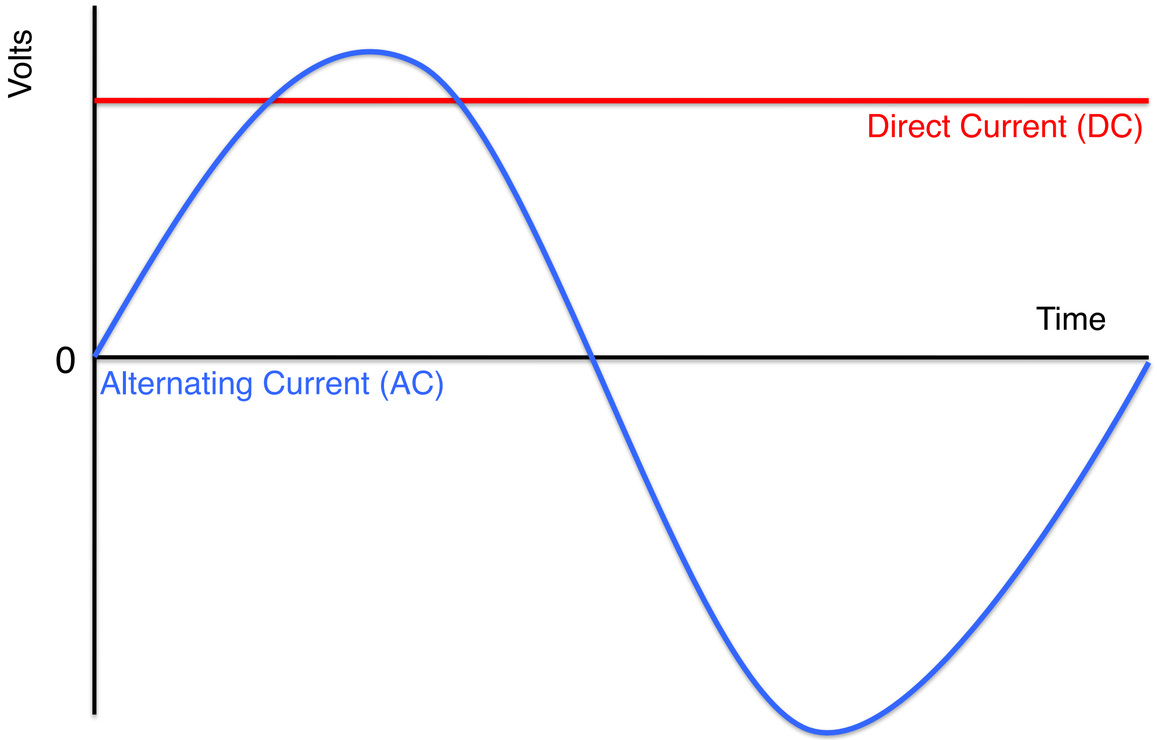
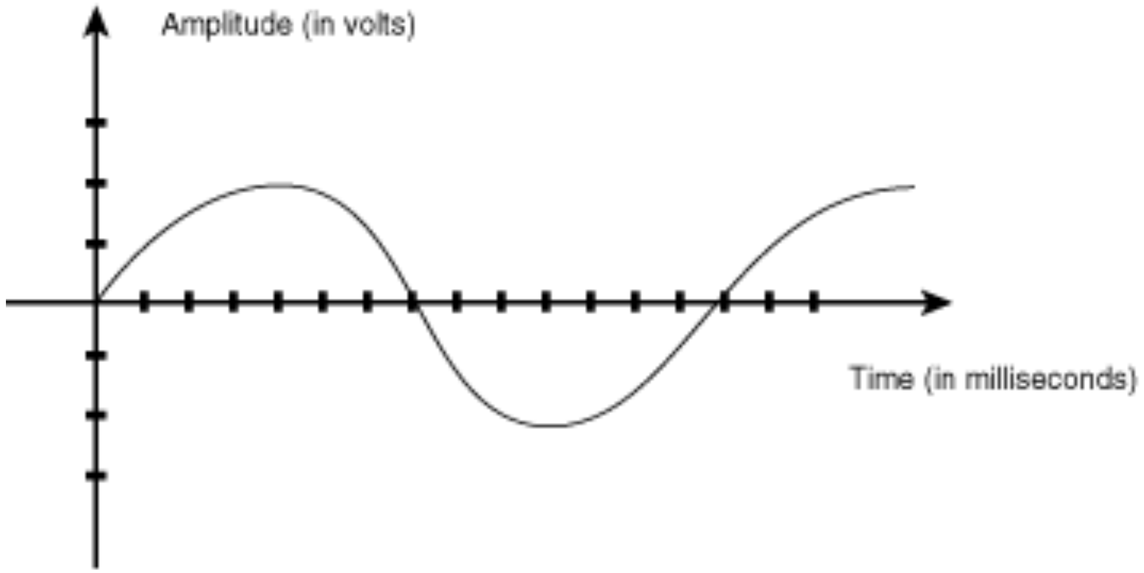
ELECTRONICS 101

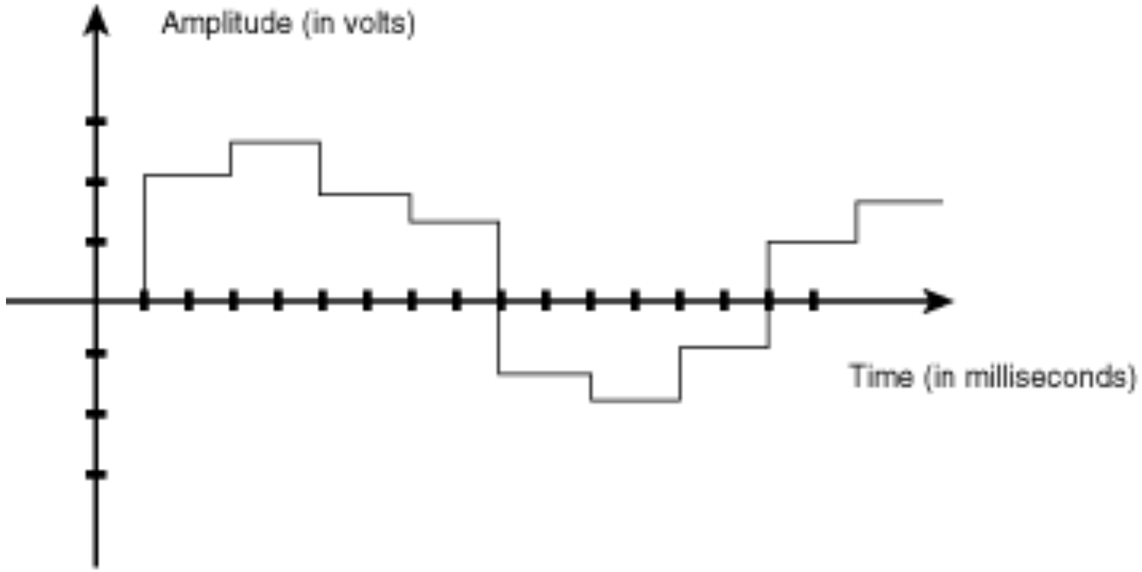
AC vs DC



DIGITAL vs ANALOG



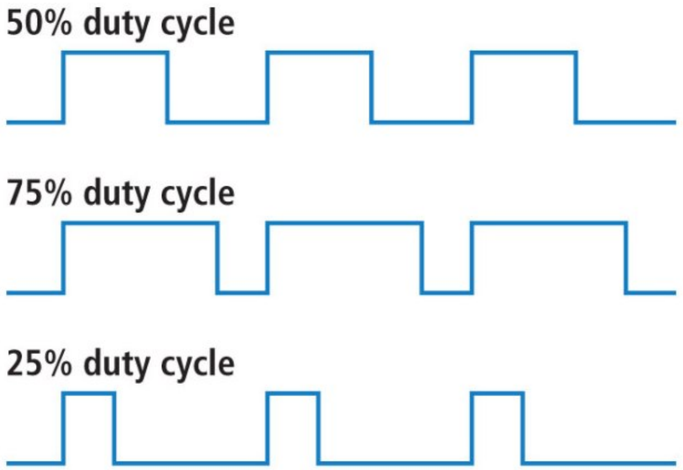
Analog signals
are continuous signals that have an infinite number
of possible values within their range.

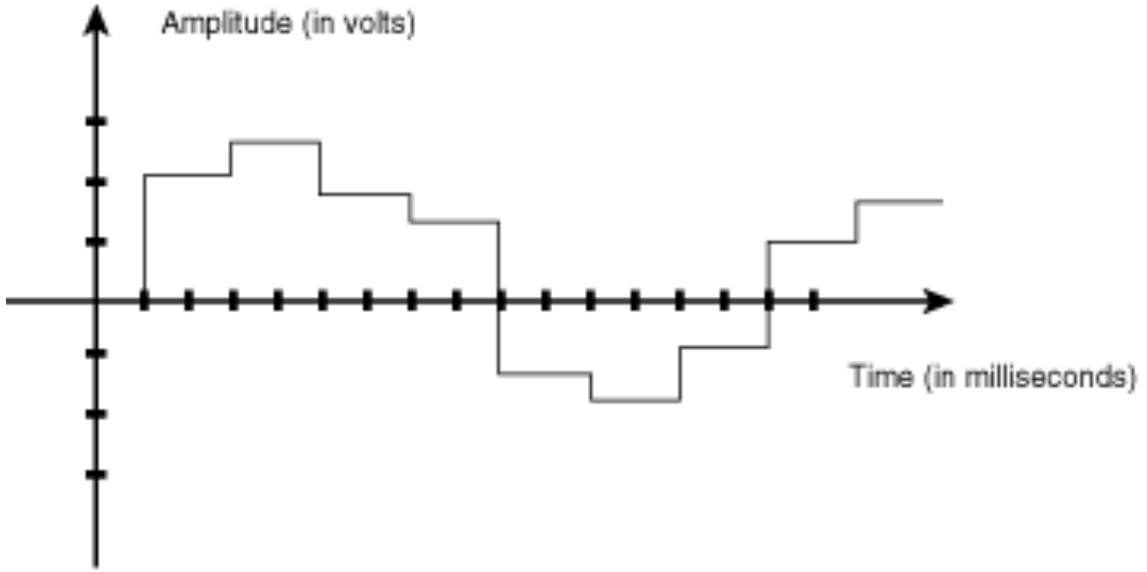


Digital signals are discrete signals.

In 8-bit resolution they allow 256 (2^8) different
values and binary ON/OFF signals.

PULSE WIDTH MODULATION (PWM)



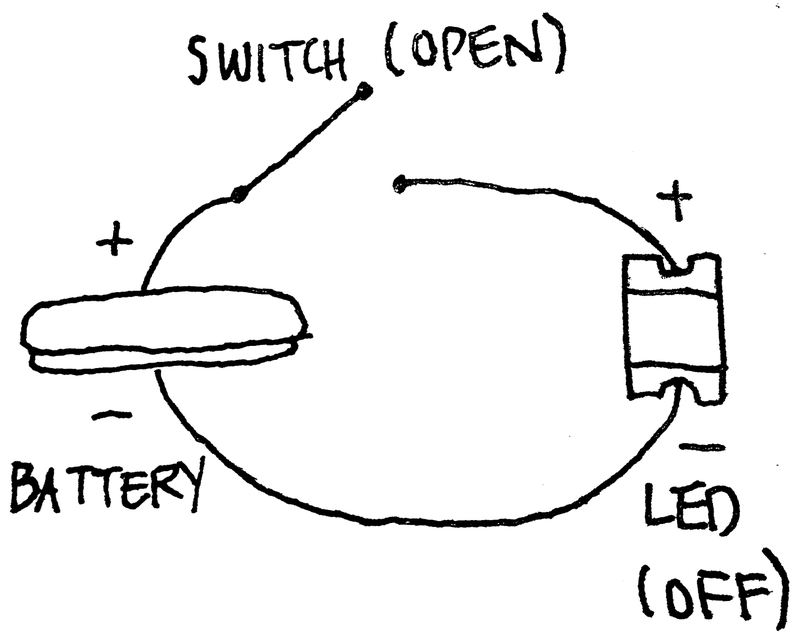


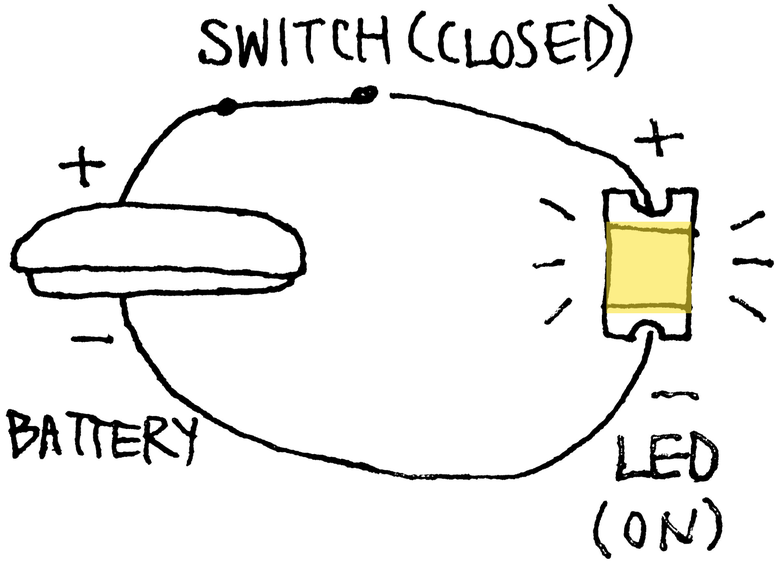
A ***Pulse Width Modulation*** (***PWM***) signal is a method for generating an analog signal using a digital source.

The percentage of ***duty cycle*** describes the percentage of time a digital signal is ON over an interval or period of time.

The ***frequency*** determines how fast the PWM completes a cycle

CIRCUIT

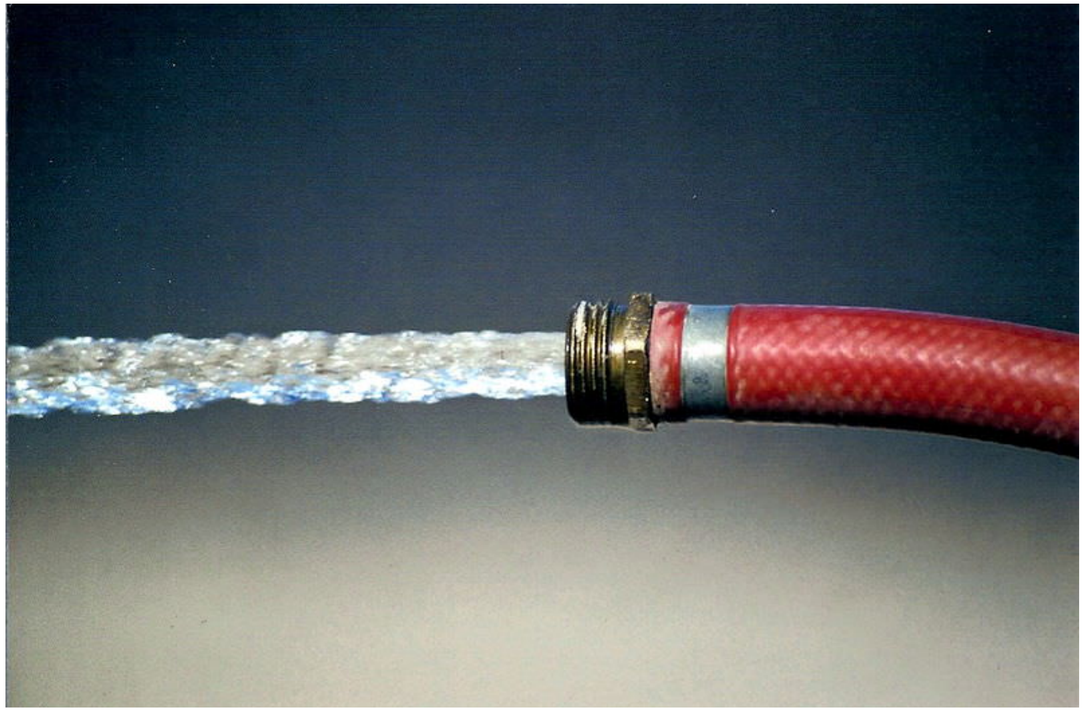




CIRCUIT ANALOGY

The flow of water through a hose is like the flow of electricity through a circuit. Turning the faucet increases the ***amount of water*** coming through the hose, or increases the ***current*** (amps). The ***diameter*** of the hose offers ***resistance*** to the current, determining how much water can flow. The ***speed*** of the water is equivalent to ***voltage***. When you put your thumb over the end of the hose, you reduce the diameter of the pathway of the water. In other words, the resistance goes up. The current (that is, how much water is flowing) doesn't change, however, so the speed of the water, or voltage, has to go up so that the water can escape.

