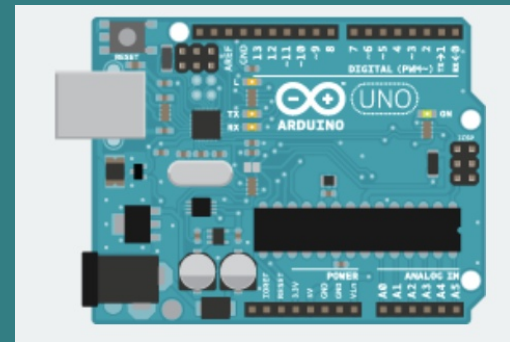


GUÍA DE ARDUINO

Lecciones y prácticas



Francisco Trigueros

Índice de contenidos

PRIMERAS LECCIONES Y PRÁCTICAS CON ARDUINO.....	3
1. Primeros pasos con Arduino.....	3
2. Protoboard y conexión de componentes.....	4
3. Variables.....	5
4. Repeticiones con el bucle for().....	6
5. Graduar la intensidad de luz de un LED. Salidas PWM.....	7
6. Comunicación serial. Recibiendo información de Arduino.....	10
7. Comunicación serial. Enviando información a Arduino.....	11
8. Entradas digitales. Pulsadores. OR, AND, if, else.....	13
9. Entradas analógicas. Sensores de luz con LDR.....	17
10. Servomotores.....	19
11. Potenciómetro. Función map().....	21
PROGRAMACIÓN C++ (PROFUNDIZACIÓN).....	23
12. Tipos de variables más usados.....	23
13. Variables para frases y caracteres.....	25
14. Variables globales y variables locales.....	26
15. Leer enteros, decimales y texto en el monitor serie.....	26
16. Formas de declarar una constante.....	26
MÁS SENSORES Y ACTUADORES PARA PROYECTOS.....	27
17. Sensor ultrasónico de distancia HC-SR04.....	27
18. Pantalla LCD1602 I2C.....	29
19. Sensor de temperatura y humedad DHT11.....	32
20. Sensor infrarrojo pasivo HC-SR501. Detector movimiento.....	33
21. Relé.....	36
22. Tira de LEDs NeoPixel WS2812B.....	37
23. Sensores de sonido (de señal digital y de señal analógica).....	41

24. Distintas formas de alimentar Arduino.....	41
25. Controlador de Motores DC L298N.....	41
26. Motores de corriente continua.....	43
27. Bluetooth. Módulo HC-05.....	44



Esta licencia permite a los reutilizadores distribuir, remezclar, adaptar y crear a partir del material en cualquier medio o formato, siempre que se cite al creador. La licencia permite el uso comercial. Si remezclas, adaptas o construyes a partir del material, debes licenciar el material modificado bajo idénticos términos. CC BY-SA incluye los siguientes elementos:



BY: debe darse crédito al creador.

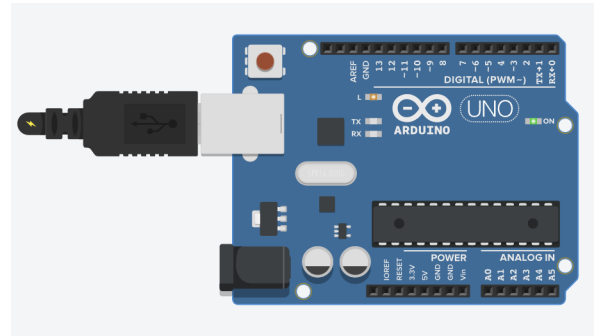


SA: Las adaptaciones deben compartirse bajo los mismos términos.

PRIMERAS LECCIONES Y PRÁCTICAS CON ARDUINO

1. Primeros pasos con Arduino.

Comienza visualizando el vídeo del enlace mientras practicas con el kit de Arduino o el simulador de circuitos online [TinkerCAD](#). De momento no tienes que entregar ninguna tarea.



 **VÍDEO:** Primer programa con Arduino IDE y simulador de TinkerCAD

Ejemplo de programa básico. Encender y apagar el LED integrado:

```
void setup()
{
  pinMode(13,OUTPUT);    //Configuramos el pin 13 como salida
}

void loop()
{
  digitalWrite(13,HIGH); //Activamos la salida de 5V en el pin 13
  delay(300);           //Definimos un retardo de 300 milisegundos
  digitalWrite(13,LOW); //Desactivamos la salida de corriente en el pin 13
  delay(300);
}
```

2. Protoboard y conexión de componentes

Visualiza el siguiente vídeo:



2-Conectar un LED a Arduino

Tarea 2-1_Parpadeo_LED

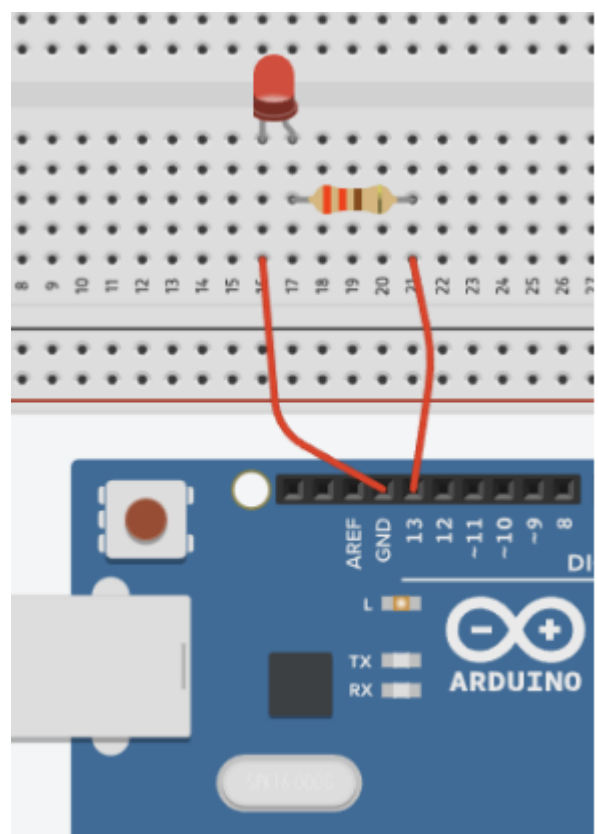
Monta el circuito de la figura y realiza el programa:

- Encender y apagar el LED 4 veces estando encendido 400 ms y apagado 250 ms
- Mantenerse apagado 1 segundo.

Después prueba a ir cambiando los tiempos de encendido y apagado.

Sería el código equivalente a:

```
Bucle
Escribir digital Pin 13 ON
Esperar 400 milisegundos
Escribir digital Pin 13 OFF
Esperar 250 milisegundos
Escribir digital Pin 13 ON
Esperar 400 milisegundos
Escribir digital Pin 13 OFF
Esperar 250 milisegundos
Escribir digital Pin 13 ON
Esperar 400 milisegundos
Escribir digital Pin 13 OFF
Esperar 250 milisegundos
Escribir digital Pin 13 ON
Esperar 400 milisegundos
Escribir digital Pin 13 OFF
Esperar 250 milisegundos
Esperar 1000 milisegundos
```



3. Variables.

Para declarar una variable global de tipo entero (“integer” en inglés) llamada pinLED, usamos el comando:

```
int pinLED = 13;
```



3 - Variables con Arduino IDE

Tarea 3_1-Variables

Realiza el mismo circuito y mismo programa de la tarea anterior, pero usa las siguientes variables:

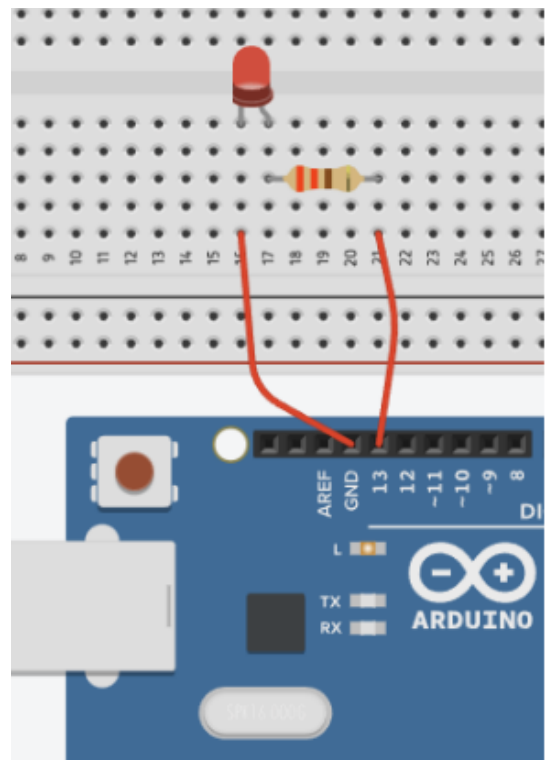
- **pinLed**: número de pin donde está conectado el LED.
- **delayOn**: tiempo que está encendido del LED
- **delayOff**: tiempo que está apagado del LED
- **finalDelay**: tiempo final

Prueba después a ir cambiando el tiempo de encendido y apagado del LED, verás cómo es más rápido ahora usando las variables.

Sería el código equivalente al siguiente programa:

```
Inicializar
Establecer pinLed = 13
Establecer delayOn = 400
Establecer delayOff = 250
Establecer delayFinal = 1000

Bucle
Escribir digital Pin pinLed On
Esperar delayOn milisegundos
Escribir digital Pin pinLed Off
Esperar delayOff milisegundos
Escribir digital Pin pinLed On
Esperar delayOn milisegundos
Escribir digital Pin pinLed Off
Esperar delayOff milisegundos
Escribir digital Pin pinLed On
Esperar delayOn milisegundos
Escribir digital Pin pinLed Off
Esperar delayOff milisegundos
Escribir digital Pin pinLed On
Esperar delayOn milisegundos
Escribir digital Pin pinLed Off
Esperar delayOff milisegundos
Esperar delayFinal milisegundos
```



4. Repeticiones con el bucle for()

4- Bucles for() en Arduino

Ejemplo de repetir 5 veces con el bucle “for()”, que podríamos interpretarlo como: comenzando con $i=1$, repetir mientras i sea menor o igual a 5, aumentando i de uno en uno ($i++$):

```
int i; //creamos la variable i, que se usará como contador
for (i = 1; i <= 5; i++) {
  digitalWrite(13, HIGH);
  delay(200);
  digitalWrite(13, LOW);
  delay(200);
}
```

Si queremos que el contador disminuya de 1 en 1:

```
for(i=254 ; i >=1 ; i--){
```

Si queremos que el contador vaya aumentando de 5 en 5, sería:

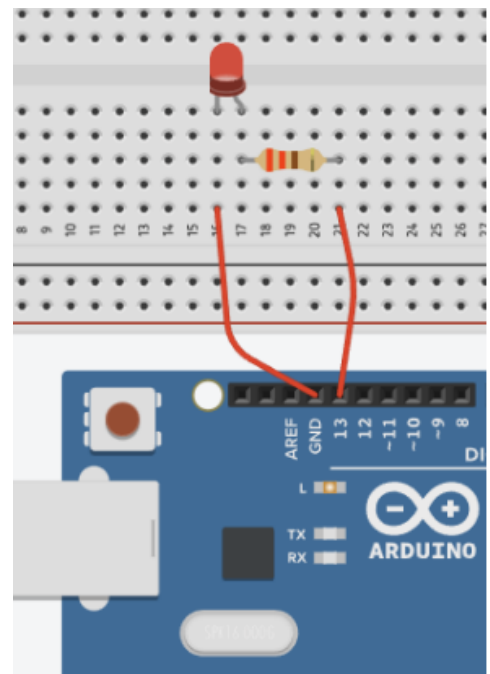
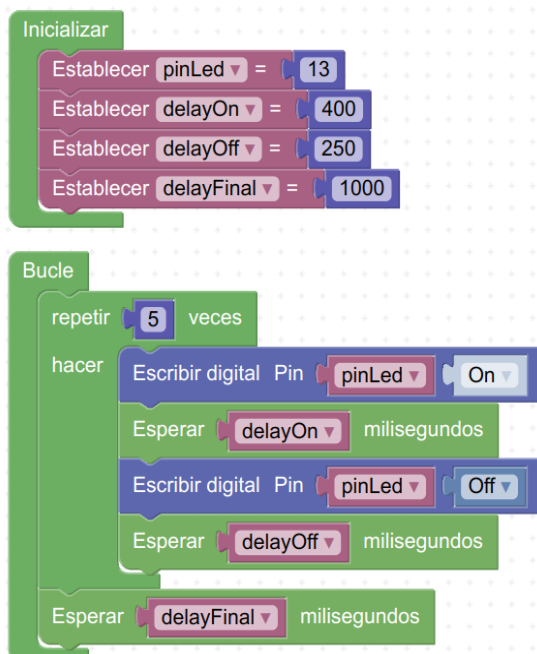
```
for(i=0 ; i <= 255 ; i=i+5) {
```

Tarea 4_1-Bucle_for

Realiza el mismo circuito y el mismo programa que el de las tareas anteriores, pero en vez de repetir las líneas con las mismas instrucciones cuatro veces, usa el bucle “for”.

Prueba a ir cambiando el número de repeticiones y los tiempos de encendido y apagado, verás cómo ahora es más rápido usando las variables y el bucle for.

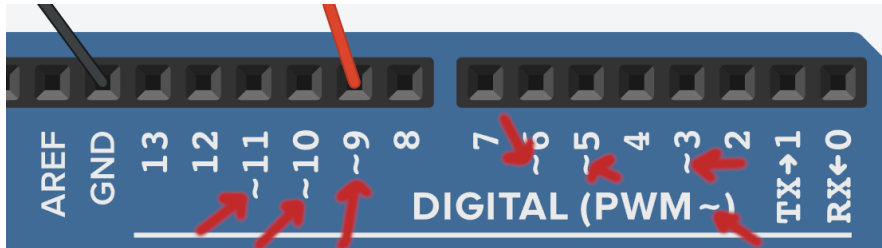
Sería el código equivalente a:



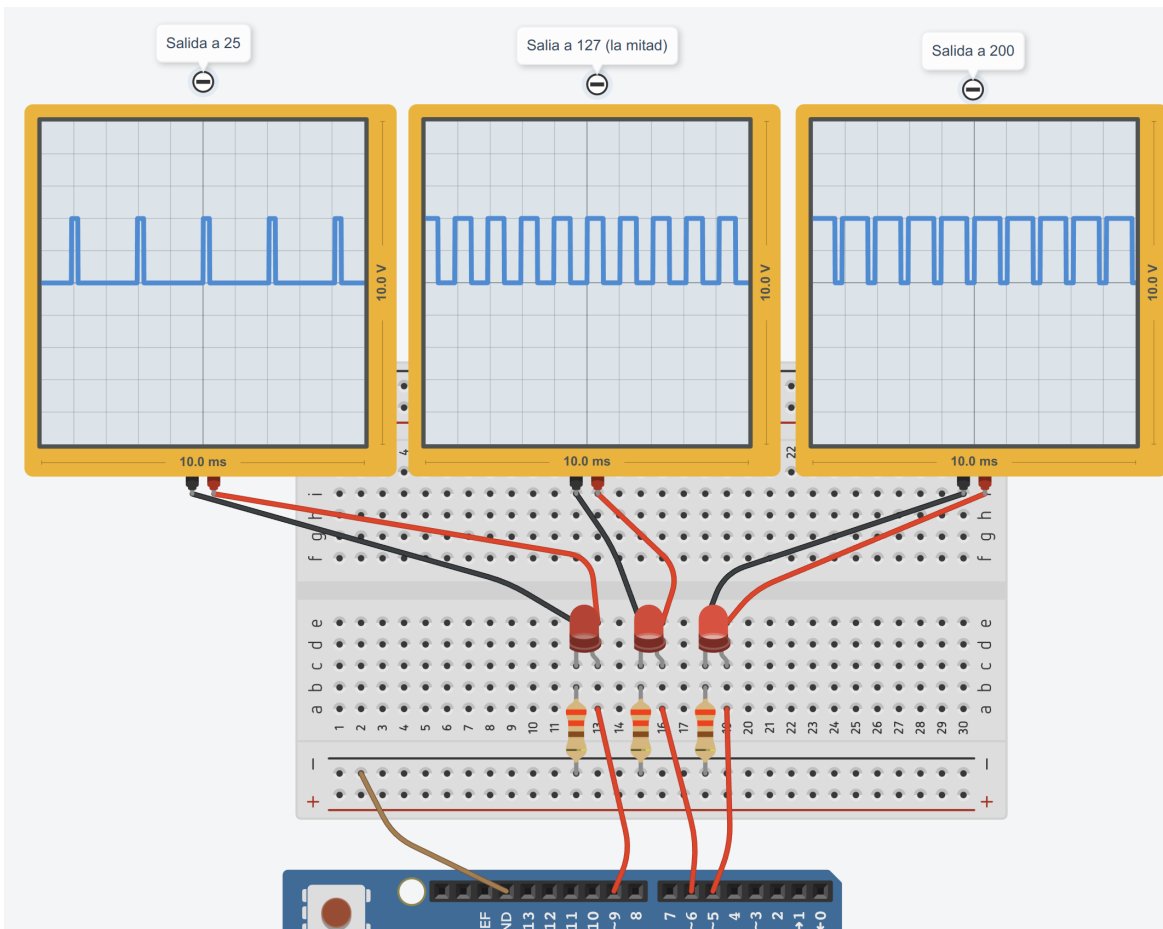
5. Graduar la intensidad de luz de un LED. Salidas PWM.

