anpri

Associação Nacional de Professores de Informática

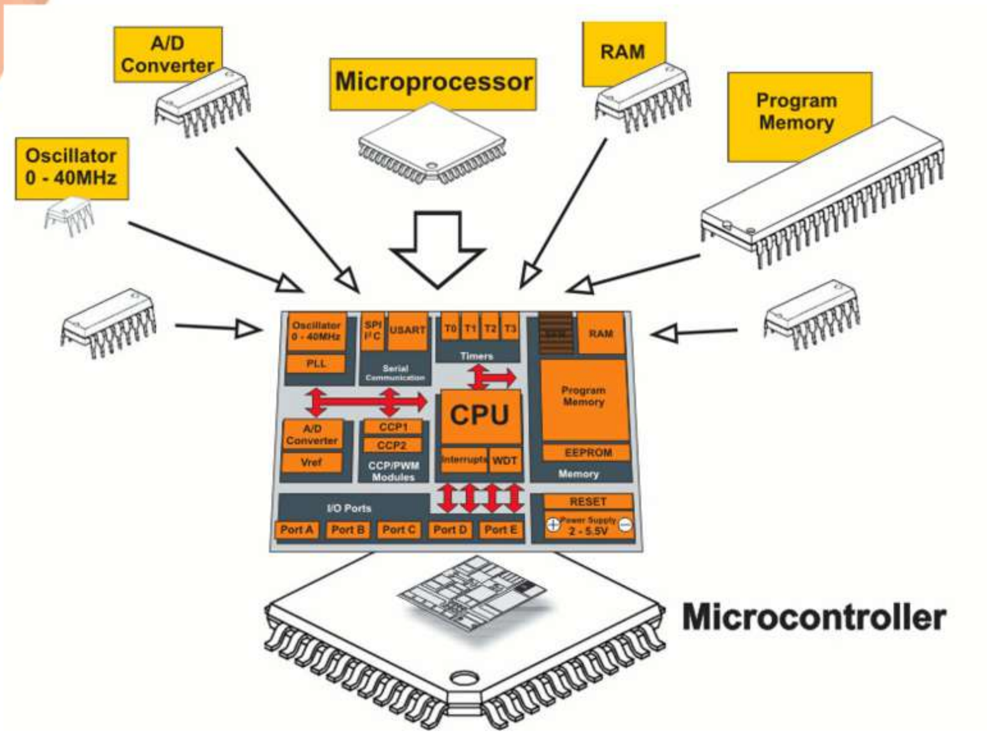
eprols

Educational Robotic and Programming and Learning Scenarios

2020-1-PT01-KA201-078670

Arduino Workshop - Tinkercad

Microcontroler



- A small computer with a simple chip that contains a processor, memory and input/output
- Typically embebed inside some device that controls it
- It's small and low cost.

Arduino

- Free hardware
- Free software
- Different models
- Simple and reliable
- Robust enough for most free robotics (and other projects) activities

Arduino models

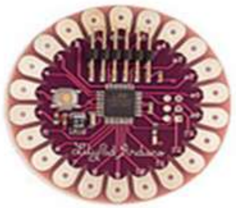
Arduino Uno

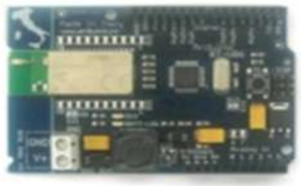
Arduino Leonardo

Arduino Ethernet

Arduino Pro

Arduino Mega 2560


Arduino LilyPad

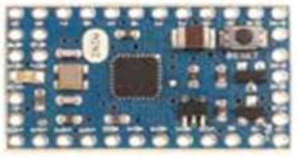

Arduino BT

Arduino Nano

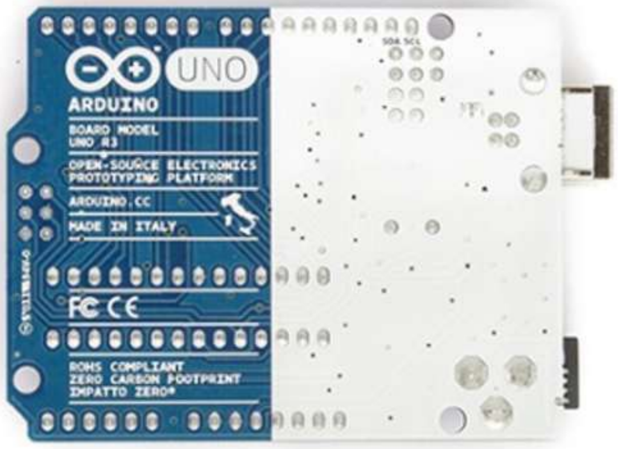
Arduino Mega ADK

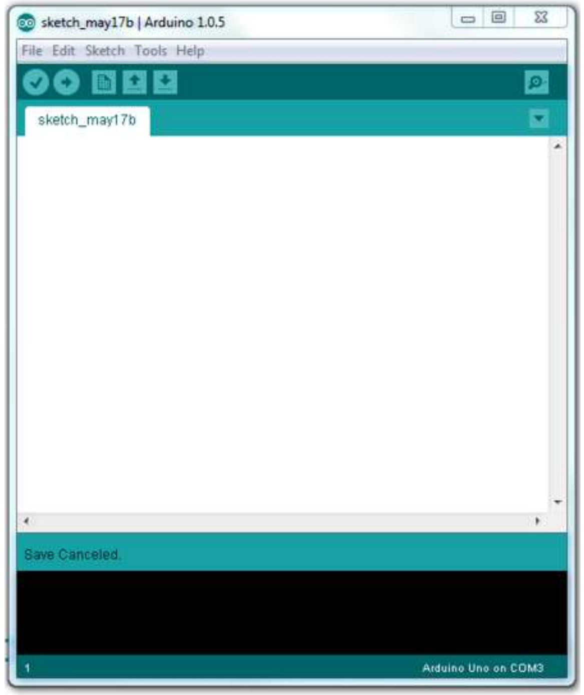
Arduino Fio

USB/Serial Light Adapter


Arduino Mini

Arduino + Software + Connection

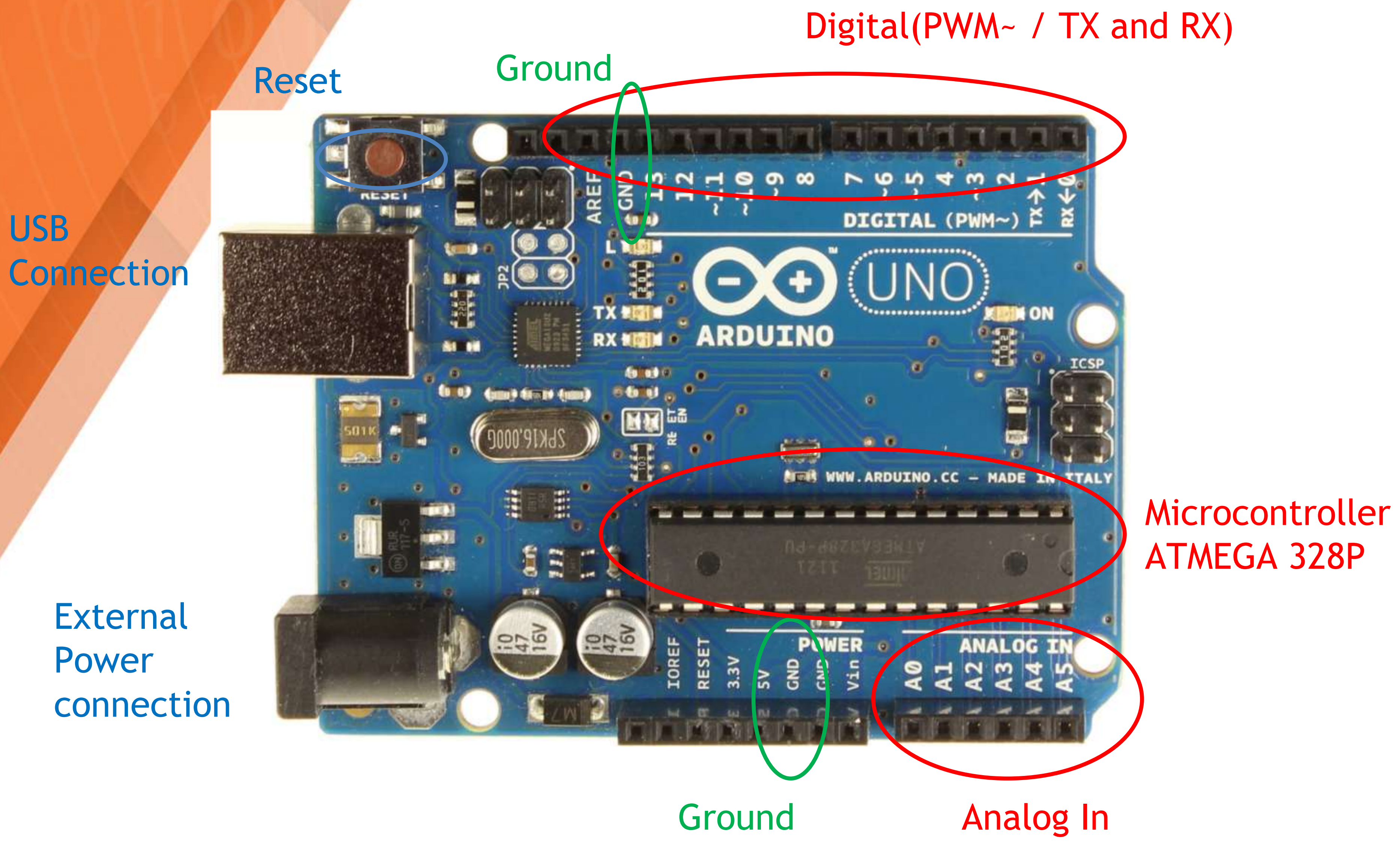




Save Canceled.

1
Arduino Uno on COM3

Arduino


Digital(PWM~ / TX and RX)
Ground
Reset
USB Connection
Microcontroller ATMEGA 328P
External Power connection
Ground
Analog In
DIGITAL (PWM~)
UNO
ARDUINO
POWER
ANALOG IN
IOREF
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
TX
RX
ON
ICSP
WWW.ARDUINO.CC – MADE IN ITALY

Ports

- Arduino has digital ports and analog ports:
- The ports are for communication between the Arduino and external devices: read a button, turn on a led or a lamp.
- Arduino UNO, has 14 digital ports and 6 analog ports (which can also be used as digital ports).
- The values read on an analog port range from 0 to 1023 (10 bits), where 0 represents 0V and 1023 represents 5V.

Ports

- *Pulse Width Modulation* is a technique used by digital systems to vary the average value of a periodic waveform, for example, motor speed control; light variation of LEDs; generation of analog signals; generation of audio signals.
- TX / RX – Arduino communication ports, for example, allow connection between 2 Arduinos, Bluetooth connection, GPS connection
- A4 (SDA) / A5 (SCL) – communication protocol between devices that “talk” I2C. It works with 1 master (coordinate communication) and up to 112 slaves (identified by memory addresses)

Digital Ports

- Digital ports work with well-defined values, that is, in case of Arduino these values are 0V and 5V.
- 0V indicates the absence of a signal and 5V indicates the presence of a signal.
- To write to a digital port, we use the digitalWrite(pin, status) function.
- To read a value from a digital port, we use the digitalRead(pin) function.

Analog Ports

- Analog ports are used for data input.
- They are identified as A0, A1, A2, A3, A4 and A5.
- By default all analog ports are defined as input data, so it is not necessary to make this definition in the setup() function.
- The values read on an analog port range from 0V to 5V.
- To read a value from an analog port, just use the analogRead(pin) function.

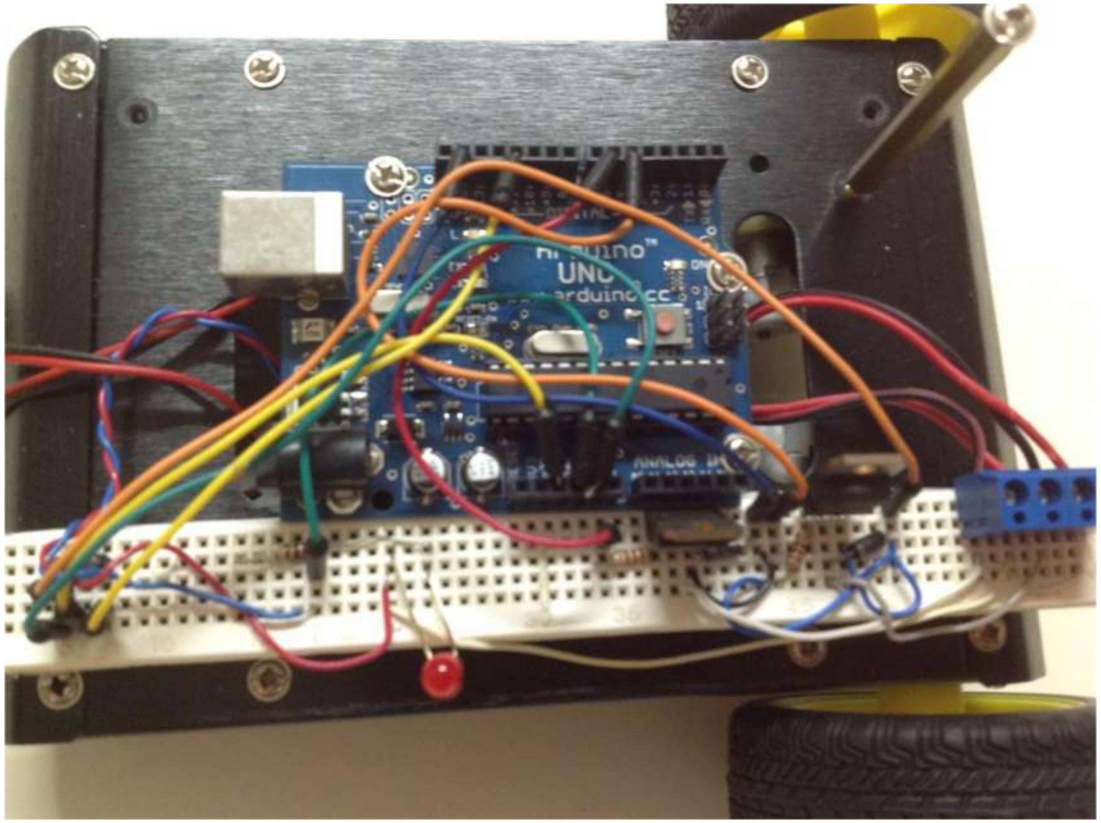
Ports

- To define a port as input or output, it is necessary to make this situation explicit in the program.
- The pinMode(pin, state) function is used to define whether the port will be input or output data.
- Example:

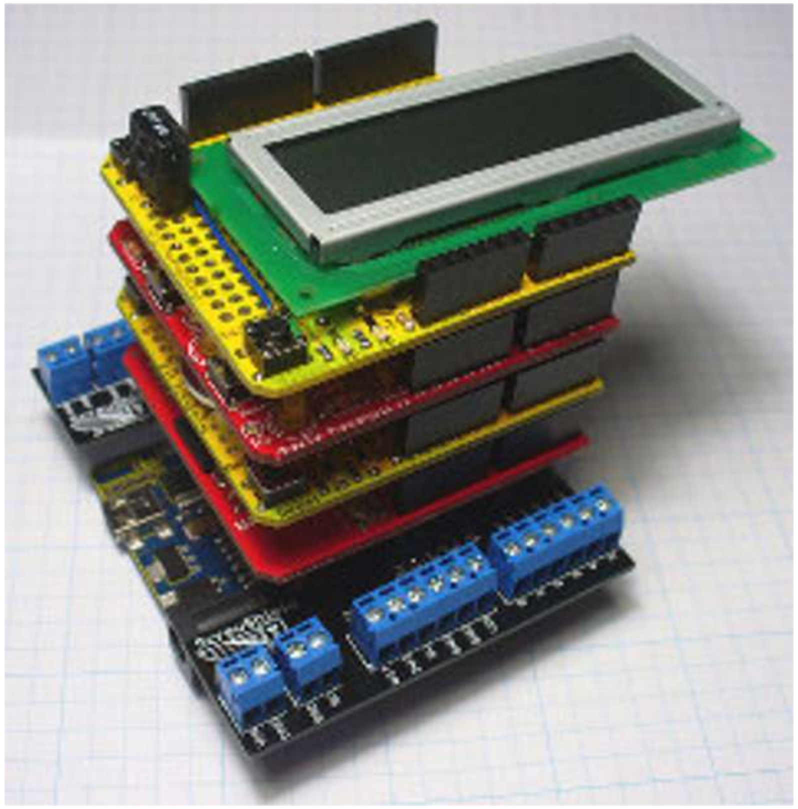
- Defines that port 13 will be output
- pinMode(13, OUTPUT)

- Defines that port 7 will be input
- pinMode(7, INPUT)

Arduino doesn’t works alone



Shields



- They are Arduino extensions and add specific features, inheriting Arduino features

