



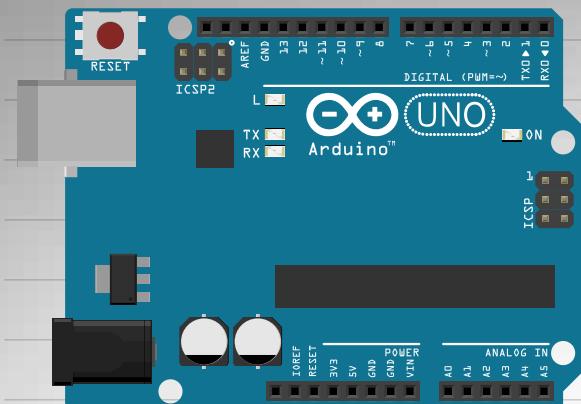
Obradorio de construcción de prototipos con Arduino



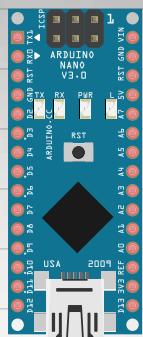
Arduino

Arduino é unha plataforma de electrónica aberta para a creación de prototipos basada en **software e hardware libres**.

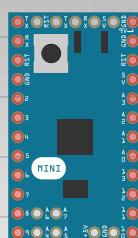
Arduino Uno



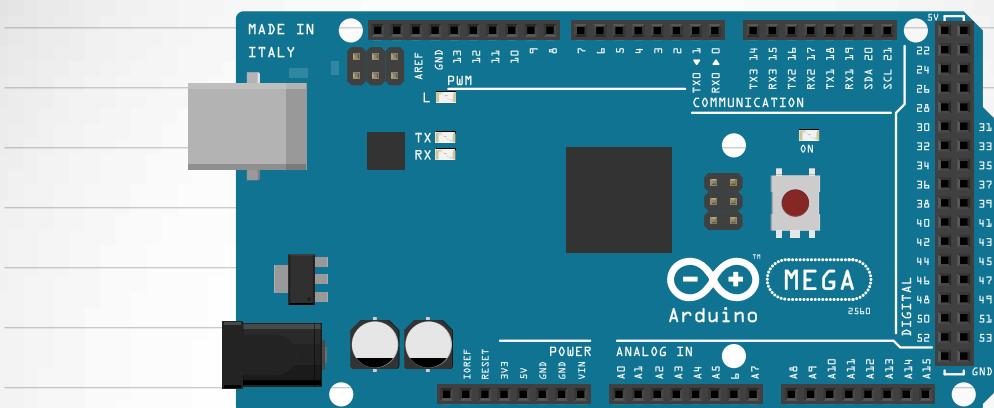
Arduino
Nano



Arduino
Micro



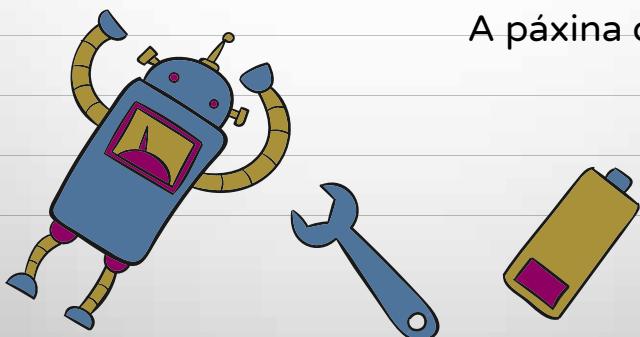
Arduino Mega



fritzing

Con Arduino podemos tomar información da contorna conectando sensores a través dos seus pinos de entrada e actuar controlando luces, motores e outros actuadores.

A páxina oficial é www.arduino.cc

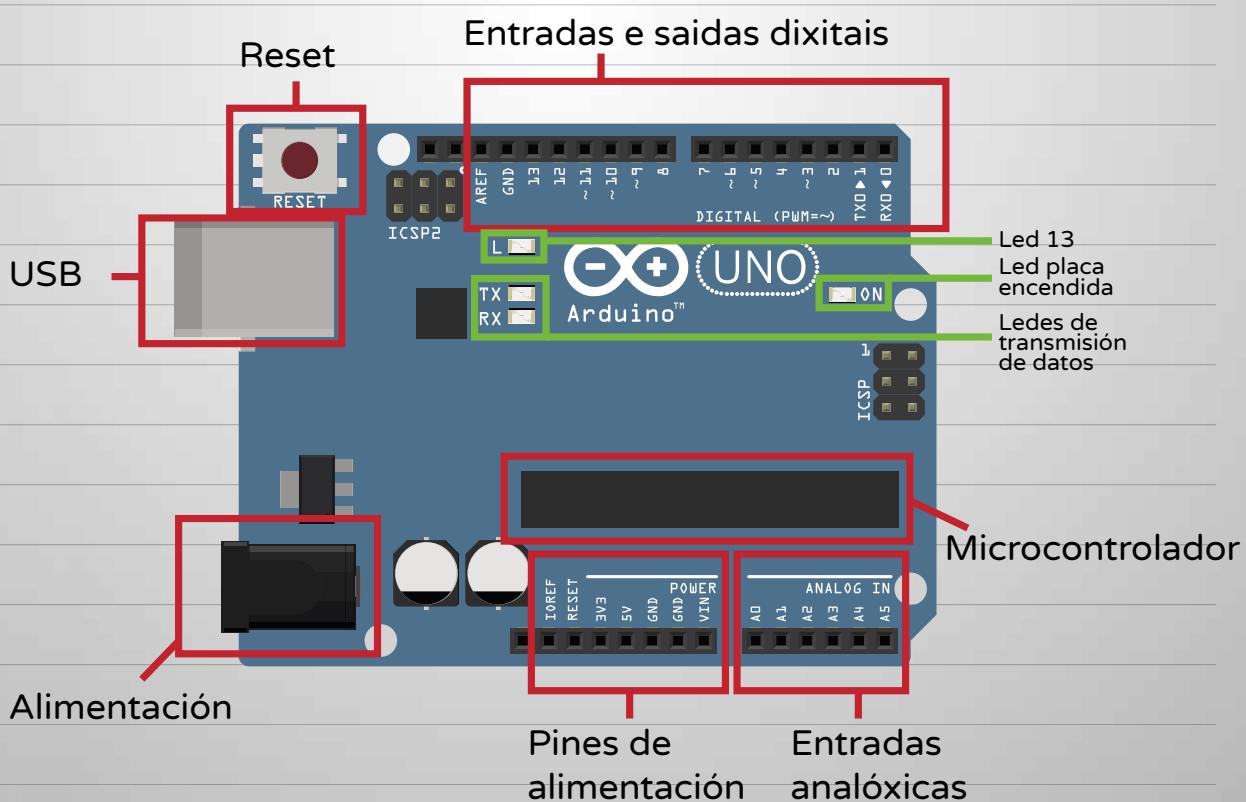


Descripción da placa Arduino Uno

Arduino Uno é unha placa electrónica baseada no microcontrolador ATmega328.

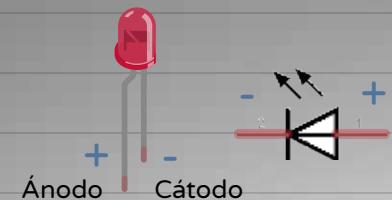
Conta con 14 entradas/saídas dixitais, das cales 6 pódense utilizar como saídas PWM ~ (Modulación por ancho de pulsos) e outras 6 son entradas analóxicas. Ademais, inclúe un resonador cerámico de 16 MHz, un conector USB, un conector de alimentación, unha cabeceira ICSP e un botón de reseteado.

A placa inclúe todo o necesario para que o microcontrolador faga o seu traballo, basta conectarla a un computador cun cable USB ou á corrente eléctrica a través dun transformador.

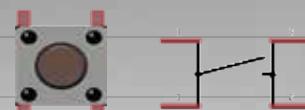


Materiais

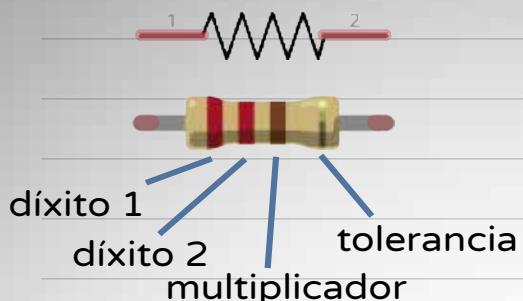
Led



Pulsador

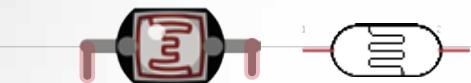


Resistencia

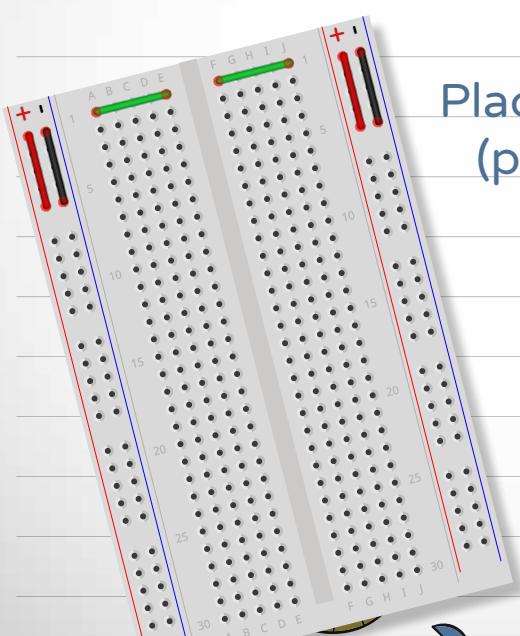
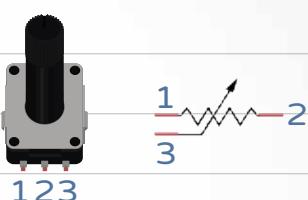


