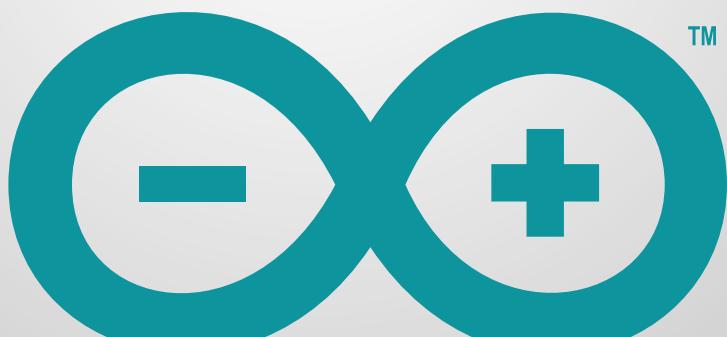
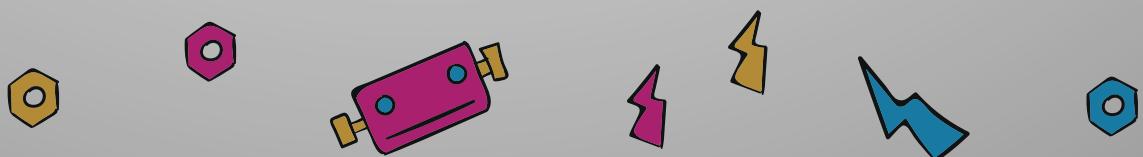




Building prototypes with Arduino



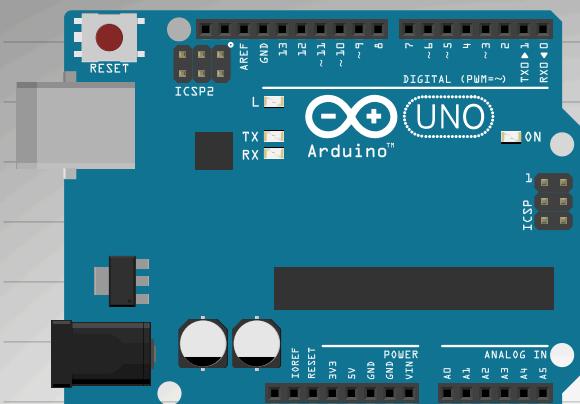
ARDUINO



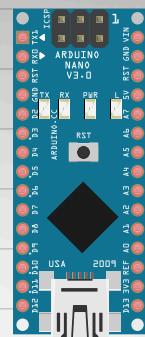
Arduino

Arduino is an open electronics platform for prototyping based on **open software and hardware**.

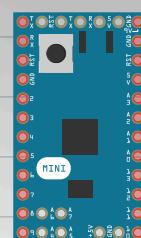
Arduino Uno



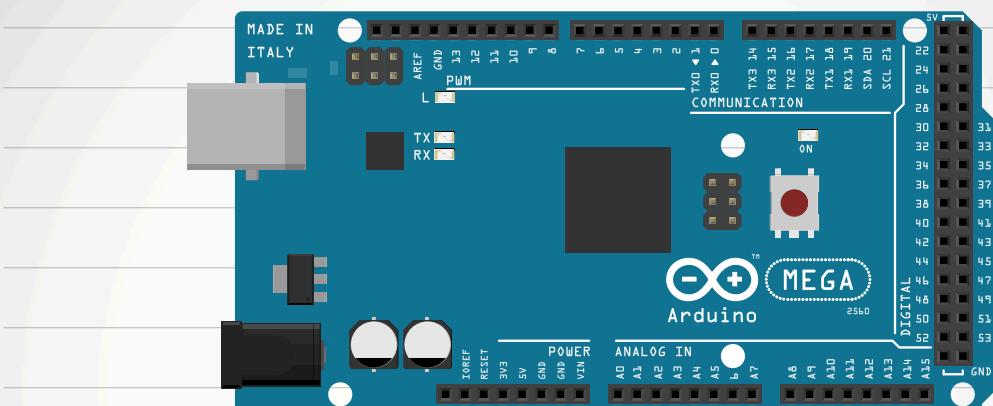
Arduino
Nano



Arduino
Micro



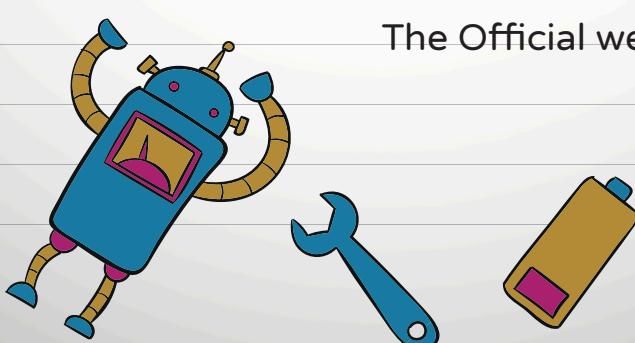
Arduino Mega



fritzing

With Arduino we can take information from the environment by connecting sensors through their input pins and act by controlling lights, motors and other actuators.

The Official website is www.arduino.cc



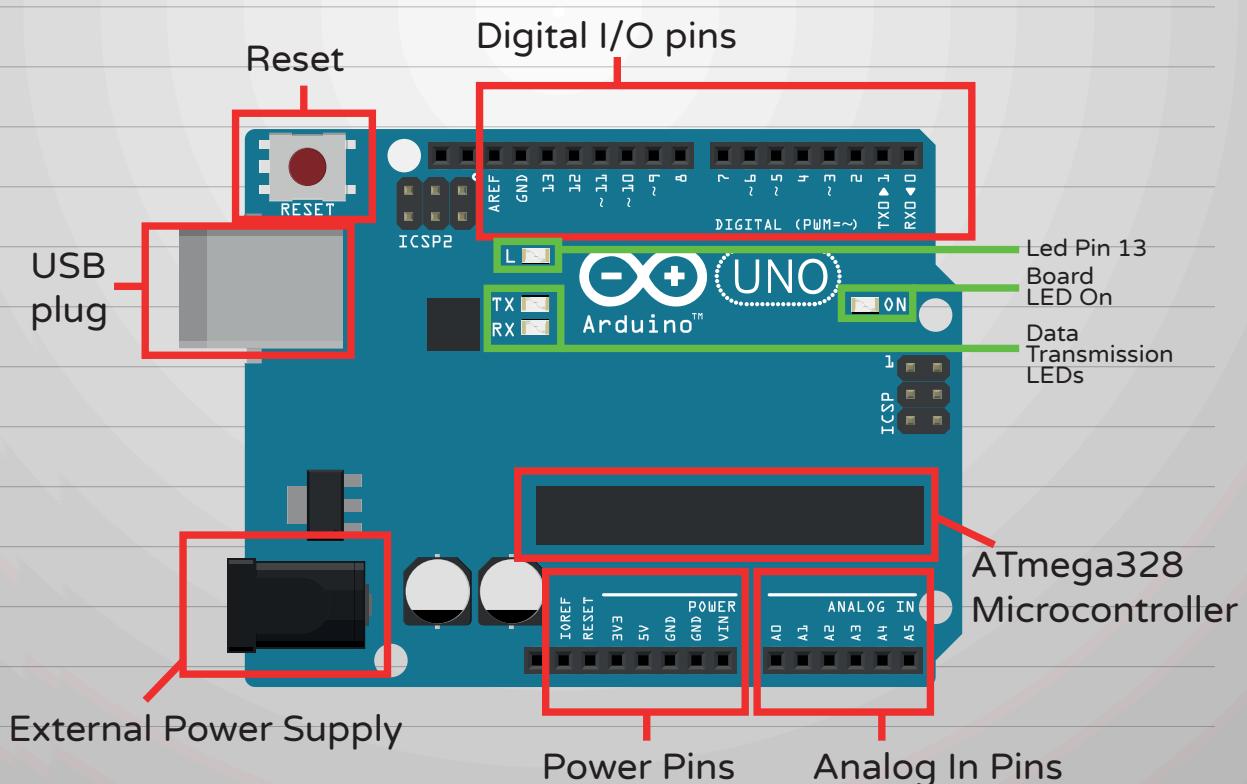
Arduino Uno Board Description

Arduino Uno is an electronic board based on the ATmega328 microcontroller.

It has 14 digital input/output pins, 6 of which can be used as PWM outputs (Pulse-Width Modulation), and other 6 analog inputs.

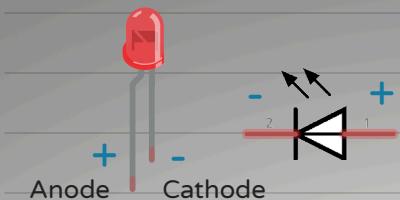
It also includes a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button.

The board contains everything needed for the microcontroller to work. Simply connect it to a computer with a USB cable or power it with an adapter.

