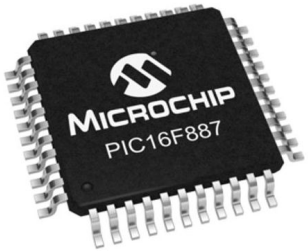
Introduction to Robotics using Arduino

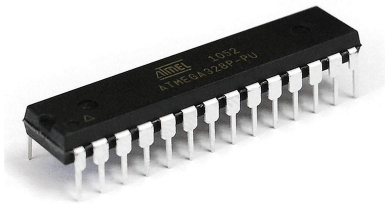
Ioannis Galatos

ioannis.galatos@rca.ac.uk

robotics@rca.ac.uk

Microcontrollers





A compact tiny computer in a single chip.

Specially build for embedded systems.

A microcontroller basically contains:

- Central Processing Unit (CPU)
- Memory
- Input/Output ports
- Timers and Counter
- Interrupt Controls
- ADCs and DACs

Development boards

A development board is a printed circuit board designed to help the experimentation and prototyping with a certain microcontroller.

Arduino Uno

Arduino Leonardo

Arduino Due

Arduino Yún

Arduino Tre

Arduino Micro

Arduino Robot

Arduino Esplora

Arduino Mega ADK

Arduino Ethernet

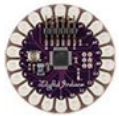
Arduino Mega 2560

Arduino Mini

LilyPad Arduino USB

LilyPad Arduino Simple

LilyPad Arduino SimpleSnap



LilyPad Arduino

Arduino Nano

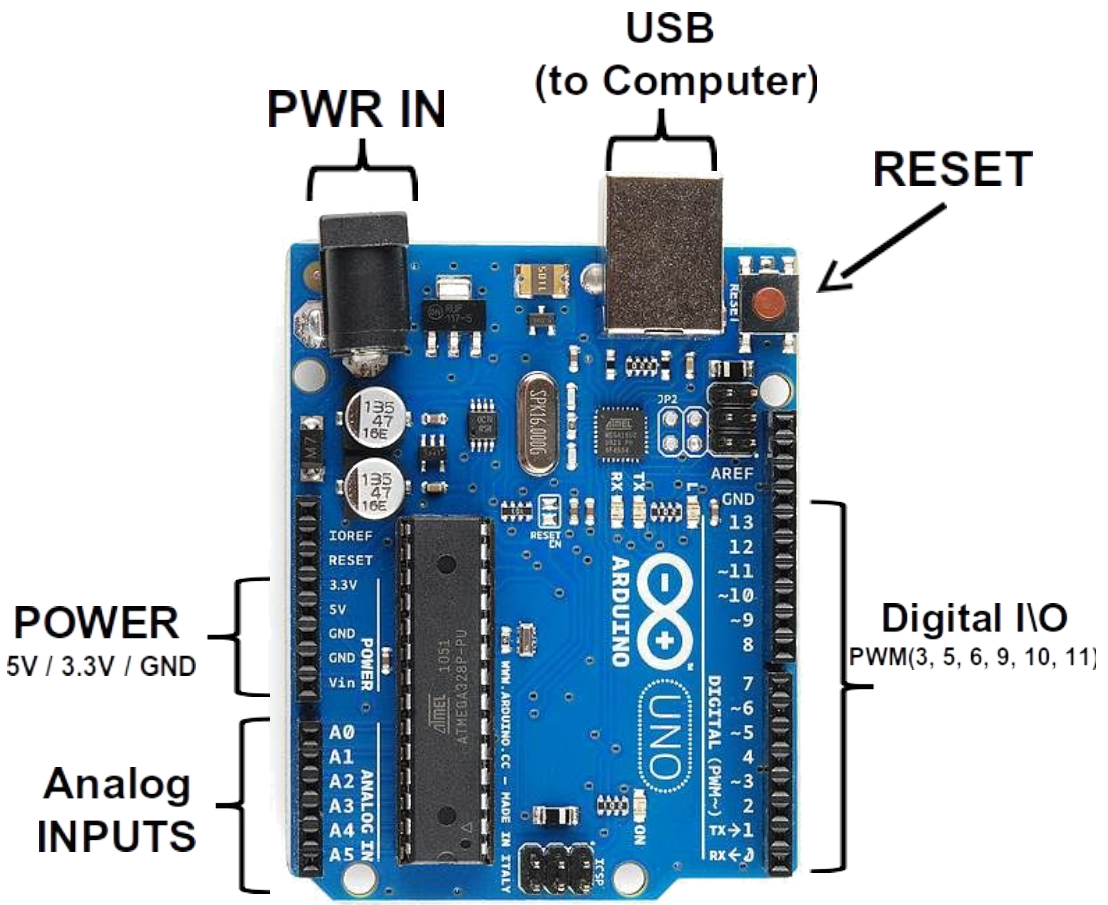
Arduino Pro Mini

Arduino

Open source prototyping platform based on an easy to use software and hardware.

Contains on-board power supply, USB port to communicate with a PC and an Atmel microcontroller.

Open source hardware, anyone can modify it or make one himself.



Analog vs Digital Signals

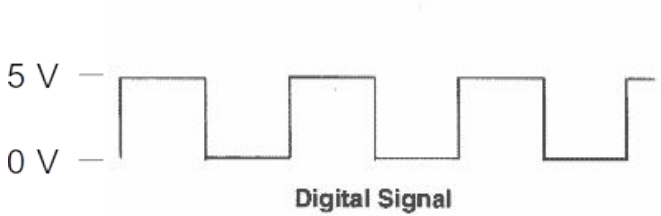
Arduino can input and output analog and digital signals.

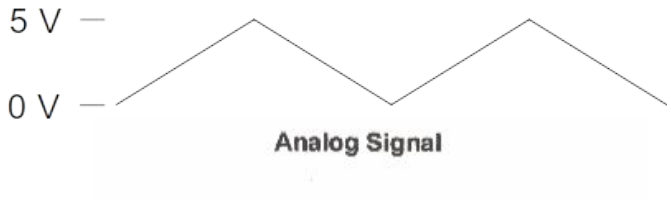
An analog signal can take on any number of values.

A digital signal has only two values: HIGH and LOW.

The ADC turns the analog voltage into a digital value(0 to 1023).

The default reference voltage is 5V in arduino UNO.



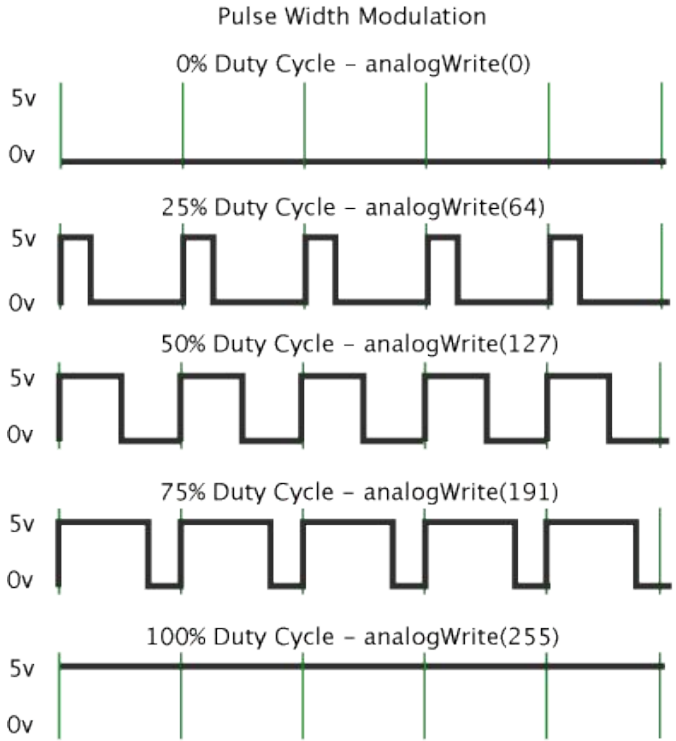


PWM

PWM stands for Pulse Width Modulation.

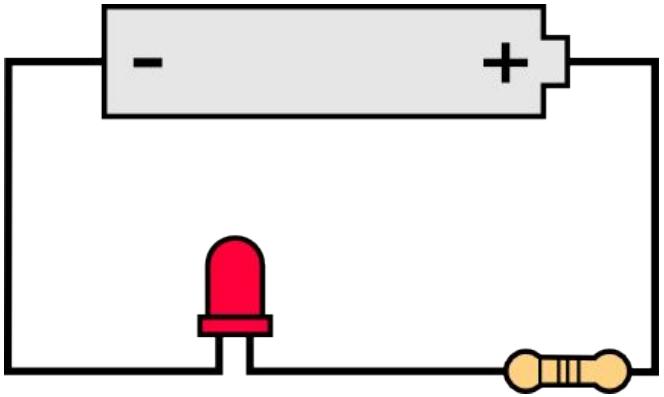
It is a technique for getting analog results with digital means.