	1º Díxito	2º Díxito	Multiplicador	Tolerancia
Negro	0	0	$\times 10^0$	
Marrón	1	1	$\times 10^1$	$\pm 1\%$
Vermelho	2	2	$\times 10^2$	$\pm 2\%$
Laranxa	3	3	$\times 10^3$	
Amarelo	4	4	$\times 10^4$	
Verde	5	5	$\times 10^5$	$\pm 0,5\%$
Azul	6	6	$\times 10^6$	$\pm 0,25\%$
Violeta	7	7	$\times 10^7$	$\pm 0,1\%$
Gris	8	8	$\times 10^8$	$\pm 0,05\%$
Branco	9	9	$\times 10^9$	
Dourado			$\times 0,1$	$\pm 5\%$
Prateado			$\times 0,01$	$\pm 10\%$

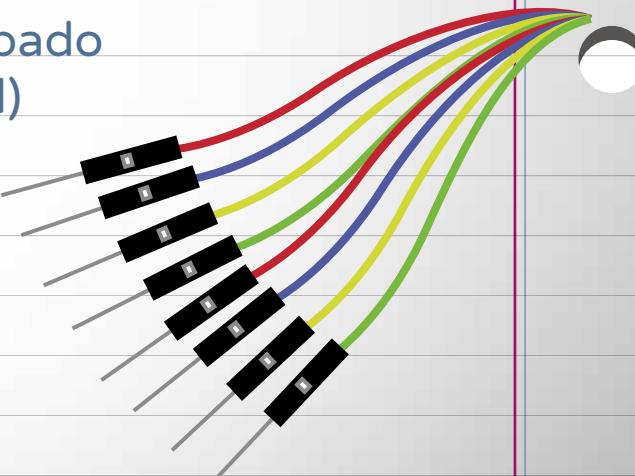
Fotoresistencia LDR



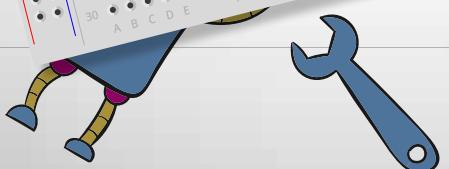
Potenciometro



Placa prototipado (protoboard)



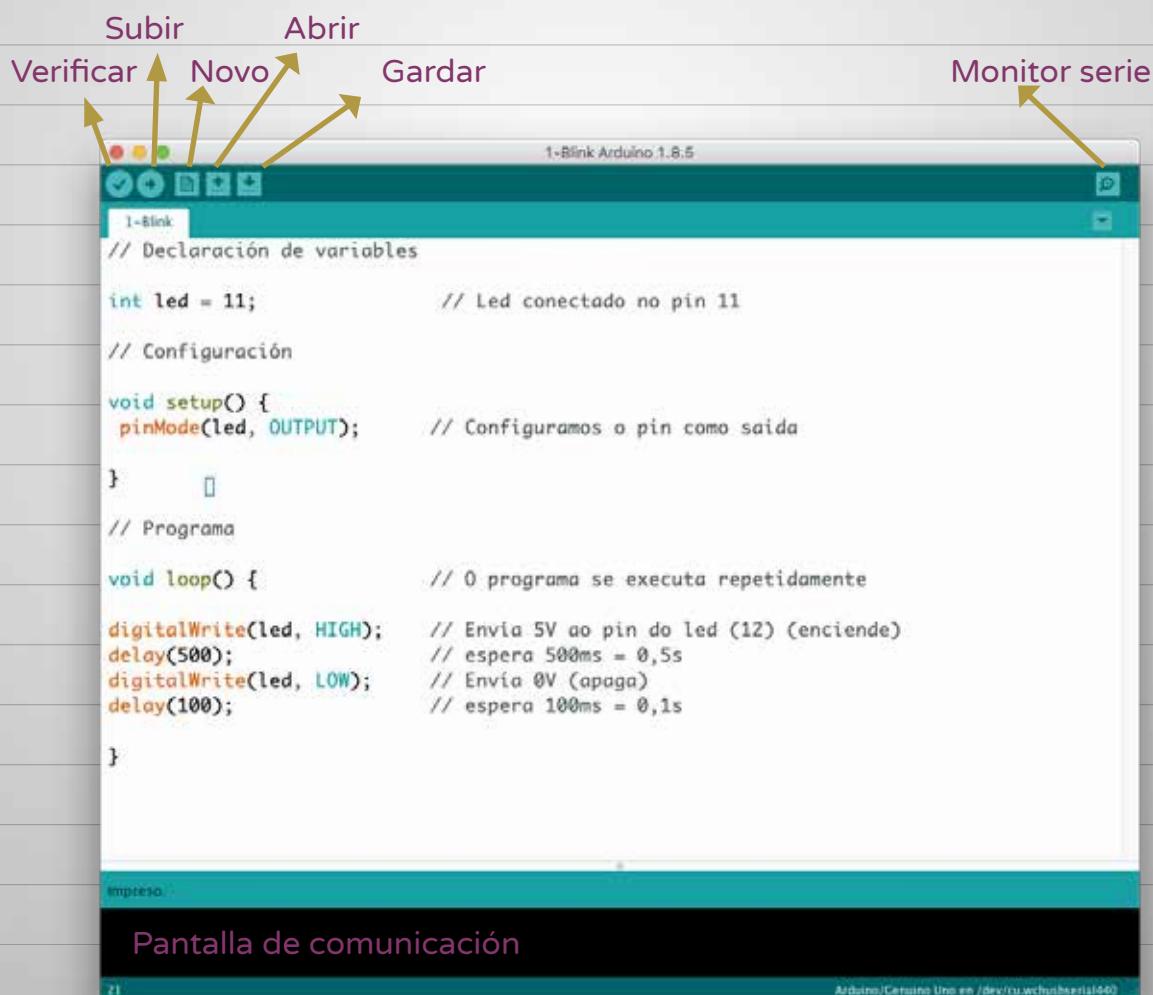
...e cables ponte



Contorna Arduino

A contorna de programación Arduino recibe o nome de IDE de Arduino, correspondente ás siglas de Integrated Development Environment ou contorna de desenvolvemento integrado.

Os ficheiros que manexan coa contorna de Arduino denomináñanse sketch e por defecto son renomeados do seguinte modo: **sketch+fecha+letra**. O ficheiro que contén o código do programa ten a extensión ***.ino**.



The screenshot shows the Arduino IDE interface. At the top, there's a menu bar with "Verificar", "Nuevo", "Abrir", "Gardar", and "Monitor serie". The main area displays a sketch named "1-Blink" with the following code:

```
1-Blink
// Declaración de variables

int led = 11; // Led conectado no pin 11

// Configuración

void setup() {
  pinMode(led, OUTPUT); // Configuramos o pin como salida
}

// Programa

void loop() { // O programa se executa repetidamente

  digitalWrite(led, HIGH); // Envía 5V ao pin do led (12) (enciende)
  delay(500); // espera 500ms = 0,5s
  digitalWrite(led, LOW); // Envía 0V (apaga)
  delay(100); // espera 100ms = 0,1s
}

Impresión...
```

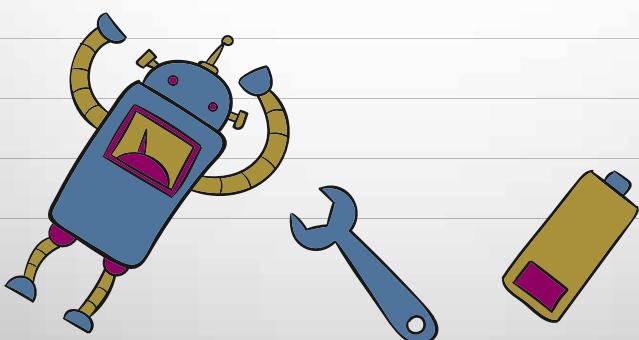
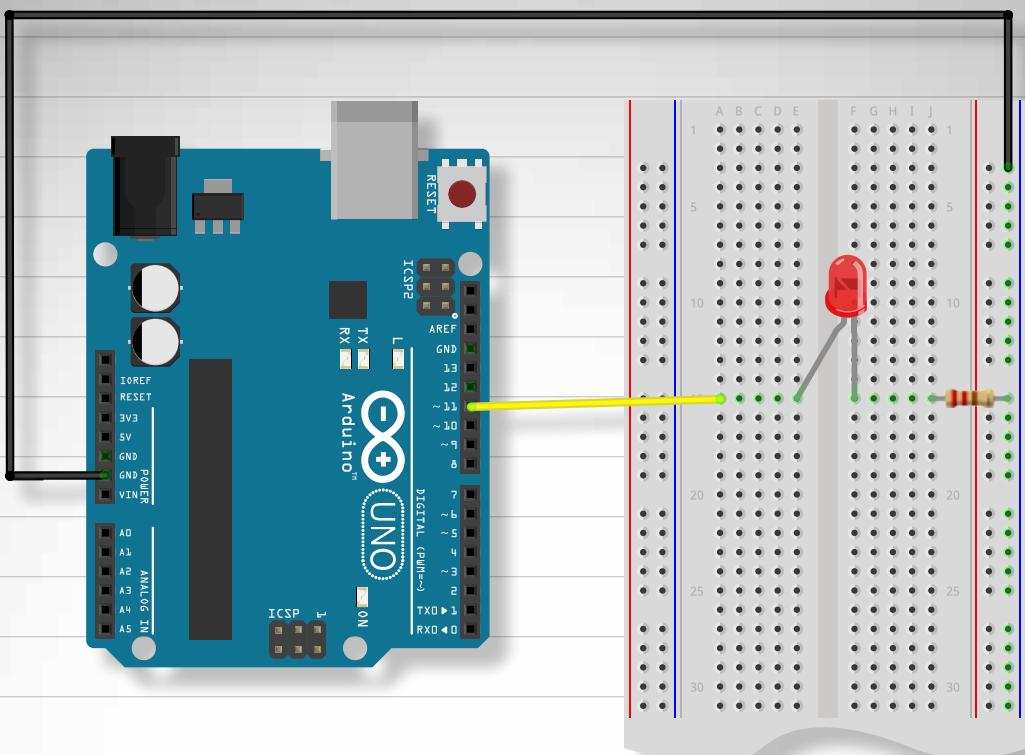
At the bottom, it says "Pantalla de comunicación" and shows the message "Arduino/Centuno Uno en /dev/cu.wchusbserial140".