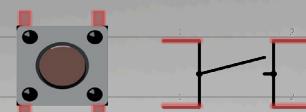


Materials

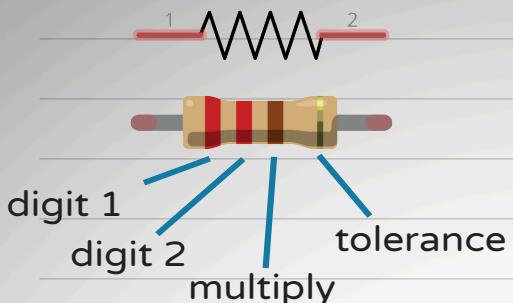
Led



Button



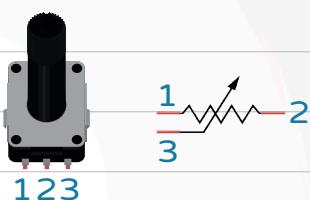
Resistencia



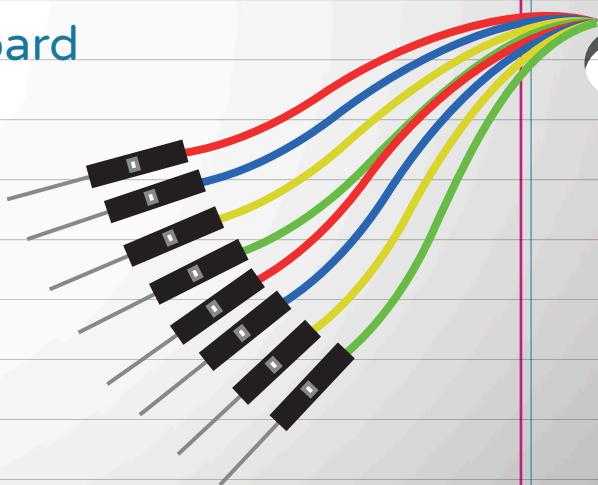
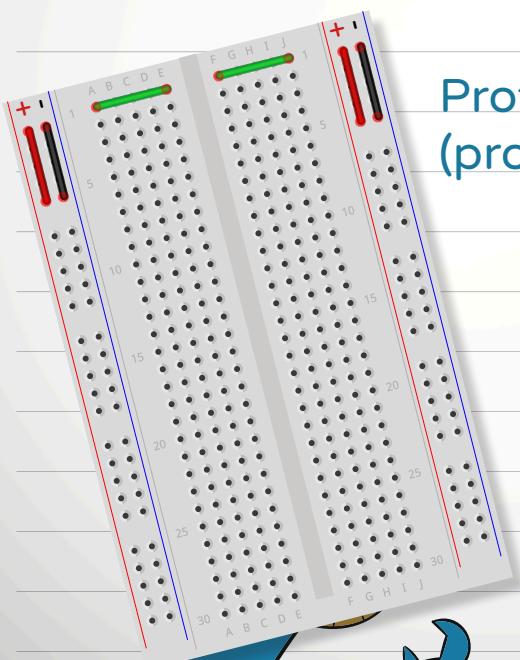
LDR Photoresistance



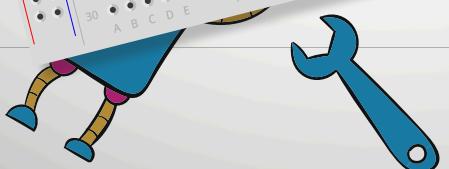
Potentiometer



Prototype Board
(protoboard)



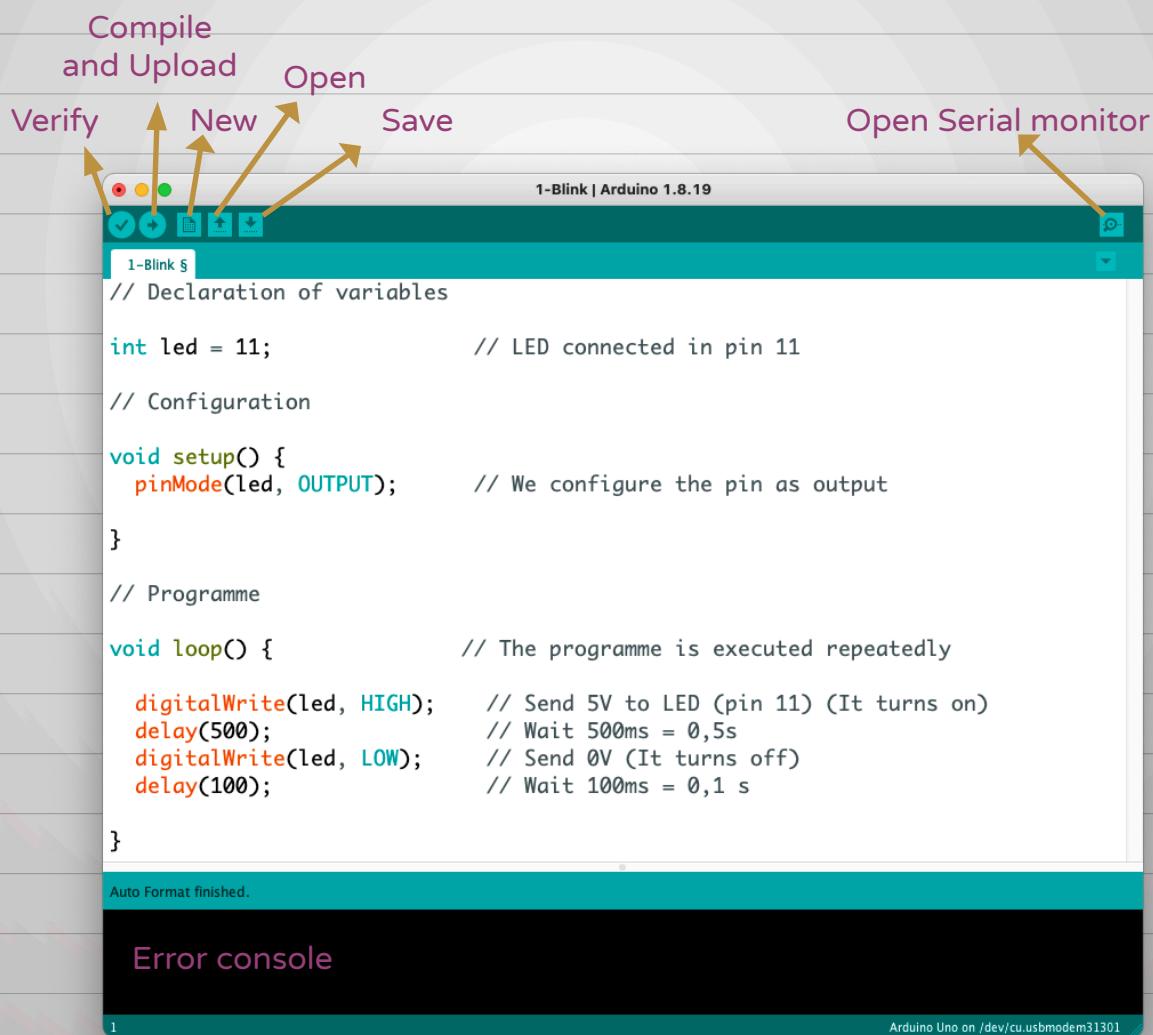
...and bridge cables



Arduino Environment

Arduino's programming environment is called Arduino's IDE, which stands for Integrated Development Environment.

The files used in Arduino's environment are called sketch and they are renamed by default as: **sketch+date+letter**. The file containing the programme code uses the ***.ino** extension.



```
1-Blink | Arduino 1.8.19
1-Blink.ino
// Declaration of variables
int led = 11; // LED connected in pin 11
// Configuration
void setup() {
  pinMode(led, OUTPUT); // We configure the pin as output
}
// Programme
void loop() { // The programme is executed repeatedly
  digitalWrite(led, HIGH); // Send 5V to LED (pin 11) (It turns on)
  delay(500); // Wait 500ms = 0,5s
  digitalWrite(led, LOW); // Send 0V (It turns off)
  delay(100); // Wait 100ms = 0,1 s
}

Auto Format finished.

Error console
```

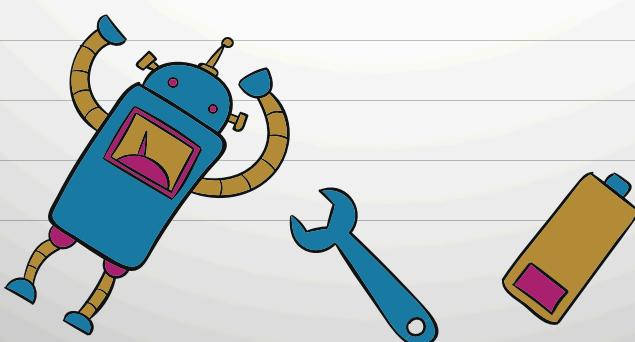
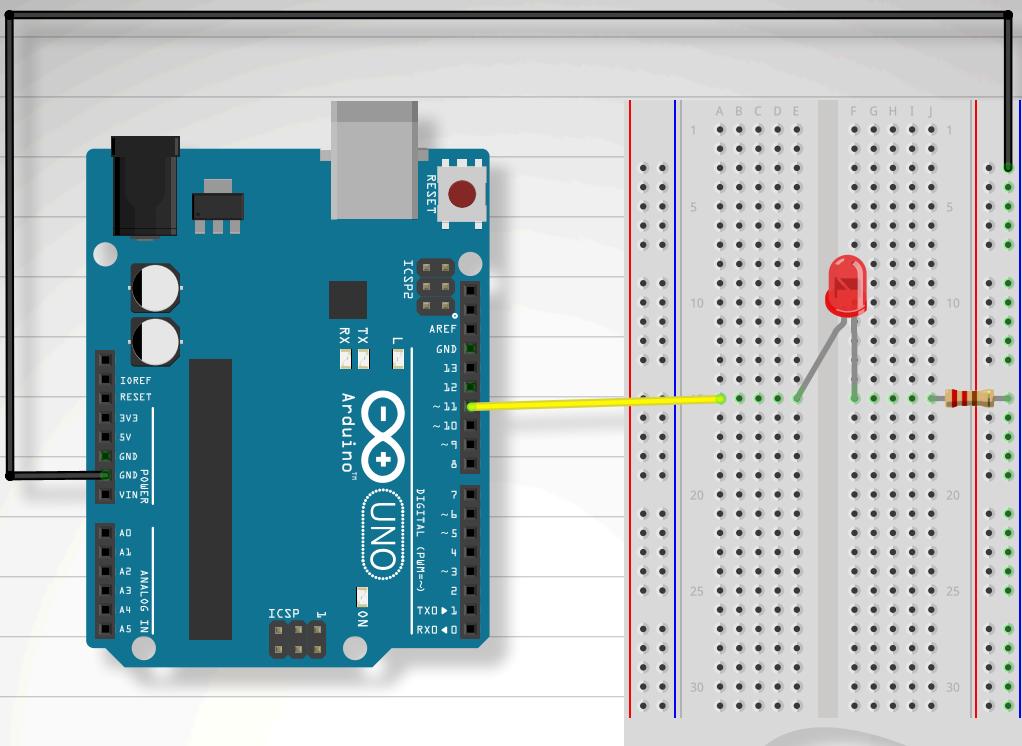
1 Arduino Uno on /dev/cu.usbmodem31301