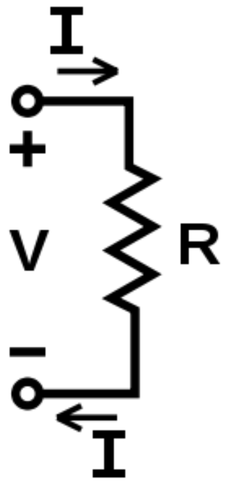
-Dan O'Sullivan & Tom Igoe

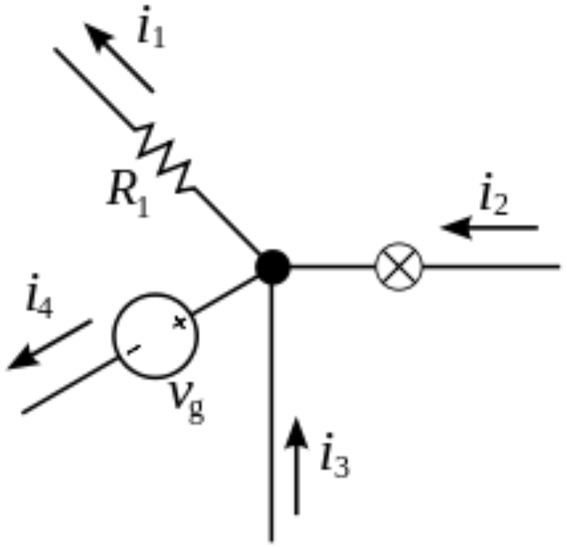


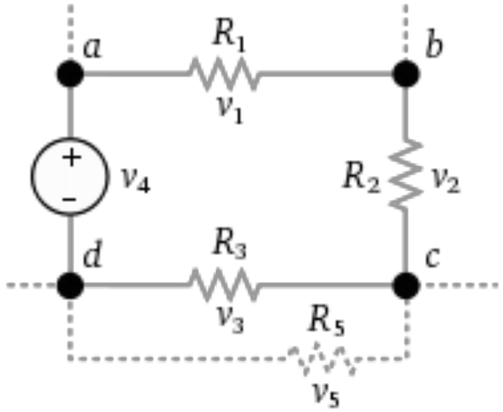
UNITS

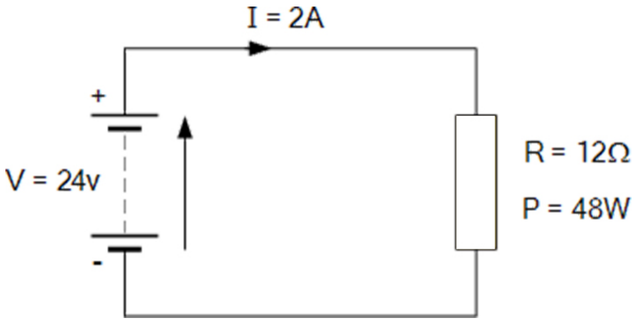
Quantity	Unit	Example
Electric potential (V)	Volt (V)	12V battery pack
Current (I)	Ampere (A)	10 mA LED
Power (P)	Watt (W)	DC motor 12W (1A @ 12V)
Resistance (R)	Ohm (Ω)	Use a 220Ω pull-up on the 10mA LED at 5V
Capacitance (C)	Farad (F)	Use a 100nF capacitor between +5V and GND

OHM, KIRCHOFF, and JOULE









Ohm's law:

V = R * I
I = V / R
R = V / I

Kirchhoff's current law (KCL):

At any node (junction) in an electrical circuit, the sum of currents flowing into that node is equal to the sum of currents flowing out of that node.

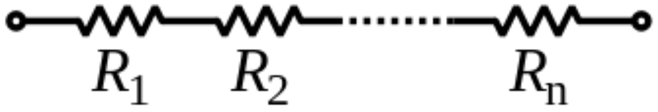
Kirchhoff's voltage law (KVL):

The directed sum of the electrical potential differences (voltage) around any closed network is zero.

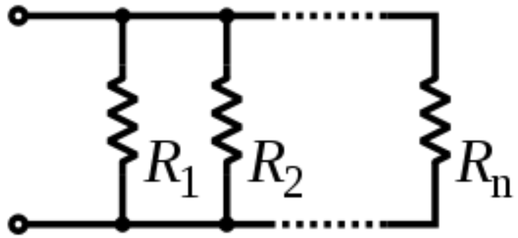
Joule's law:

$P = V * I$
$P = V^2 / R$
$P = I^2 * R$

SERIES and PARALLEL



Series circuits are electrical circuits in which components are joined in a sequence so that the same current flows through all of them.

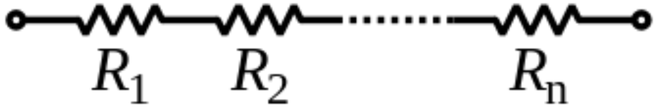


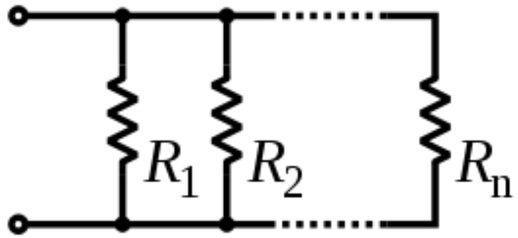
Parallel circuits are electrical circuits in which components are connected in parallel so that identical voltage occurs in all components, with current dividing among the components based on their resistances.

In ***series*** circuits, the circuit will not be complete if one component burns out.

Parallel circuits will still continue to operate (at least with other component) if one component burns out

SERIES and PARALLEL





Series

$V_{total} = V_1 + V_1 + \ldots + V_n$

$I_{total} = I_1 = I_2 = \ldots = I_n$

$R_{total} = R_1 + R_2 + \ldots + R_n$

Parallel

$V_{total} = V_1 = V_2 = \ldots = V_n$

$I_{total} = I_1 + I_2 + \ldots + I_n$

$1/R_{total} = 1/R_1 + 1/R_2 + \ldots + 1/R_n$

SAFETY

Due to resistance of the body, a dangerous current (10 mA or higher) can only be reached with a high voltage.

Generally, voltages of less than 50V are relatively safe.

… Use common sense

CIRCUIT COMPONENTS

LEARNING RESOURCES AND VENDORS

Hardware

https://www.sparkfun.com/
http://www.adafruit.com/
http://www.adafruit.com/product/1438
http://www.robotshop.com/
http://www.digikey.com/
http://www.pololu.com/
http://www.dynalloy.com/
http://samstechlib.com/24614782/en/read/4_Band_Resistor_Color_Codes

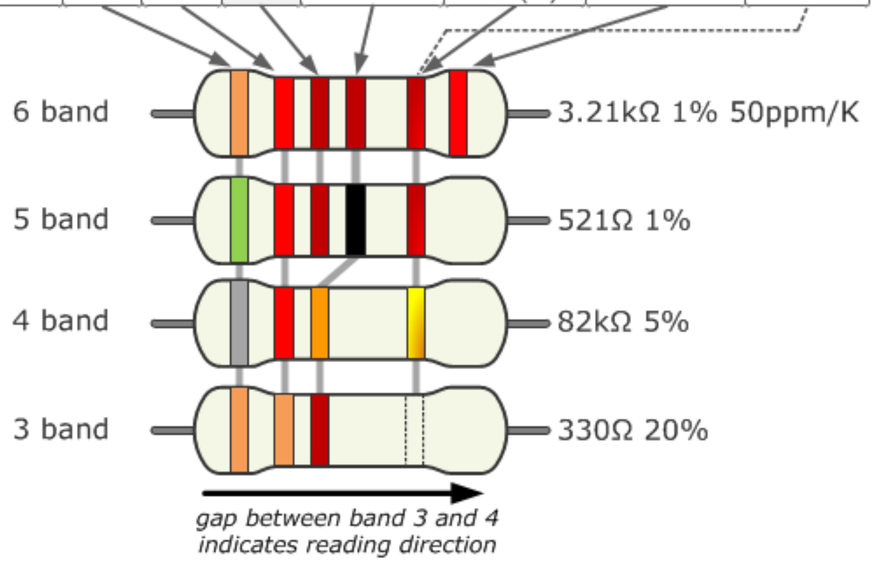
SOLDERLESS BREADBOARD

RESISTORS

A ***resistor*** limits the flow of electricity. They are measured in ohms (Ω). 10Ω is less resistance than 1kΩ.

www.resistorguide.com

	Color	Signficant figures			Multiply	Tolerance (%)	Temp. Coeff. (ppm/K)	Fail Rate (%)
Bad	black	0	0	0	x 1		250 (U)	
Beer	brown	1	1	1	x 10	1 (F)	100 (S)	1
Rots	red	2	2	2	x 100	2 (G)	50 (R)	0.1
Our	orange	3	3	3	x 1K		15 (P)	0.01
Young	yellow	4	4	4	x 10K		25 (Q)	0.001
Guts	green	5	5	5	x 100K	0.5 (D)	20 (Z)	
But	blue	6	6	6	x 1M	0.25 (C)	10 (Z)	
Vodka	violet	7	7	7	x 10M	0.1 (B)	5 (M)	
Goes	grey	8	8	8	x 100M	0.05 (A)	1(K)	
Well	white	9	9	9	x 1G			
Get	gold			3th digit only for 5 and 6 bands	x 0.1	5 (J)		
Some	silver				x 0.01	10 (K)		
Now!	none					20 (M)		



CAPACITORS

A ***capacitor*** stores electrons. It stores electrical charge when current is applied, and discharges when a current is removed. This can smooth out the dips and spikes in a current signal. They are measured in farads.

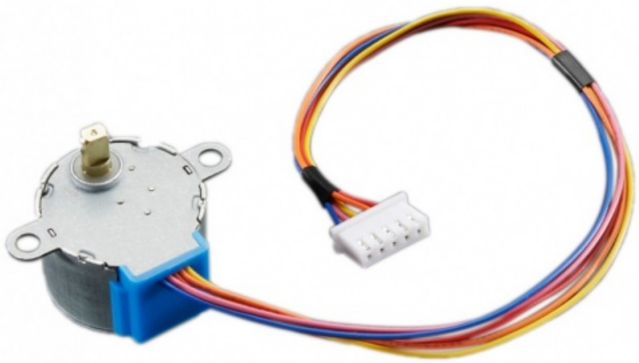
DIODES

Current flows only in one direction through a ***diode***. One side is called the anode, the other the cathode, with current flowing from the anode to the cathode.

Diodes can help protect components in circuitry that can be damaged from reverse current.

ACTUATORS

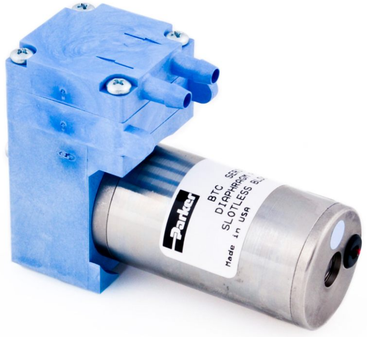
dc motor



stepper motor

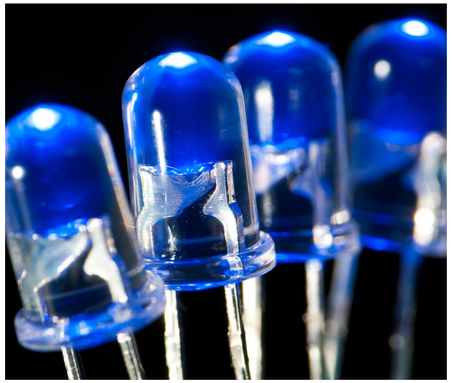
servo motor

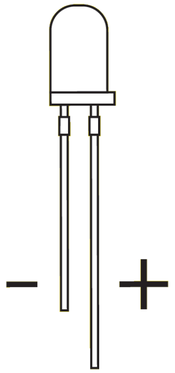
solenoid



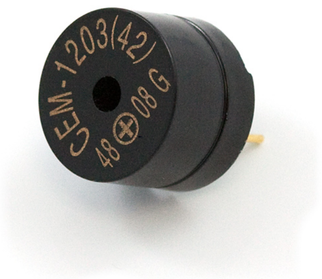
pump

ACTUATORS





LED
(light emitting diode)

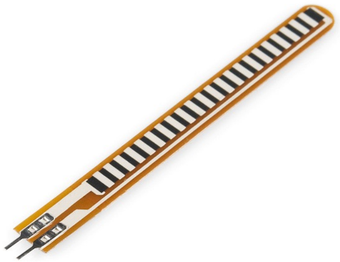


speaker

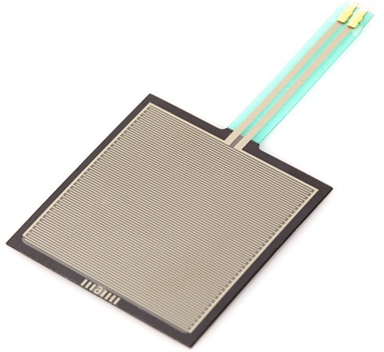
SENSORS

photocell

potentiometer



flex

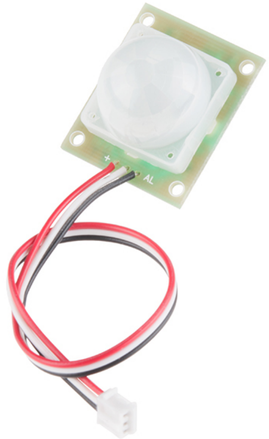


pressure

Variable resistors are resistors of which the electric resistance value can be adjusted.

