# creativity
# & computation
# lab

week 7 || gettin’ physical: arduino

# review

WHERE WE HAVE BEEN

## What we have done:

- What IS electricity?!
- Voltage, resistance, and current
  - Ohm's law of course
- Breadboards
- Field Trip!
- Components
- Circuits
  - Parallel v. Series
- Switches

# agenda

WHERE WE ARE GOING

## What's on for today:

Review Ohm's Law + Intro Kirchoff's Law

What is a microcontroller?

Arduino

// the IDE

// the board

Digital vs. Analog

//INPUT = Switches + Variable resistors

//OUTPUT = PWM

Debugging

# last assignment

PRESENT

Working circuits!

Voltage calculations!

O my!

# ohm’s law

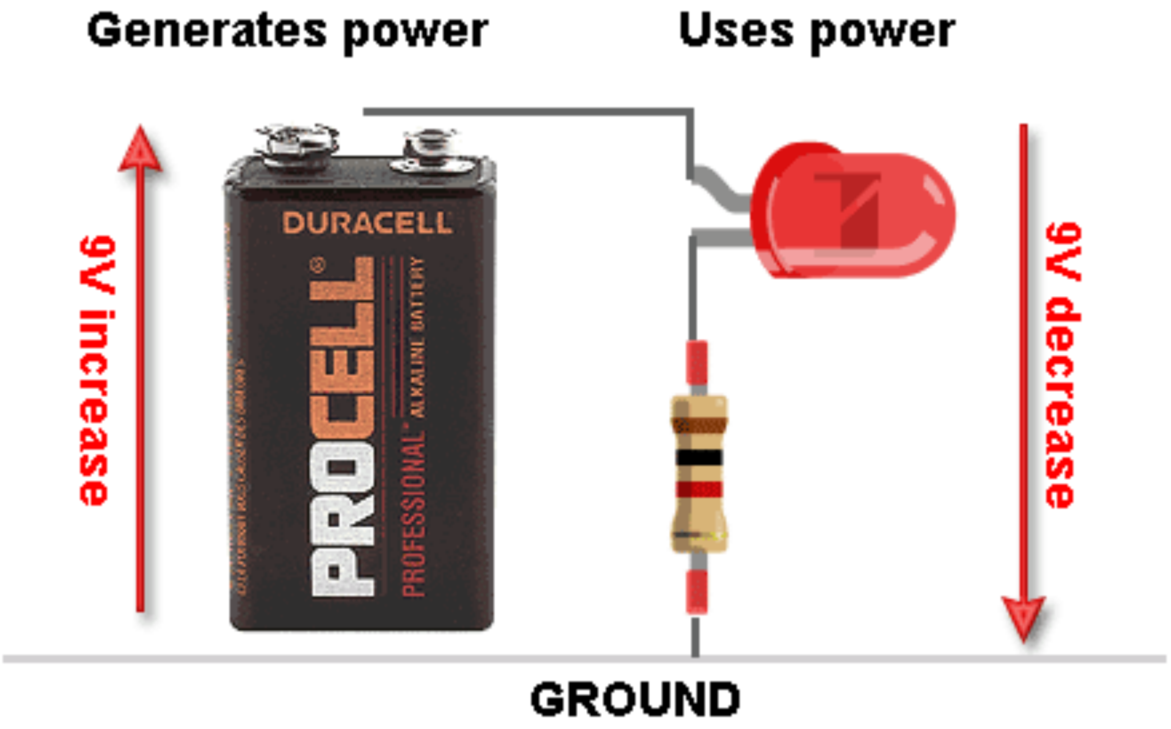
REVIEW

Let’s step back a little bit and dissect our calculations.

# kirchoff’s voltage law

LOOPS!

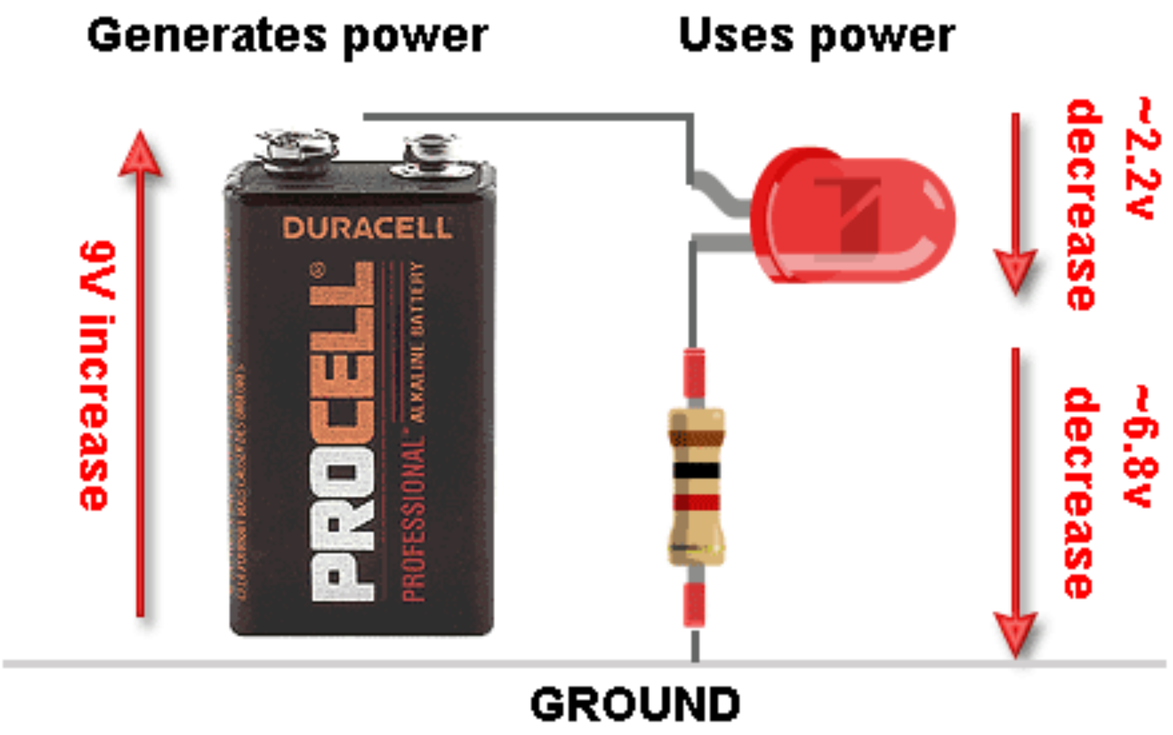
In any 'loop' of a circuit, the voltages must balance: the amount generated = the amount used



# kirchoff's voltage law

LOOPS!

Let's break it down:



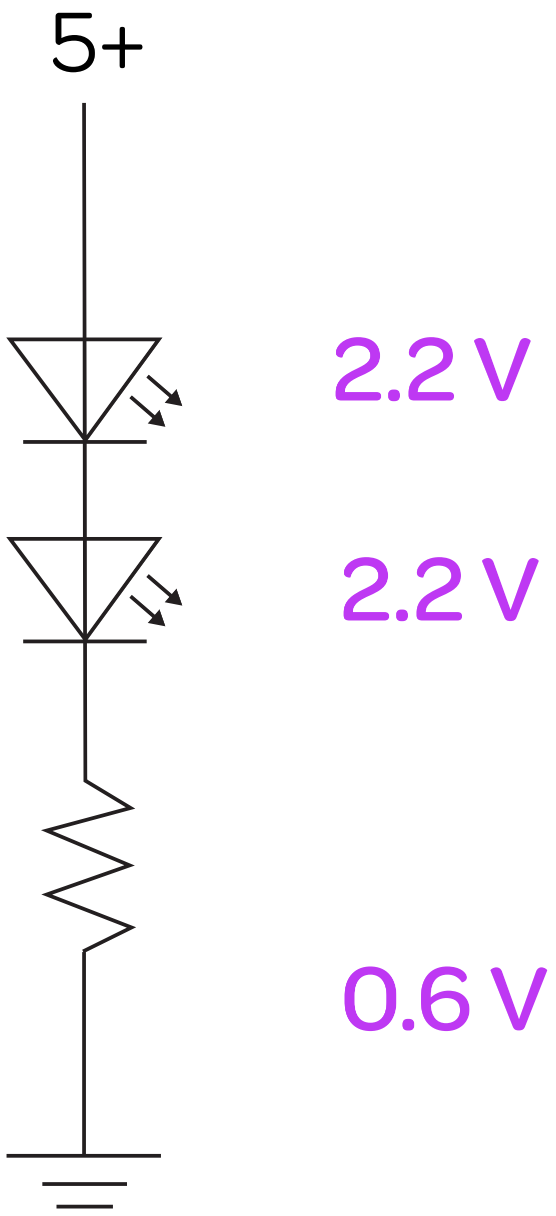
Forward Voltage ($V_F$)
= voltage lost to the LED (or other component)

# kvl + ohm

AN EXAMPLE

We want to know the resistance to we need to have the LEDs running at full brightness.

$$R = V/I$$

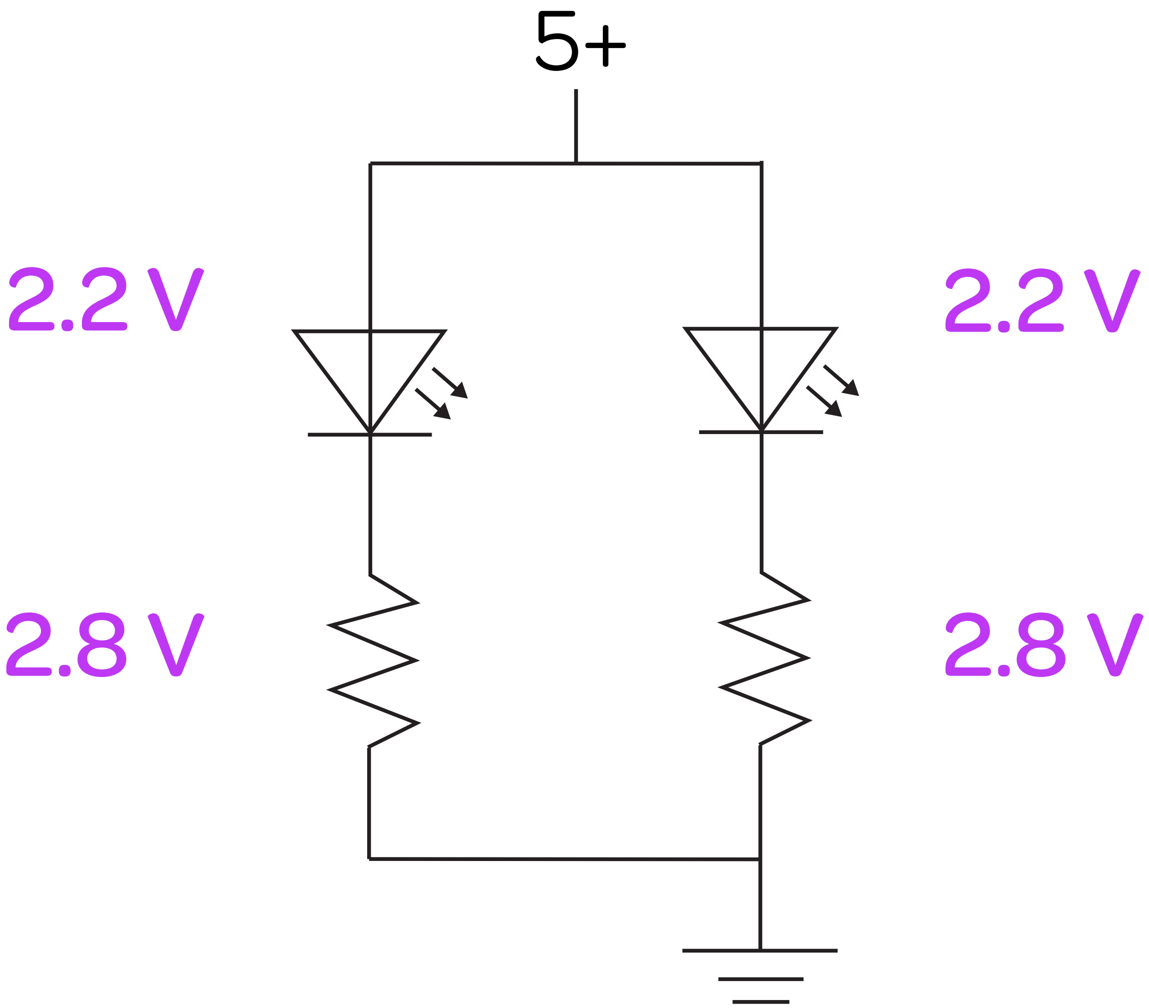


# kvl + ohm

## AN EXAMPLE

We want to know the current to know how bright the LED will be.

$$I = V/R$$

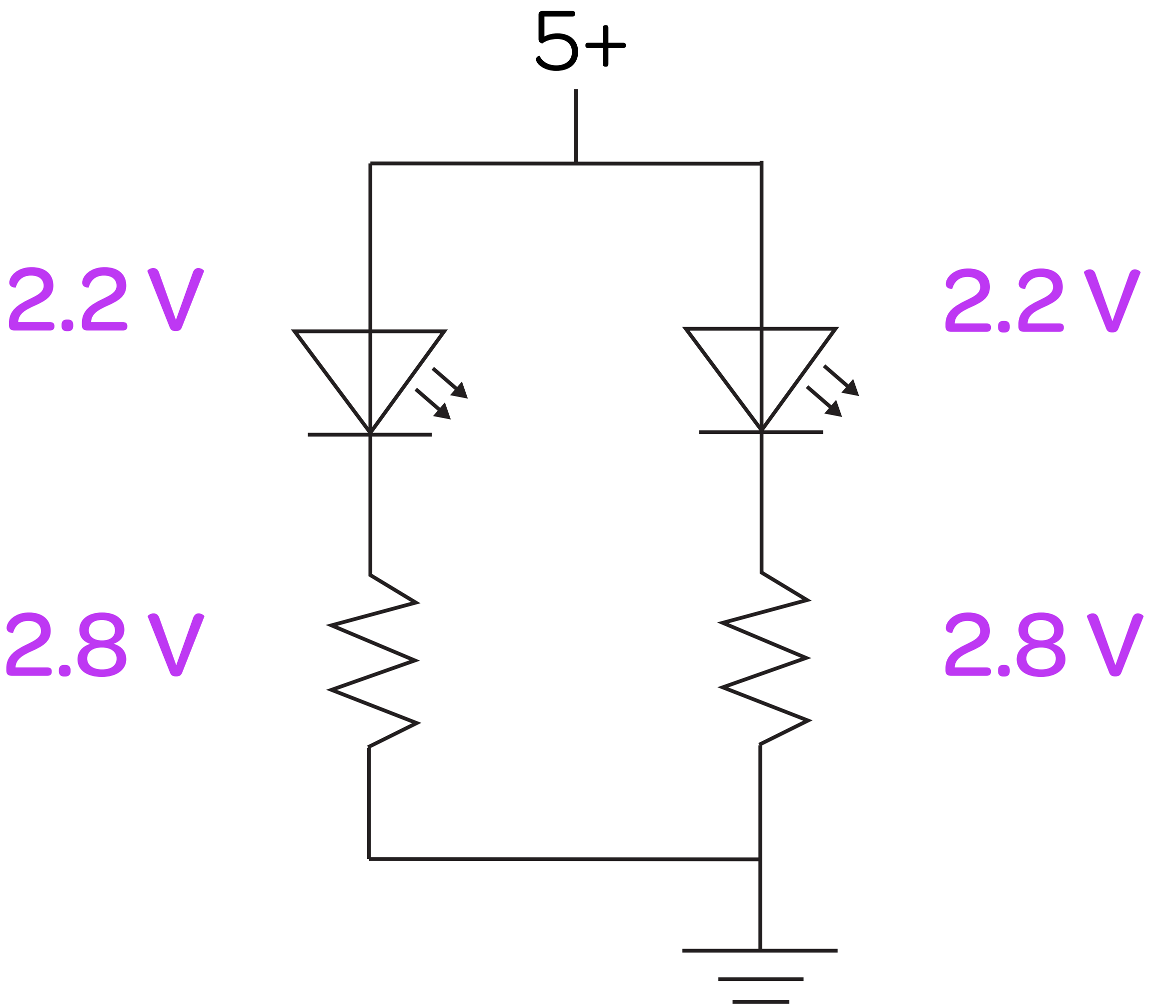


# kvl + ohm

## AN EXAMPLE

We want to know the current to know how bright the LED will be.

$$I = 2.8/R$$

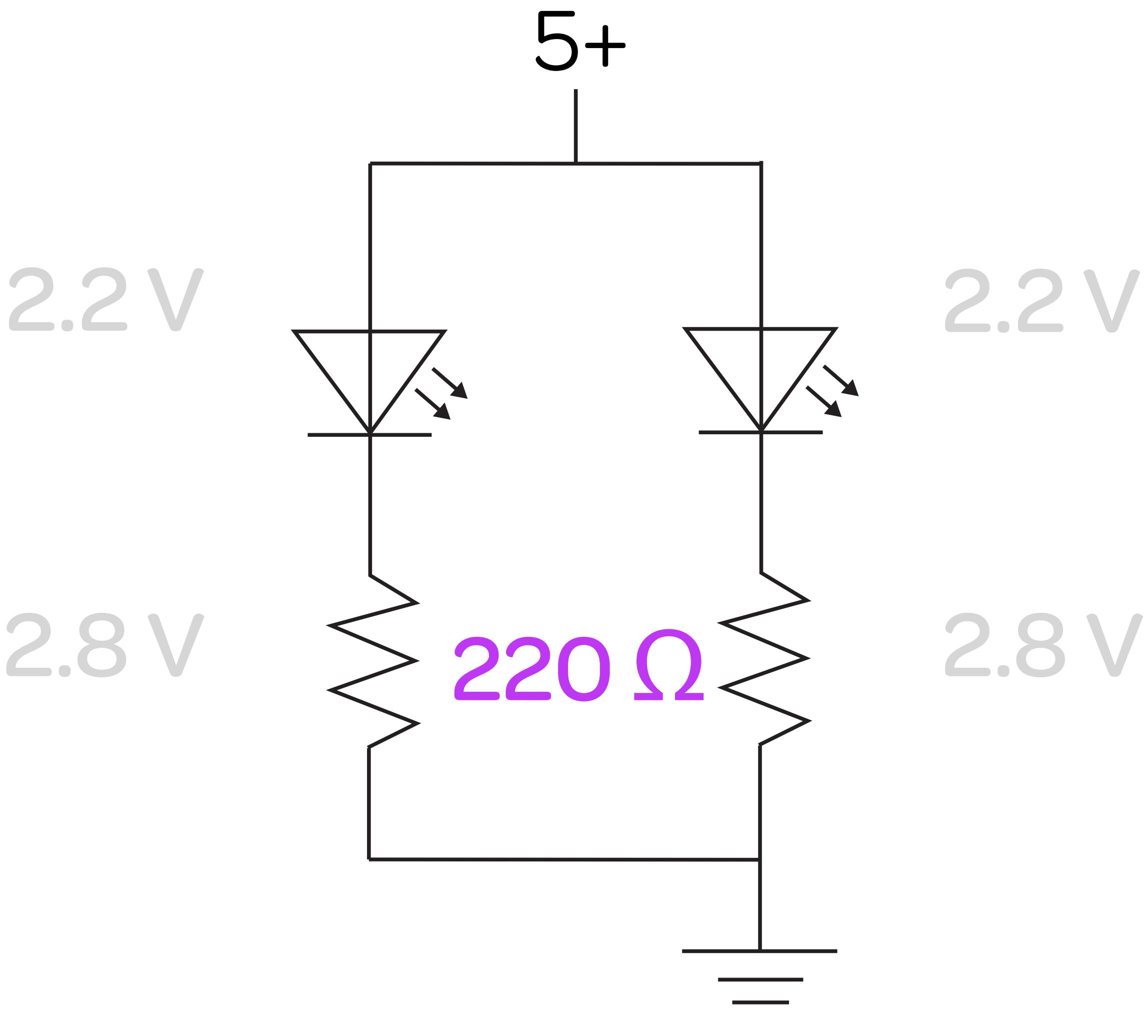


# kvl + ohm

AN EXAMPLE

We want to know the current to know how bright the LED will be.