Exercise 1.- Blink

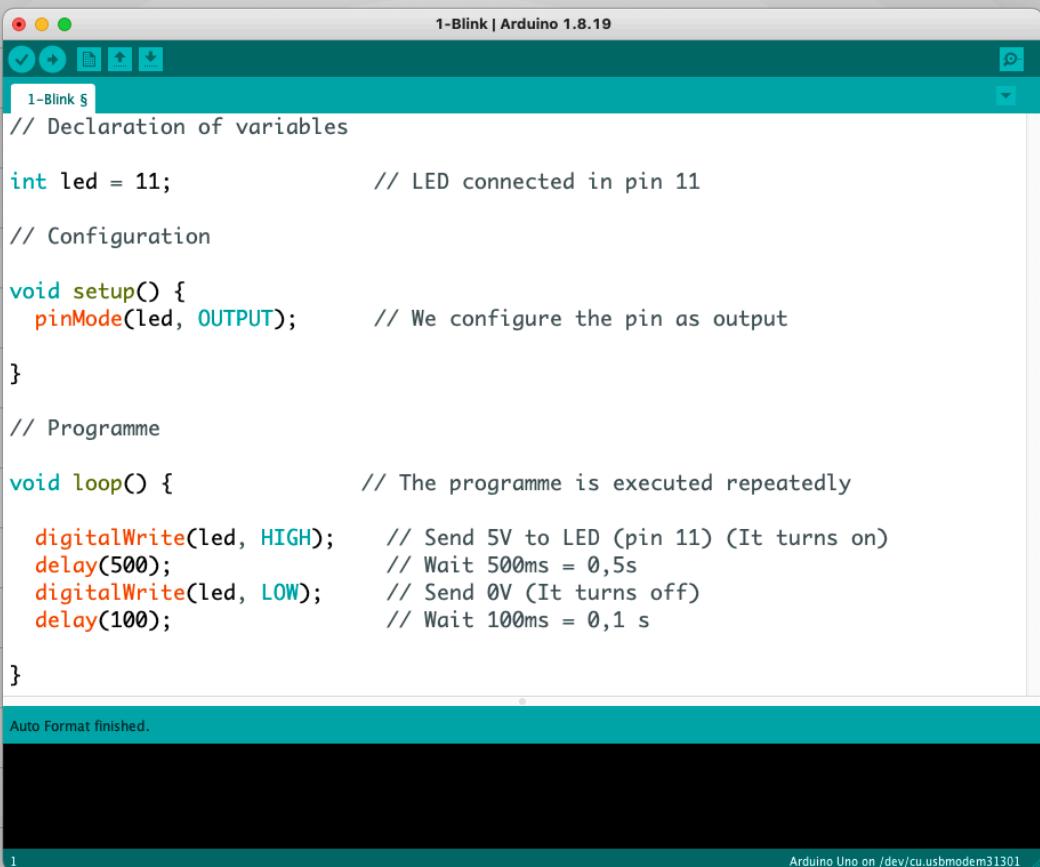
- Configure Arduino
- Make an LED blink and change the blinking frequency

Assembly

LED in pin 11 with a protection resistance of 220Ω .



Code



The screenshot shows the Arduino IDE interface with the title bar "1-Blink | Arduino 1.8.19". The code editor contains the following sketch:

```
// Declaration of variables
int led = 11; // LED connected in pin 11

// Configuration

void setup() {
    pinMode(led, OUTPUT); // We configure the pin as output
}

// Programme

void loop() { // The programme is executed repeatedly

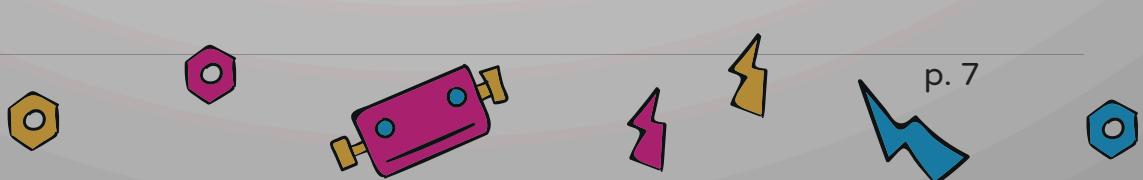
    digitalWrite(led, HIGH); // Send 5V to LED (pin 11) (It turns on)
    delay(500); // Wait 500ms = 0,5s
    digitalWrite(led, LOW); // Send 0V (It turns off)
    delay(100); // Wait 100ms = 0,1 s
}

Auto Format finished.
```

The status bar at the bottom right indicates "Arduino Uno on /dev/cu.usbmodem31301".

Now it is your turn...

1. Try to change the blinking time.
2. Would you be able to make the LED blink just one time?
3. Use variables to define the blinking time.

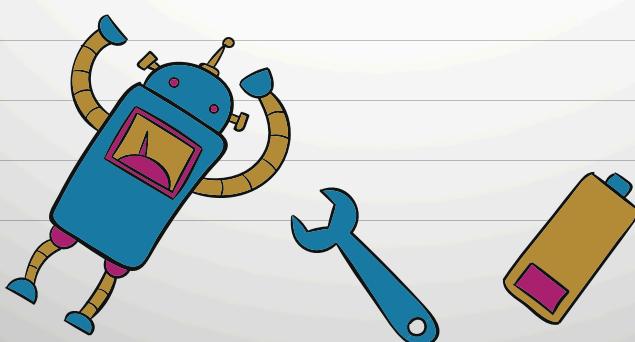
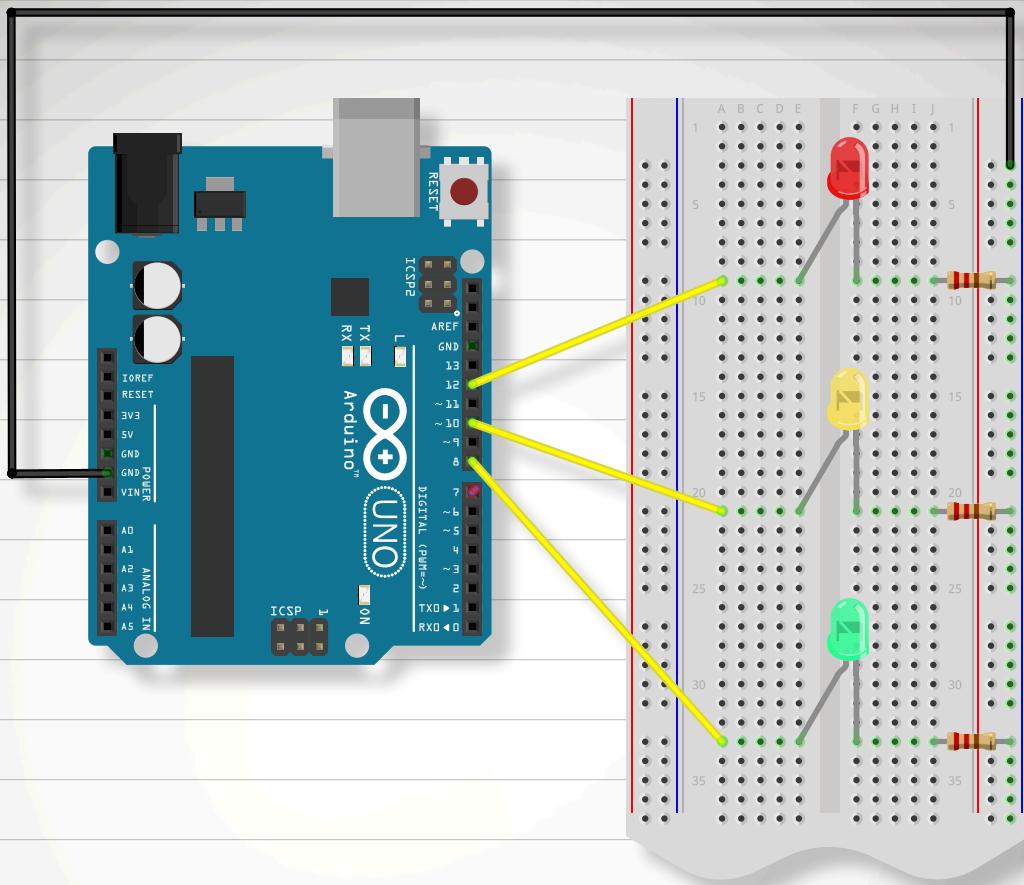