5- PWM en Arduino para regular la intensidad de brillo de un LED

Nota: no todos los pines digitales se pueden usar con PWM, solamente los que se indican con la “onda”:



PWM: Pulse Wide Modulation (modulación de ancho de pulso). Se utiliza para simular valores de distinta intensidad analógicos mediante pulsos o “parpadeos” de la señal:

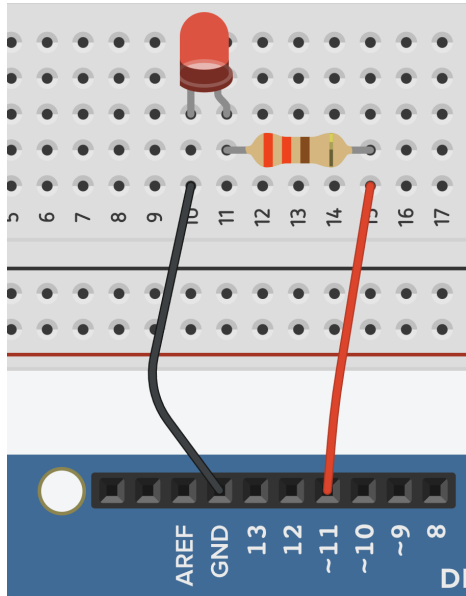


Código para encender un LED conectado en el pin 9 con una intensidad de 25:

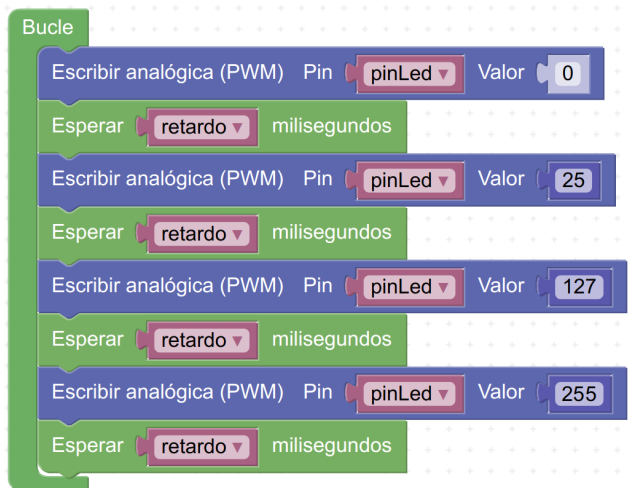
```
analogWrite(9,25);
```

Tarea 5-1-PWM_brillo

Monta el circuito de la figura



y realiza el siguiente programa con el código equivalente a:



6- Arduino: Ajuste de brillo usando la variable "i" del bucle for()

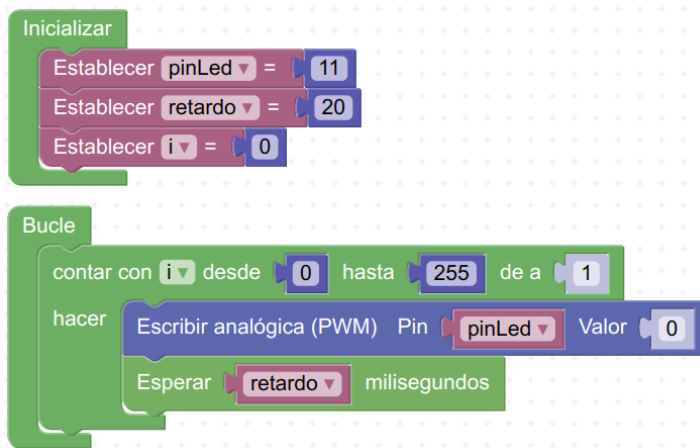
Tarea: 5-2-PWM_for

Usando el bucle "for", programa el LED para que se encienda progresivamente de 0 a 255 con un retardo de 20 ms entre cambios de intensidad. Fíjate en la imagen, te servirá de guía:

```
for (i = 0; i <= 255; i++) {  
  analogWrite(pinLed, i);  
  delay(retardo);  
}
```

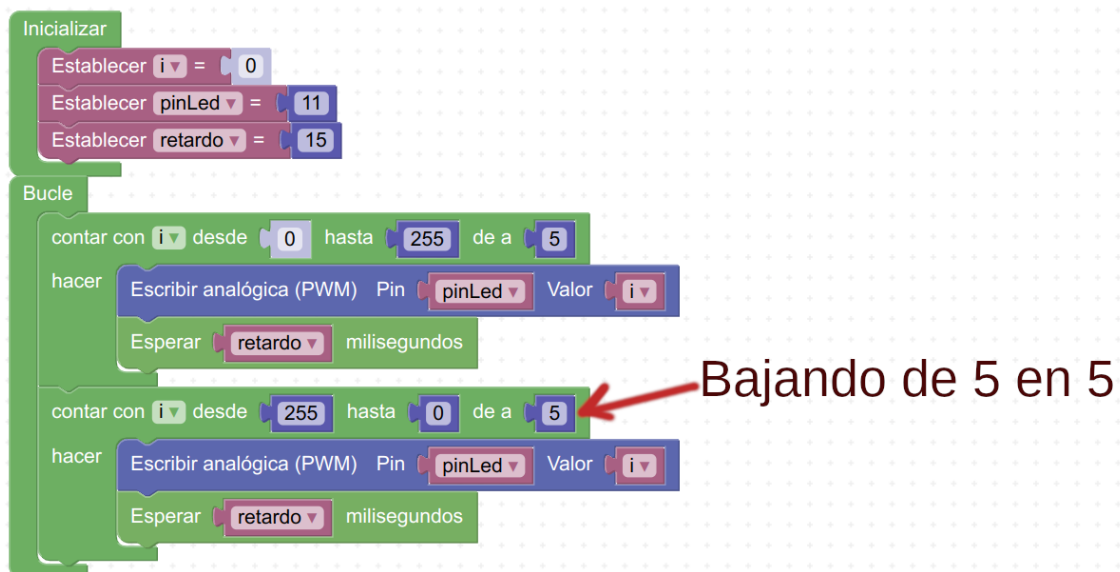
Figura 1: i aumenta de 1 en 1 (i++)

Sería el equivalente en código a los siguientes bloques:



Tarea: 5-3-Brillo_sube_baja

Usando el bucle "for", programa el LED para que se encienda progresivamente de 0 a 255 de 5 en 5 y luego de 255 a 0 de 5 en 5, con un retardo de 15 ms entre cambios de intensidad. Fíjate en la imagen, te servirá de guía:



6. Comunicación serial. Recibiendo información de Arduino.

7- Arduino nos envía mensajes (comunicación serial)

Vamos a comunicarnos con Arduino. Primero vamos a ver cómo Arduino nos va a transmitir información. Dentro de void setup(), debemos iniciar la comunicación serial, indicando la velocidad de transmisión, por defecto 9600 bauds:

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

Dentro de void loop(), el comando para que se imprima en pantalla una línea sería:

```
Serial.println(i); // imprime en una línea el valor de "i"
```

Si queremos que no haya salto de línea, simplemente **Serial.print**:

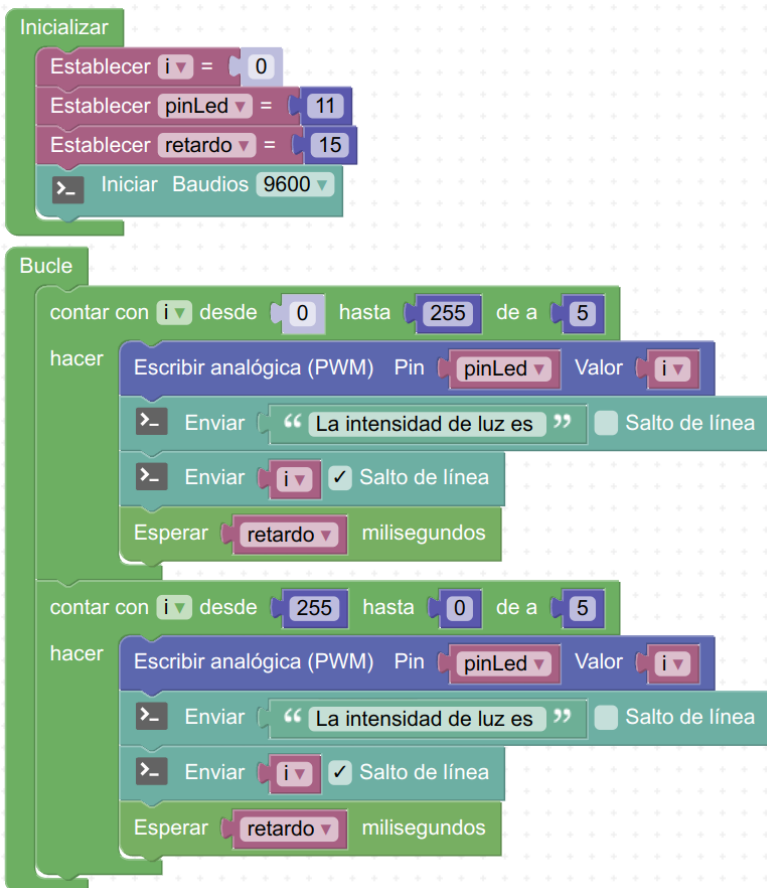
```
Serial.print("El valor de la intensidad es ");  
Serial.println(i);
```

El código anterior nos daría como mensajes en el monitor serie:

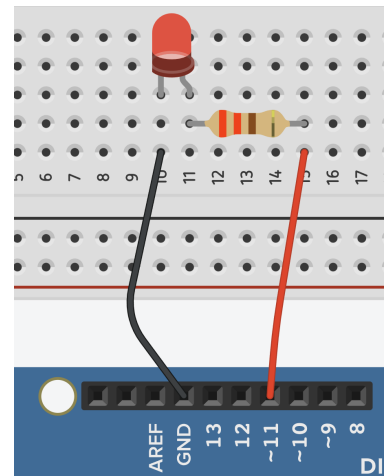
```
El valor de la intensidad es 124  
El valor de la intensidad es 119
```

Tarea 6-1-Visualiza el nivel de brillo

Monta el mismo circuito que el del ejercicio anterior y realiza el programa 6-2-Brillo_sube_baja, pero visualizando la intensidad de brillo en el monitor serie:



```
void setup() {  
  pinMode(11, OUTPUT);  
  Serial.begin(9600);  
}  
  
void loop() {  
  for (int i = 0; i < 255; i += 5) {  
    analogWrite(11, i);  
    Serial.print("La intensidad de luz es ");  
    Serial.println(i);  
    delay(15);  
  }  
  for (int i = 255; i > 0; i -= 5) {  
    analogWrite(11, i);  
    Serial.print("La intensidad de luz es ");  
    Serial.println(i);  
    delay(15);  
  }  
}
```



7. Comunicación serial. Enviando información a Arduino.

8-Comunicación serial: enviando información a Arduino

Para esperar hasta que Arduino reciba nuestro mensaje:

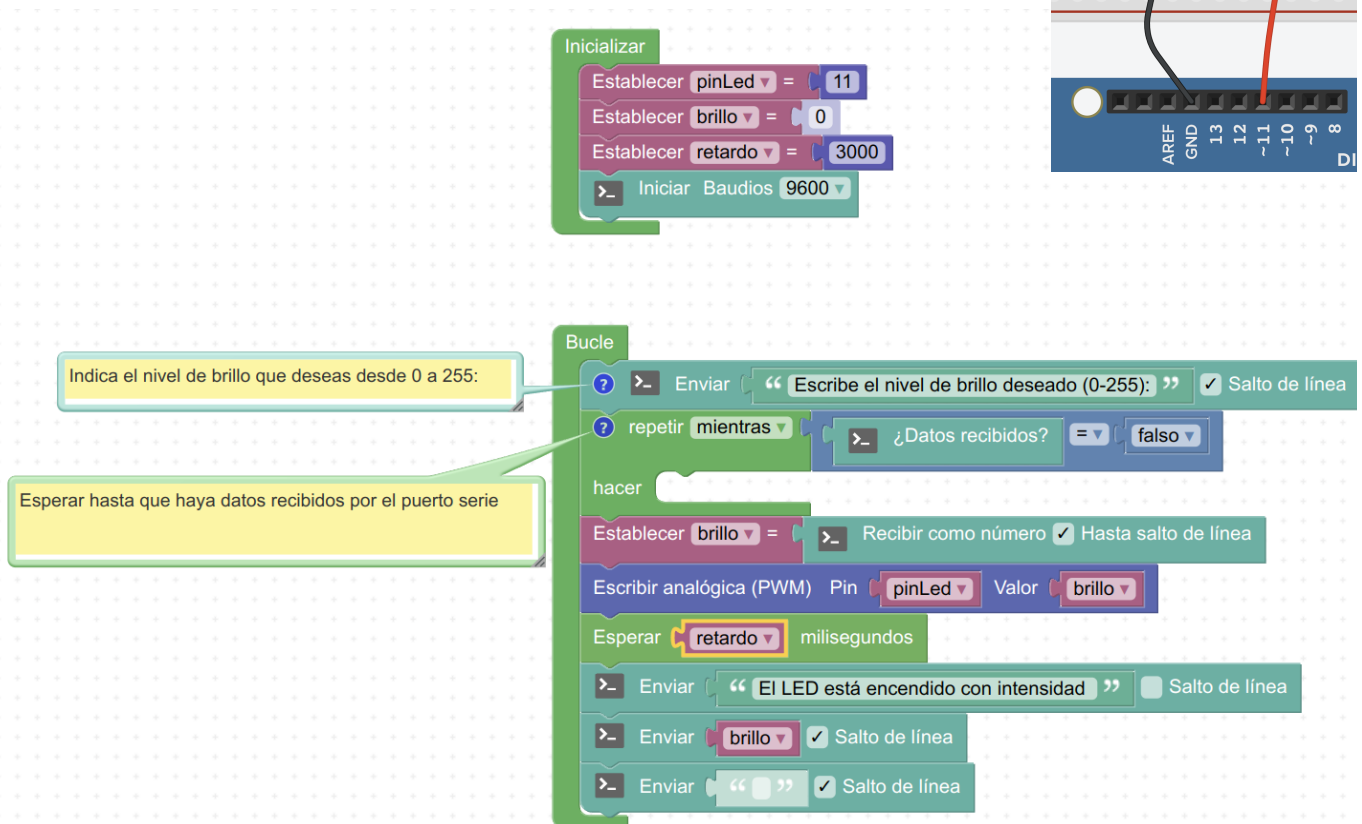
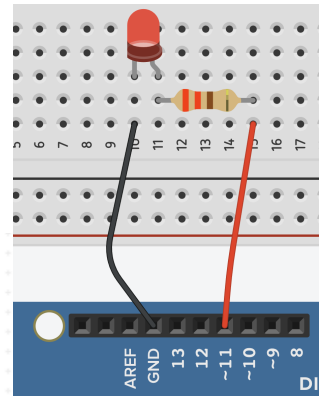
```
while (Serial.available() == 0) { //Mientras no haya respuesta (=0)
  //no hacer nada (en blanco)
}
```

