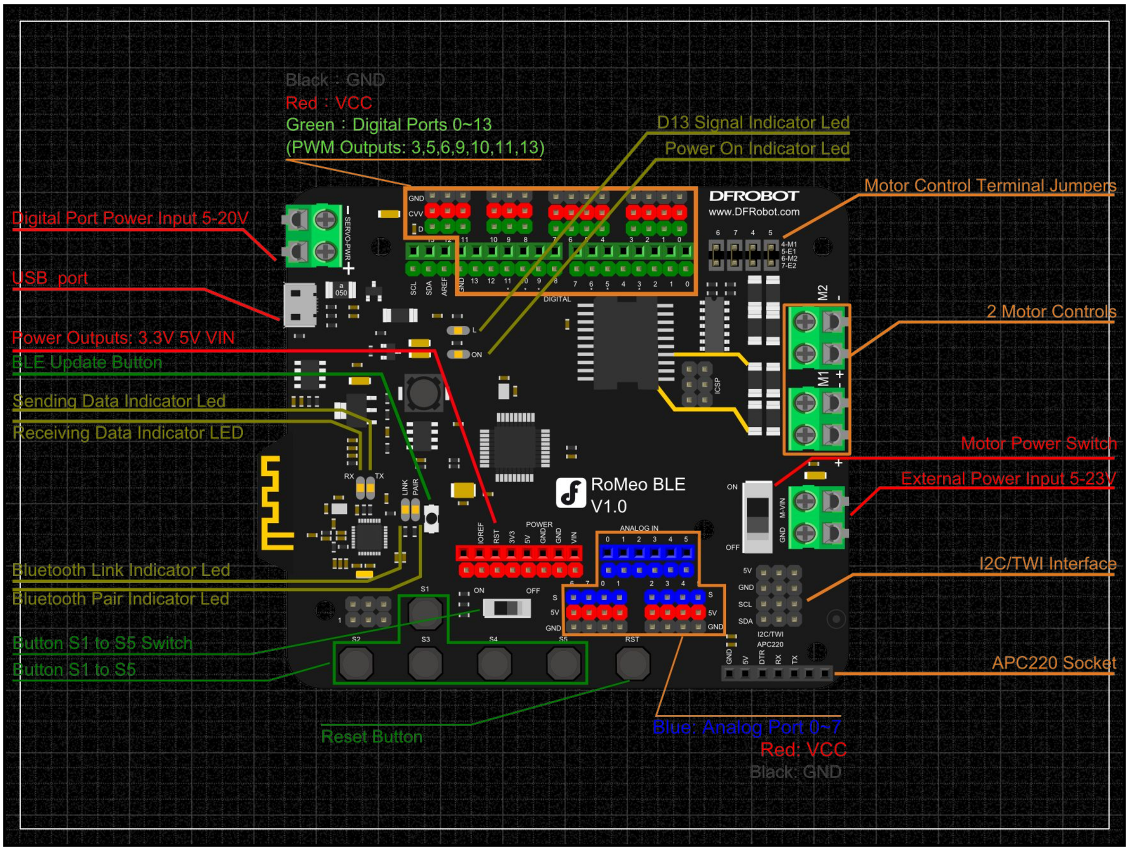
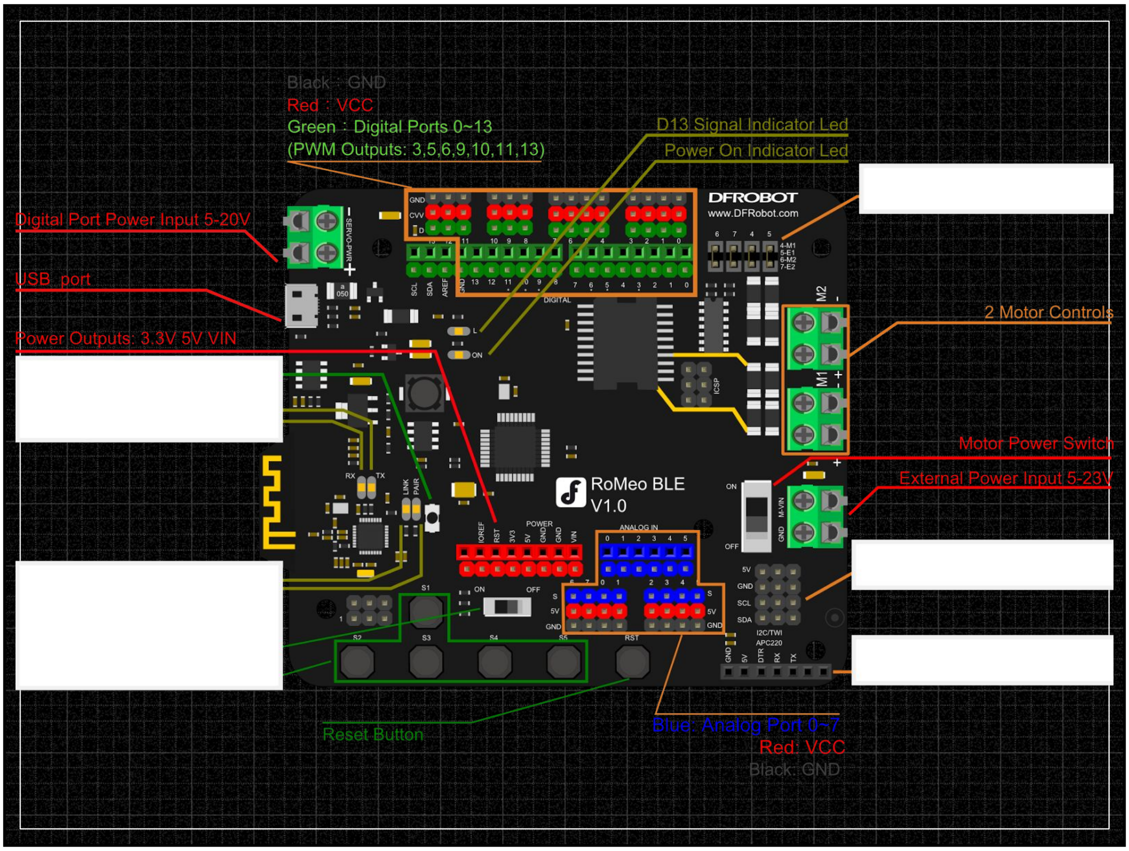
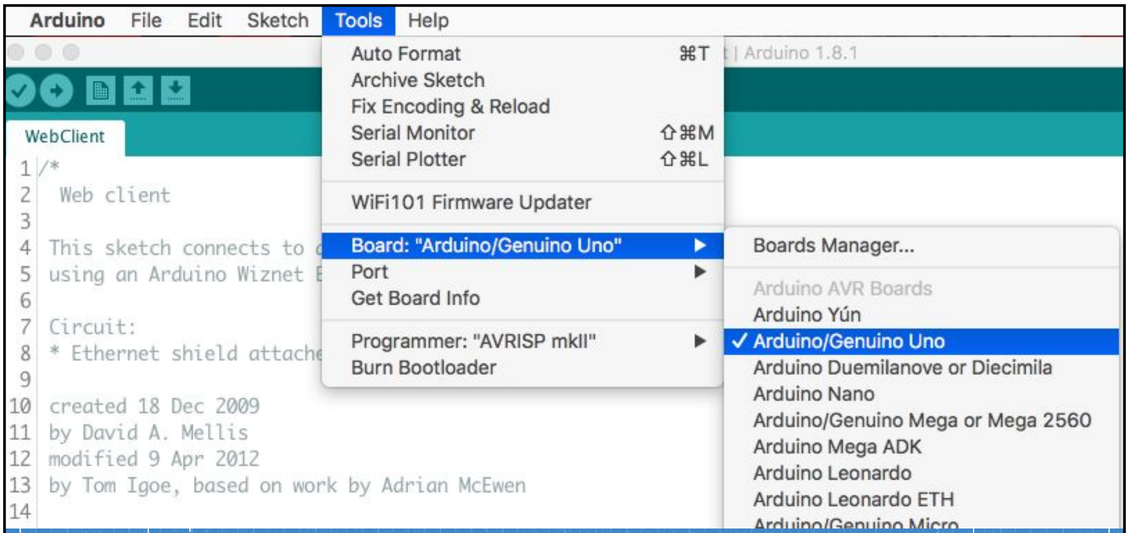
Arduino


Black：GND
Red：VCC
Green：Digital Ports 0~13
(PWM Outputs: 3,5,6,9,10,11,13)
D13 Signal Indicator Led
Power On Indicator Led
Motor Control Terminal Jumpers
Digital Port Power Input 5-20V
USB port
Power Outputs: 3.3V 5V VIN
BLE Update Button
Sending Data Indicator Led
Receiving Data Indicator LED
2 Motor Controls
Motor Power Switch
External Power Input 5-23V
I2C/TWI Interface
Bluetooth Link Indicator Led
Bluetooth Pair Indicator Led
Button S1 to S5 Switch
Button S1 to S5
APC220 Socket
Reset Button
Blue: Analog Port 0~7
Red: VCC
Black: GND
DFROBOT
www.DFRobot.com
RoMeo BLE
V1.0


Black：GND
Red：VCC
Green：Digital Ports 0~13
(PWM Outputs: 3,5,6,9,10,11,13)
D13 Signal Indicator Led
Power On Indicator Led
Digital Port Power Input 5-20V
USB port
Power Outputs: 3.3V 5V VIN
2 Motor Controls
Motor Power Switch
External Power Input 5-23V
DFROBOT
www.DFRobot.com
RoMeo BLE
V1.0
Reset Button
Blue: Analog Port 0~7
Red: VCC
Black: GND

Setup

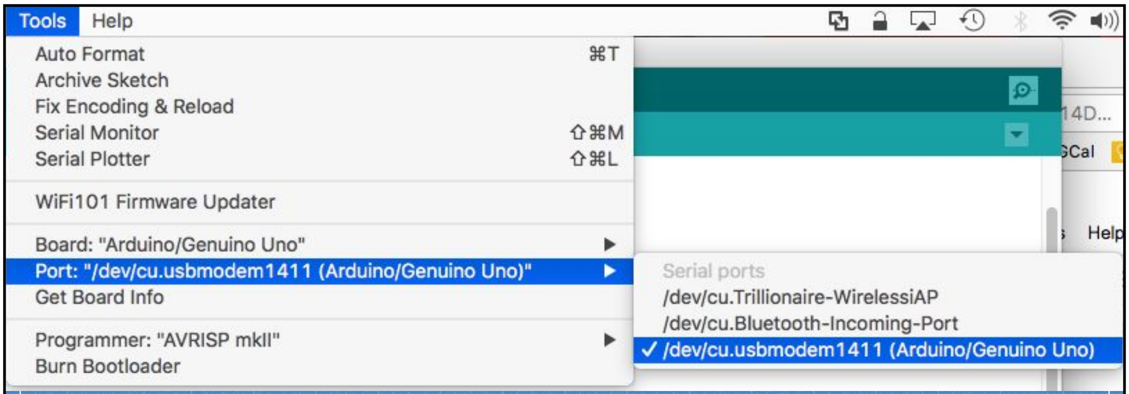


Board: "Arduino/Genuino Uno" ▶

✓ Arduino/Genuino Uno

```
/*
 Web client

 This sketch connects to a
 using an Arduino Wiznet E

 Circuit:
 * Ethernet shield attache

 created 18 Dec 2009
 by David A. Mellis
 modified 9 Apr 2012
 by Tom Igoe, based on work by Adrian McEwen