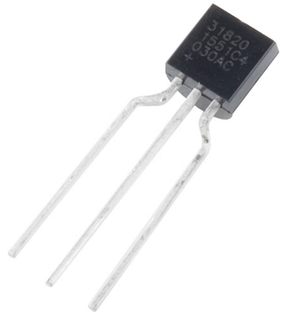
SENSORS

buttons and switches

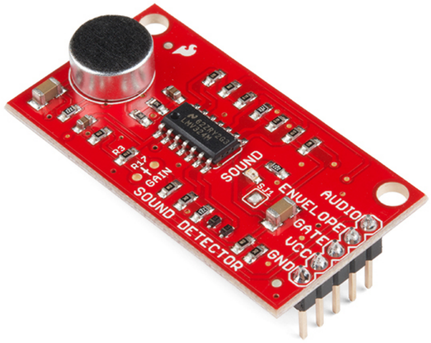


motion

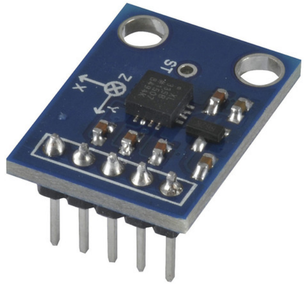
sonar (distance)



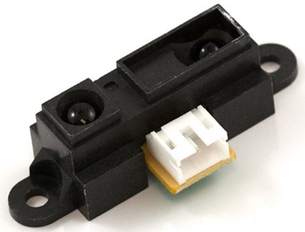
temperature

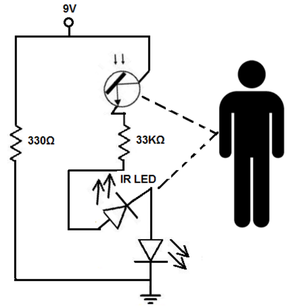


sound



acceleration, orientation, rotation

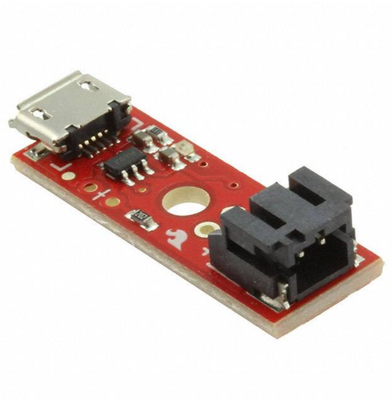




infrared (color and distance)

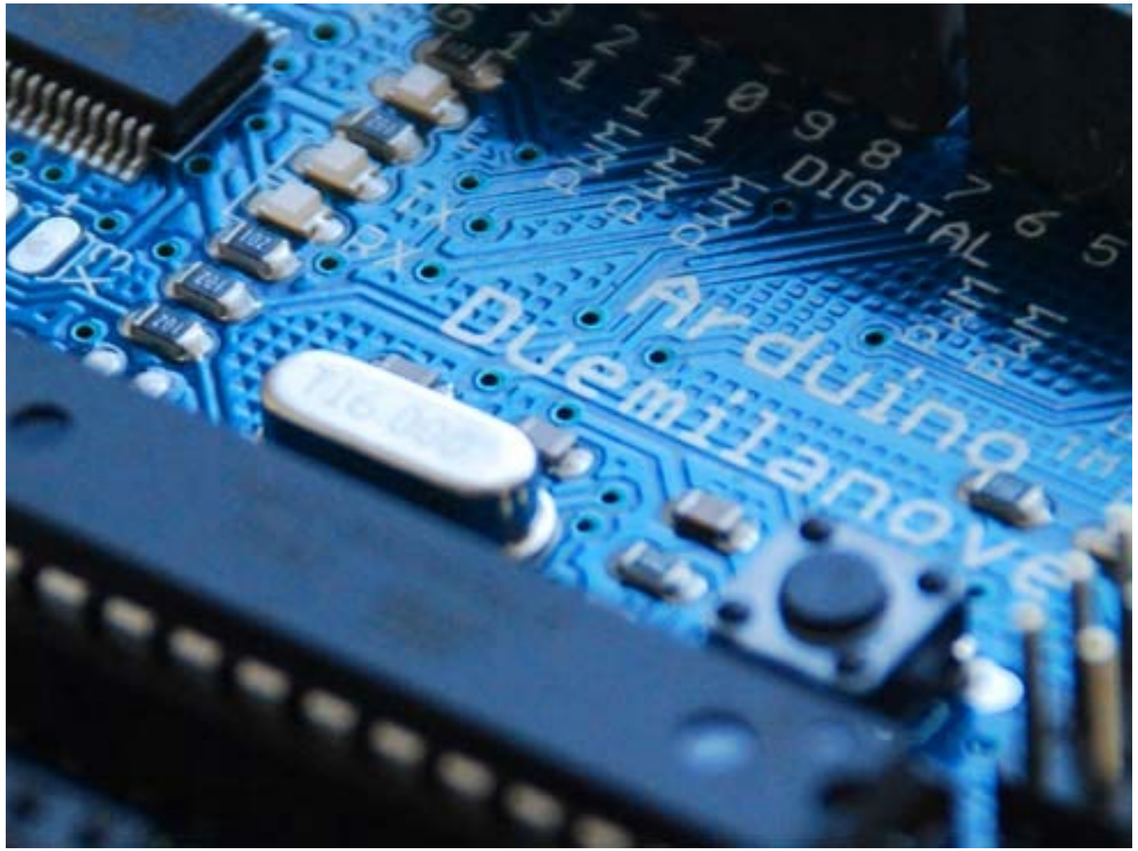
POWER SUPPLY

Benchtop power supply, voltage regulator



A lithium polymer ***(lipo) battery*** of 5000mAh 3.7V will be around 3.7V when fully charged. The minimum voltage is ~3V per cell. 5000mAh means that you can draw 5A for an hour, or 10A for 30 minutes. You can charge a lipo battery with a ***battery charger*** connected via micro usb.

An Analog-to-Digital Converter (ADC) digitizes an analog signal by assigning discrete, digital values to a defined signal value, in order to utilise the information.



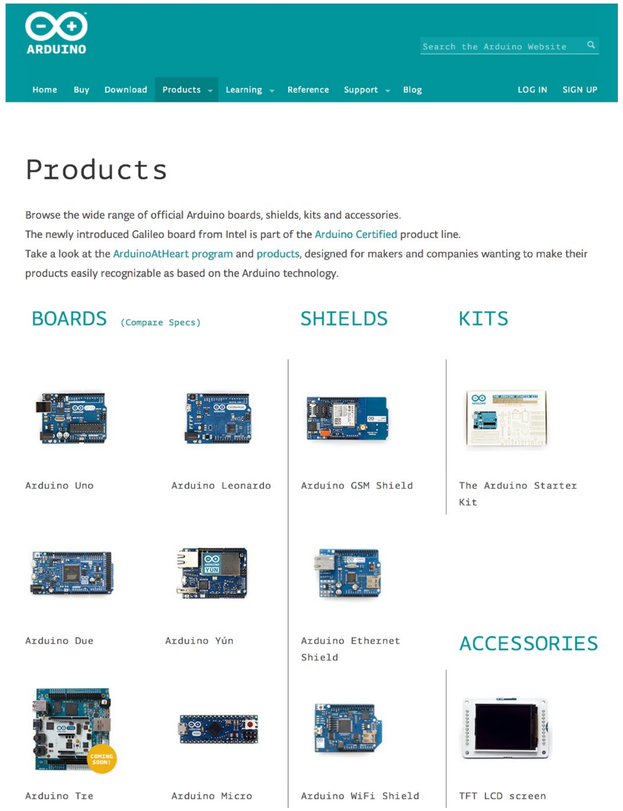
MICROPROCESSING

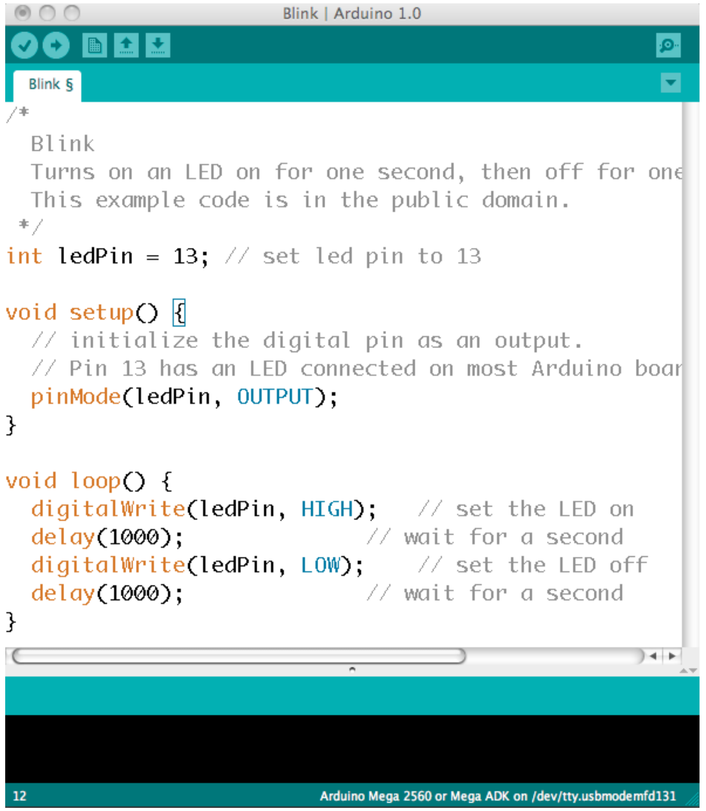
LEARNING RESOURCES AND VENDORS

Microcontrollers
http://www.arduino.cc/
http://www.raspberrypi.org/
http://www.atmel.com/products/microcontrollers/avr/
http://fab.cba.mit.edu/classes/863.13/

Tutorials
http://highlowtech.org/?p=1695

ARDUINO

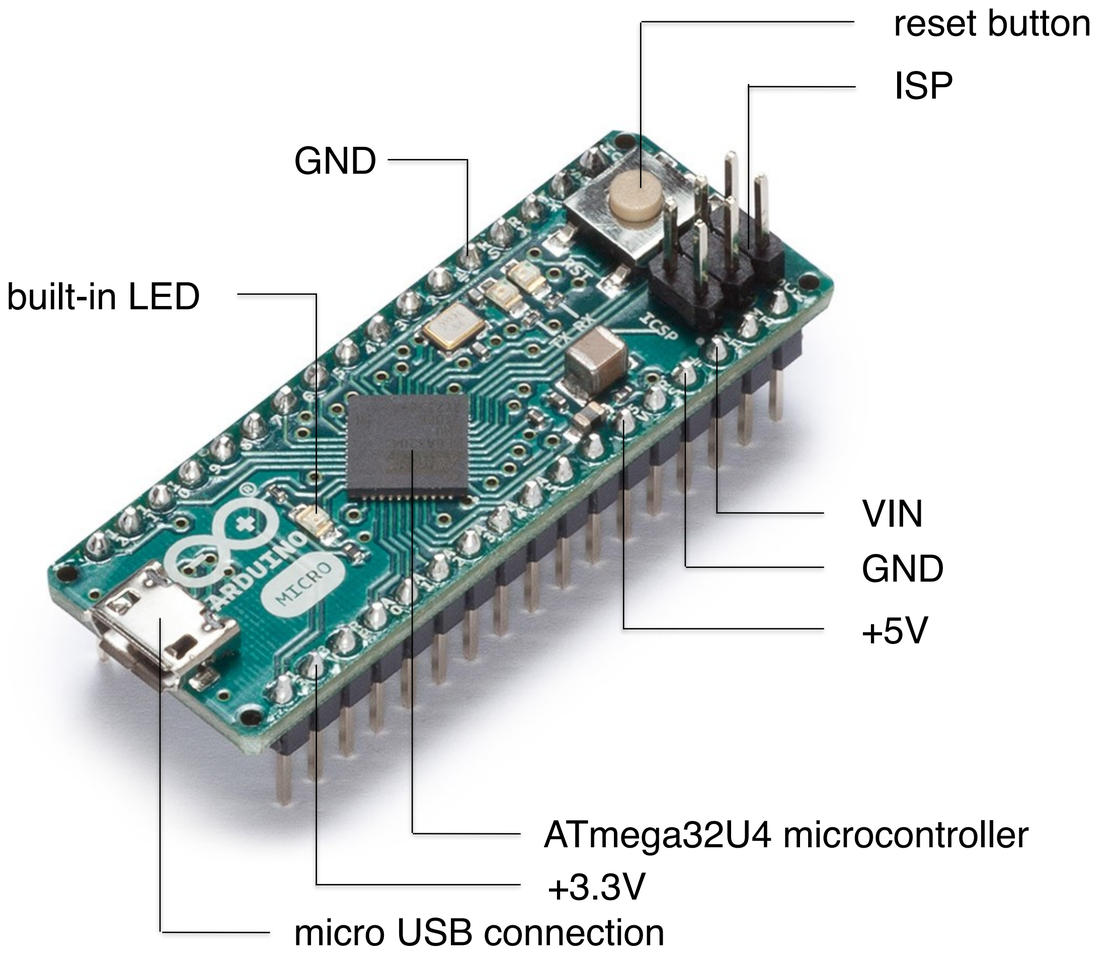


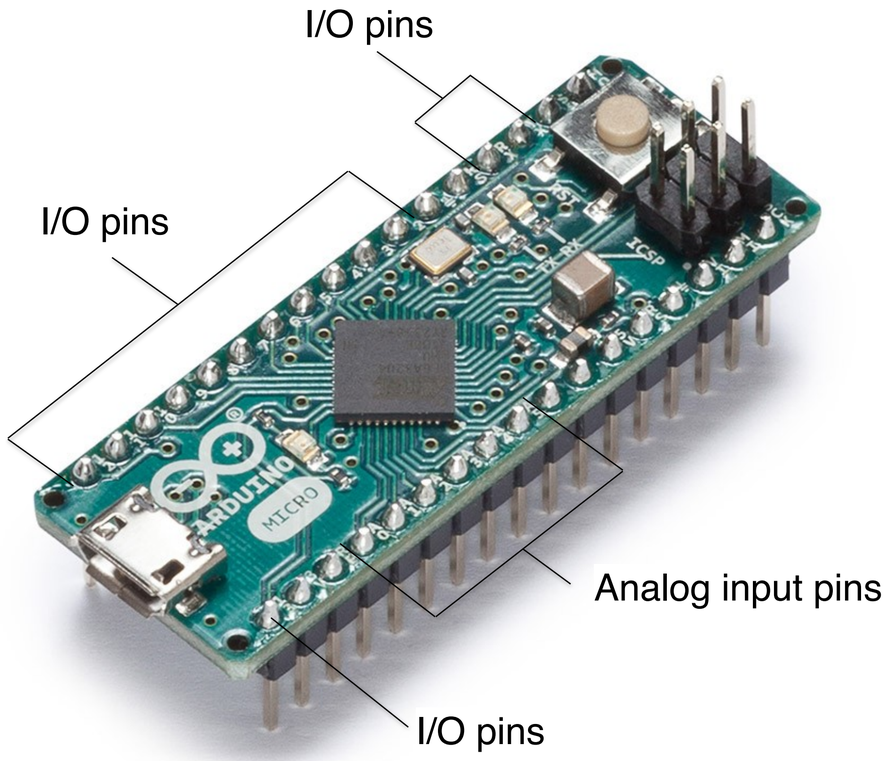


Arduino boards can read inputs and send in outputs. By sending a set of instructions to the microcontroller on the board you can tell the board what to do. All Arduino boards are open-source.

Arduino's ***Integrated Development Environment (IDE)*** allows you to write programs (in C++) and upload them to your board. Programs written in the IDE are called sketches.