Exercicio 1.- Blink

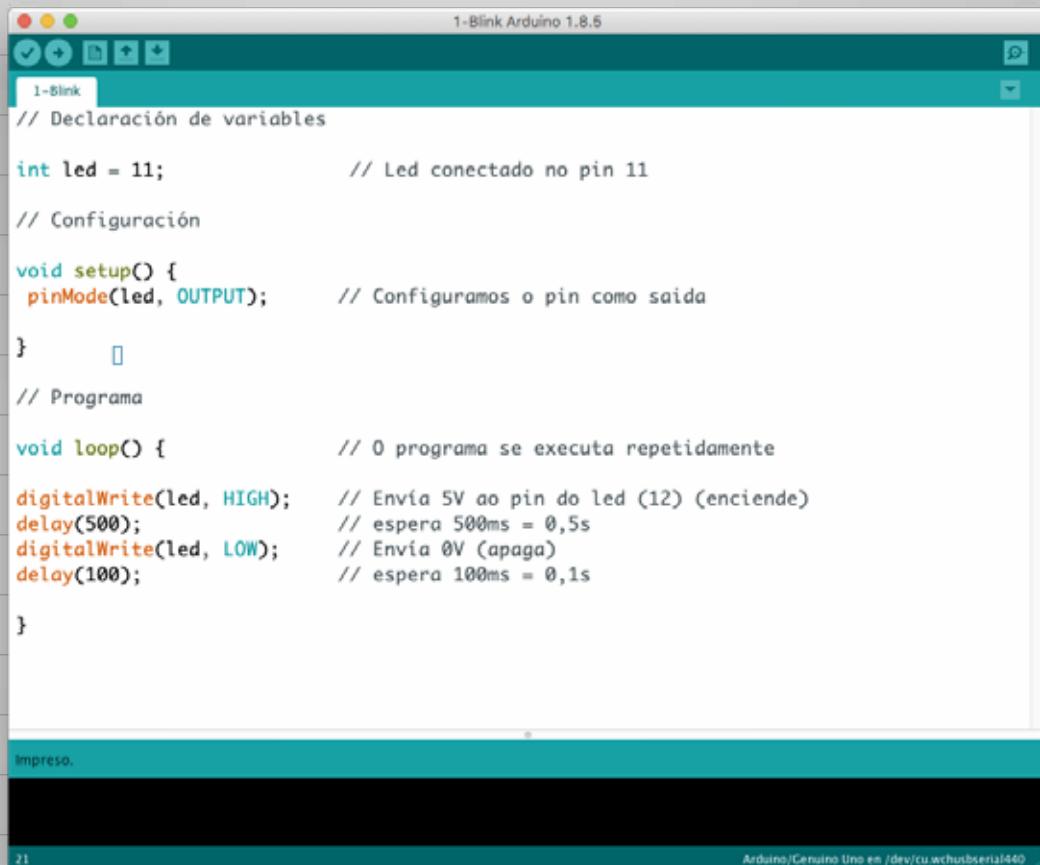
- Configurar Arduino
- Facer parpadear un LED e variar a frecuencia de parpadeo

Montaxe

Led no pin 11 cunha resistencia de protección de 220Ω .



Código



The screenshot shows the Arduino IDE interface with the title bar "1-Blink Arduino 1.8.5". The code window contains the following code:

```
// Declaración de variables
int led = 11; // Led conectado no pin 11

// Configuración

void setup() {
  pinMode(led, OUTPUT); // Configuramos o pin como saída
}

// Programa

void loop() { // O programa se executa repetidamente
  digitalWrite(led, HIGH); // Envia 5V ao pin do led (12) (enciende)
  delay(500); // espera 500ms = 0,5s
  digitalWrite(led, LOW); // Envia 0V (apaga)
  delay(100); // espera 100ms = 0,1s
}
```

The status bar at the bottom indicates "Impreso." and "Arduino/Genuino Uno en /dev/cu.wchusbserial440".

E agora ti.....

1. Proba a cambiar o tempo de parpadeo
2. Poderías facer que o LED parpadeara unha soa vez?
3. Usa variables para definir o tempo de parpadeo

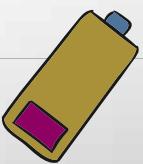
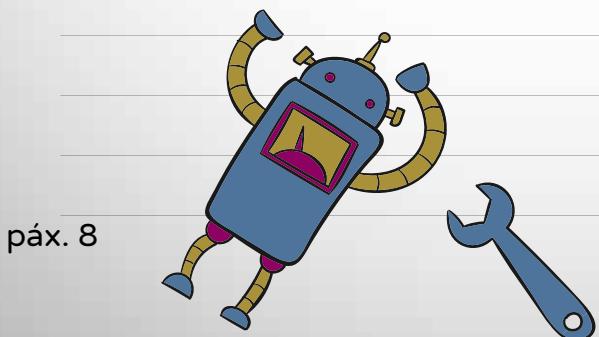
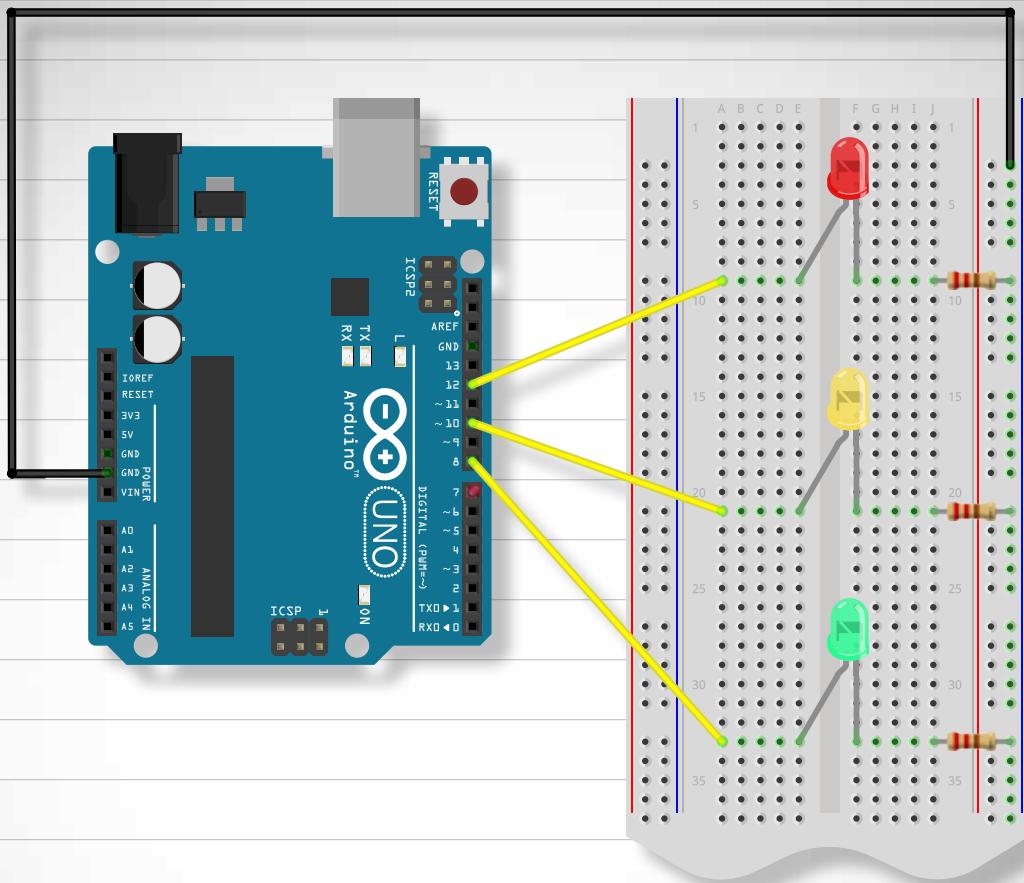