I = 2.8/220



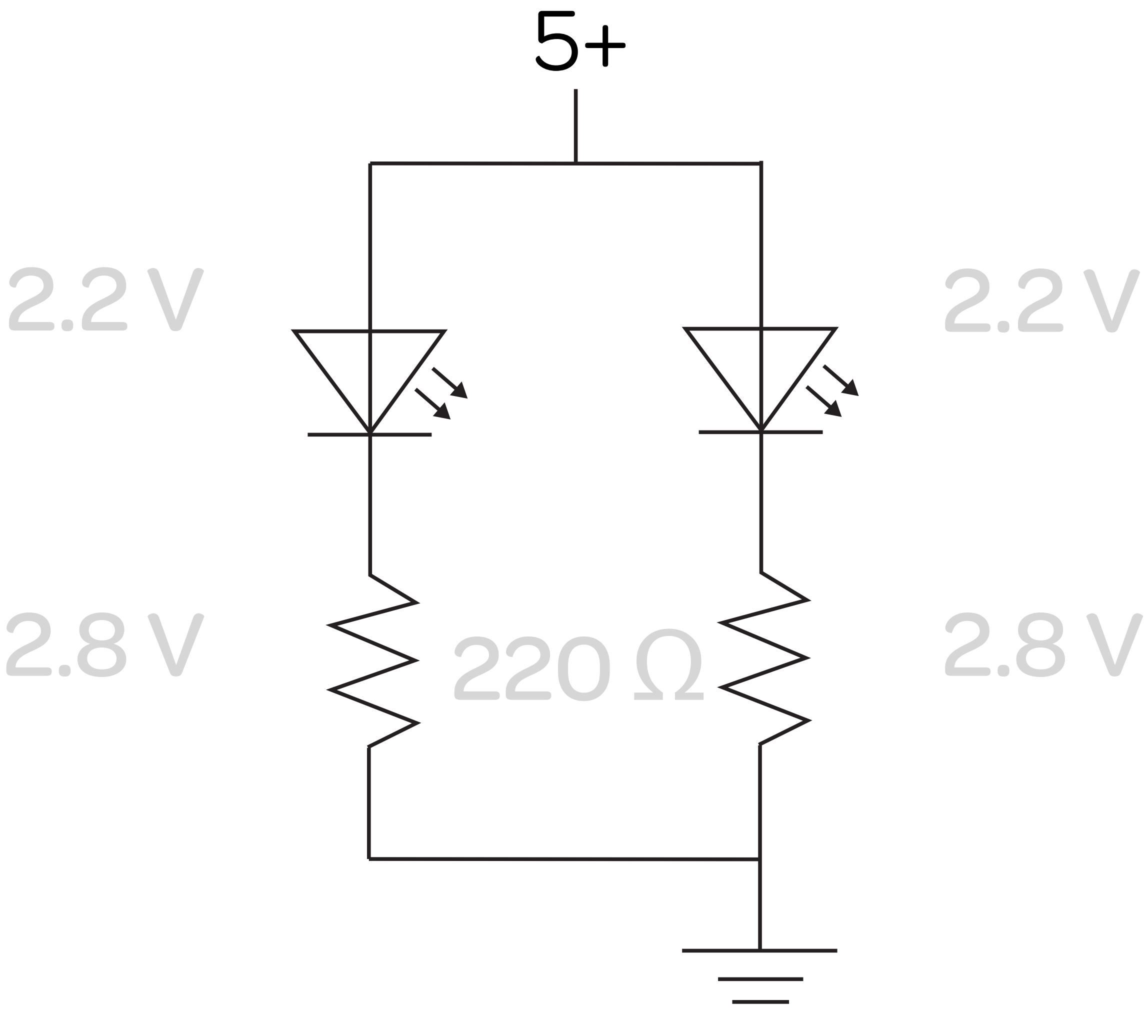
# kvl + ohm

AN EXAMPLE

We want to know the current to know how bright the LED will be.

0.0127 = 2.8/220

12.7 mA

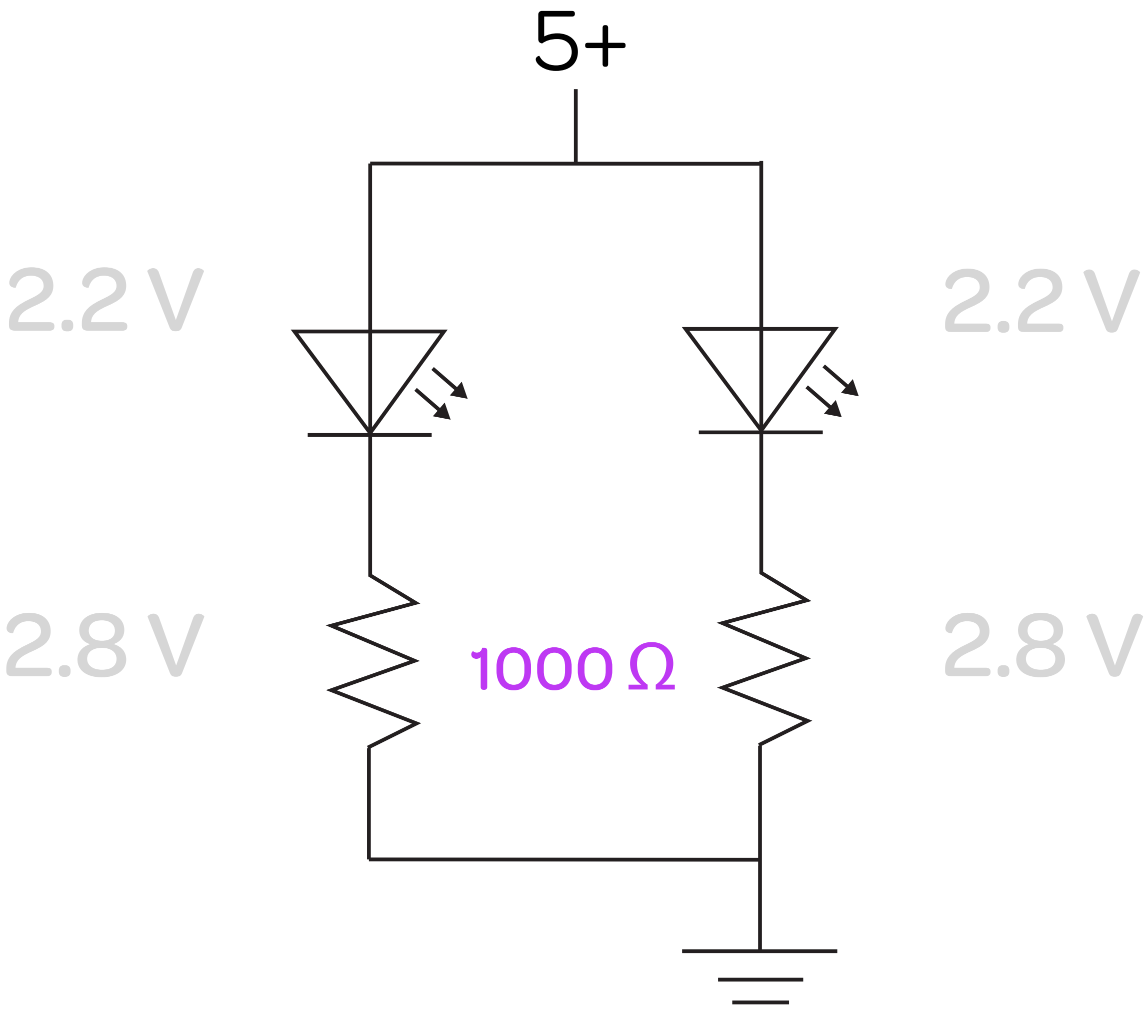


# kvl + ohm

AN EXAMPLE

AGAIN! This time with 1K!

$$I = 2.8/1000$$



# kvl + ohm

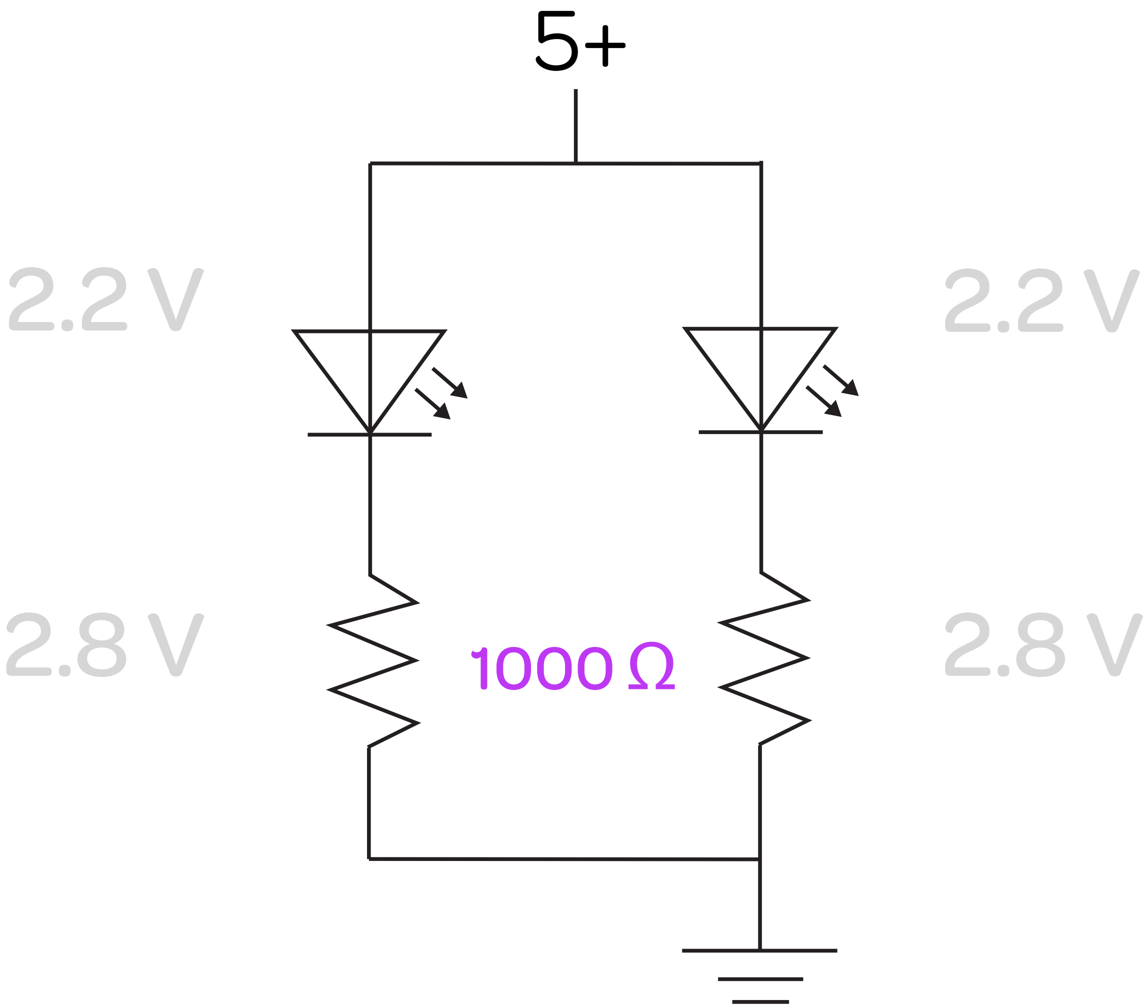
AN EXAMPLE

AGAIN! This time with 1K!

I = 2.8/1000

2.8 mA

So which is brighter?

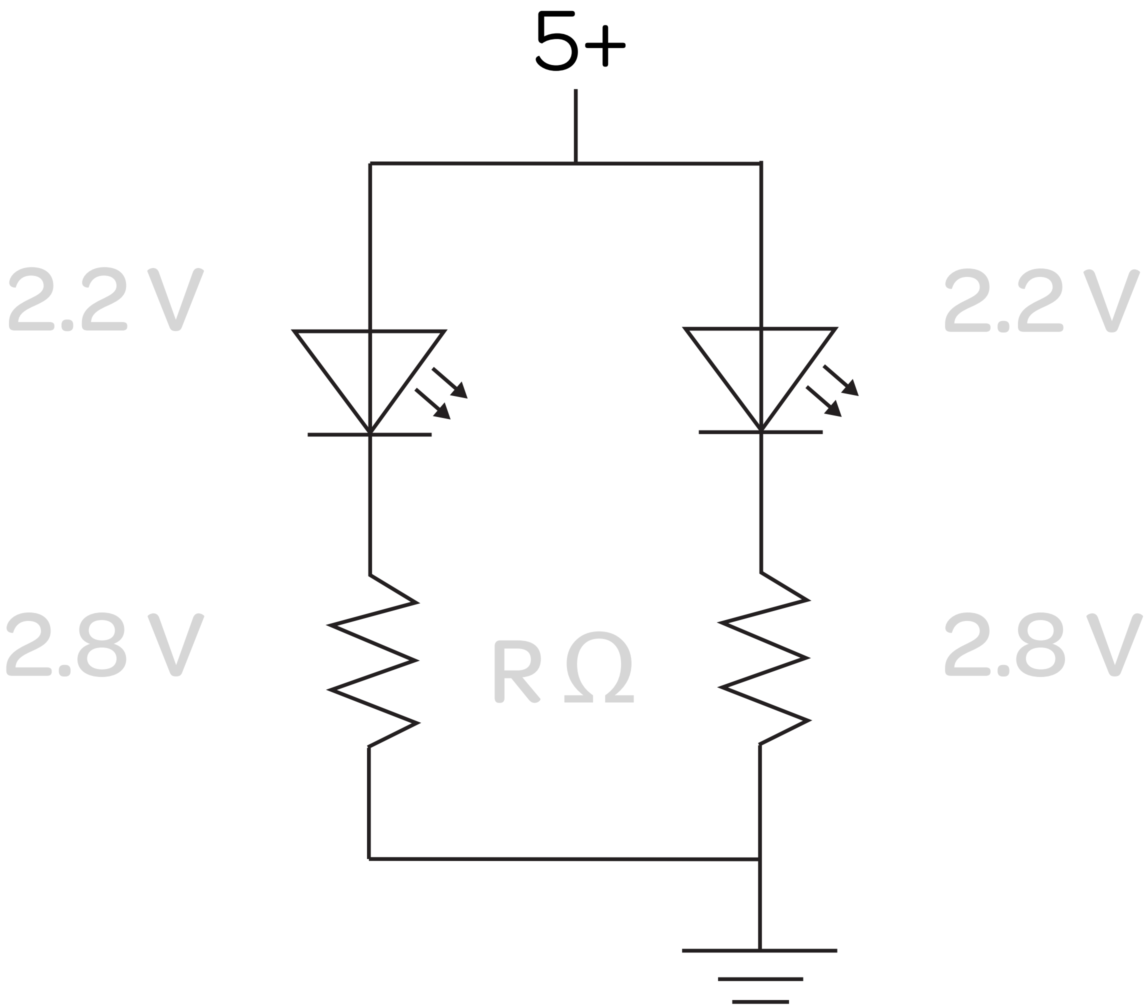


# kvl + ohm

ANOTHER EXAMPLE

Let’s find the least valued resistor we can use without burning our LED.