```

Choose the board



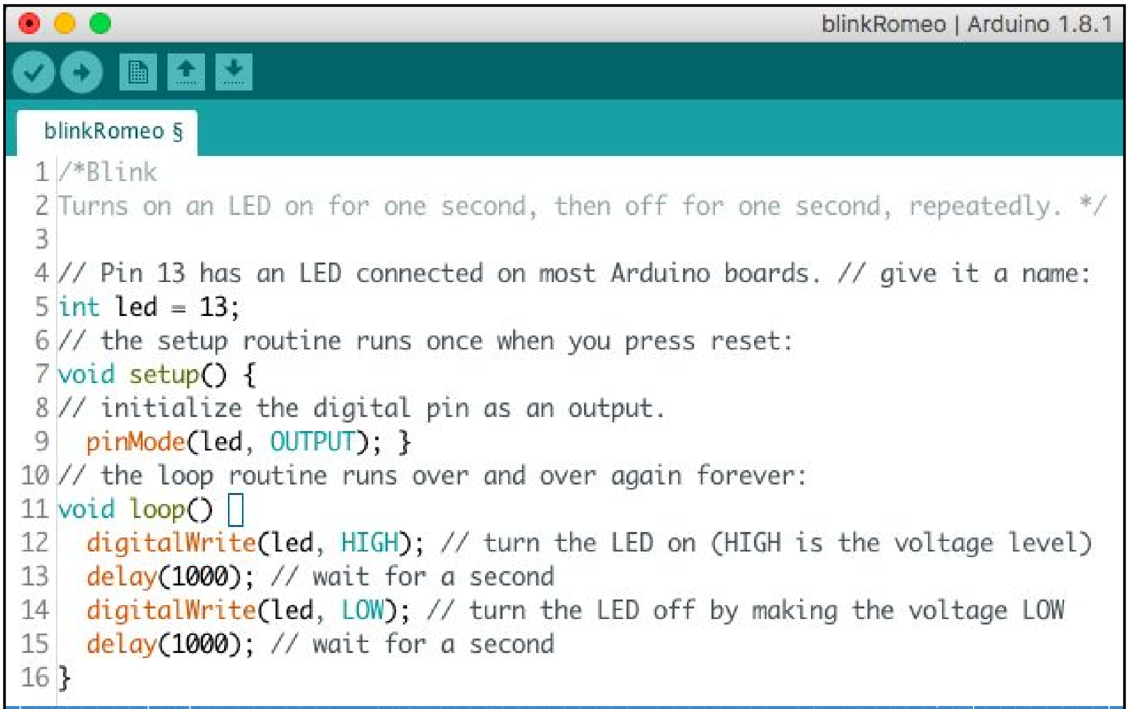
Board: "Arduino/Genuino Uno" ▶

Port: "/dev/cu.usbmodem1411 (Arduino/Genuino Uno)" ▶

✓ /dev/cu.usbmodem1411 (Arduino/Genuino Uno)

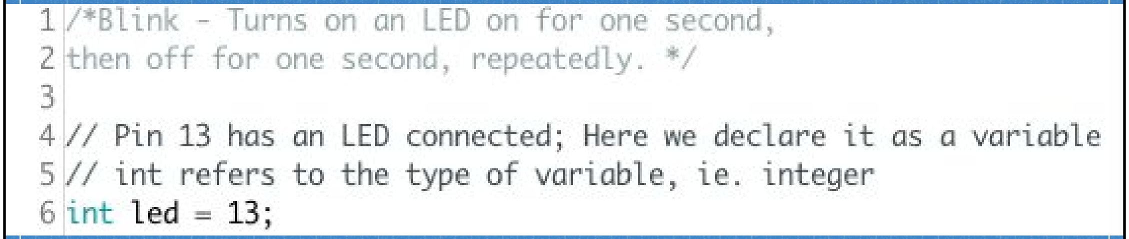
Choose the port

Your First Sketch



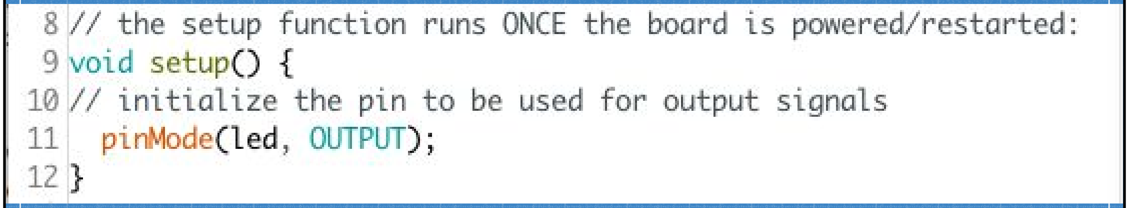
```
/*Blink
Turns on an LED on for one second, then off for one second, repeatedly. */

// 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
}
```

Blink



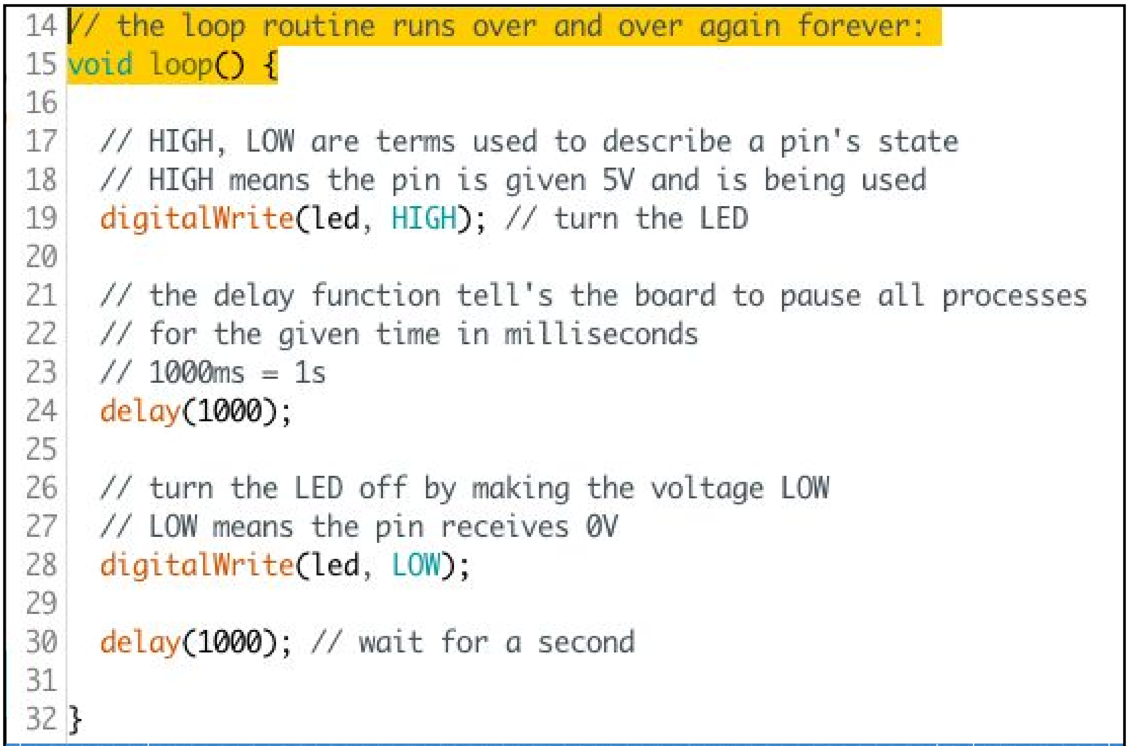
```
/*Blink - Turns on an LED on for one second,
then off for one second, repeatedly. */

// Pin 13 has an LED connected; Here we declare it as a variable
// int refers to the type of variable, ie. integer
int led = 13;
```

Declaring a variable



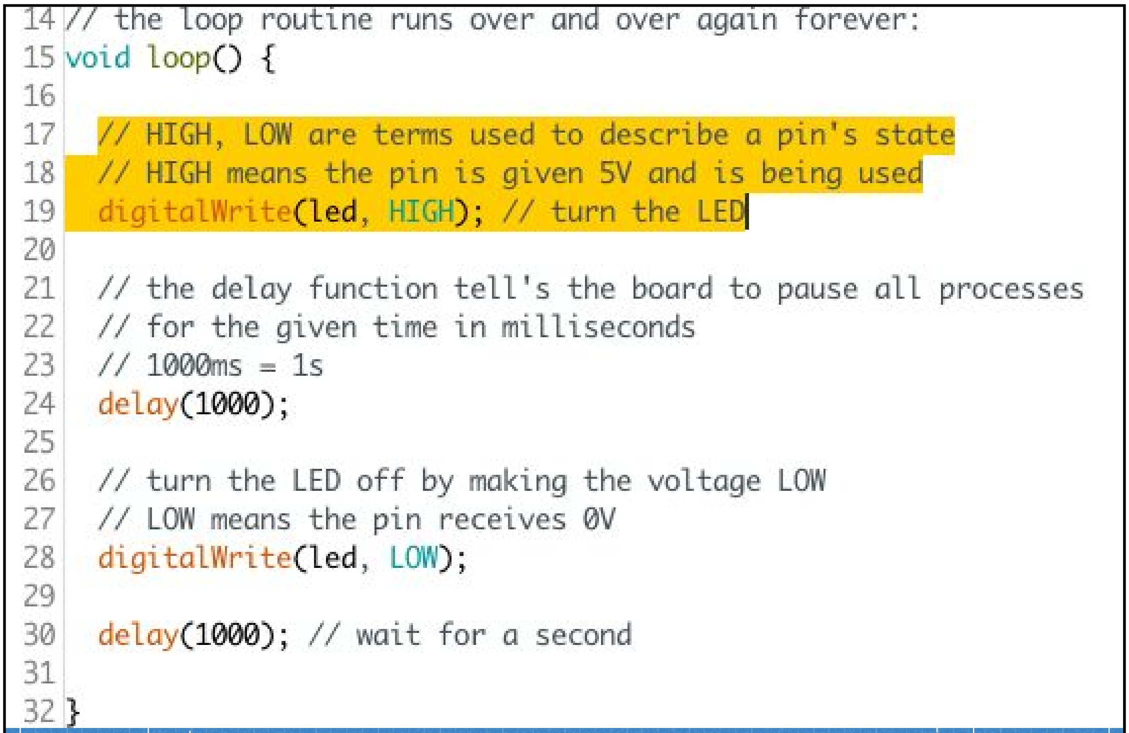
```
// the setup function runs ONCE the board is powered/restarted:
void setup() {
// initialize the pin to be used for output signals
  pinMode(led, OUTPUT);
}
```

setup()