Shields

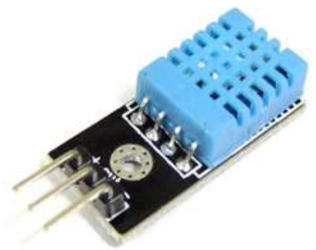
Ethernet

Motors

Sensors

Sensors

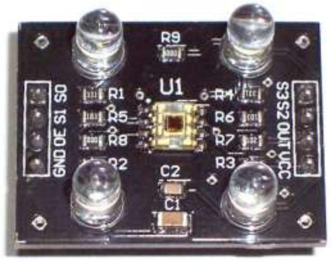
Sound


Temperatur
e

Ultrasonic

Laser

IR


Color

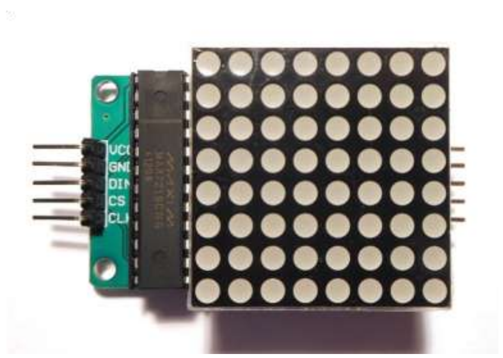
QTR-8x

Gas

Actuators

Servo

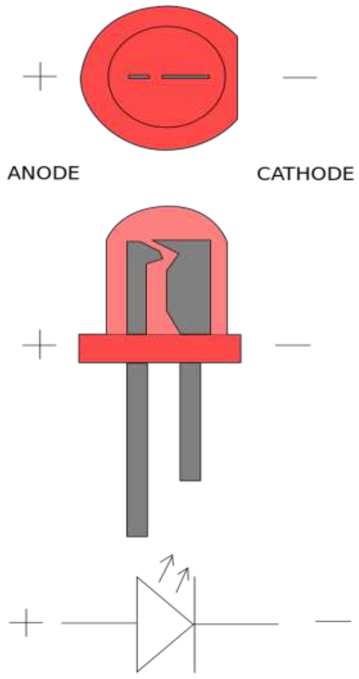
Motors



Matrix

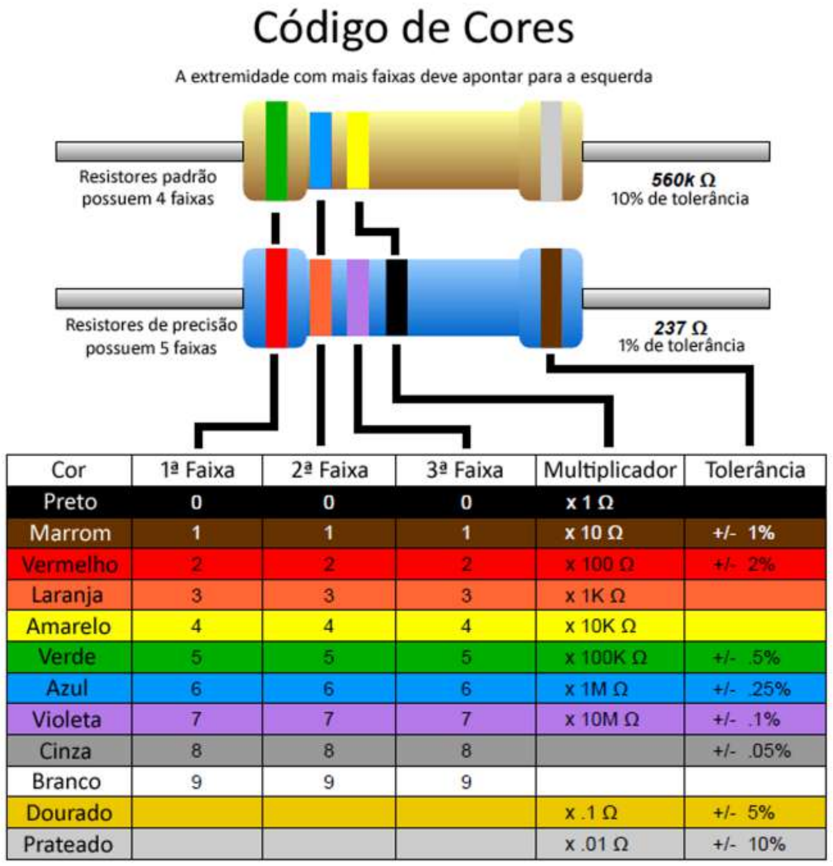
Led's

- The LED (Light Emitting Diode), as it is a diode and not a lamp, lets current pass through in a single direction.



Resistors

- Resistors limit the passage of electrical current, preventing some components from being damaged by excess electrical voltage.


Código de Cores
A extremidade com mais faixas deve apontar para a esquerda
Resistores padrão possuem 4 faixas
560k Ω
10% de tolerância
Resistores de precisão possuem 5 faixas
237 Ω
1% de tolerância

Cor	1ª Faixa	2ª Faixa	3ª Faixa	Multiplicador	Tolerância
Preto	0	0	0	x 1 Ω	
Marrom	1	1	1	x 10 Ω	+/- 1%
Vermelho	2	2	2	x 100 Ω	+/- 2%
Laranja	3	3	3	x 1K Ω	
Amarelo	4	4	4	x 10K Ω	
Verde	5	5	5	x 100K Ω	+/- .5%
Azul	6	6	6	x 1M Ω	+/- .25%
Violeta	7	7	7	x 10M Ω	+/- .1%
Cinza	8	8	8		+/- .05%
Branco	9	9	9		
Dourado				x .1 Ω	+/- 5%
Prateado				x .01 Ω	+/- 10%

Resistors

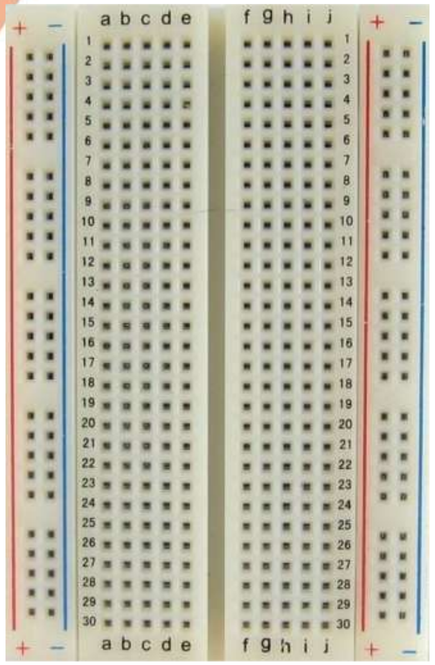
Coal resistance
(Fixed Value Resistance)

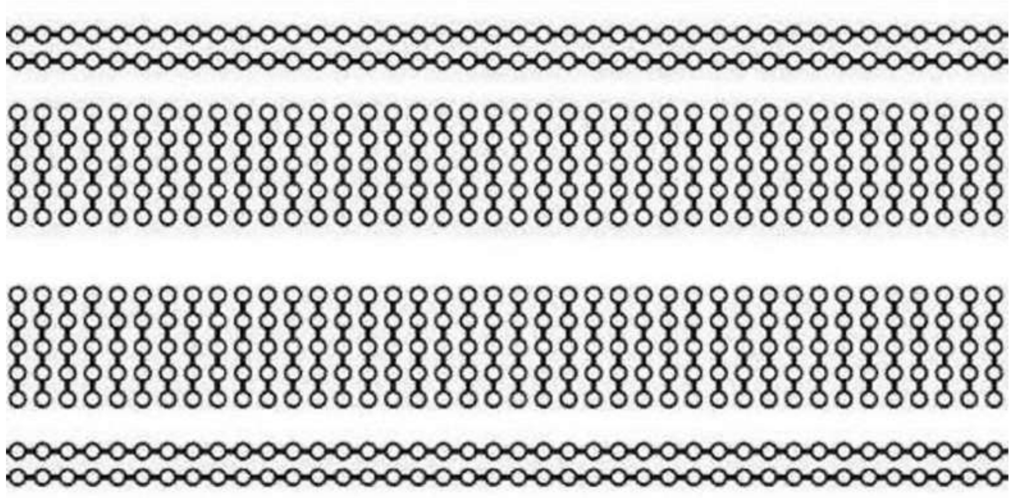
Potentiometer

(Variable resistance)

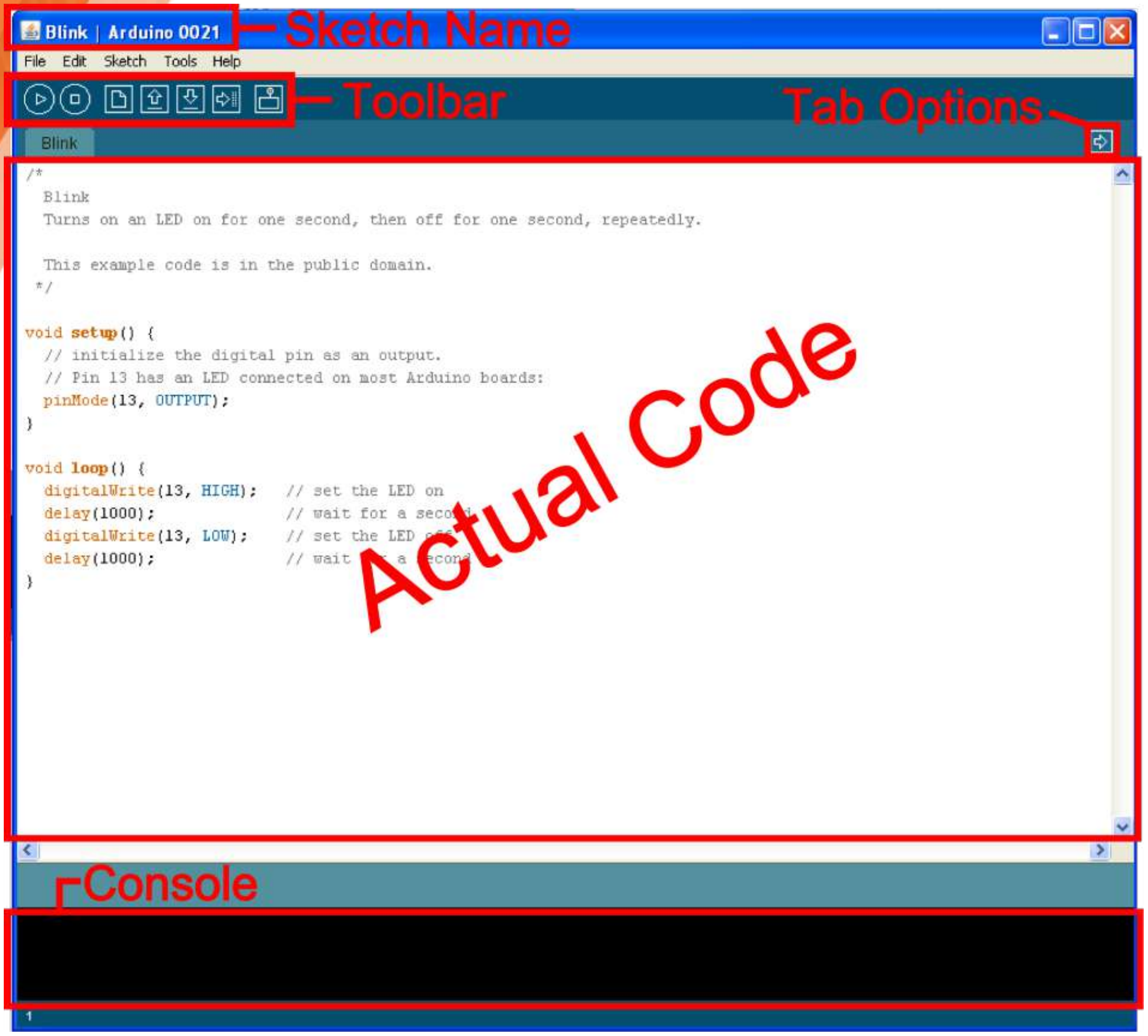
Breadboard

- Tests must be carried out on a prototyping board where the components are fixed without the need for soldering.



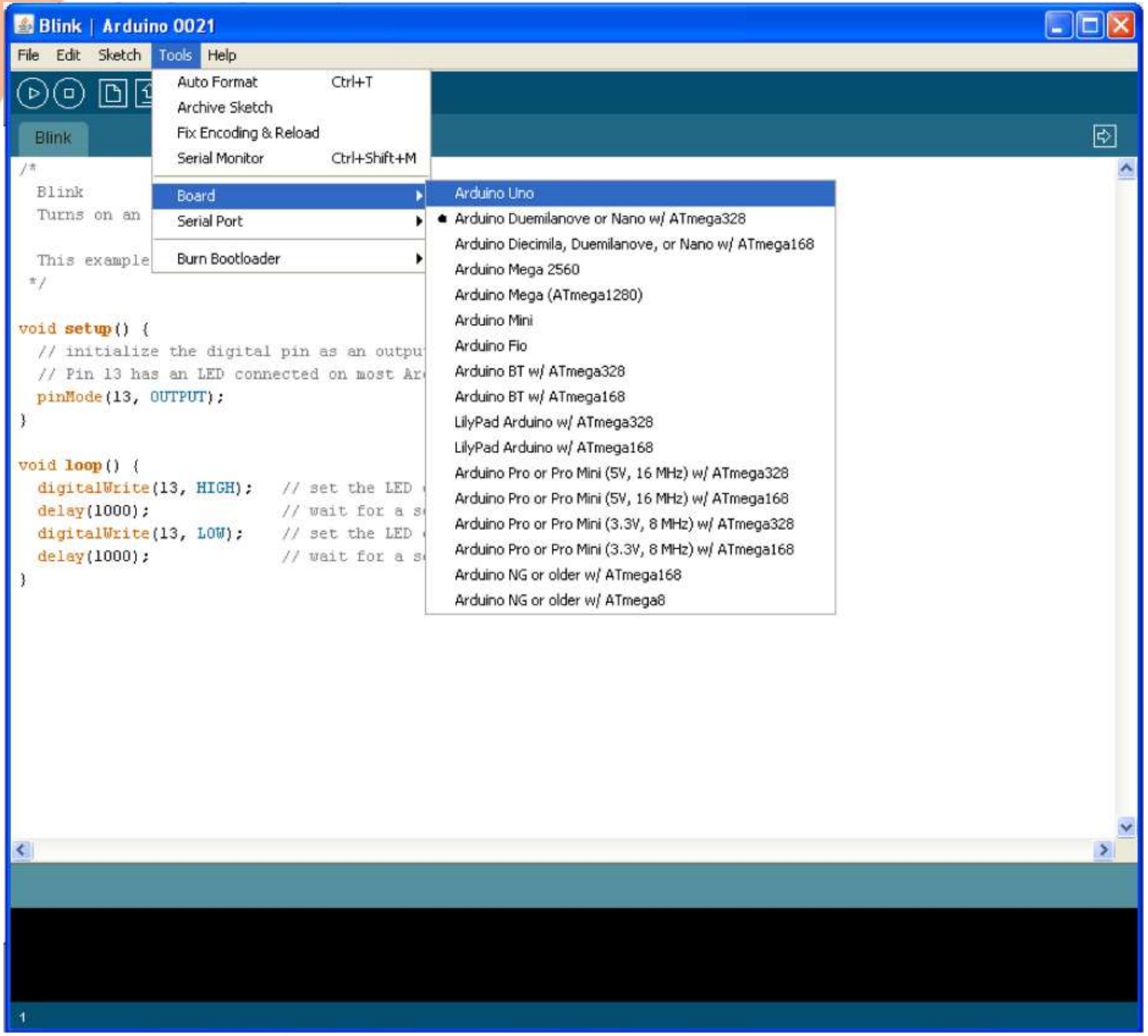


Arduino's IDE



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

  This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  delay(1000);              // wait for a second
}
```

Board



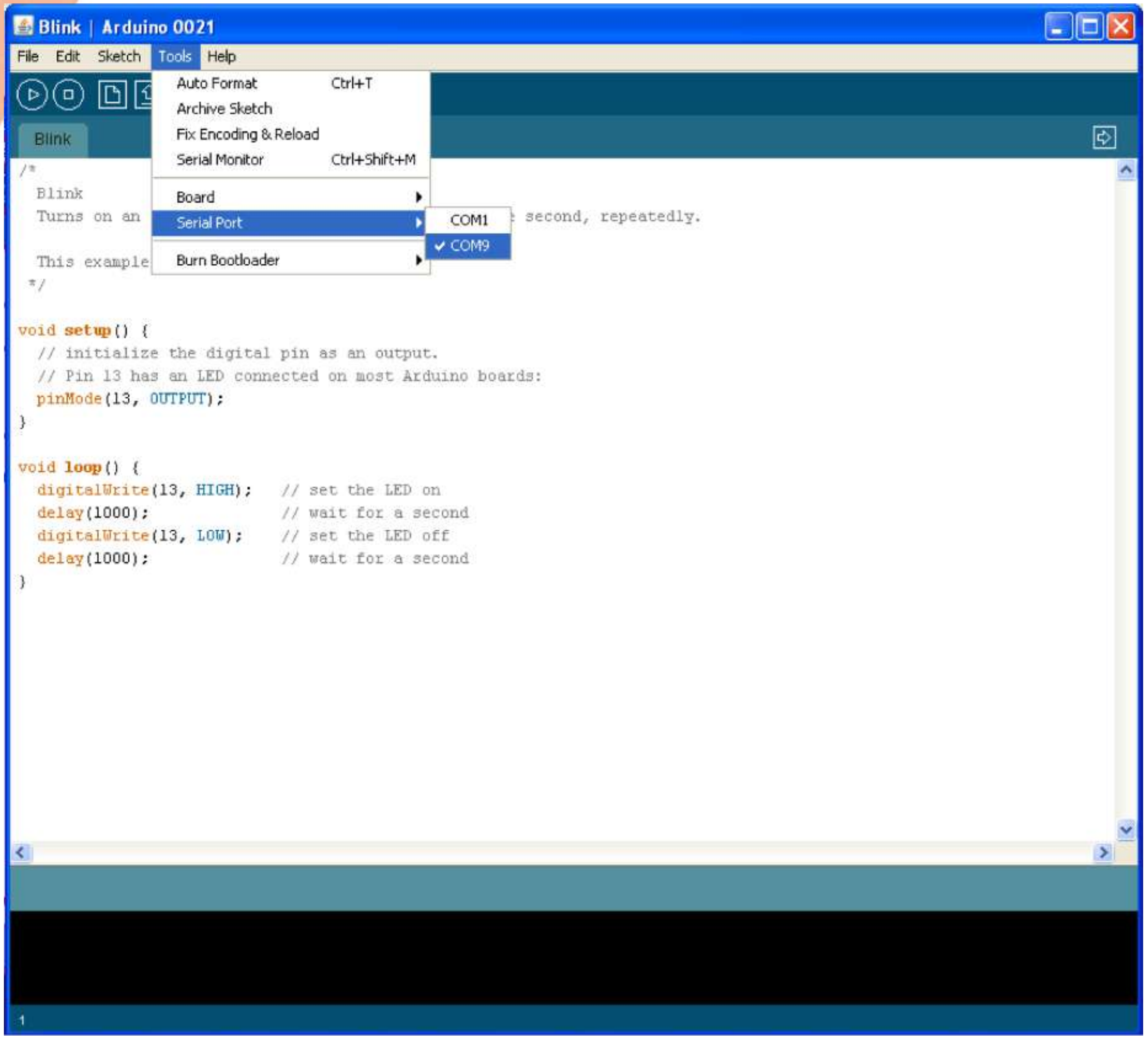
Blink | Arduino 0021

File Edit Sketch Tools Help