Exercise 2.- Traffic Light

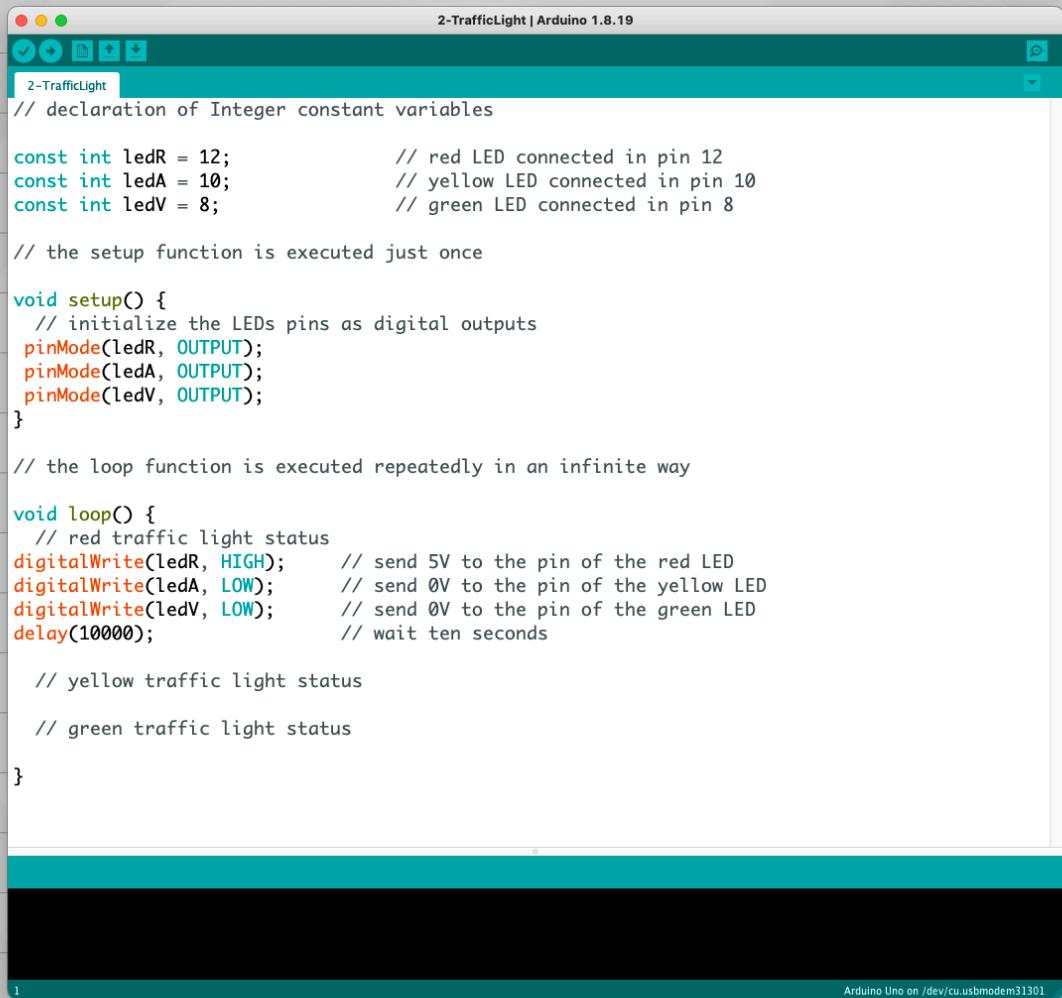
Simulate a car traffic light (red, yellow and green) working in a cyclical way.

Assembly

We have to connect 3 LEDs with their protection resistances to pins 12, 10 and 8.



Code



The screenshot shows the Arduino IDE interface with the title bar "2-TrafficLight | Arduino 1.8.19". The code is written in C++ and defines three integer constant variables for LED pins: ledR (pin 12), ledA (pin 10), and ledV (pin 8). The setup() function initializes the pins as outputs. The loop() function alternates between setting the red, yellow, and green LEDs to HIGH (5V) and LOW (0V), with a 10-second delay between each color. A status bar at the bottom indicates "Arduino Uno on /dev/cu.usbmodem31301".

```
// declaration of Integer constant variables

const int ledR = 12;           // red LED connected in pin 12
const int ledA = 10;           // yellow LED connected in pin 10
const int ledV = 8;            // green LED connected in pin 8

// the setup function is executed just once

void setup() {
    // initialize the LEDs pins as digital outputs
    pinMode(ledR, OUTPUT);
    pinMode(ledA, OUTPUT);
    pinMode(ledV, OUTPUT);
}

// the loop function is executed repeatedly in an infinite way

void loop() {
    // red traffic light status
    digitalWrite(ledR, HIGH);    // send 5V to the pin of the red LED
    digitalWrite(ledA, LOW);     // send 0V to the pin of the yellow LED
    digitalWrite(ledV, LOW);     // send 0V to the pin of the green LED
    delay(10000);               // wait ten seconds

    // yellow traffic light status

    // green traffic light status
}
```

Now it is your turn...

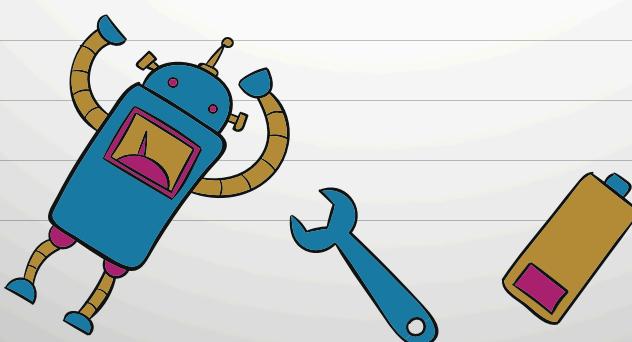
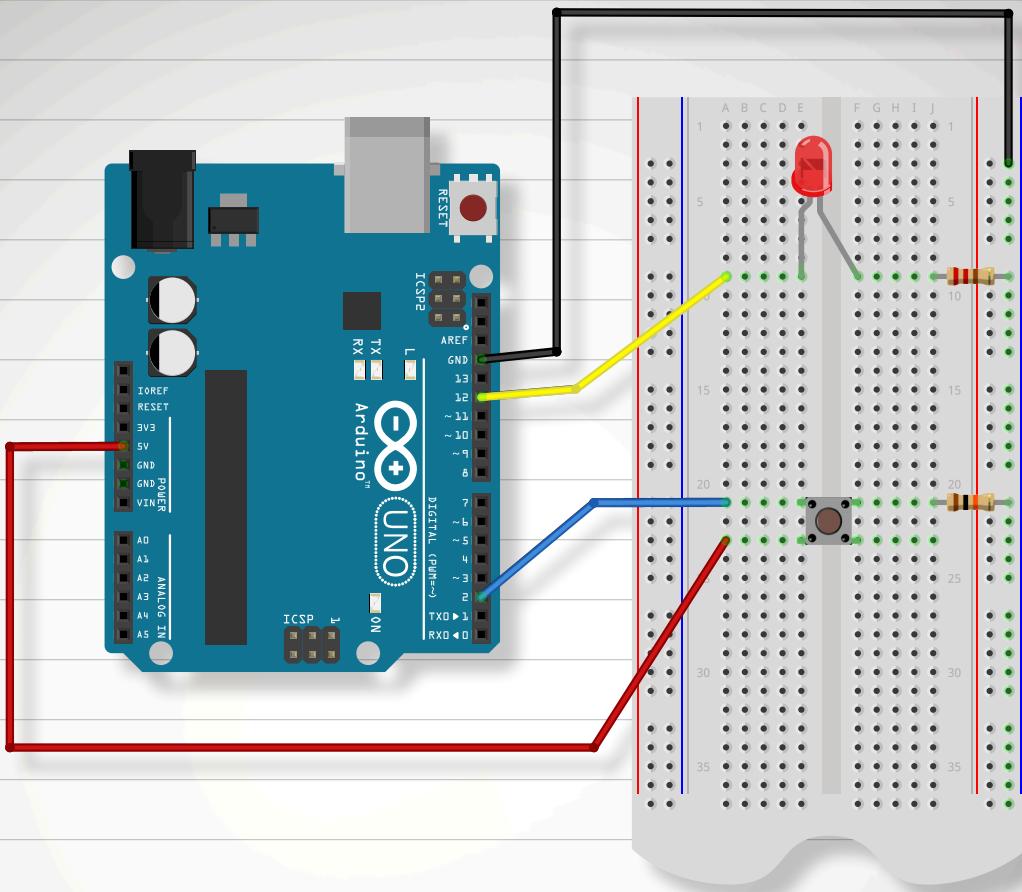
1. Complete the code with the traffic light in yellow and green.
2. Add a pedestrian traffic light (red and green LEDs) synchronised with the car traffic light.