$R = V/I$



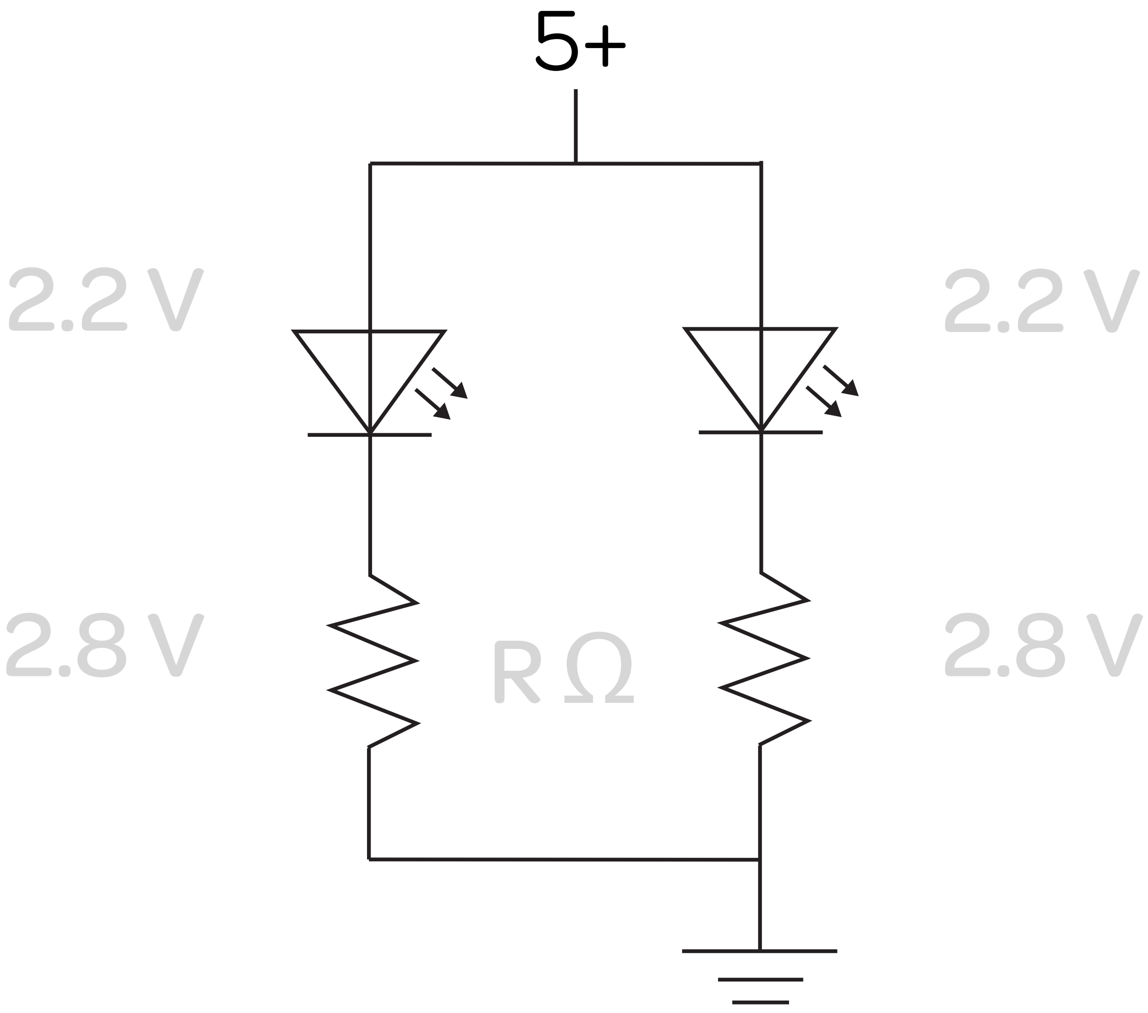
# kvl + ohm

ANOTHER EXAMPLE

Let’s find the least valued resistor we can use without burning our LED.

$$R = V/I$$

$$R = 2.8 / 0.02$$



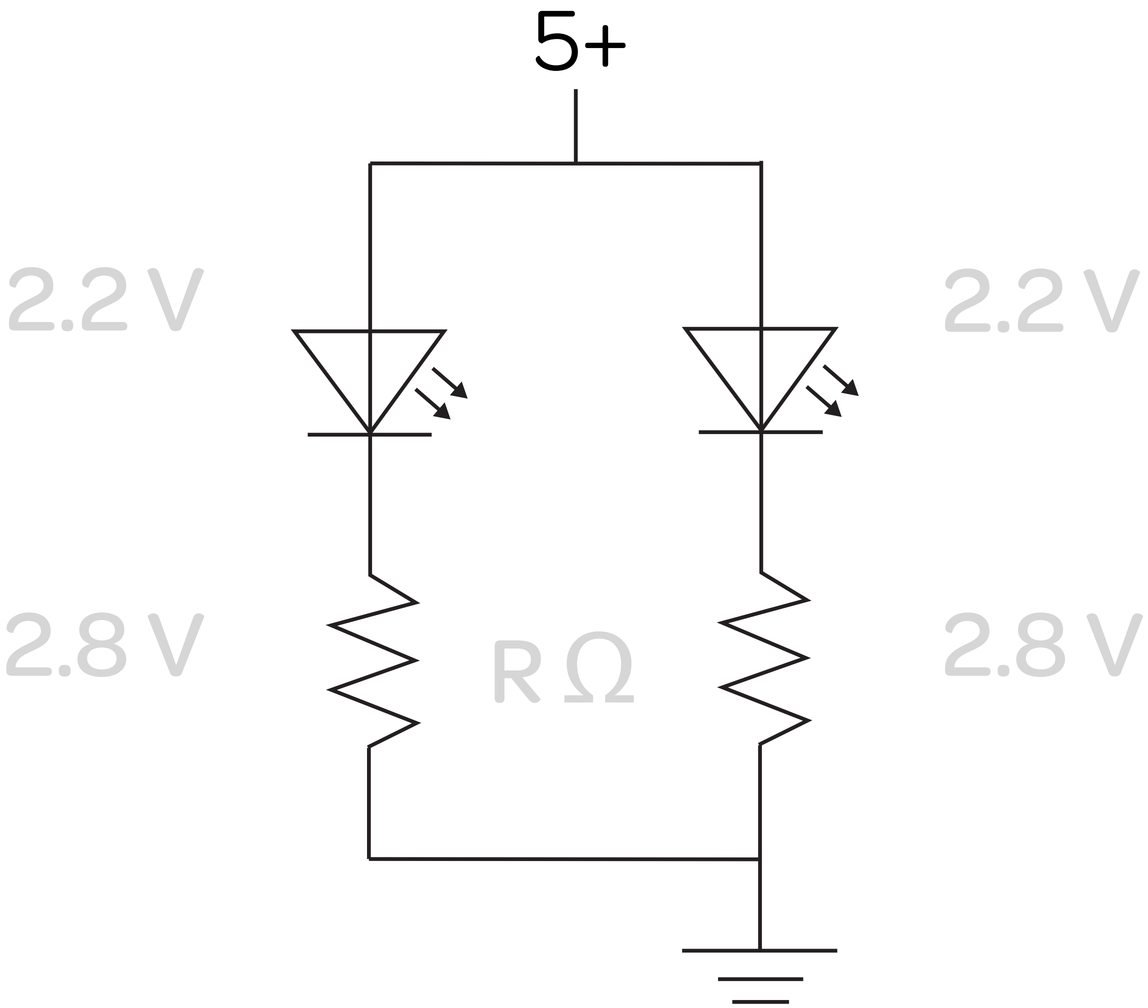
# kvl + ohm

ANOTHER EXAMPLE

Let's find the least valued resistor we can use without burning our LED.

$R = V/I$

$R = 140\ \Omega$



# microcontrollers

THEY'RE JUST LITTLE COMPUTERS

## Microcontroller = mini-computer

IC (integrated circuit) with a processor, memory, and programmable input/output.

Every day objects embedded with them to controller behaviors.

Not usually reprogrammable because developed for one use.

# the arduino

TA DAAAA!!

## An open-source electronics prototyping platform.

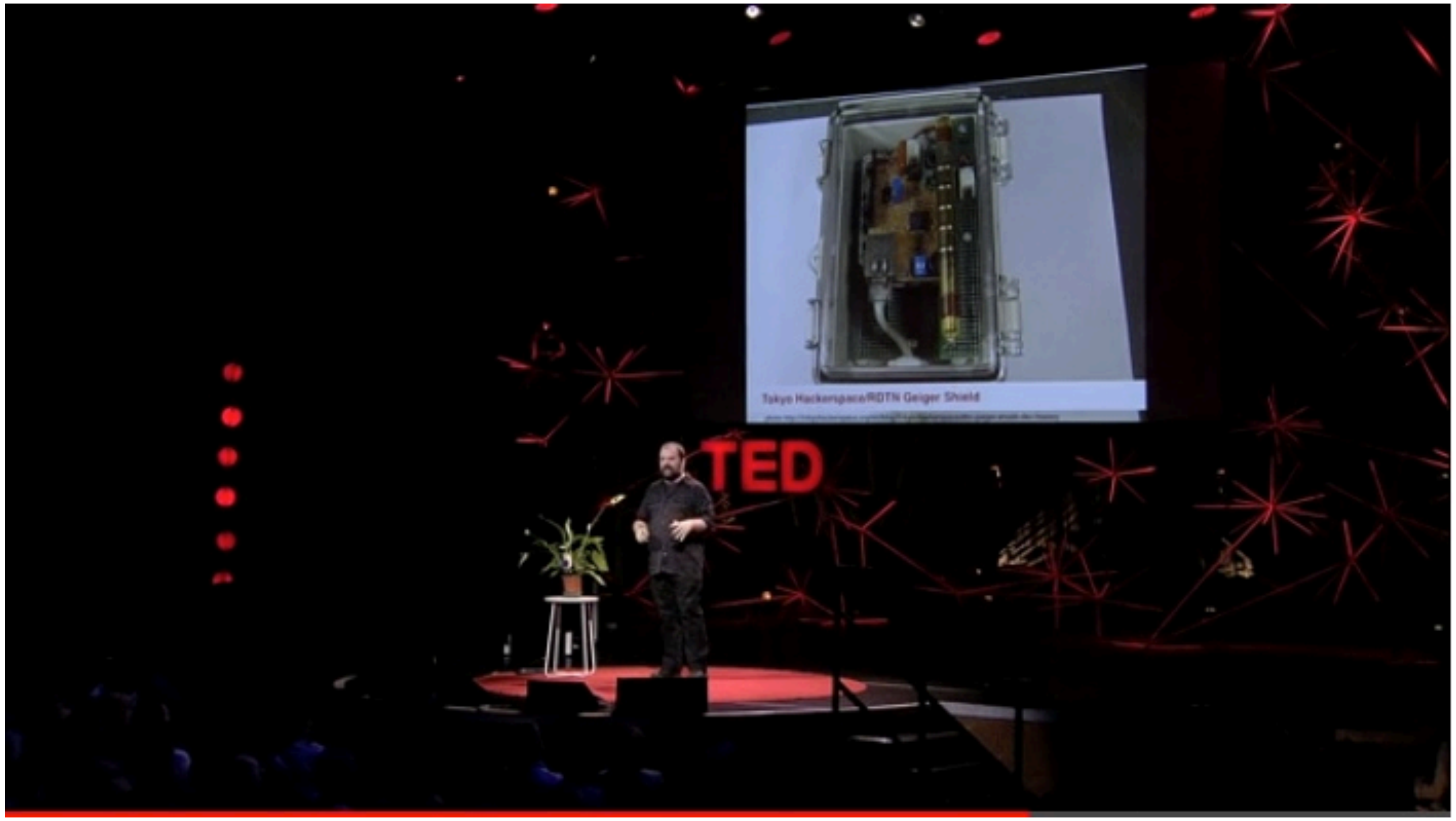
Hardware + software

Makes our lives SO much easier!

# the arduino

MASSIMO BANZI'S TED TALK

## What did you think?



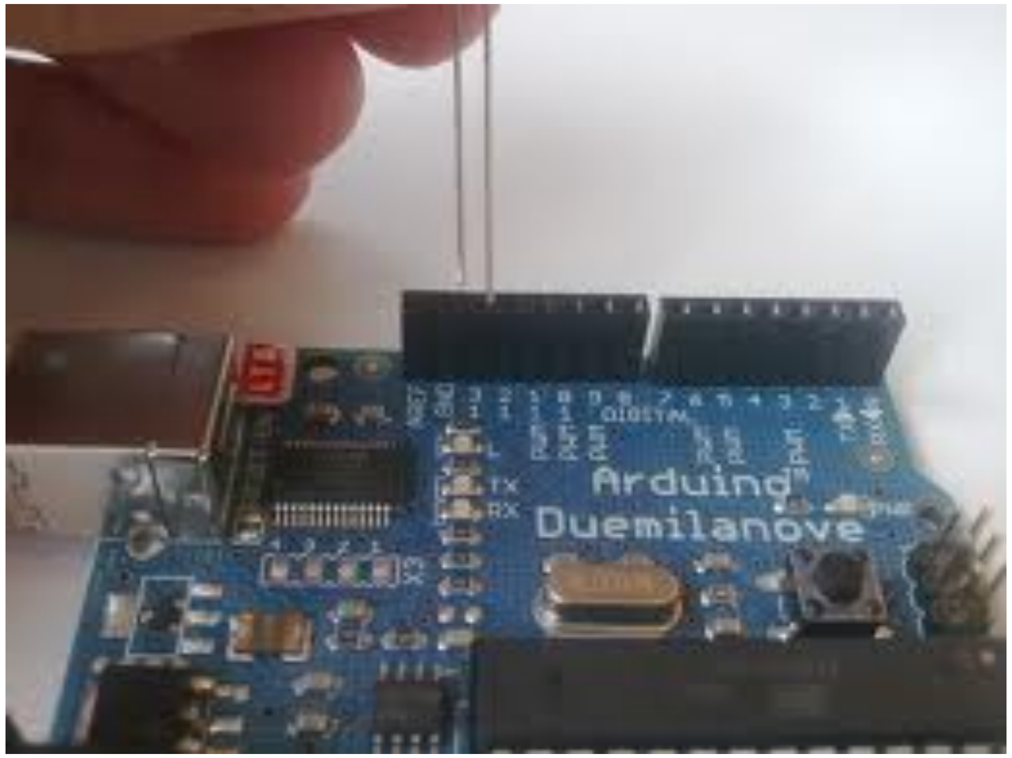
# hello world

YOUR FIRST PROGRAM

## Let's make a light blink.

Put the positive end of the LED into pin 13 and the short end into ground.

You can only do this on pin 13!!
//There is a resistor enabled
on pin 13



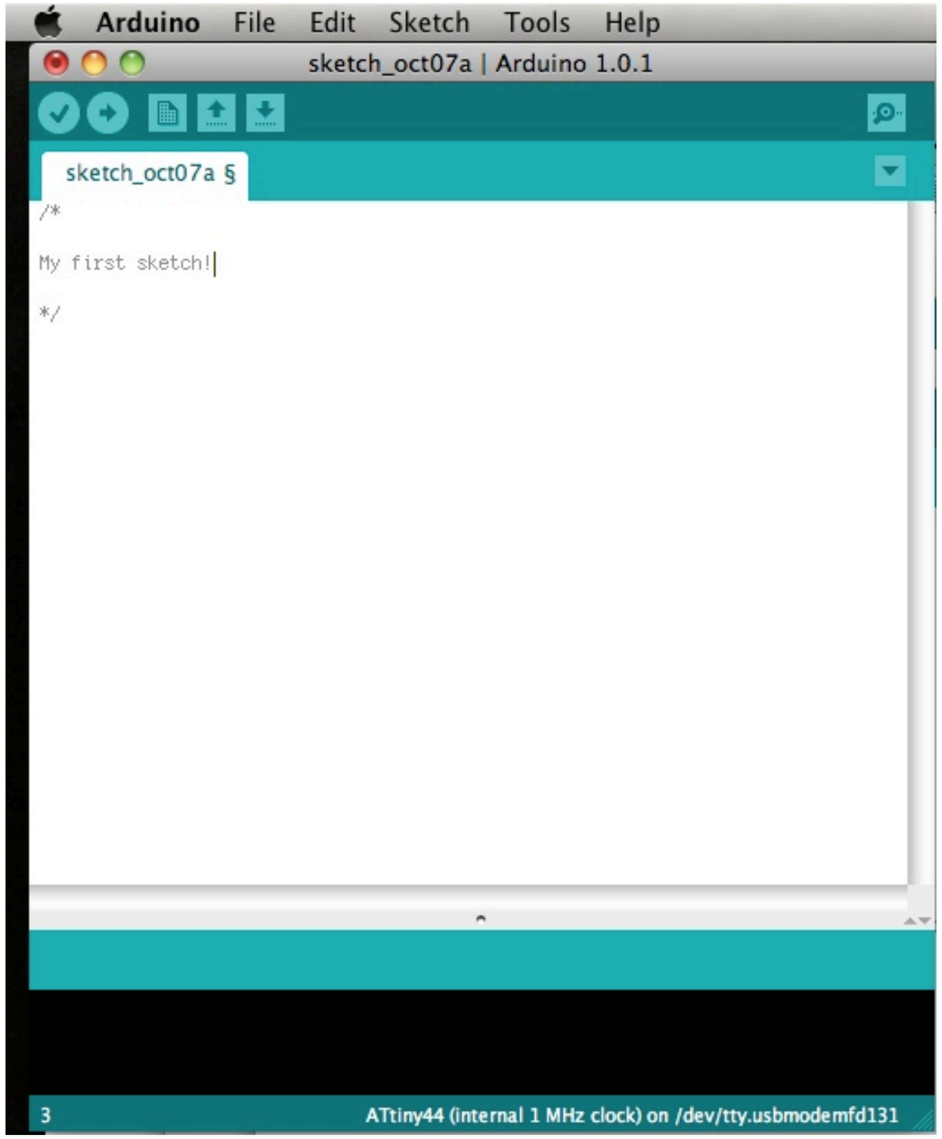
# hello world

YOUR FIRST PROGRAM

## Let’s make a light blink.

STEP 0: Plug your Arduino into your computer.