```
/*
  Blink
  Turns on an
  
  This example
 */

void setup() {
  // initialize the digital pin as an outpu
  // Pin 13 has an LED connected on most Ar
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED
  delay(1000);              // wait for a s
  digitalWrite(13, LOW);    // set the LED
  delay(1000);              // wait for a s
}
```

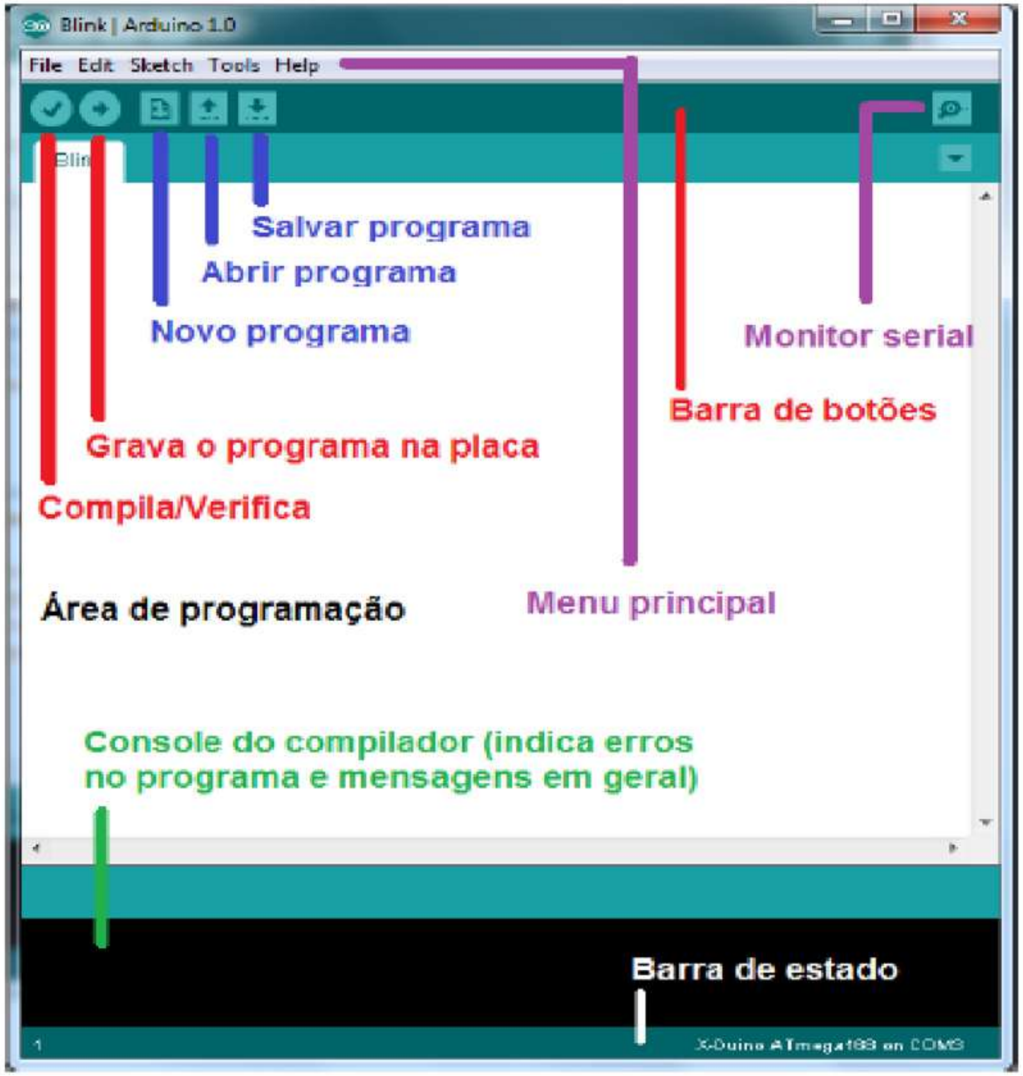
Blink
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board
Serial Port
Burn Bootloader
Arduino Uno
● Arduino Duemilanove or Nano w/ ATmega328
Arduino Diecimila, Duemilanove, or Nano w/ ATmega168
Arduino Mega 2560
Arduino Mega (ATmega1280)
Arduino Mini
Arduino Fio
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
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega168
Arduino NG or older w/ ATmega168
Arduino NG or older w/ ATmega8
1

Serial Port / COM

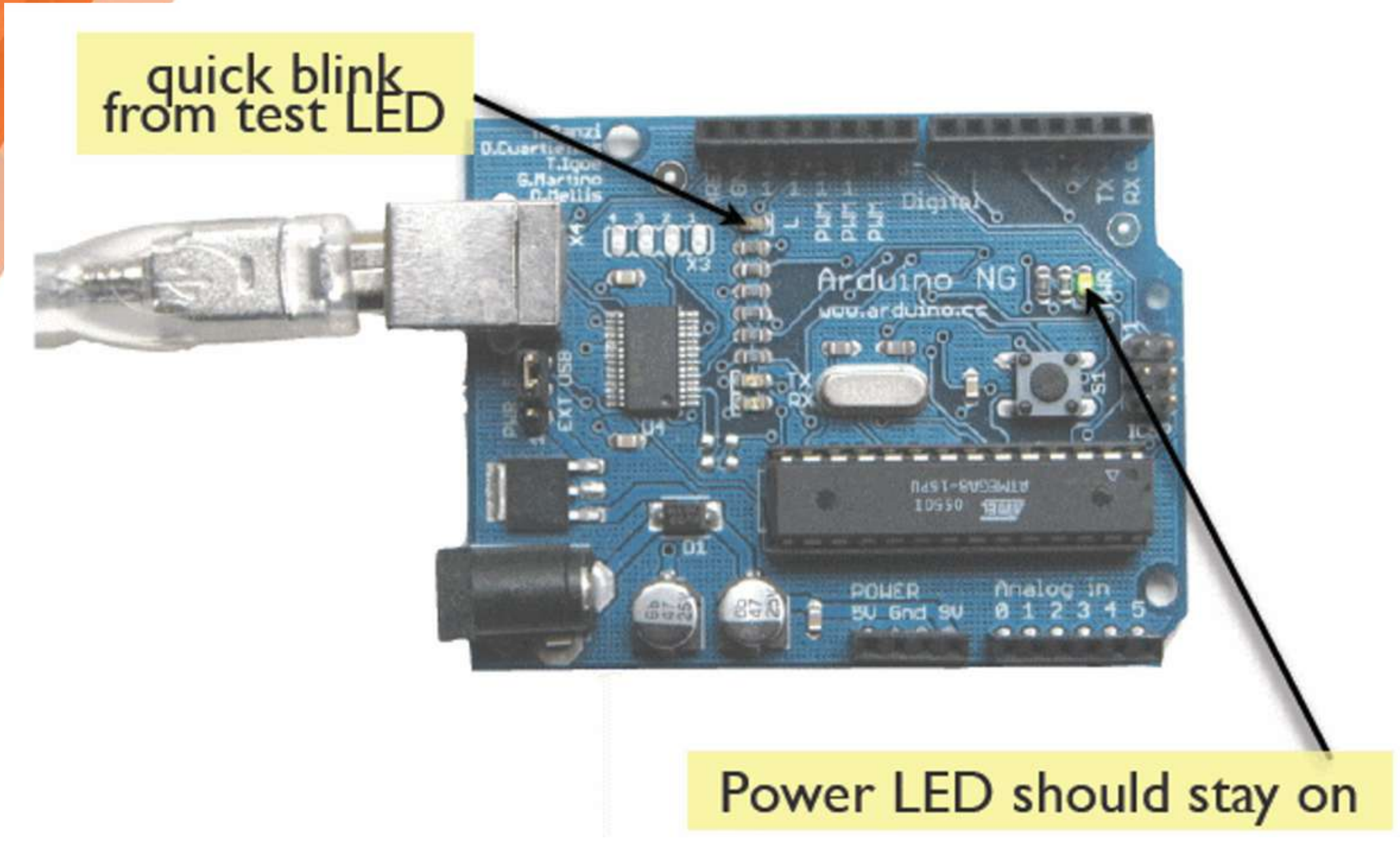


```
/*
  Blink
  Turns on an                                    second, repeatedly.

  This example
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  delay(1000);              // wait for a second
}
```

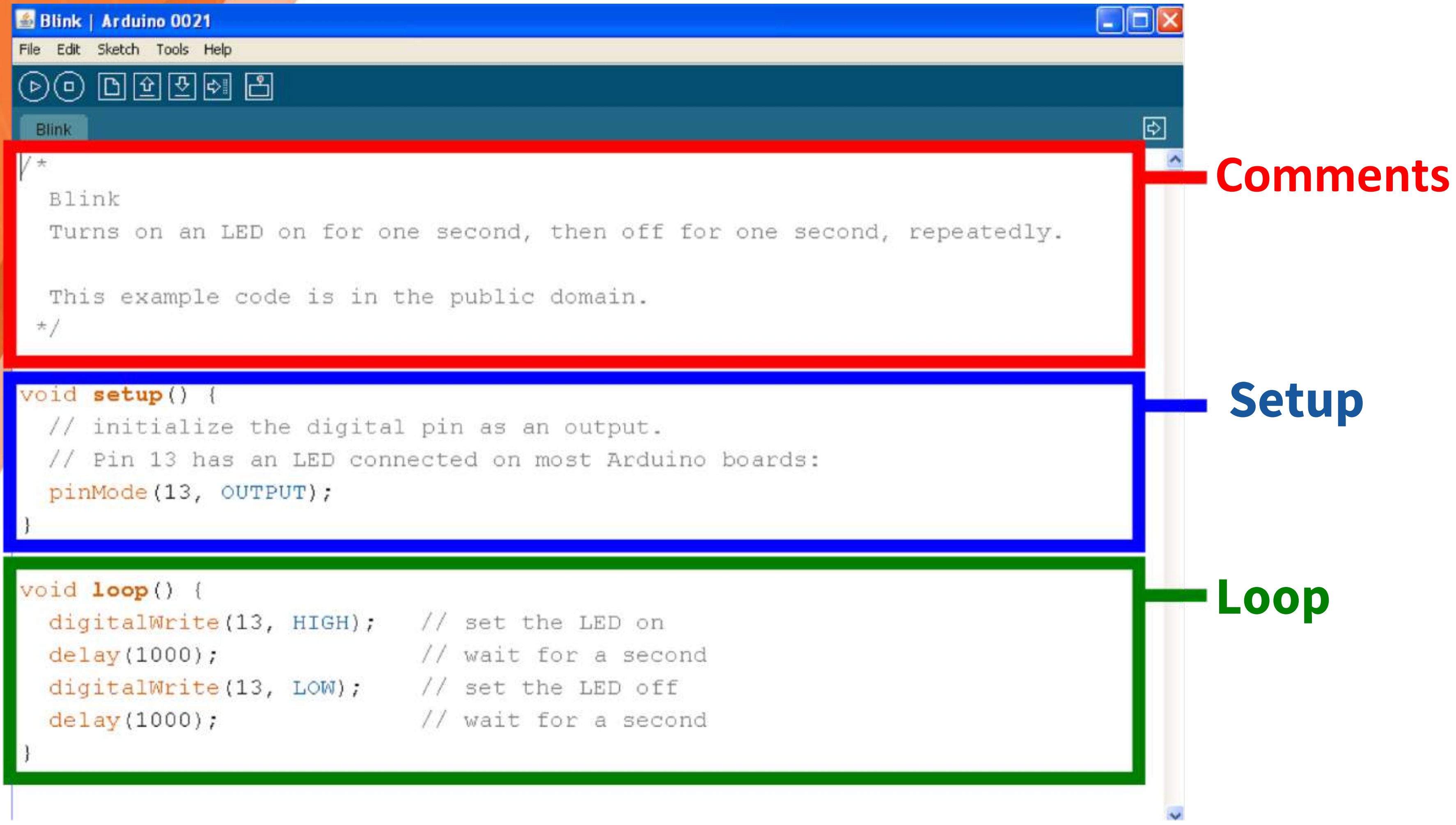
Working area



Programming



Sketch



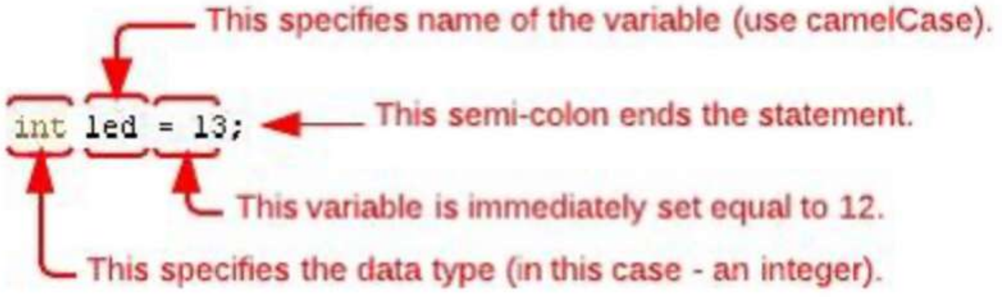
```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  delay(1000);              // wait for a second
}
```

Constants

- In Arduino there are some previously defined constants and are considered reserved words:
  - true – indicates true logical value
  - false – indicates false logical value
  - HIGH – indicates that a port is activated, that is, it is at 5V
  - LOW – indicates that a port is disabled, that is, it is at 0V
  - INPUT – indicates that a port will be input data
  - OUTPUT – indicates that a port will be output data.

Variables

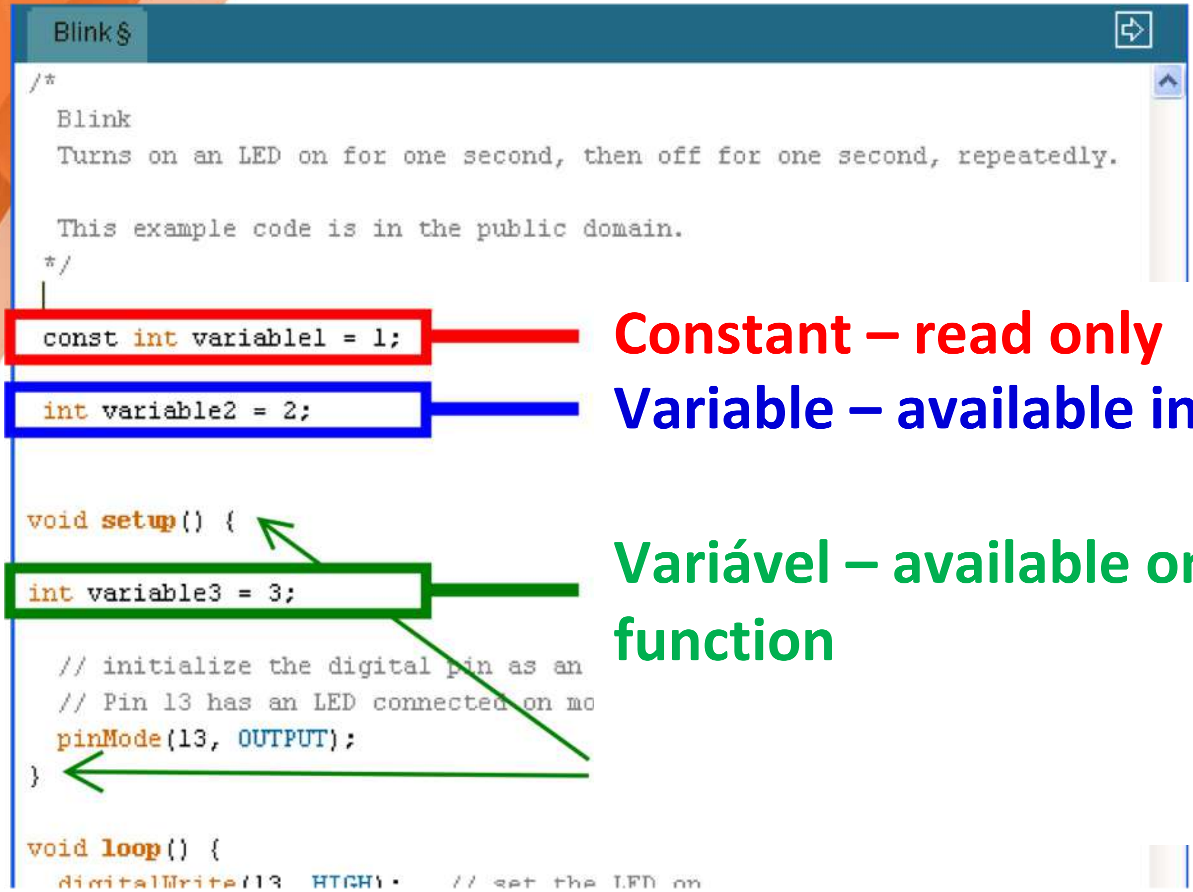
- Basic types of variables:
  - Boolean: **boolean variableName;**
  - Integer: **int variableName;**
  - Float: **float variableName;**
  - Character: **char variableName;**
  - String: **string variableName;**
  - Arrays: **int variableName[number];**