Exercise 3.- Push Button and Led

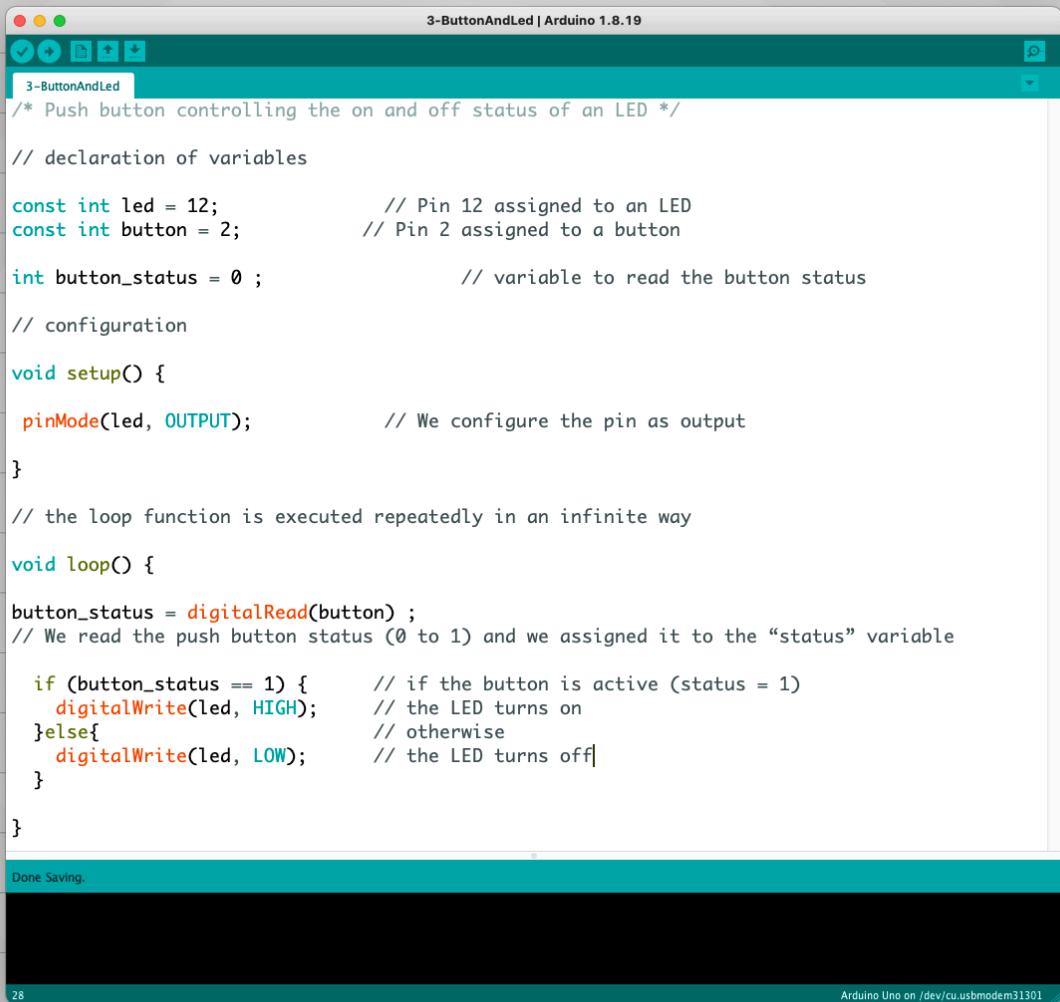
Control the on and off status of an LED with a push button, so that it turns on or off every time we press the button.

Assembly

We have to connect a push button with pin 2 with a $10\text{ k}\Omega$ resistance as shown on the diagram.



Code



The screenshot shows the Arduino IDE interface with the title bar "3-ButtonAndLed | Arduino 1.8.19". The code is for a push button controlling an LED. It includes declarations for an LED on pin 12 and a push button on pin 2, a variable for reading button status, setup() and loop() functions, and an if-else block to toggle the LED state based on the button's status. The status bar at the bottom indicates "Done Saving." and "Arduino Uno on /dev/cu.usbmodem31301".

```
/* Push button controlling the on and off status of an LED */

// declaration of variables

const int led = 12;           // Pin 12 assigned to an LED
const int button = 2;         // Pin 2 assigned to a button

int button_status = 0;        // variable to read the button status

// configuration

void setup() {
    pinMode(led, OUTPUT);      // We configure the pin as output
}

// the loop function is executed repeatedly in an infinite way

void loop() {
    button_status = digitalRead(button); // We read the push button status (0 to 1) and we assigned it to the "status" variable

    if (button_status == 1) {          // if the button is active (status = 1)
        digitalWrite(led, HIGH);       // the LED turns on
    } else{                           // otherwise
        digitalWrite(led, LOW);        // the LED turns off
    }
}

Done Saving.

28
Arduino Uno on /dev/cu.usbmodem31301
```

Now it is your turn...

1. Add another LED and make them turn on alternatively when pressing and releasing the push button.
2. Every time we press the push button make the LED turn on and off twice.
3. Push button with memory. If the LED was initially on, turn it off, and if it was off, turn it on.



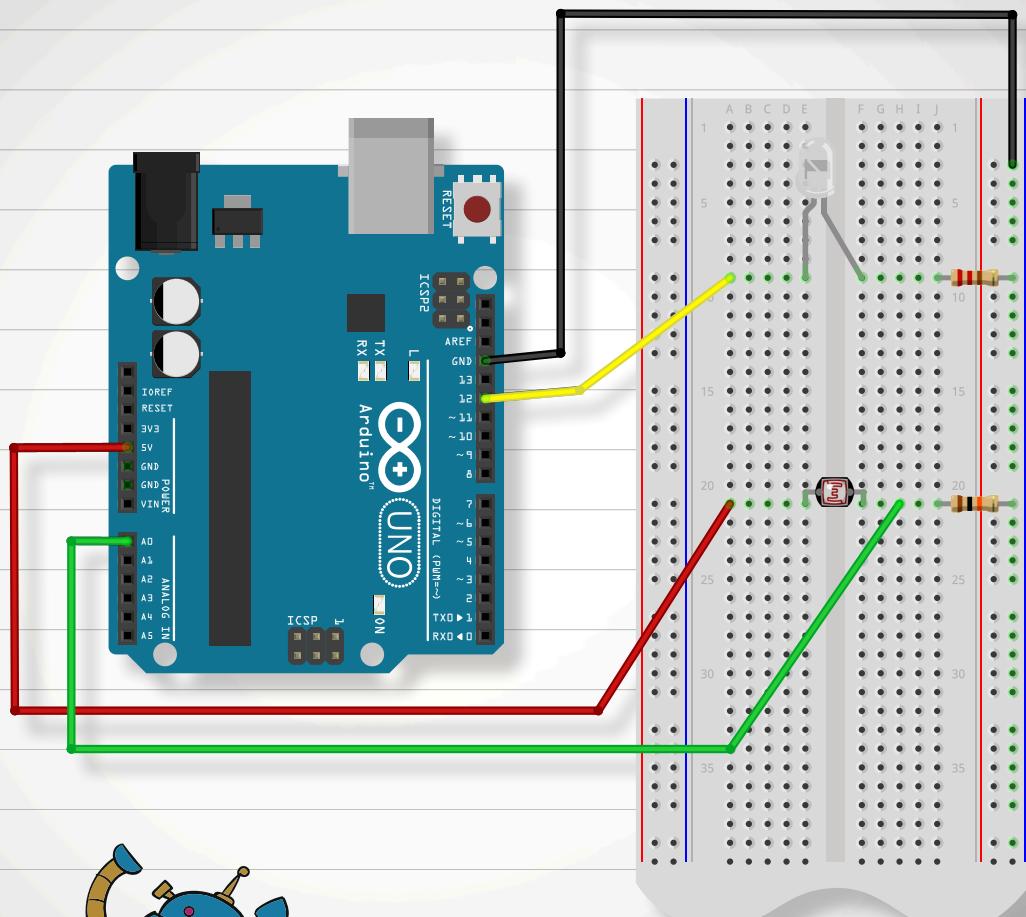
Exercise 4.- Light Switch

Turn on an LED depending on the light intensity that the light sensor receives. In the dark the LED is on and in high lighting is off.