Open the Arduino IDE.



```
/*

My first sketch!

*/
```

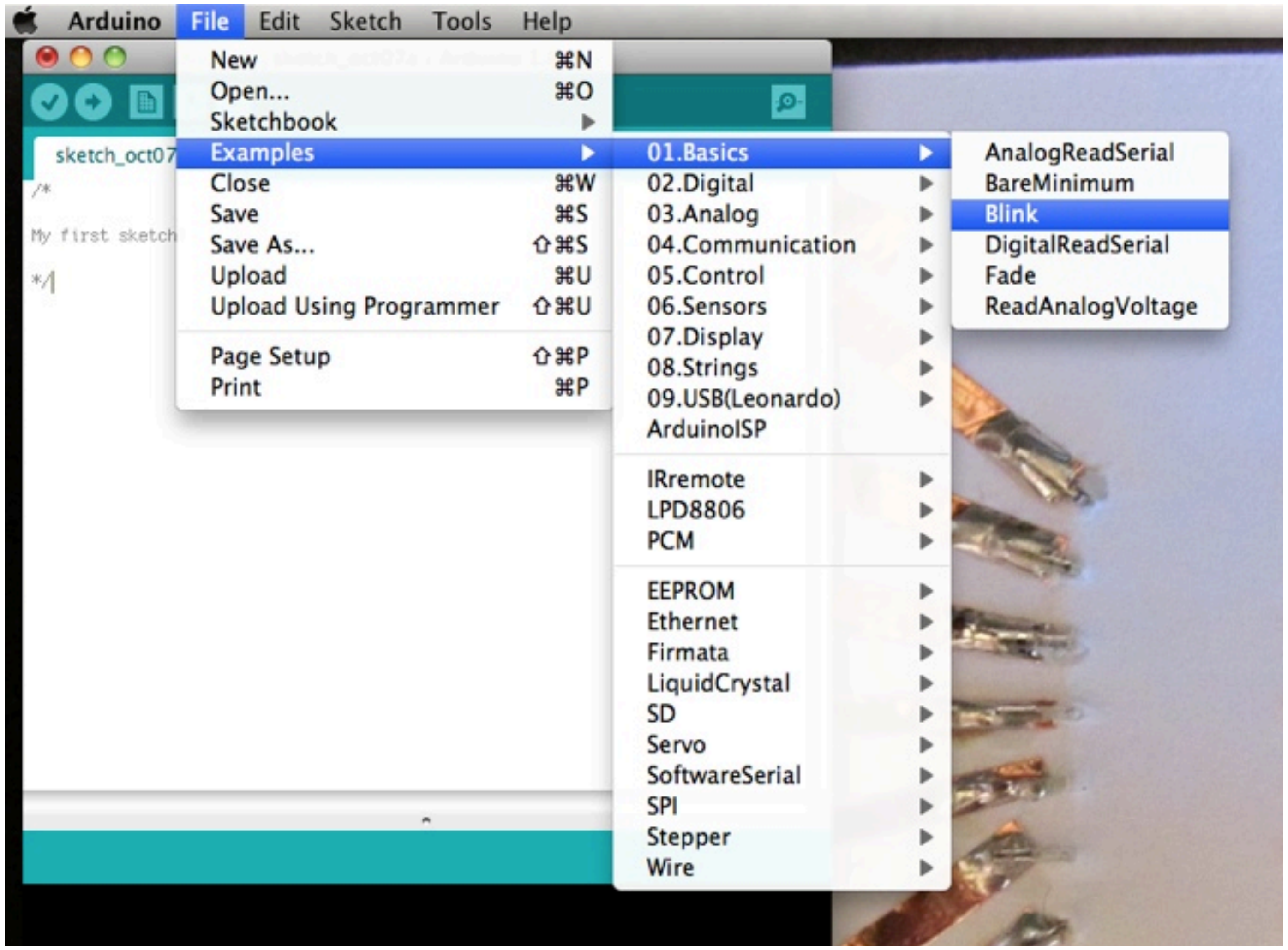
3 ATtiny44 (internal 1 MHz clock) on /dev/tty.usbmodemfd131

# hello world

YOUR FIRST PROGRAM

## Let’s make a light blink.

STEP 1:
Open the sketch you want to upload.
File >> Examples
>> Basics >> Blink



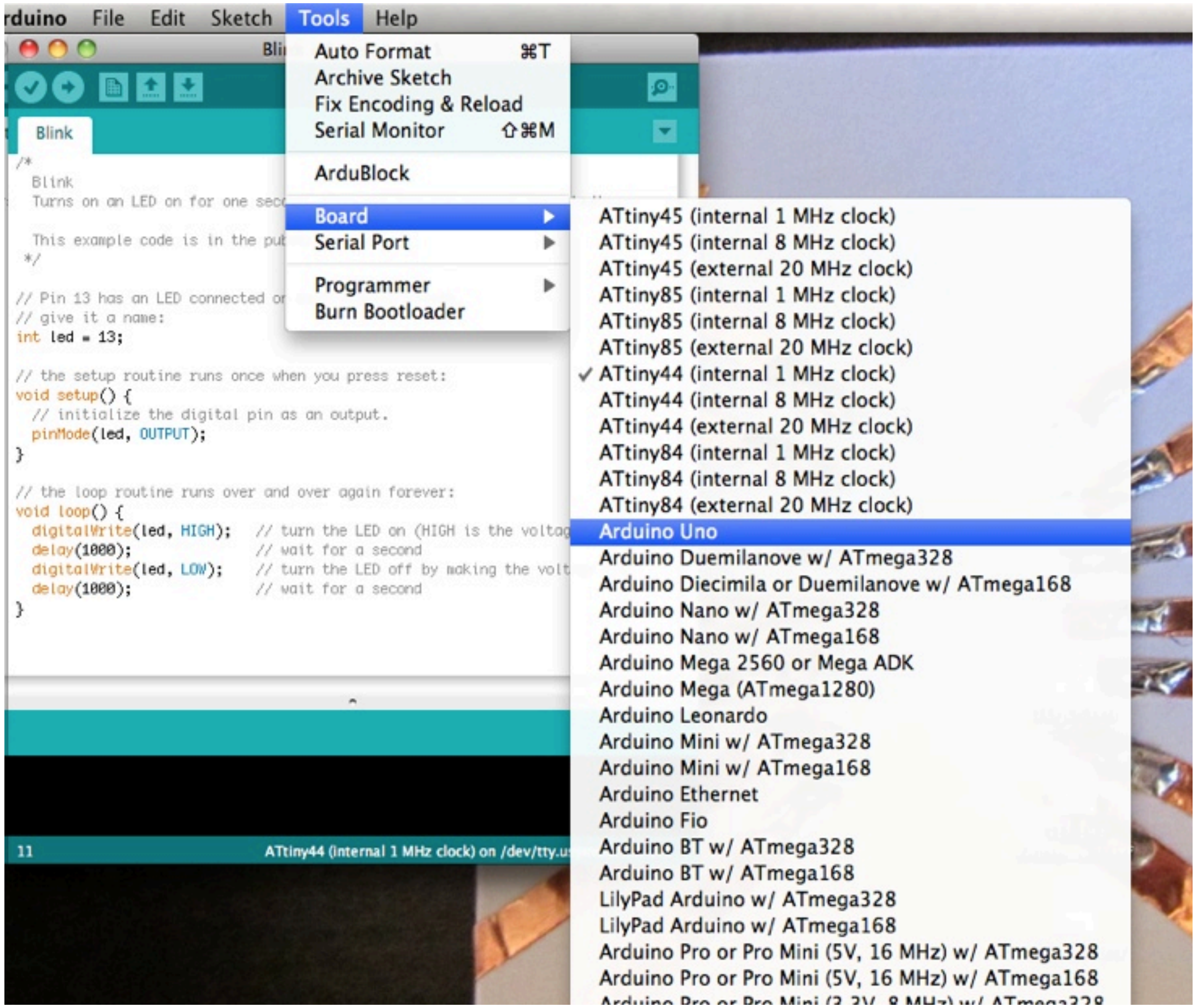
# hello world

YOUR FIRST PROGRAM

## Let’s make a light blink.

STEP 2:
Make sure Arduino knows the board you are using.

Tools >> Board
>> Arduino Uno


rduino
File
Edit
Sketch
Tools
Help
Bli
Blink
Auto Format ⌘T
Archive Sketch
Fix Encoding & Reload
Serial Monitor ⇧⌘M
ArduBlock
Board ▶
Serial Port ▶
Programmer ▶
Burn Bootloader
ATtiny45 (internal 1 MHz clock)
ATtiny45 (internal 8 MHz clock)
ATtiny45 (external 20 MHz clock)
ATtiny85 (internal 1 MHz clock)
ATtiny85 (internal 8 MHz clock)
ATtiny85 (external 20 MHz clock)
✓ ATtiny44 (internal 1 MHz clock)
ATtiny44 (internal 8 MHz clock)
ATtiny44 (external 20 MHz clock)
ATtiny84 (internal 1 MHz clock)
ATtiny84 (internal 8 MHz clock)
ATtiny84 (external 20 MHz clock)
Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Leonardo
Arduino Mini w/ ATmega328
Arduino Mini w/ ATmega168
Arduino Ethernet
Arduino Fio
Arduino BT w/ ATmega328
Arduino BT w/ ATmega168
LilyPad Arduino w/ ATmega328
LilyPad Arduino w/ ATmega168
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega168
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328

```
/*
  Blink
  Turns on an LED on for one seco
 
  This example code is in the pub
 */

// Pin 13 has an LED connected or
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltag
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the volt
  delay(1000);               // wait for a second
}
```

11 ATtiny44 (internal 1 MHz clock) on /dev/tty.u

# hello world

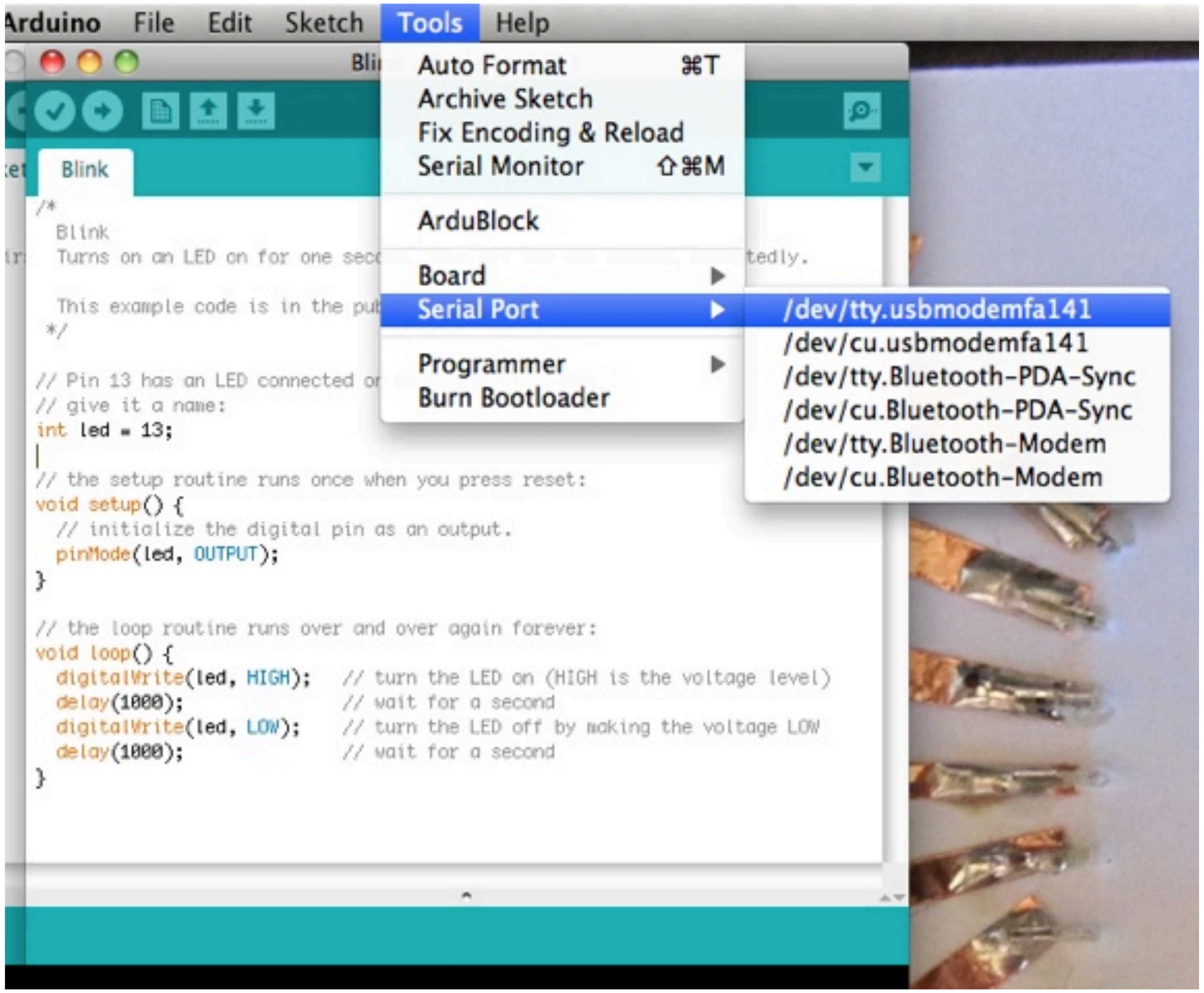
YOUR FIRST PROGRAM

## Let’s make a light blink.

STEP 3:
Make sure Arduino knows the Serial port you are using.

Tools >> Serial Port >> /dev/tty.usbmodemmfa141
(or something that looks like that!)

//Don’t worry about knowing what this means just yet.



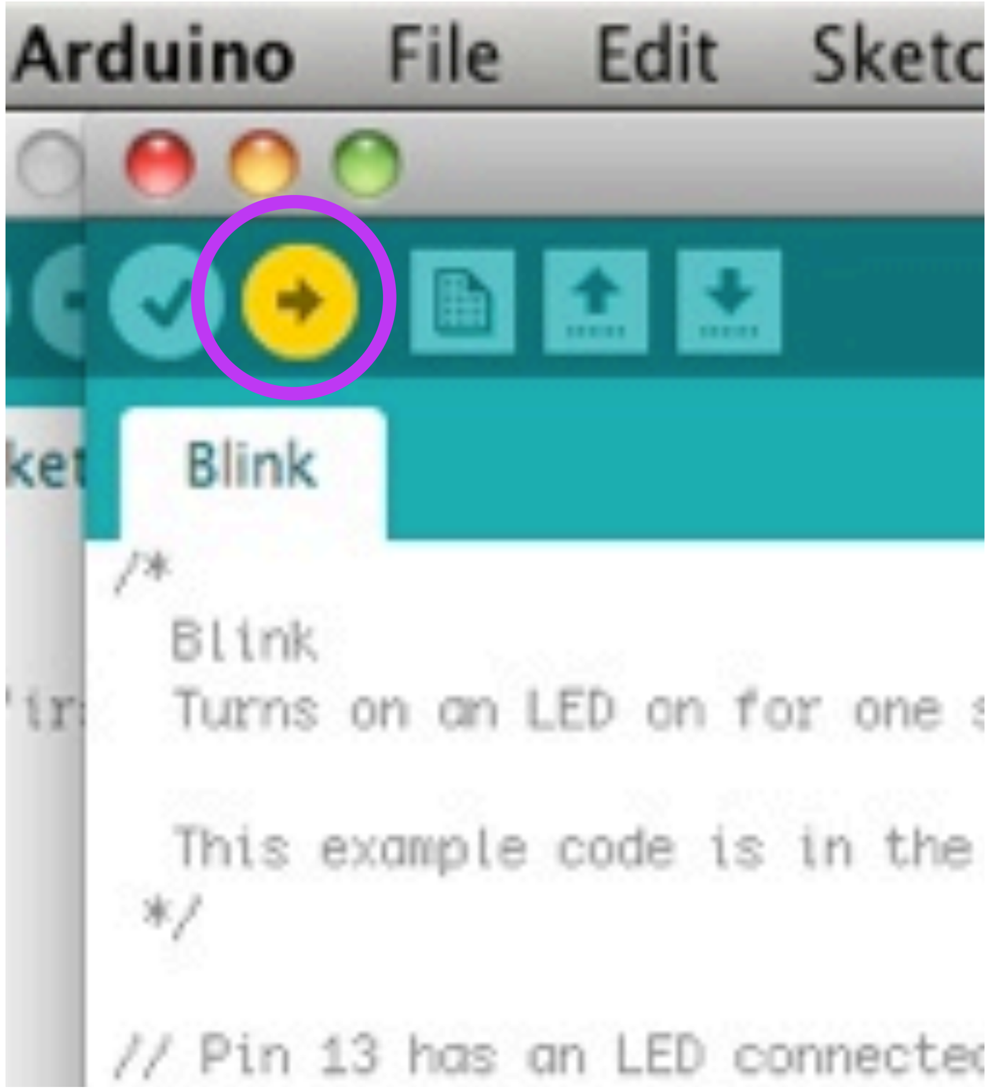
```
/*
  Blink
  Turns on an LED on for one seco                          tedly.

  This example code is in the pu
 */

// Pin 13 has an LED connected o
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

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

# hello world

YOUR FIRST PROGRAM

## Let’s make a light blink.

STEP 4:
Upload the sketch onto
the Arduino.



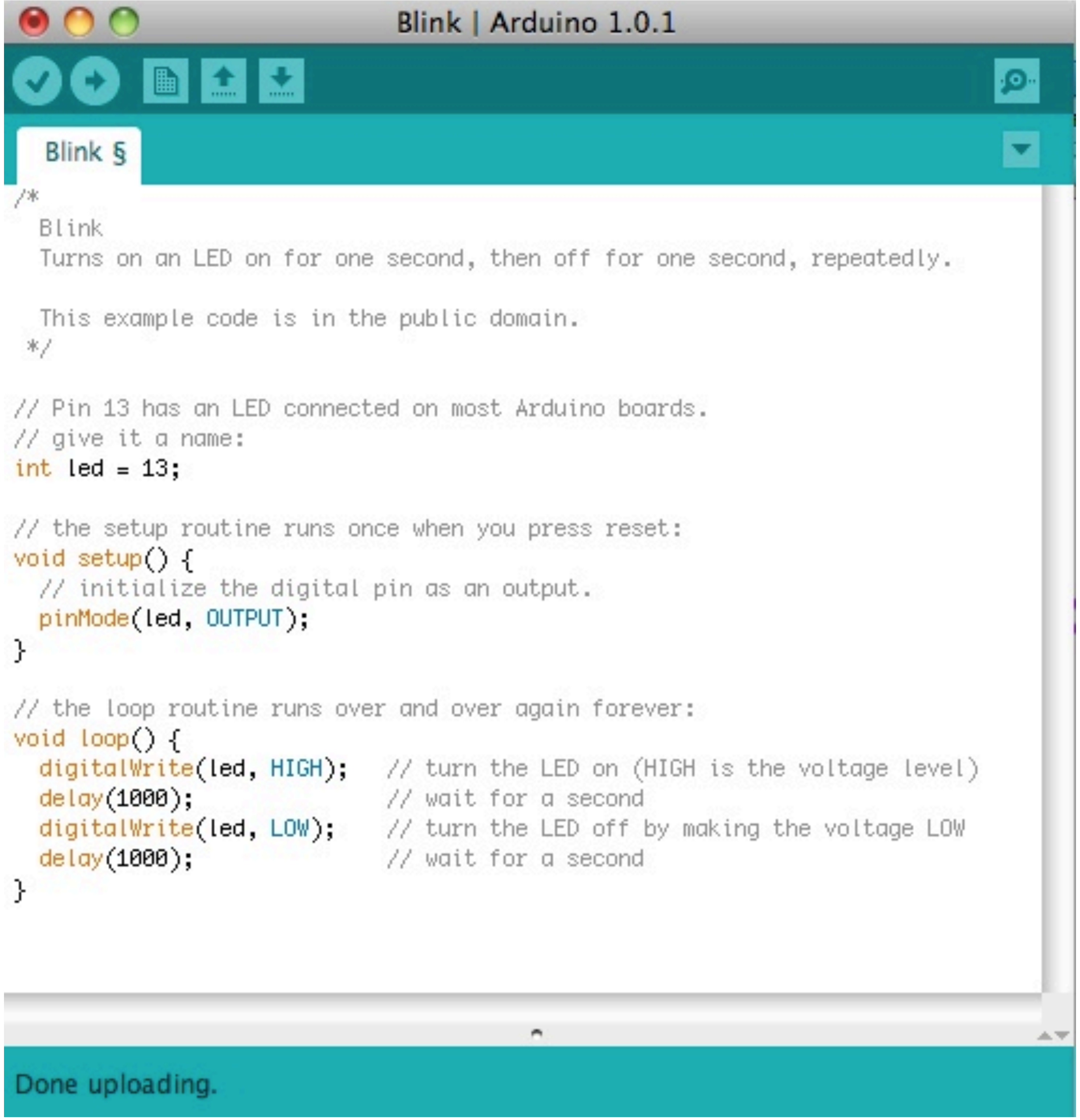
```
/*
  Blink
  Turns on an LED on for one s

  This example code is in the
 */

// Pin 13 has an LED connecte
```

# hello world

YOUR FIRST PROGRAM

## Let’s make a light blink.

STEP 5:
What happens?