```
int led = 13;
```

This specifies name of the variable (use camelCase).
This semi-colon ends the statement.
This variable is immediately set equal to 12.
This specifies the data type (in this case - an integer).

Scope of variables



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

  This example code is in the public domain.
 */

 const int variable1 = 1;
 int variable2 = 2;


void setup() {
int variable3 = 3;

  // initialize the digital pin as an
  // Pin 13 has an LED connected on mo
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED on
```

Constant – read only

Variable – available in all program

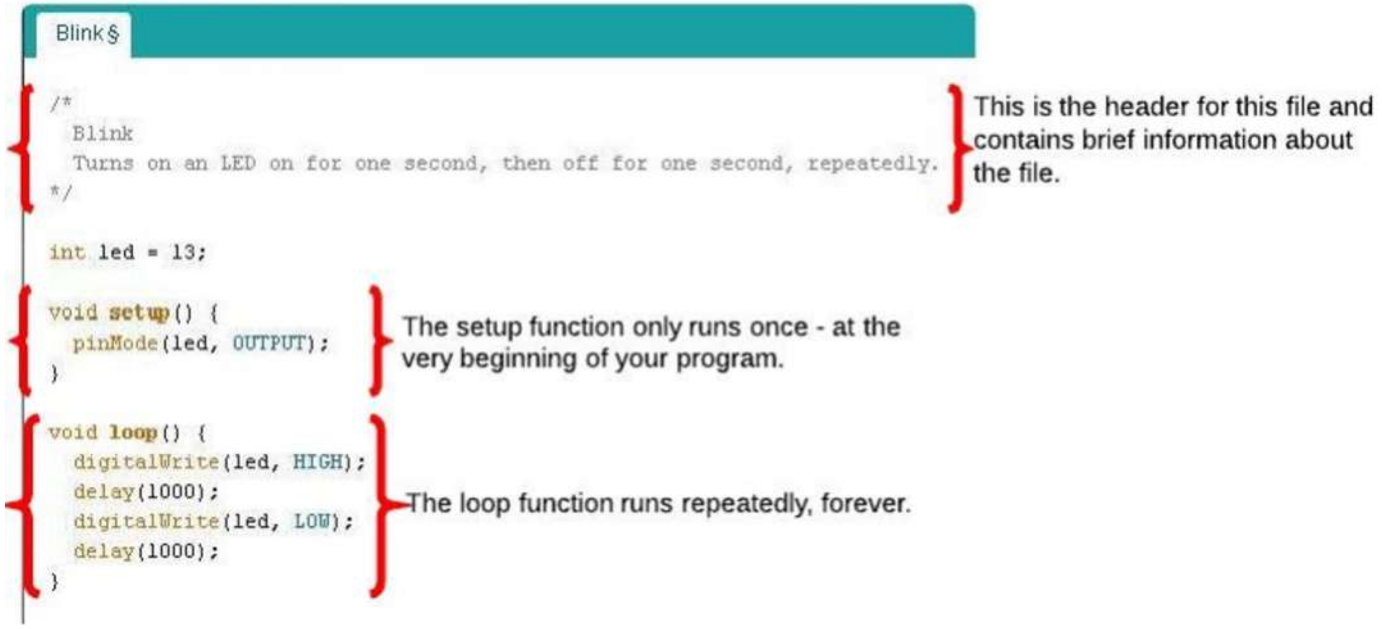
Variável – available only inside this function

Comments

- Can be placed anywhere
- They are created with // or /* and */
- Do not affect the code
- They may not be accurate, but they are always useful.

Main functions

- void setup() - is executed once and is normally used for setting the pins (as input or output), starting to use serial communication, among others.
- void loop() - makes a “loop”, that is, all commands are successively repeated.



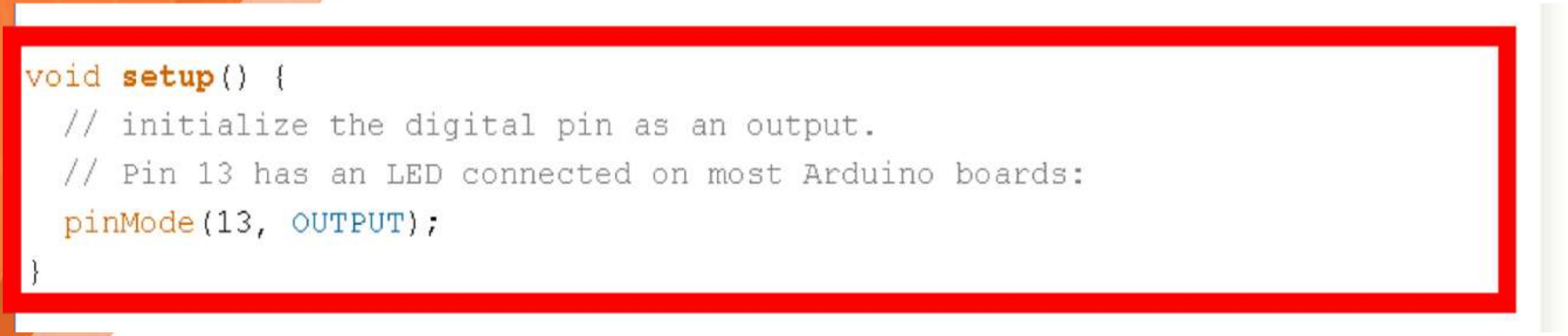
```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.
*/

int led = 13;

void setup() {
  pinMode(led, OUTPUT);
}

void loop() {
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
}
```

This is the header for this file and contains brief information about the file.

The setup function only runs once - at the very beginning of your program.

The loop function runs repeatedly, forever.

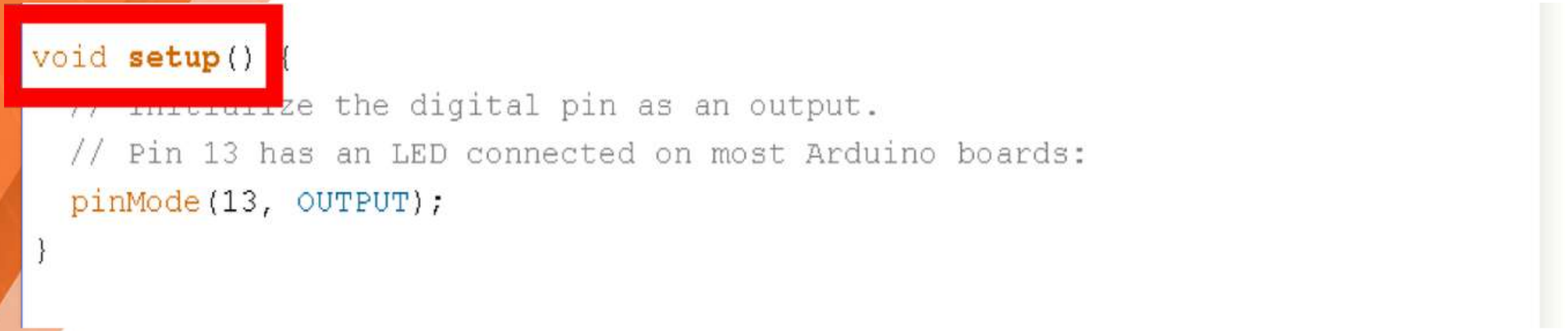
Setup



```
void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}
```

- **void setup() {}**
- The SETUP function comes BEFORE the LOOP function and is mandatory in all sketches

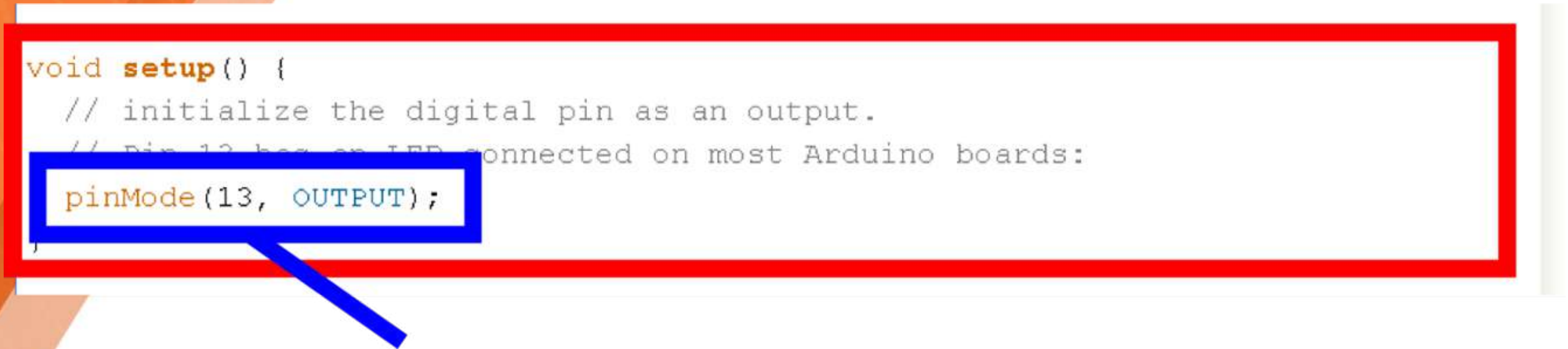
Setup



```
void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}
```

- **void setup() {}**
- The SETUP header never changes.
- Everything that happens inside SETUP must be inside the braces

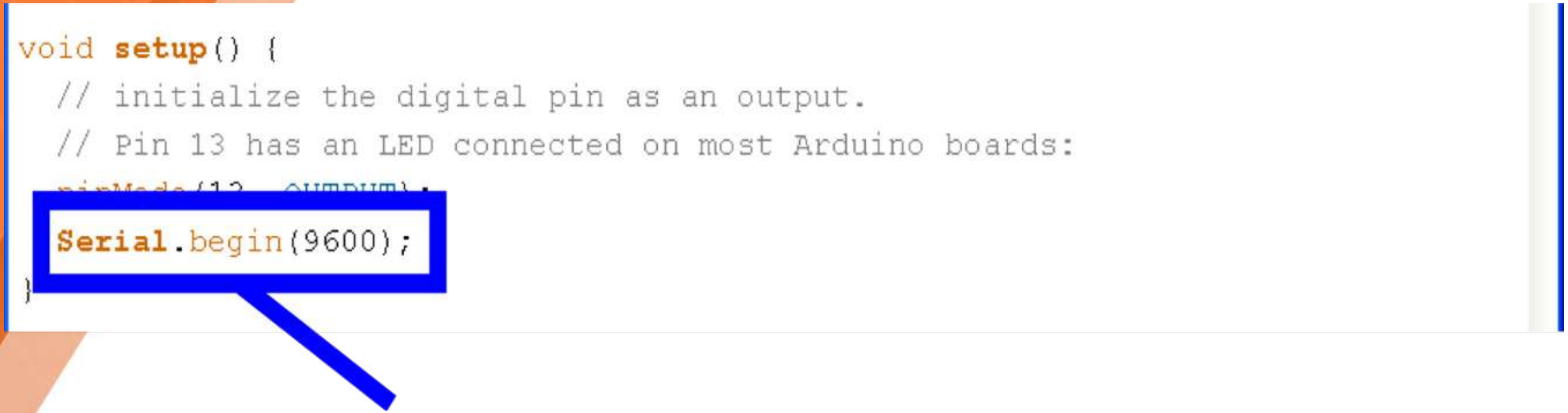
Setup



```
void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}
```

- **void setup() {*pinMode (13, OUTPUT);* }**
- The outputs are declared inside the setup, and it is done through the pinMode function.

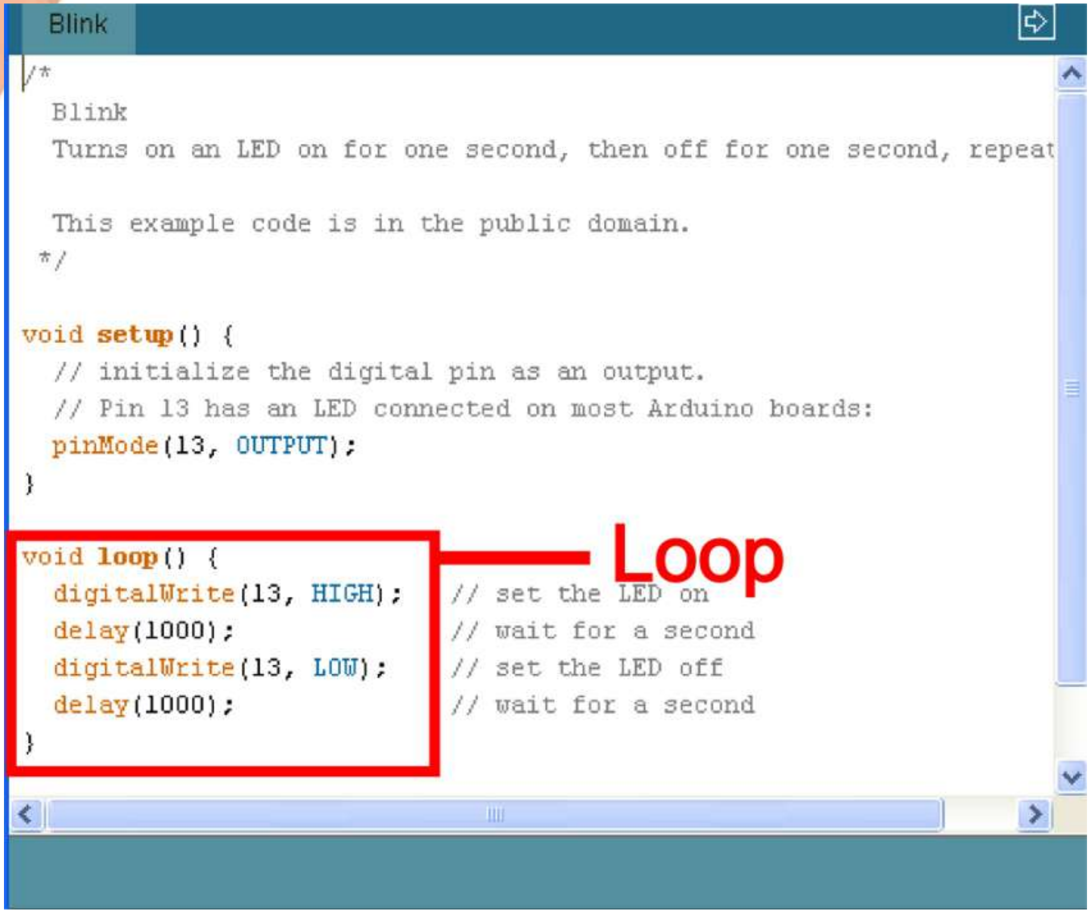
Setup



```
void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
  Serial.begin(9600);
}
```

- **void setup() {*Serial.begin(value);* }**
- The outputs are declared inside the setup, and it is done through the pinMode function.