Para guardar en la variable "brillo" el número entero que hemos escrito como respuesta:

```
brillo = Serial.parseInt(); //guarda la respuesta en brillo
```

Tarea 7-1-Pedir_intensidad_de_brillo

Monta el mismo circuito que el del ejercicio anterior y realiza un programa en Arduino IDE que haga el algoritmo de los bloques siguientes:



El programa de Arduino IDE se divide en dos secciones principales:

- Inicializar:**
 - Establecer pinLed = 11
 - Establecer brillo = 0
 - Establecer retardo = 3000
 - Iniciar Baudios 9600
- Bucle:**
 - Enviar "Escribe el nivel de brillo deseado (0-255):" con salto de línea.
 - repetir mientras ¿Datos recibidos? = falso.
 - hacer:
 - Establecer brillo = Recibir como número (con opción "Hasta salto de línea" marcada).
 - Escribir analógica (PWM) Pin pinLed, Valor brillo.
 - Esperar retardo milisegundos.
 - Enviar "El LED está encendido con intensidad" (sin salto de línea).
 - Enviar brillo (con salto de línea).
 - Enviar " " (con salto de línea).

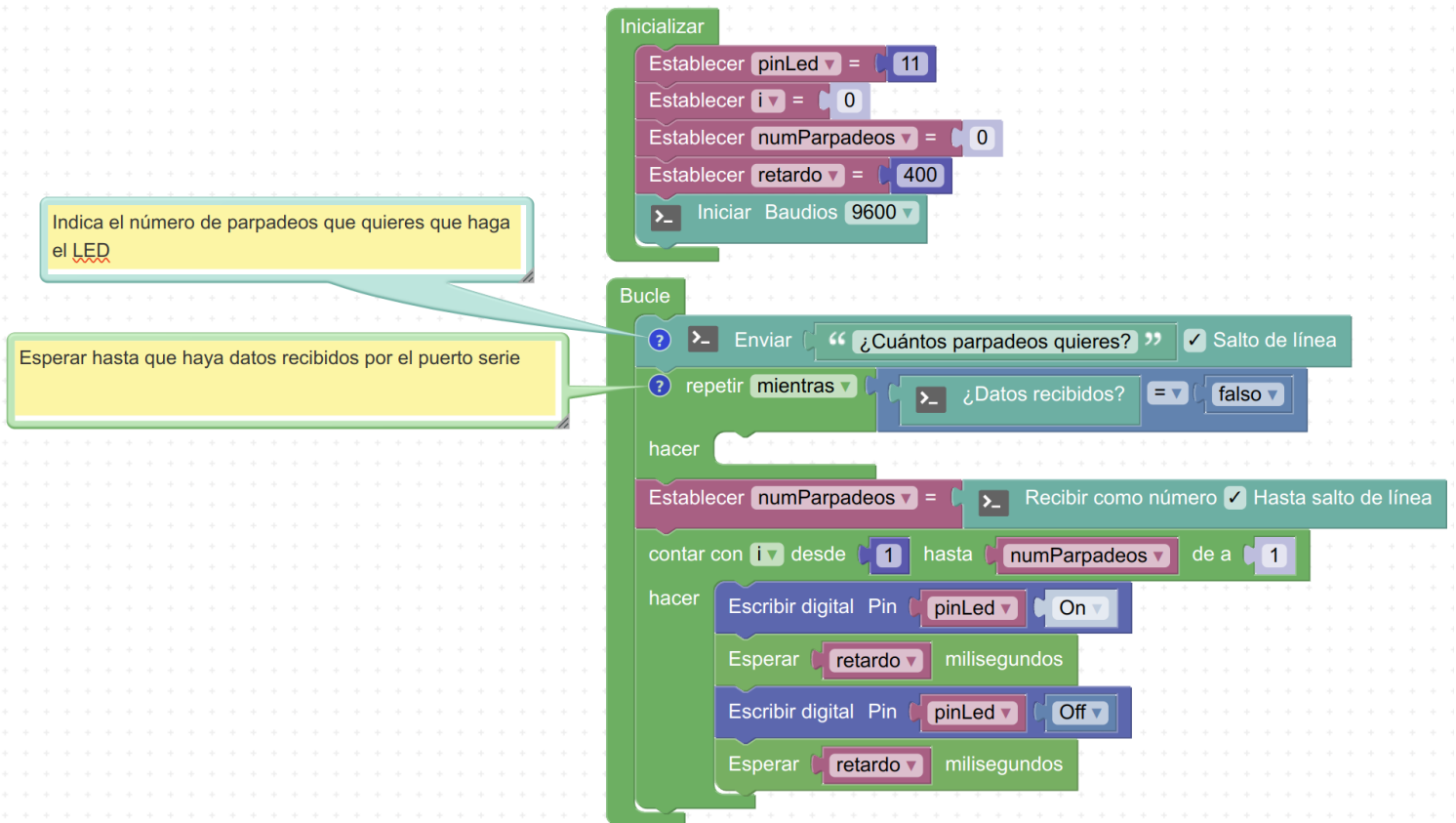
Explicaciones de los bloques de código:

- Indica el nivel de brillo que deseas desde 0 a 255: (señala al bloque de envío de texto).
- Esperar hasta que haya datos recibidos por el puerto serie (señala al bloque de repetición).

Tarea 7-2-Pedir_numero_de_parpadeos

Con el mismo circuito que la práctica anterior, realiza un programa que te pida el número de parpadeos de LED que quieras, una vez que introduzcas el número, el LED parpadeará ese número de veces.

Si necesitas ayuda, puedes guiarte con el siguiente programa de bloques:



Indica el número de parpadeos que quieres que haga el LED

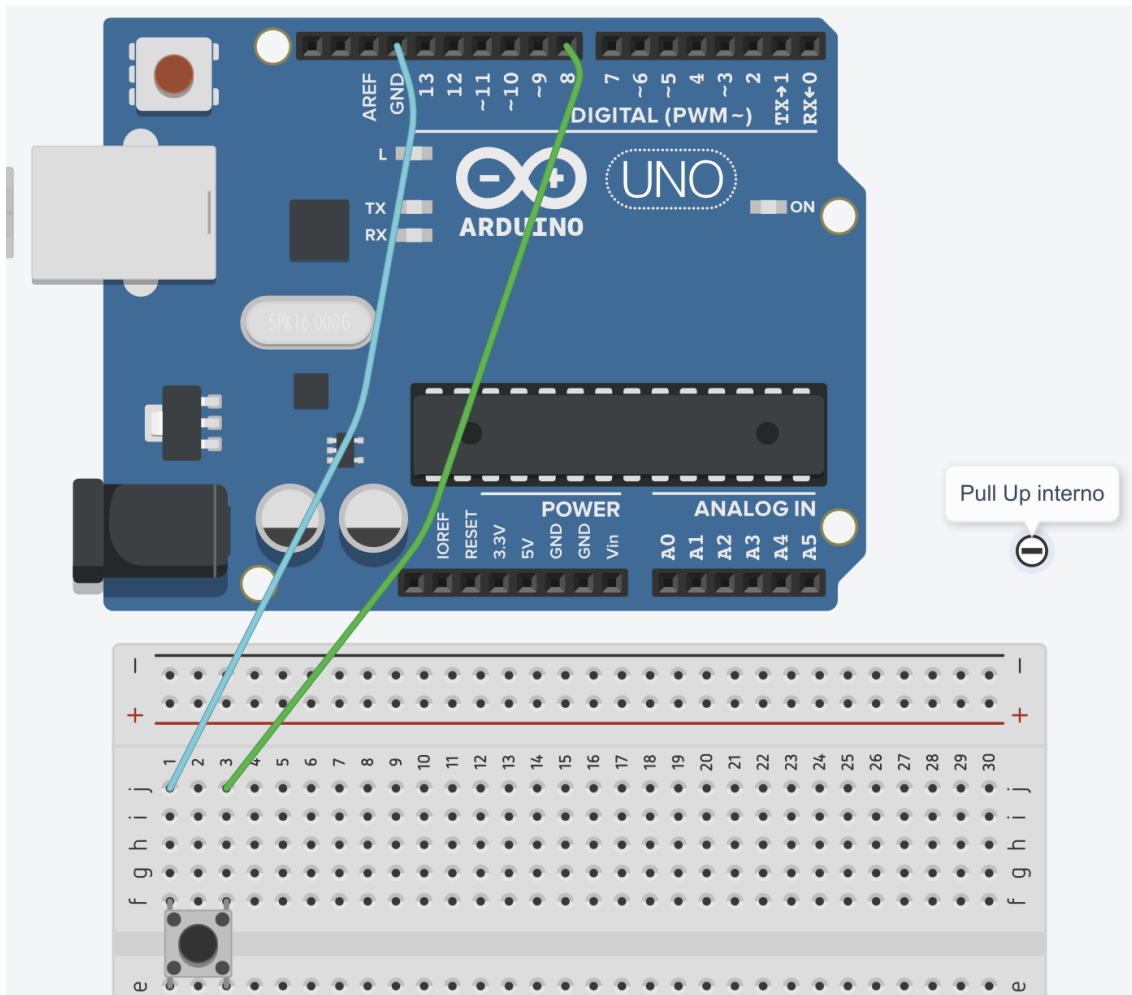
Esperar hasta que haya datos recibidos por el puerto serie

8. Entradas digitales. Pulsadores. OR, AND, if, else.



9-Pulsadores, entradas digitales en Arduino

La conexión más sencilla usando el modo PULLUP INTERNO sería (se podría usar cualquier pin digital, no tiene por qué ser el 8):



Las instrucciones básicas para los pulsadores en modo PULLUP INTERNO serían:

```
int pinPulsador = 8;
int valorPulsador ;

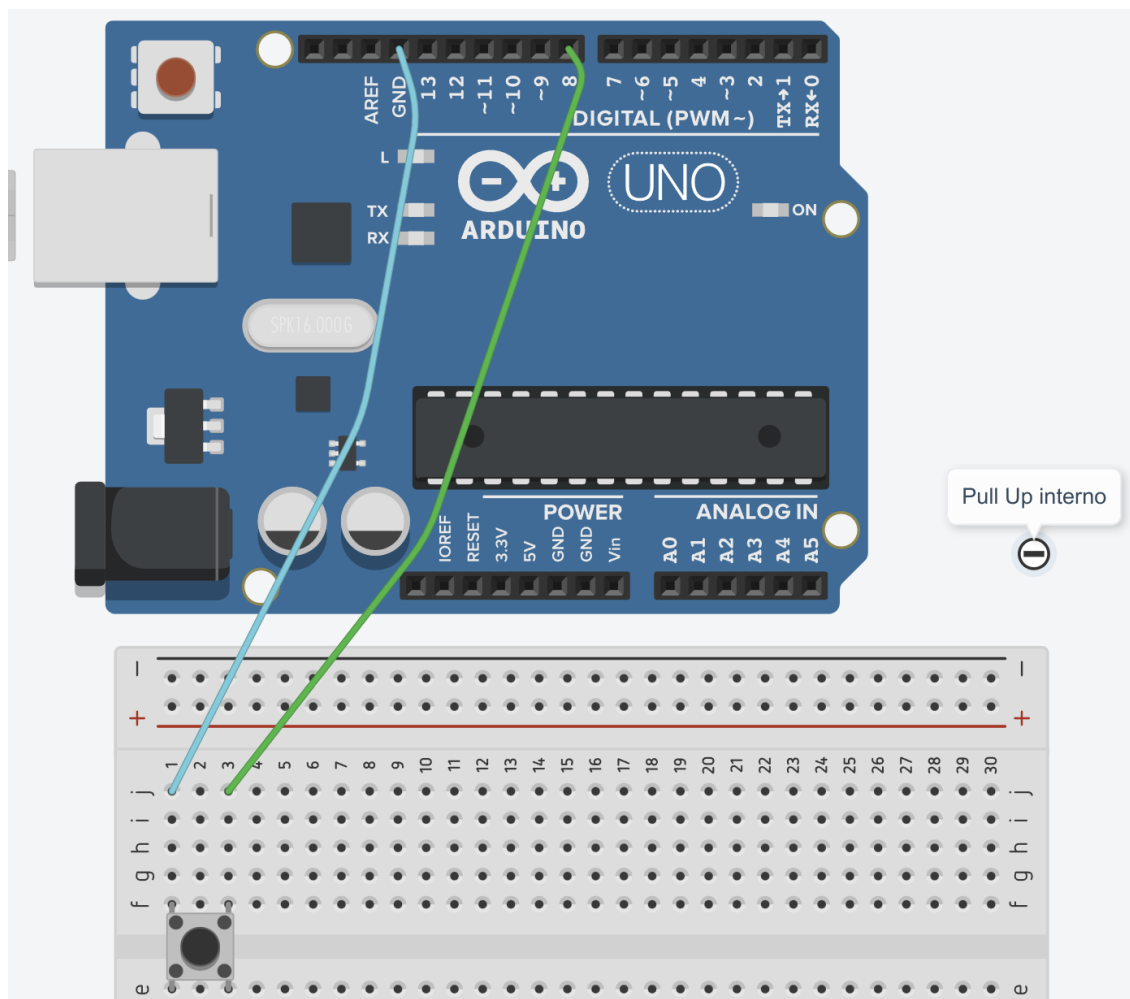
void setup() {
  pinMode(pinPulsador,INPUT_PULLUP); //Configura el pin como entrada "PULLUP"
}

void loop() {
  valorPulsador = !digitalRead(pinPulsador); // En PULLUP invertimos el valor con ! (NOT)
}
```

Nota: el singo ! antes de digitalRead se usa para invertir el valor digital: de 0 se transforma en 1 y el 1 se transforma a 0 .

Tarea 8-1-Pulsador_monitor

Realiza el montaje de la imagen:



Crea el programa que se explica en el vídeo que has visualizado (**9-Pulsadores, entradas digitales en Arduino**) para que se muestre en el monitor serie:

- 1 mientras pulsamos el pulsador
- 0 si no lo pulsamos

Tarea 8-2-Pulsadores_AND

▶ 10- if, else, AND y OR, variables booleanas y constantes

Imagina que tienes que diseñar una máquina para una fábrica que es peligrosa si el operario introduce una mano en ella. Para ello, debes diseñarla de tal manera que para que el operario la pueda poner en marcha, deba pulsar dos pulsadores a la vez, uno con cada mano, de tal manera de que no le quede ninguna libre que pueda introducir por error en la máquina. La máquina la vamos a simbolizar con un LED.