```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

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

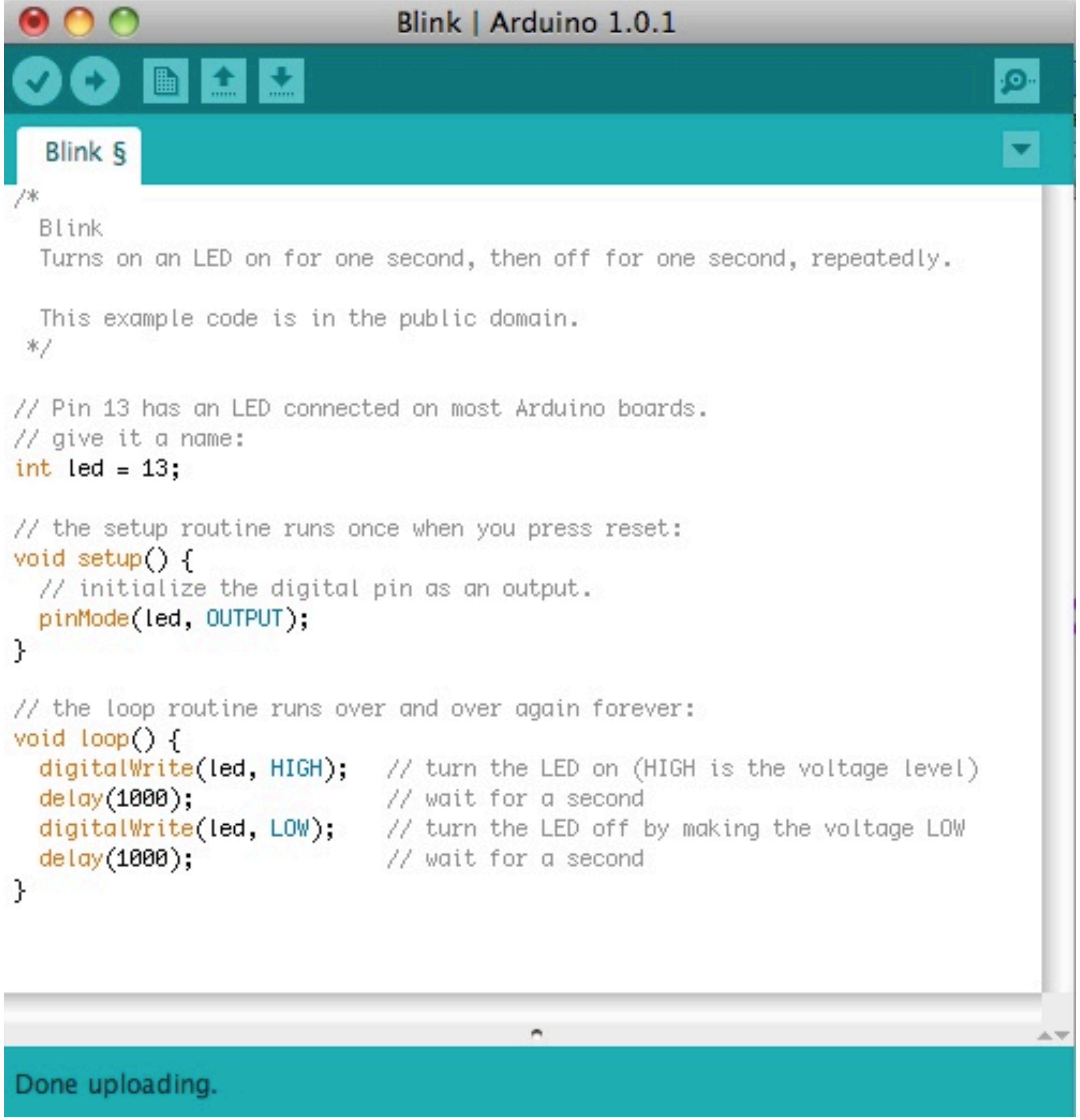
Done uploading.

# hello world

YOUR FIRST PROGRAM

## Let's make a light blink.

STEP 5:
What happens?



```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

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

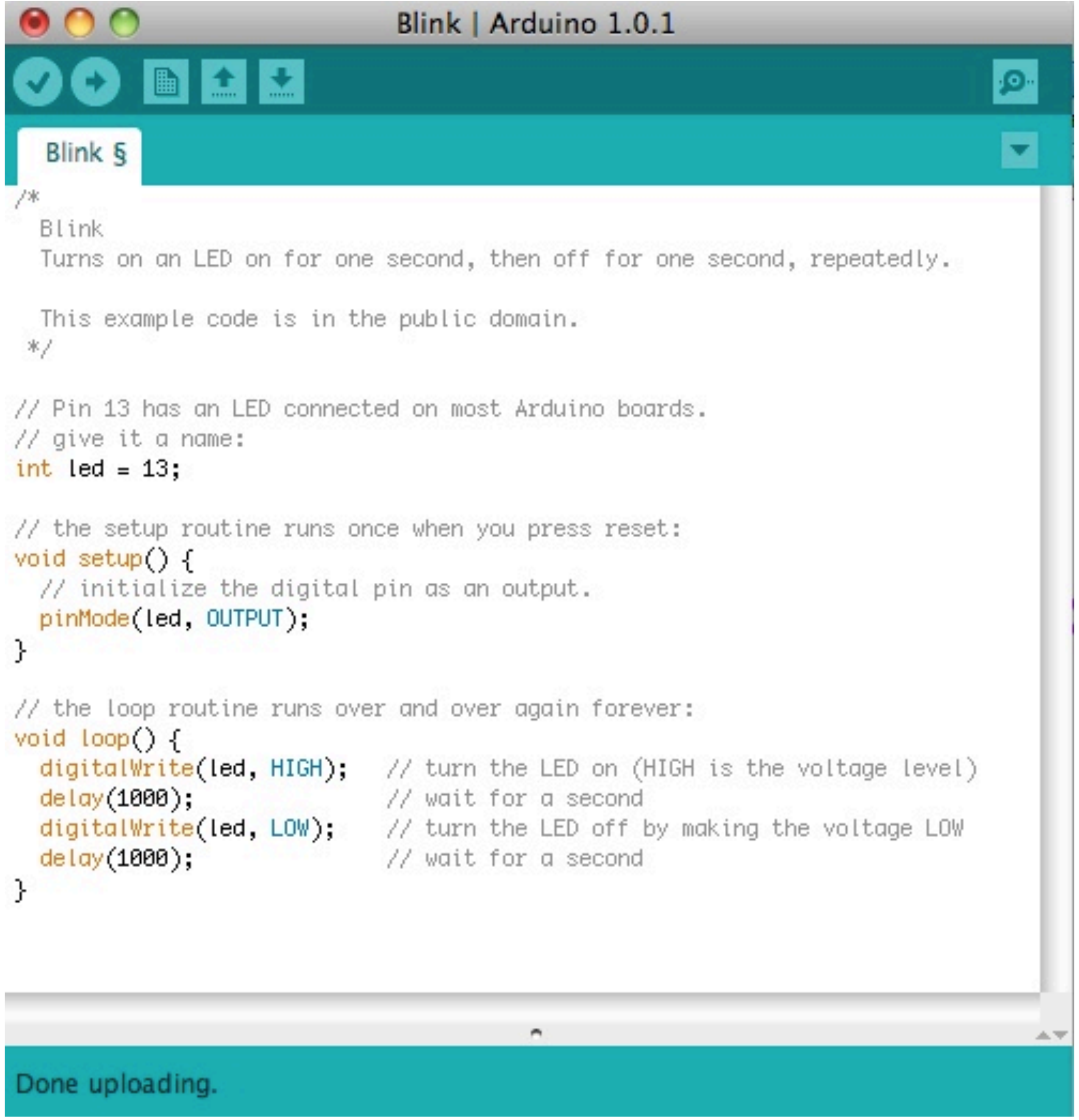
Done uploading.

# hello world

YOUR FIRST PROGRAM

## So this looks familiar...

```
void setup(){
//Initialize your pins here
//Set your baud rate here
}

void loop(){

}
```

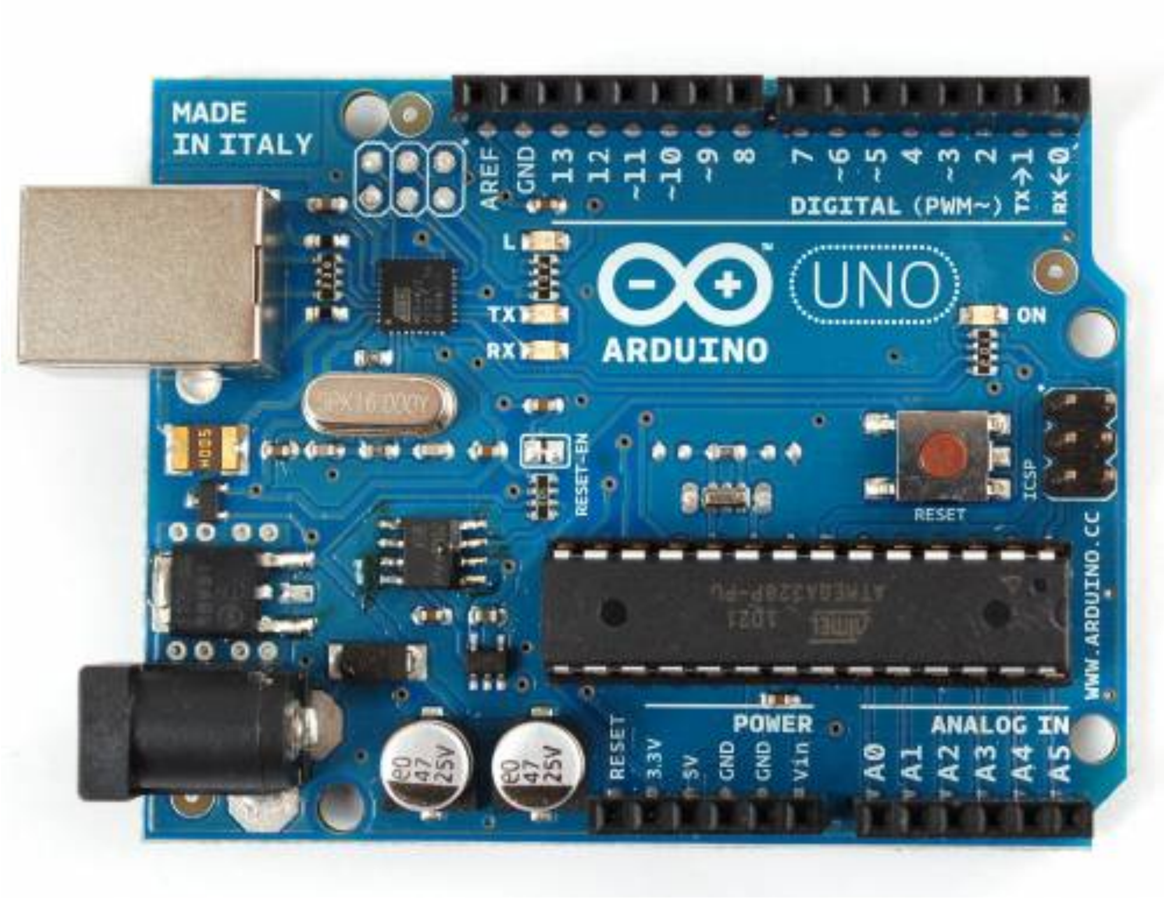


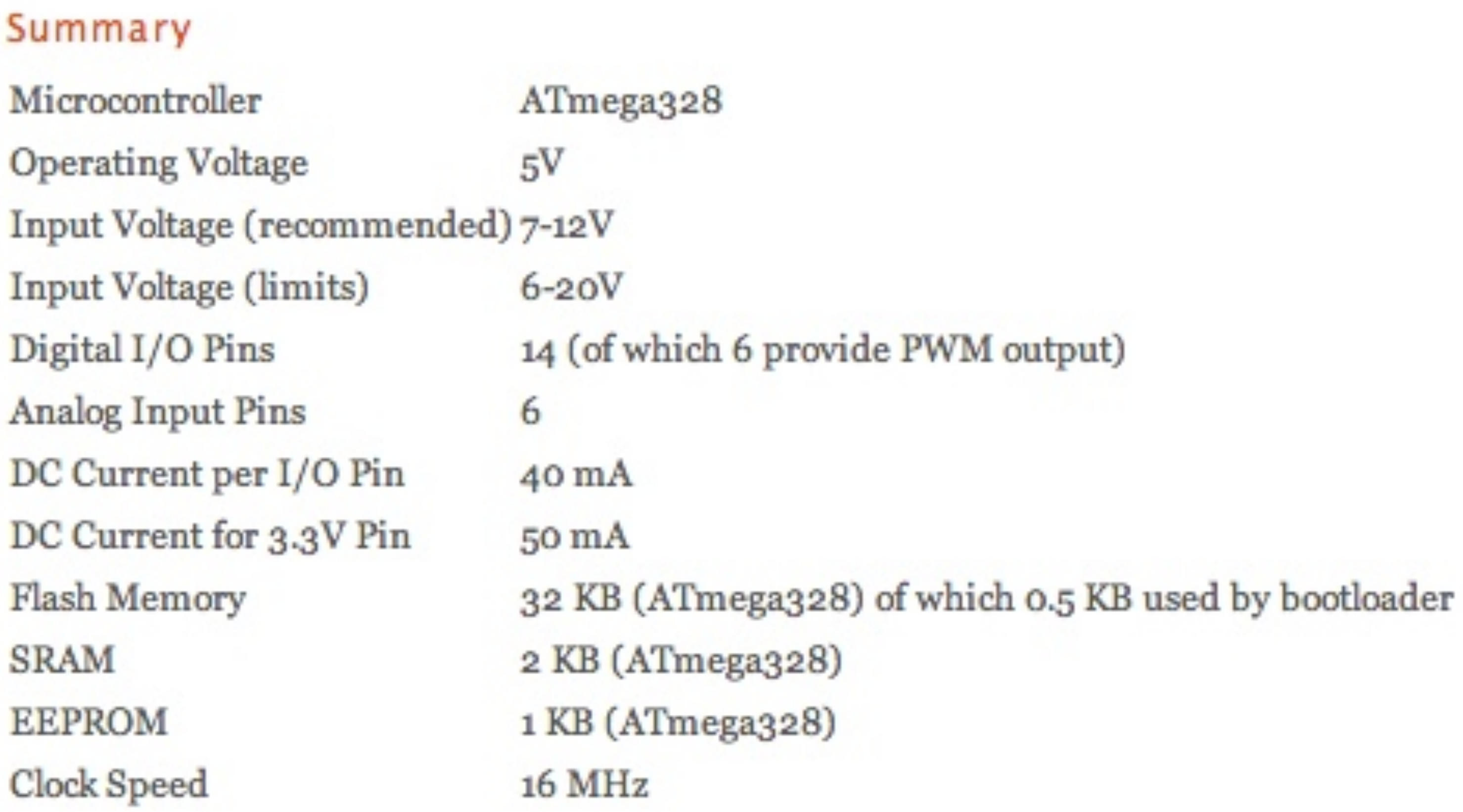
```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

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

Done uploading.