Loop

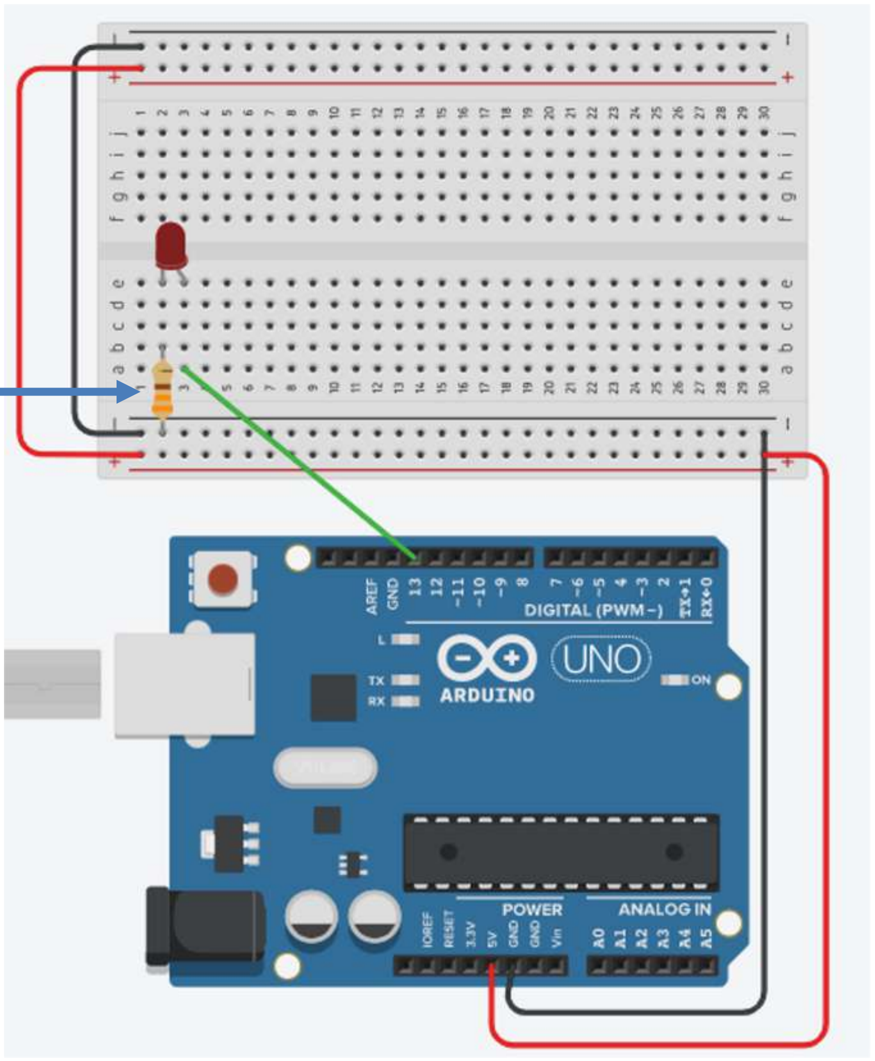


```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeat

  This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  delay(1000);              // wait for a second
}
```

- **void loop() {}**

Blink led

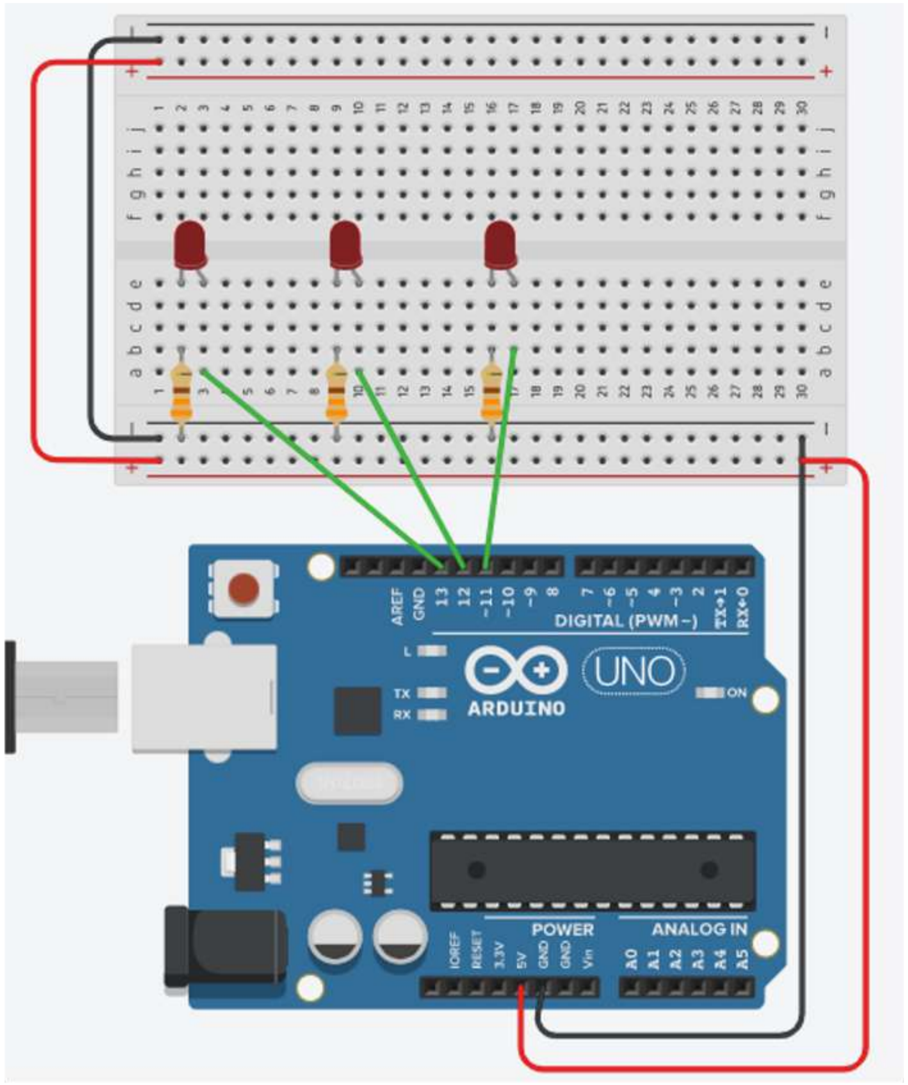
330Ω



Blink led

```
int led = 13;
void setup() {
        pinMode(led, OUTPUT);}
void loop() {
        digitalWrite(led, HIGH);
        delay(1000);
        digitalWrite(led, LOW);
        delay(1000);
}
```

Blink 3 leds

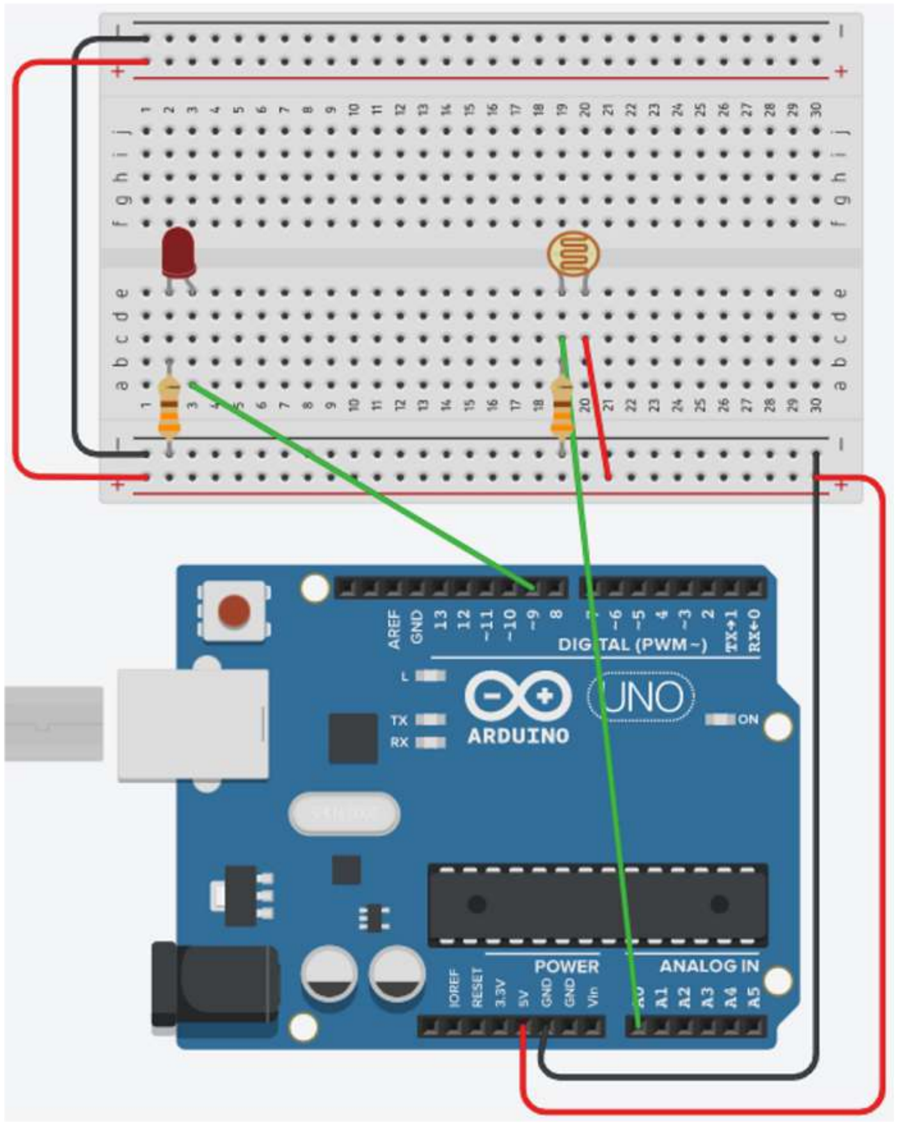


DIGITAL (PWM ~)
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
ARDUINO
UNO
ON
POWER
ANALOG IN
IOREF
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5

Blink 3 leds

```
int led1 = 13, led2 = 12, led3 = 11;
void setup() {
		pinMode(led1, OUTPUT);
		pinMode(led2, OUTPUT);
		pinMode(led3, OUTPUT);
}
void loop() {
		digitalWrite(led1, HIGH);
		delay(200);
		digitalWrite(led1, LOW);
		delay(200);
		digitalWrite(led2, HIGH);
		delay(200);
		digitalWrite(led2, LOW);
		delay(200);
		digitalWrite(led3, HIGH);
		delay(200);
		digitalWrite(led3, LOW);
		delay(200);
}
```

LDR with a led



LDR with a led

```
int sensorValue = 0;
void setup() {
  pinMode(A0, INPUT);
  pinMode(9, OUTPUT);
  Serial.begin(9600);
}
void loop() {
sensorValue = analogRead(A0);
Serial.println(sensorValue);
analogWrite(9, map(sensorValue, 0, 1023, 0, 255));
//port, fromLow, fromHigh, toLow, toHigh
delay(100); // Wait for 100 millisecond(s)
}
```

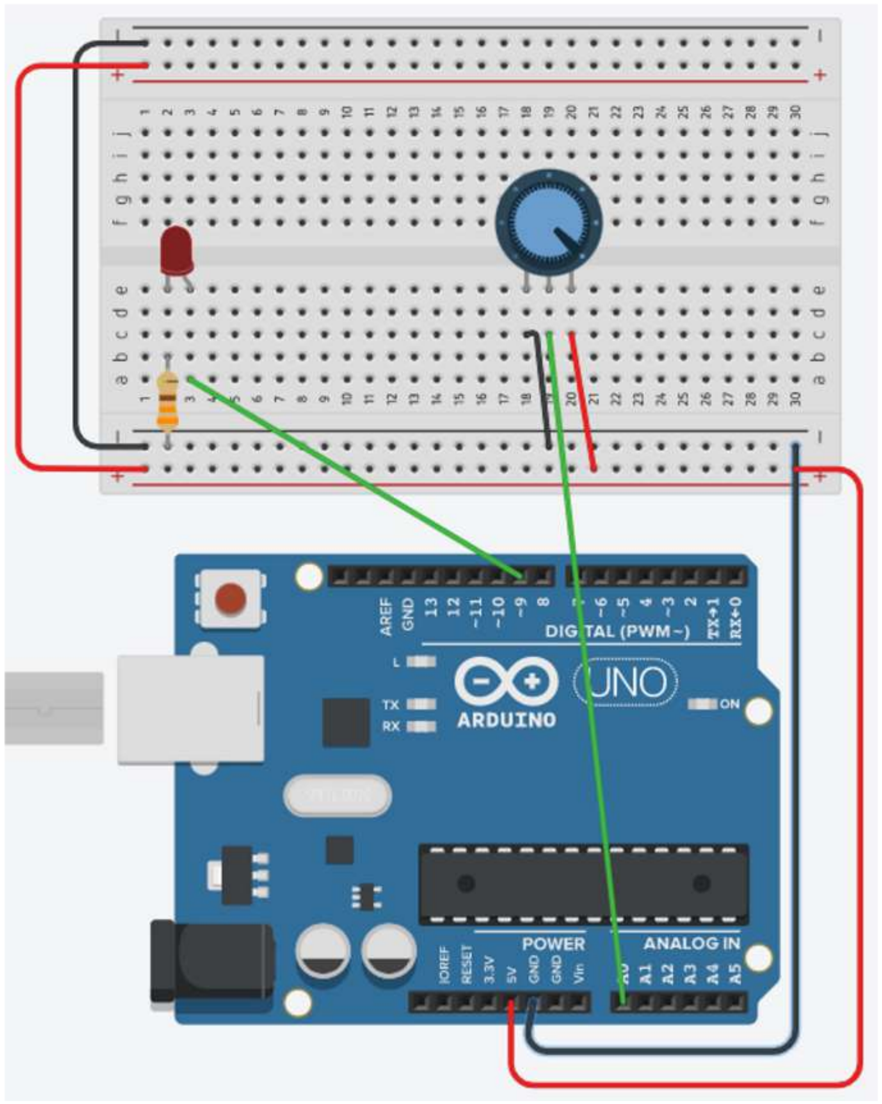
Add a library

- There are several libraries available on the internet that you can download and use.
- These libraries have to be added to the IDE so that it recognizes the commands you are using.
- After downloading, the .zip file is unzipped. In the folder where the IDE for Arduino is installed, look for the folder libraries. Inside this directory we copy the folder that was extracted before.

Add a library

- Finally, it is verified whether the library was actually detected by the IDE. Go to Files > Examples and check if the library that was added is there.
- Now just “call” it in the code, which will look like: #include <library.h>.
- It is worth noting that on the page where the library was downloaded, there are instructions on how to use it.

Turn on a led with a potentiometer

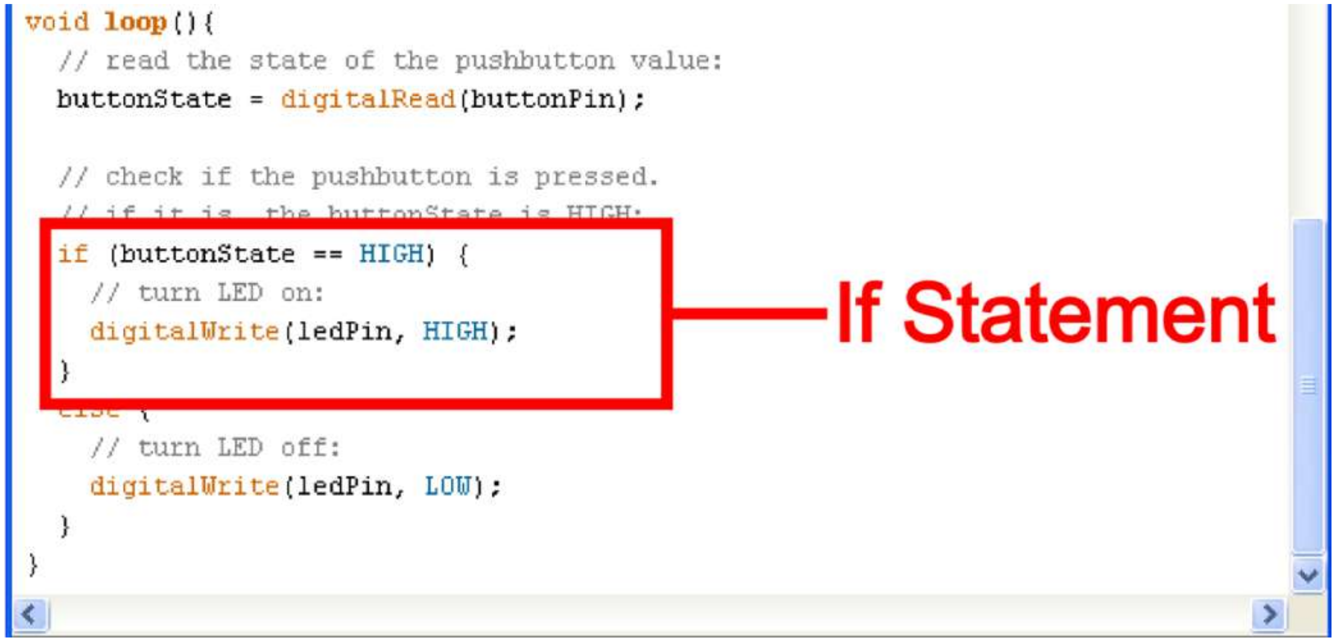


Turn on a led with a potentiometer

```
#include <SoftwareSerial.h> //library
#define LED 9 //const
#define KNOB 0 //const
void setup() {
  pinMode(LED, OUTPUT);
  Serial.begin(9600);
}
void loop() {
  int val = analogRead(KNOB);
  int ledPower = map(val, 1, 1024, 1, 255);
  analogWrite(LED, ledPower);
}
```

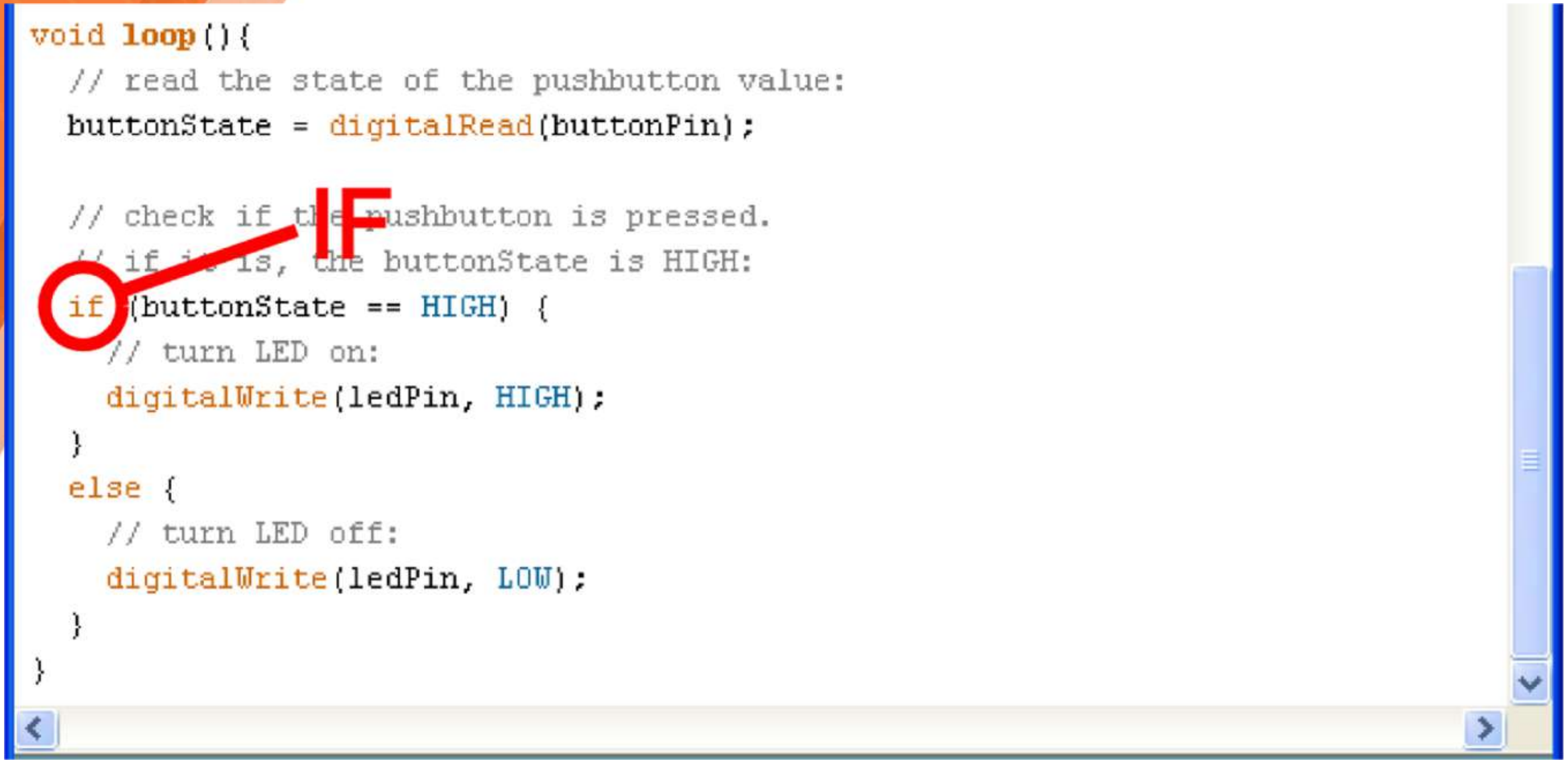
If condition

```
if (true) {
        code;
} else { //optional
        code;
}
```