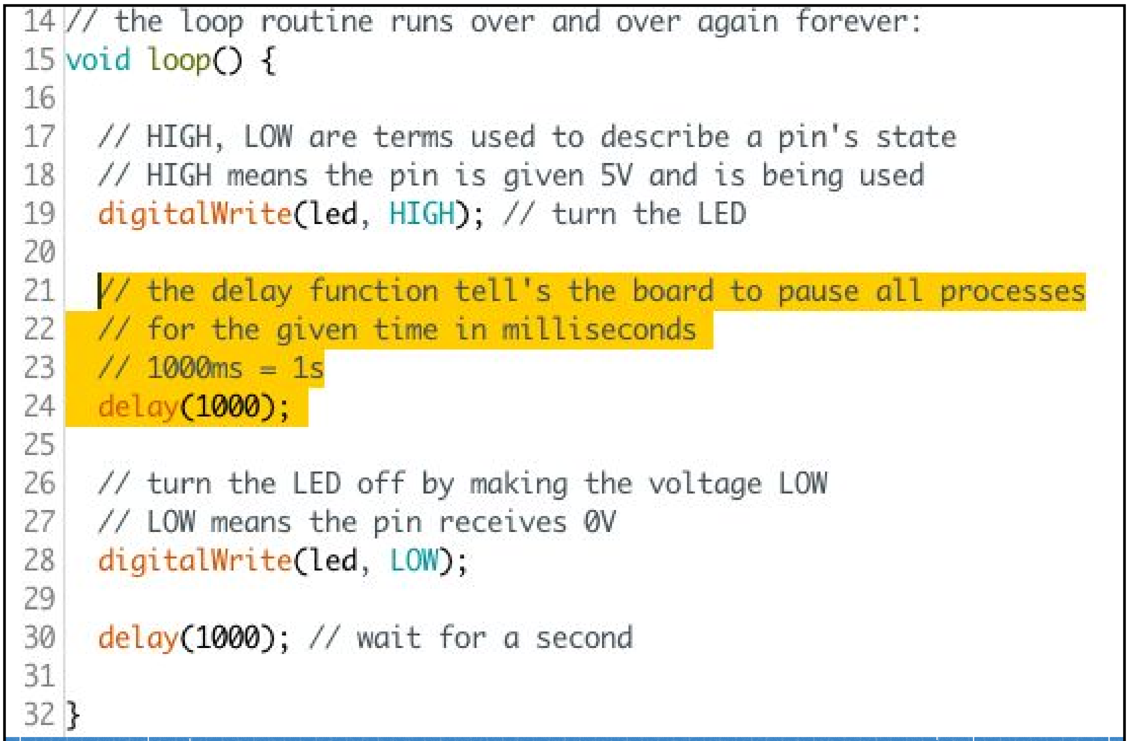
```
// the loop routine runs over and over again forever:
void loop() {

  // HIGH, LOW are terms used to describe a pin's state
  // HIGH means the pin is given 5V and is being used
  digitalWrite(led, HIGH); // turn the LED

  // the delay function tell's the board to pause all processes
  // for the given time in milliseconds
  // 1000ms = 1s
  delay(1000);

  // turn the LED off by making the voltage LOW
  // LOW means the pin receives 0V
  digitalWrite(led, LOW);

  delay(1000); // wait for a second

}
```

loop()



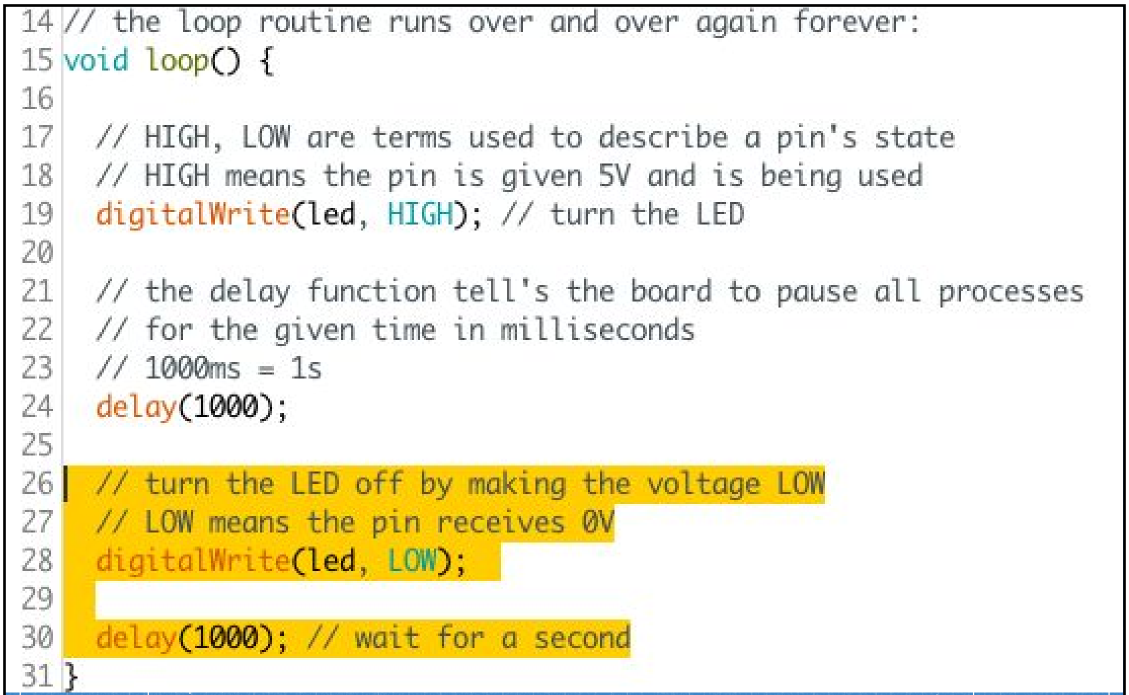
```
// the loop routine runs over and over again forever:
void loop() {

  // HIGH, LOW are terms used to describe a pin's state
  // HIGH means the pin is given 5V and is being used
  digitalWrite(led, HIGH); // turn the LED

  // the delay function tell's the board to pause all processes
  // for the given time in milliseconds
  // 1000ms = 1s
  delay(1000);

  // turn the LED off by making the voltage LOW
  // LOW means the pin receives 0V
  digitalWrite(led, LOW);

  delay(1000); // wait for a second

}
```

digitalWrite(name, HIGH/LOW)



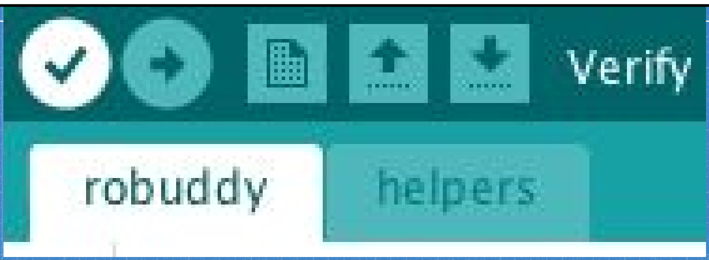
```
// the loop routine runs over and over again forever:
void loop() {

  // HIGH, LOW are terms used to describe a pin's state
  // HIGH means the pin is given 5V and is being used
  digitalWrite(led, HIGH); // turn the LED

  // the delay function tell's the board to pause all processes
  // for the given time in milliseconds
  // 1000ms = 1s
  delay(1000);

  // turn the LED off by making the voltage LOW
  // LOW means the pin receives 0V
  digitalWrite(led, LOW);

  delay(1000); // wait for a second

}
```

delay()

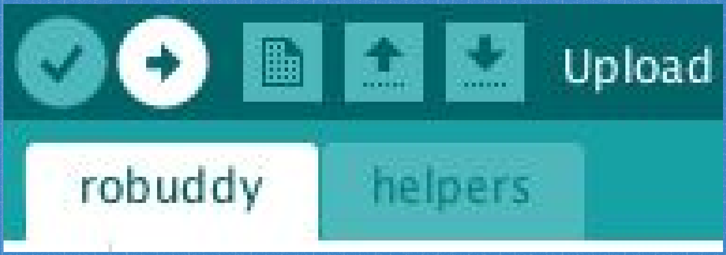


```
// the loop routine runs over and over again forever:
void loop() {

  // HIGH, LOW are terms used to describe a pin's state
  // HIGH means the pin is given 5V and is being used
  digitalWrite(led, HIGH); // turn the LED

  // the delay function tell's the board to pause all processes
  // for the given time in milliseconds
  // 1000ms = 1s
  delay(1000);

  // turn the LED off by making the voltage LOW
  // LOW means the pin receives 0V
  digitalWrite(led, LOW);

  delay(1000); // wait for a second
}
```