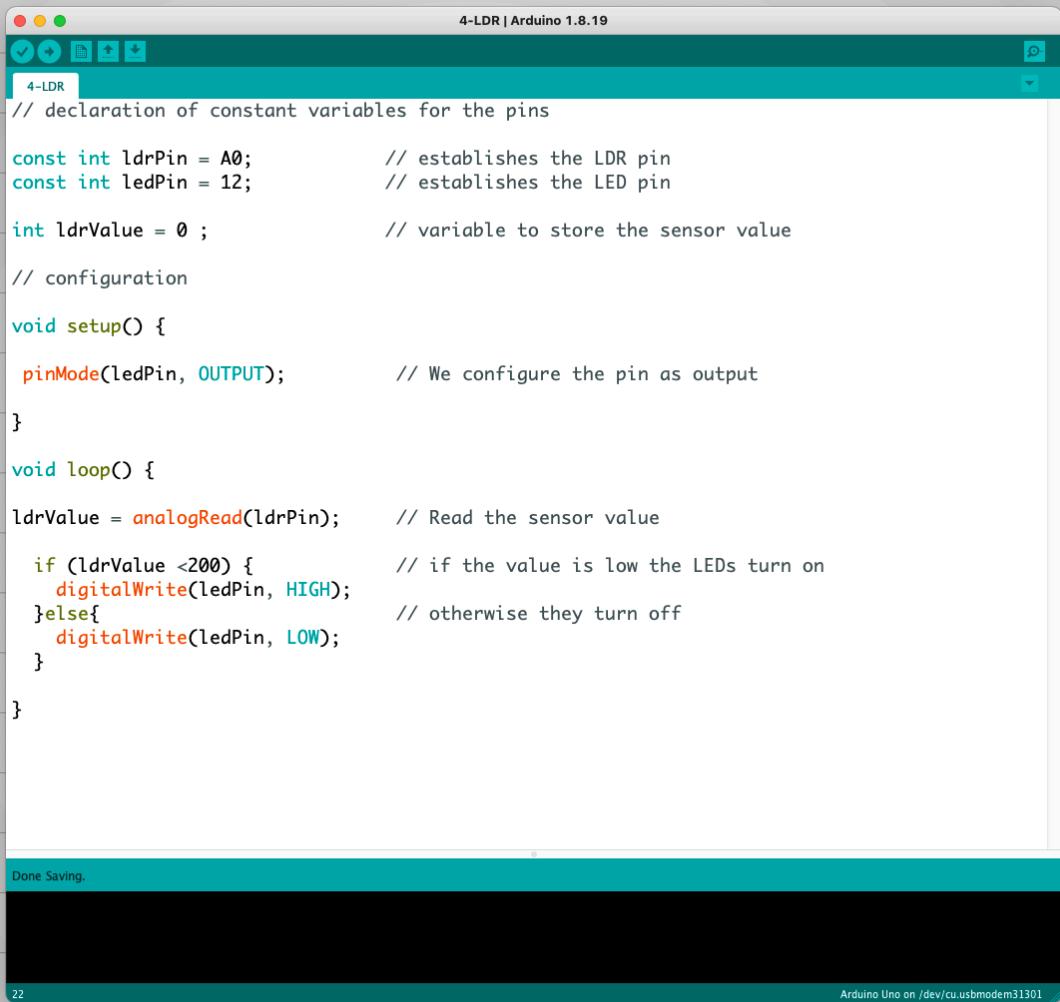
First, we have to read the values provided by the sensor and secondly the different light conditions.

Assembly

Connect an LDR to the analog pin A0 with a 10 kΩ resistance as shown on the diagram.



Code



The screenshot shows the Arduino IDE interface with the title bar "4-LDR | Arduino 1.8.19". The code window contains the following sketch:

```
// declaration of constant variables for the pins
const int ldrPin = A0;           // establishes the LDR pin
const int ledPin = 12;           // establishes the LED pin
int ldrValue = 0;                // variable to store the sensor value

// configuration

void setup() {
    pinMode(ledPin, OUTPUT);      // We configure the pin as output
}

void loop() {
    ldrValue = analogRead(ldrPin); // Read the sensor value
    if (ldrValue <200) {          // if the value is low the LEDs turn on
        digitalWrite(ledPin, HIGH);
    }else{                        // otherwise they turn off
        digitalWrite(ledPin, LOW);
    }
}
```

The status bar at the bottom indicates "Done Saving." and "Arduino Uno on /dev/cu.usbmodem31301".

Now it is your turn...

1. Add two more LEDs in a way that you create a light scale. When there is a lot of light, they all can be off, and as soon as the light intensity diminishes more LEDs are being turned on.
2. Based on the previous proposal, make the LEDs blink when the light intensity is very low.



Exercise 5.- PWM Signals

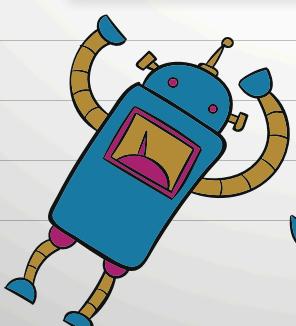
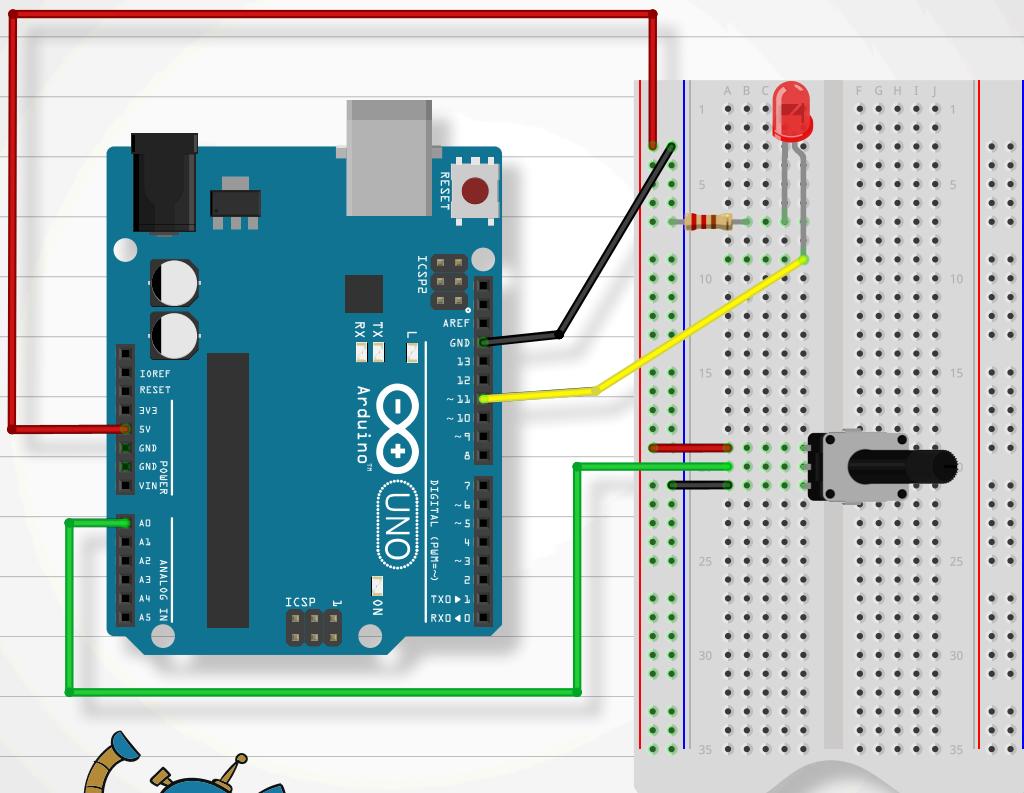
We are going to control the brightness of one LED by sending a PWM signal that varies according to the analog reading of a potentiometer.



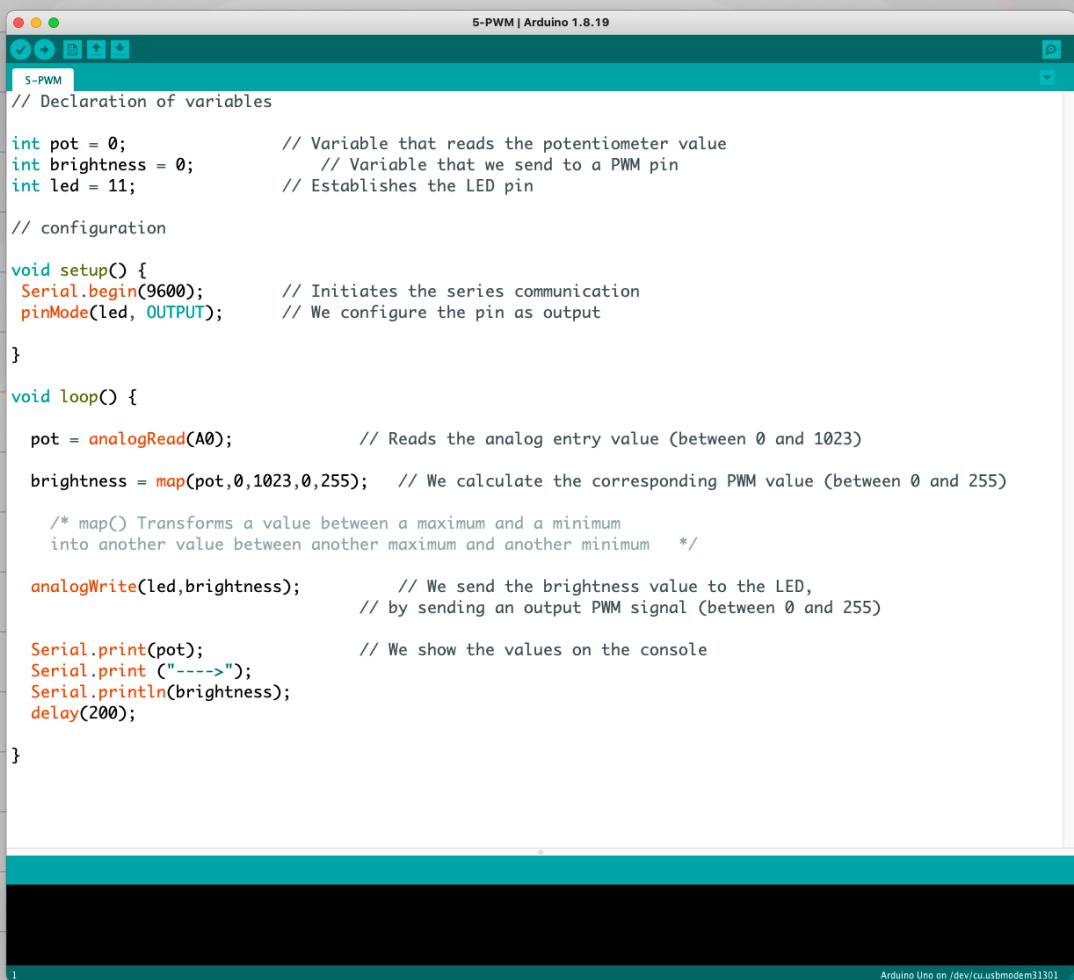
Assembly

We have to connect a potentiometer with an analog input and make the reading according to the position of the cursor. The reading values will range from 0 to 1023.