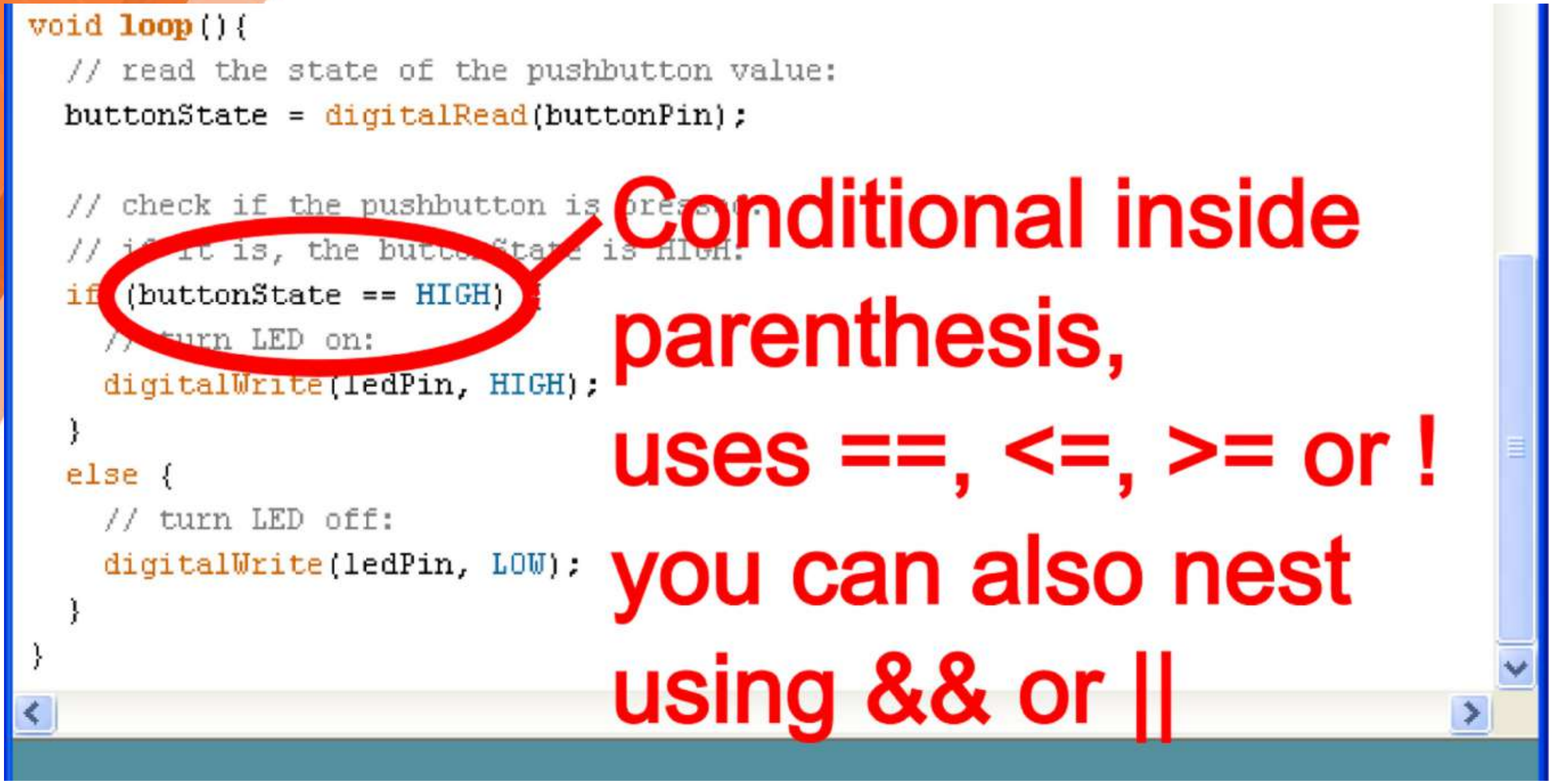
```
void loop(){
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // 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);
  }
}
```

If condition



```
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);
  }
}
```

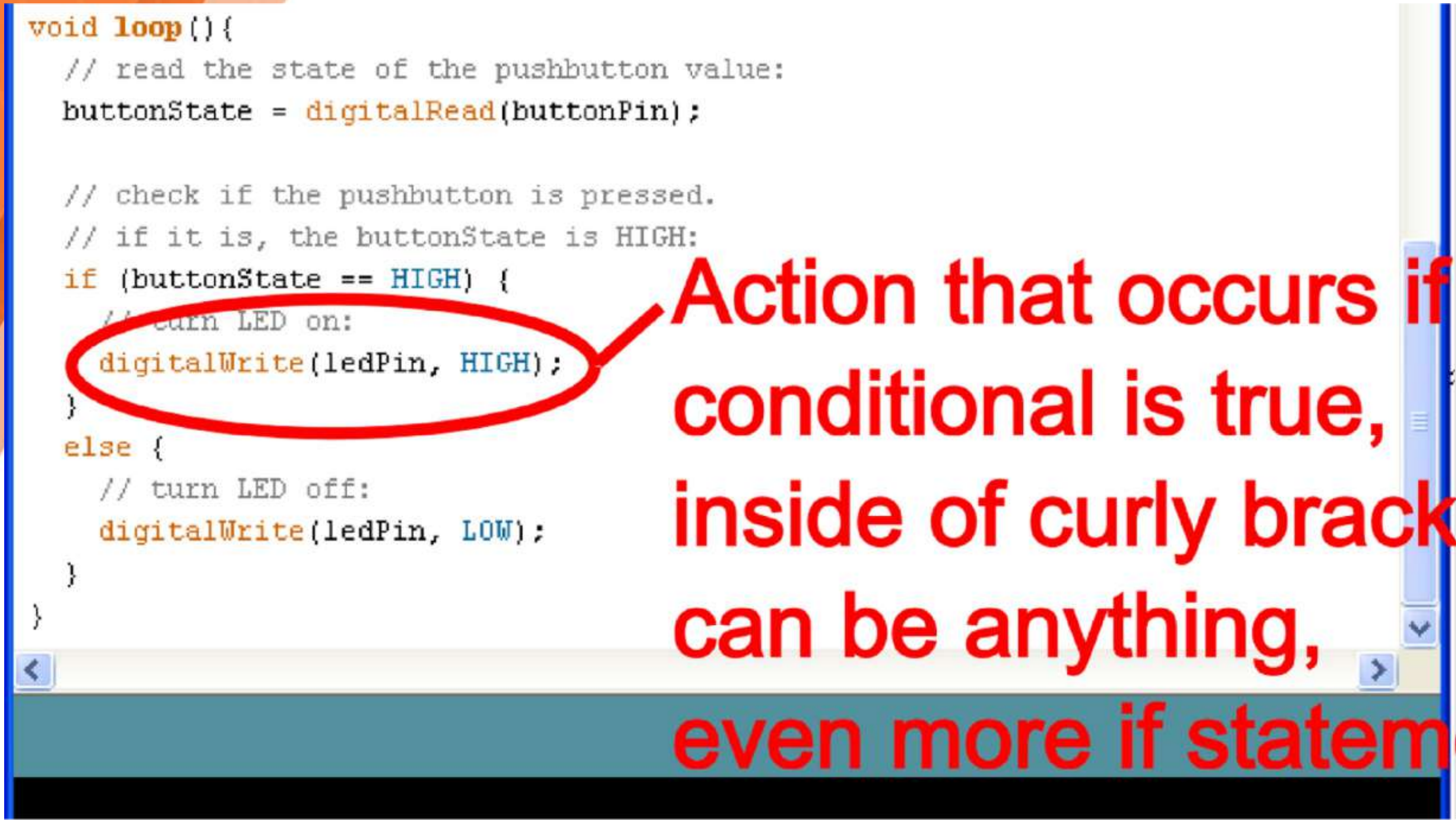
If condition



```
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);
  }
}
```

Conditional inside parenthesis, uses ==, <=, >= or ! you can also nest using && or ||

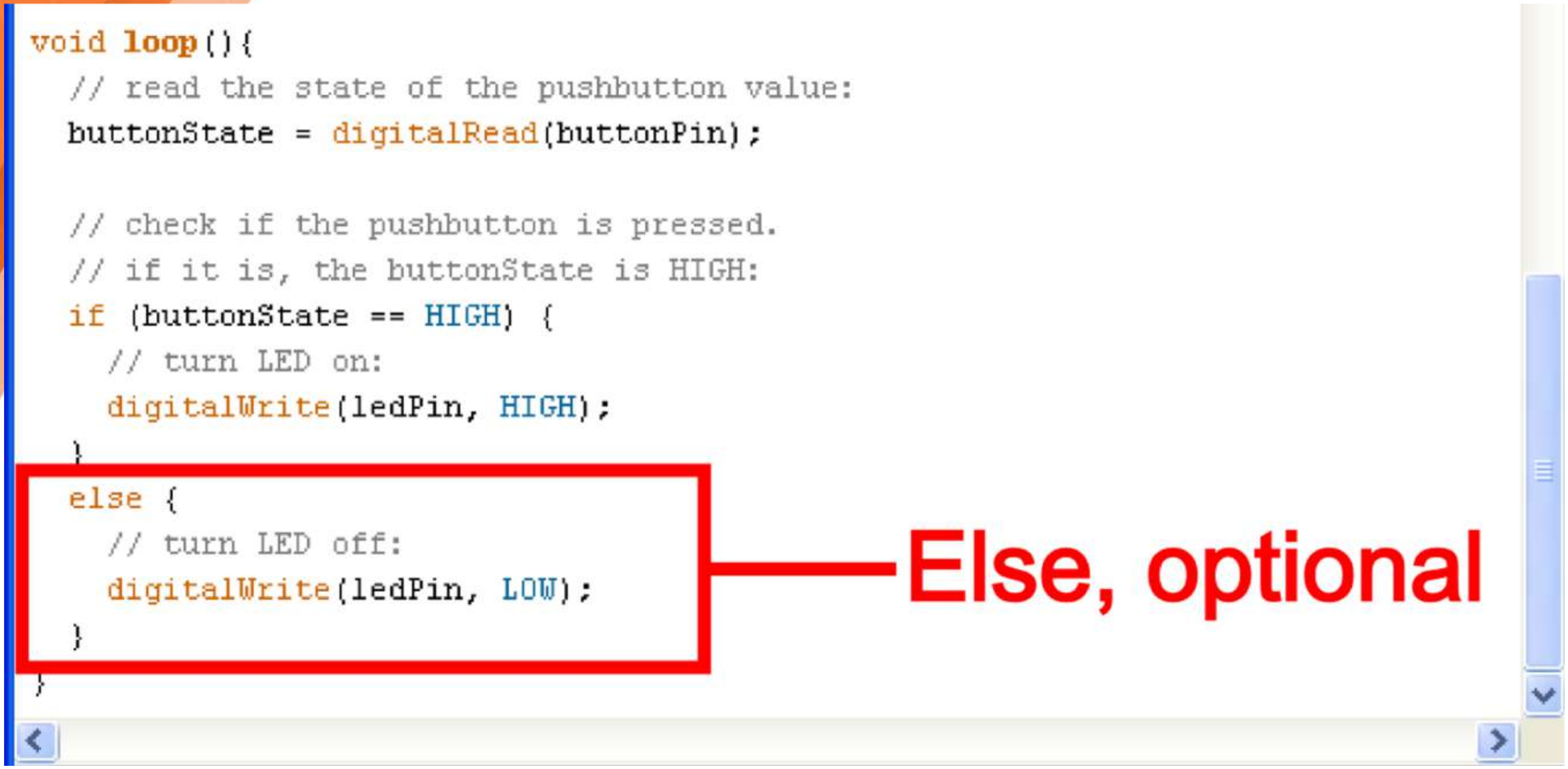
If condition



```
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);
  }
}
```

Action that occurs if conditional is true, inside of curly brackets, can be anything, even more if statements

If condition

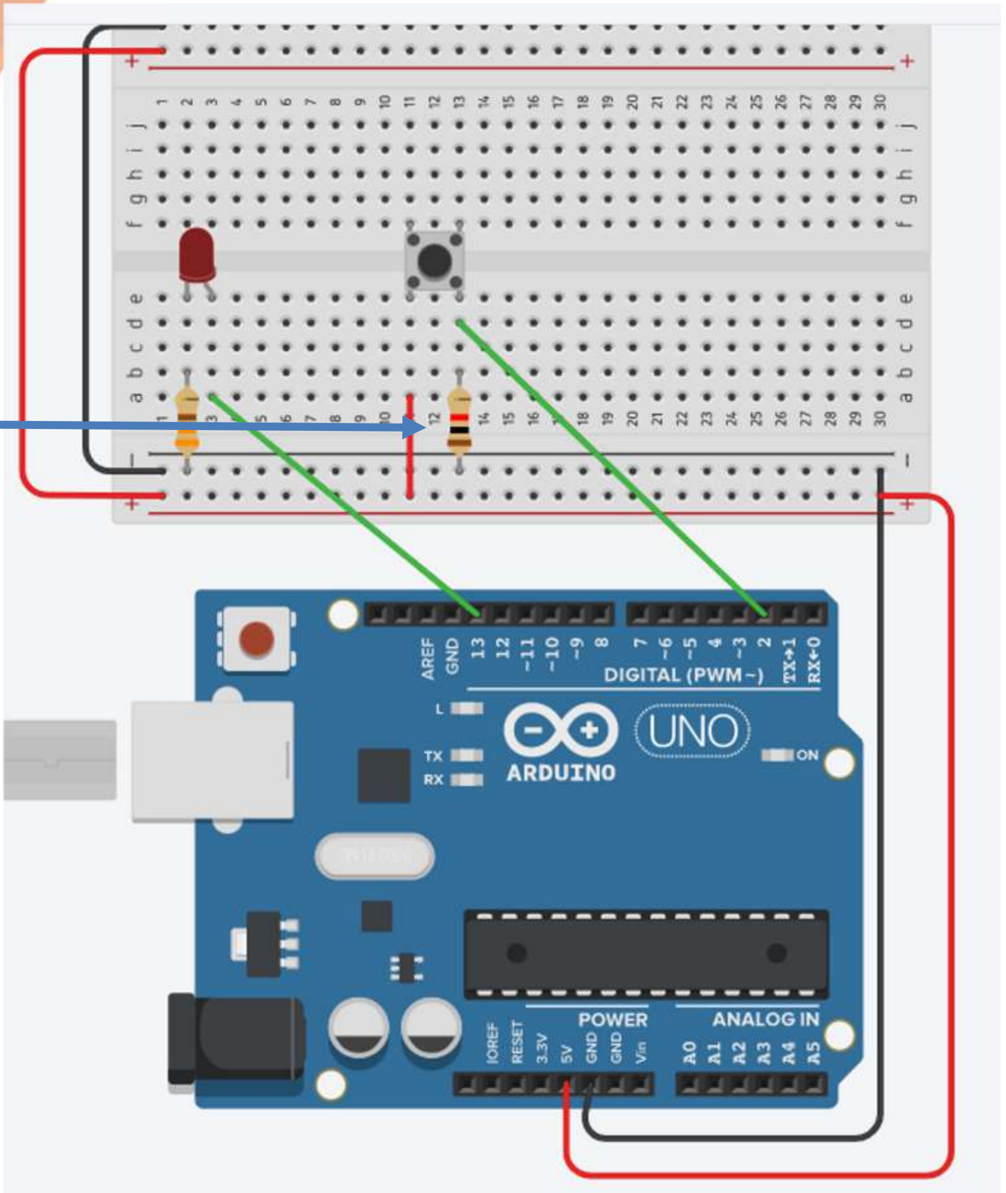


```
void loop(){
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // 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);
  }
}
```

Fade a led

```
int led = 13; //change to 9
int shine = 0;
int fade = 5;
void setup() {
        pinMode(led, OUTPUT);}
void loop() {
        analogWrite(led, shine);
        shine = shine + fade;
        if (shine <= 0 || shine >= 255) {
                fade = -fade;  }
        delay(30);
}
```