# arduino

TECH SPECS





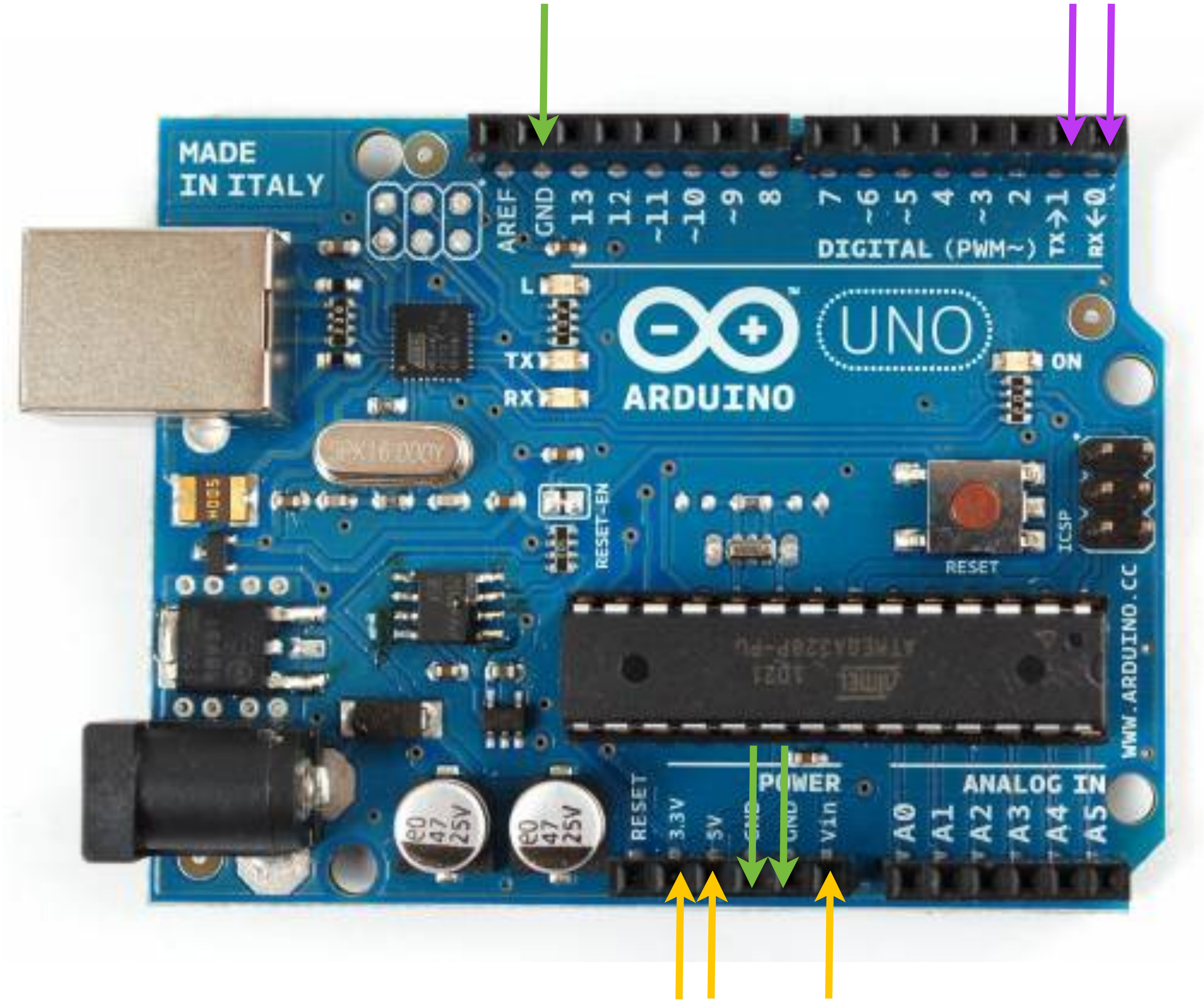
Summary

| | |
|---|---|
| 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 |

# arduino

## PIN DESCRIPTION

A pin provides an input or output through which the controller can communicate with components.



TX/RX (serial - transmit/receive)

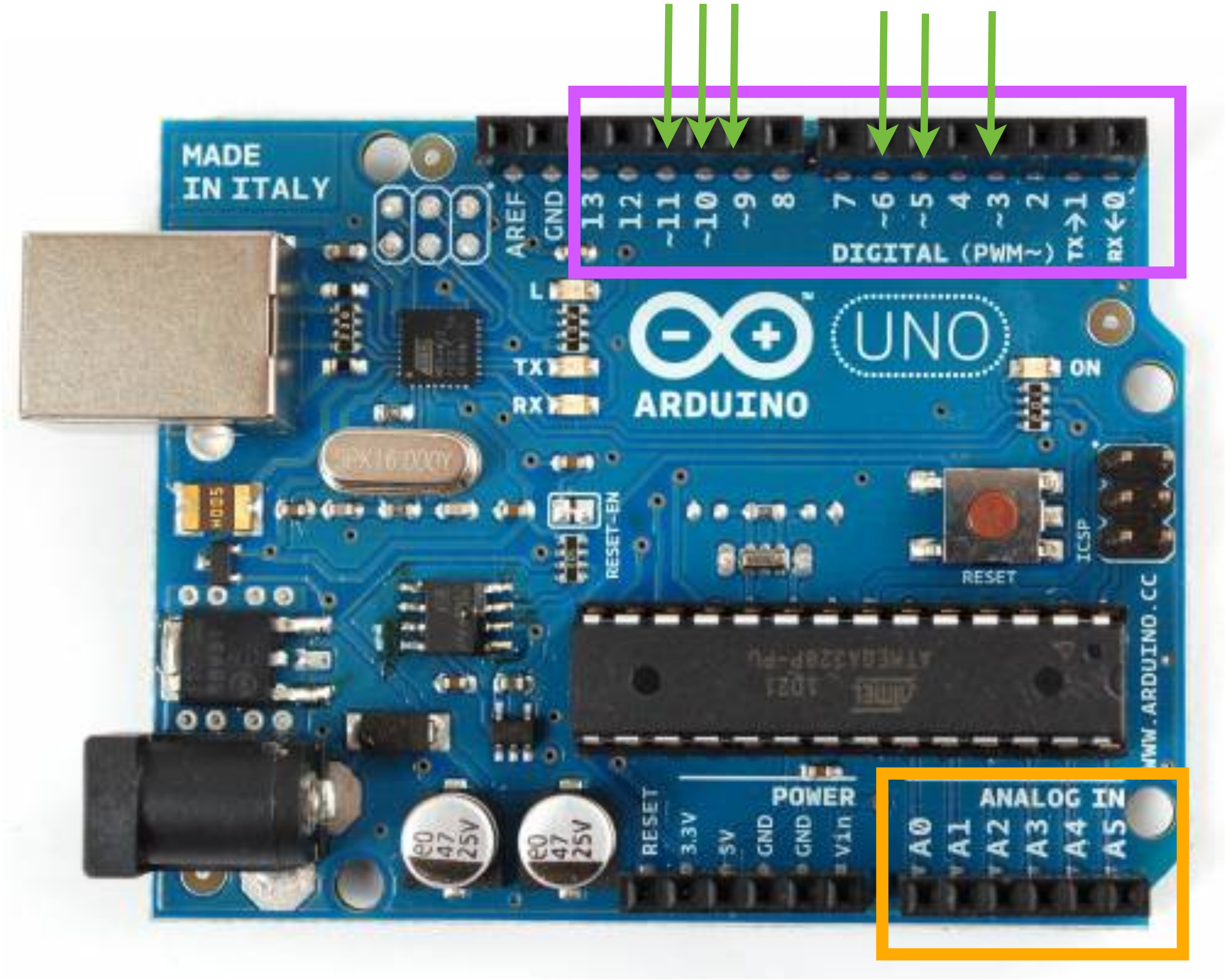
3 ground pins

3 power pins
// 5 volts
// 3 volts
// VIN - can plug 9 volts here

# arduino

PIN DESCRIPTION

A pin provides an input or output through which the controller can communicate with components.



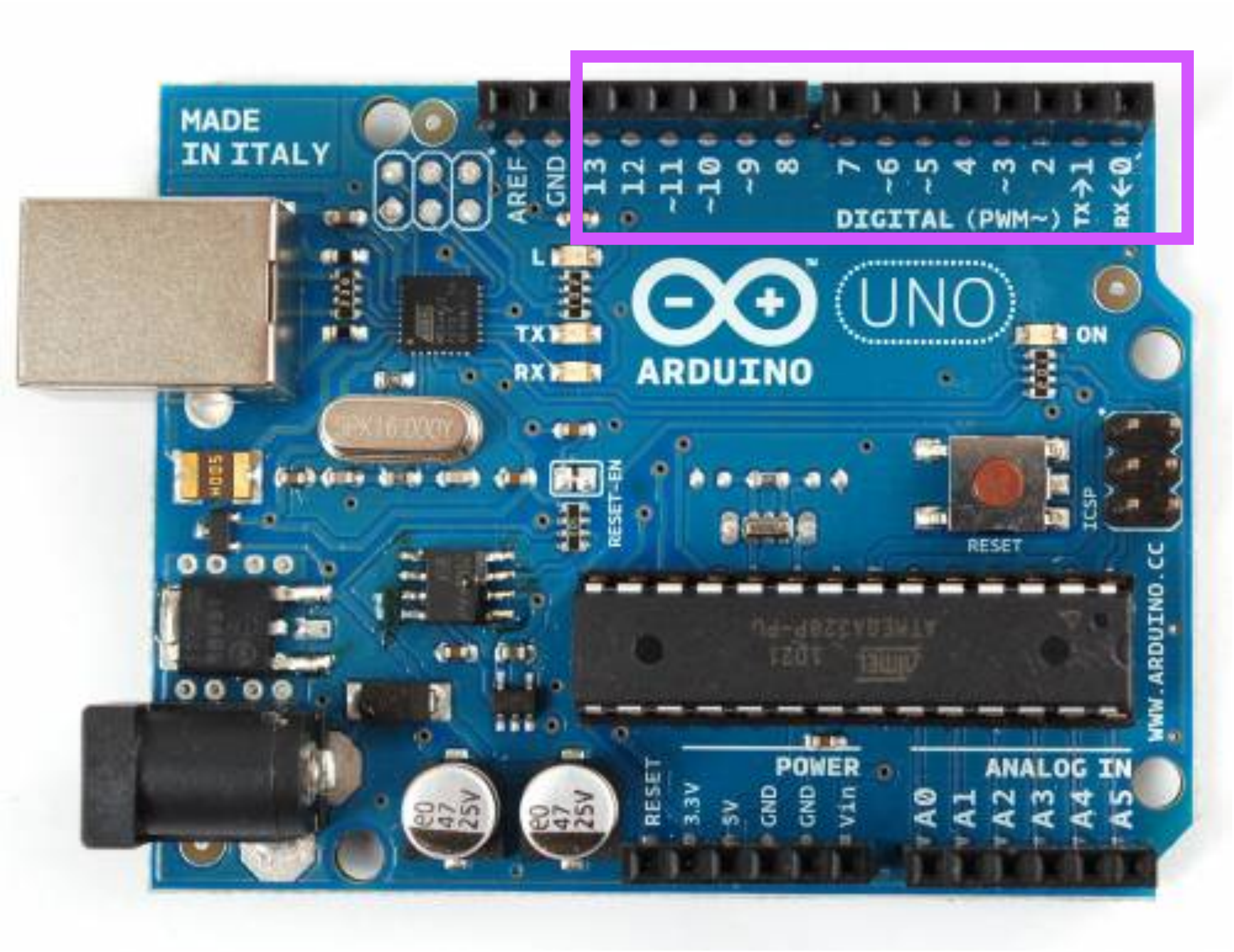
14 Digital pins

6 Pulse Width Modulation enabled pins

6 Analog input pins

# arduino

DIGITAL PINS



14 Digital pins

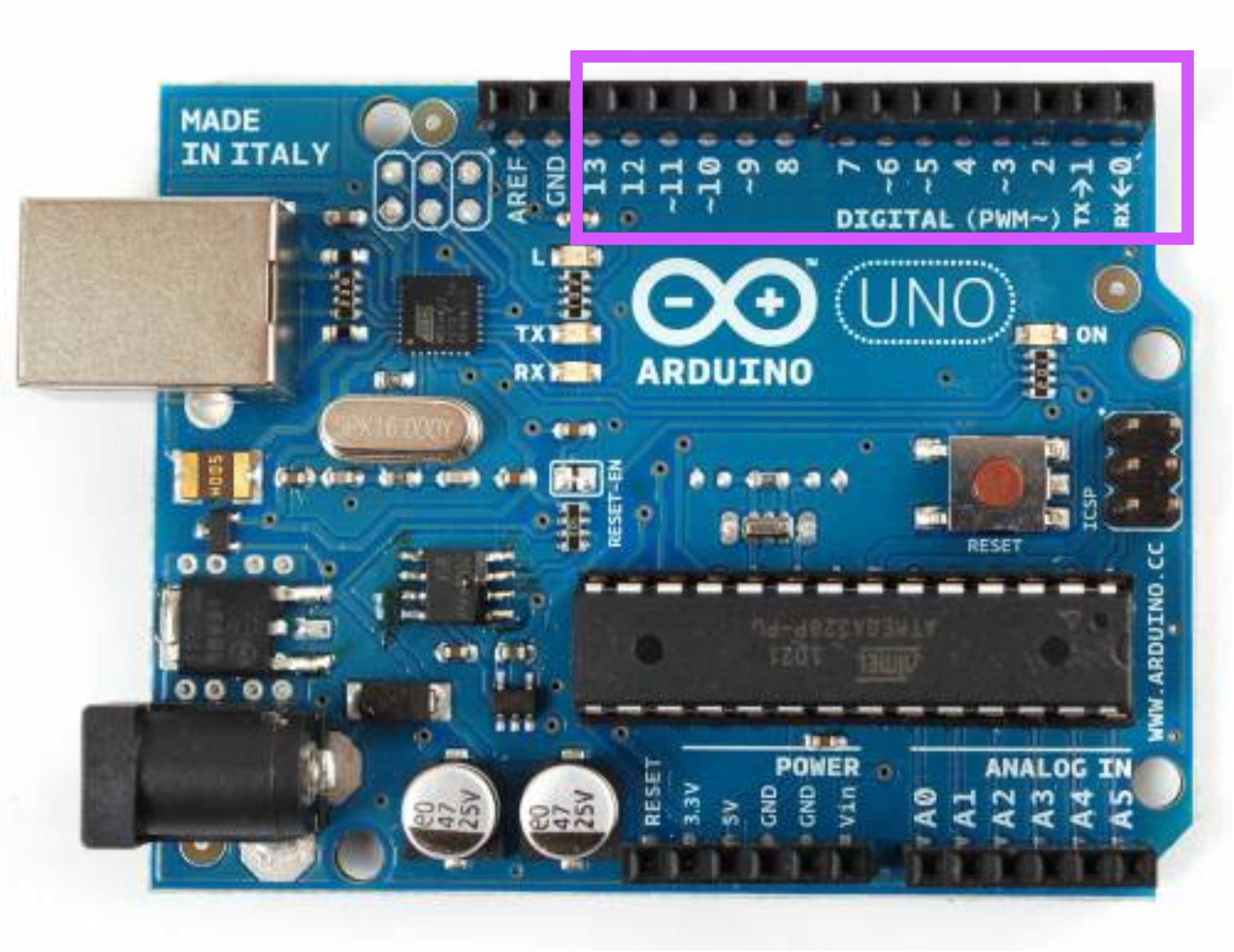
You can read or write 2 different values to them:

| HIGH | LOW |
|---|---|
| 5 volts | 0 volts |

You can think of HIGH as on and LOW as off

# arduino

## DIGITAL PINS



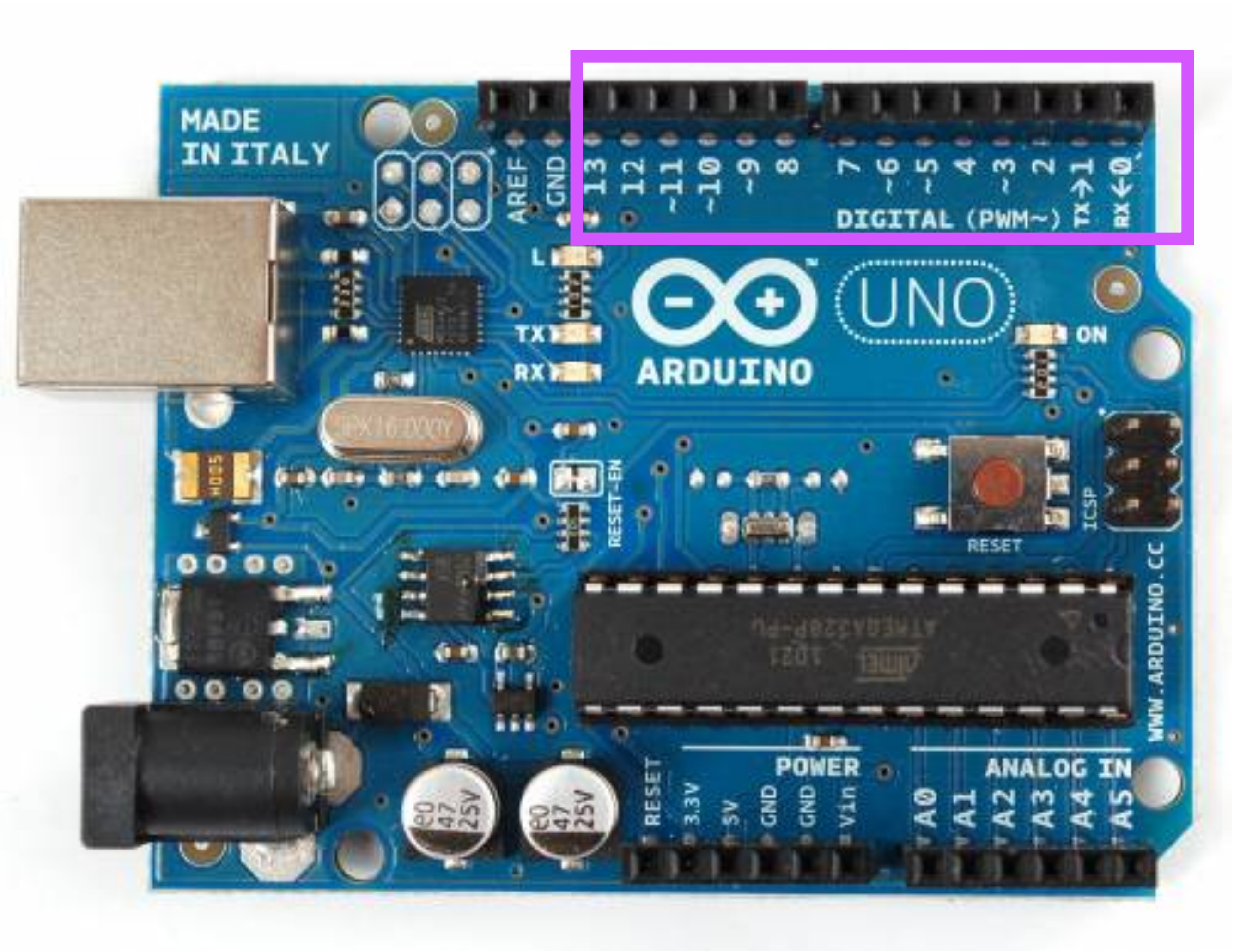
```
pinMode(pin,mode)
```

Sets the pin to be INPUT or OUTPUT

You don’t have to do this every time but it is GOOD PRACTICE

# arduino

DIGITAL PINS



`pinMode(pin,mode)`

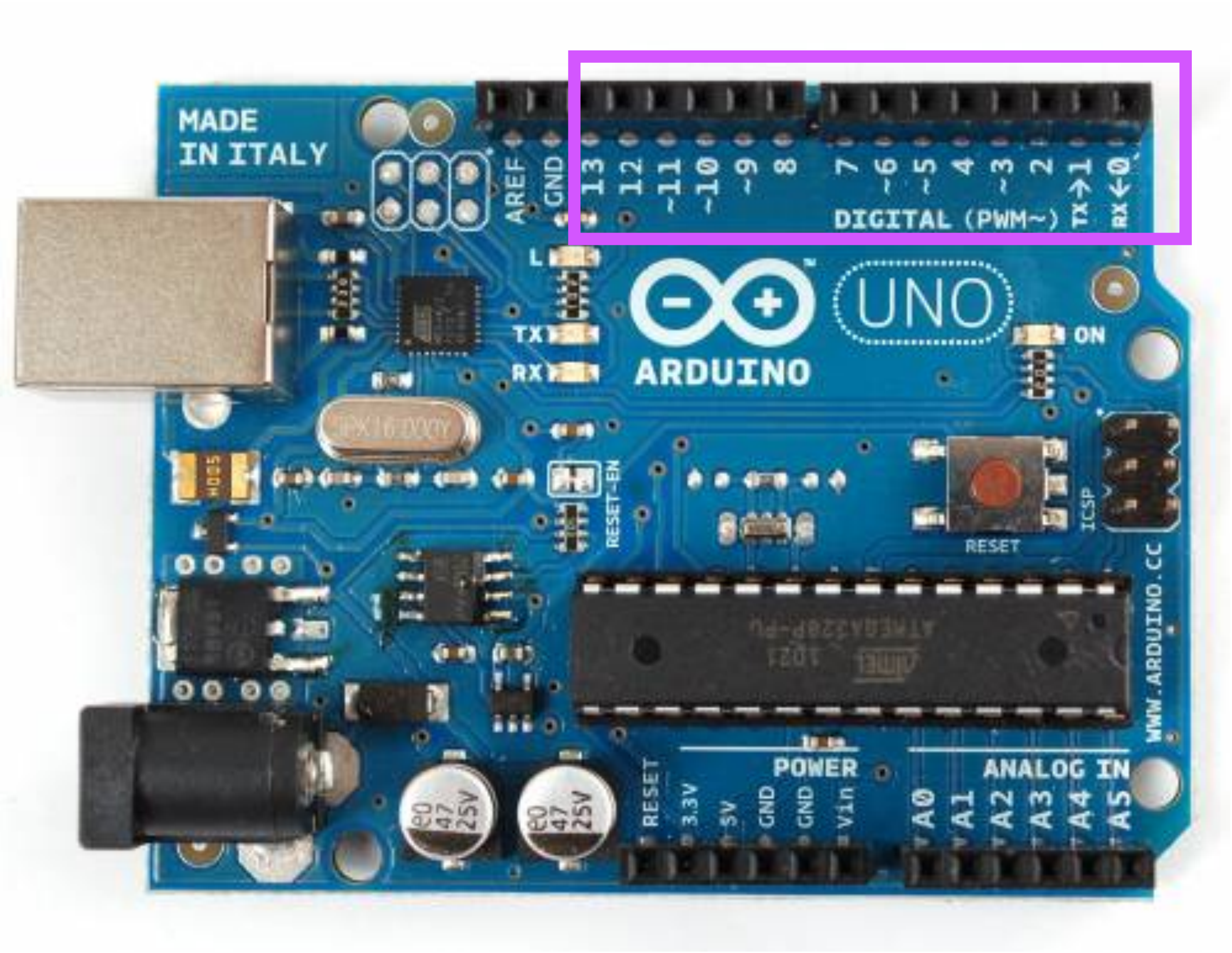
Sets the pin to be INPUT or OUTPUT
You don't have to do this every time but
it is GOOD PRACTICE

`digitalRead(pinNumber)`

Returns value from specified
For INPUT

# arduino

## DIGITAL PINS



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

Sets the pin to be INPUT or OUTPUT
You don’t have to do this every time but it is GOOD PRACTICE

```
digitalRead(pinNumber)
```

Returns value from specified
For INPUT

```
digitalWrite(pinNumber,value)
```

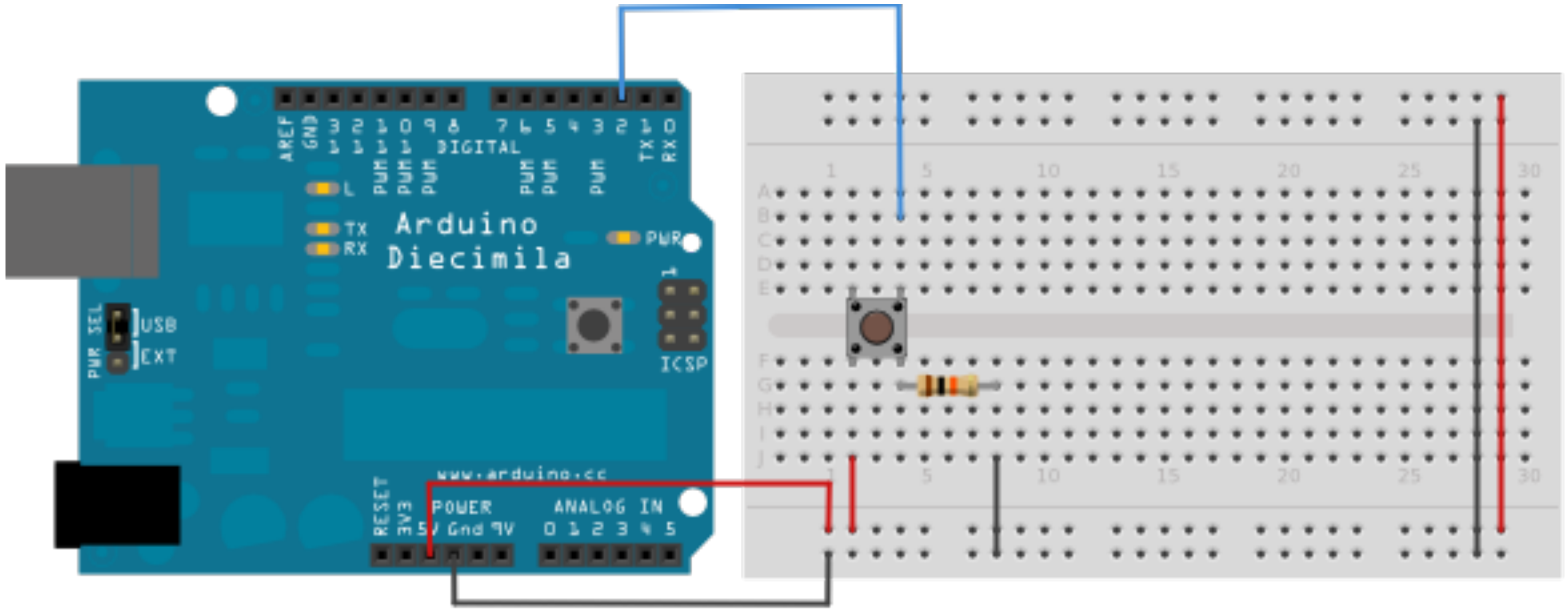
Writes a value to the pin. Here we are talking HIGH (5V/on) or LOW (0V/off)
For OUTPUT

arduino

MAKE IT

# Let’s build a button.

You need:
_ 10K resistor
_ button
_ LED
_ jump wire

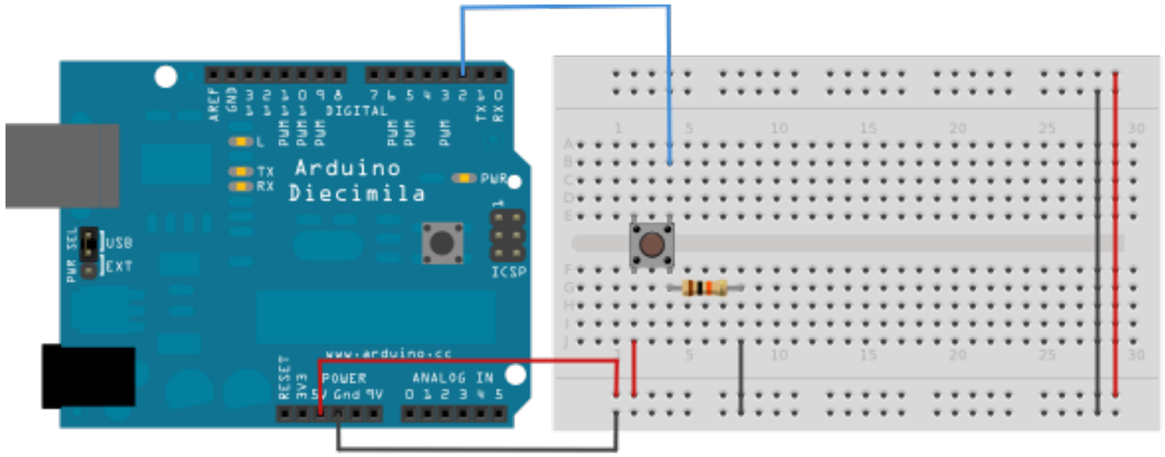


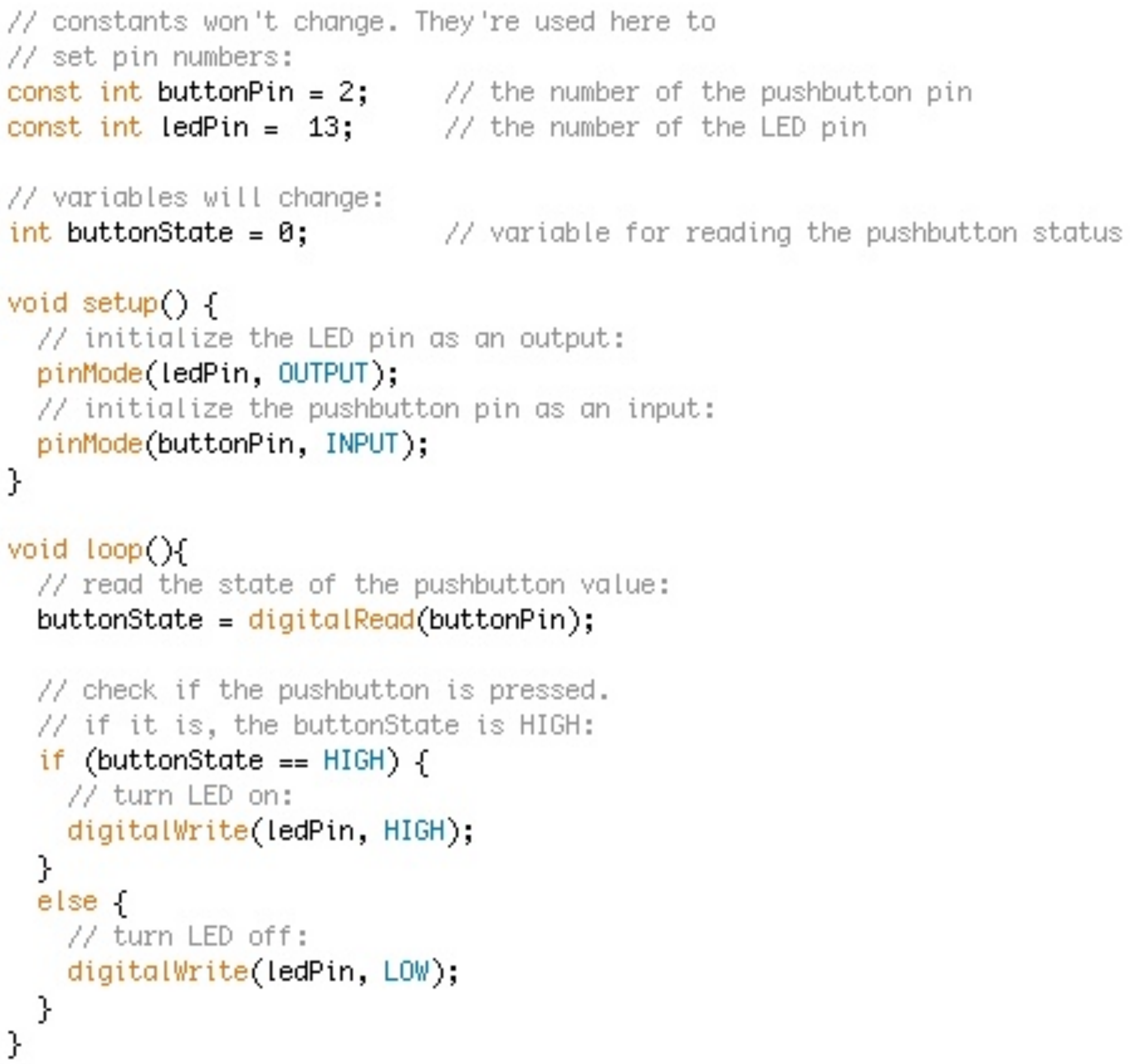
# Go!

# arduino

MAKE IT

## Now to examine:

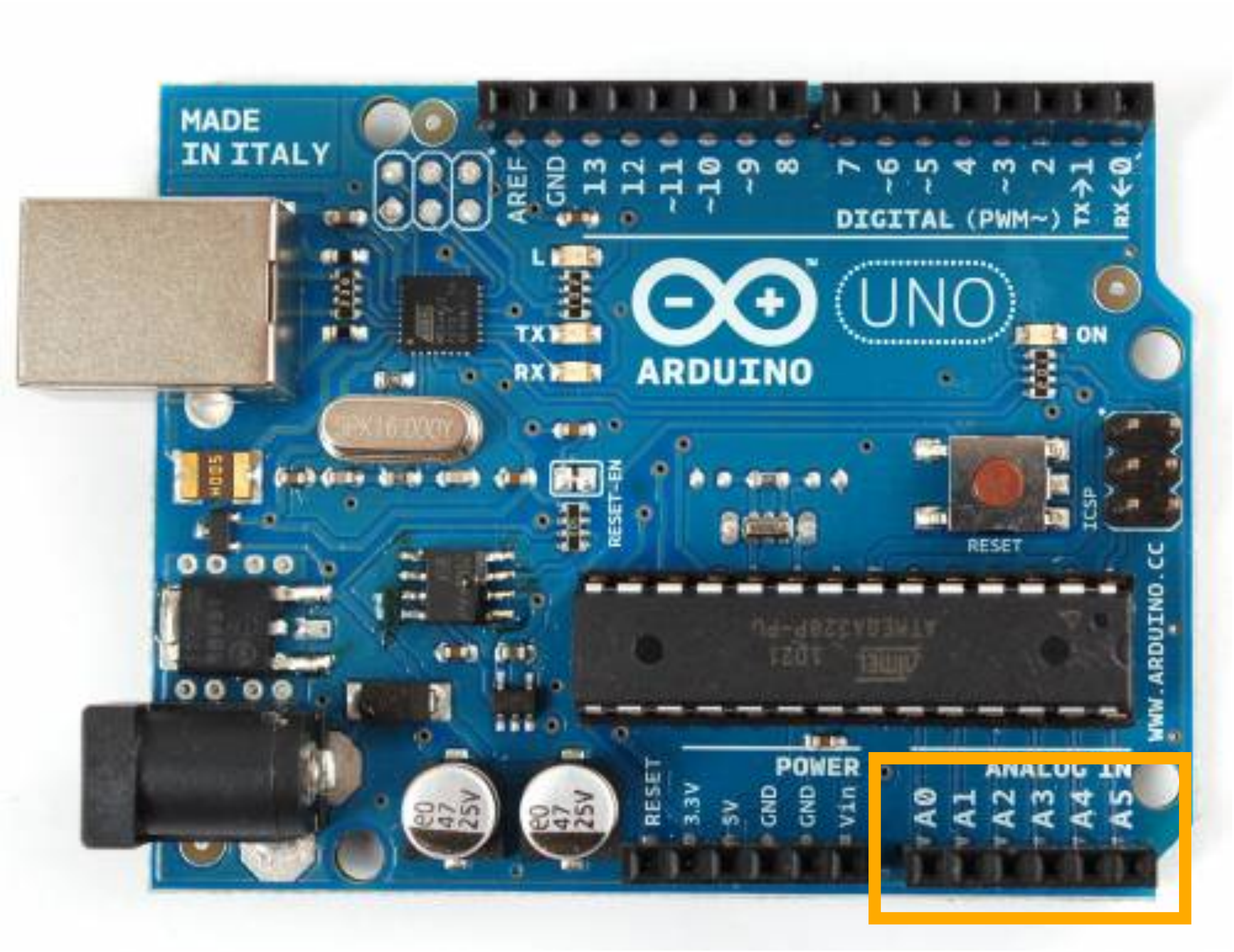




```
// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin =  13;      // the number of the LED pin

// 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);
  }
}
```