We will connect an LED with its protection resistance to a digital output PWM (~). We will send it a value between 0 and 255, getting different brightness levels.



Code



The screenshot shows the Arduino IDE interface with the title bar "5-PWM | Arduino 1.8.19". The code is for a potentiometer-controlled LED brightness. It includes variable declarations, setup, and loop functions. The setup function initializes serial communication at 9600 baud and sets pin 11 as an output for the LED. The loop function reads the analog value from pin A0, maps it to a PWM value between 0 and 255, and then sends this value to pin 11 via analogWrite. It also prints the potentiometer value and the calculated brightness to the Serial Monitor. A black redaction box covers the serial output area.

```
// Declaration of variables

int pot = 0;           // Variable that reads the potentiometer value
int brightness = 0;     // Variable that we send to a PWM pin
int led = 11;          // Establishes the LED pin

// configuration

void setup() {
  Serial.begin(9600);      // Initiates the series communication
  pinMode(led, OUTPUT);    // We configure the pin as output
}

void loop() {
  pot = analogRead(A0);      // Reads the analog entry value (between 0 and 1023)
  brightness = map(pot,0,1023,0,255); // We calculate the corresponding PWM value (between 0 and 255)

  /* map() Transforms a value between a maximum and a minimum
   * into another value between another maximum and another minimum */
  analogWrite(led,brightness); // We send the brightness value to the LED,
                             // by sending an output PWM signal (between 0 and 255)

  Serial.print(pot);
  Serial.print ("---->");
  Serial.println(brightness);
  delay(200);
}
```

Now it is your turn...

1. Simulate the effect of fire by means of an LED. You can generate an arbitrary brightness and an arbitrary waiting time for an LED.

You can use the **random** operator, both for the brightness and for the time. **Brightness = random (20,255)**.

2. Create a programme where the LED blinks in a different frequency according to the potentiometer position. For example, for a 0 value in the potentiometer the blinking will be performed with a 100 ms interval, and for a 1023 value in the potentiometer the interval will be 1000 ms. Use a map function in order to transform the analog interval into the time interval.



Exercise 6.- Loops

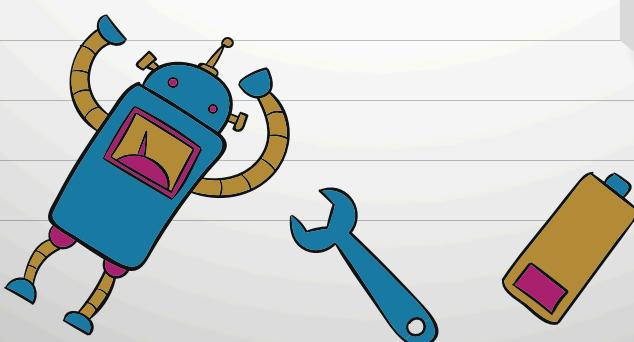
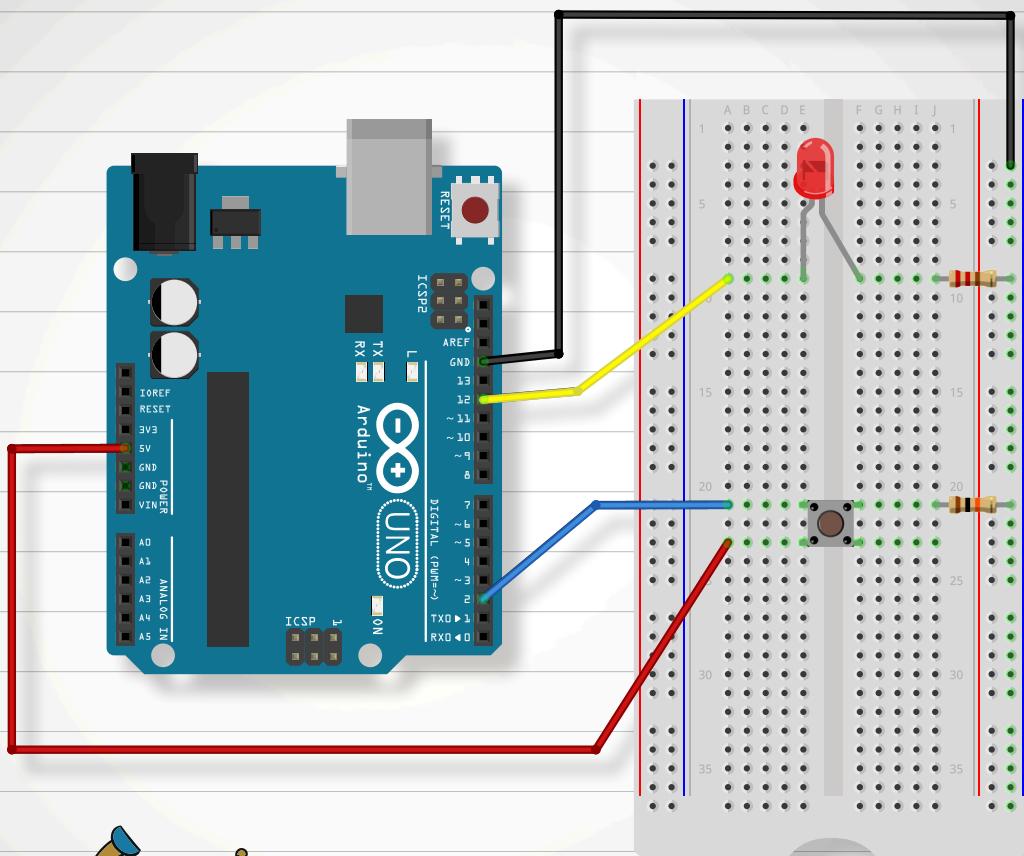
In this example we will make some instructions be repeated a determined number of times.

For example, we will make the LED turn on and off several times, we wait some time and we repeat the process.

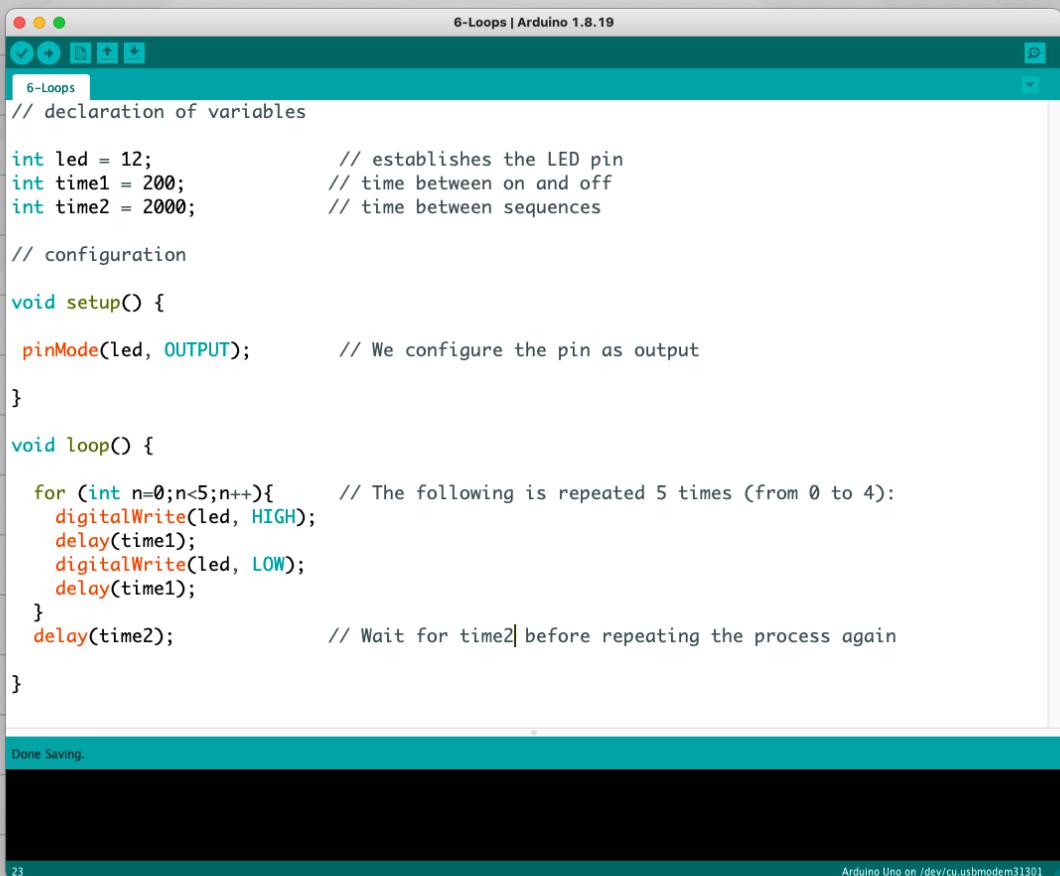
Assembly

We need an LED and a push button with their respective protection resistances. For this case we will use the function:

`for (start, condition, modifier) {process}`



Code



The screenshot shows the Arduino IDE interface with the title bar "6-Loops | Arduino 1.8.19". The code window contains the following Arduino sketch:

```
// declaration of variables

int led = 12;           // establishes the LED pin
int time1 = 200;         // time between on and off
int time2 = 2000;        // time between sequences

// configuration

void setup() {
    pinMode(led, OUTPUT); // We configure the pin as output
}

void loop() {
    for (int n=0;n<5;){      // The following is repeated 5 times (from 0 to 4):
        digitalWrite(led, HIGH);
        delay(time1);
        digitalWrite(led, LOW);
        delay(time1);
    }
    delay(time2);           // Wait for time2 before repeating the process again
}
```

The status bar at the bottom indicates "Done Saving." and "Arduino Uno on /dev/cu.usbmodem31301".