Exercicio 2.- Semáforo

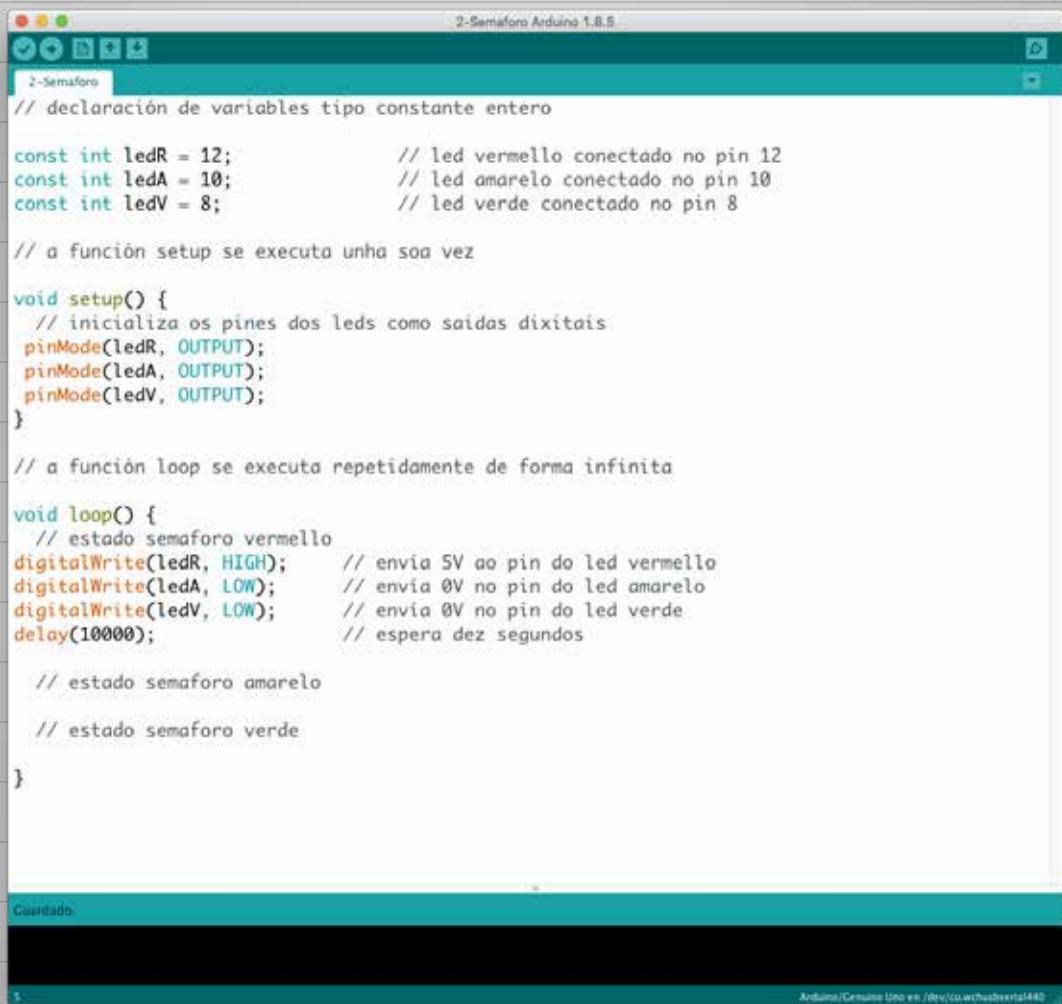
Simular un semáforo de coches (vermello, amarelo e verde) que funcione de forma cíclica.

Montaxe

Debemos conectar 3 leds coas súas resistencias de protección aos pines 12, 10 e 8.



Código



```
2-Semáforo Arduino 1.8.5

// declaración de variables tipo constante entero

const int ledR = 12;           // led vermello conectado no pin 12
const int ledA = 10;           // led amarelo conectado no pin 10
const int ledV = 8;            // led verde conectado no pin 8

// a función setup se executa unha soa vez

void setup() {
    // inicializa os pines dos leds como saídas dixitais
    pinMode(ledR, OUTPUT);
    pinMode(ledA, OUTPUT);
    pinMode(ledV, OUTPUT);
}

// a función loop se executa repetidamente de forma infinita

void loop() {
    // estado semáforo vermello
    digitalWrite(ledR, HIGH);    // envia 5V ao pin do led vermello
    digitalWrite(ledA, LOW);     // envia 0V no pin do led amarelo
    digitalWrite(ledV, LOW);     // envia 0V no pin do led verde
    delay(10000);               // espera dez segundos

    // estado semáforo amarelo

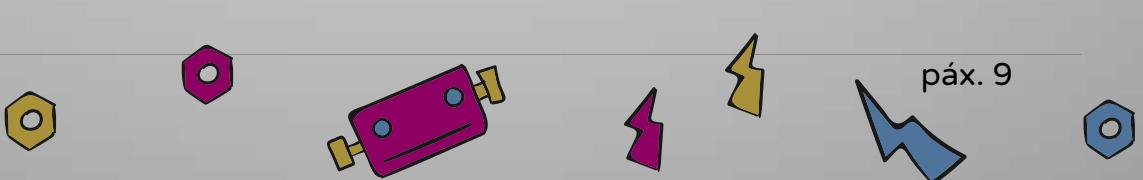
    // estado semáforo verde
}
```

Guardado.

Arduino/Cerebro Usado en /dev/cu.usbserial-1440

E agora ti.....

1. Completa o código co semáforo en amarelo e verde
2. Engade un semáforo de peóns (leds vermello e verde) sincronizado co de coches

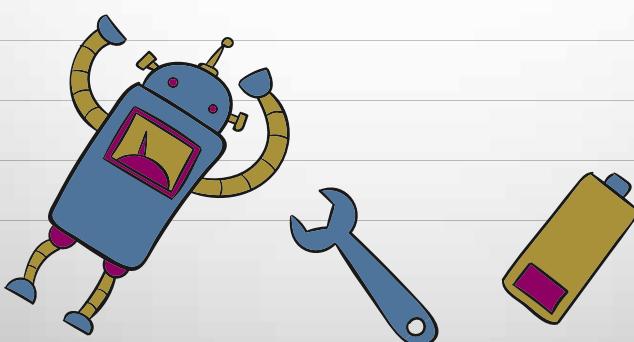
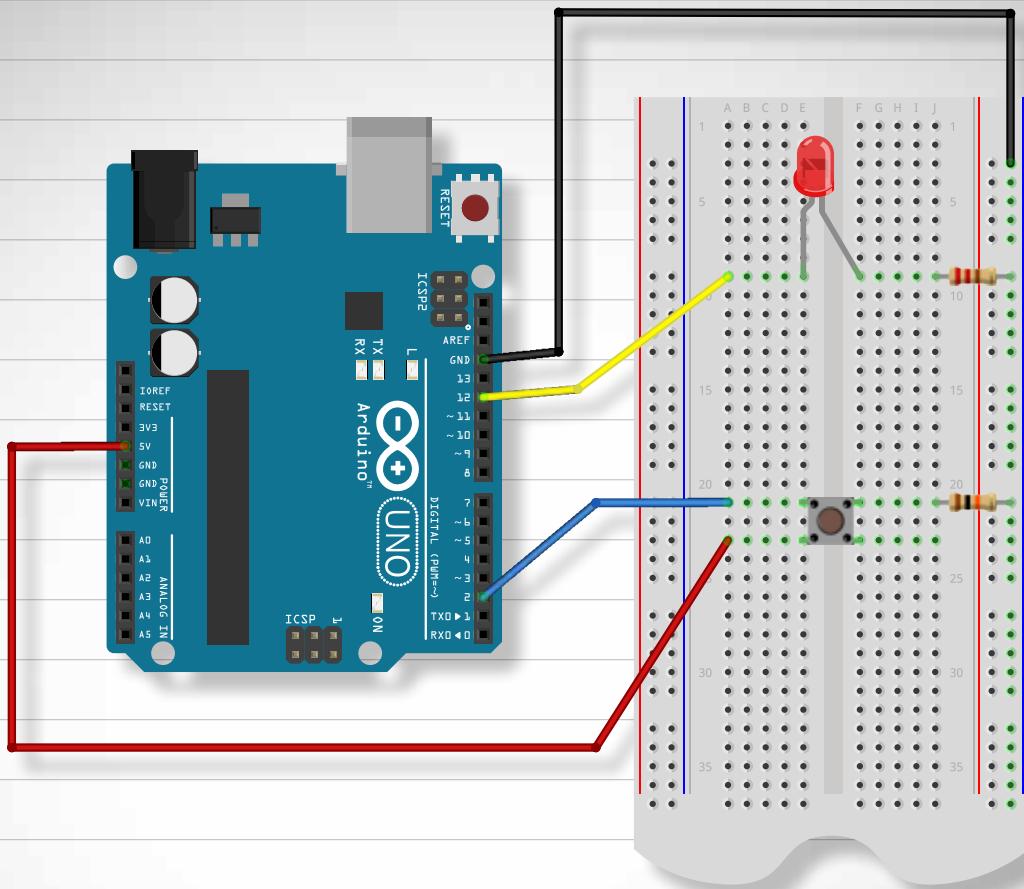