A signal switched between on and off (square wave) can simulate voltages in between ON (5V) and OFF (0V).

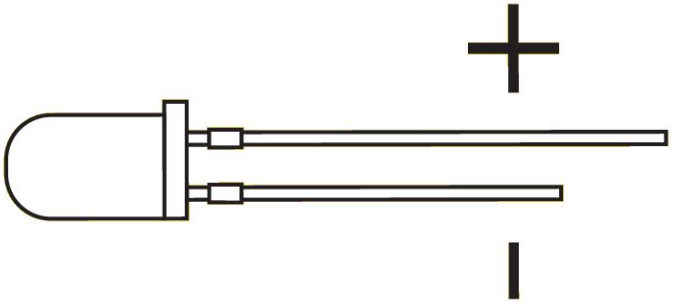


Project #1 - Blink an LED

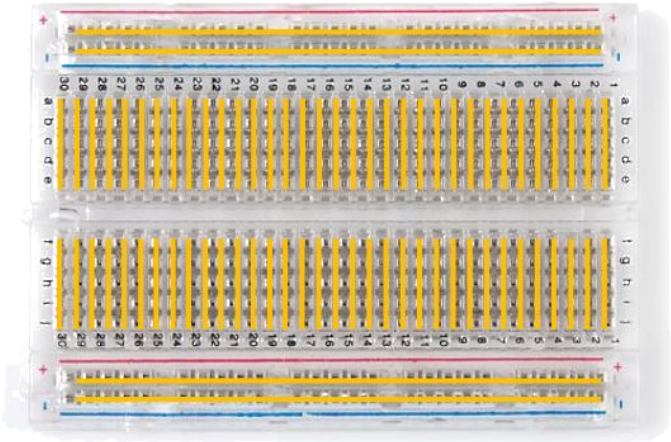
Circut



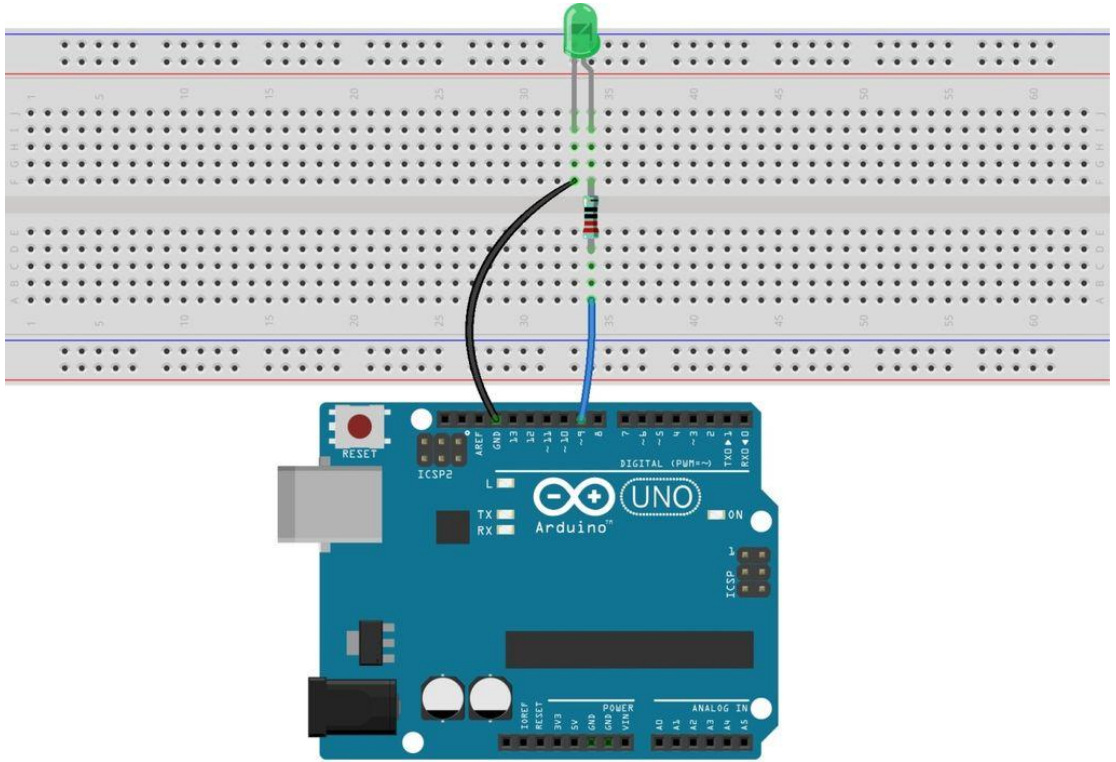
Led Polarity

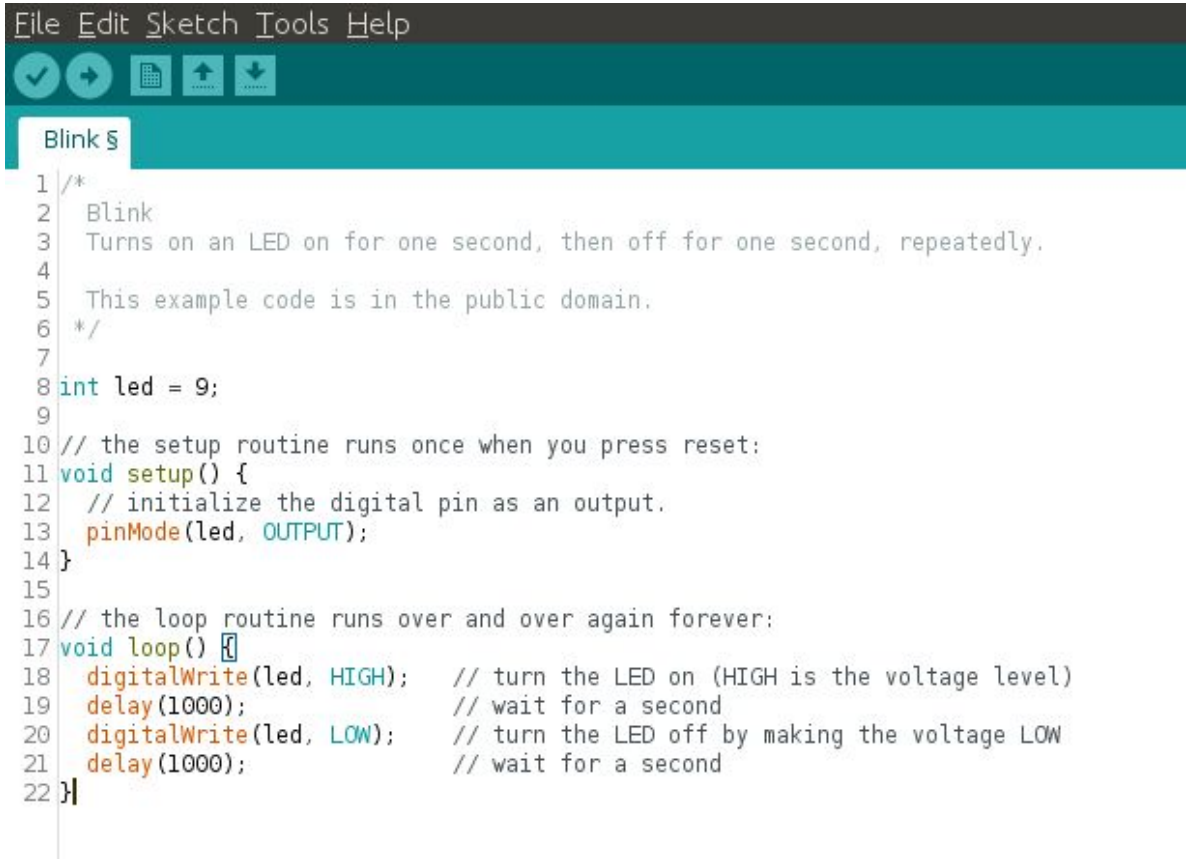


Breadboard is a way of prototyping and testing electronic circuits without having to use a soldering iron. Components are pushed into the sockets on the breadboard and we use wires to make the connections.