Now it is your turn...

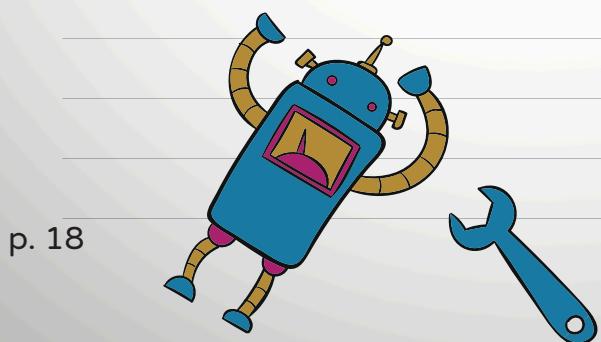
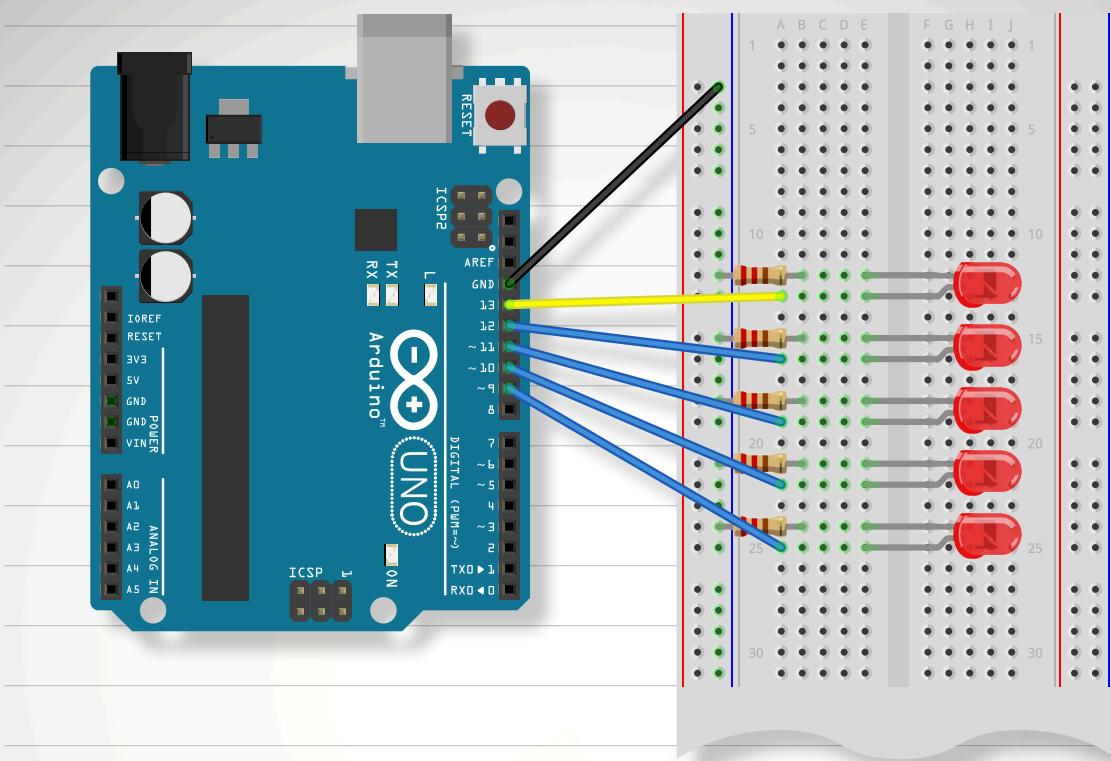
1. Connect a second LED and in each cycle make the first LED blink 6 times and the second LED three times.
2. Make possible that every time we press the push button a LED turns on and off three times. When the push button is not active, the LED remains off.

Exercise 7.- Knight Rider

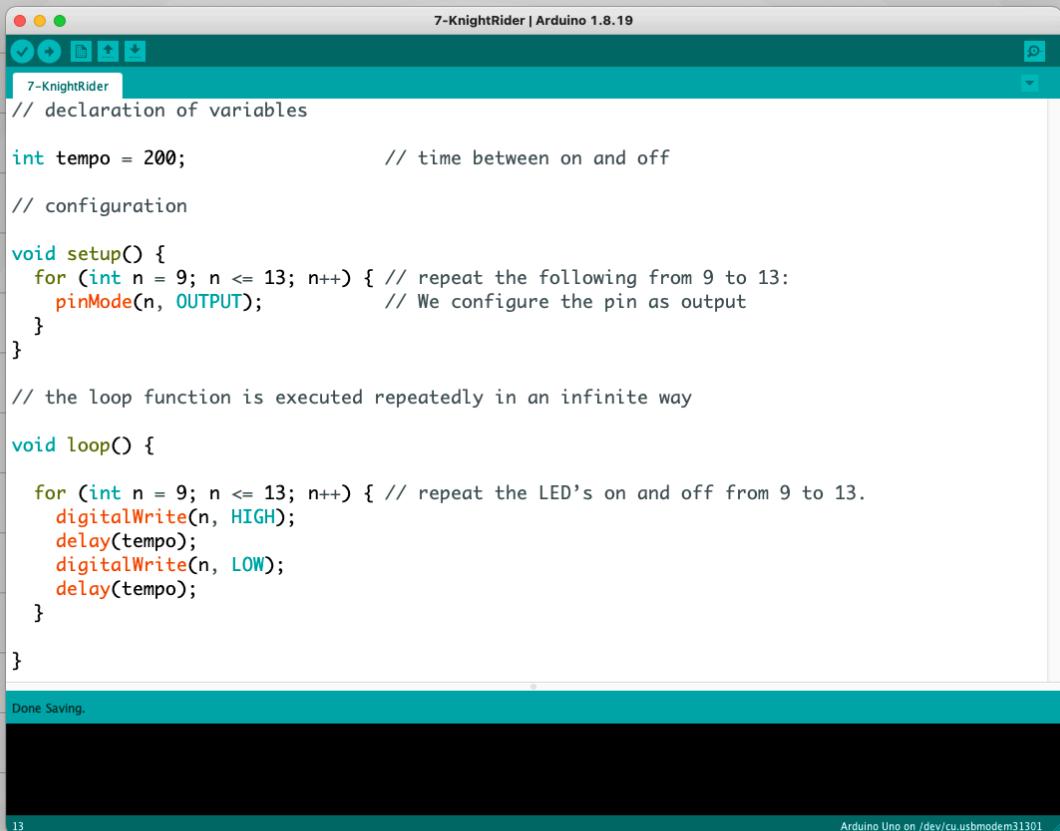
We will simulate the car from “Knight Rider” making a series of LEDs turn on and off in a consecutive order.

Assembly

We have to connect 5 LEDs with their corresponding protection resistances in pins 9 to 13.



Code



The screenshot shows the Arduino IDE interface with the title bar "7-KnightRider | Arduino 1.8.19". The code is written in C++ and defines a function to set up pins 9 through 13 as outputs, and a loop function that alternates the state of these pins (HIGH then LOW) with a delay of 200ms between each pin. The status bar at the bottom right indicates "Arduino Uno on /dev/cu.usbmodem31301".

```
// declaration of variables

int tempo = 200; // time between on and off

// configuration

void setup() {
  for (int n = 9; n <= 13; n++) { // repeat the following from 9 to 13:
    pinMode(n, OUTPUT); // We configure the pin as output
  }
}

// the loop function is executed repeatedly in an infinite way

void loop() {

  for (int n = 9; n <= 13; n++) { // repeat the LED's on and off from 9 to 13.
    digitalWrite(n, HIGH);
    delay(tempo);
    digitalWrite(n, LOW);
    delay(tempo);
  }
}

Done Saving.
```

Now it is your turn...

1. Add a descending loop to the previous programme so that the sequence takes place first in one way and then in another.
2. Make the LEDs turn on consecutively without turning off, and then, turn them off one by one in the same way or the opposite one.
3. We can control the sequence speed with a potentiometer connected into an analog input. Use the map function to transform the analog range (from 0 to 1023) into a time range (for example from 50 ms to 1000 ms).





arte e creación dixital

Sponsored by:



Developed by:



Support



www.amiguslabs.org