Exercicio 3.- Pulsador e Led

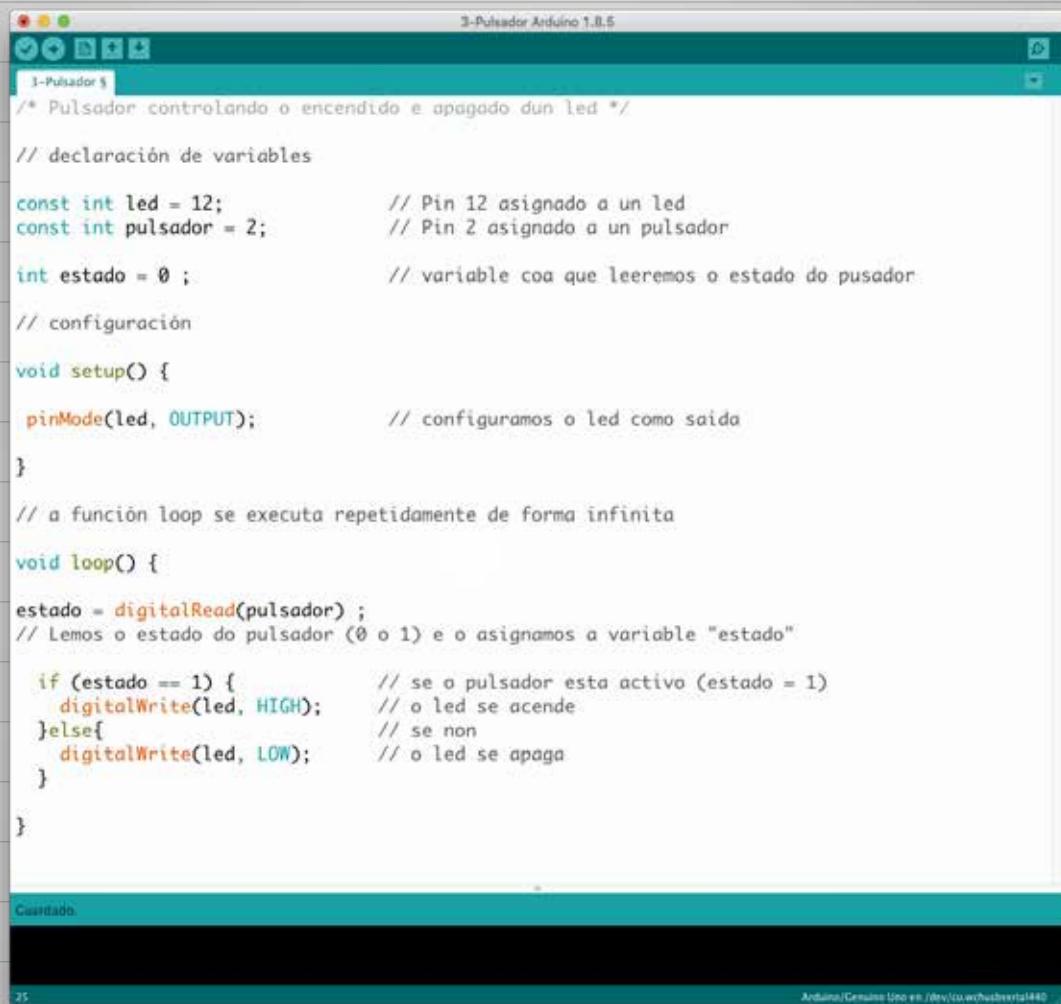
Controlar o aceso e apagado dun LED cun pulsador, de forma que se acenda ou se apague cando presionamos o botón.

Montaxe

Debemos conectar un pulsador ao pin 2 cunha resistencia de 10 kΩ como aparece no esquema.



Código



```
3-Pulsador Arduino 1.8.5

3-Pulsador
/*
 * Pulsador controlando o encendido e apagado dun led *
 */

// declaración de variables

const int led = 12;           // Pin 12 asignado a un led
const int pulsador = 2;        // Pin 2 asignado a un pulsador

int estado = 0;               // variable coa que leeremos o estado do pulsador

// configuración

void setup() {
    pinMode(led, OUTPUT);      // configuramos o led como salida
}

// a función loop se executa repetidamente de forma infinita

void loop() {

estado = digitalRead(pulsador); // Lemos o estado do pulsador (0 o 1) e o asignamos a variable "estado"

if (estado == 1) {             // se o pulsador está activo (estado = 1)
    digitalWrite(led, HIGH);    // o led se acende
} else{                        // se non
    digitalWrite(led, LOW);    // o led se apaga
}

}

Guardado.
```

E agora ti.....

1. Engade outro LED e fai que se acendan de maneira alternativa ao presionar e soltar o pulsador
2. Fai que cada vez que presionamos o pulsador o led acéndase e apáguese dúas veces.
3. Pulsador con memoria; se inicialmente o led estaba acendido, apagase, e se estaba apagado acéndese.



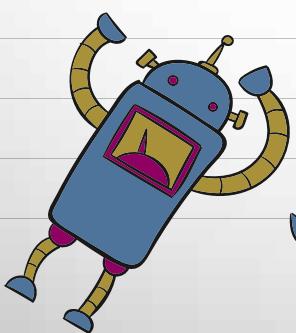
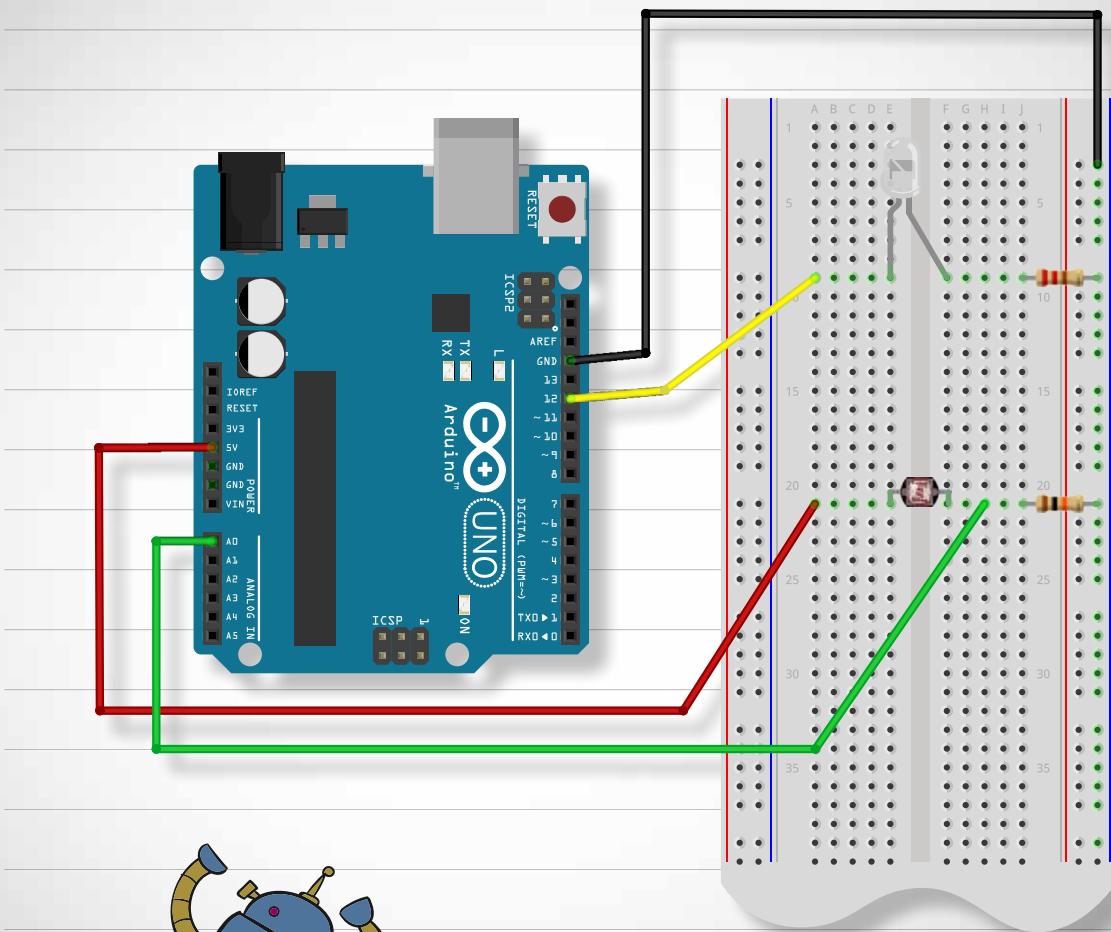
Exercicio 4.- Interruptor de luz

Acender un LED en función da intensidade luminosa que recibe o sensor de luz. En escuridade o LED está aceso e con alta iluminación apagado.

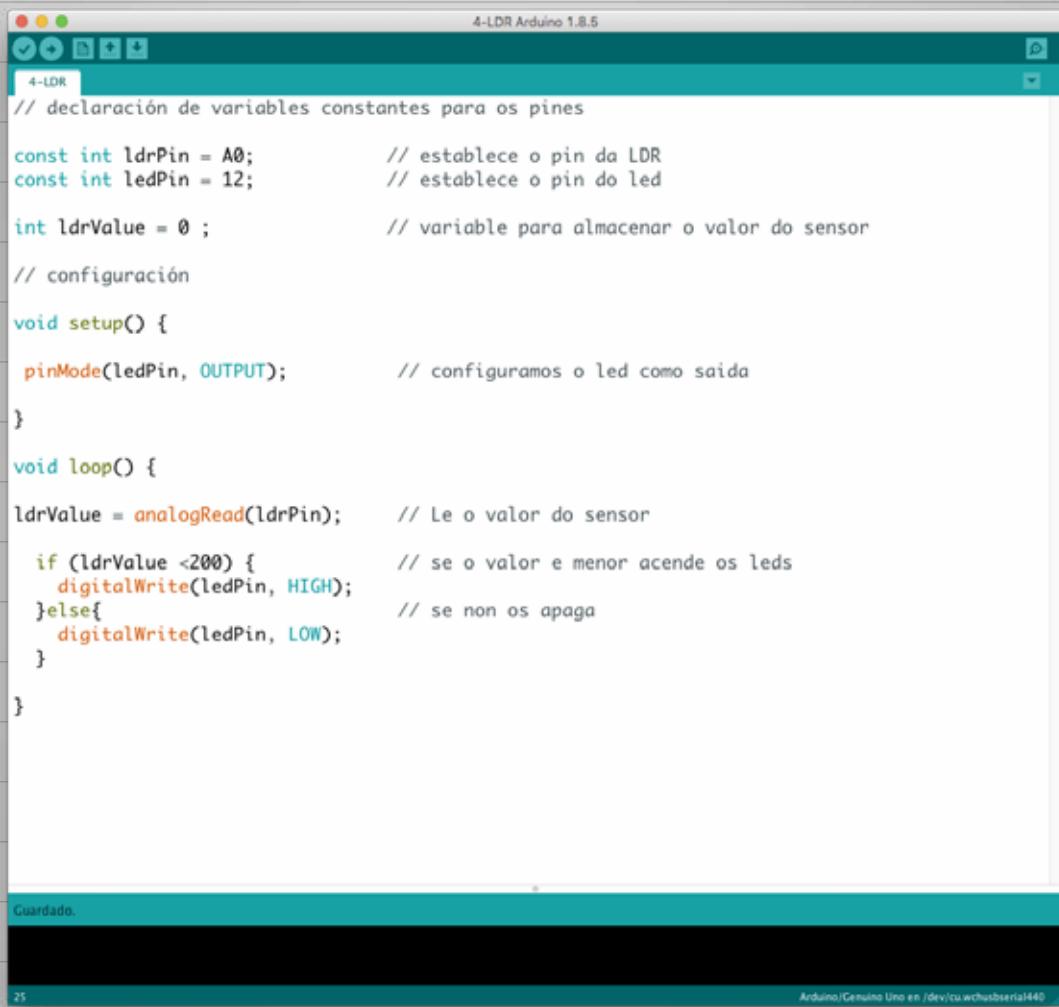
Primeiro debemos ler que valores dános o sensor segundo as diferentes condicións de luz.