ARDUINO MICRO





RASPBERRY PI

IO pins

quad-core CPU processor

SD card slot

USB 2.0

ethernet

audio/video jack

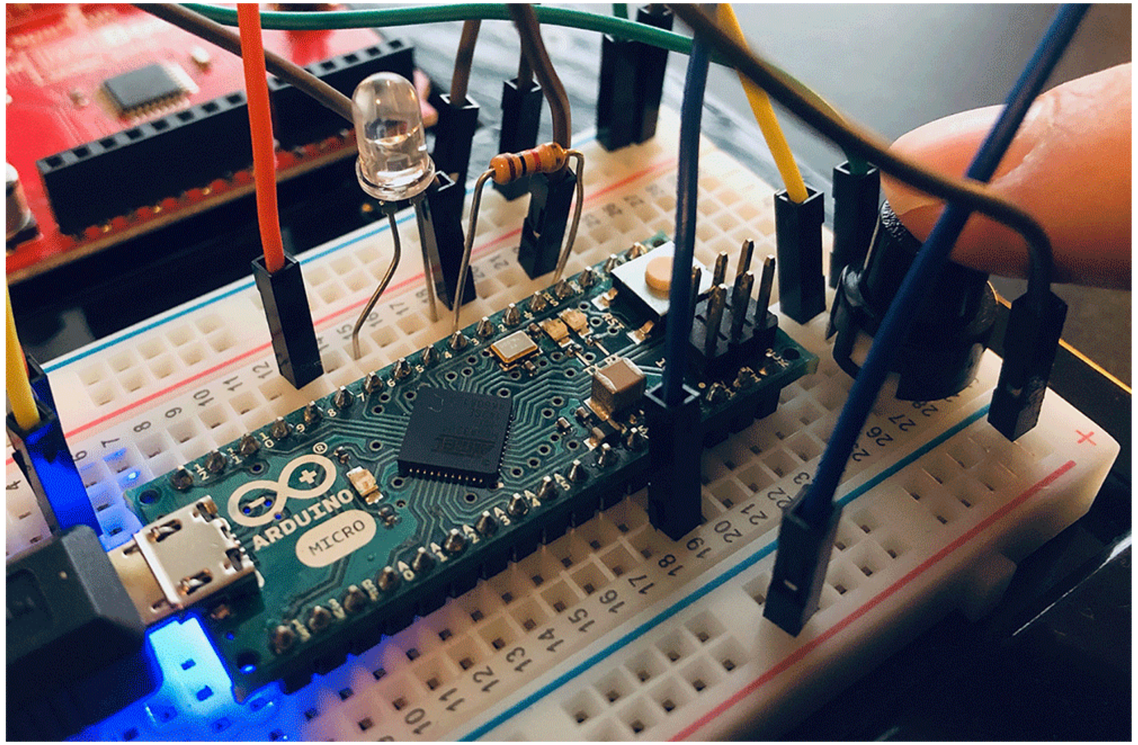
CSI camera connector

HDMI

micro USB connection

INTRODUCTION TO ARDUINO

Today's workshop: controlling an LED



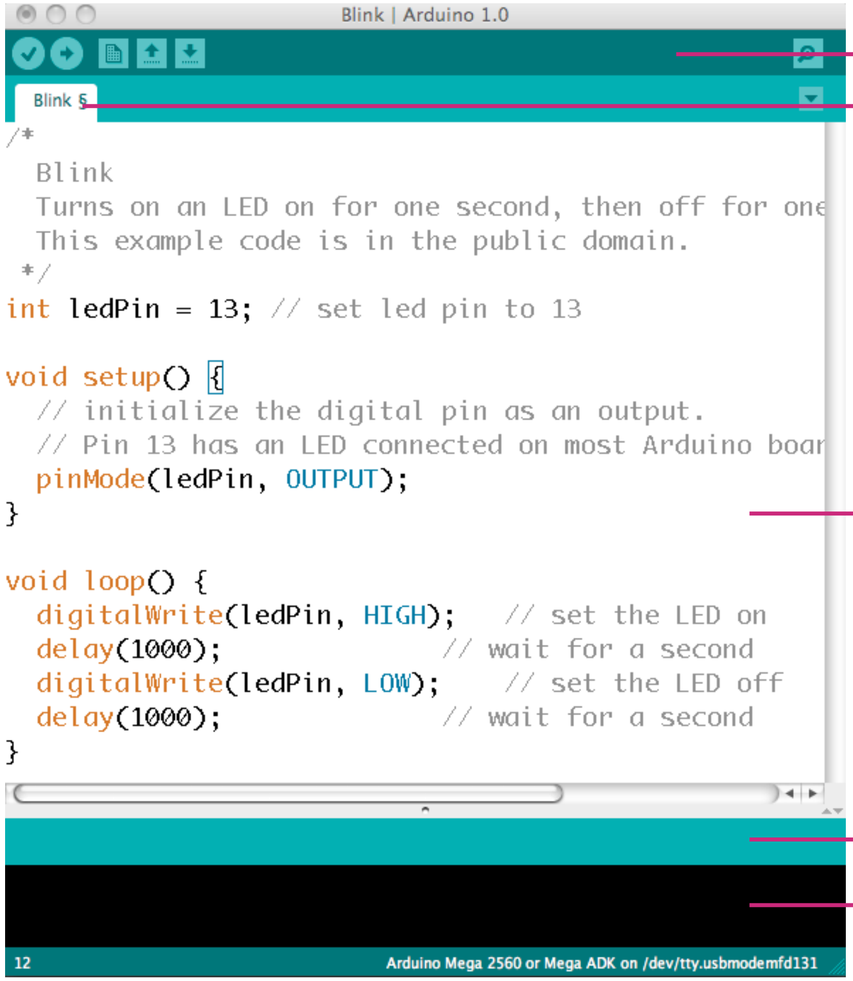
LEARNING RESOURCES AND VENDORS

IDE
https://www.arduino.cc/reference/en/
http://processing.org/
http://www.arduino.cc/
https://www.arduino.cc/reference/en/
http://www.modkit.com/

Tutorials
http://www.arduino.cc/
https://www.sparkfun.com/
http://www.adafruit.com/
http://highlowtech.org/
http://www.instructables.com/
http://bildr.org/
http://www.fabfoundation.org/
http://makezine.com/

PCB design
http://fritzing.org/home/

ARDUINO SOFTWARE (IDE)



Toolbar

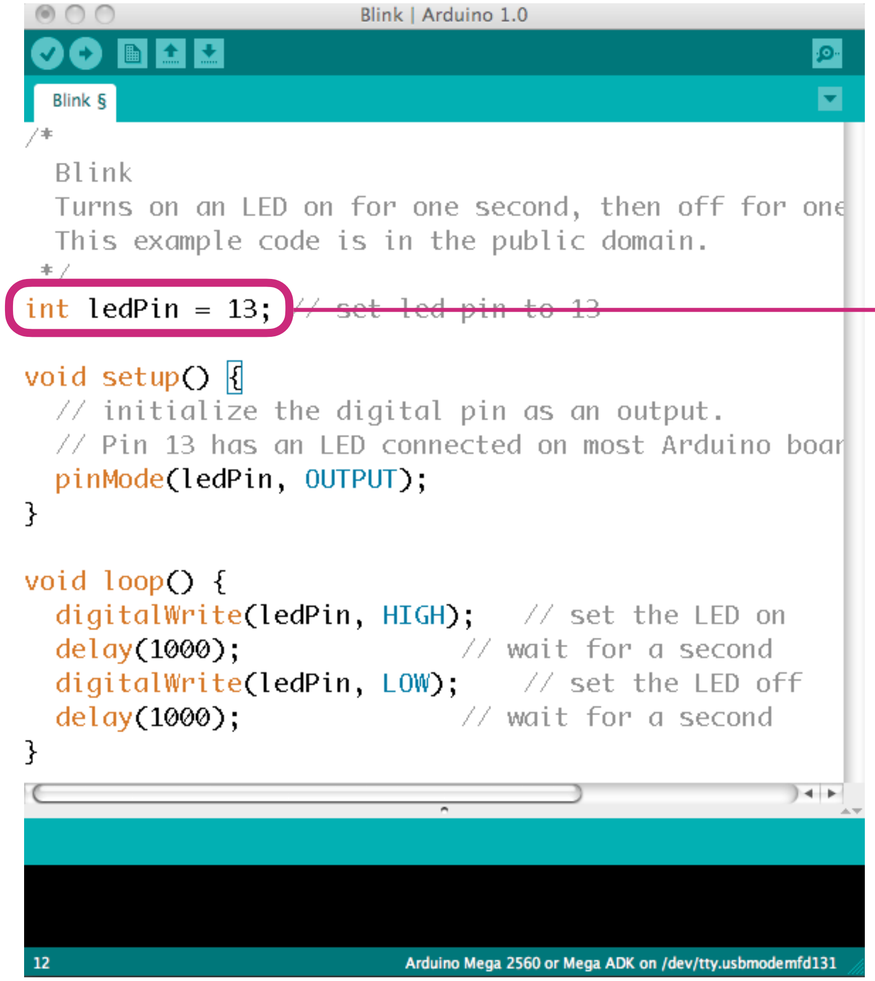
Tabs

Text editor

Message

Console

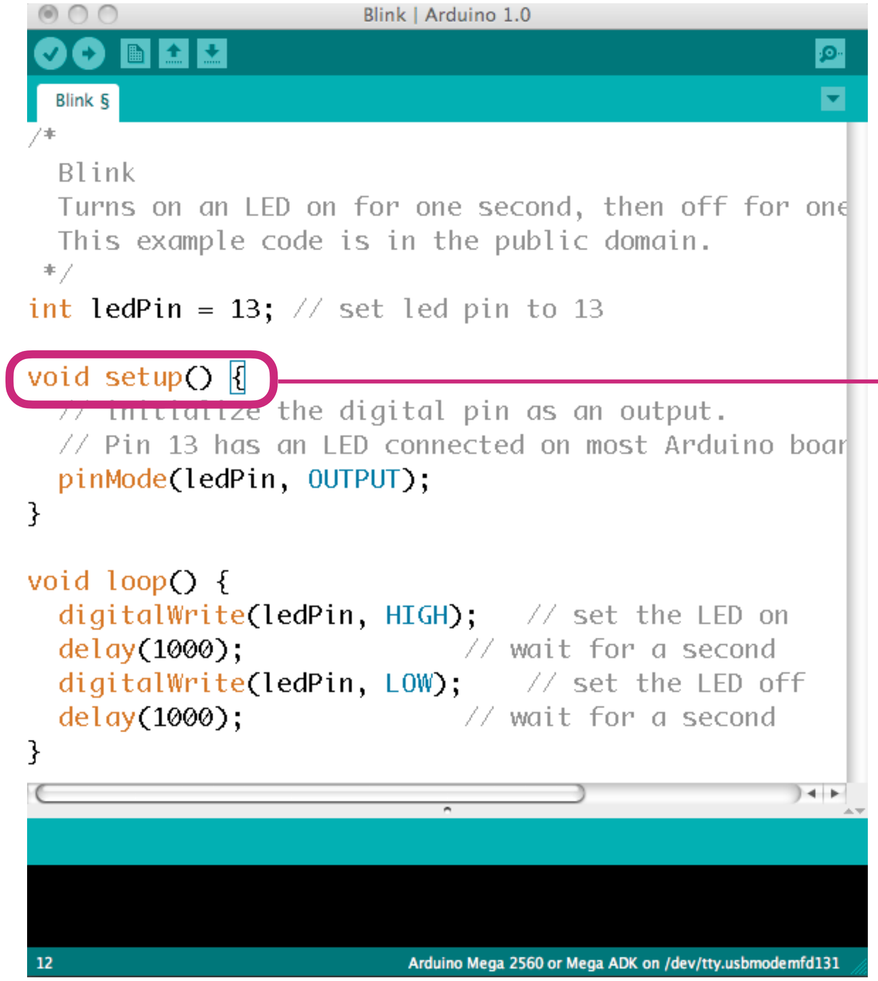
STRUCTURE



A **variable** is a place to store a piece of data. It has a name, a value, and a type.

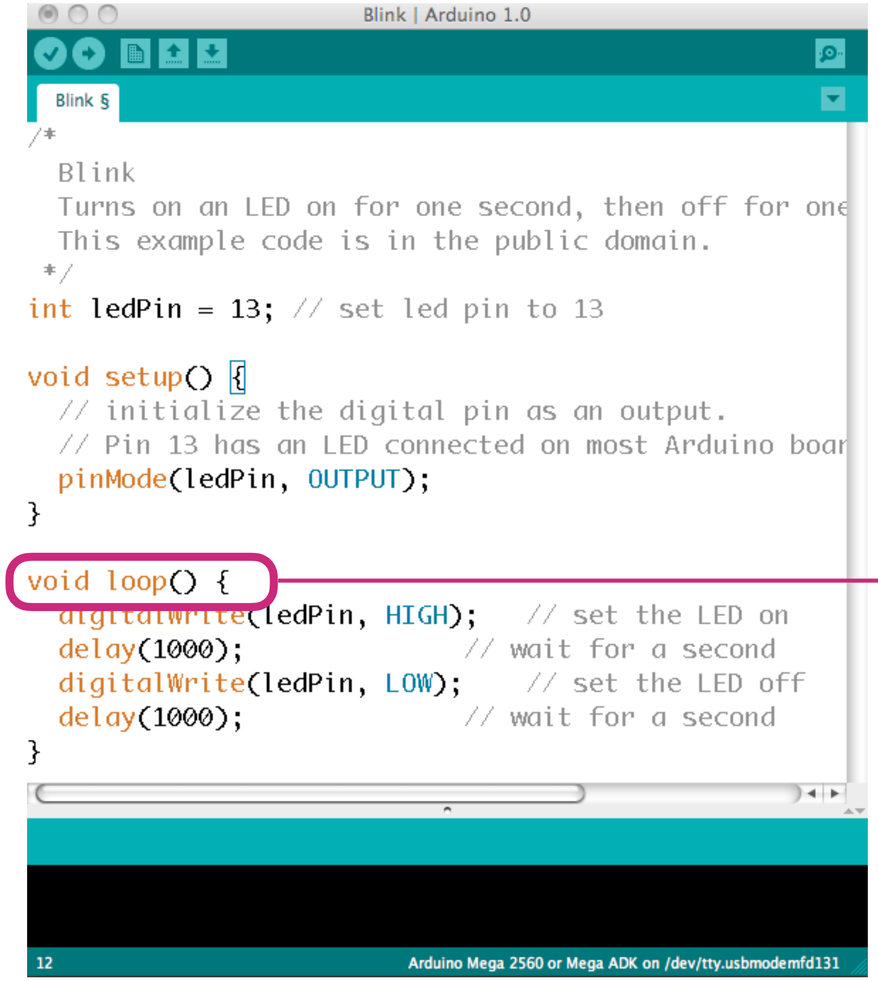
Variables can be named anywhere, but you should consider where. Variables named outside of setup() and loop() are global variables and are public, or accessible, by the entire code. Variables named inside setup() can only be accessed within the setup() function.

STRUCTURE



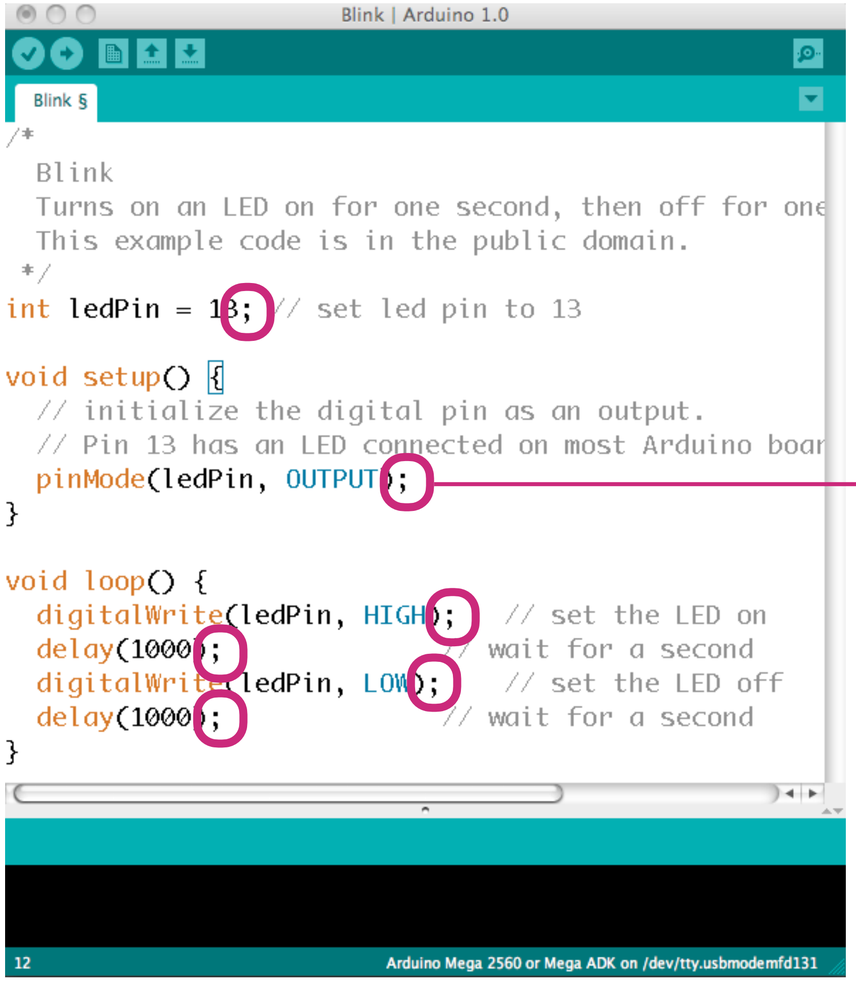
setup() is called once when a sketch starts. You use it to initialize variables, pin modes, start using libraries, etc. The setup function will only run once, after each power up or reset of the Arduino board.