Breadboard



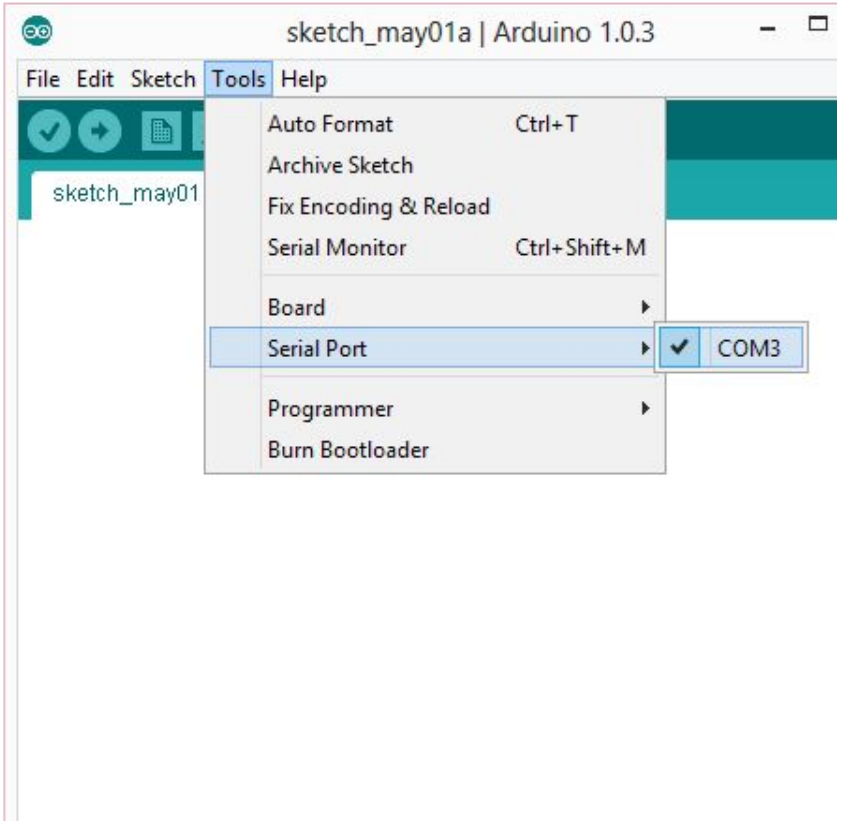


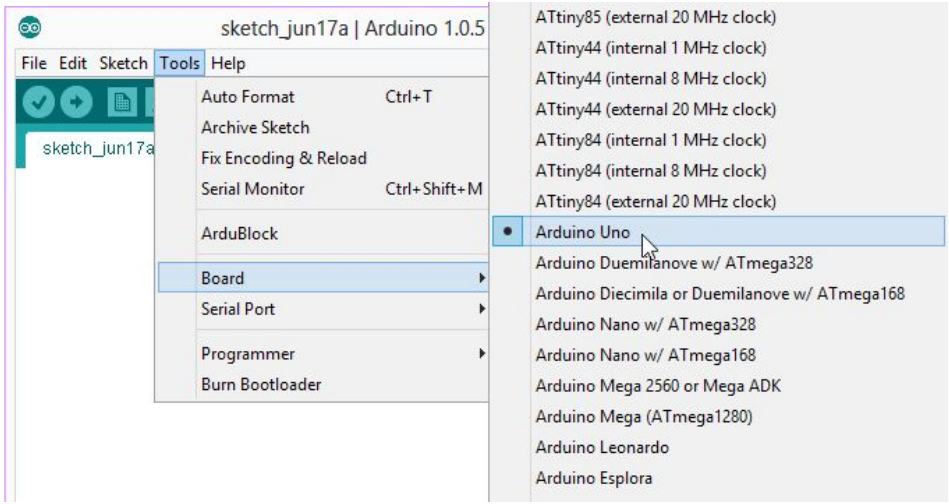
Programs written using Arduino Software(IDE) are called sketches.

Find the Sketch:
“File → Examples → 01.Basics → Blink”

Change:
int led = 13;
with
int led = 9;

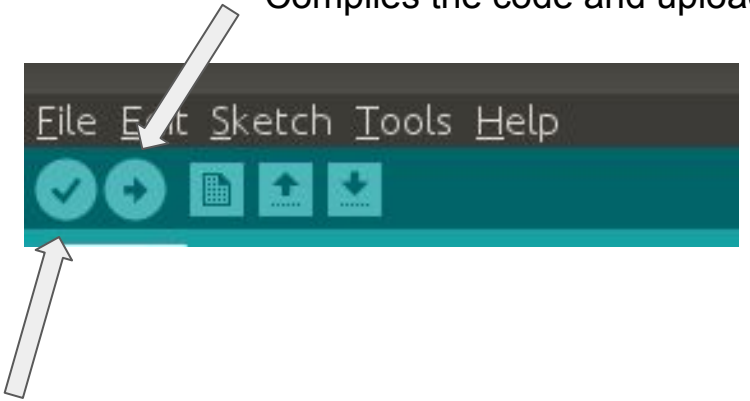
Select Serial Port and Board





Verify and Upload Code

Upload:
Compiles the code and uploads it to the configured board.

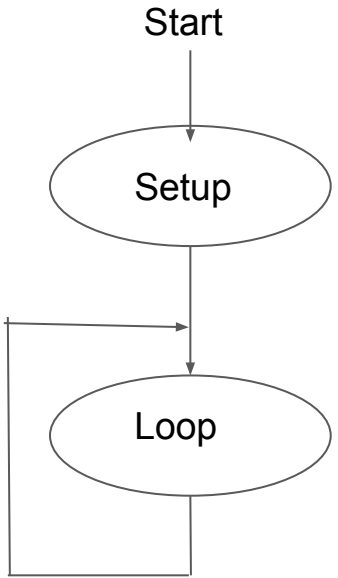


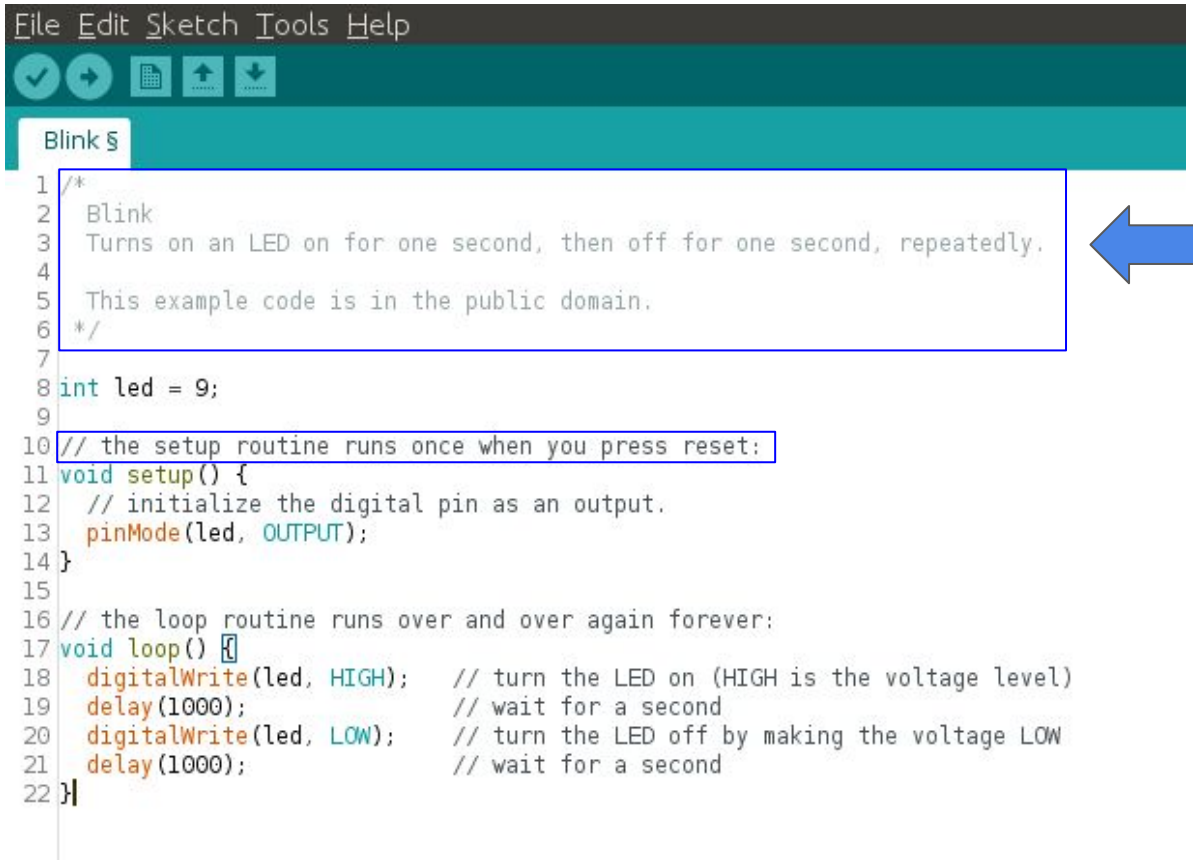
Verify:
Checks your code for errors compiling it.