Monta el circuito de la figura con dos pulsadores, un LED y una resistencia de 220Ω.

Para sacar un 10: si en vez de con 1 LED lo haces con 3 LEDs que se enciendan secuencialmente, tendrás un 10 en la práctica

El programa sería similar al siguiente:

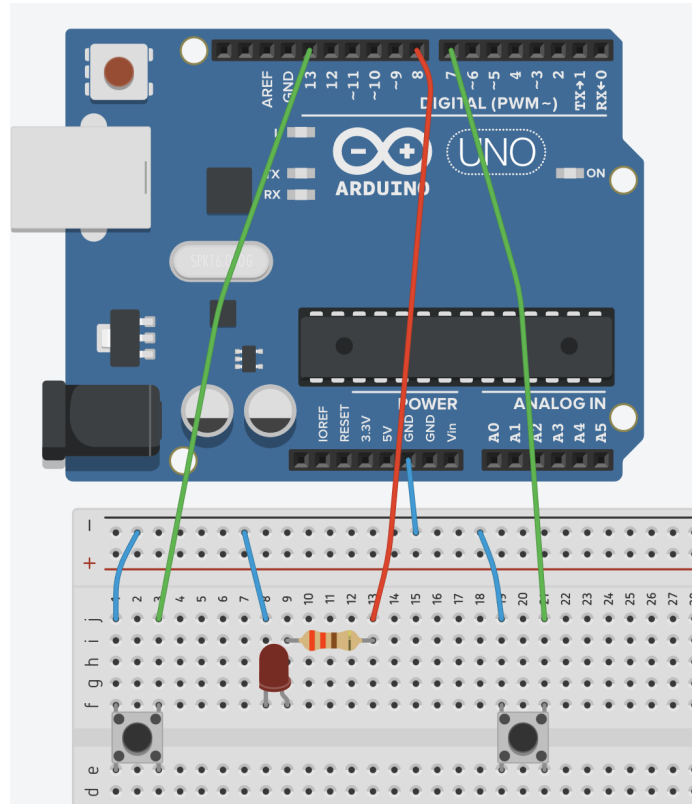


Diagrama de un programa de Arduino en modo bloques:

- Inicializar**
 - Establecer pinPulsador1 = 13
 - Establecer pinPulsador2 = 7
 - Establecer valorPulsador1 = falso
 - Establecer valorPulsador2 = falso
- Bucle**
 - Establecer valorPulsador1 = no
 - Leer digital Pin pinPulsador1
 - Establecer valorPulsador2 = no
 - Leer digital Pin pinPulsador2
 - si
 - valorPulsador1 = On y valorPulsador2 = On
 - hacer
 - Escribir digital Pin pinLed = On
 - sino
 - Escribir digital Pin pinLed = Off

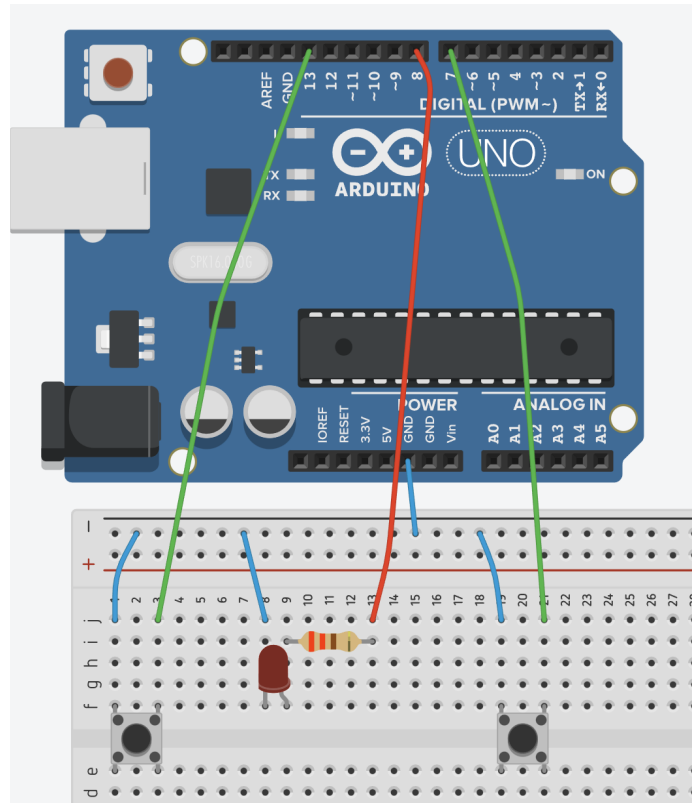
El código fuente correspondiente es:

```
if (valorPulsador1 == 1 & valorPulsador2 == 1){
  digitalWrite(pinLed,1);
}
else {
  digitalWrite(pinLed,0);
}
```

Tarea 8-3-Pulsadores_OR

Con lo que has aprendido del vídeo de la tarea anterior, imagina que tienes que diseñar una máquina y quieres que se pueda poner en marcha desde un pulsador ubicado en una zona o desde otro pulsador ubicado en otro lugar.

Monta el mismo circuito de la práctica anterior con dos pulsadores, un LED y una resistencia de 220Ω.



El programa sería similar al siguiente:

Inicializar

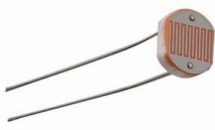
- Establecer pinPulsador1 = 13
- Establecer pinPulsador2 = 7
- Establecer valorPulsador1 = falso
- Establecer valorPulsador2 = falso

Bucle

- Establecer valorPulsador1 = no Leer digital Pin pinPulsador1
- Establecer valorPulsador2 = no Leer digital Pin pinPulsador2
- + si
 - valorPulsador1 = On o valorPulsador2 = On
- hacer Escribir digital Pin pinLed On
- sino Escribir digital Pin pinLed Off

```
if (valorPulsador1 == 1 || valorPulsador2 == 1){
  digitalWrite(pinLed,1);
}
else {
  digitalWrite(pinLed,0);
}
```

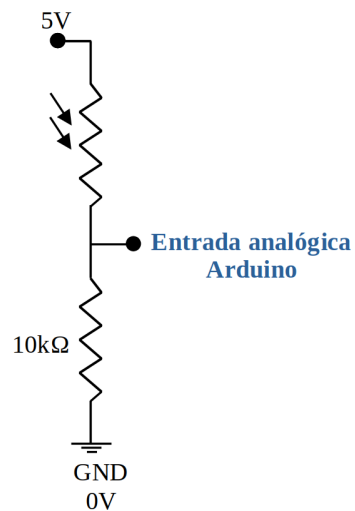

9. Entradas analógicas. Sensores de luz con LDR



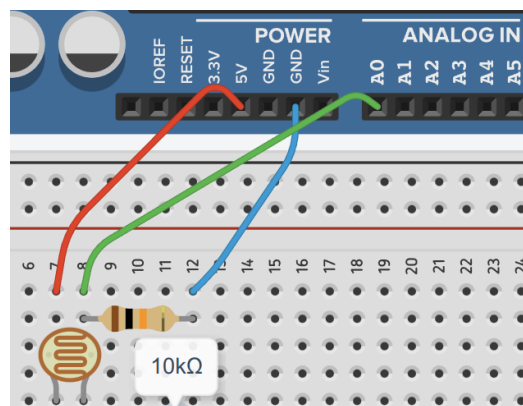
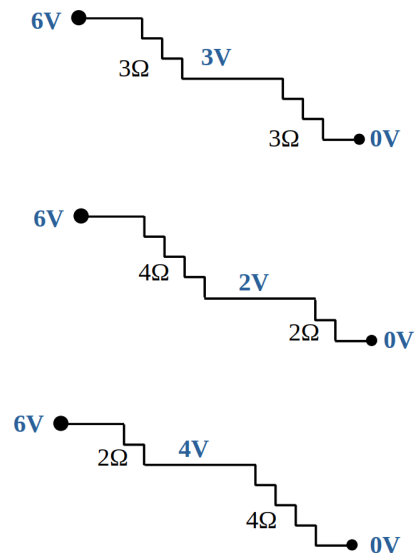
Fotorresistencia o LDR (Light Depending Resistor): resistencia que varía con la luz. A más luz deja pasar más corriente, por lo tanto, a más luz tiene menos resistencia.

La conexión eléctrica forma un divisor de tensión:

Conexión de LDR en Arduino



Divisor de tensión



Instrucciones básicas para una LDR (fotorresistencia):

```
int pinLdr =A0 ;
int valorLuz ;

void setup() {
  pinMode(pinLdr,INPUT); //configuramos el pin como entrada
}

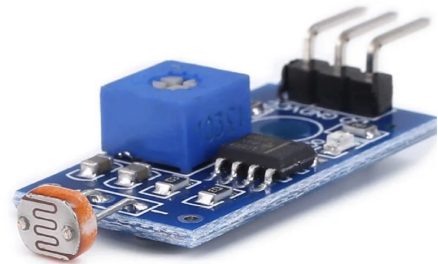
void loop() {
  valorLuz = analogRead(pinLdr); //guardamos el valor de entrada en valorLuz
}
```

Módulos LDR de 3 pines con potenciómetro:

Hay módulos LDR con 3 pines en los que **no hace falta resistencia**, la lleva incorporada además de un potenciómetro para calibrarla.

Pines y su conexión a la placa Arduino:

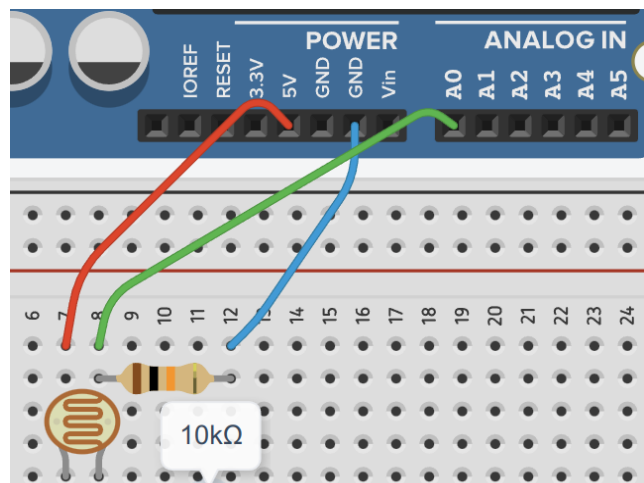
- **Vcc:** a 5V de Arduino
- **GND:** a GND de Arduino
- **DO (data output):** a entrada analógica de Arduino



Tarea 9-1-LDR_monitor

11- Entradas digitales en Arduino. LDR y monitor serie

Monta el siguiente circuito con Arduino y siguiendo las instrucciones del vídeo realiza un programa para visualizar en el monitor serie cómo detecta la LDR el nivel de luz.



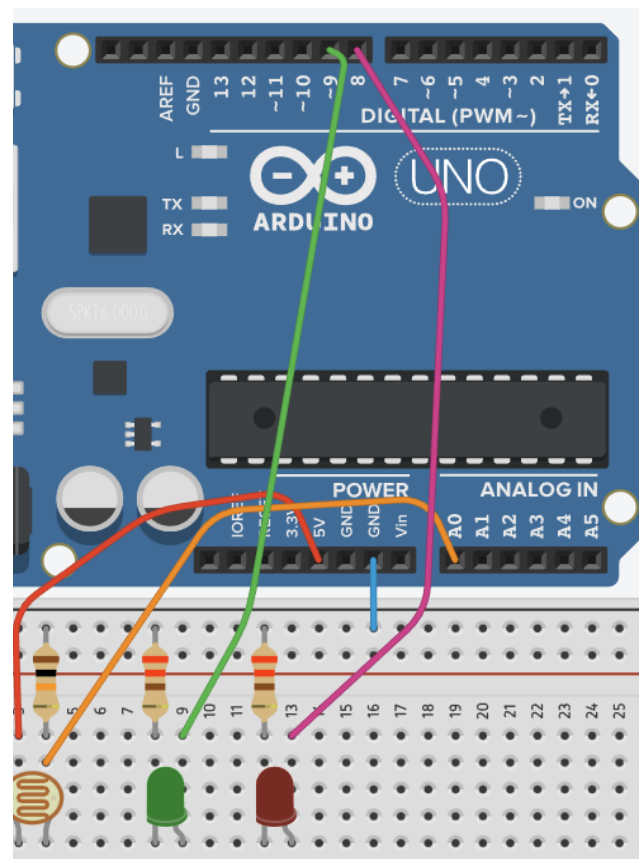
Tarea 9-2-Sensor_luz_LEDs

12- Arduino: Sensor de luz con LEDs

Monta el circuito de la imagen y realiza el programa de un sensor de luz que muestre con una luz verde un nivel normal de luz y se encienda una luz roja de alarma en el caso de una luz muy brillante como sería la luz directa sobre la LDR de la linterna del móvil.

Para sacar un 10: el LED rojo deberá parpadear para lograr el 10 en la práctica.

Necesitarás dos resistencias de 220Ω para los LEDs y otra de 10kΩ que va junto a la LDR.



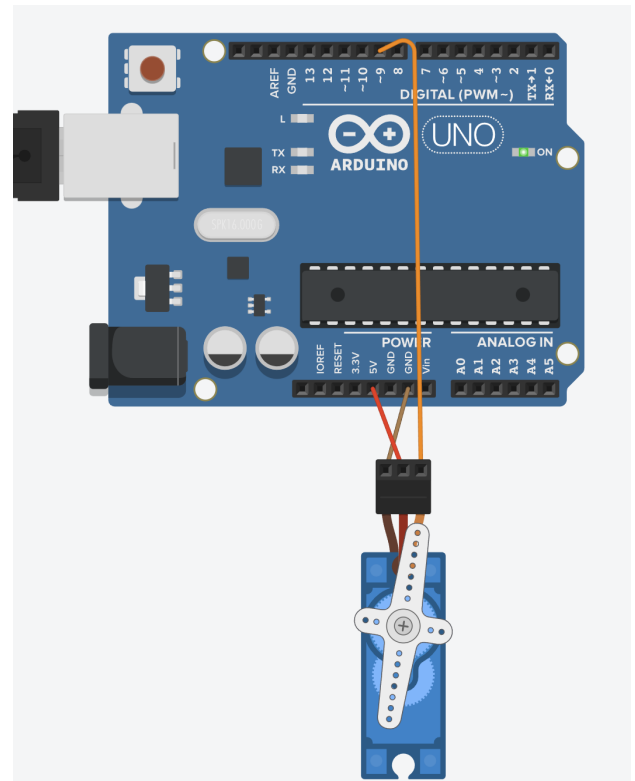
10. Servomotores

13- Servomotores en Arduino. Guía básica

Son motores en los que podemos controlar el ángulo de giro mediante la programación con Arduino IDE.

El modelo SG90 (el de la figura) es muy utilizado porque es el más económico, pero no puede transmitir mucha potencia. Puede girar de 0° a 180°.

La conexión básica sería como en la imagen, teniendo que conectar **el cable naranja a cualquier salida PWM**.