STRUCTURE



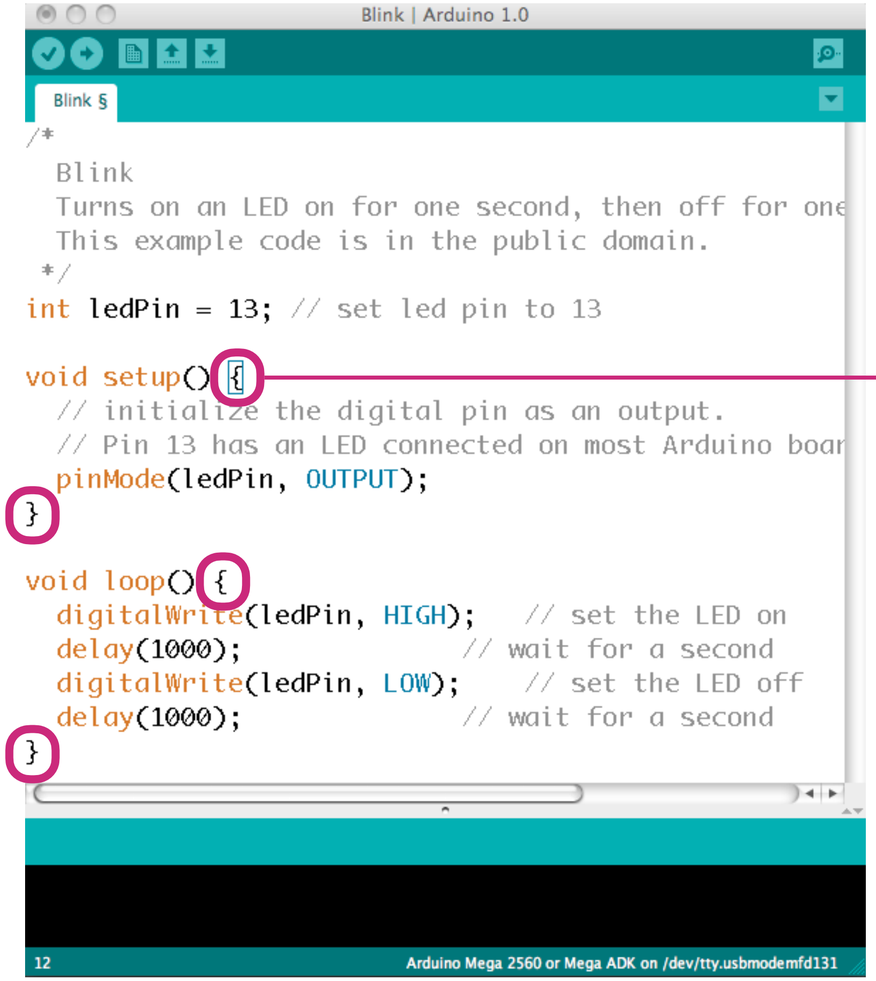
Loop () is called directly after **setup()** and the function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. You use it to actively control the Arduino board.

SYNTAX



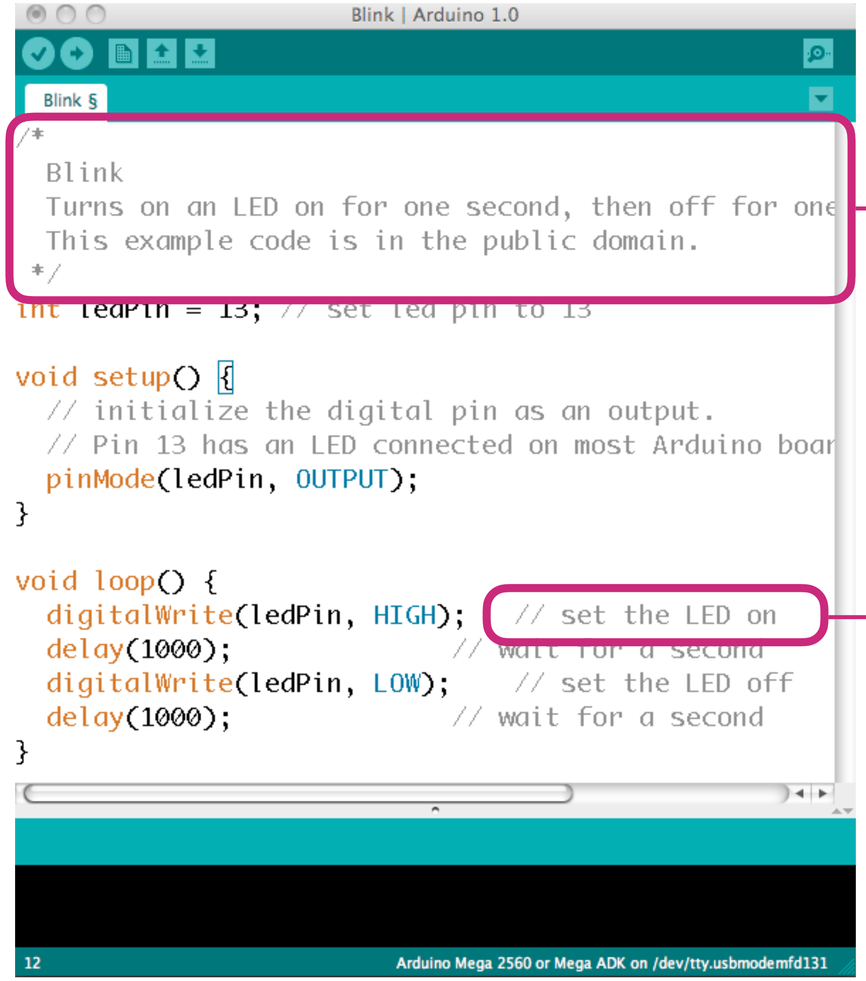
A semicolon (;) denotes the end of a statement and must be used for all expressions

SYNTAX



The curly brackets (**{** and **}**) denote the beginning and the end of each function.

SYNTAX

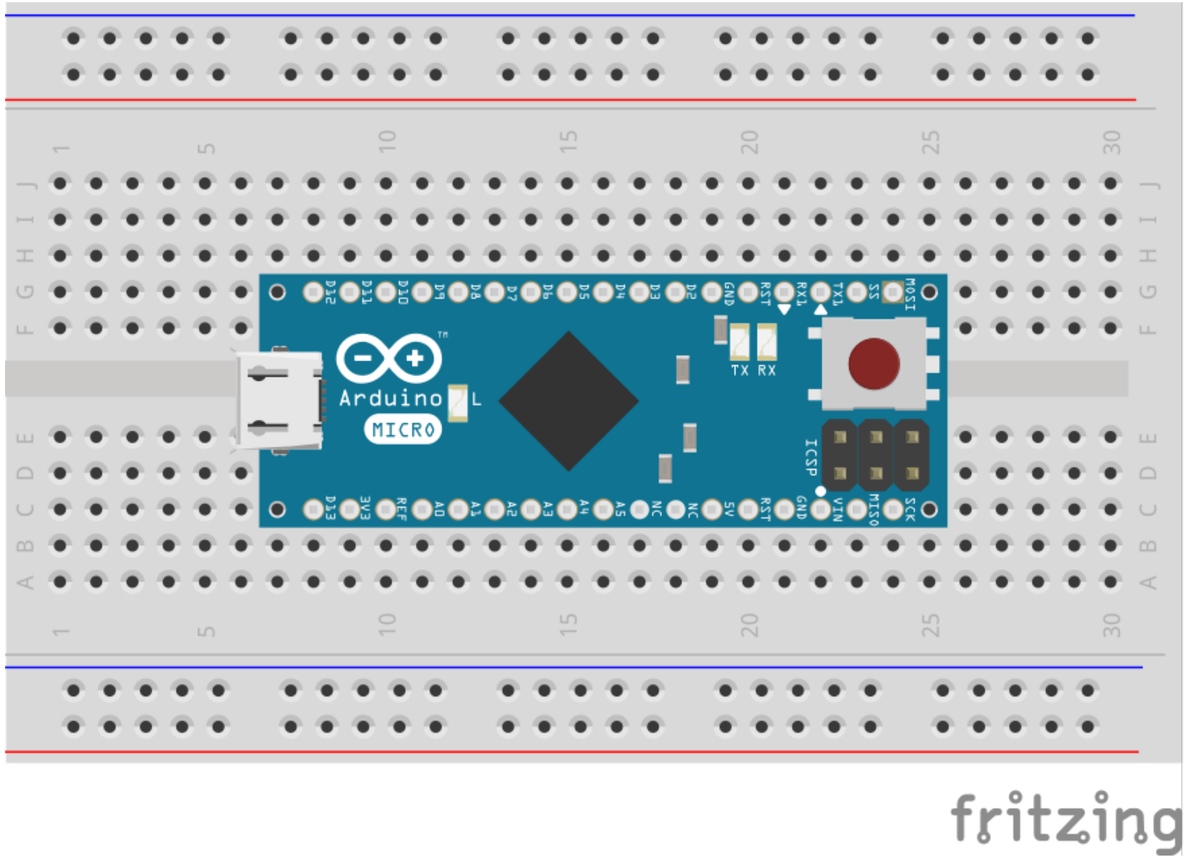


Comments are ignored by the coding environment and are extremely useful to communicate complex procedures

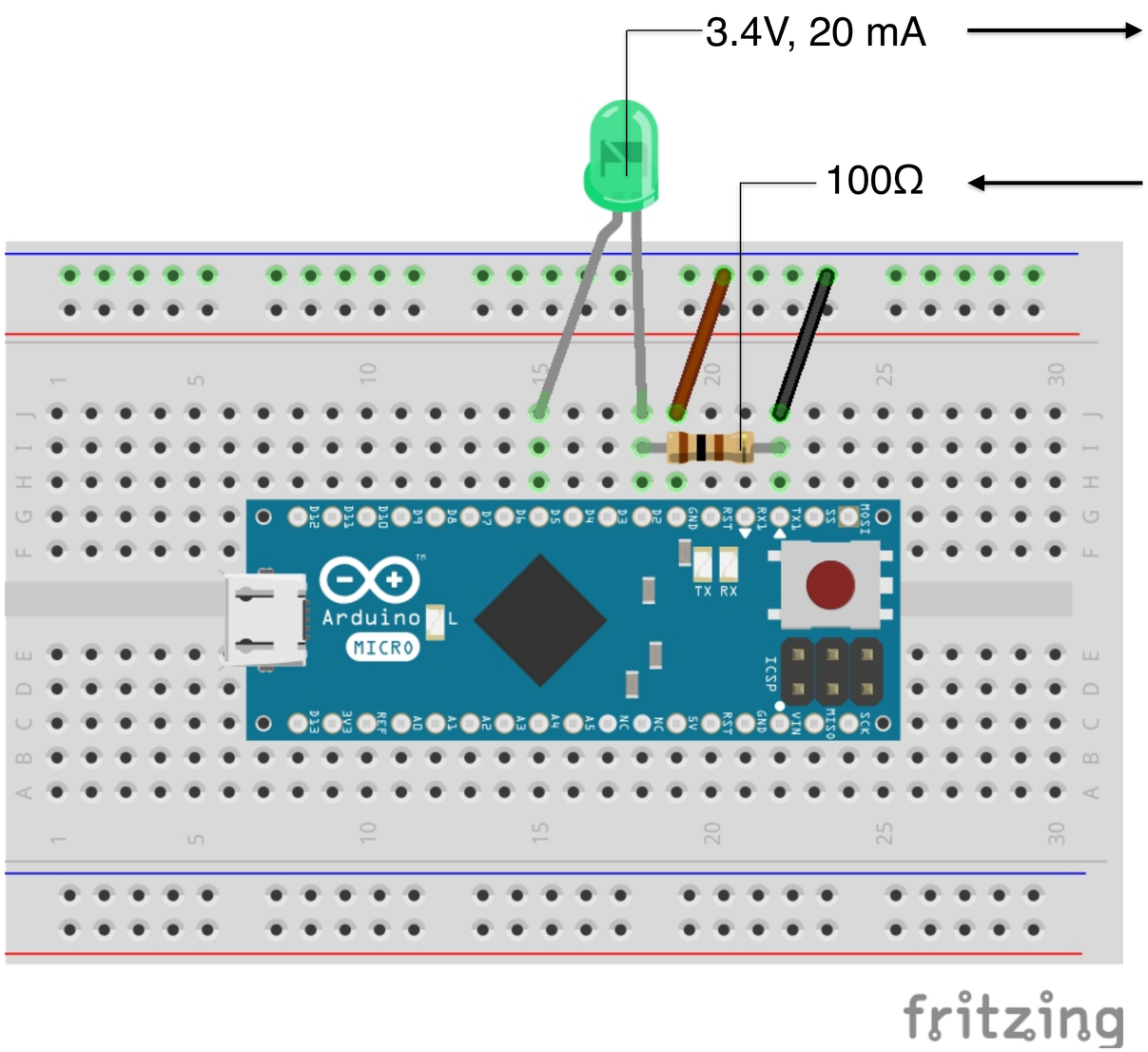
/* a forward slash followed by an asterisk allows for a continuous (multi-line) comment until the reverse ***/***

// two forward comments all text on the same line

LET’S START!



LET’S START!



Ohm’s law:
R = V / I
LED resistor:
R = (V_{SUPPLY} - V_{LED}) / I
R = (5V - 3.4V) / 20 mA = 80Ω

VARIABLES

Variables to hold data and that can be of different types, such as boolean, int, and string.

Syntax

```
int temperature = analogRead(0);
```

data type variable name operator function parameter statement terminator

or

```
int temperature = 24;
```

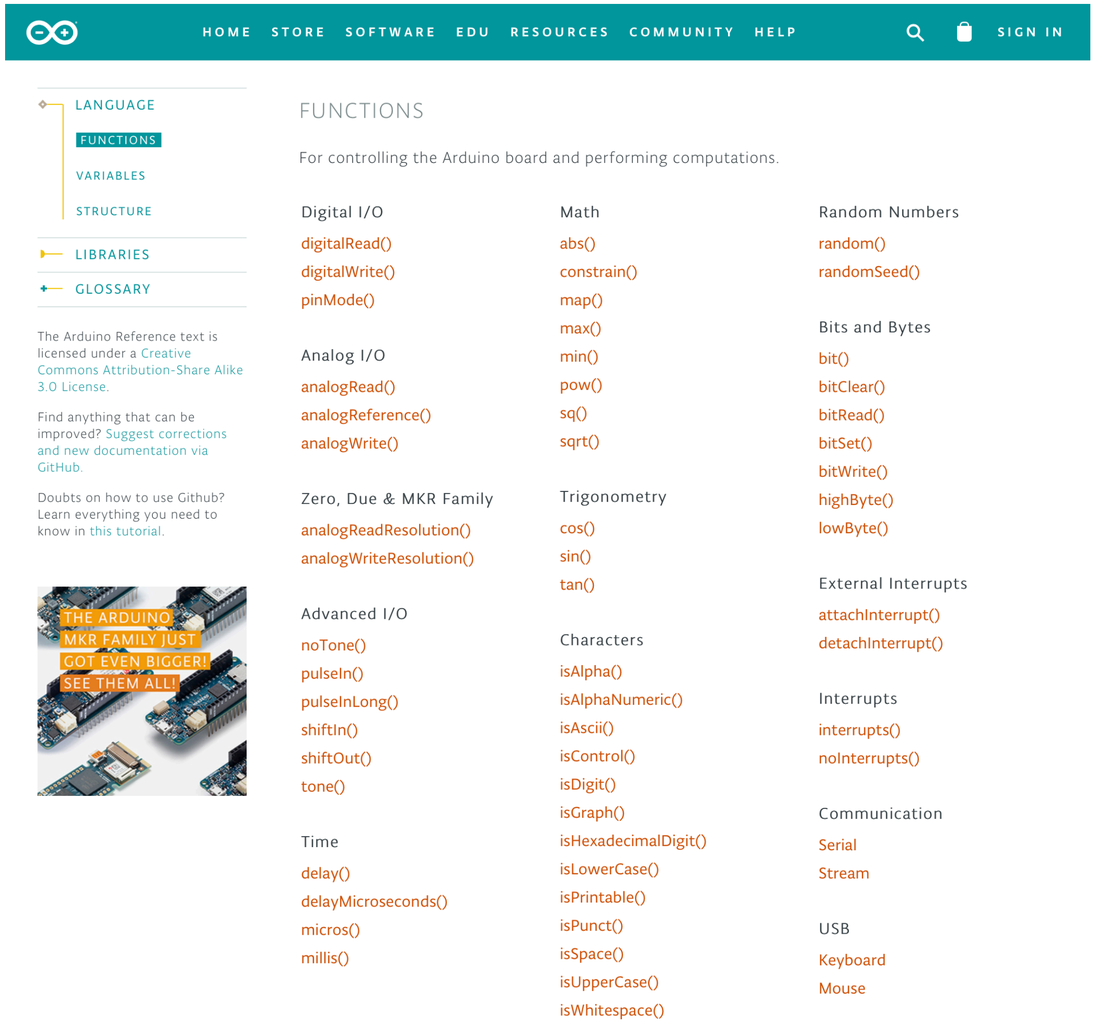
data type variable name operator value statement terminator

Note:
A variable name cannot start with a number, contain an empty space or any special characters except the underscore.