Arduino required functions

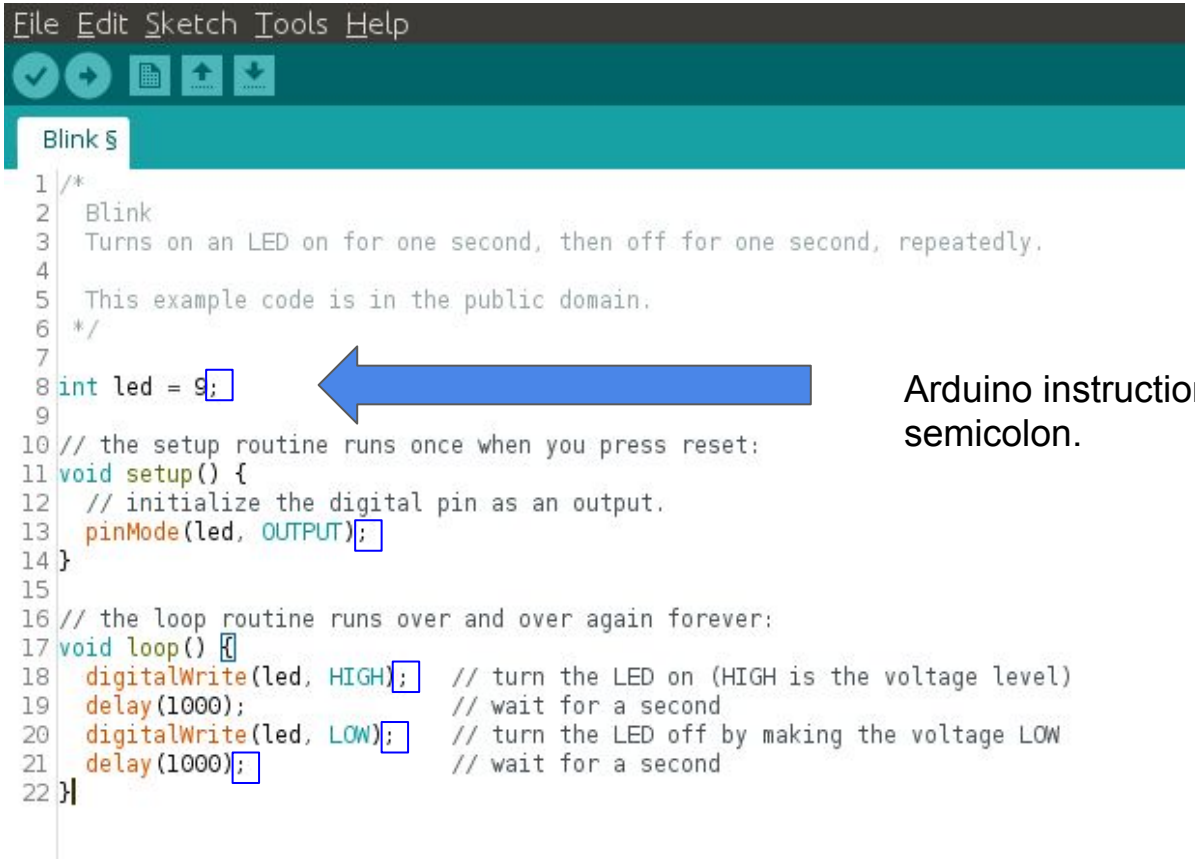
Functions allow structuring the program in segments of code that perform individual tasks.

```
void setup()
{
        //Runs at the beginning, only once after each powerup or reset
        // Use it to initialize variables and pin modes.
}


void loop()
{
        //Contains the instructions that get repeated
        // over and over until the board is turned off

}
```



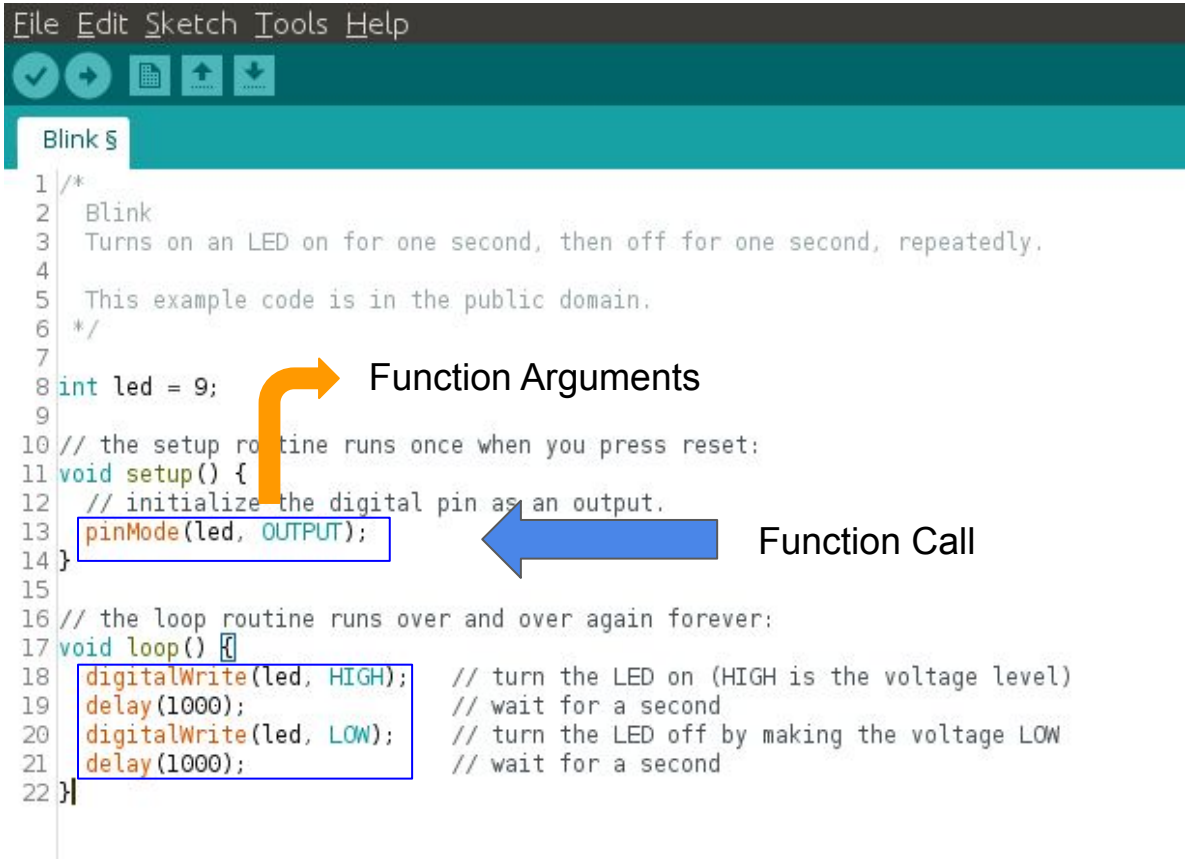

File Edit Sketch Tools Help
Blink §
/*
Blink
Turns on an LED on for one second, then off for one second, repeatedly.
This example code is in the public domain.
*/
int led = 9;
// the setup routine runs once when you press reset:
void setup() {
// initialize the digital pin as an output.
pinMode(led, 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
}

Comments



Arduino instructions must end with a semicolon.

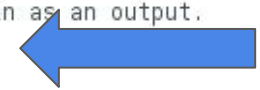
Variable: is a place to store a piece of data.