Closing the loop



verify



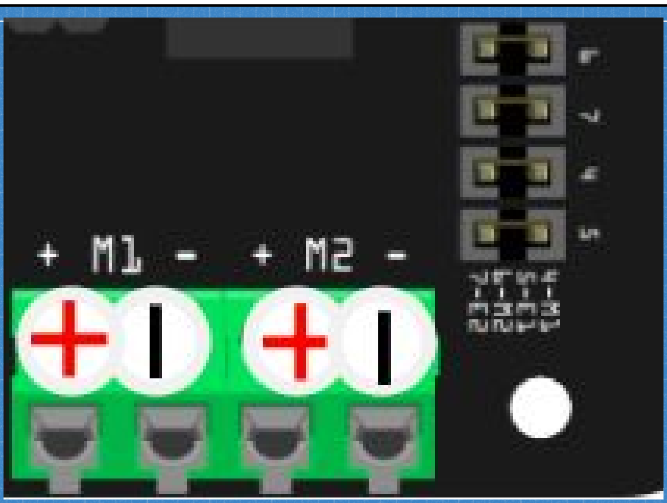
upload

□

👏 🎉

Boom bam your first code ran

Controlling a motor

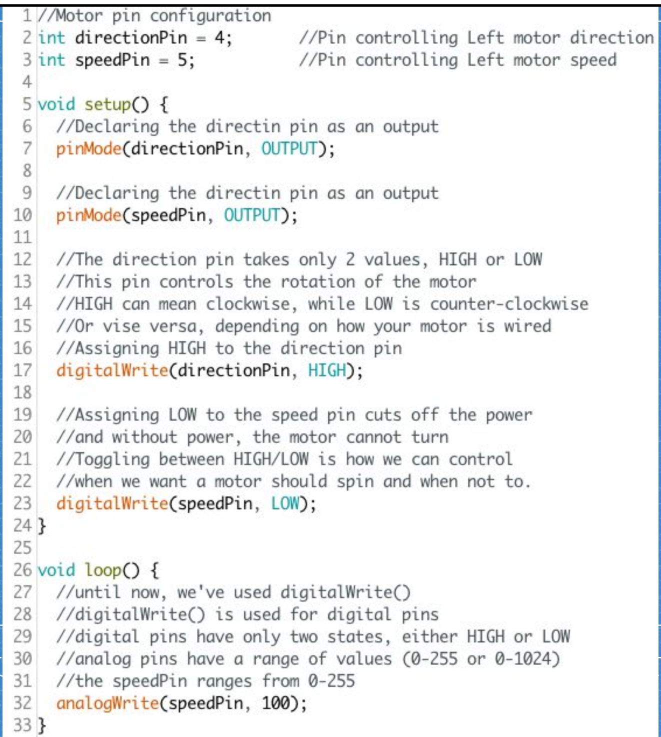


Always double check the polarity of the terminals before wiring

+ to +

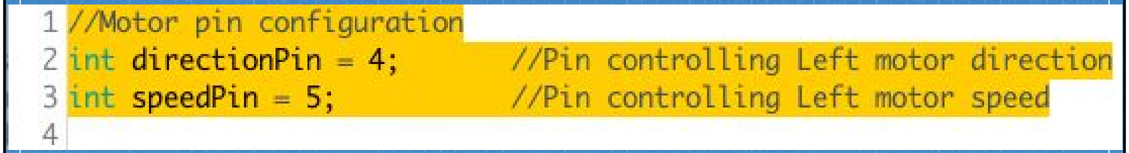
- to -

	Direction Pin#	Speed Pin#
Motor 1	4	5
Motor 2	6	7



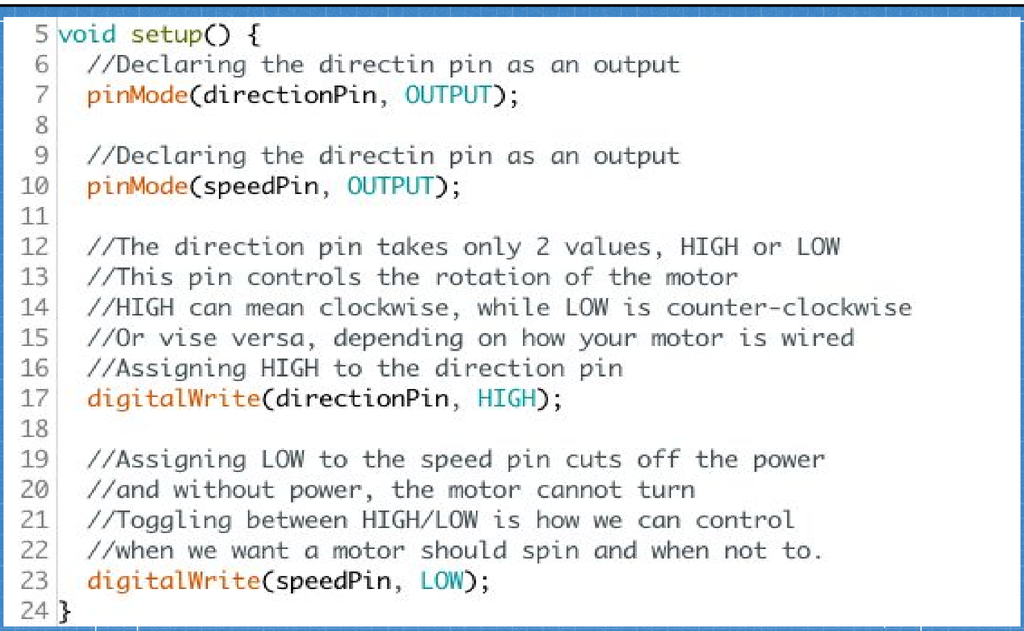
```
//Motor pin configuration
int directionPin = 4;       //Pin controlling Left motor direction
int speedPin = 5;           //Pin controlling Left motor speed

void setup() {
  //Declaring the directin pin as an output
  pinMode(directionPin, OUTPUT);

  //Declaring the directin pin as an output
  pinMode(speedPin, OUTPUT);

  //The direction pin takes only 2 values, HIGH or LOW
  //This pin controls the rotation of the motor
  //HIGH can mean clockwise, while LOW is counter-clockwise
  //Or vise versa, depending on how your motor is wired
  //Assigning HIGH to the direction pin
  digitalWrite(directionPin, HIGH);

  //Assigning LOW to the speed pin cuts off the power
  //and without power, the motor cannot turn
  //Toggling between HIGH/LOW is how we can control
  //when we want a motor should spin and when not to.
  digitalWrite(speedPin, LOW);
}

void loop() {
  //until now, we've used digitalWrite()
  //digitalWrite() is used for digital pins
  //digital pins have only two states, either HIGH or LOW
  //analog pins have a range of values (0-255 or 0-1024)
  //the speedPin ranges from 0-255
  analogWrite(speedPin, 100);
}
```

The motor



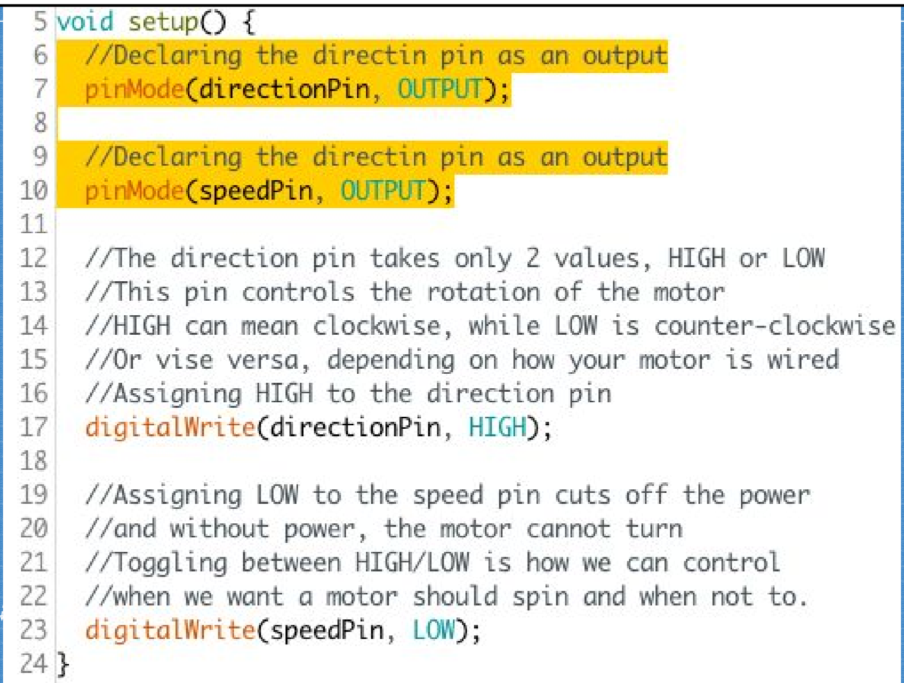
```
//Motor pin configuration
int directionPin = 4;       //Pin controlling Left motor direction
int speedPin = 5;           //Pin controlling Left motor speed

```

variables



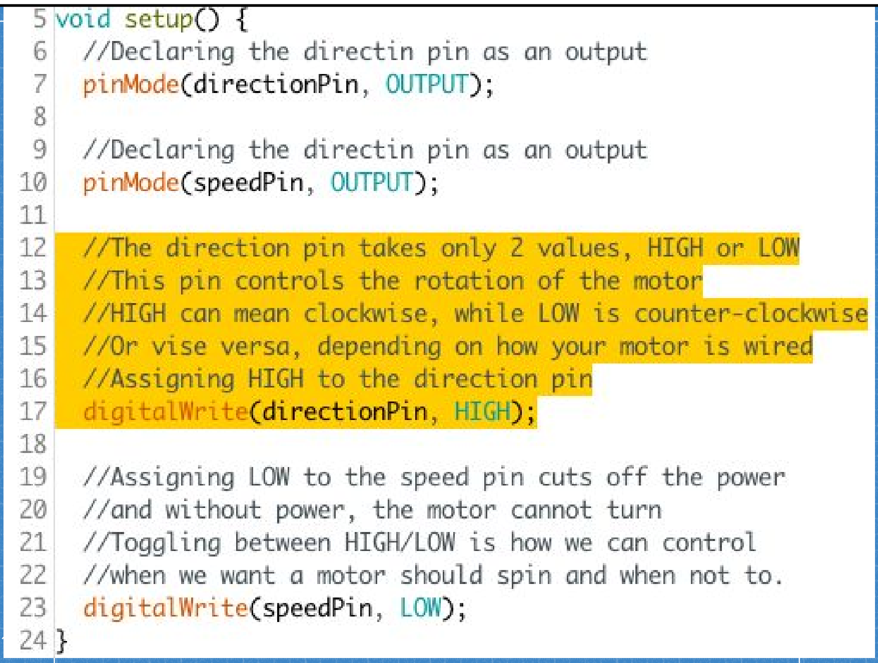
```
void setup() {
  //Declaring the directin pin as an output
  pinMode(directionPin, OUTPUT);

  //Declaring the directin pin as an output
  pinMode(speedPin, OUTPUT);

  //The direction pin takes only 2 values, HIGH or LOW
  //This pin controls the rotation of the motor
  //HIGH can mean clockwise, while LOW is counter-clockwise
  //Or vise versa, depending on how your motor is wired
  //Assigning HIGH to the direction pin
  digitalWrite(directionPin, HIGH);

  //Assigning LOW to the speed pin cuts off the power
  //and without power, the motor cannot turn
  //Toggling between HIGH/LOW is how we can control
  //when we want a motor should spin and when not to.
  digitalWrite(speedPin, LOW);
}
```

setup()



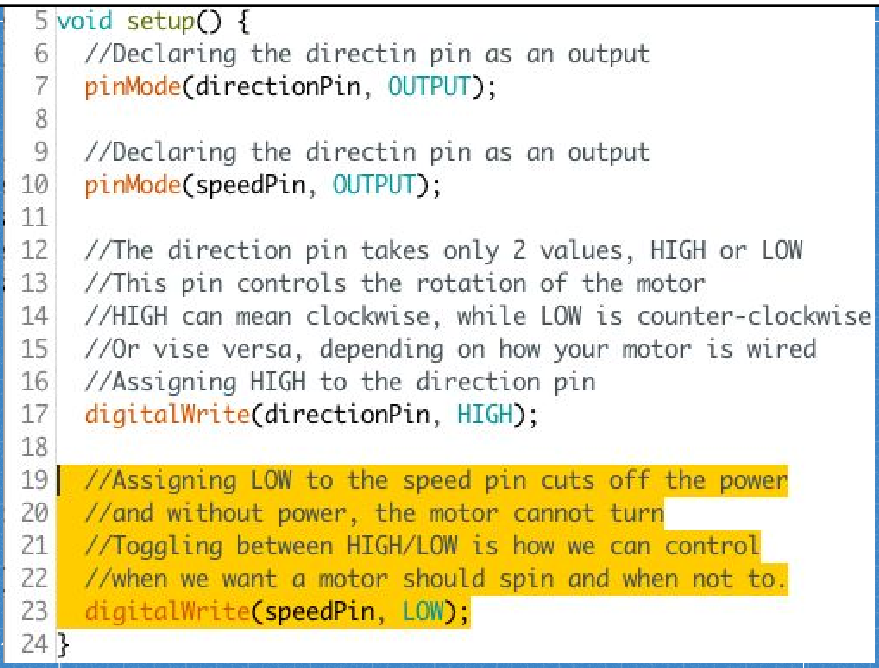
```
void setup() {
  //Declaring the directin pin as an output
  pinMode(directionPin, OUTPUT);

  //Declaring the directin pin as an output
  pinMode(speedPin, OUTPUT);

  //The direction pin takes only 2 values, HIGH or LOW
  //This pin controls the rotation of the motor
  //HIGH can mean clockwise, while LOW is counter-clockwise
  //Or vise versa, depending on how your motor is wired
  //Assigning HIGH to the direction pin
  digitalWrite(directionPin, HIGH);

  //Assigning LOW to the speed pin cuts off the power
  //and without power, the motor cannot turn
  //Toggling between HIGH/LOW is how we can control
  //when we want a motor should spin and when not to.
  digitalWrite(speedPin, LOW);
}
```

pinMode()