COMMON DATA TYPES

Type	Unit	Example
boolean	Is 1-bit long and can take a value of either true or false	boolean isInside = true;
char	A char is 16-bit long and can therefore store 2^{16} (= 65,536) different values. (ASCII code is the numerical representation of a character such as 'a')	char firstLetter = ‘L’;
int	Integers are 32-bit long and can define whole numbers ranging from -2,147,483,647 to 2,147,483,648.	int number = 25;
float	Floating point numbers are real fractional numbers and can have decimal values and can range between -3.40282347E+38 and 3.40282347E+38.	float pi = 3.14;
long	A long is 64-bits and can define large whole numbers ranging from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807. Arduino also uses unsigned longs, which are 32-bit and won't store negative numbers, making their range from 0 to 4,294,967,295.	long a = 100000; unsigned long b = 200000;
string	A string is a collection of characters used for words and phrases.	String myName = “Lara”;

CALLING FUNCTIONS



Arduino has numerous built-in functions that you can “call”.

<u>Syntax</u>
functionName(*argument1*, *argument2*)

The function name is the identifier by which the function can be called.

<u>Parameters</u>
Arguments are parameters passed to the function, which can be of any data type.

<u>Returns</u>
A function either returns a value (of any data type) or does not return anything.

https://www.arduino.cc/reference/en/

FUNCTIONS

pinMode() (digital pins only)

Configures the specified pin to behave either as an input or an output (either receive signals, or send them)

Syntax

pinMode(*pin*, *mode*)

Parameters

pin: the number of the pin whose mode you wish to set
mode: either INPUT or OUTPUT

Returns

nothing

FUNCTIONS

digitalWrite()

Writes a HIGH or LOW (1 or 0) value to a digital pin.

Syntax

digitalWrite(*pin*, *value*)

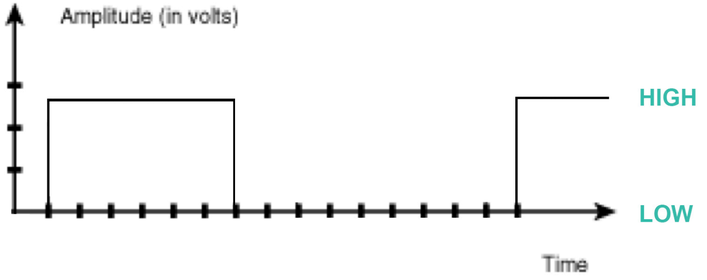
Parameters

pin: the pin number

value: HIGH or LOW (1 or 0)

Returns

Nothing



analogWrite()

Writes an analog value (PWM wave) to a pin. Can be used to light a LED at varying brightnesses or drive a motor at various speeds. After a call to analogWrite(), the pin will generate a steady square wave of the specified duty cycle until the next call to analogWrite() (or a call to digitalRead() or digitalWrite() on the same pin).

You do not need to call pinMode() to set the pin as an output before calling analogWrite().

The analogWrite function has nothing whatsoever to do with the analog pins or the analogRead function.

Syntax

analogWrite(*pin*, *value*)

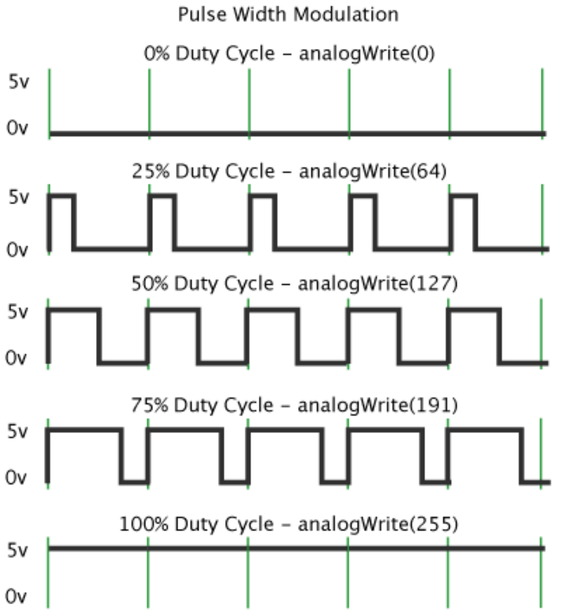
Parameters

pin: the pin to write to.

value: the duty cycle: between 0 (always off) and 255 (always on).

Returns

Nothing



FUNCTIONS

digitalRead()

Reads the value from a specified digital pin, either HIGH or LOW (1 or 0).

Syntax

digitalRead(*pin*)

Parameters

pin: the number of the digital pin you want to read (int)

Returns

HIGH or LOW (1 or 0)

analogRead()

Reads the value from the specified analog pin. An Arduino board contains a 10-bit analog to digital converter. This means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023.

Syntax

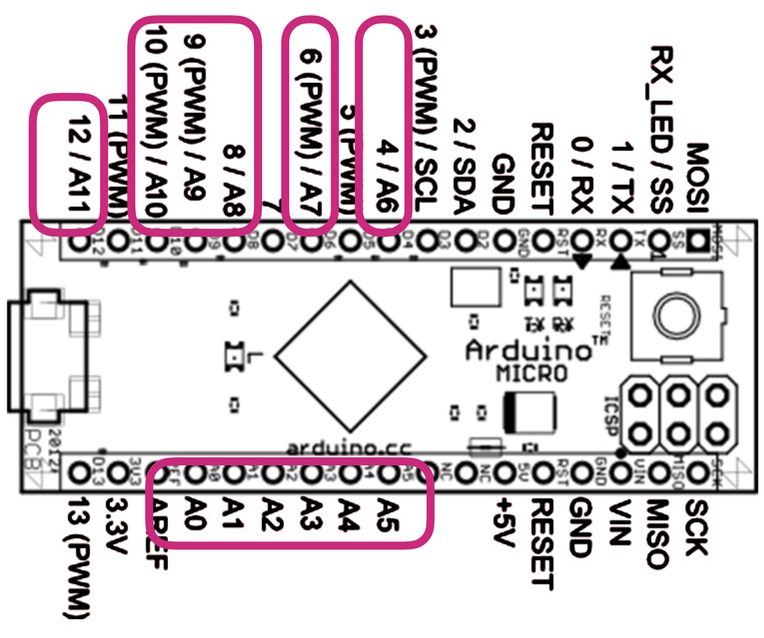
analogRead(*pin*)

Parameters

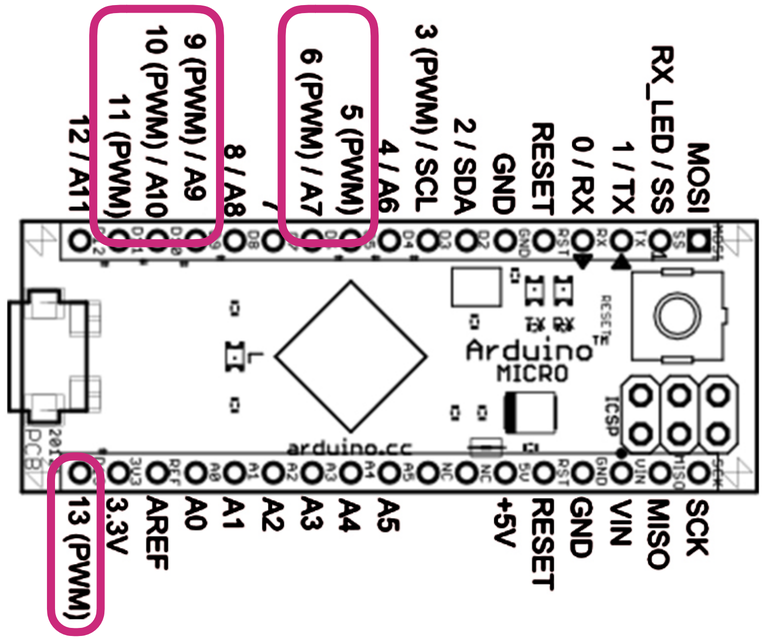
pin: the number of the analog input pin to read from

Returns

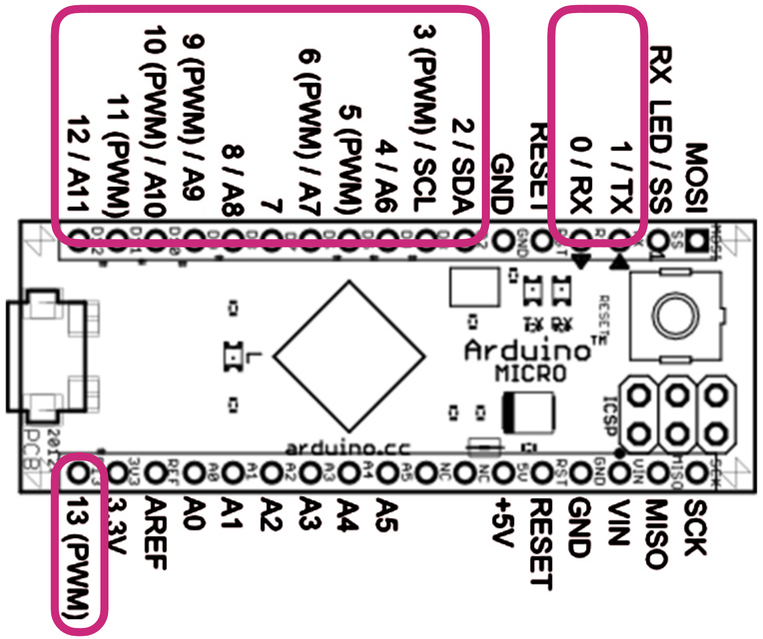
int (0 to 1023)



Analog pins are labeled A0, A1, A2 etc…
They can be used with the **analogRead()** function.

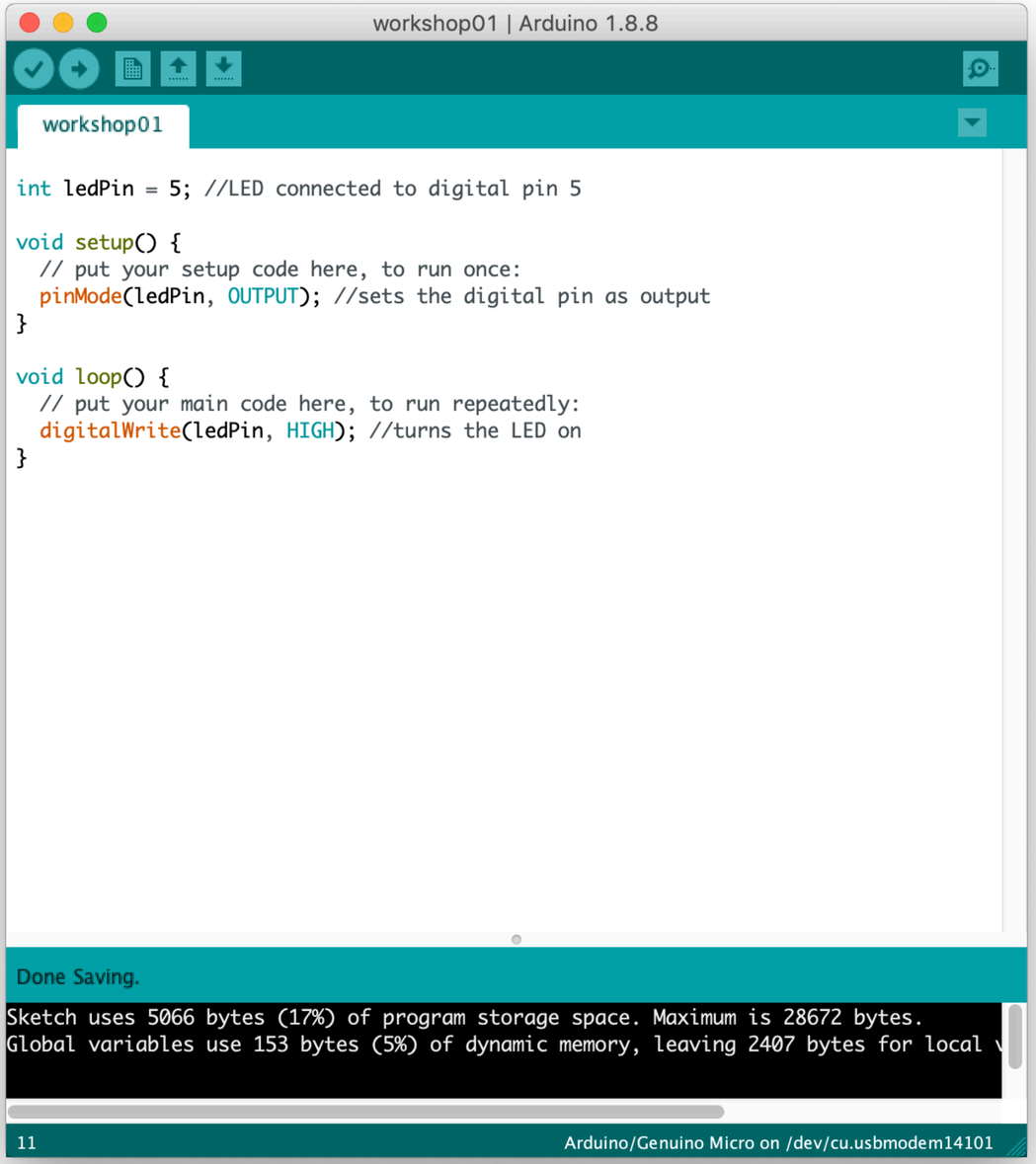


The **PWM pins** are labeled with the **~** sign.
They can be used with the **analogWrite()** function.

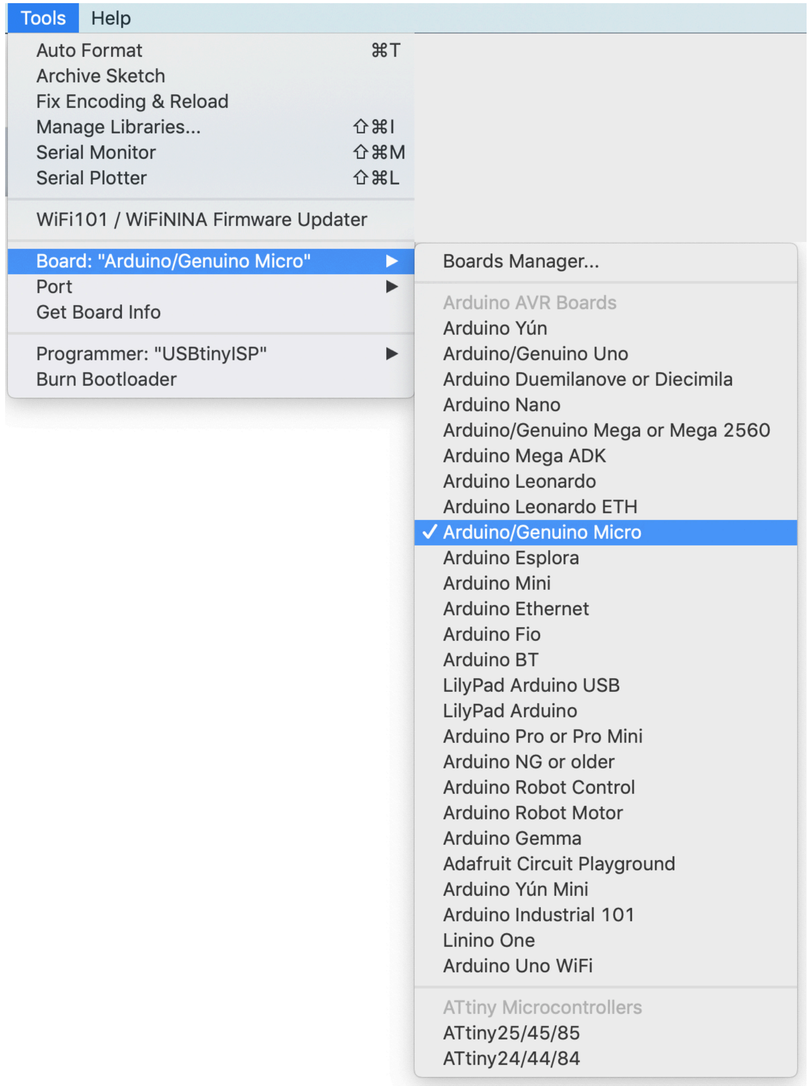


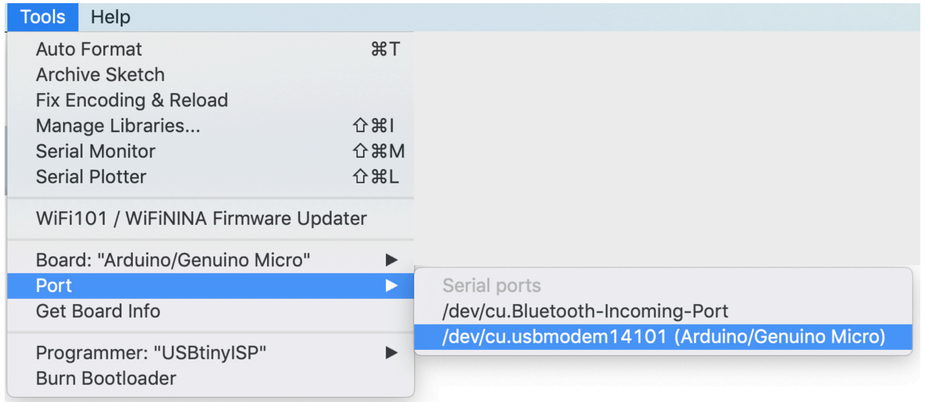
The **digital pins** are labeled 0, 1, 2 etc….
They can be used with the **digitalWrite()** and **digitalRead()** function.