Instrucciones básicas para Arduino IDE:

```
#include <Servo.h> //iniciamos la librería Servo.h

Servo miServo;      //asignamos al servo un nombre: "miServo" en este caso
int pinServo = 9;   //declaramos la variable del pin PWM donde está conectado
int angulo = 90;    //declaramos la variable con el ángulo de giro

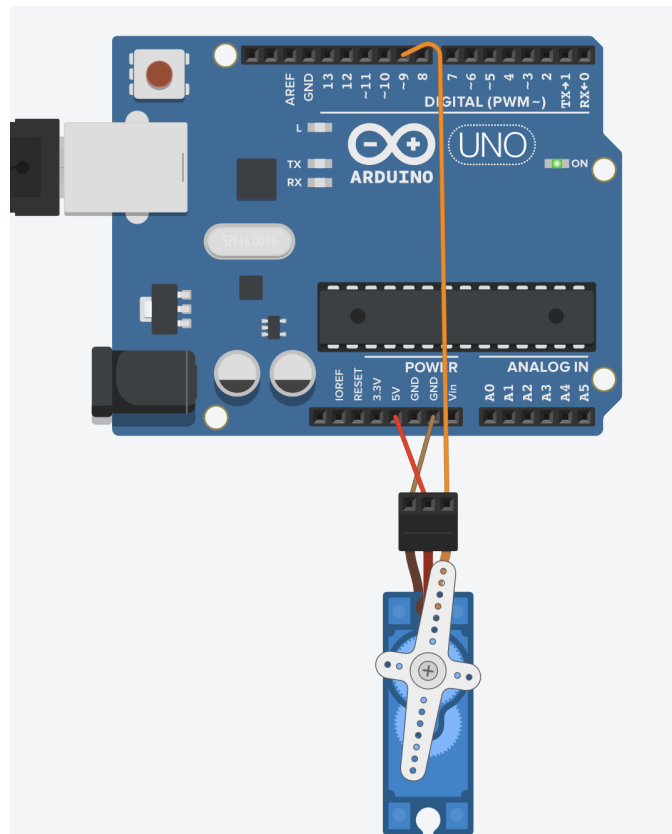
void setup() {
  miServo.attach(pinServo); //iniciamos miServo en el pin "pinServo"
}

void loop() {
  miServo.write(angulo); //movemos el miServo al ángulo "anguloServo"
}
```

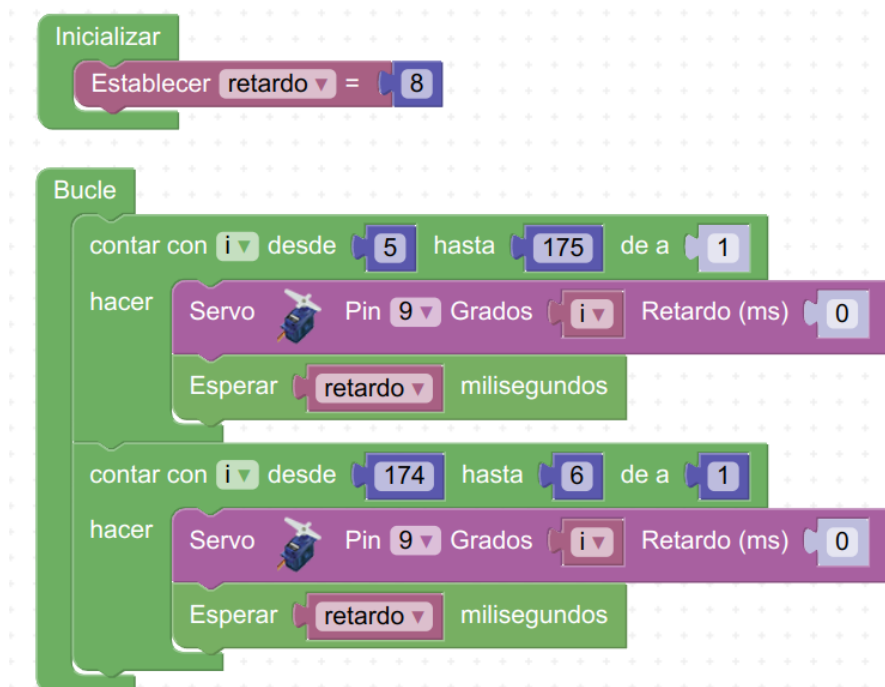
Tarea 10-limpiaparabrisas

Con el montaje básico de la figura, programa el servo usando el bucle "for()" para que haga un movimiento de vaivén como el de un limpiaparabrisas de un coche:

- Que vaya aumentando el ángulo de uno en uno desde 5° hasta 175°, con una espera de 8 ms entre una posición y la siguiente.
- Que vaya disminuyendo el ángulo de uno en uno desde 174° hasta 6° con una espera de 8 ms entre una posición y la siguiente.



Aquí tienes el programa con bloques que puedes usar como guía:



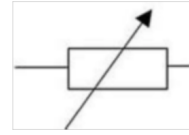
11. Potenciómetro. Función map()



Potenciómetro de usuario



Potenciómetro técnico



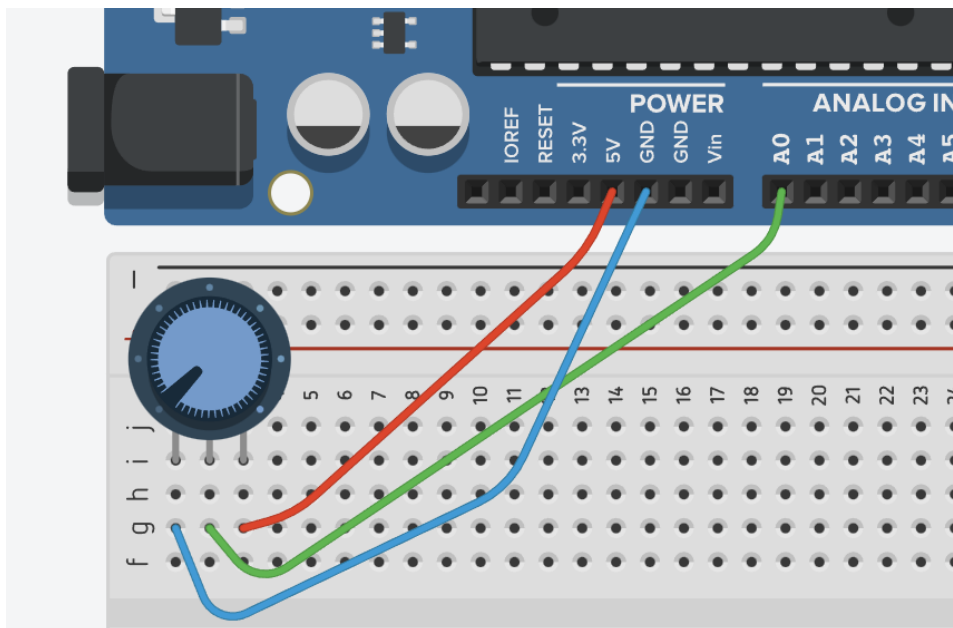
Símbolo del potenciómetro

Los potenciómetros son resistencias variables, las ajustamos al hacer girar su mando, aunque también las hay deslizables.



14- Potenciómetro. Conexión y programación en Arduino

Conexión básica a Arduino:



La instrucción para leer la entrada analógica será la estándar:

```
void loop()
{
  valorPotenciometro = analogRead(pinPotenciometro);
  Serial.println(valorPot);
}
```

15- Función map() de Arduino

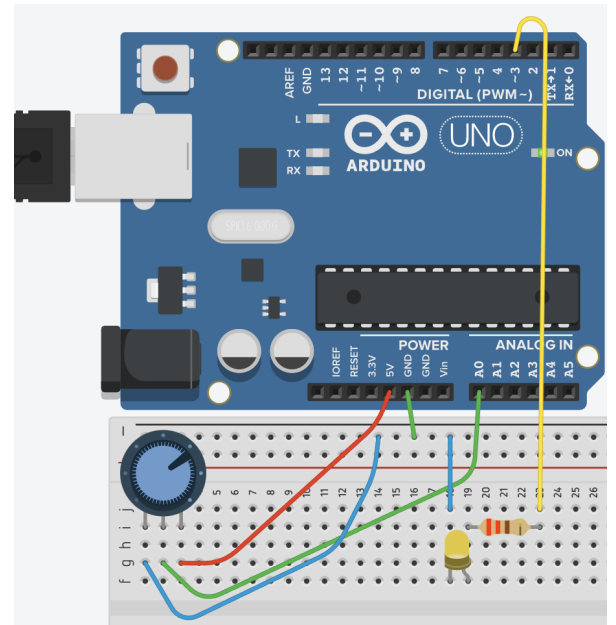
La **función map()** en Arduino permite remapear un número desde un rango a otro. Es útil para hacer equivalencias entre diferentes rangos, por ejemplo para remapear una lectura analógica que está en un rango de 0-1023 a un nivel de salida PWM que está en un rango 0-255. Ejemplo:

```
brillo = map(valorPotenciometro, 0, 1023, 0, 255);
```

la variable brillo sería proporcional a la entrada analógica del potenciómetro y se usará para el nivel de brillo de un LED.

Tarea 11-regulador_de_luz

Realiza un circuito que permita ajustar el nivel de luz de un LED mediante un potenciómetro.



Sería el programa equivalente al siguiente en bloques:

```
Inicializar
  Establecer pinPotenciometro = A0
  Establecer valorPotenciometro = 0
  Establecer pinLed = 3
  Establecer brillo = 0
  Establecer retardo = 200
  Iniciar Baudios 9600

Bucle
  Establecer valorPotenciometro = Leer analógica Pin A pinPotenciometro
  Enviar valorPotenciometro Salto de línea
  Esperar retardo milisegundos
  Establecer brillo = mapear valorPotenciometro de 0 a 1023 a 0 a 255
  Escribir analógica (PWM) Pin pinLed Valor brillo
```

Tarea 11-2-potenciometro_servo

Realiza la práctica que se visualiza al final del vídeo ([clic para verla](#)) en la que un potenciómetro se usa para mover un servomotor.

PROGRAMACIÓN C++ (PROFUNDIZACIÓN)

12. Tipos de variables más usados



Variables en Arduino II. Tipos, advertencias, ámbitos...

Existen más tipos de variables, en nuestros proyectos los que más necesitaremos serán:

Tipo	Tamaño	Rango
bool	1 byte	0 a 1 (true o false)
byte	1 byte	0 a 255
int	2 bytes	-32.768 a 32.767
unsigned int	2 bytes	0 a 65535
long	4 bytes	-2.147.483.648 a 2.147.483.647
unsigned long	4 bytes	0 a 4.294.967.295
float	4 bytes	3,4028235E-38 a 3,4028235E+38

Notas y advertencias a tener en cuenta.

Desbordamiento al sobrepasar un rango.

Si sobrepasamos el rango de una variable se produce un **desbordamiento**, reiniciándose a los primeros valores del rango. Por ejemplo, damos a la variable “a = 255”, el máximo para el tipo “byte” (255), si después le sumamos 1 guardándolo en “b” se produce “desbordamiento” y vemos en el monitor serie que el resultado de “b” no será 256, sino se reinicia al mínimo valor del rango (0):

```
1 byte a = 255 ; //le asigno el máximo valor posible al tipo "byte"
2 byte b ;
3
4 void setup() {
5     Serial.begin(9600);
6 }
7
8 void loop() {
9     b = a + 1; //le sumamos 1 para ver qué ocurre al producirse desbordamiento
10    Serial.println (b) ;
11    delay(500);
12 }
13
```

Salida Monitor Serie x

Mensaje (Intro para mandar el mensaje de 'Arduino Uno' a '/dev/ttyACM0') Sin ajuste de línea 9600 baud

0
0
0

Float: limitaciones y aspectos a tener en cuenta.

- El tipo de datos float tiene sólo entre 6 y 7 dígitos decimales de precisión. Eso significa el número total de dígitos, no el número a la derecha del punto decimal.
- Si hacemos cálculos con float, debemos agregar un punto decimal; de lo contrario, se tratará como un int:

<pre>1 int a = 9; 2 float b ; 3 void setup() { 4 Serial.begin(9600); 5 } 6 void loop() { 7 b = a/2; 8 Serial.println(b); 9 Serial.println(); 10 }</pre>	<pre>1 int a = 9; 2 float b ; 3 void setup() { 4 Serial.begin(9600); 5 } 6 void loop() { 7 b = a/2.; 8 Serial.println(b); 9 Serial.println(); 10 }</pre>
Salida Monitor Serie x	Salida Monitor Serie x
Mensaje (Intro para mandar el mensaje de 'Arduino U	Mensaje (Intro para mandar el mensaje de 'Arduino
4.00	4.50

- Los números de coma flotante no son exactos y pueden producir resultados extraños al compararlos. Por ejemplo, es posible que $9,0/0,3$ no sea igual a $30,0$ al compararlos.
- Para que realice un redondeo correcto al pasar a int, hay que usar `round()` :

```
float x = 2.9; // A float type variable
int y = x; // nos daría como resultado y = 2
int y = round(x); // nos daría como resultado y = 3
```
- Las matemáticas de punto flotante **son mucho más lentas** que las matemáticas de enteros al realizar cálculos, por lo que deben evitarse si, por ejemplo, un bucle tiene que ejecutarse a máxima velocidad para una función de sincronización crítica.
- Para obtener un número de decimales concreto en el monitor serie, se indica de la siguiente manera (ejemplo para que se muestren 4 decimales):

```
8   Serial.println(resultado,4);
9   }
10  }
```

Salida Monitor Serie x

Mensaje (Intro para mandar el mensaje de 'Arduino Uno' a '/dev/ttyAC

4.5000

13. Variables para frases y caracteres.

Tipo		Ejemplo
char	Para caracteres	char letra = 'a'
String	Para frases (cadenas de texto)	String mensaje = "¡Hola mundo!"

Ejemplo para declarar una variable de tipo "char" con nombre "letra" y cuyo valor queremos que sea la letra "a":

```
1 char letra = 'a';
2
3 void setup() {
4   Serial.begin(9600);
5 }
6
7 void loop() {
8   Serial.println (letra) ;
9   delay(500);
10 }
11
```

Salida Monitor Serie x

Mensaje (Intro para mandar el mensaje de 'Arduino Uno' a '/dev/

a
a
a

Ejemplo para crear un objeto de la clase String:

```
1 String miFrase = "Hola, mundo!";
2
3 void setup() {
4   Serial.begin(9600);
5 }
6
7 void loop() {
8   Serial.println (miFrase) ;
9   delay(500);
10 }
11
```

Salida Monitor Serie x

Mensaje (Intro para mandar el mensaje de 'Arduino Uno' a '/dev/ttyACM0')

Hola, mundo!
Hola, mundo!
Hola, mundo!

14. Variables globales y variables locales.

Las variables pueden ser de ámbito global o local:

- **Variables globales:** su ámbito es todo el programa.
- **Variables locales:** su ámbito es solamente la función donde se encuentran

Ejemplo:

```
int a = 2 ; //variable global, "funciona" en todo el programa

void setup() {

}

void loop() {
  int b = 5 ; //variable local, nada más afecta a la función loop()
}
```

Las variables locales funcionan en un ámbito más pequeño pero van a consumir menos de memoria, por lo que el programa será algo más eficiente.