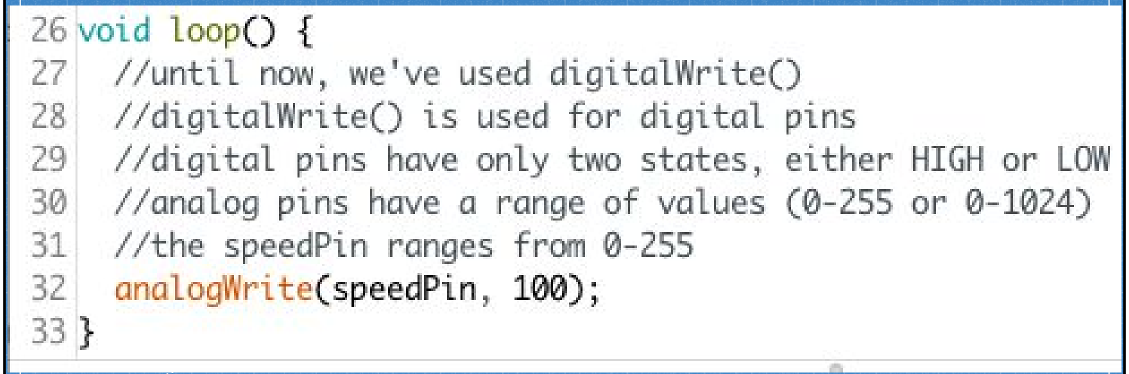
```
void setup() {
  //Declaring the directin pin as an output
  pinMode(directionPin, OUTPUT);

  //Declaring the directin pin as an output
  pinMode(speedPin, OUTPUT);

  //The direction pin takes only 2 values, HIGH or LOW
  //This pin controls the rotation of the motor
  //HIGH can mean clockwise, while LOW is counter-clockwise
  //Or vise versa, depending on how your motor is wired
  //Assigning HIGH to the direction pin
  digitalWrite(directionPin, HIGH);

  //Assigning LOW to the speed pin cuts off the power
  //and without power, the motor cannot turn
  //Toggling between HIGH/LOW is how we can control
  //when we want a motor should spin and when not to.
  digitalWrite(speedPin, LOW);
}
```

digitalWrite(directionPin,HIGH)



```
void setup() {
  //Declaring the directin pin as an output
  pinMode(directionPin, OUTPUT);

  //Declaring the directin pin as an output
  pinMode(speedPin, OUTPUT);

  //The direction pin takes only 2 values, HIGH or LOW
  //This pin controls the rotation of the motor
  //HIGH can mean clockwise, while LOW is counter-clockwise
  //Or vise versa, depending on how your motor is wired
  //Assigning HIGH to the direction pin
  digitalWrite(directionPin, HIGH);

  //Assigning LOW to the speed pin cuts off the power
  //and without power, the motor cannot turn
  //Toggling between HIGH/LOW is how we can control
  //when we want a motor should spin and when not to.
  digitalWrite(speedPin, LOW);
}
```

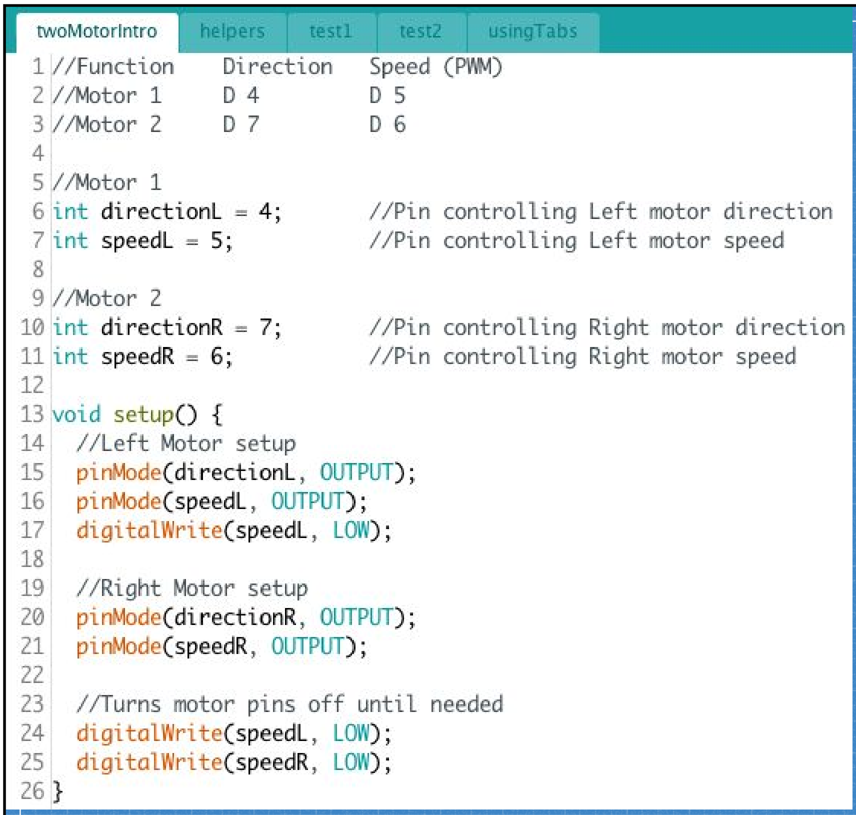
digitalWrite(speedPin,HIGH)



```
void loop() {
  //until now, we've used digitalWrite()
  //digitalWrite() is used for digital pins
  //digital pins have only two states, either HIGH or LOW
  //analog pins have a range of values (0-255 or 0-1024)
  //the speedPin ranges from 0-255
  analogWrite(speedPin, 100);
}
```

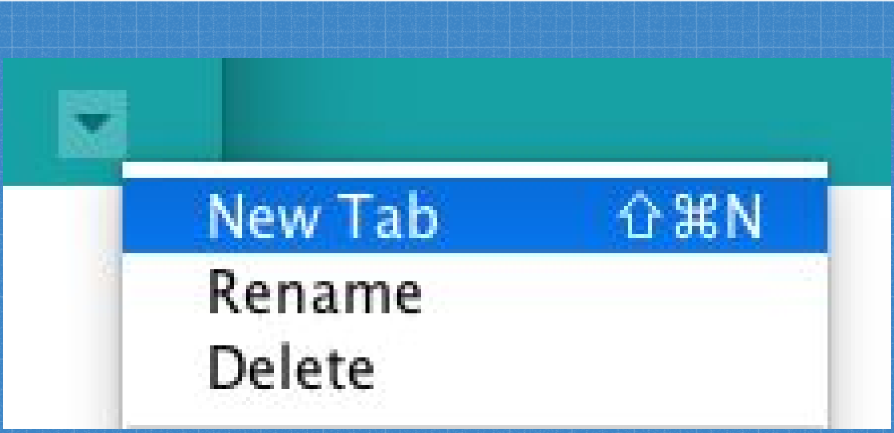
loop()

Controlling 2 motors

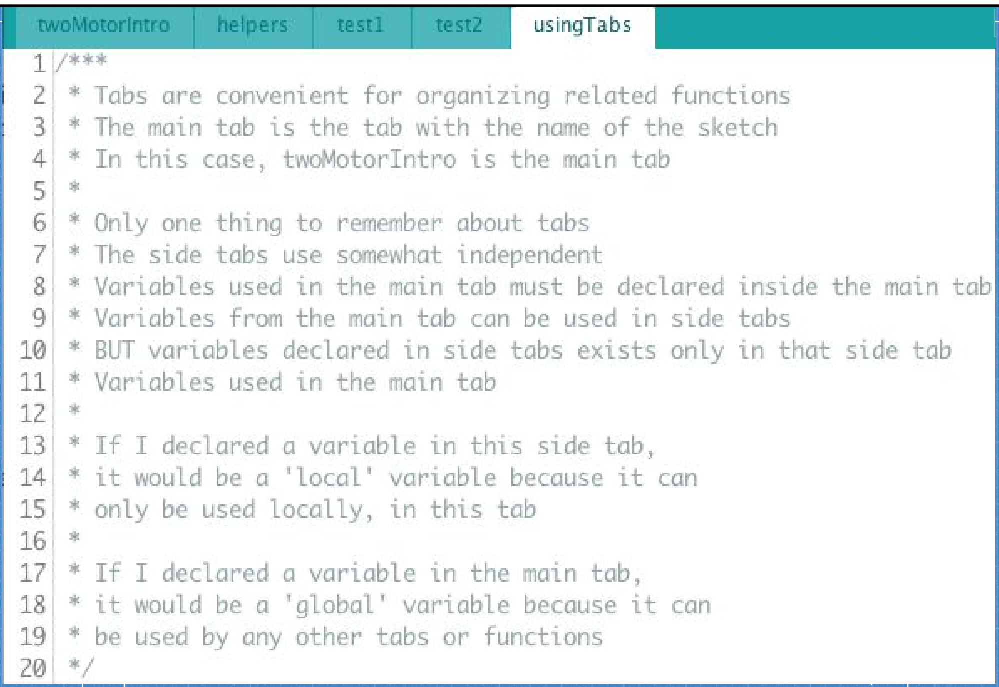


```
//Function     Direction     Speed (PWM)
//Motor 1      D 4           D 5
//Motor 2      D 7           D 6

//Motor 1
int directionL = 4;          //Pin controlling Left motor direction
int speedL = 5;              //Pin controlling Left motor speed

//Motor 2
int directionR = 7;          //Pin controlling Right motor direction
int speedR = 6;              //Pin controlling Right motor speed

void setup() {
  //Left Motor setup
  pinMode(directionL, OUTPUT);
  pinMode(speedL, OUTPUT);
  digitalWrite(speedL, LOW);

  //Right Motor setup
  pinMode(directionR, OUTPUT);
  pinMode(speedR, OUTPUT);

  //Turns motor pins off until needed
  digitalWrite(speedL, LOW);
  digitalWrite(speedR, LOW);
}
```