Turn on led with a button

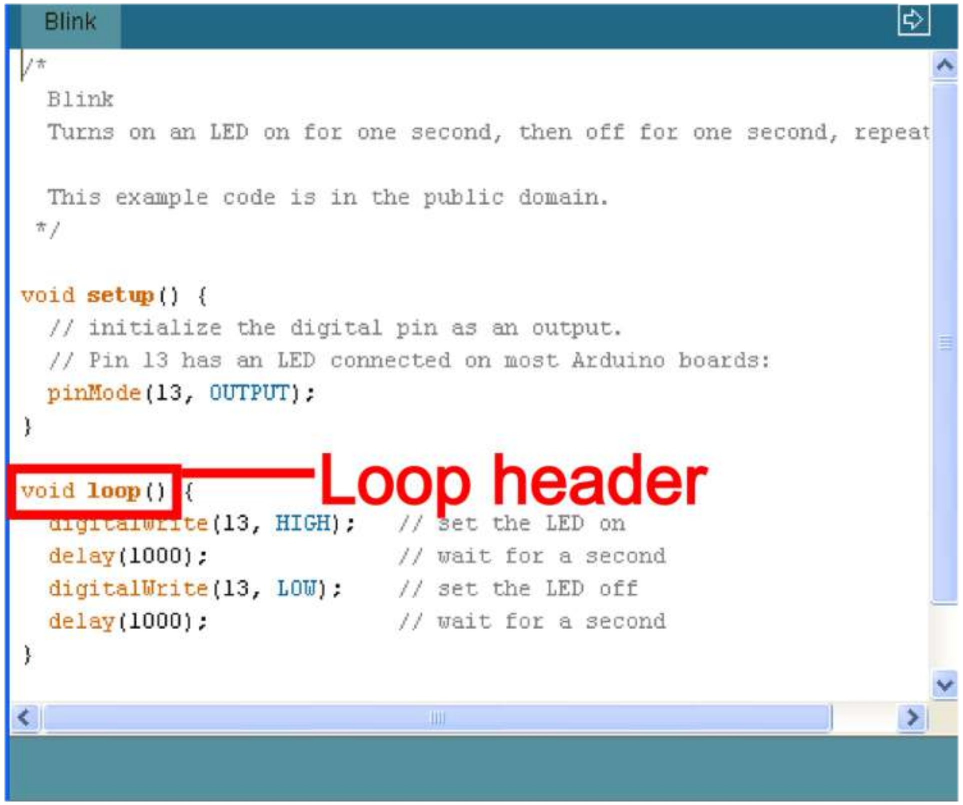


Turn on led with a button

```
const int buttonPin = 2;
int ledPin =  13;
int buttonState = 0;
void setup() {
        pinMode(ledPin, OUTPUT);
        pinMode(buttonPin, INPUT);}
void loop() {
        buttonState = digitalRead(buttonPin);
        if  (buttonState == HIGH) {
                digitalWrite(ledPin, HIGH);
        } else {
                digitalWrite(ledPin, LOW);  }
}
```

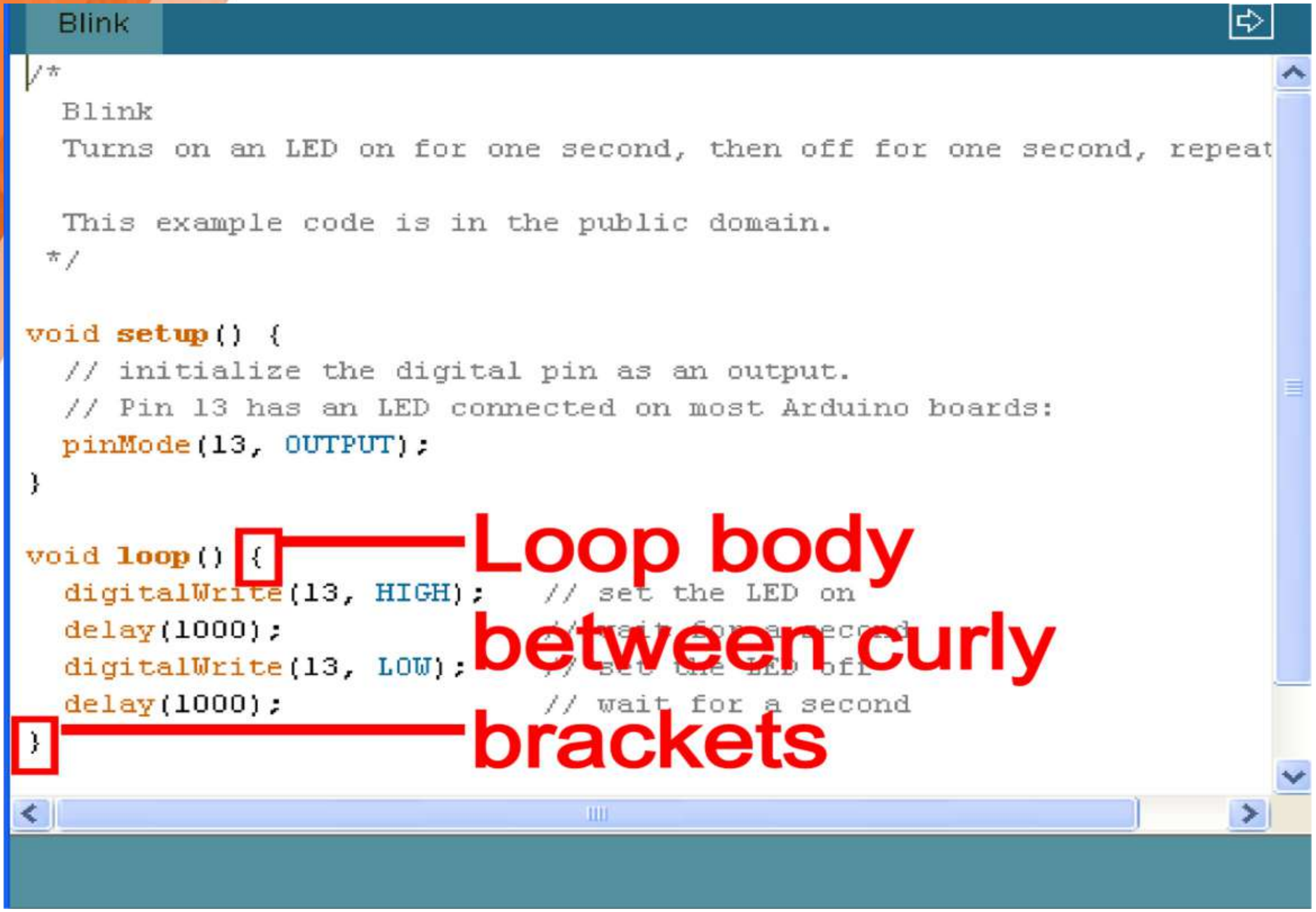
Loop

- The “loop” in the header is the function name. The setup and loop functions already have the declared name. The rest are created by the user.



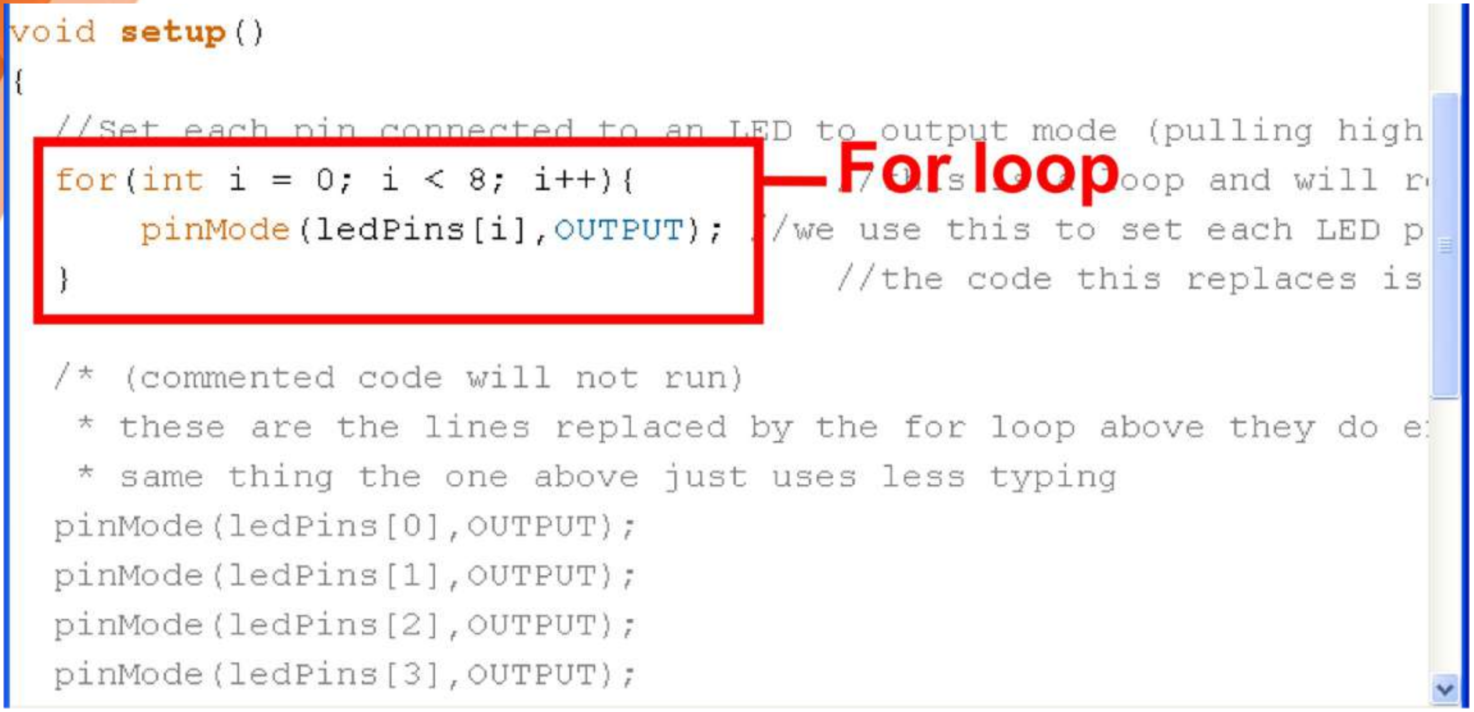
```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeat

  This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  delay(1000);              // wait for a second
}
```

Loop



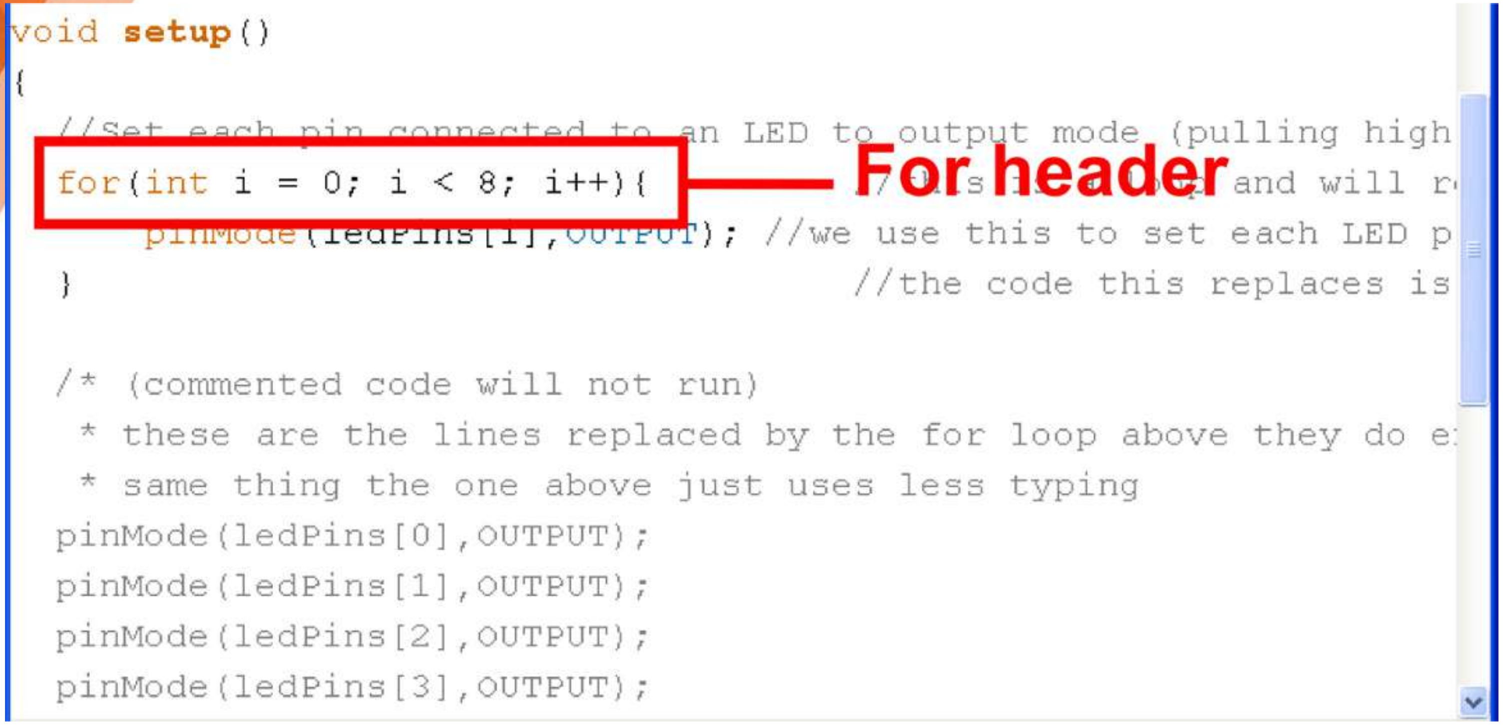
```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeat

  This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  delay(1000);              // wait for a second
}
```

Structure For



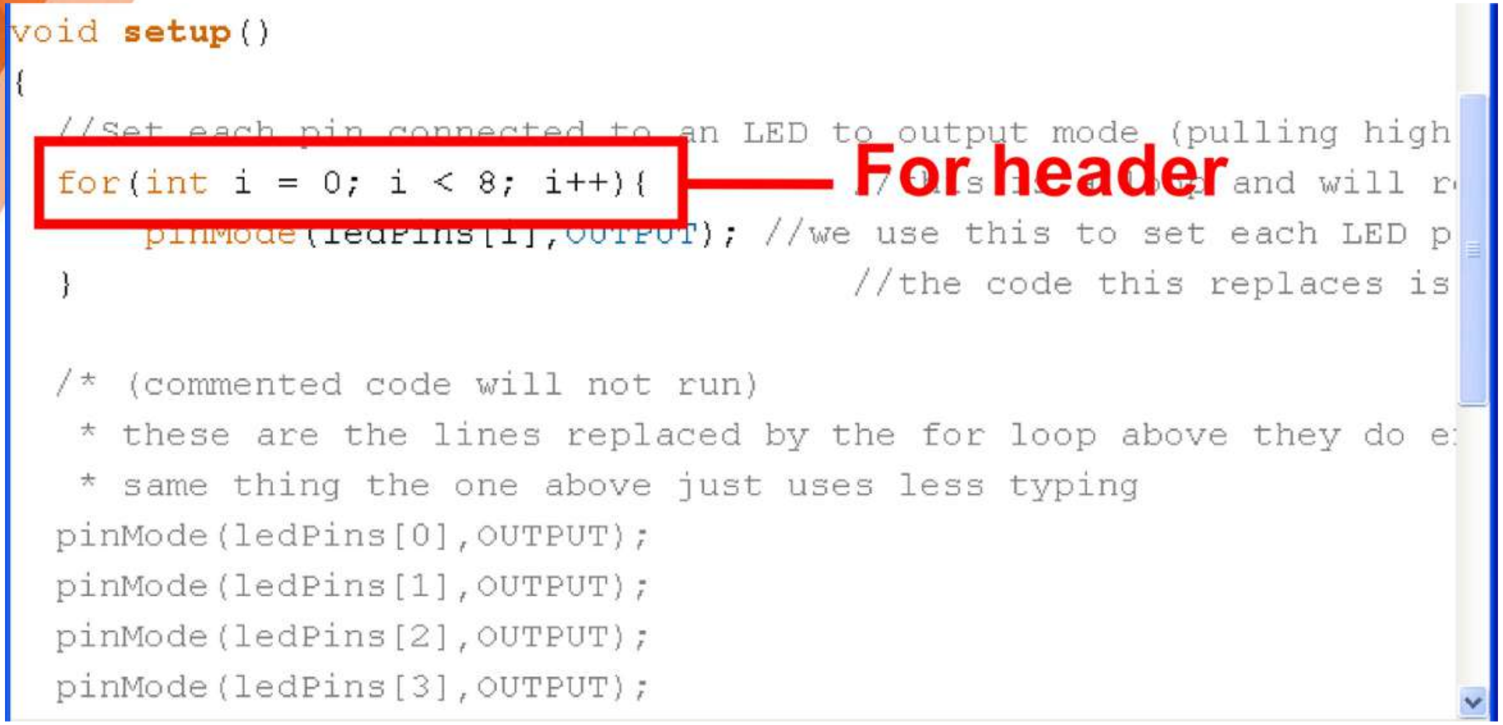
```
void setup()
{
  //Set each pin connected to an LED to output mode (pulling high
  for(int i = 0; i < 8; i++){           //this is a for loop and will r
      pinMode(ledPins[i],OUTPUT);  //we use this to set each LED p
  }                                   //the code this replaces is

  /* (commented code will not run)
   * these are the lines replaced by the for loop above they do e
   * same thing the one above just uses less typing
  pinMode(ledPins[0],OUTPUT);
  pinMode(ledPins[1],OUTPUT);
  pinMode(ledPins[2],OUTPUT);
  pinMode(ledPins[3],OUTPUT);
```