Montaxe

Conectar unha LDR ao pin analóxico A0 cunha resistencia de $10\text{ k}\Omega$ tal e como aparece no esquema.



Código



The screenshot shows the Arduino IDE interface with the title bar "4-LDR" and version "4-LDR Arduino 1.8.5". The code is written in C++ and controls an LDR sensor connected to pin A0 and an LED connected to pin 12. It reads the analog value from the LDR and turns the LED on or off based on the value.

```
// declaración de variables constantes para os pines

const int ldrPin = A0;           // establece o pin da LDR
const int ledPin = 12;          // establece o pin do led

int ldrValue = 0 ;              // variable para almacenar o valor do sensor

// configuración

void setup() {
    pinMode(ledPin, OUTPUT);    // configuramos o led como salida
}

void loop() {
    ldrValue = analogRead(ldrPin); // Le o valor do sensor

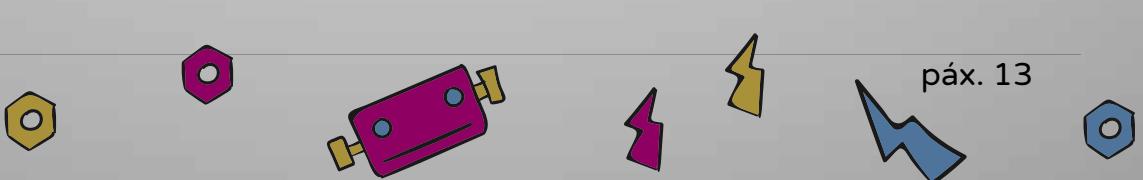
    if (ldrValue <200) {         // se o valor e menor acende os leds
        digitalWrite(ledPin, HIGH);
    }else{                      // se non os apaga
        digitalWrite(ledPin, LOW);
    }
}
```

Guardado.

25 Arduino/Genuino Uno en /dev/cu.wchusbserial1440

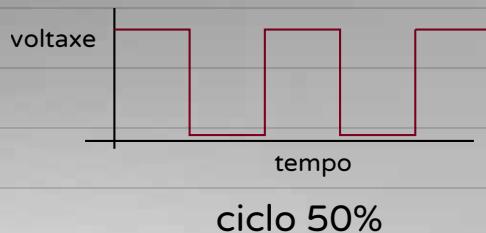
E agora ti.....

1. Engade dous leds más, de forma que crees unha escala luminosa, con moita luz poden estar todos apagados e a medida que diminúa a intensidade luminosa vanse acendendo máis leds.
2. Na proposta anterior, fai que os leds parpadeen cando a intensidade luminosa é moi baixa.



Exercicio 5.- Sinais PWM

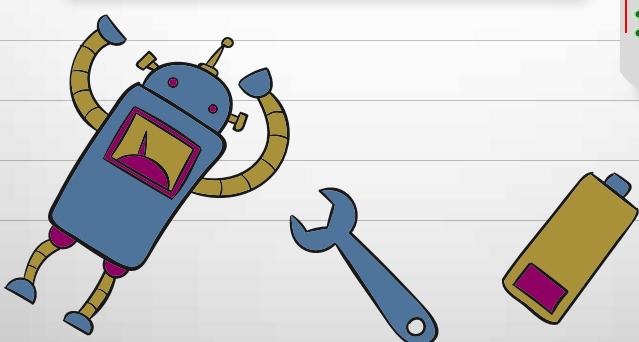
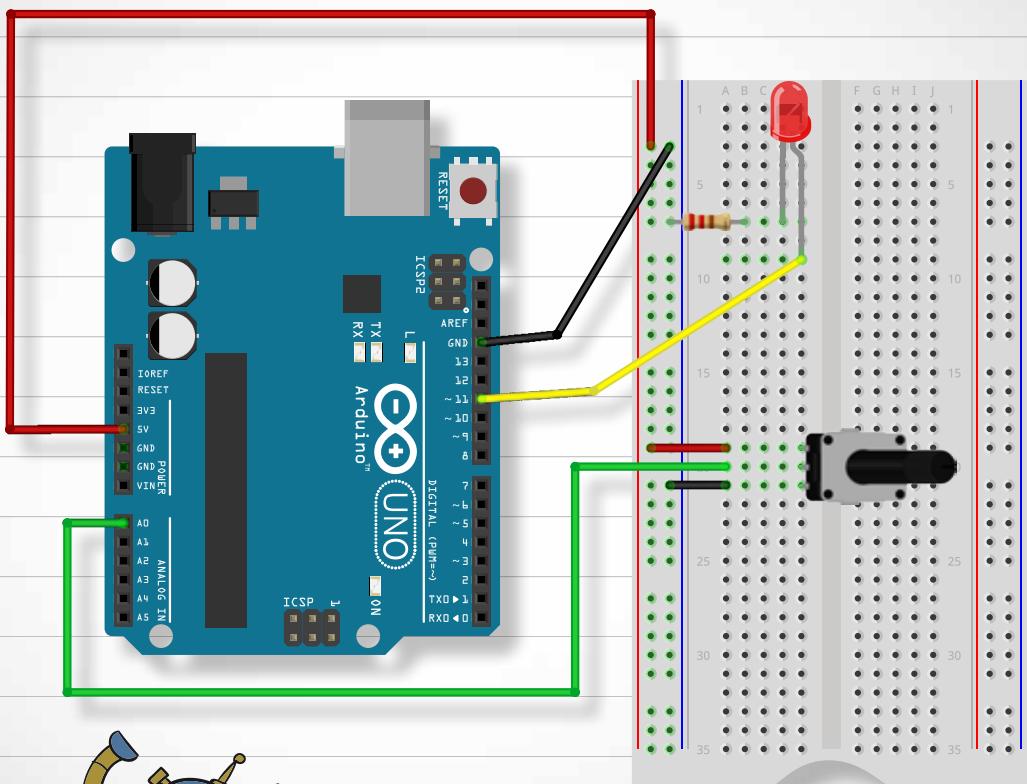
Imos controlar a luminosidade dun LED enviando unha sinal PWM que varía segundo a lectura analóxica dun potenciómetro.



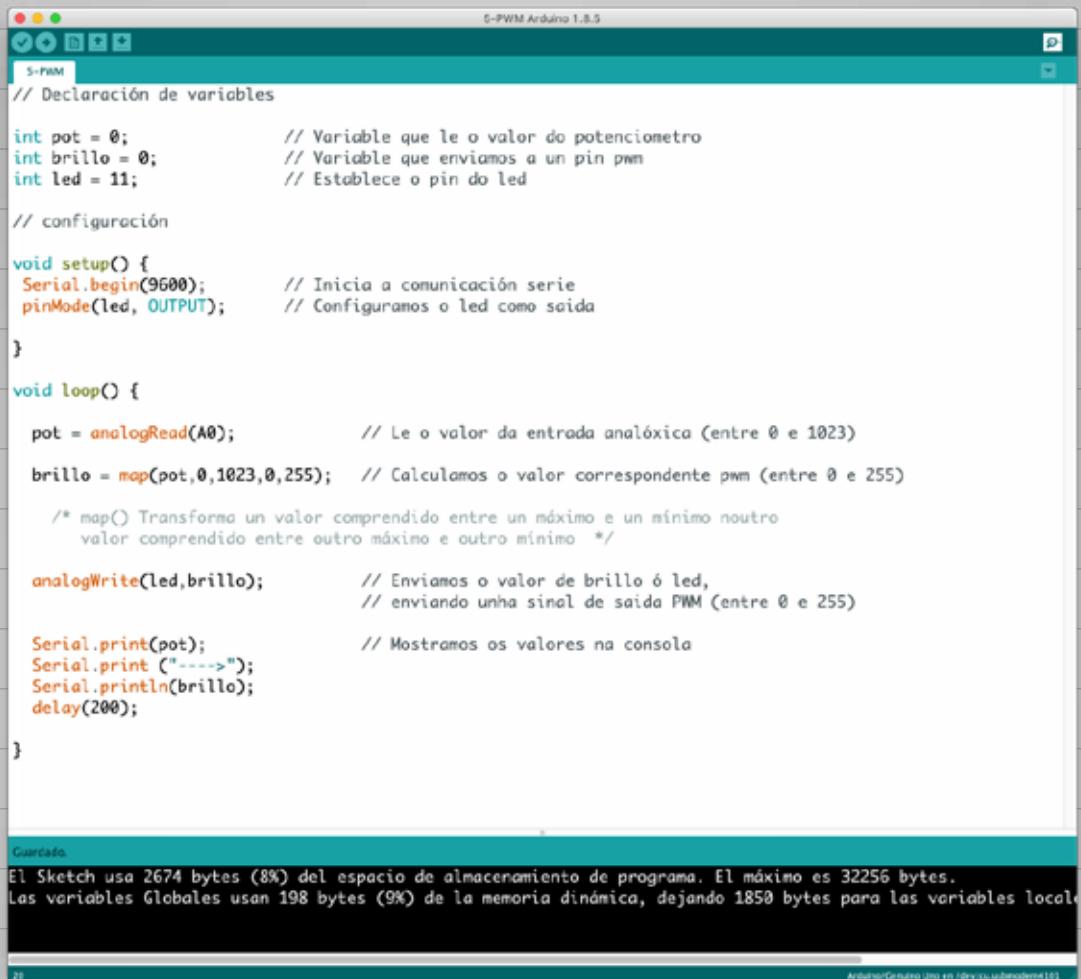
Montaxe

Debemos conectar un potenciómetro a unha entrada analóxica e realizar a lectura segundo a posición do cursor. Os valores lidos irán desde 0 a 1023.