Variables
&
setup

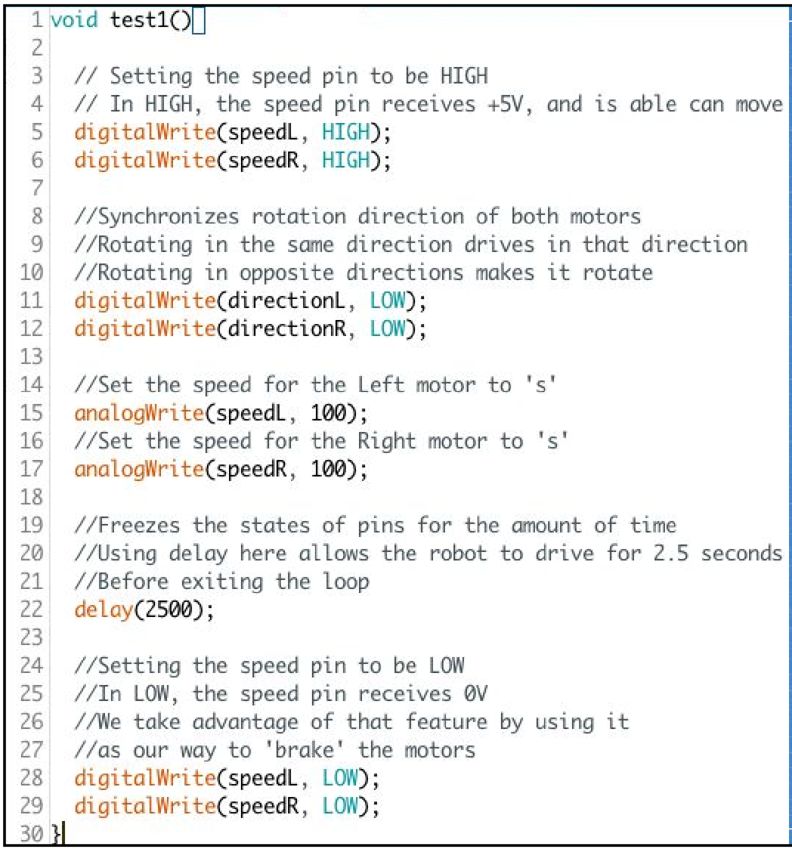


tabs



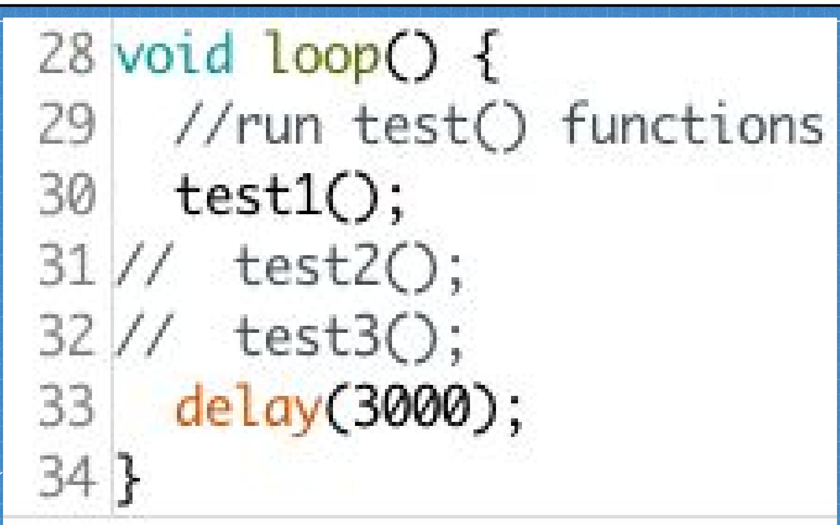
```
/***
 * Tabs are convenient for organizing related functions
 * The main tab is the tab with the name of the sketch
 * In this case, twoMotorIntro is the main tab
 *
 * Only one thing to remember about tabs
 * The side tabs use somewhat independent
 * Variables used in the main tab must be declared inside the main tab
 * Variables from the main tab can be used in side tabs
 * BUT variables declared in side tabs exists only in that side tab
 * Variables used in the main tab
 *
 * If I declared a variable in this side tab,
 * it would be a 'local' variable because it can
 * only be used locally, in this tab
 *
 * If I declared a variable in the main tab,
 * it would be a 'global' variable because it can
 * be used by any other tabs or functions
 */
```

tabs



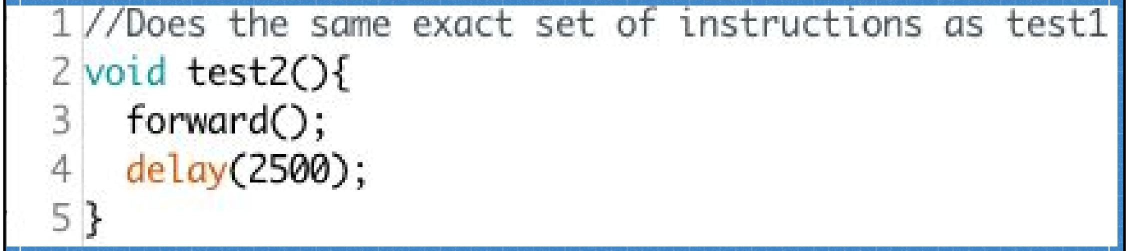
```
void test1(){

  // Setting the speed pin to be HIGH
  // In HIGH, the speed pin receives +5V, and is able can move
  digitalWrite(speedL, HIGH);
  digitalWrite(speedR, HIGH);

  //Synchronizes rotation direction of both motors
  //Rotating in the same direction drives in that direction
  //Rotating in opposite directions makes it rotate
  digitalWrite(directionL, LOW);
  digitalWrite(directionR, LOW);

  //Set the speed for the Left motor to 's'
  analogWrite(speedL, 100);
  //Set the speed for the Right motor to 's'
  analogWrite(speedR, 100);

  //Freezes the states of pins for the amount of time
  //Using delay here allows the robot to drive for 2.5 seconds
  //Before exiting the loop
  delay(2500);

  //Setting the speed pin to be LOW
  //In LOW, the speed pin receives 0V
  //We take advantage of that feature by using it
  //as our way to 'brake' the motors
  digitalWrite(speedL, LOW);
  digitalWrite(speedR, LOW);
}
```

test1()



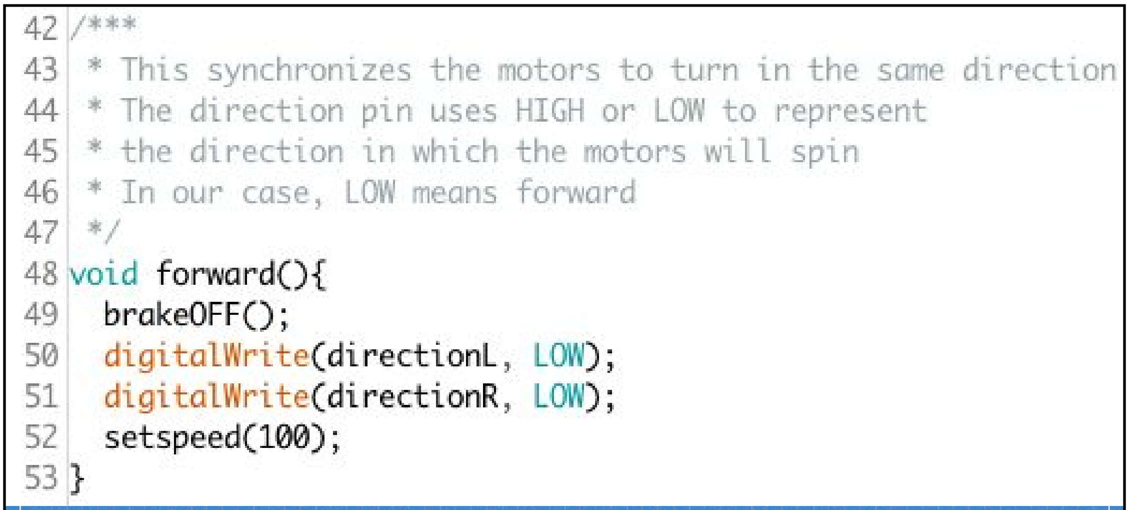
```
void loop() {
  //run test() functions
  test1();
//  test2();
//  test3();
   delay(3000);
}
```

running test1()



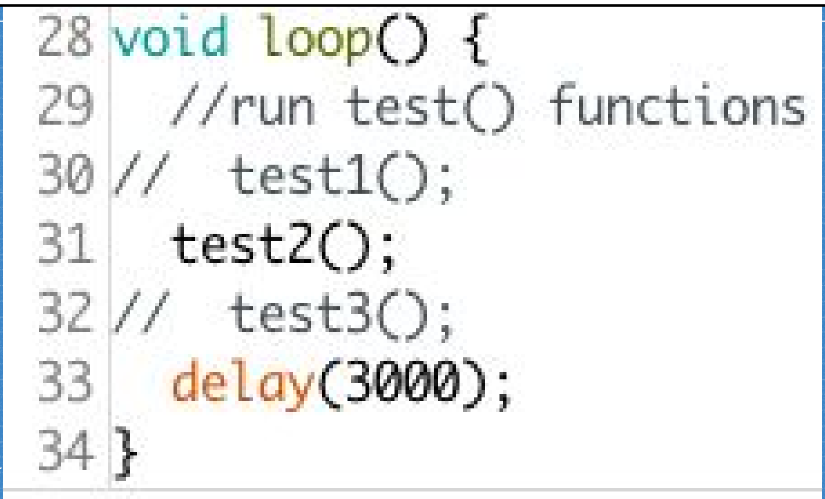
```
//Does the same exact set of instructions as test1
void test2(){
  forward();
  delay(2500);
}
```

test2()



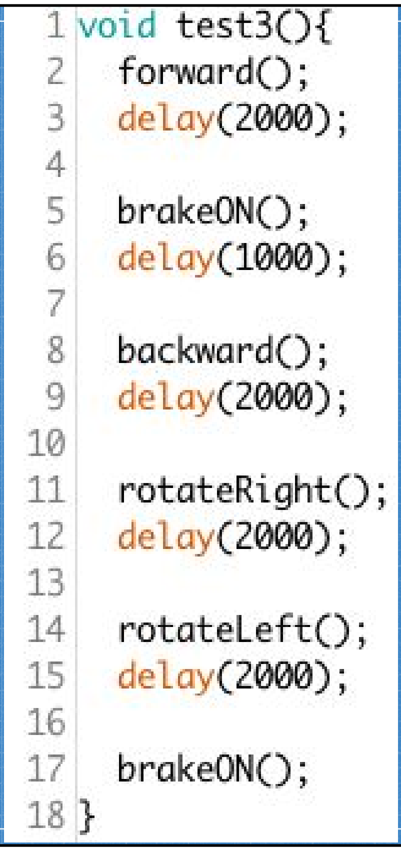
```
/***
 * This synchronizes the motors to turn in the same direction
 * The direction pin uses HIGH or LOW to represent
 * the direction in which the motors will spin
 * In our case, LOW means forward
 */
void forward(){
  brakeOFF();
  digitalWrite(directionL, LOW);
  digitalWrite(directionR, LOW);
  setspeed(100);
}
```

forward()



