {
  val = digitalRead(BUTTON); // read input value and store it
  // check if the input is HIGH (button pressed)
  // and change the state
  if (val == HIGH) {
    state = 1 - state;
  }
  if (state == 1) {
    digitalWrite(LED, HIGH); // turn LED ON
  } else {
    digitalWrite(LED, LOW);
  }
}
```



New and improved button press formula!

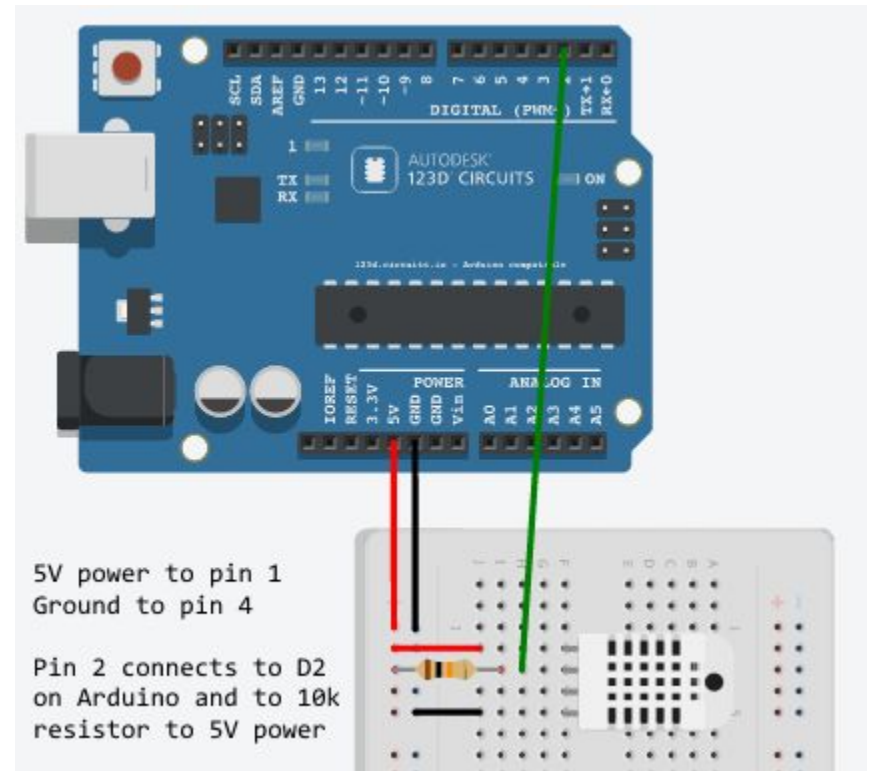
```
const int LED = 13; // the pin for the LED
const int BUTTON = 7; // the input pin where the
// pushbutton is connected
int val = 0; // val will be used to store the state
// of the input pin
int old_val = 0; // this variable stores the previous
// value of "val"
int state = 0; // 0 = LED off and 1 = LED on
void setup() {
  pinMode(LED, OUTPUT); // tell Arduino LED is an output
  pinMode(BUTTON, INPUT); // and BUTTON is an input
}
void loop(){
  val = digitalRead(BUTTON); // read input value and store it
  // yum, fresh
  // check if there was a transition
  if ((val == HIGH) && (old_val == LOW)){
    state = 1 - state;
  }
  old_val = val; // val is now old, let's store it
  if (state == 1) {
    digitalWrite(LED, HIGH); // turn LED ON
  } else {
    digitalWrite(LED, LOW);
  }
}
```




Temperature and Humidity Sensor


First [download](#) the DHT library by Lady Ada.

If your Arduino folder is in the default location, go to My Documents > Arduino > libraries, and make a folder named DHT. Place the DHT.cpp and DHT.h files in there.





```
#include "DHT.h"
// The DHT data line is connected to pin 2 on the Arduino
#define DHTPIN 2
// Leave as is if you're using the DHT22. Change if not.
// #define DHTTYPE DHT11 // DHT 11
#define DHTTYPE DHT22 // DHT 22 (AM2302)
// #define DHTTYPE DHT21 // DHT 21 (AM2301)
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  // Start the Serial Monitor and print a first line of text
  Serial.begin(9600);
  Serial.println("DHTxx test!");
  // Initialize the DHT library
  dht.begin();
}
void loop() {
  // Wait 2 seconds between measurements
  delay(2000);
  // Read humidity (percent)
  float h = dht.readHumidity();
  // Read temperature as Celsius
  float t = dht.readTemperature();
  // Read temperature as Fahrenheit
  float f = dht.readTemperature(true);
```



```
// Check if any reads failed and exit early (to try again)
if (isnan(h) || isnan(t) || isnan(f)) {
  Serial.println("Failed to read from DHT sensor :-(");
  return;
}

// Compute heat index
// Must send in temp in Fahrenheit!
float hi = dht.computeHeatIndex(f, h);
Serial.print("Humidity: ");
Serial.print(h);
Serial.print(" %\t");
Serial.print("Temperature: ");
Serial.print(t);
Serial.print(" *C ");
Serial.print(f);
Serial.print(" *F\t");
Serial.print("Heat index: ");
Serial.print(hi);
Serial.println(" *F");
}
```