File Edit Sketch Tools Help
Blink §
/*
Blink
Turns on an LED on for one second, then off for one second, repeatedly.
This example code is in the public domain.
*/
int led = 9;
// 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
}
Function Arguments
Function Call

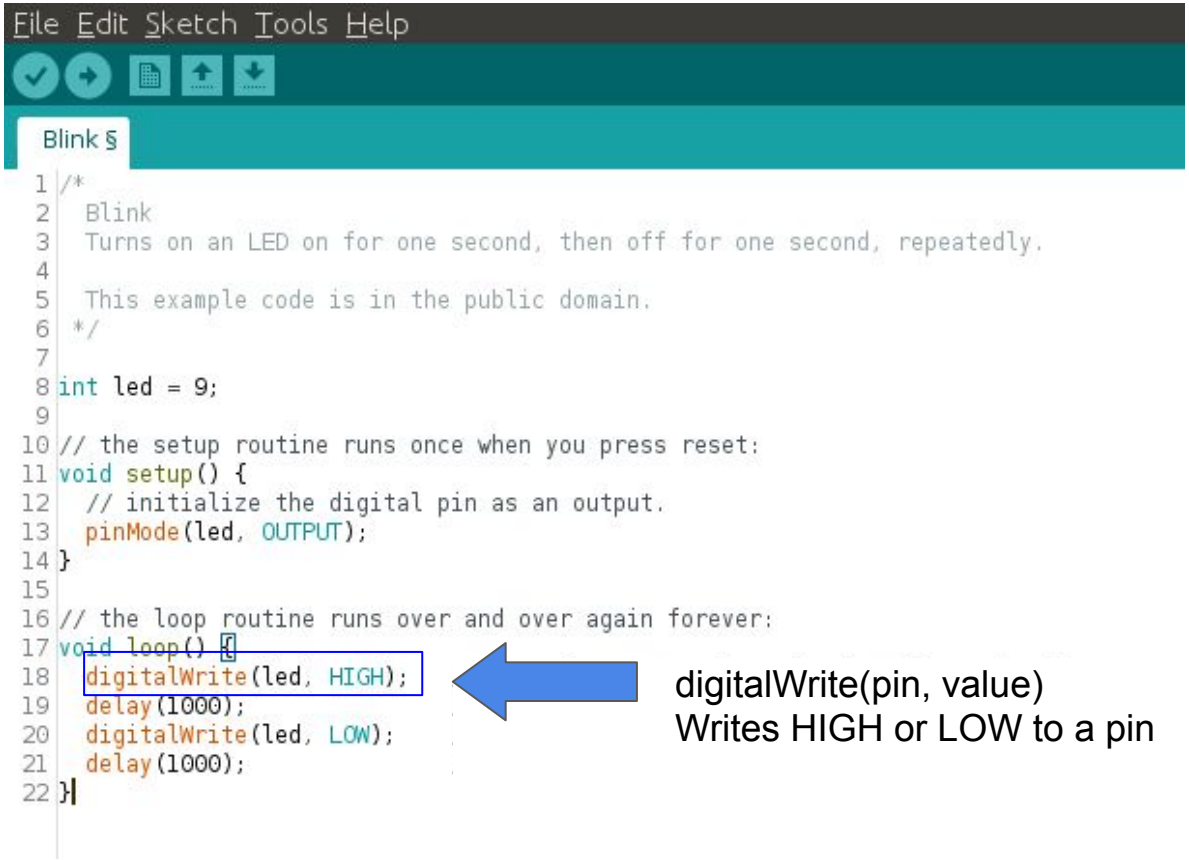
File Edit Sketch Tools Help

Blink §

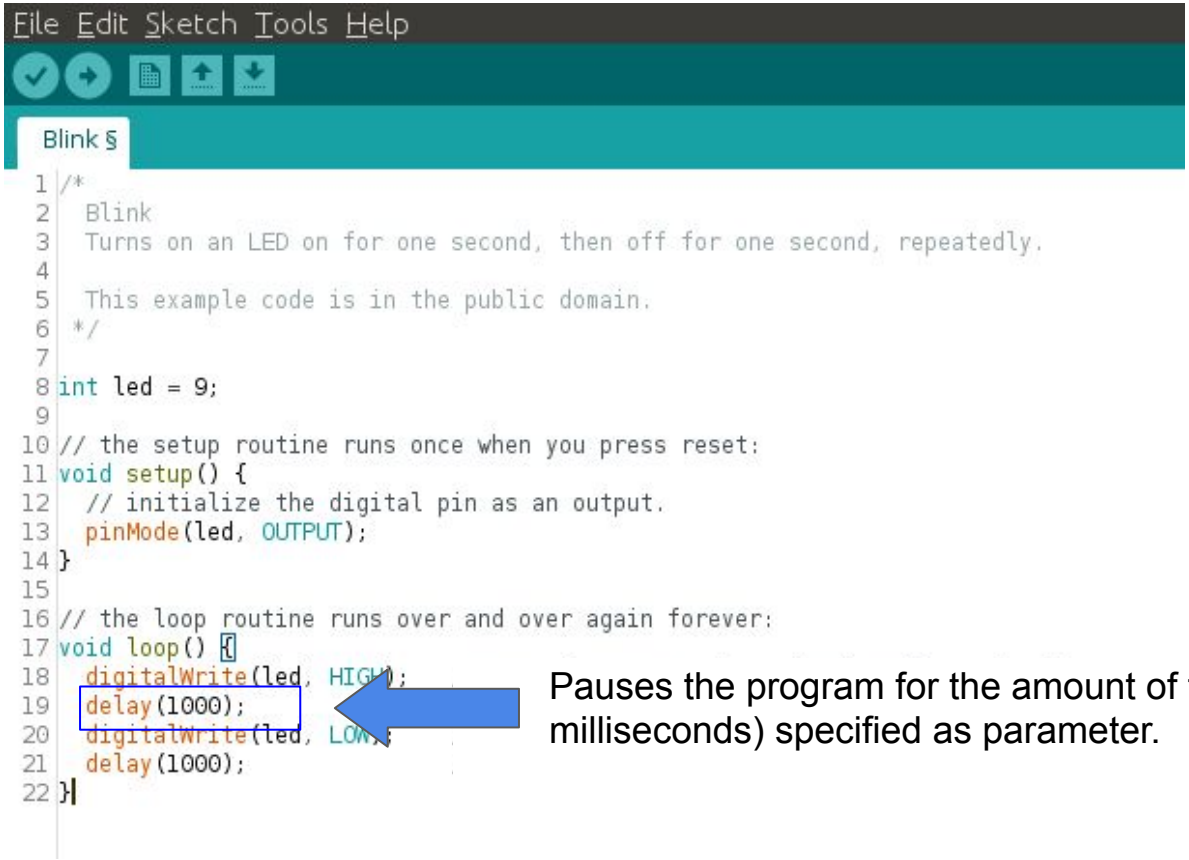
```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

int led = 9;

// 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
}
```



pinMode(pin, mode)
Sets pin to either INPUT or OUTPUT

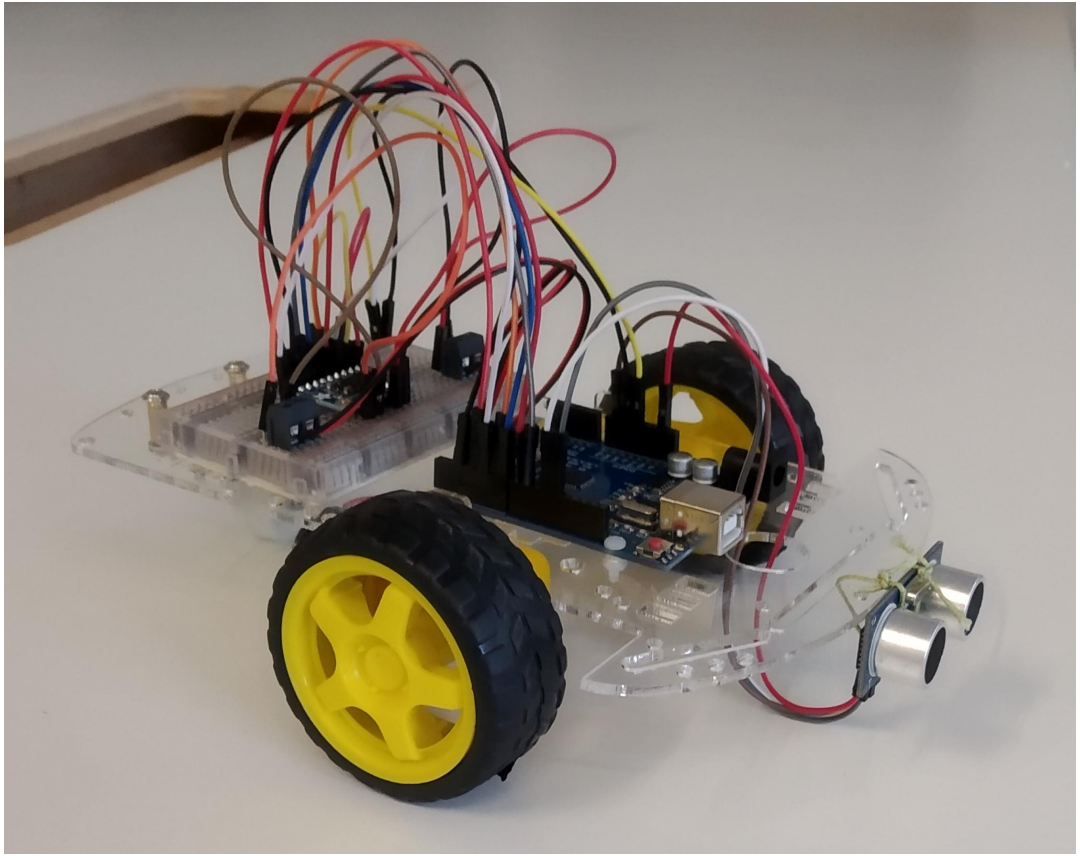


digitalWrite(pin, value)
Writes HIGH or LOW to a pin



Pauses the program for the amount of time (in milliseconds) specified as parameter.