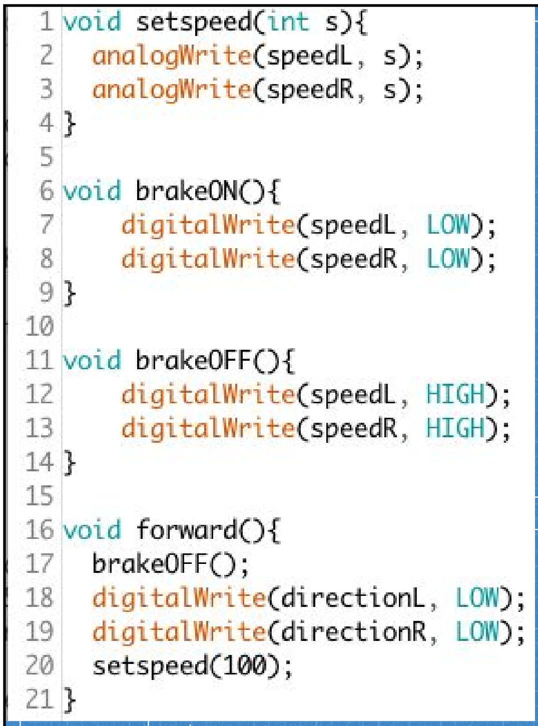
```
void loop() {
  //run test() functions
//  test1();
  test2();
//  test3();
  delay(3000);
}
```

running test2()

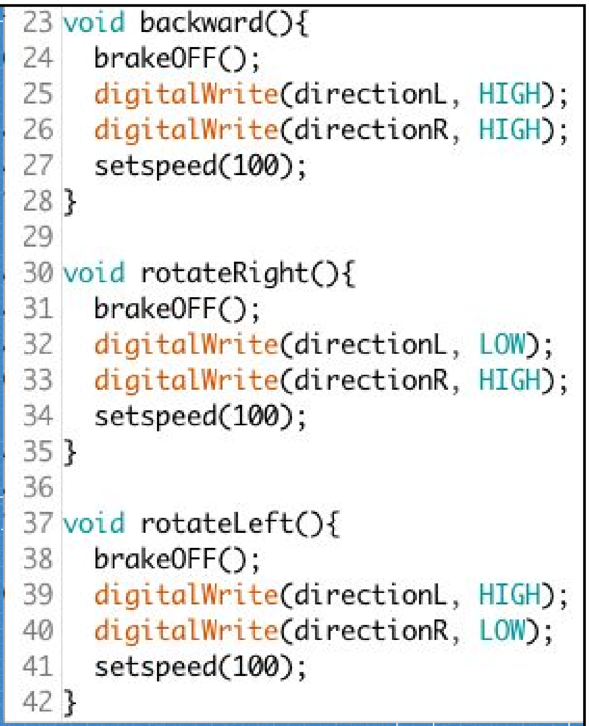


```
void test3(){
  forward();
  delay(2000);

  brakeON();
  delay(1000);

  backward();
  delay(2000);

  rotateRight();
  delay(2000);

  rotateLeft();
  delay(2000);

  brakeON();
}
```

test3()

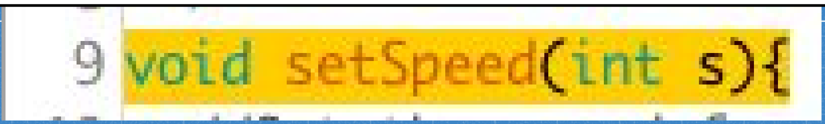


```
void setspeed(int s){
  analogWrite(speedL, s);
  analogWrite(speedR, s);
}

void brakeON(){
    digitalWrite(speedL, LOW);
    digitalWrite(speedR, LOW);
}

void brakeOFF(){
    digitalWrite(speedL, HIGH);
    digitalWrite(speedR, HIGH);
}

void forward(){
  brakeOFF();
  digitalWrite(directionL, LOW);
  digitalWrite(directionR, LOW);
  setspeed(100);
}
```



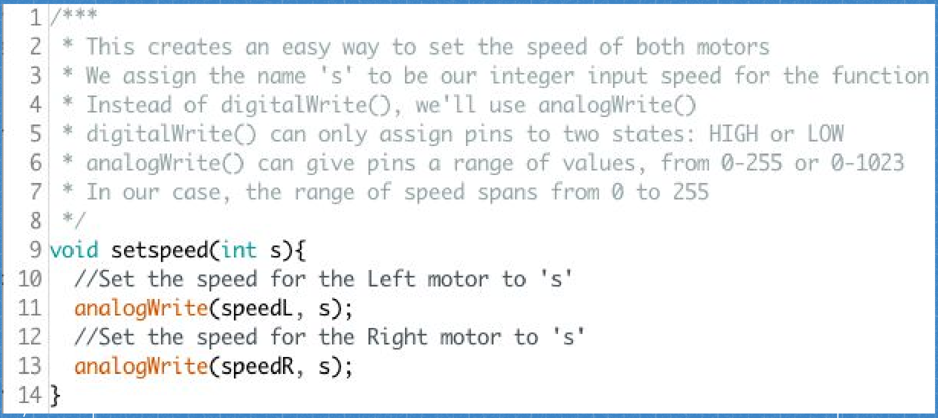
```
void backward(){
  brakeOFF();
  digitalWrite(directionL, HIGH);
  digitalWrite(directionR, HIGH);
  setspeed(100);
}

void rotateRight(){
  brakeOFF();
  digitalWrite(directionL, LOW);
  digitalWrite(directionR, HIGH);
  setspeed(100);
}

void rotateLeft(){
  brakeOFF();
  digitalWrite(directionL, HIGH);
  digitalWrite(directionR, LOW);
  setspeed(100);
}
```

helpers



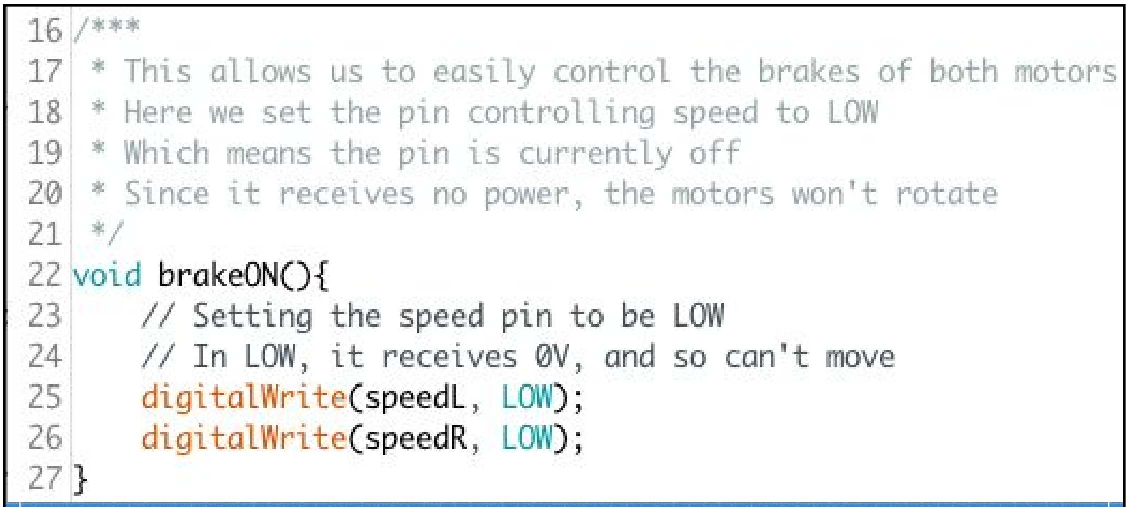
```
void setSpeed(int s){
```

Sometimes function name is taken by a core Arduino function. Core functions are colored orange.



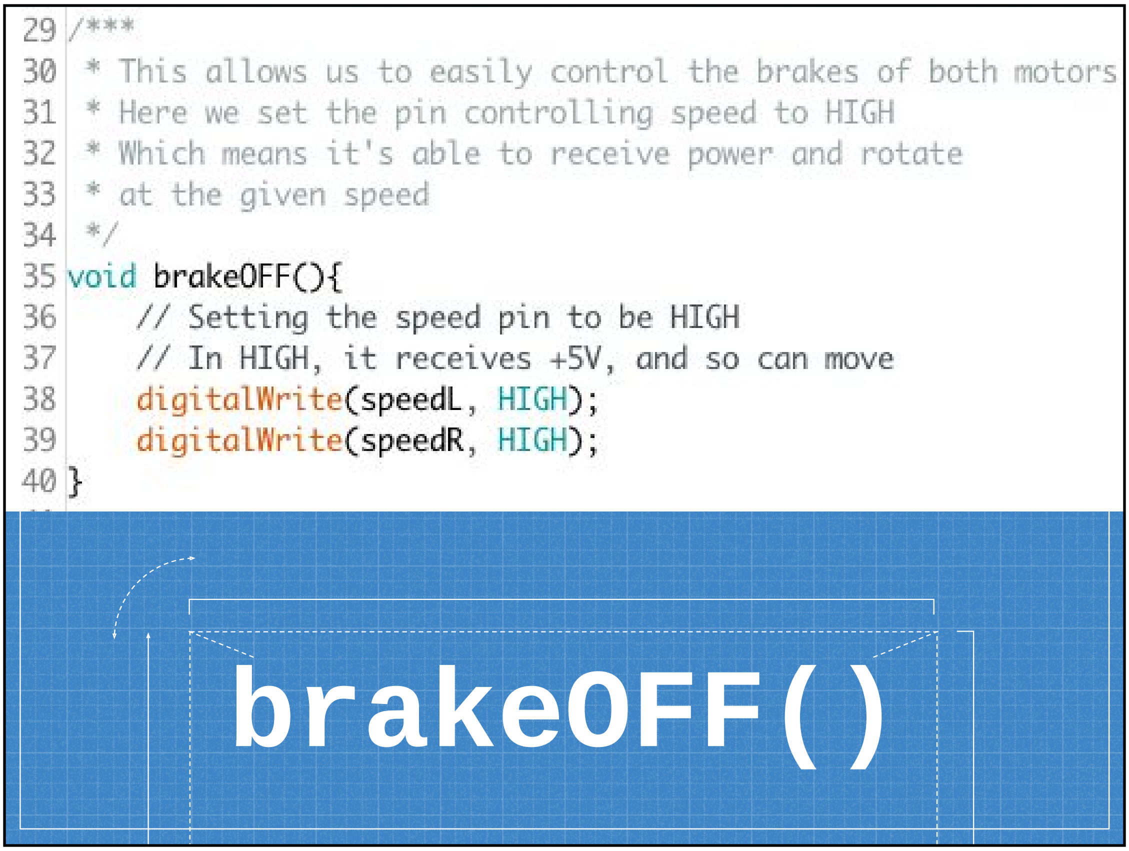
```
/***
 * This creates an easy way to set the speed of both motors
 * We assign the name 's' to be our integer input speed for the function
 * Instead of digitalWrite(), we'll use analogWrite()
 * digitalWrite() can only assign pins to two states: HIGH or LOW
 * analogWrite() can give pins a range of values, from 0-255 or 0-1023
 * In our case, the range of speed spans from 0 to 255
 */
void setspeed(int s){
  //Set the speed for the Left motor to 's'
  analogWrite(speedL, s);
  //Set the speed for the Right motor to 's'
  analogWrite(speedR, s);
}
```

setspeed()