FUNCTIONS


workshop01 | Arduino 1.8.8
workshop01
int ledPin = 5; //LED connected to digital pin 5
void setup() {
// put your setup code here, to run once:
pinMode(ledPin, OUTPUT); //sets the digital pin as output
}
void loop() {
// put your main code here, to run repeatedly:
digitalWrite(ledPin, HIGH); //turns the LED on
}
Done Saving.
Sketch uses 5066 bytes (17%) of program storage space. Maximum is 28672 bytes.
Global variables use 153 bytes (5%) of dynamic memory, leaving 2407 bytes for local
11
Arduino/Genuino Micro on /dev/cu.usbmodem14101

SELECTING BOARD AND PORT





FUNCTIONS

delay()

Pauses the program for the amount of time (in milliseconds) specified as parameter. (1 second = 1000 ms)

Syntax

delay(*ms*)

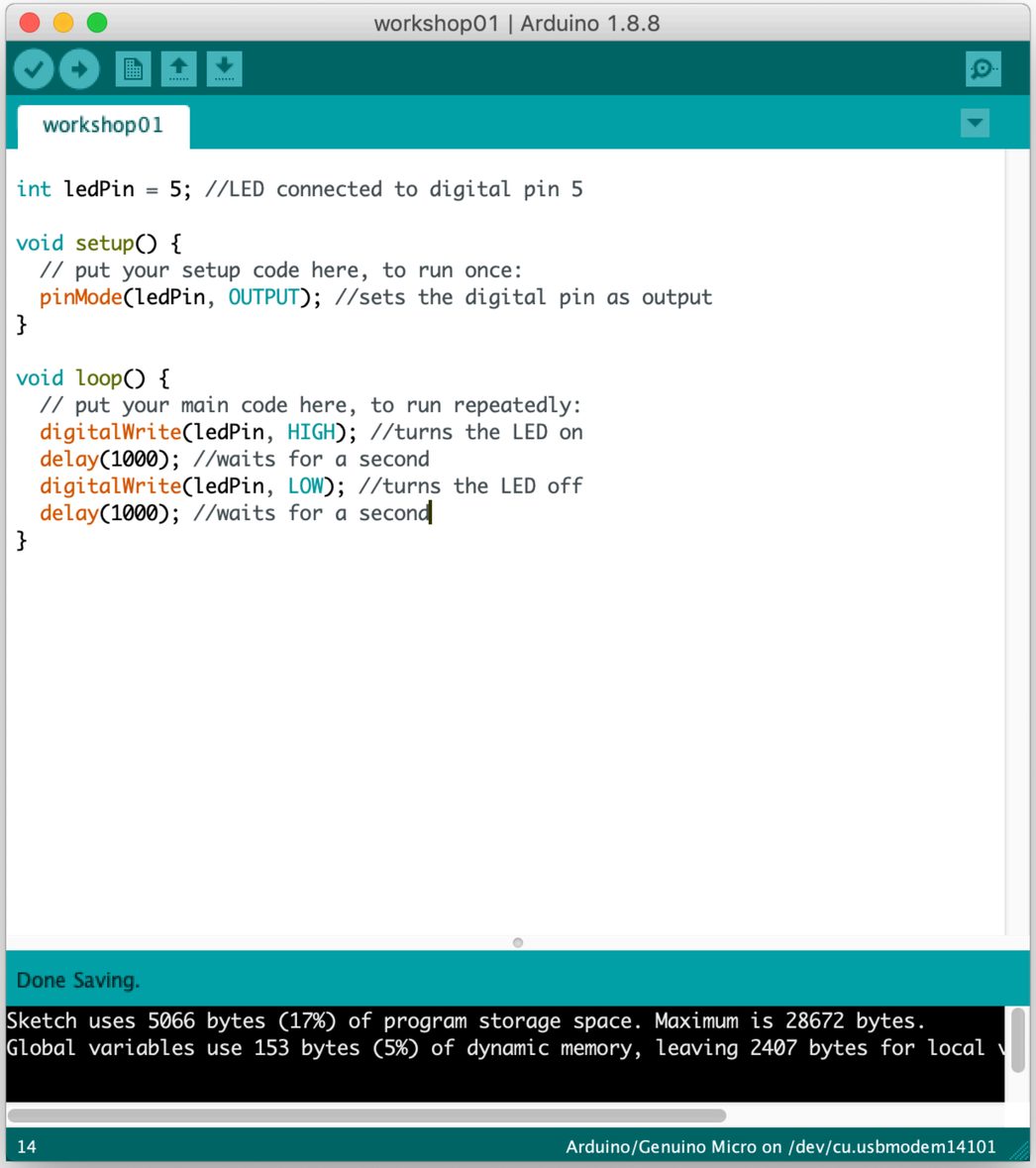
Parameters

ms: the number of milliseconds to pause

Returns

nothing

FUNCTIONS


workshop01 | Arduino 1.8.8
workshop01
int ledPin = 5; //LED connected to digital pin 5
void setup() {
// put your setup code here, to run once:
pinMode(ledPin, OUTPUT); //sets the digital pin as output
}
void loop() {
// put your main code here, to run repeatedly:
digitalWrite(ledPin, HIGH); //turns the LED on
delay(1000); //waits for a second
digitalWrite(ledPin, LOW); //turns the LED off
delay(1000); //waits for a second
}
Done Saving.
Sketch uses 5066 bytes (17%) of program storage space. Maximum is 28672 bytes.
Global variables use 153 bytes (5%) of dynamic memory, leaving 2407 bytes for local
14
Arduino/Genuino Micro on /dev/cu.usbmodem14101

FUNCTIONS

Serial.begin()

Sets the data rate in bits per second (baud) for serial data transmission. For communicating with the computer, use one of these rates: 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, or 115200.

Syntax

Serial.begin(*speed*)

Parameters

speed: in bits per second (baud) - long

Returns

Nothing

Serial.print() and Serial.println()

Prints data to the serial port as human-readable ASCII text. This command can take many forms. Numbers are printed using an ASCII character for each digit. Floats are similarly printed as ASCII digits, defaulting to two decimal places. Characters and strings are sent as is. For example:

Serial.print(78) gives "78"
Serial.print(1.23456) gives "1.23”
Serial.print('N') gives "N"
Serial.print("Hello world.") gives "Hello world.”

Serial.print() prints on the same line:
Serial.print(“WORD”) gives “WORDWORDWORD”

Serial.println() prints on a returned line:
Serial.println(“WORD”) gives “WORD
WORD
WORD”

Syntax

Serial.print(*val*)
Serial.print(*val, format*)

Serial.println(*val*)
Serial.println(*val, format*)

Parameters

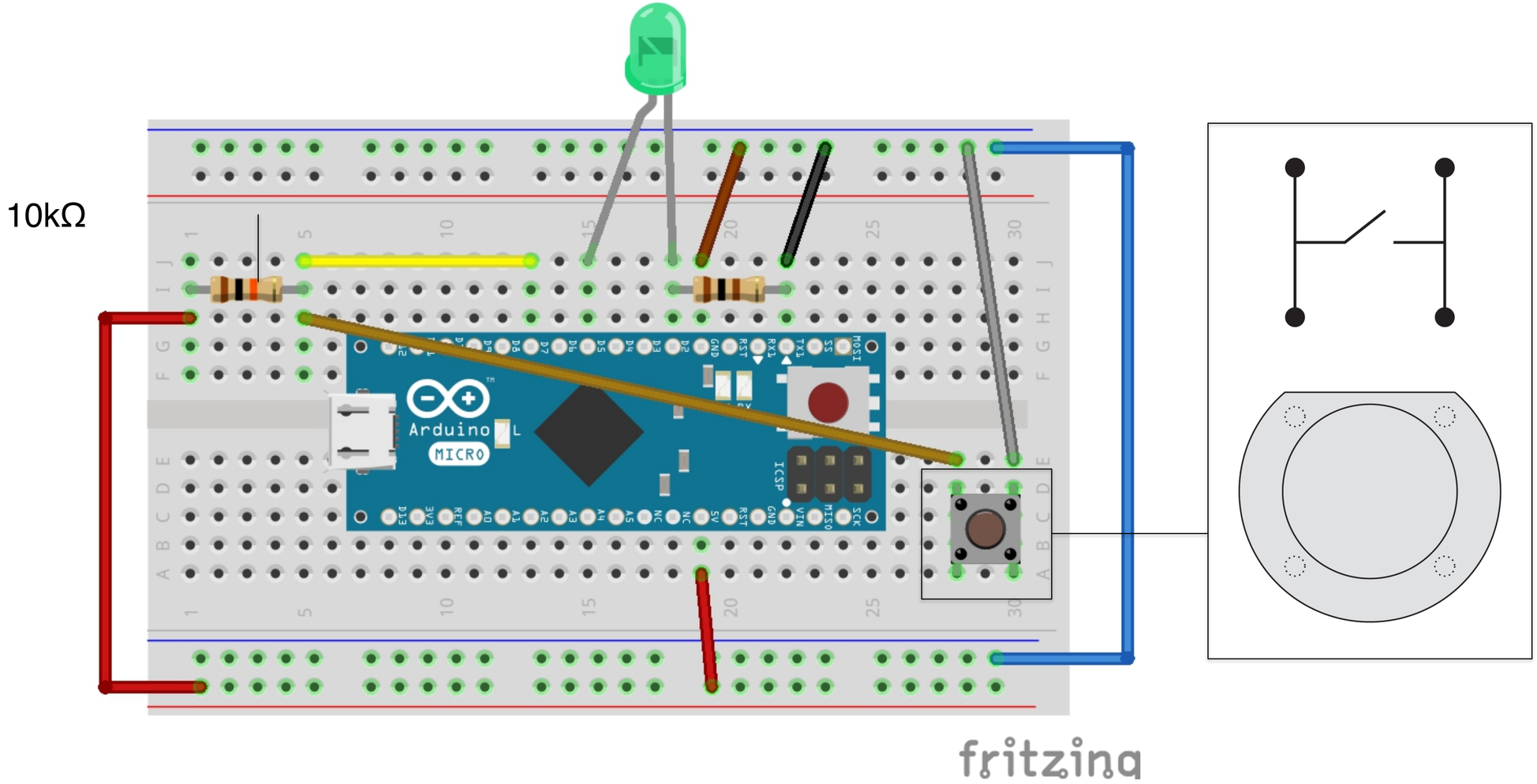
val: the value to print - any data type

format: specifies the number base (for integral data types) or number of decimal places (for floating point types)