15. Leer enteros, decimales y texto en el monitor serie.

Dependiendo del tipo, usaríamos:

- Para enteros → `Serial.parseInt()`;

```
int numEntero;
numEntero = Serial.parseInt();
```

- Para decimales → `Serial.parseFloat()`;

```
float numDecimal;
numDecimal = Serial.parseFloat();
```

- Para texto → `Serial.readString()`;

```
String = mensaje;
mensaje = Serial.readString();
```

16. Formas de declarar una constante

Hay una manera de definir constantes muy extendida que sigue la siguiente forma (sin punto y coma al final):

```
#define pinLed 13
```

Sería equivalente a:

```
const int pinLed = 13;
```

Cada forma tiene sus pros y contras, en la documentación de Arduino se recomienda en general usar la forma:

```
const int pinLed = 13;
```

MÁS SENSORES Y ACTUADORES PARA PROYECTOS

17. Sensor ultrasónico de distancia HC-SR04

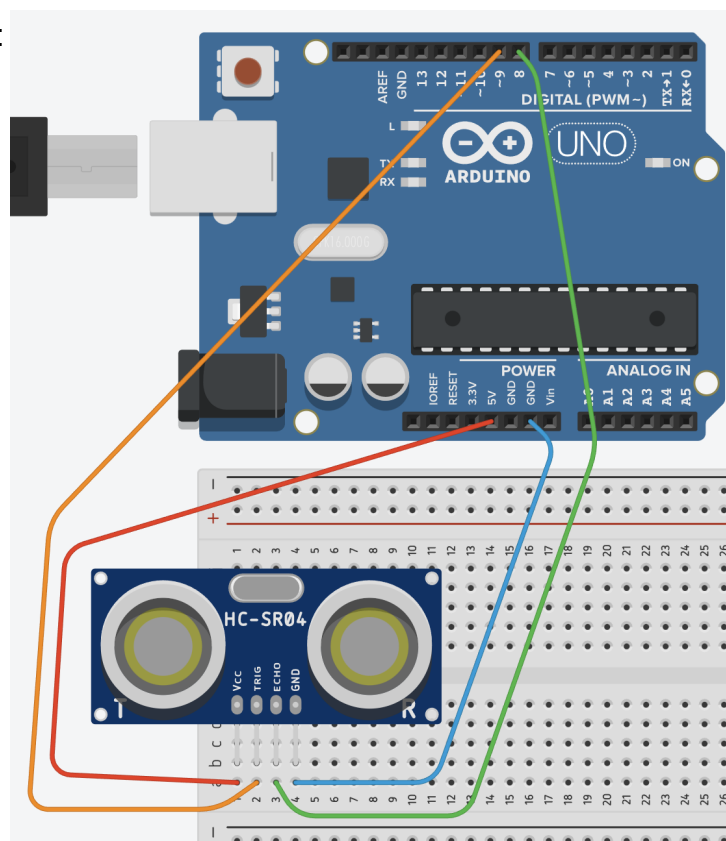
 [Dos maneras de programar el sensor de distancia](#)

Los sensores **HC-SR04** utilizan ondas ultrasónicas para determinar la distancia entre el sensor y un objeto. Mide con una precisión aceptable entre 2 y 400cm



Las patillas TRIG (trigguer o disparador) y ECHO (eco) pueden ir a cualquier pin digital de Arduino, configurando el TRIG como salida digital y ECHO como entrada.

El montaje básico sería:



Dos posibles formas para programarlo:

1. Sin instalar biblioteca adicional, pero con más líneas de código:

```
const int TRIG = 9; //definimos un pin digital para TRIG
const int ECHO = 8 ; //definimos un pin digital para ECHO
long tiempo, distancia; //variables de tipo long por ser cantidades grandes en µs
int retardo = 500; //definimos una espera entre mediciones

void setup()
{
  pinMode(TRIG,OUTPUT); //el pin TRIG es de salida
  pinMode(ECHO,INPUT); // el pin ECHO de entrada
  Serial.begin(9600);
}

void loop()
{
  //Emitimos un pulso de 10µs
  digitalWrite(TRIG,LOW); //nos aseguramos de que el trigger está desactivado
  delayMicroseconds(2); //esperamos 2µs, así que usamos "delayMicroseconds()"
  digitalWrite(TRIG,HIGH); //mandamos un pulso de 10µs al TRIG
  delayMicroseconds(10);
  digitalWrite(TRIG,LOW);

  tiempo = pulseIn(ECHO,HIGH); //nos mide en µs el tiempo en detectar el eco
  distancia = tiempo * 0.034 / 2 ; // V_sonido=340m/s pero convertimos µs-> s y m->cm

  Serial.print(distancia); // mostramos la distancia en el monitor serial
  Serial.println(" cm");
  delay(retardo);
}
```

2. Instalando la biblioteca Ultrasonic.h de Erick Simões, con mucho menos código:

```
#include <Ultrasonic.h> //Librería Ultrasonic de Erick Simões

Ultrasonic ultrasonic(9,8); //podemos cambiar ultrasonic por otro nombre
int distancia;

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

void loop() {
  distancia = ultrasonic.read(); //nos da la distancia en cm

  Serial.print(distancia);
  Serial.println(" cm");
  delay(1000);
}
```

18. Pantalla LCD1602 I2C

Pantalla LCD1602 con Arduino

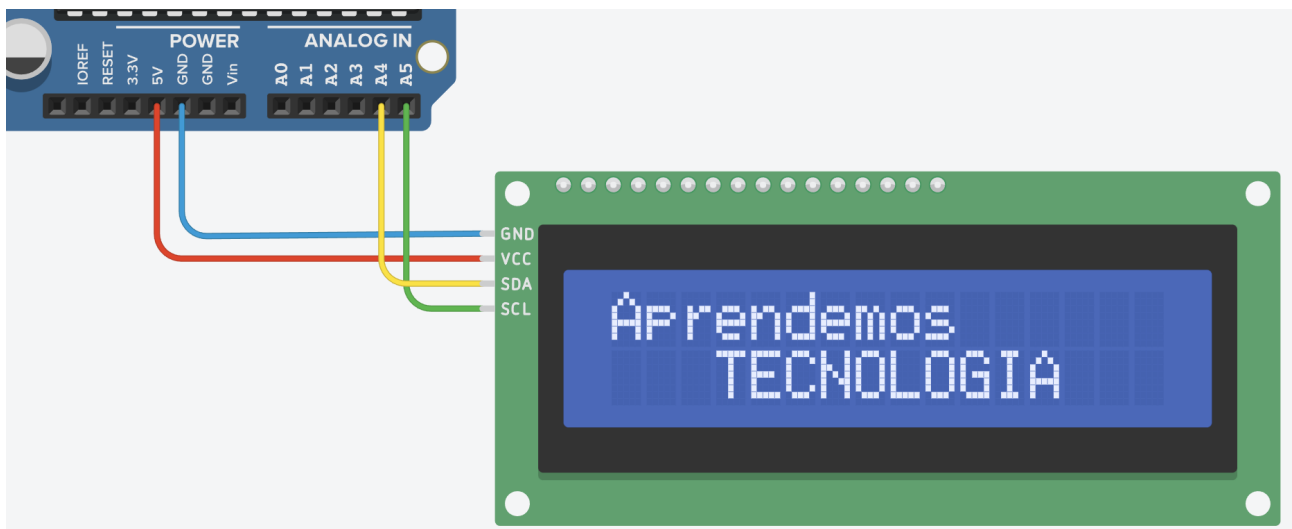
La pantalla LCD 16×02 para microcontrolador Arduino permite mostrar visualmente los datos de los sensores.



Nosotros vamos a usar la que tiene un módulo I2C que reduce las conexiones a 4 cables:

- GND
- Vcc (5V)
- SDA: **serial data**, lo conectamos **siempre a A4 en Arduino UNO**.
- SCL: **serial clock** lo conectamos **siempre a A5 en Arduino UNO**.

Conexiones



Instalación de biblioteca necesaria.

Nos vamos al menú Herramientas > Gestionar bibliotecas e instalamos la siguiente:

- **En Arduino IDE 1:** LiquidCrystal I2C de Marco Schwartz (ponemos **Schwartz** en el buscador):

LiquidCrystal I2C

by Marco Schwartz

A library for I2C LCD displays. The library allows to control I2C displays with functions extremely similar to LiquidCrystal library. THIS LIBRARY MIGHT NOT BE COMPATIBLE WITH EXISTING SKETCHES.

[More info](#)

- **En Arduino IDE 2:** LiquidCrystal I2C de Frank de Brabander (ponemos **Brabander** en el buscador):

LiquidCrystal I2C de Frank de Brabander

1.1.2 instalado

A library for I2C LCD displays. The library allows to control I2C displays with functions extremely similar to LiquidCrystal library. THIS LIBRARY MIGHT NOT BE COMPATIBLE WITH EXISTING SKETCHES.

[Más información](#)

Programación básica:

```
// Incluimos las dos bibliotecas siguientes:
#include "Wire.h"
#include "LiquidCrystal_I2C.h"

//definimos el display con el nombre lcd,
//la LCD está en la dirección I2C 0x27, tiene 16 columnas y 2 filas:
LiquidCrystal_I2C lcd(0x27,16,2);

void setup() {
  lcd.init(); //iniciamos
  lcd.backlight(); //encendemos la retroiluminación
  //colocamos en el setup los caracteres que vayan a estar fijos en pantalla

  //colocamos el cursor donde queremos que inicie el texto
  //comenzamos a contar desde cero para la posición del cursor: 0, 1, 2...
  lcd.setCursor(2, 0); //tercera posición de la primera fila
  lcd.print("APRENDEMOS"); //mostramos en pantalla de esta forma
  lcd.setCursor(3, 1); //cuarta posición de la segunda fila
  lcd.print("TECNOLOGIA"); //NO ES COMPATIBLE CON TILDES
}

void loop() {
  // colocamos en el loop() caracteres no fijos (variables, animaciones...)
}
```

>>> Si no se ven o se ven mal los caracteres, ajustamos el contraste de la pantalla usando el potenciómetro azul de la parte de atrás:



Para elegir el número de decimales mostrado de una variable float.

Ponemos una coma y el número de decimales deseado. Ejemplo para mostrar el valor de la variable temp con un decimal:

```
lcd.print(temp, 1);
```

Caracteres especiales y personalizados.

- Tenemos en cuenta de que **no es compatible con tildes**.
- Para poner el símbolo ° de grados, escribimos la siguiente línea:

```
lcd.print((char)223);
```

La programación básica de un corazón sería así, fíjate que los 0 y 1 forman el dibujo del corazón:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

// Define el carácter personalizado del corazón
byte heart[8] = {
  B00000,
  B01010,
  B11111,
  B11111,
  B11111,
  B01110,
  B00100,
  B00000
};

void setup() {
  lcd.init();
  lcd.backlight();
  // Cargar el carácter personalizado en la memoria del LCD:
  lcd.createChar(0, heart);
  // Limpiar la pantalla:
  lcd.clear();
  // Muestra el carácter personalizado del corazón en la posición (0, 0)
  lcd.setCursor(0, 0);
  lcd.write(0); // Usa el carácter personalizado 0 (corazón)
}

void loop() {
}
```

Para hacer caracteres personalizados de forma más visual, también encontramos editores online como el [editor de Arduinoblocks](#) que nos generan el código.

19. Sensor de temperatura y humedad DHT11

El sensor DHT11 permite detectar valores de temperatura y humedad.

Conexiones:

- **Vcc:** a 5V
- **GND**
- **Data:** cualquier pin digital



Instalación de biblioteca necesaria.

Si no lo hemos hecho antes, en Arduino IDE ir al menú Herramientas → Gestionar bibliotecas e instalar la siguiente librería: **DHT sensor library de Adafruit**

Programación básica mostrando los datos en el monitor serie:

```
#include <DHT.h> //incluimos la biblioteca necesaria

// definimos con un nombre (dht en este caso)
// pin donde conectamos (8) y tipo de DHT (DHT11):
DHT dht(8,DHT11);

void setup() {
  Serial.begin(9600);
  dht.begin(); //iniciamos el dht
}

void loop() {
  delay(2000);
  // guardamos en la variable humedad el valor leído:
  int humedad = dht.readHumidity();

  // guardamos en una variable float (con decimales) el valor leído:
  float temp = dht.readTemperature();

  Serial.print("Humedad: ");
  Serial.print(humedad);
  Serial.print("%");