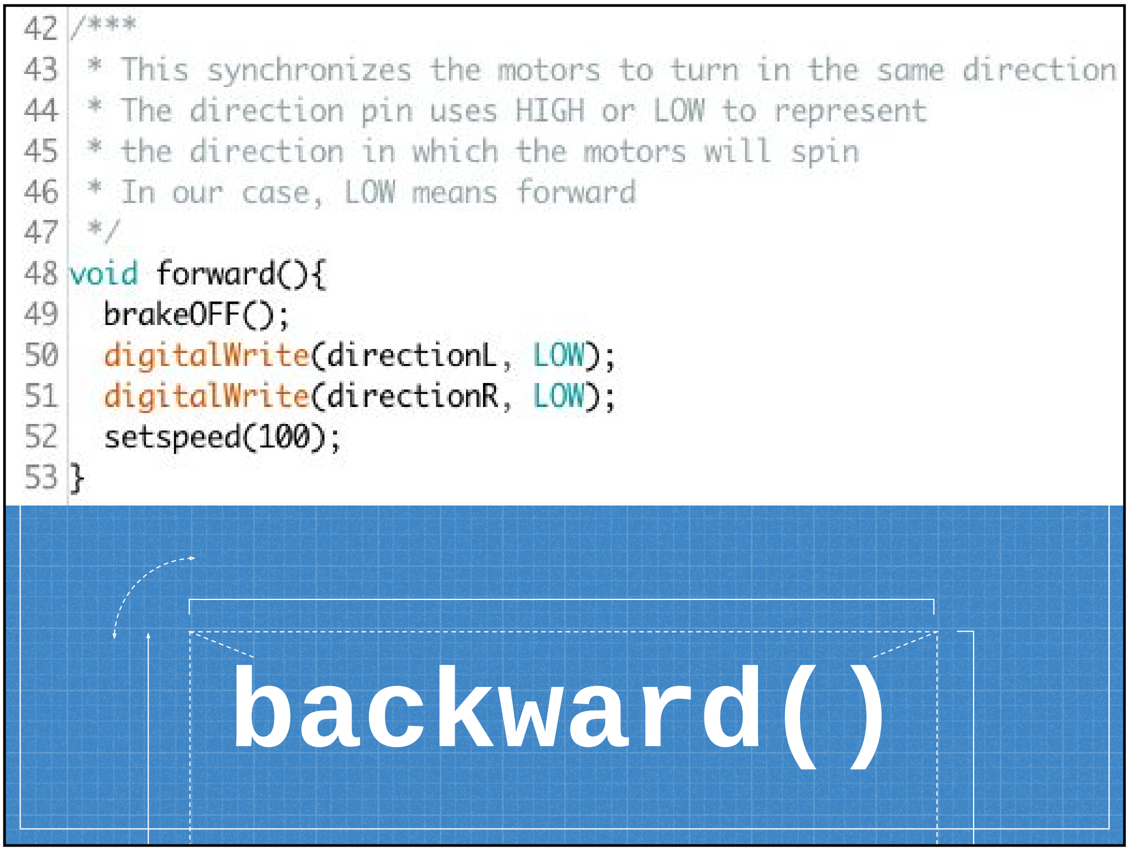
```
/***
 * This allows us to easily control the brakes of both motors
 * Here we set the pin controlling speed to LOW
 * Which means the pin is currently off
 * Since it receives no power, the motors won't rotate
 */
void brakeON(){
    // Setting the speed pin to be LOW
    // In LOW, it receives 0V, and so can't move
    digitalWrite(speedL, LOW);
    digitalWrite(speedR, LOW);
}
```

brakeON()



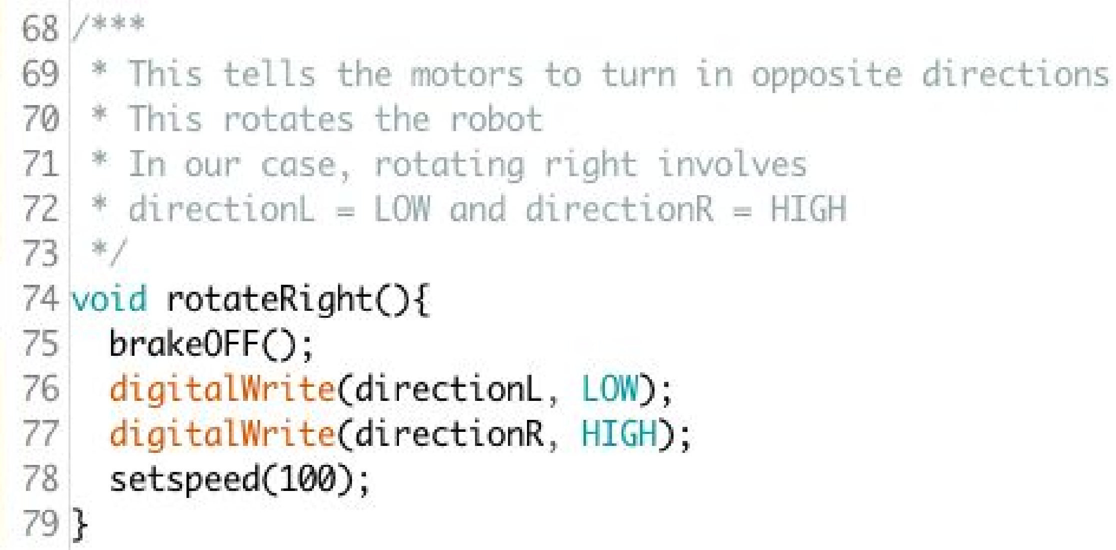
```
/***
 * This allows us to easily control the brakes of both motors
 * Here we set the pin controlling speed to HIGH
 * Which means it's able to receive power and rotate
 * at the given speed
 */
void brakeOFF(){
    // Setting the speed pin to be HIGH
    // In HIGH, it receives +5V, and so can move
    digitalWrite(speedL, HIGH);
    digitalWrite(speedR, HIGH);
}
```

brakeOFF()



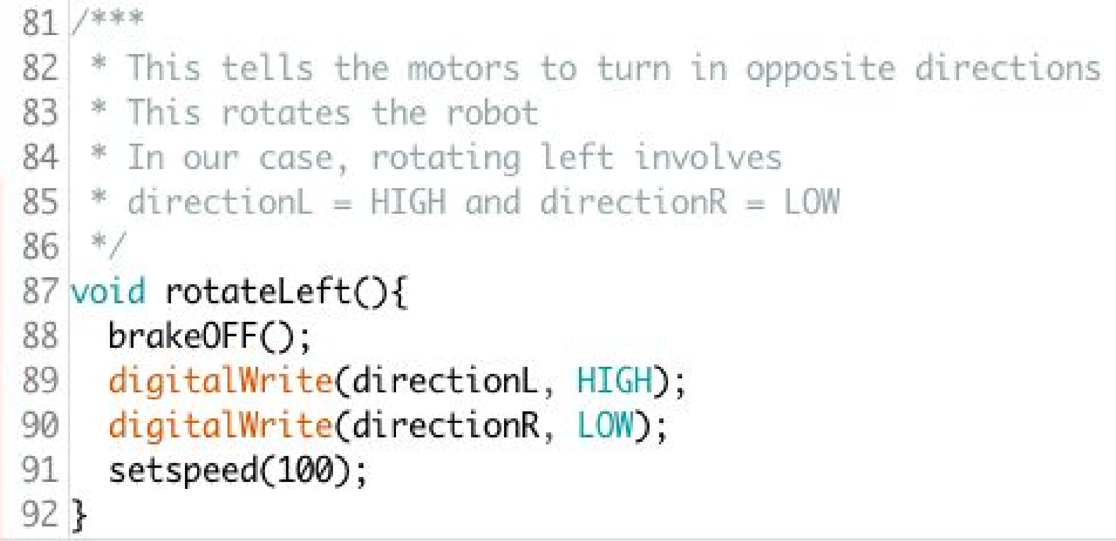
```
/***
 * This synchronizes the motors to turn in the same direction
 * The direction pin uses HIGH or LOW to represent
 * the direction in which the motors will spin
 * In our case, LOW means forward
 */
void forward(){
  brakeOFF();
  digitalWrite(directionL, LOW);
  digitalWrite(directionR, LOW);
  setspeed(100);
}
```

backward()



```
/***
 * This tells the motors to turn in opposite directions
 * This rotates the robot
 * In our case, rotating right involves
 * directionL = LOW and directionR = HIGH
 */
void rotateRight(){
  brakeOFF();
  digitalWrite(directionL, LOW);
  digitalWrite(directionR, HIGH);
  setspeed(100);
}
```

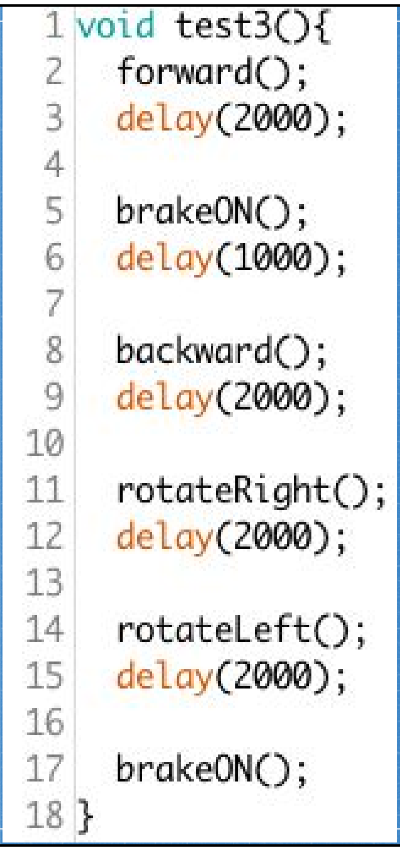
rotateRight()



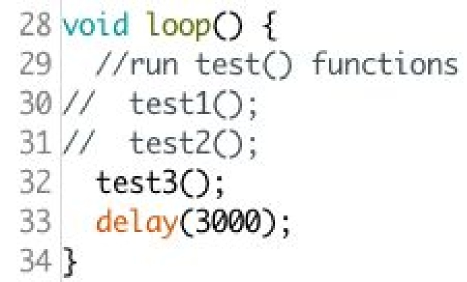
```
/***
 * This tells the motors to turn in opposite directions
 * This rotates the robot
 * In our case, rotating left involves
 * directionL = HIGH and directionR = LOW
 */
void rotateLeft(){
  brakeOFF();
  digitalWrite(directionL, HIGH);
  digitalWrite(directionR, LOW);
  setspeed(100);
}
```

rotateLeft()

test3()



```
void test3(){
  forward();
  delay(2000);

  brakeON();
  delay(1000);

  backward();
  delay(2000);

  rotateRight();
  delay(2000);

  rotateLeft();
  delay(2000);

  brakeON();
}
```



```
void loop() {
  //run test() functions
//  test1();
//  test2();
  test3();
  delay(3000);
}
```