Project #2 - Wheeled robot

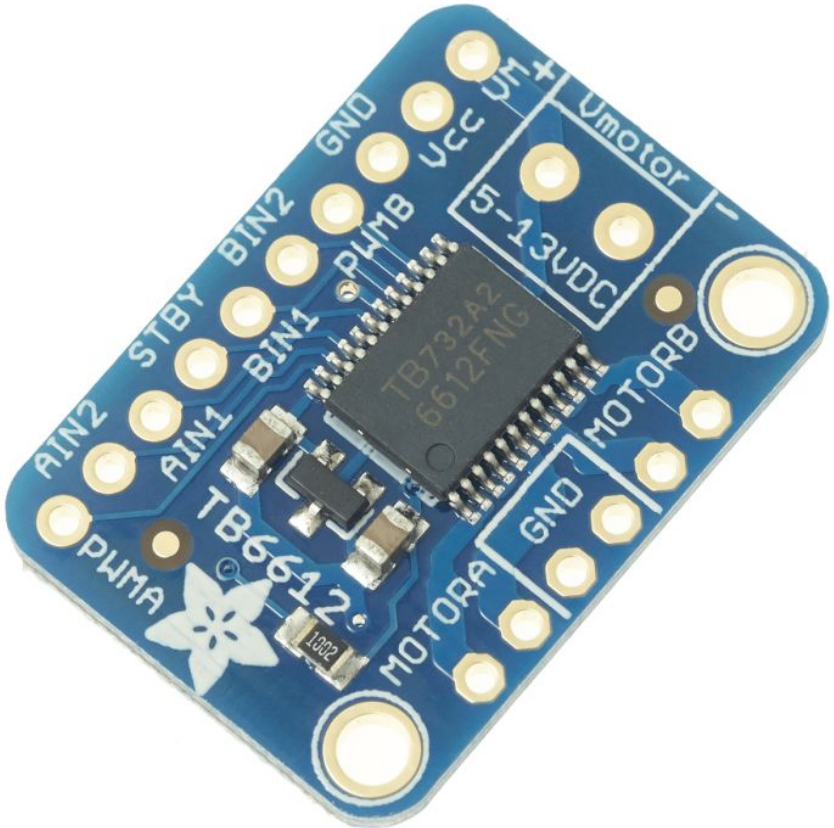


Servo motors

Stepper Motors

DC motors

TB6612FNG

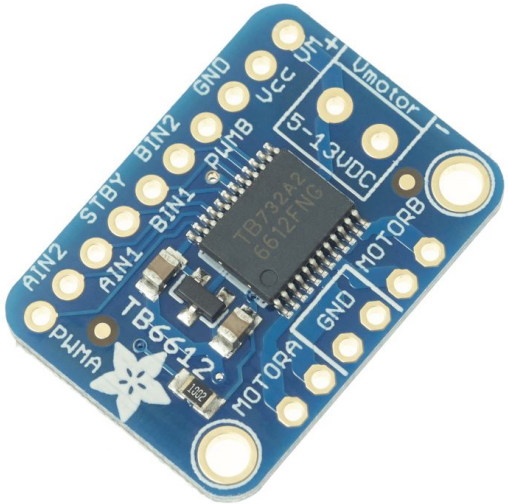


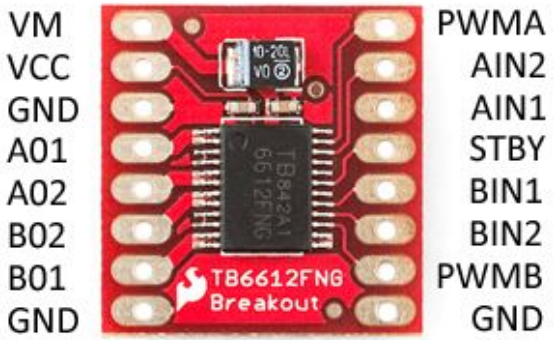
DC dual motor driver.

Allow to switch direction of current.

The motor can spin in both directions and at different speed.

Pin Label	Function	Power/Input /Output	Notes
VM	Motor Voltage	Power	This is where you provide power for the motors (2.2V to 13.5V)
VCC	Logic Voltage	Power	This is the voltage to power the chip and talk to the microcontroller (2.7V to 5.5V)
GND	Ground	Power	Common Ground for both motor voltage and logic voltage (all GND pins are connected)
STBY	Standby	Input	Allows the H-bridges to work when high (has a pulldown resistor so it must actively pulled high)
AIN1/BIN1	Input 1 for channels A/B	Input	One of the two inputs that determines the direction.
AIN2/BIN2	Input 2 for channels A/B	Input	One of the two inputs that determines the direction.
PWMA/PWMB	PWM input for channels A/B	Input	PWM input that controls the speed
A01/B01	Output 1 for channels A/B	Output	One of the two outputs to connect the motor
A02/B02	Output 2 for channels A/B	Output	One of the two outputs to connect the motor





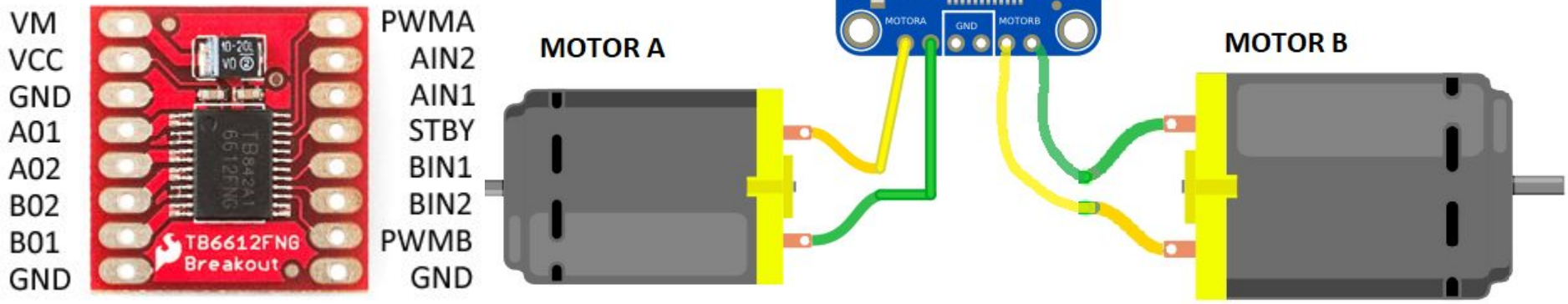
H-SW Control Function

Input				Output		
IN1	IN2	PWM	STBY	OUT1	OUT2	Mode
H	H	H/L	H	L	L	Short brake
L	H	H	H	L	H	CCW
		L	H	L	L	Short brake
H	L	H	H	H	L	CW
		L	H	L	L	Short brake
L	L	H	H	OFF (High impedance)		Stop
H/L	H/L	H/L	L	OFF (High impedance)		Standby

Arduino ↔ TB6612

5V supply → VCC
GND → GND

AIN1 -> pin 2
AIN2 -> pin 4
BIN1 -> pin 7
BIN2 -> pin 8
PWMA -> pin 5
PWMB -> pin 6
STBY -> pin 9



```
1
2  //Define the pins for all the inputs/outputs
3  //PWM defines must be on PWM pins
4  #define PWMA 5
5  #define AIN1 2
6  #define AIN2 4
7  #define PWMB 6
8  #define BIN1 7
9  #define BIN2 8
10 #define STBY 9
11
12
13
14 void setup()
15 {
16   //Set all the pin as outputs.
17   pinMode(PWMA,OUTPUT);
18   pinMode(AIN1,OUTPUT);
19   pinMode(AIN2,OUTPUT);
20   pinMode(PWMB,OUTPUT);
21   pinMode(BIN1,OUTPUT);
22   pinMode(BIN2,OUTPUT);
23   pinMode(STBY,OUTPUT);
24 }
25
```

#Define is a “text-replace” macro.

The pre-processor will replace PWMA for example with “5”
Before the compiler compiles the code.