# arduino

ANALOG INPUT PINS



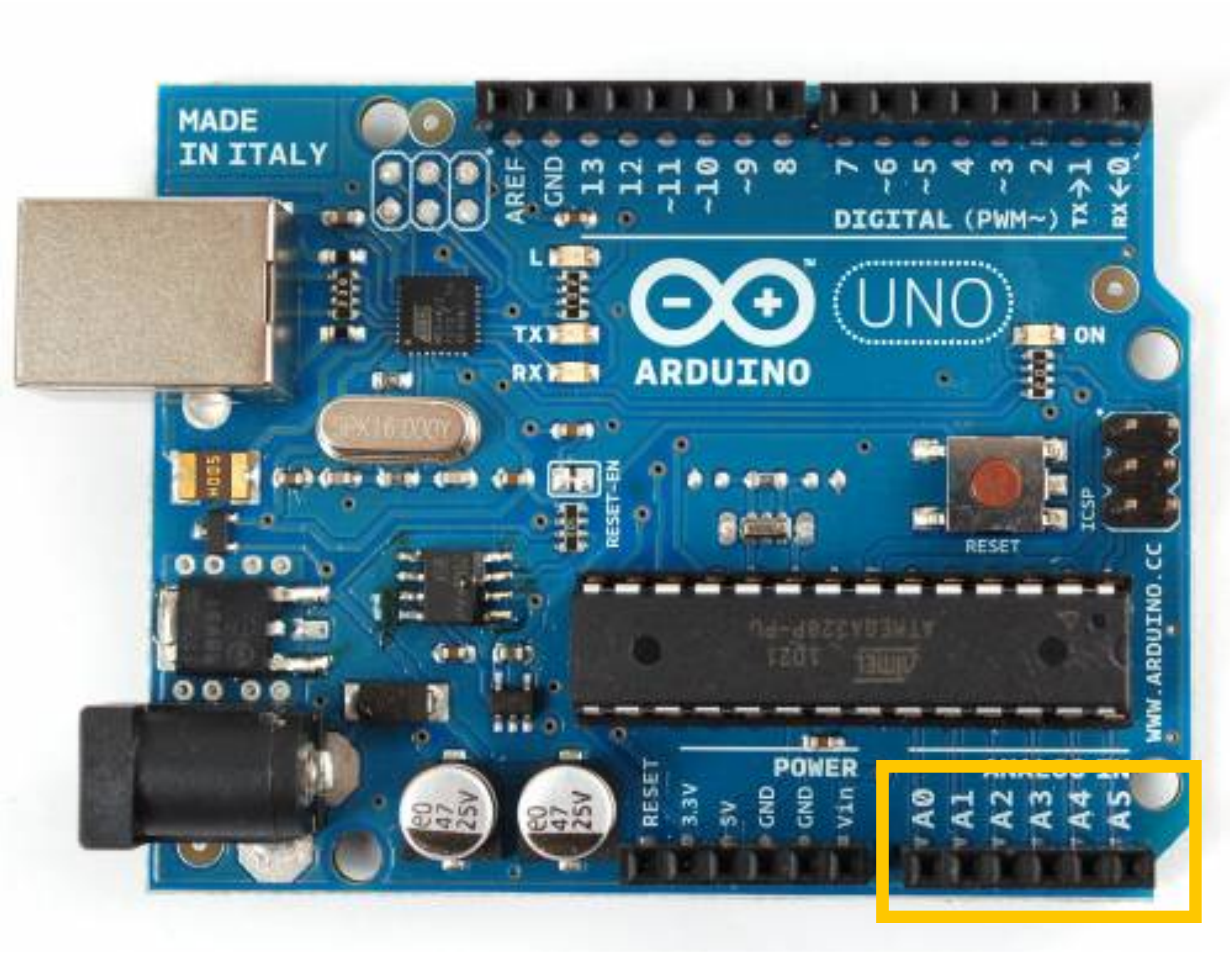
## 6 Analog Input pins

You can read or write a wide range of values

| | |
|---|---|
| Read | 0 - 1023 |
| Written | 0 - 255 |

# arduino

## ANALOG INPUT PINS



```
analogRead(pinNumber)
```

Reads value from specified analog in pin

For INPUT

# serial communication

THIS AIN'T YO BREAKFAST

If you want to read the values coming in from `analogRead()`, use the serial monitor.

Arduino usually communicates at a baud rate of 9600. This is the rate Arduino and the computer agree to exchange information.

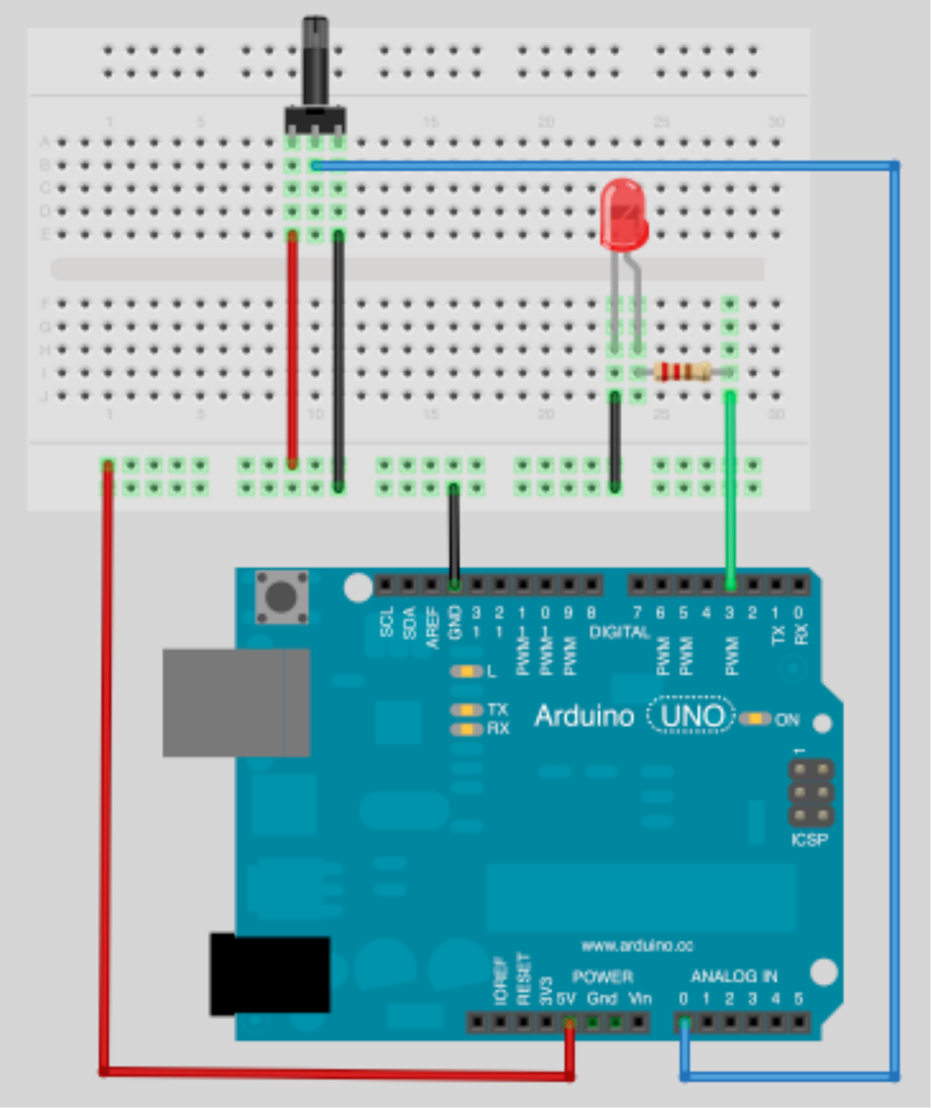
This is also your best way to debug your program.

# Build a circuit using a potentiometer.

Let’s work through the code together.

Pot signal = A0

LED = 9



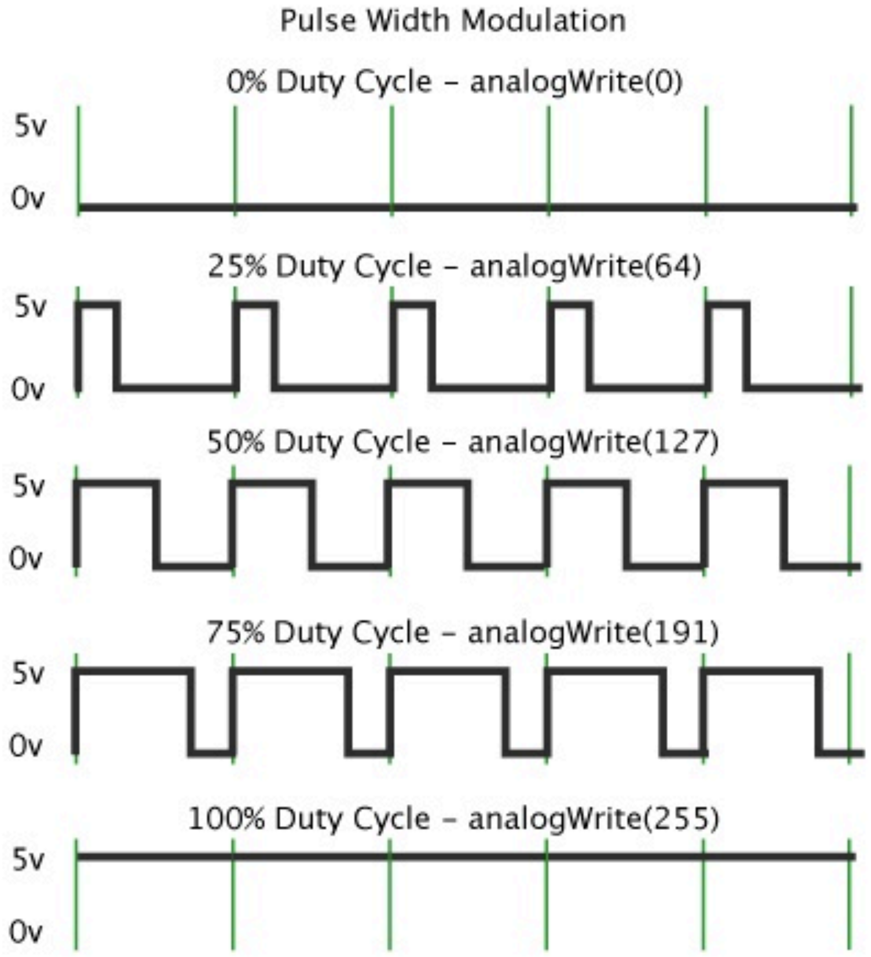
# arduino

ANALOG OUTPUT PINS

So we can blink an LED, but what about fading it?

Bad news: Arduino is not truly analog. BUT!

You can simulate analog behavior by turning a signal on and off at different frequencies.


Pulse Width Modulation
0% Duty Cycle - analogWrite(0)
5v
0v
25% Duty Cycle - analogWrite(64)
5v
0v
50% Duty Cycle - analogWrite(127)
5v
0v
75% Duty Cycle - analogWrite(191)
5v
0v
100% Duty Cycle - analogWrite(255)
5v
0v

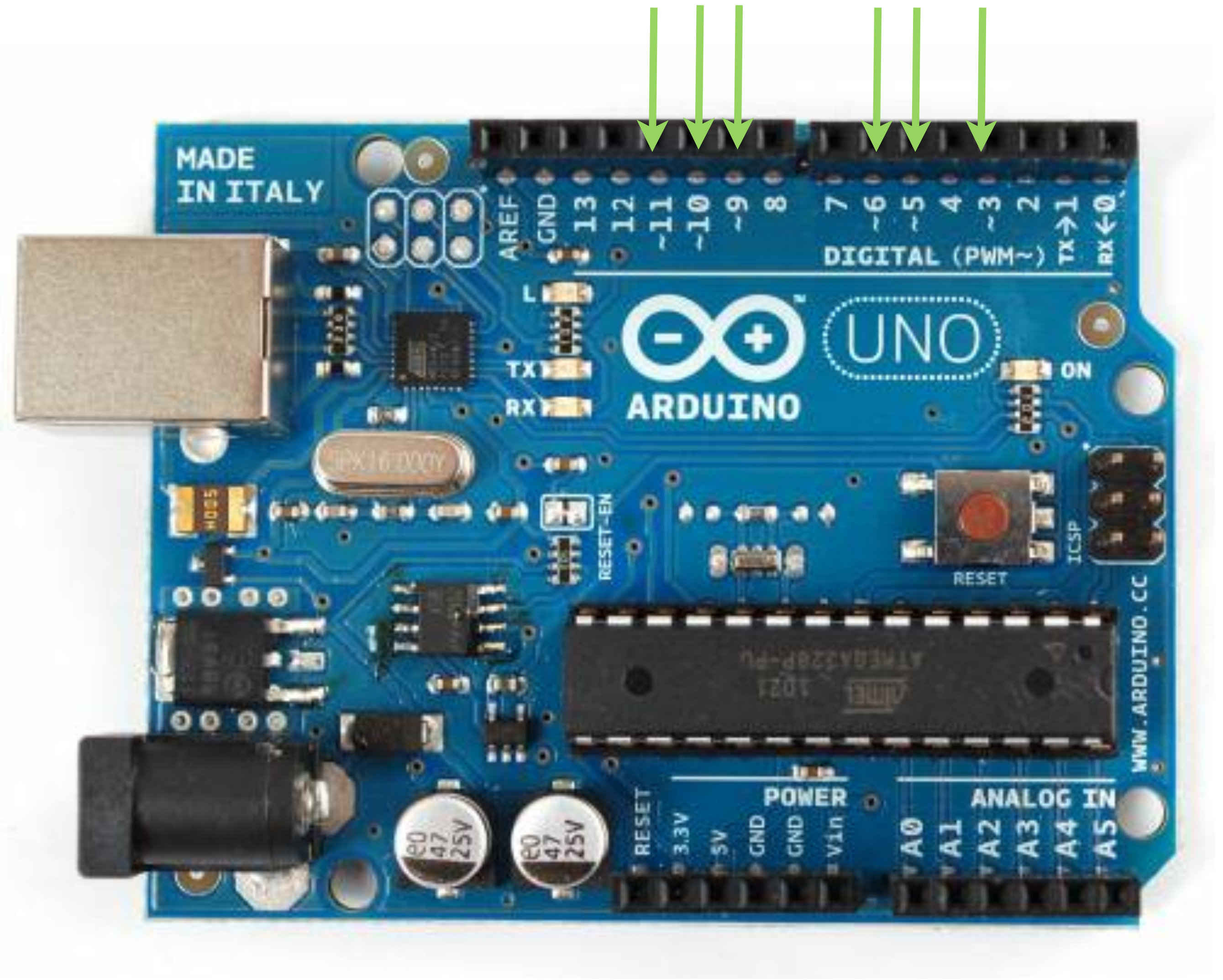
# arduino

ANALOG OUTPUT PINS

This is called
Pulse Width Modulation.

Only a few pins can execute this function: 3, 5, 6, 9, 10, and 11.

These are special because they can be digital I/O OR analog out.



```
analogWrite(pin,value)
```

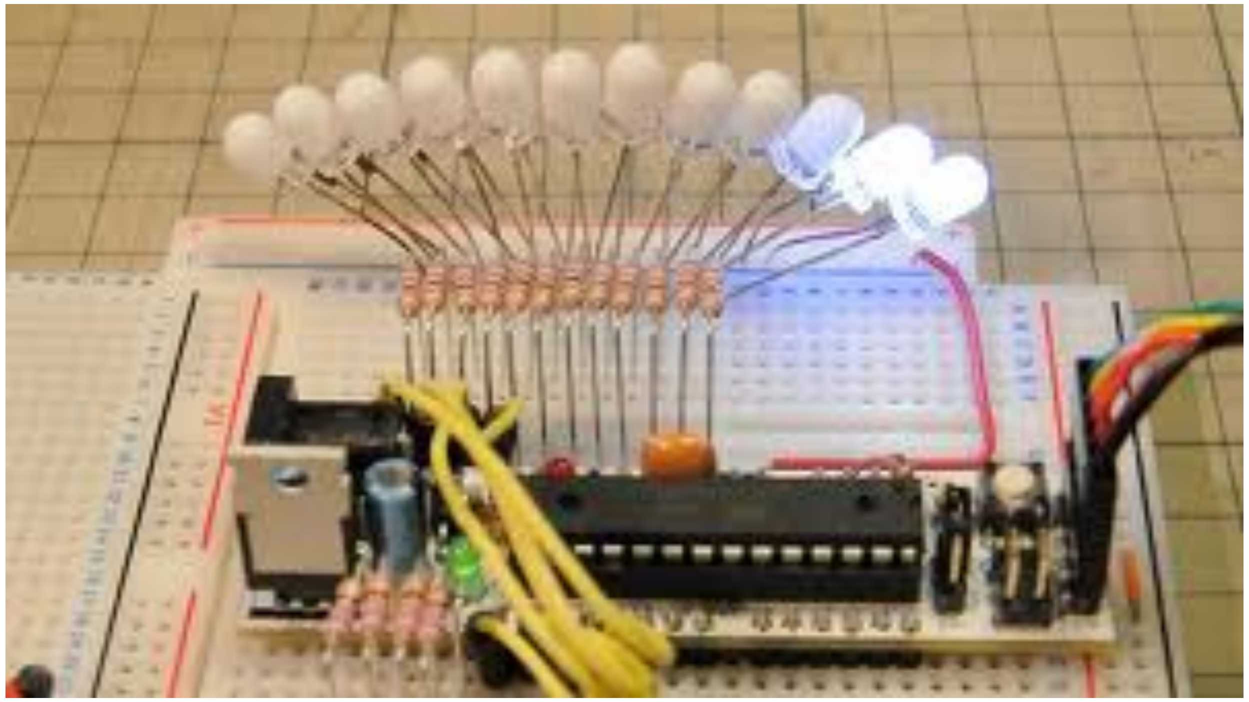
The value can be between 0 - 255.
For OUTPUT

# arduino

DIGITAL PINS

## Everyone fade an LED.

Try different values to see
how the behavior changes.



# fun functions

THAT YOU MIGHT WANT TO USE FOR HOMEWORK

`random()`

`map()`

`constrain()`

`switch case`

`if then`, `for`, and all those other fun control statements also apply!