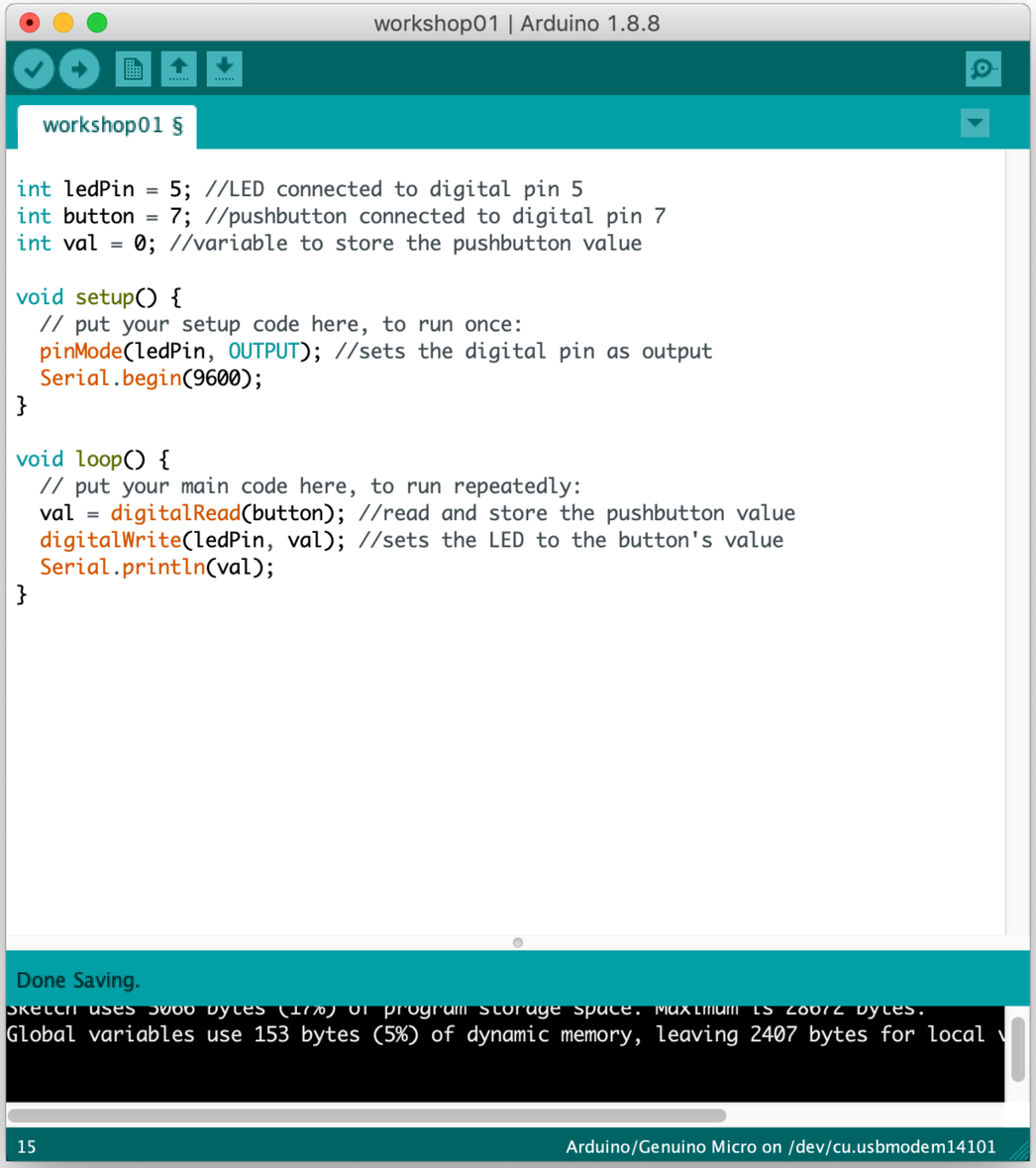
Returns

Nothing

FUNCTIONS

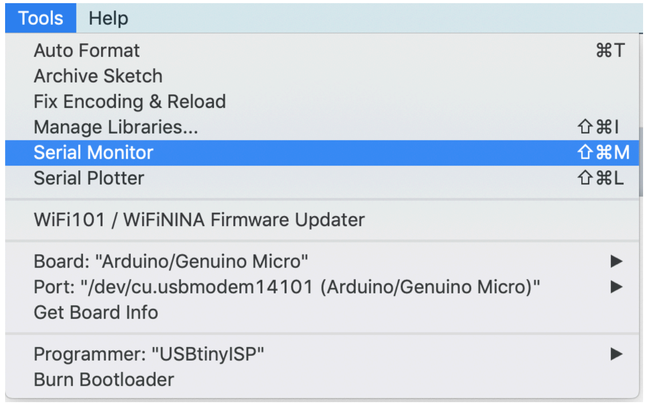


FUNCTIONS

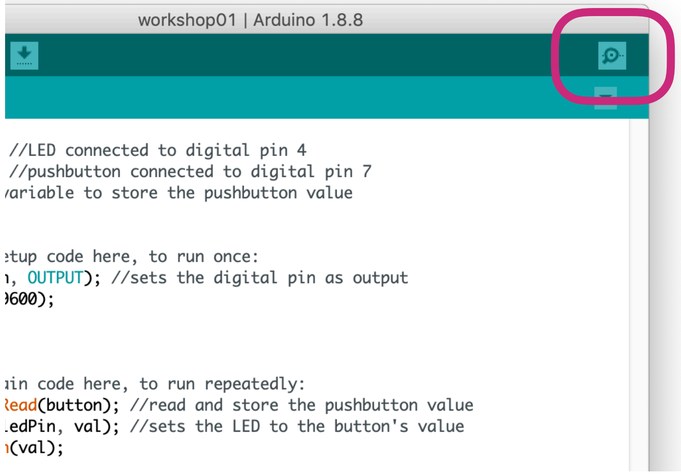


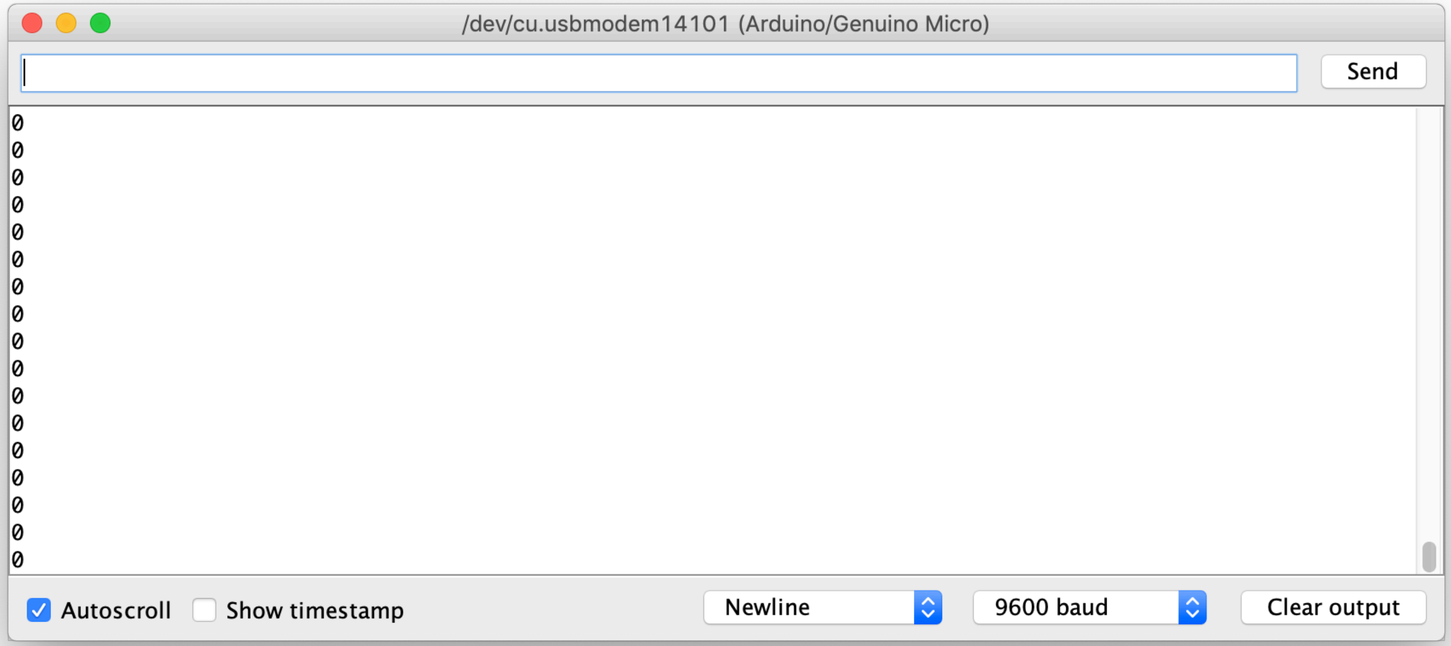
Using a button with a pull-up resistor (as in this setup), the value will be read as 1 when the button is not pushed, and 0 when the button is pushed

SERIAL MONITOR



or





The serial monitor shows Serial Data

LOGICAL AND RELATIONAL STATEMENTS

Logical operations define the truthfulness of something. For example, the word ‘if’ represents a guess that needs to be tested.

If statements

Syntax

```
if (condition) {
  // do thing A
}
else if (other condition){
  // do thing B
}
else {
  // do thing C
}
```

Conditions

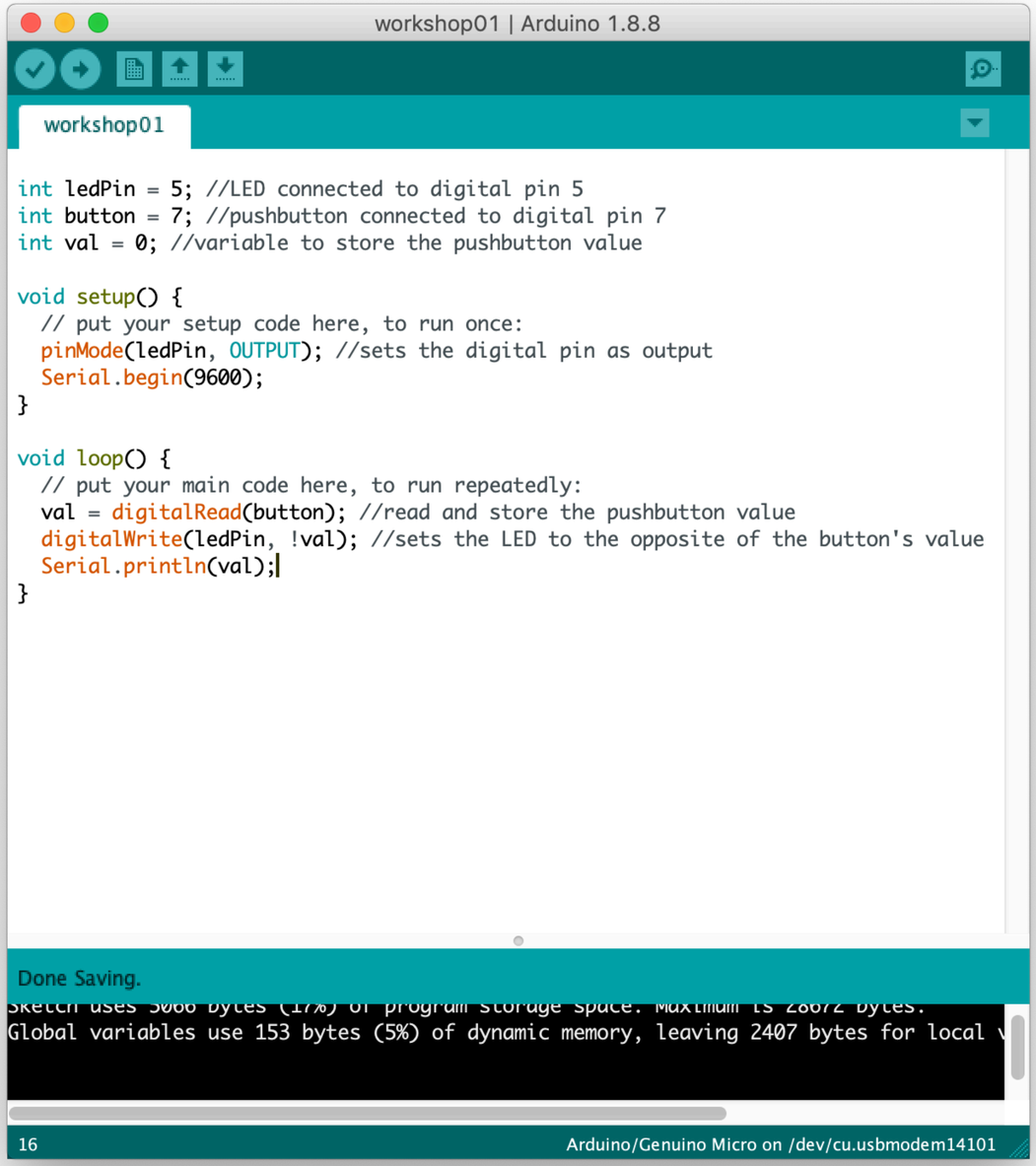
Syntax

```
if(a==b)  // if a is equal to b (double equal sign!)
if(a!=b)  // if a is not equal to b
if(a>b)   // if a is greater than b
if(a>=b)  // if a is grater than or equal to b
if(a<b)   // if a is less than b
if (a<=b) // if a is less than or equal to b
```

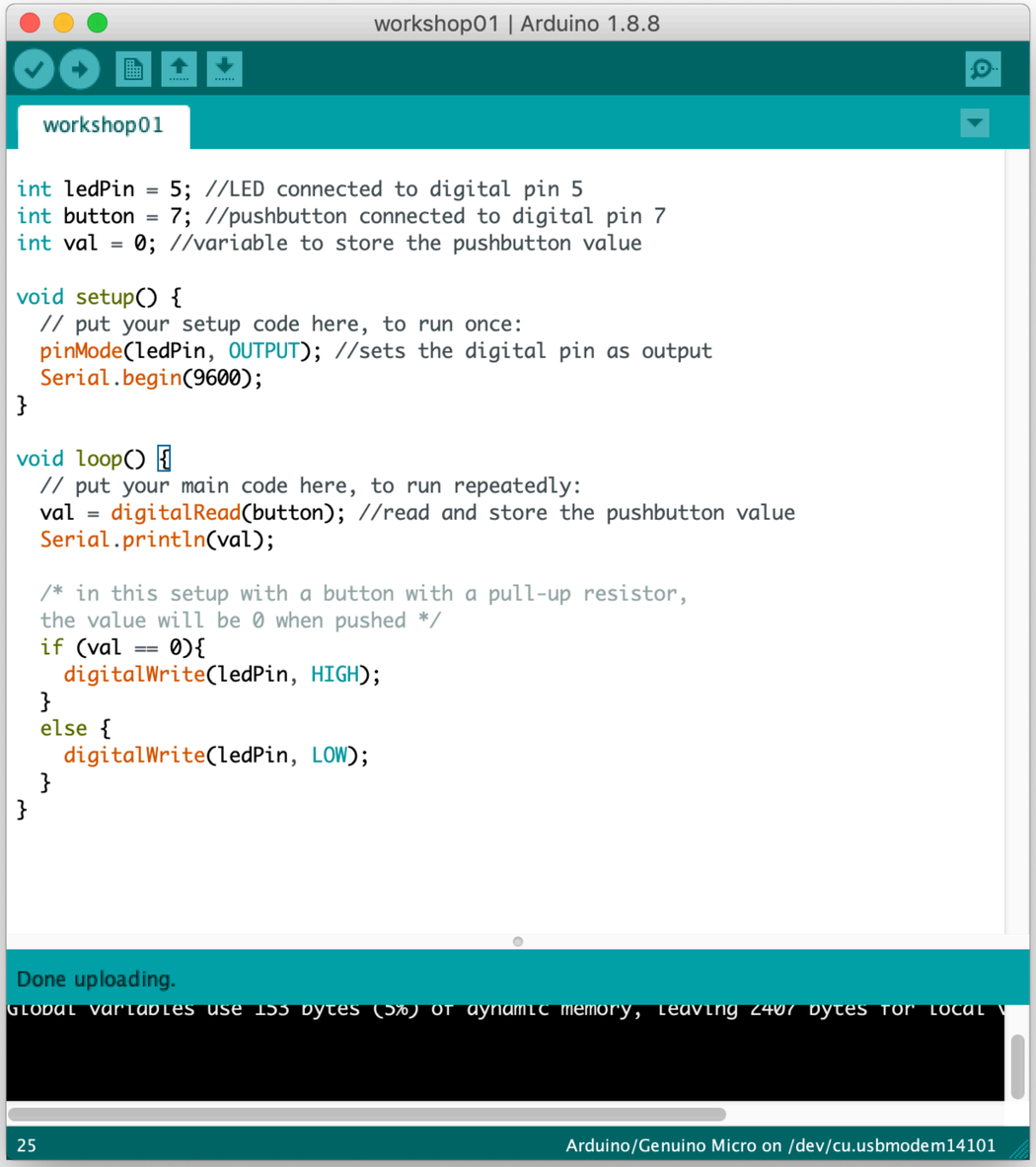
To combine conditions we use the AND and OR operators represented by && and II symbols.

```
if(a>b && a>c) // if a is greater than b and a is greater than c
if(a>b II a>c)   // if a is greater than b or a is greater than c
if(!a)           // if not a (a having been declared)
```

LOGICAL AND RELATIONAL STATEMENTS



LOGICAL AND RELATIONAL STATEMENTS



FUNCTIONS

norm()

Divides value by the maximum value within a set i.e. returns a value between 0.0 and 1.0.

Syntax
norm(*value, low, high*)

Parameters
value: value to normalize
low: lower end of the range
high: higher end of the range

Returns

normalized value

Example

```
float x = norm(102.0, 0.0, 255.0); // x = 0.4
```

constrain()

Constrains a value within a range.

Syntax
norm(*value, min, max*)

Parameters
value: value to normalize
min: lower end of the range
max: higher end of the range

Returns

constrained value

Example

```
float x = norm(102.0, 0.0, 100.0); // x = 100.0
```

map()

Remaps a value from one range to another range. Note: map() will not constrain values.

Syntax
map(*value, fromMin, fromMax, toMin, toMax*)

Parameters
value: value to normalize
fromMin: lower end of the value's range
fromMax: higher end of the value's range
toMin: lower end of the range to map to
toMax: higher end of the range to map to

Returns

re-mapped value

Example

```
float x = map(0.4, 0.0, 1.0, 0.0, 255.0); // x = 102.0
```

FUNCTIONS


workshop01 | Arduino 1.8.8
workshop01
int ledPin = 5; //LED connected to digital pin 5
int button = 7; //pushbutton connected to digital pin 7
int val = 0; //variable to store the pushbutton value
int count = 0; //variable to store the number of times the button has been pushed
int maxcount = 5; //maximum nr of pushes before the led turns off again
int lightlevel = 0; //variable to store the LED light intensity
void setup() {
// put your setup code here, to run once:
pinMode(ledPin, OUTPUT); //sets the digital pin as output
Serial.begin(9600);
}
void loop() {
// put your main code here, to run repeatedly:
val = digitalRead(button); //read and store the pushbutton value
if (val == 0){ //in this setup with a button with pull-up resistor, the value will be 0 when pushed
delay(500); //add delay to allow for debouncing
if (count < maxcount) {
count++;
}
else {
count = 0;
}
}
//analogWrite values range from 0 to 255
lightlevel = map(count, 0, maxcount, 0, 255);
analogWrite(ledPin, lightlevel);
Serial.println(count); //prints the count
}
Done Saving.
Global variables use 153 bytes (5%) of dynamic memory, leaving 2407 bytes for local variables. Maximum i
11
Arduino/Genuino Micro on /dev/cu.usbmodem14101