Conectaremos un LED coa súa resistencia de protección a unha saída dixital PWM (~). Enviarémoslle un valor entre 0 e 255, obtendo diferentes niveis de brillo.



Código



```
5-PWM
// Declaración de variables
int pot = 0;           // Variable que le o valor do potenciómetro
int brillo = 0;         // Variable que enviamos a un pin pwm
int led = 11;          // Establece o pin do led

// configuración

void setup() {
  Serial.begin(9600);      // Inicia a comunicación serie
  pinMode(led, OUTPUT);    // Configuramos o led como saída
}

void loop() {
  pot = analogRead(A0);      // Le o valor da entrada analóxica (entre 0 e 1023)
  brillo = map(pot,0,1023,0,255); // Calculamos o valor correspondente pwm (entre 0 e 255)

  /* map() Transforma un valor comprendido entre un máximo e un mínimo noutro
   valor comprendido entre outro máximo e outro mínimo */
  analogWrite(led,brillo);    // Enviamos o valor de brillo ó led,
                             // enviando unha sinal de saída PWM (entre 0 e 255)

  Serial.print(pot);         // Mostramos os valores na consola
  Serial.print ("---->");
  Serial.println(brillo);
  delay(200);
}

Guardado.
El Sketch usa 2674 bytes (8%) del espacio de almacenamiento de programa. El máximo es 32256 bytes.
Las variables Globales usan 198 bytes (9%) de la memoria dinámica, dejando 1850 bytes para las variables locales.
```

E agora ti.....

1. Simula mediante un LED o efecto de lume. Podes xerar un brillo aleatorio e un tempo de espera aleatorio para un LED. Podes utilizar o operador **random**, tanto para o brillo como para o tempo. **brillo = random(20,255);**

2. Fai un programa no que un LED parpadee a diferente frecuencia en función da posición do potenciómetro. Por exemplo, para un valor 0 no potenciómetro o parpadeo realizase con 100 ms de intervalo e para un valor de 1023 do potenciómetro o intervalo é de 1000 ms. Utiliza a función **map** para transformar o intervalo analóxico no intervalo de tempo.



Exercicio 6.- Bucles

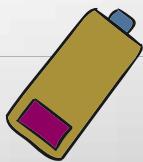
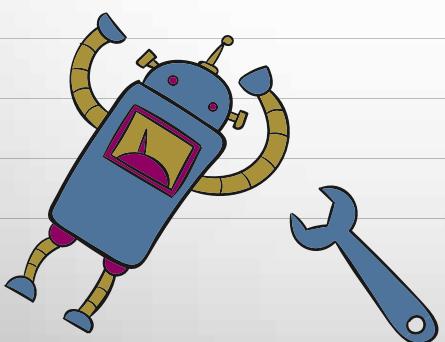
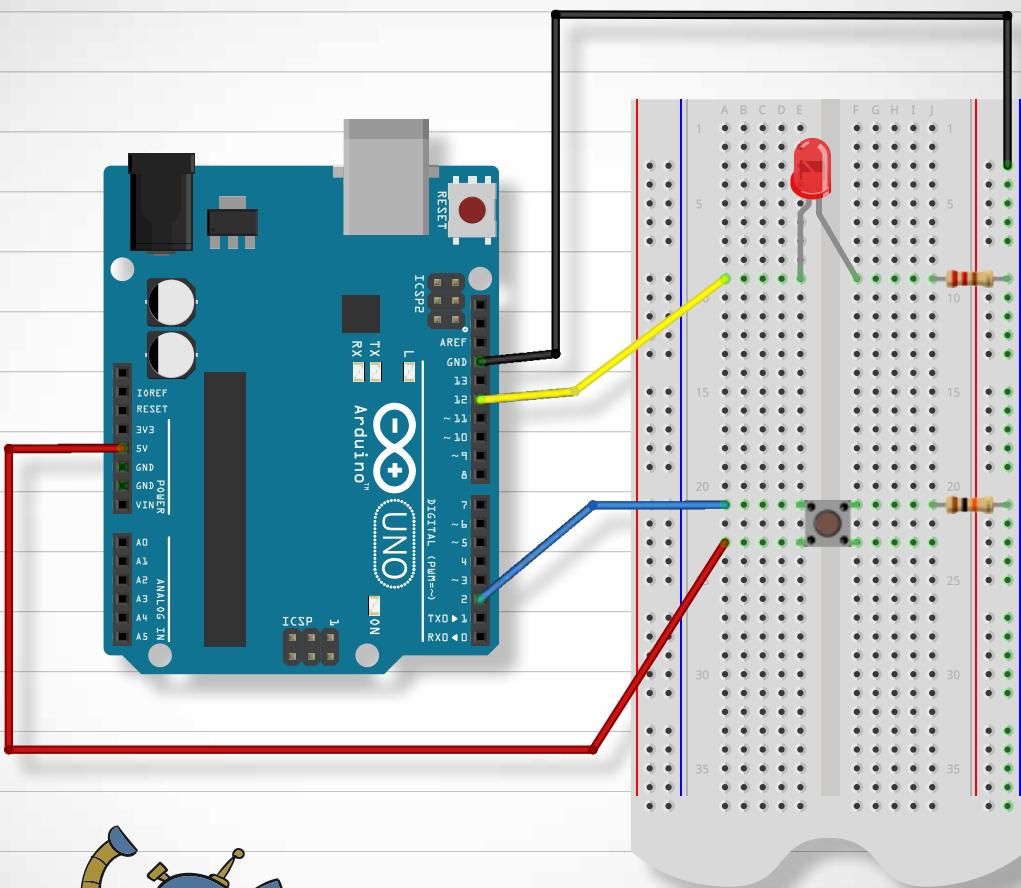
Neste exemplo faremos que unhas instrucións repítanse un número determinado de veces.

Por exemplo, faremos que un LED acéndase e apague varias veces, esperamos un tempo e repetimos o proceso.

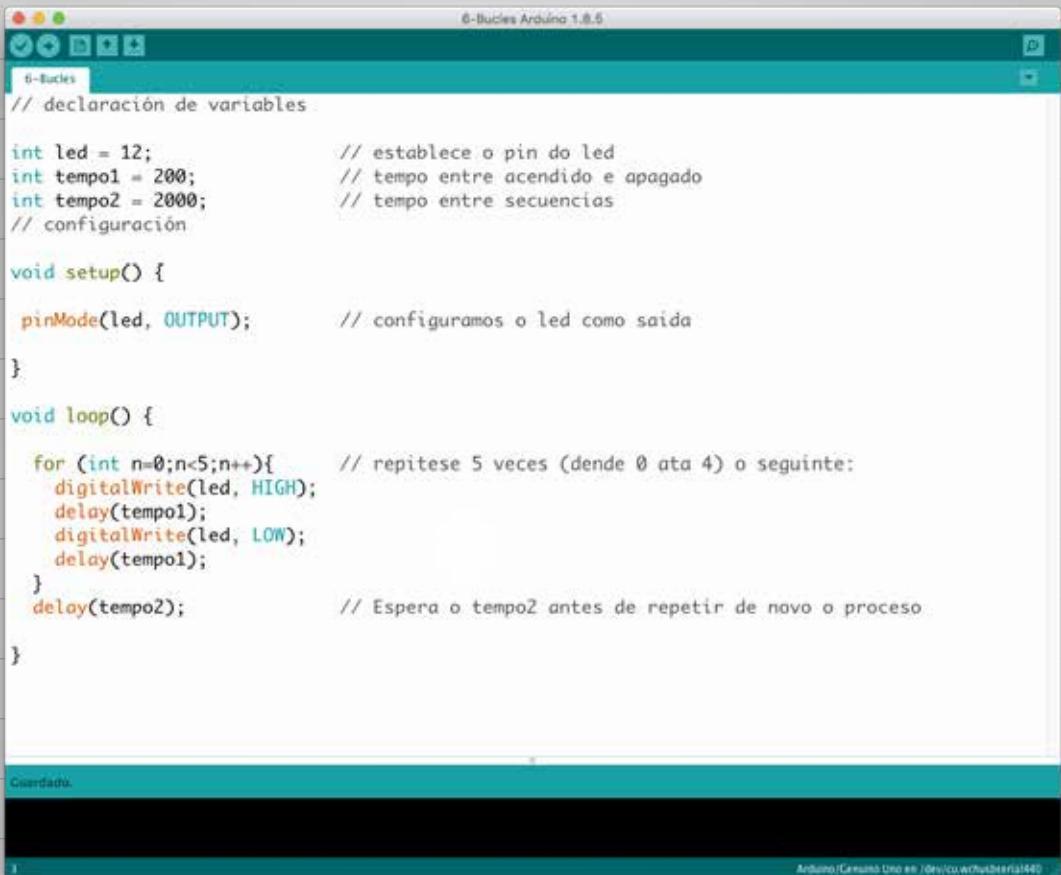
Montaxe

Necesitamos un LED e un pulsador coas súas resistencias de protección. Para este caso utilizamos a función:

`for (inicio,condición, modificador) { proceso }`



Código



```
6-Bucleo Arduino 1.8.5
// declaración de variables

int led = 12;           // establece o pin do led
int tempo1 = 200;        // tempo entre acendido e apagado
int tempo2 = 2000;       // tempo entre secuencias
// configuración

void setup() {
    pinMode(led, OUTPUT); // configuramos o led como saída
}

void loop() {
    for (int n=0;n<5;){ // repite 5 veces (dende 0 ata 4) o seguinte:
        digitalWrite(led, HIGH);
        delay(tempo1);
        digitalWrite(led, LOW);
        delay(tempo1);
    }
    delay(tempo2); // Espera o tempo2 antes de repetir o novo o proceso
}

```

Guardado.