Structure For



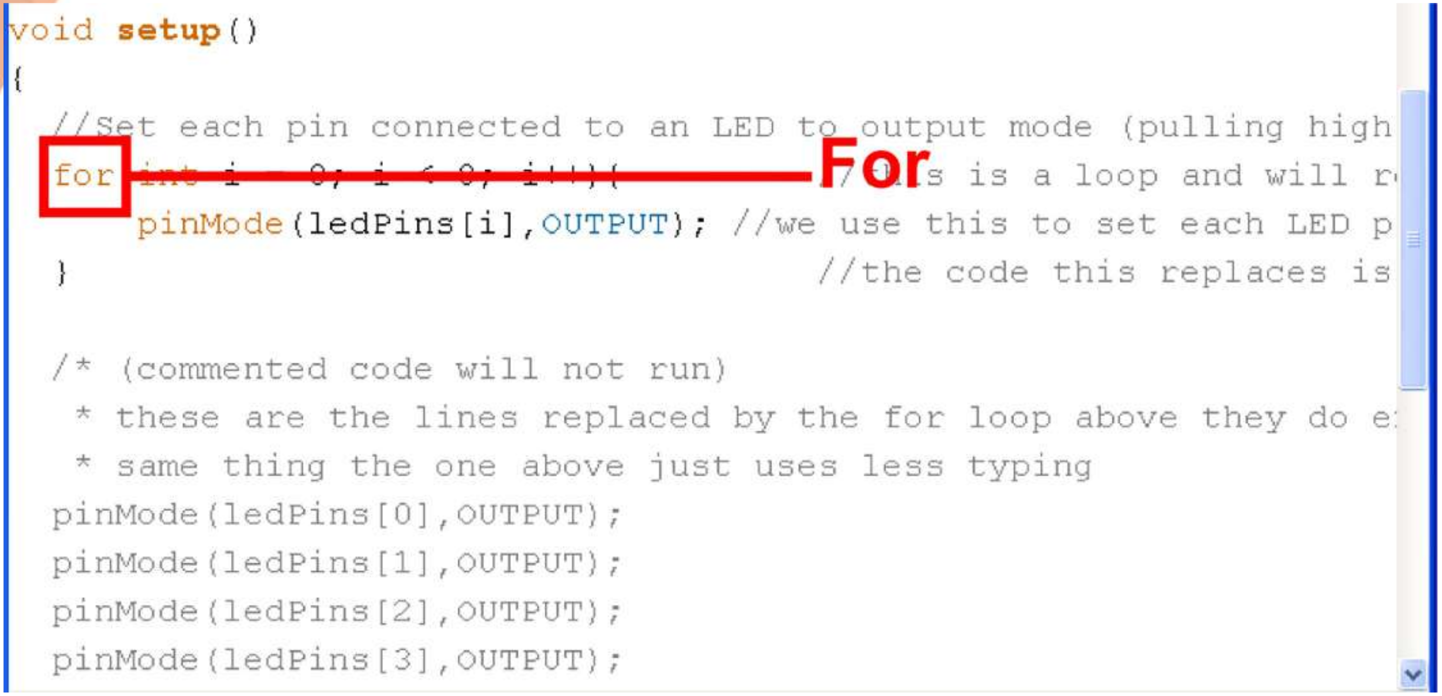
```
void setup()
{
  //Set each pin connected to an LED to output mode (pulling high
  for(int i = 0; i < 8; i++){         //this is a loop and will r
      pinMode(ledPins[i],OUTPUT); //we use this to set each LED p
  }                                  //the code this replaces is

  /* (commented code will not run)
   * these are the lines replaced by the for loop above they do e
   * same thing the one above just uses less typing
  pinMode(ledPins[0],OUTPUT);
  pinMode(ledPins[1],OUTPUT);
  pinMode(ledPins[2],OUTPUT);
  pinMode(ledPins[3],OUTPUT);
```

Structure For



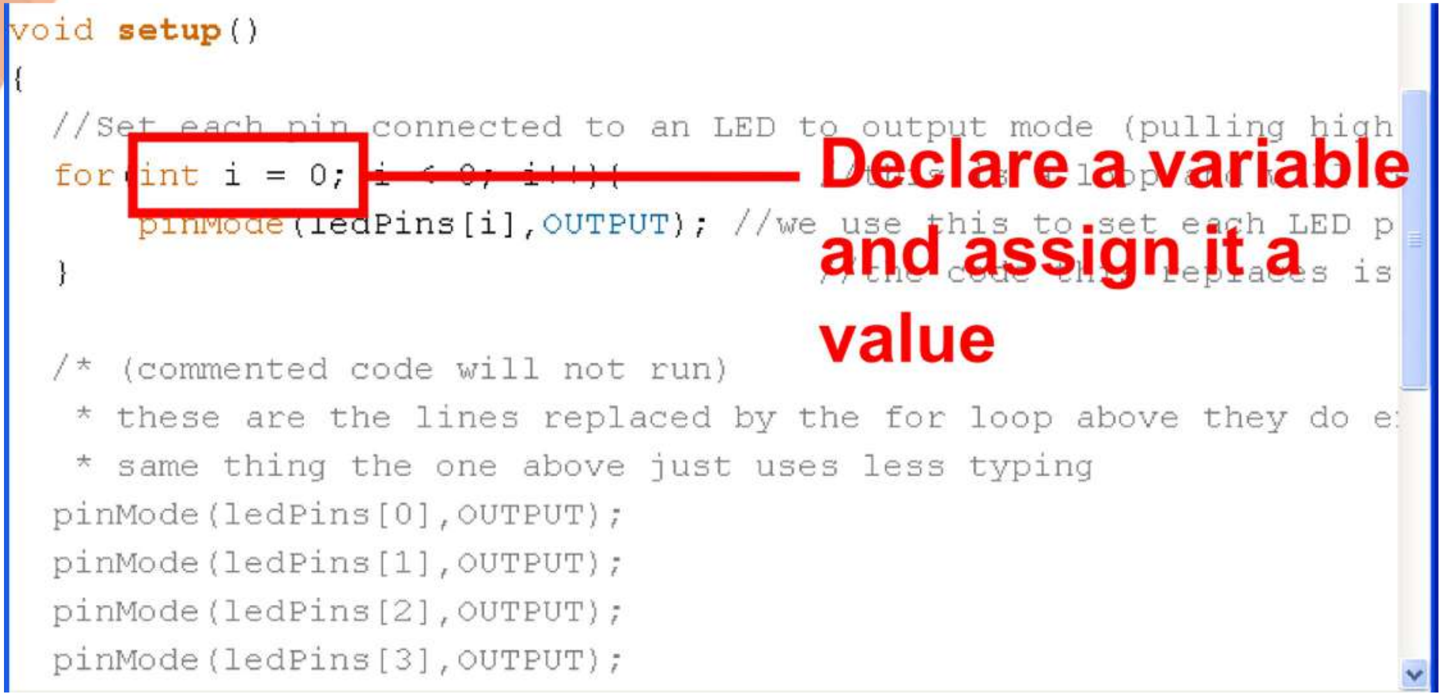
```
void setup()
{
  //Set each pin connected to an LED to output mode (pulling high
  for(int i = 0; i < 8; i++){         //this is a loop and will r
      pinMode(ledPins[i],OUTPUT); //we use this to set each LED p
  }                                  //the code this replaces is

  /* (commented code will not run)
   * these are the lines replaced by the for loop above they do e
   * same thing the one above just uses less typing
  pinMode(ledPins[0],OUTPUT);
  pinMode(ledPins[1],OUTPUT);
  pinMode(ledPins[2],OUTPUT);
  pinMode(ledPins[3],OUTPUT);
```

Structure For



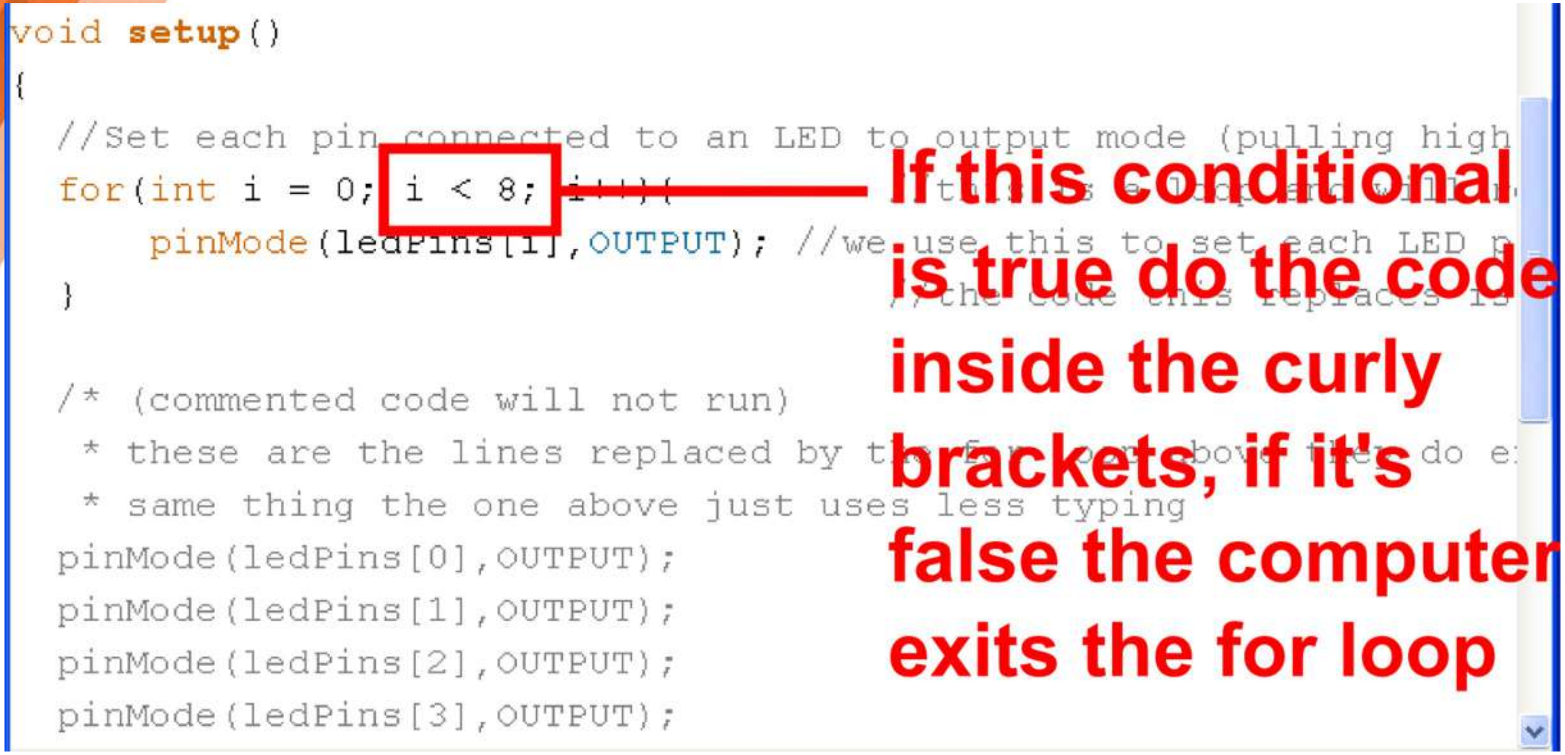
```
void setup()
{
  //Set each pin connected to an LED to output mode (pulling high
  for (int i = 0; i < 8; i++){              //this is a loop and will r
      pinMode(ledPins[i],OUTPUT); //we use this to set each LED p
  }                                   //the code this replaces is

  /* (commented code will not run)
   * these are the lines replaced by the for loop above they do e
   * same thing the one above just uses less typing
  pinMode(ledPins[0],OUTPUT);
  pinMode(ledPins[1],OUTPUT);
  pinMode(ledPins[2],OUTPUT);
  pinMode(ledPins[3],OUTPUT);
```

Structure For



```
void setup()
{
  //Set each pin connected to an LED to output mode (pulling high
  for(int i = 0; i < 8; i++){          //this is a loop and [illegible]
      pinMode(ledPins[i],OUTPUT); //we use this to set each LED p
  }                                    // the code this replaces is

  /* (commented code will not run)
   * these are the lines replaced by the for loop above they do e
   * same thing the one above just uses less typing
  pinMode(ledPins[0],OUTPUT);
  pinMode(ledPins[1],OUTPUT);
  pinMode(ledPins[2],OUTPUT);
  pinMode(ledPins[3],OUTPUT);
```

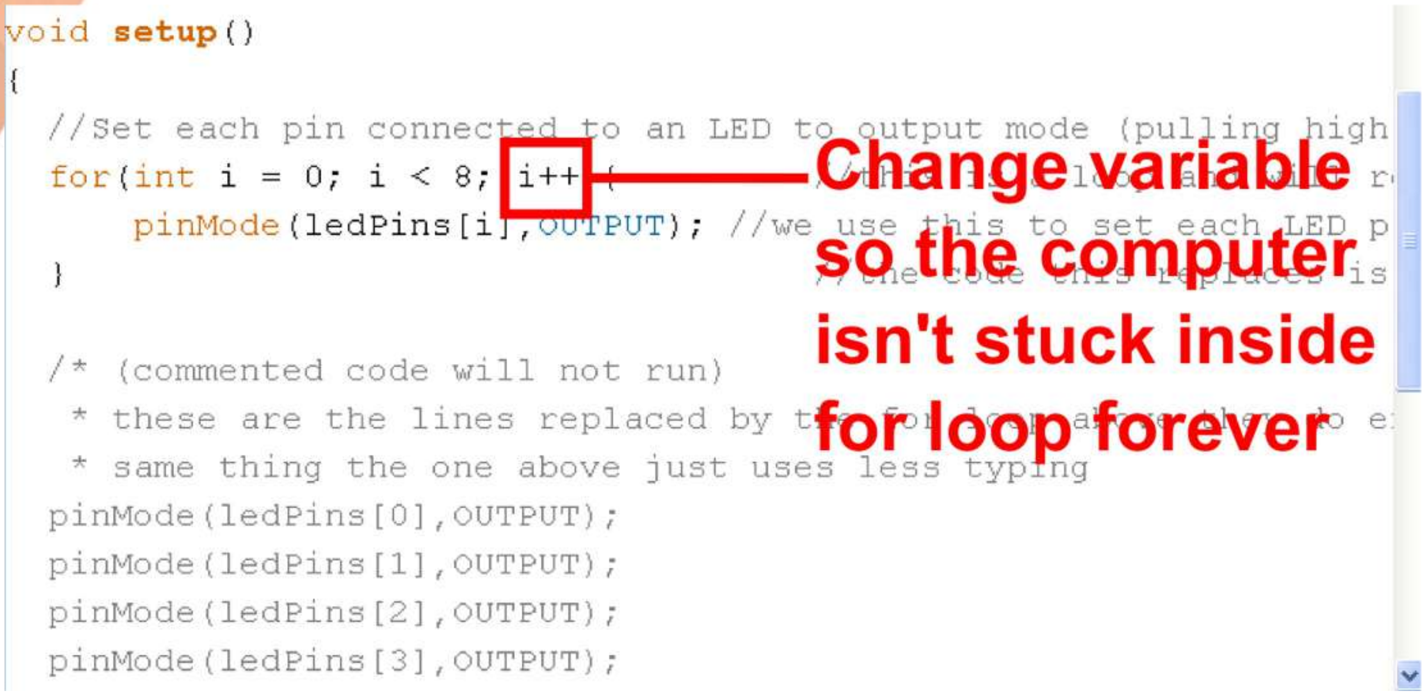
Structure For



```
void setup()
{
  //Set each pin connected to an LED to output mode (pulling high
  for(int i = 0; i < 8; i++){           //this is a loop and will r
      pinMode(ledPins[i],OUTPUT); //we use this to set each LED p
  }                                      //the code this replaces is

  /* (commented code will not run)
   * these are the lines replaced by the for loop above they do e
   * same thing the one above just uses less typing
  pinMode(ledPins[0],OUTPUT);
  pinMode(ledPins[1],OUTPUT);
  pinMode(ledPins[2],OUTPUT);
  pinMode(ledPins[3],OUTPUT);
```

If this conditional is true do the code inside the curly brackets, if it's false the computer exits the for loop

Structure For


Change variable so the computer isn't stuck inside for loop forever

```
void setup()
{
  //Set each pin connected to an LED to output mode (pulling high
  for(int i = 0; i < 8; i++) {                  //this is a loop and will r
      pinMode(ledPins[i],OUTPUT); //we use this to set each LED p
  }                                             //the code this replaces is

  /* (commented code will not run)
   * these are the lines replaced by the for loop above they do e
   * same thing the one above just uses less typing
  pinMode(ledPins[0],OUTPUT);
  pinMode(ledPins[1],OUTPUT);
  pinMode(ledPins[2],OUTPUT);
  pinMode(ledPins[3],OUTPUT);
```

Structure For

```
int timer = 200; // The higher the number, the slower the
timing.
int ledPins[] = {11, 12, 13}; // an array of pin numbers to
which LEDs are attached
int pinCount = 3; // the number of pins
void setup() {
   for (int thisPin = 0; thisPin < pinCount; thisPin++) {
         pinMode(ledPins[thisPin], OUTPUT);
  }
 }
void loop() {
for (int thisPin = 0; thisPin < pinCount; thisPin++) {
     digitalWrite(ledPins[thisPin], HIGH);
     delay(timer);
     digitalWrite(ledPins[thisPin], LOW);
  }
}
```