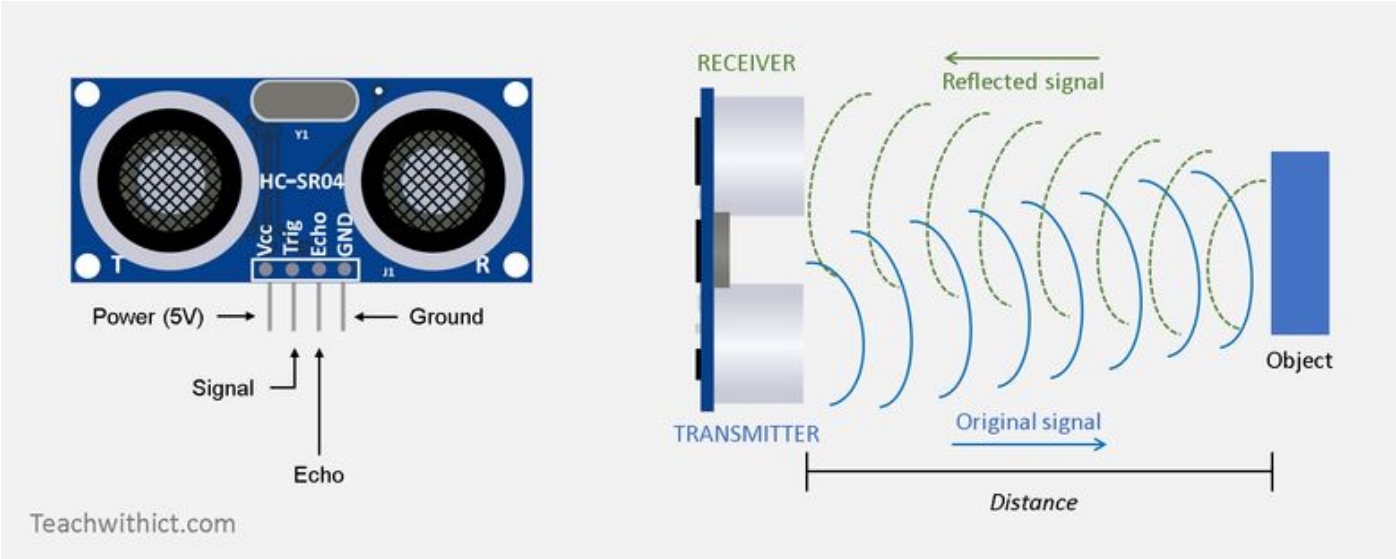
```
27 void loop()
28 {
29
30
31   startUp();
32   goForward();
33   delay(2000);
34   brake();
35   delay(1000);
36   rotateRight();
37   delay(1000);
38   goForward();
39   delay(1000);
40   rotateLeft();
41   delay(1000);
42   goBackward();
43   delay(1000);
44 }
```

```
104 void startUp()
105 {
106   digitalWrite(STBY,HIGH);
107 }
108
```

```
47 void goForward()
48 {
49   digitalWrite (AIN1,HIGH);
50   digitalWrite (AIN2,LOW);
51   analogWrite  (PWMA,192);
52   digitalWrite (BIN1,HIGH);
53   digitalWrite (BIN2,LOW);
54   analogWrite  (PWMB,192);
55 }
56
57
58 void goBackward()
59 {
60   digitalWrite (AIN1,LOW);
61   digitalWrite (AIN2,HIGH);
62   analogWrite  (PWMA,192);
63   digitalWrite (BIN1,LOW);
64   digitalWrite (BIN2,HIGH);
65   analogWrite  (PWMB,192);
66 }
```

```
69 void rotateRight()
70 {
71   digitalWrite (AIN1,HIGH);
72   digitalWrite (AIN2,LOW);
73   analogWrite  (PWMA,192);
74   digitalWrite (BIN1,LOW);
75   digitalWrite (BIN2,HIGH);
76   analogWrite  (PWMB,192);
77 }
78
79
80 void rotateLeft()
81 {
82   digitalWrite (AIN1,LOW);
83   digitalWrite (AIN2,HIGH);
84   analogWrite  (PWMA,192);
85   digitalWrite (BIN1,HIGH);
86   digitalWrite (BIN2,LOW);
87   analogWrite  (PWMB,192);
88 }
89
```

Ultrasonic Sensor - HC SR04



- The Transmitter (trig pin) sends a high frequency sound signal.
- When the sound signal finds an obstacle, it is reflected.
- The transmitter (echo pin) receives the signal.
- The time between the transmission and reception of the signal allows us to calculate the distance of the obstacle.

Ultrasonic Sensor - Calculation of distance

```
int check_distance()
{
  //clear the trig pin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  //create a pulse of 10 microseconds to triger the trig pin
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  //read the echo pin, HIGH pulse  = duration of time between sending
  //the pulse and receiving the echo from the obstacle
  duration = pulseIn(echoPin, HIGH);

  // Convert the time into a distance
  cm = duration * 0.034 / 2;     // speed of sound = 340m/s = 0.034c/us

  //return the value
  return cm;

}
```

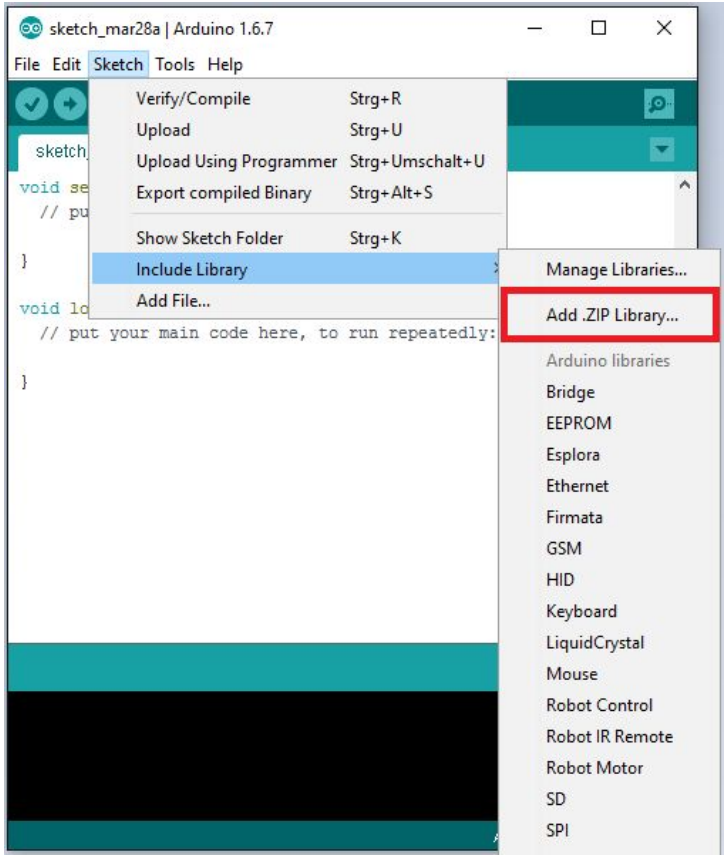
pulseIn(), reads a pulse (either HIGH or LOW), and returns the length of the pulse in microseconds.

Ultrasonic Sensor - Using NewPing Library

Download the library from here: https://is.gd/H9pLjB

```
// ---------------------------------------------------------------------------
// Example NewPing library sketch that does a ping about 20 times per second.
// ---------------------------------------------------------------------------

#include <NewPing.h>

#define TRIGGER_PIN  12
#define ECHO_PIN     11
// Maximum distance we want to ping for (in centimeters). Maximum sensor distance is rated at 400-500cm.
#define MAX_DISTANCE 200

int distance;

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); // NewPing setup of pins and maximum distance.

void setup() {
  Serial.begin(9600);
}

void loop() {
  delay(50); // Wait 50ms between pings (about 20 pings/sec). 29ms should be the shortest delay between pings.
  distance = sonar.ping_cm();
  Serial.print("Ping: ");
  Serial.print(distance);
  Serial.println("cm");
}
```

How to install the library



- Open the Arduino IDE
- Go to “Sketch/Include Library/Add .ZIP Library”
- Find the .zip file you just download it
- Select the library .zip file and open this

Arduino Conditional Statements

Conditional statements allow a program to execute a piece of code based on a decision.

They usually start with the world IF.

The part that follows the word IF, is then evaluated.

If it is TRUE, then the last part is executed.

```
if(<logical expression>){
  <programming statement>
  <programming statement>
  <programming statement>
  …
}
```

Programming statements to be executed if the logical expression resolves to true

```
int Num1 = 10;
int Num2 = 15;

if (Num1 > Num2)
{
  Serial.println("Num1 is greater than Num2");
}

if (Num2 > Num2)
{
  Serial.println("Num2 is greater than Num1");
}
```

```
int Num1 = 10;
int Num2 = 15;

if (Num1 > Num2)
{
  Serial.println("Num1 is greater than Num2");
}
else
{
  Serial.println("Num2 is greater than Num1");
}

}
```

While Loop

A “while” loop will loop continuously until the expression inside the parenthesis becomes false.