Arduino/Genuino Uno en JDE/luwahdheria2440

E agora ti.....

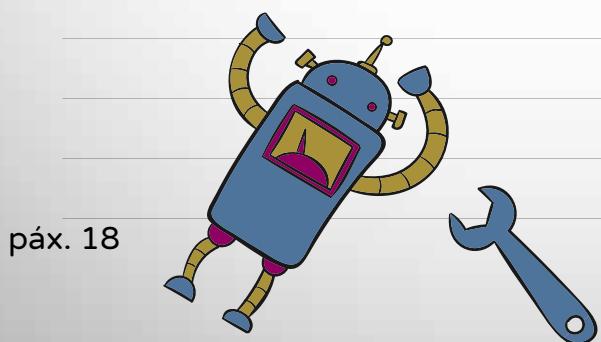
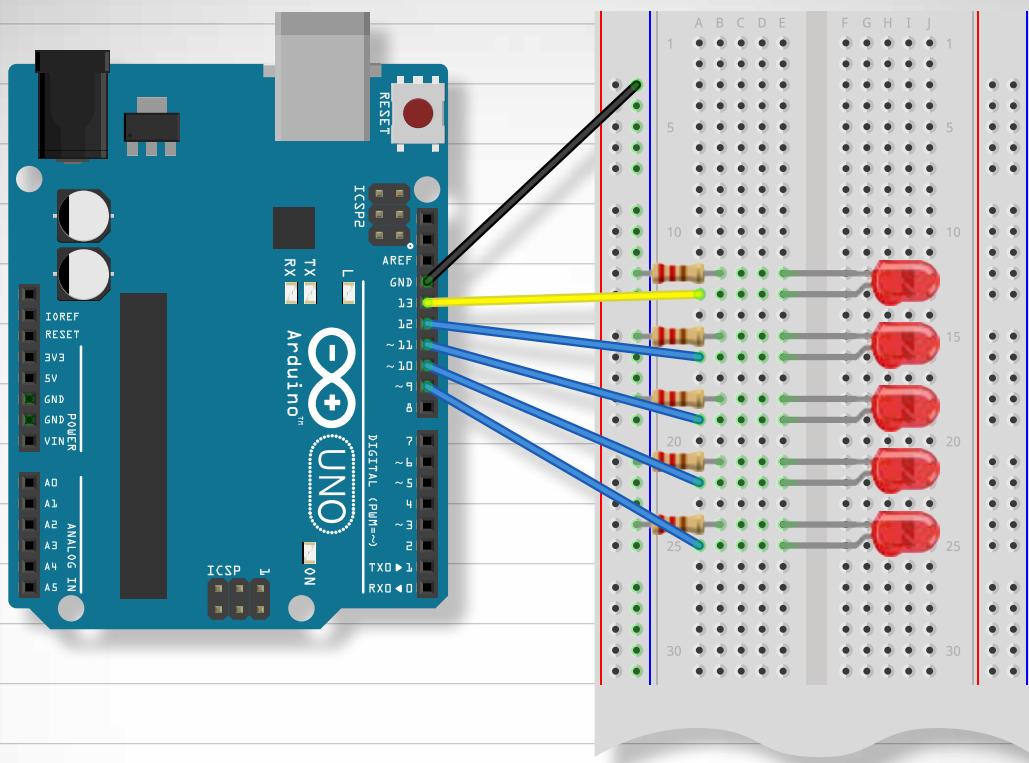
1. Conecta un segundo LED e fai que en cada ciclo o primeiro LED parpadee 6 veces e o segundo LED tres veces.
2. Fai que cada vez que presionamos un pulsador un LED acéndase e apáguese tres veces. Cando o pulsador non está activado o LED permanece apagado.

Exercicio 7.- O coche fantástico

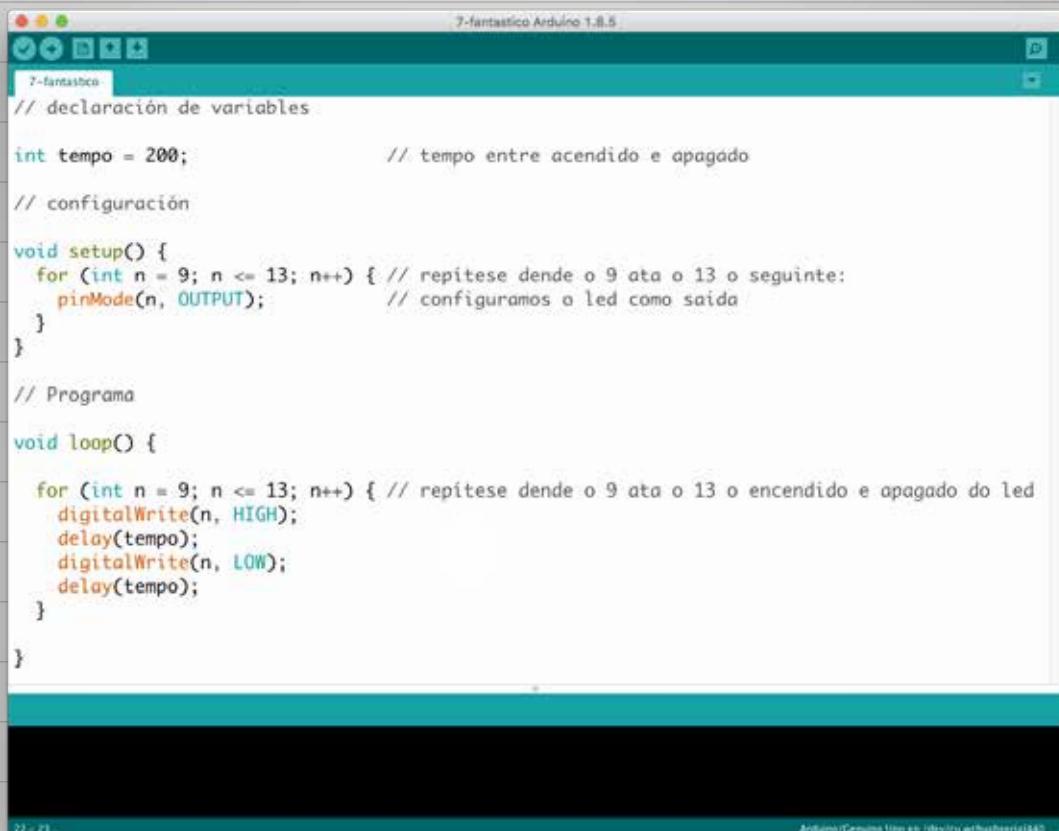
Simularemos o coche fantástico ordenando que se acendan e apáguese unha serie de LEDs un a un de forma consecutiva.

Montaxe

Debemos conectar 5 LEDs coas súas resistencias de protección nos pinos de 9 a 13.



Código



The screenshot shows the Arduino IDE interface with a sketch titled "7-fantastico". The code is as follows:

```
// declaración de variables

int tempo = 200; // tiempo entre encendido e apagado

// configuración

void setup() {
    for (int n = 9; n <= 13; n++) { // repítese dende o 9 ata o 13 o seguinte:
        pinMode(n, OUTPUT); // configuramos o led como saída
    }
}

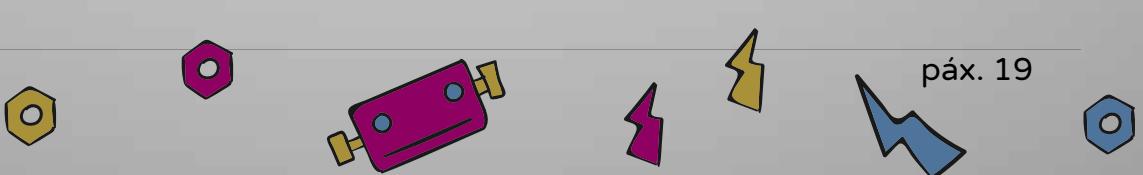
// Programa

void loop() {
    for (int n = 9; n <= 13; n++) { // repítese dende o 9 ata o 13 o encendido e apagado do led
        digitalWrite(n, HIGH);
        delay(tempo);
        digitalWrite(n, LOW);
        delay(tempo);
    }
}
```

The code initializes pins 9 through 13 as outputs. In the loop, it toggles each LED on pin sequentially, with a 200ms delay between each toggle. The Arduino/Genuino Uno is connected to port /dev/cu.usbserial-14D0.

E agora ti.....

1. Engade un bucle descendente ao programa anterior de forma que a secuencia prodúzase primeiro nun sentido e despois no outro.
2. Fai que os LEDs váianse acendendo consecutivamente sen apagarse e, despois, que se vaian apagando un a un, no mesmo sentido ou no contrario.
3. Podemos controlar a velocidade da secuencia cun potenciómetro conectado a unha entrada analólica. Utiliza a función map para transformar o rango analóxico (de 0 a 1023) a un rango de tempos (por exemplo de 50 ms a 1000 ms).





Patrocinado por:



Entidades Colaboradoras:



www.amiguslabs.org