Something must change the tested variable, or the while loop will never exit.

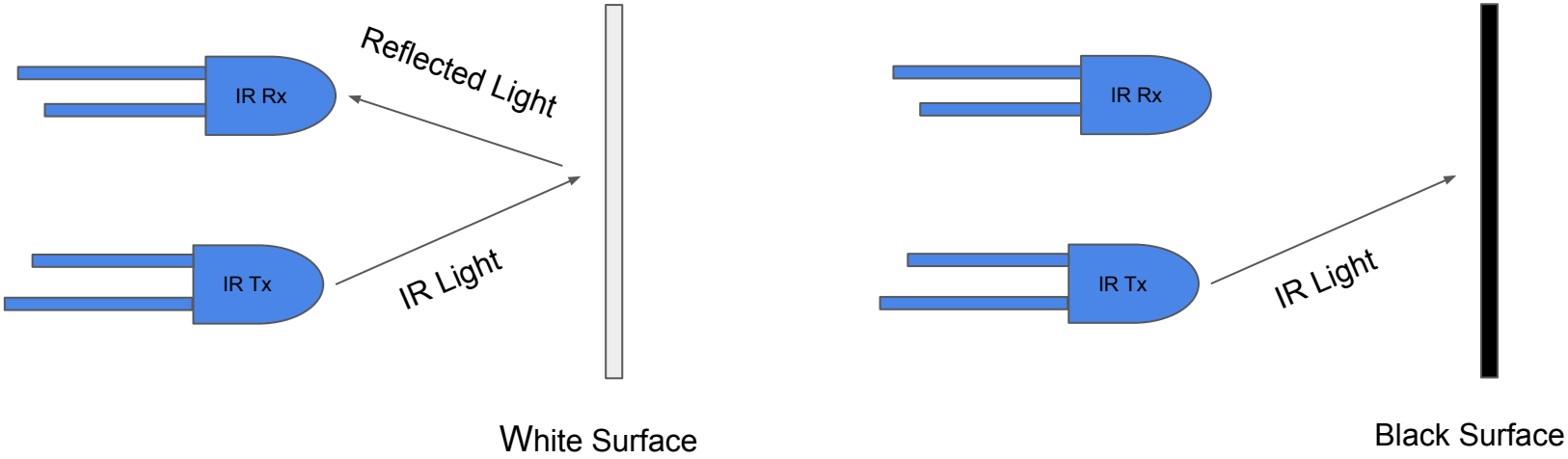
Syntax

```
while (condition) {
  // statement(s)
}
```

Example Code

```
var = 0;
while (var < 200) {
  // do something repetitive 200 times
  var++;
}
```

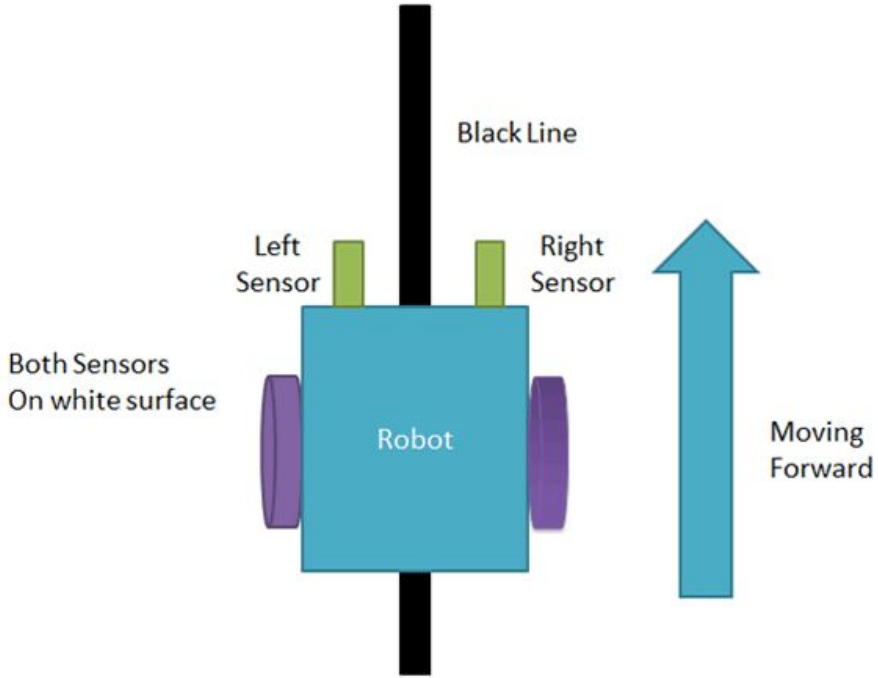
IR Proximity Sensor



The sensor transmits infrared light. If there is an object in front of the IR sensor, the transmitted infrared light reflects from the object and received by the IR receiver. The IR sensor gives 0.

If there is no object in front of the IR sensor then the transmitted IR light is not received by the IR receiver. If the surface is black the IR light is absorbed. The sensor gives 0 in this case.

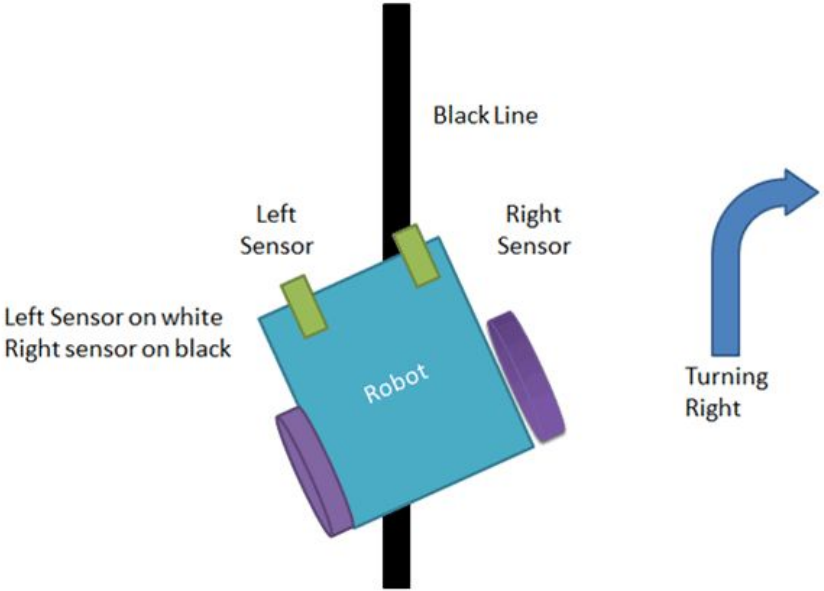
Line Follower Robot



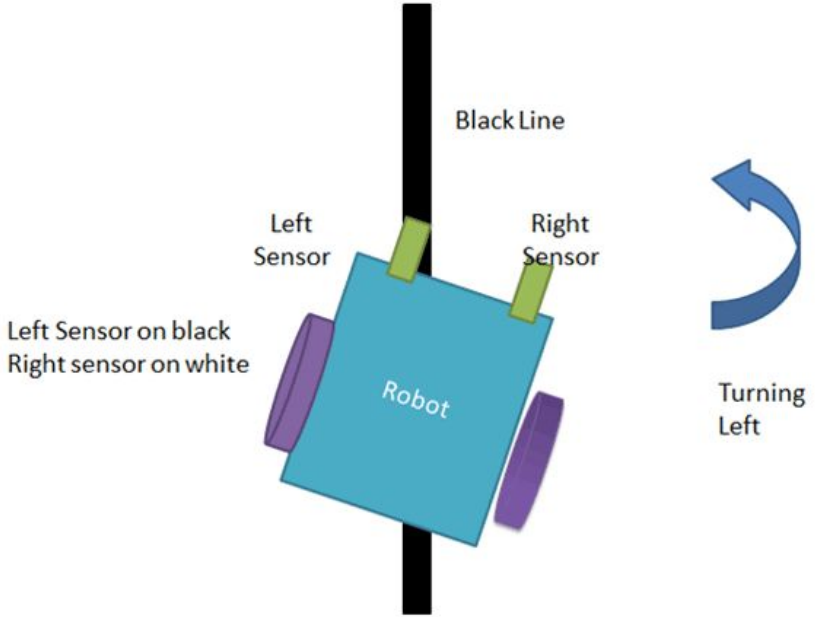
We are using two IR sensors.

When both left and right sensor senses white then the robot move forward.

Line Follower Robot



If the left sensor comes to a black line then the robot turn towards left side.



If the right sensor senses the black line then the robot turn to the right side.