  Serial.print(" Temperatura: ");
  Serial.print(temp,1); //(muestra la Tª con un decimal)
  Serial.println("°C");
}
```

Recordatorio: para mostrar en una pantalla LCD1602 el símbolo ° de grados, escribimos la siguiente línea:

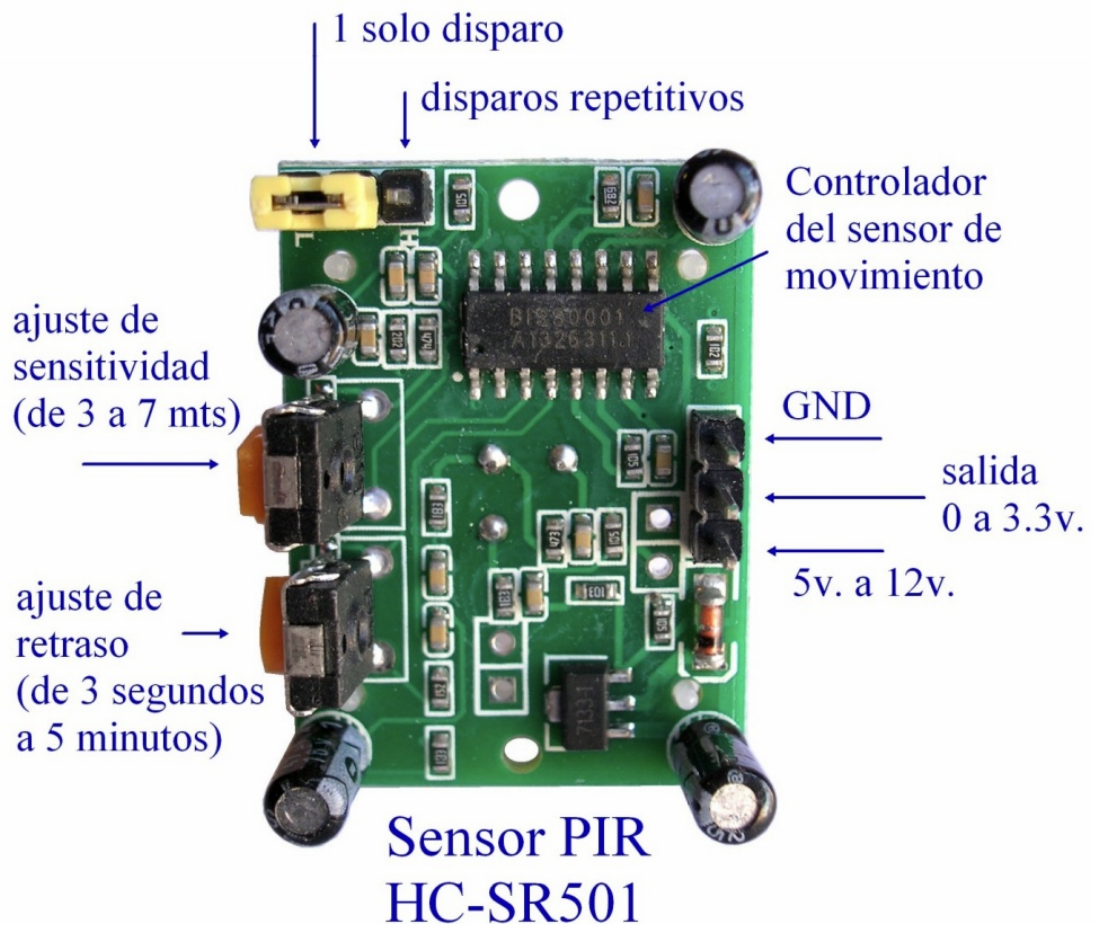
```
lcd.print((char)223);
```


20. Sensor infrarrojo pasivo HC-SR501. Detector movimiento

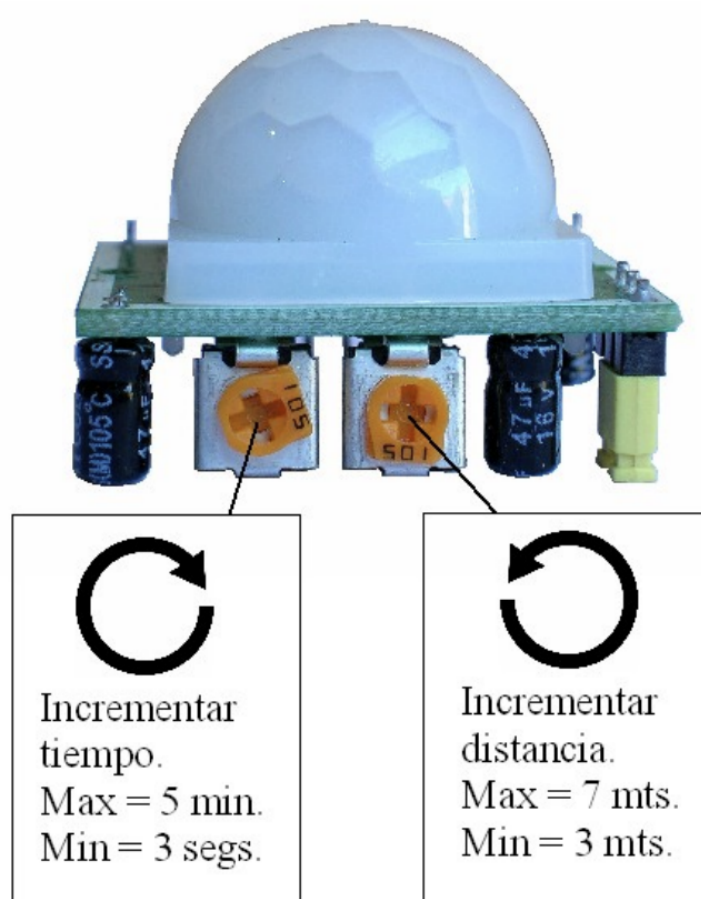
Los sensores infrarrojos pasivos (PIR) son dispositivos para la detección de movimiento.



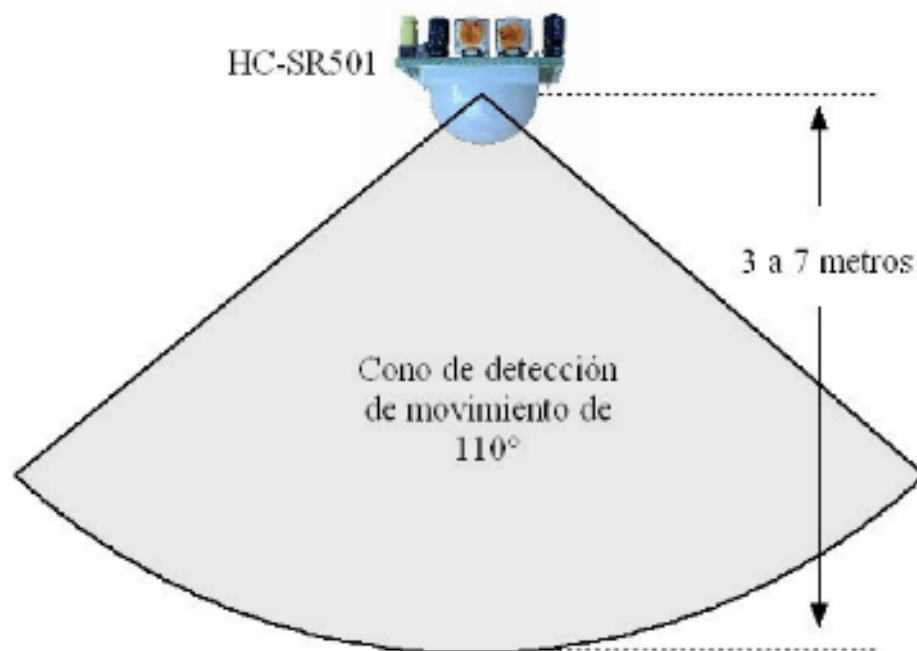
Esquema del patillaje:



Sensor PIR
HC-SR501

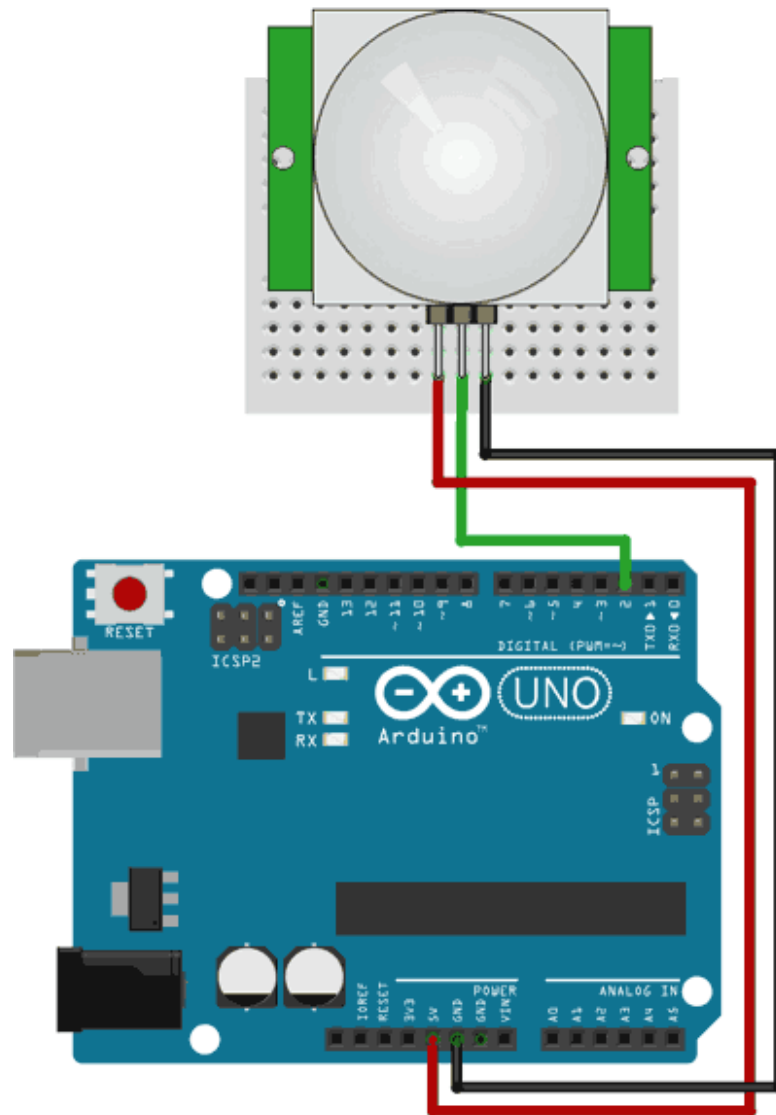


El rango de detección de movimiento de los PIR es ajustable y generalmente funcionan con alcances de hasta 7 metros, y con aperturas de 90° a 110°, como se muestra en la figura. El montaje del PIR puede realizarse tanto en piso, muro ó techo, según convenga a la aplicación.



Esquema eléctrico:

Debemos conectarlo a una entrada digital



Ejemplo de programación de un LED que se enciende al detectar movimiento:

```
const int pinPir = 8; // Pin al que está conectado el sensor PIR
const int pinLed = 13; // Pin al que está conectado el LED
bool movimiento = false;

void setup() {
  pinMode(pinPir, INPUT);
  pinMode(pinLed, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  movimiento = digitalRead(pinPir);

  if (movimiento == true) {
    digitalWrite(pinLed, HIGH); // Enciende el LED
    Serial.println("¡Movimiento detectado!");
    delay(5000); // Puedes ajustar el tiempo de encendido del LED
según tus necesidades
  }
  else {
    digitalWrite(pinLed, LOW); // Apaga el LED
    Serial.println("Nada detectado");
  }
}
```

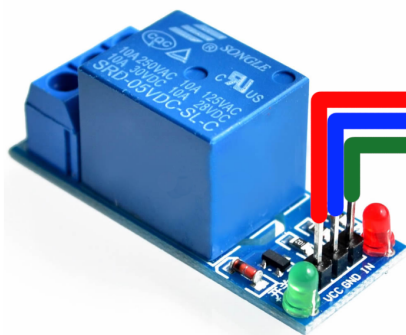
21. Relé.

¿Qué es un relé (relay)?

Un relé funciona como un interruptor controlado por un circuito eléctrico en el que, por medio de una bobina y un electroimán, se acciona un juego de uno o varios contactos que permiten abrir o cerrar otros circuitos eléctricos independientes.



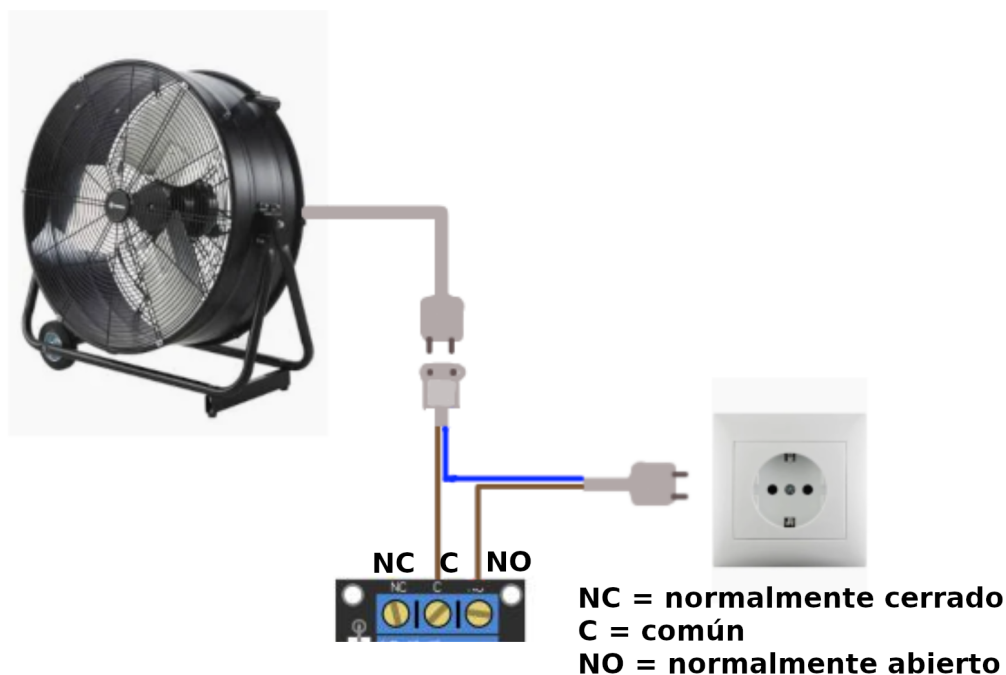
Conexiones a Arduino:



Vcc a 5V de Arduino
GND a GND de Arduino
IN a salida digital de Arduino

Ejemplo de conexiones a un circuito actuador

El segundo circuito puede ser de corriente continua o de corriente alterna a 230V



Ejemplo de programa básico

Activa y desactiva el relé intermitentemente cada 2 segundos:

```
const int pinRele = 8;
int retardo = 2000;

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

void loop() {
  digitalWrite(pinRele, HIGH);
  delay(retardo);
  digitalWrite(pinRele, LOW);
  delay(retardo);
}
```

22. Tira de LEDs NeoPixel WS2812B



[Aprende a programar las tiras de LED con Arduino.](#)

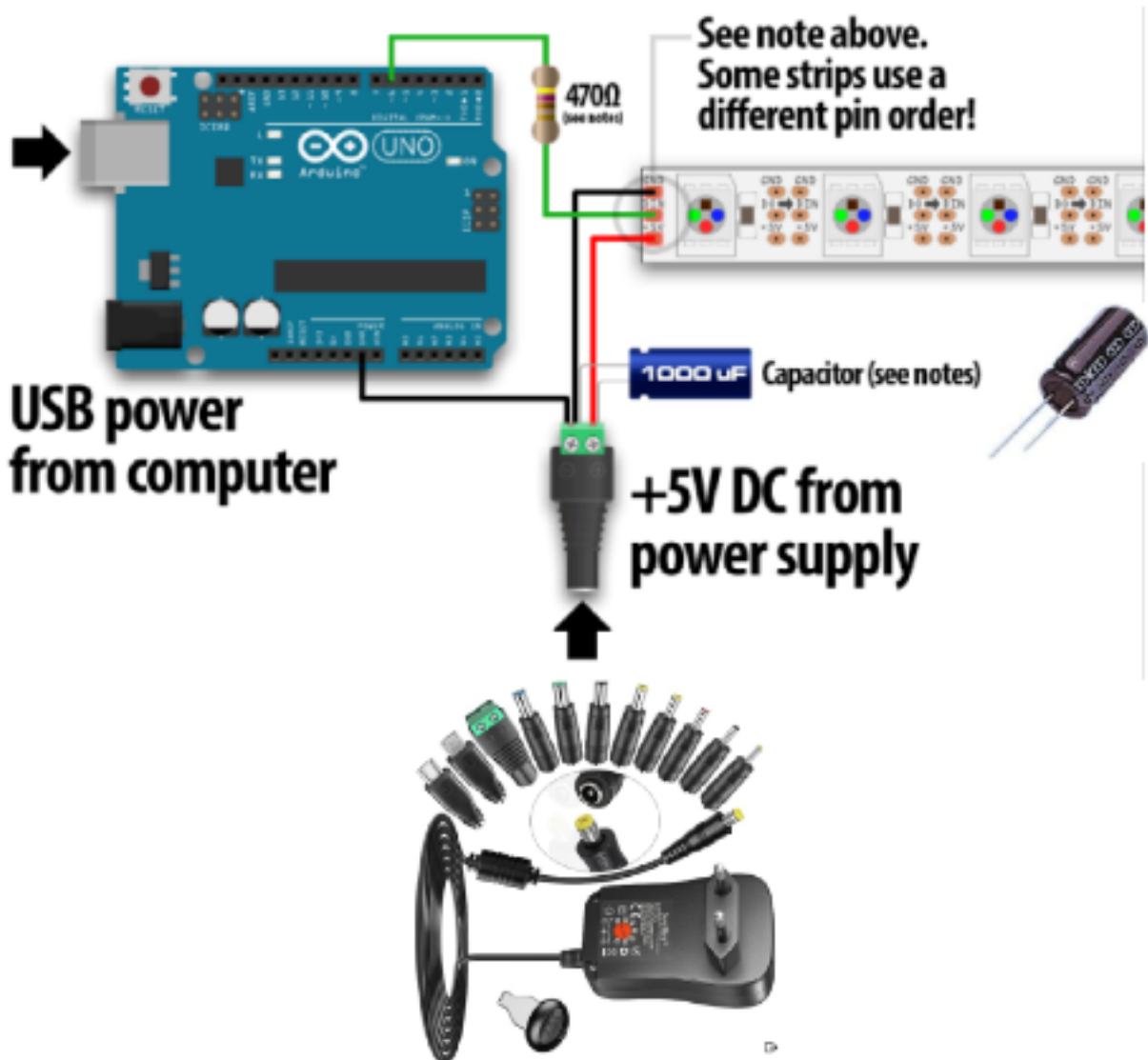
Es una tira LED en la que podemos controlar el color de los LEDs de forma independiente, pudiendo cambiar el color y brillo para realizar efectos y usarla en distintos proyectos.



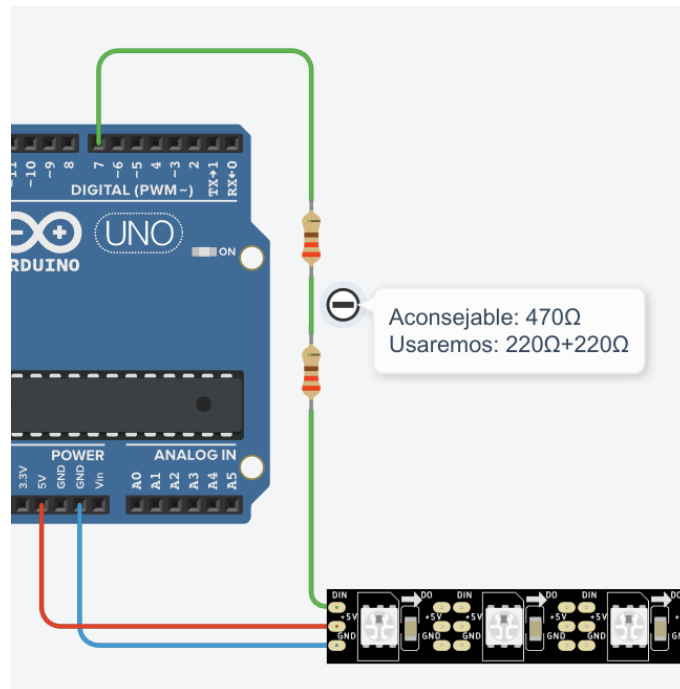
Conexiones:

Para el pin de señal, podemos emplear cualquiera de las salidas digitales de Arduino (no hace falta que sea PWM). Es necesario emplear una resistencia de 470Ω entre el pin digital y el pin de señal del WS2818b, o podéis dañar los primeros LED, si no tenemos, podemos usar dos resistencias en serie de 220Ω de las que usamos con los LEDs.

- **Opción 1 de conexión (aconsejable si se dispone de los componentes):**
También se aconseja instalar un condensador de al menos $1000\ \mu\text{F}$ entre GND y 5V si la fuente de tensión no tiene la suficiente capacidad. **MUY IMPORTANTE:** respetar la polaridad del condensador, la patilla corta al polo negativo (lleva una franja blanca en el lateral indicándolo también).



- **Opción 2 de conexión (sin necesidad de fuente de alimentación externa ni condensador).** Es la que en principio usaremos en clase, pero dada la limitación de corriente de salida de la placa de Arduino UNO alimentada por USB, **usaremos de intensidad máxima para los colores RGB de los LEDs de 50 en vez de 255.**
 - **GND:** a GND de Arduino
 - **+5V:** a 5V de Arduino
 - **Din (entrada de datos):** cualquier pin digital (da igual que no sea PWM)



Programación básica.

Programa que enciende el primer píxel (LED de la tira) azul con una intensidad de 50:

```
#include <Adafruit_NeoPixel.h>

const int pinLeds = 7; // pin al que está conectada la tira de LEDs
const int numLeds = 30; // Número total de LEDs en la tira

Adafruit_NeoPixel leds = Adafruit_NeoPixel(numLeds, pinLeds, NEO_GRB +
NEO_KHZ800);

void setup() {
  leds.begin();
  leds.show(); // Inicializa todos los píxeles a 'apagado'
}

void loop() {
  leds.setPixelColor(0,0 , 0, 50); //configura los LEDs (Nº
LED,brillo_rojo,brillo_verde,brillo_azul)
  leds.show(); //muestra en la tira de LEDs la instrucción anterior
}
```

Para conseguir el código numérico de los colores, podemos usar el siguiente enlace:

- **Selector de colores RGB online (www.w3schools.com)**

Programa que enciende y apaga intermitentemente todos los LEDs de color verde con una intensidad de brillo 20:

```
#include <Adafruit_NeoPixel.h>

const int pinLeds = 7; // El pin al que está conectada la tira de luces
const int numLeds = 30; // Número total de LEDs en la tira
int retardo = 2000; //retardo entre parpadeos

Adafruit_NeoPixel leds = Adafruit_NeoPixel(numLeds, pinLeds, NEO_GRB +
NEO_KHZ800);

void setup() {
leds.begin();
leds.show(); // Inicializa todos los píxeles a 'apagado'
}

void loop() {
for (int i = 0; i < numLeds; i++) {
leds.setPixelColor(i, 0, 20, 0); //(Nº pixel, rojo, verde, azul)
leds.show();
}
delay(retardo);

for (int i = 0; i < numLeds; i++) {
leds.setPixelColor(i, 0, 0, 0);
leds.show();
}
delay(retardo);
}
```

Luces rojas que suben y bajan en secuencia que enciende y apaga intermitentemente todos los LEDs de color rojo con una intensidad de brillo 50:

```
#include <Adafruit_NeoPixel.h>

const int pinLeds = 7; // El pin al que está conectada la tira de luces
const int numPixels = 30; // Número total de LEDs en la tira
int retardo = 20; //retardo entre parpadeos

Adafruit_NeoPixel leds = Adafruit_NeoPixel(numPixels, pinLeds, NEO_GRB +
NEO_KHZ800);

void setup() {
leds.begin();
leds.show(); // Inicializa todos los píxeles a 'apagado'
}

void loop() {
for (int i = 0; i < numPixels; i++) {
leds.setPixelColor(i, 50, 0, 0); //(Nº pixel, rojo, verde, azul)
leds.show();
delay(retardo);
}
for (int i = numPixels; i >= 0; i--) {
leds.setPixelColor(i, 0, 0, 0);
leds.show();
delay(retardo);
}
}
```


23. Sensores de sonido (de señal digital y de señal analógica)

Este sensor de sonido nada más nos sirve para captar una **señal digital** si un sonido supera un umbral. Servirá como “interruptor por sonido”, para activar un circuito con una palmada, por ejemplo.



Sensor de Sonido KY-038 y KY-037 (modo digital)



El siguiente sensor de sonido capta **señal analógica** a través del pin OUT:



> > > Hay que tener en cuenta que a más sonido, la señal de entrada analógica a Arduino disminuye, va al revés

Calibración de los sensores:

Hay que posicionar el potenciómetro hasta la zona donde se encienden los dos LEDs integrados, luego girar hasta el punto en el que se apaga uno de ellos. Se puede comprobar que al haber un sonido más intenso, se debe encender el LED indicador..

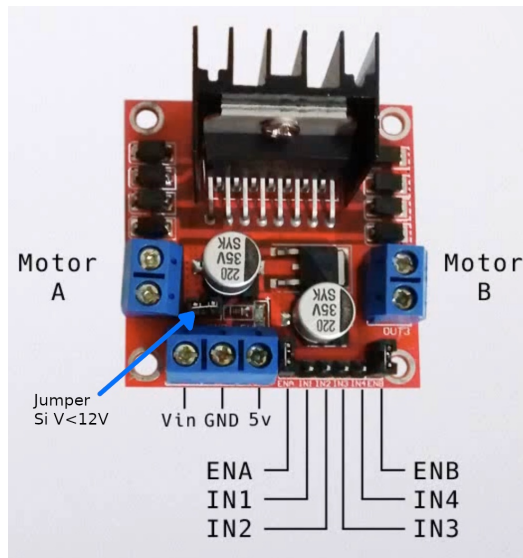
24. Distintas formas de alimentar Arduino.

Consultar: [Alimentar el Arduino: La guía definitiva](#)

25. Controlador de Motores DC L298N

Lo usaremos para controlar motores de corriente continua (DC) o paso a paso.
Características:

- $V_{in} = 5$ a 35 V
- Se pueden conectar motores de 3 a 30 V y de máximo 2A.
 - Para un motor a 6V, la tensión de alimentación debe ser algo superior (7,5-9 V)
 - Para motor de 12 V, la alimentación debe ser de 12V o ligeramente superior.
- El borne indicado como 5V sirve para alimentar a otros dispositivos, pero **si la $V_{in} > 12V$, entonces hay que quitar el jumper** indicado en la imagen y ya no tendremos disponible la salida a 5V
- **Quitamos los jumpers ENA y ENB** para el control de los motores con Arduino



Velocidad por PWM	Control del sentido de giro		Resultado del giro
ENA	IN1	IN2	Motor A
0	-	-	Parado
0-255	0	1	↻
0-255	1	0	↻

Velocidad por PWM	Control del sentido de giro		Resultado del giro
ENB	IN3	IN4	Motor B
0	-	-	Parado
0-255	0	1	↻
0-255	1	0	↻

Código básico de ejemplo para que un motor conectado a ENA, IN1 e IN2 gire a la velocidad almacenada en la variable "velocidad", en este caso 100:

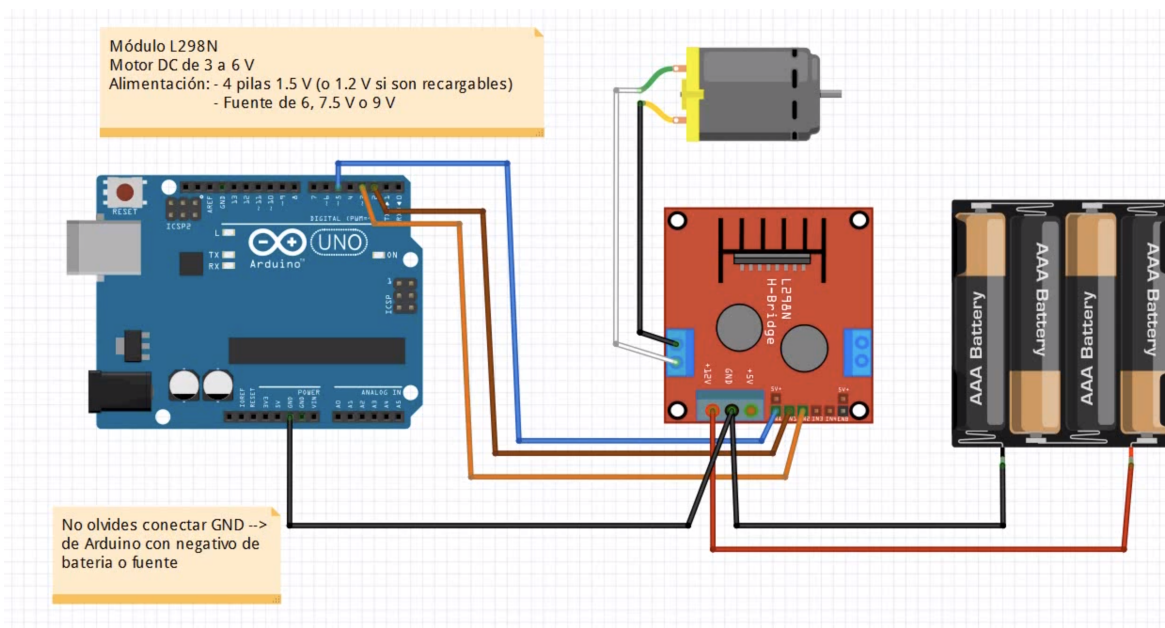
```
int IN1 = 2;
int IN2 = 3;
int ENA = 5; //Debe ser PWM
int velocidad = 100;

void setup() {
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(ENA, OUTPUT);
}

void loop() {
  //Para que gire a la velocidad almacenada en la variable "velocidad":
  analogWrite(ENA, velocidad);
  // Para que gire hacia delante (IN1=0 y IN2=1):
  digitalWrite(IN1, 0);
  digitalWrite(IN2, 1);
}
```

26. Motores de corriente continua.

- Alimentación de 3 a 6 V
- Con reductora de velocidad 1:48



Fuente: [Arduino desde cero en Español - Capítulo 19 - L298N Controlador de Motores DC \(y velocidad PWM\)](#)

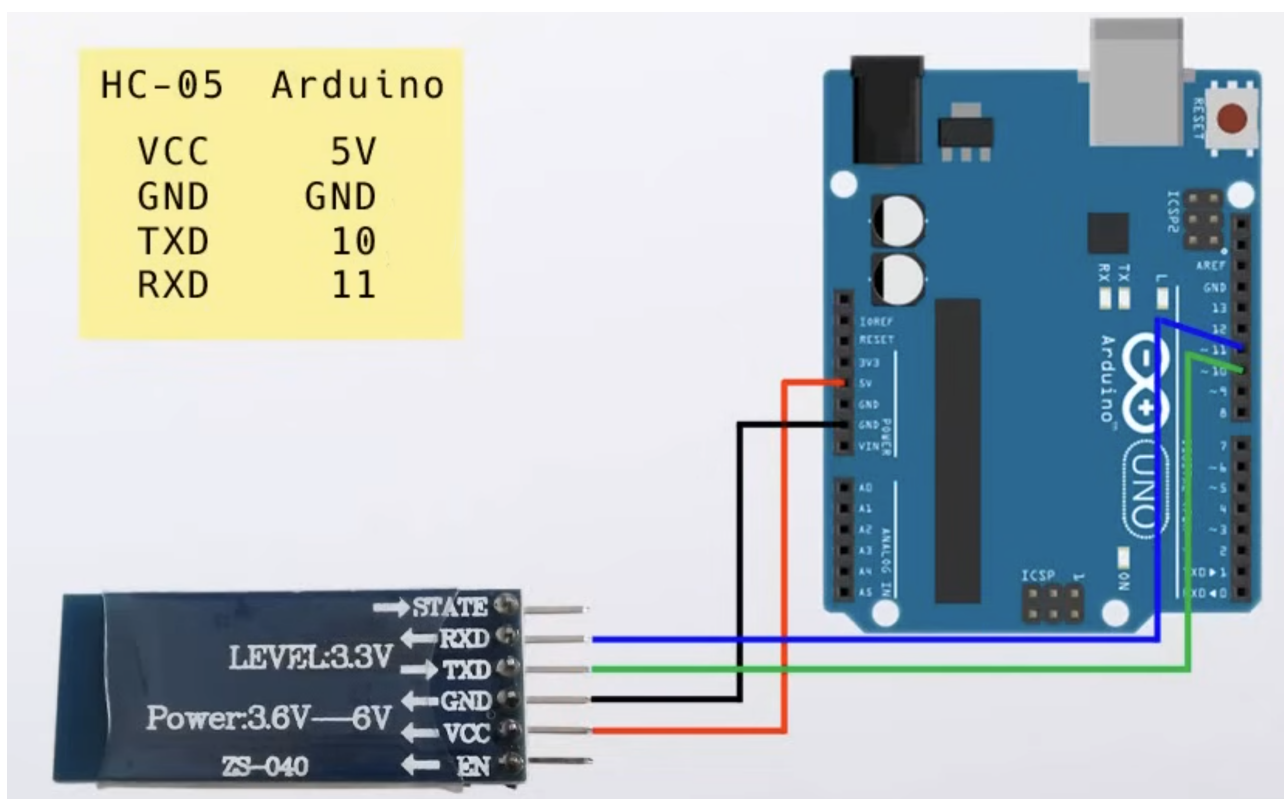
27. Bluetooth. Módulo HC-05

Para asignar un nombre o identificador al módulo, contraseña, configurarlo como maestro o esclavo, etc, debemos entrar en el modo configuración.

Modo configuración del módulo Bluetooth HC-05.

Antes de realizar nuestro proyecto, si queremos configurar el módulo HC-05 con otro nombre, contraseña, etc, debemos entrar en el modo configuración.

Para ello, realizamos el siguiente montaje:



Fuente: *Arduino desde cero en Español - Capítulo 24 - Bluetooth HC-05 Introducción y comandos AT*

Después cargamos el siguiente código en Arduino:

```
#include <SoftwareSerial.h>

SoftwareSerial miBT(10, 11); // pin 10 al TX del HC-05, pin 11 al RX

void setup(){
  Serial.begin(9600);          // comunicacion de monitor serial a 9600 bps
  Serial.println("Listo");    // escribe Listo en el monitor
  miBT.begin(38400);          // comunicacion serie entre Arduino y el modulo a 38400 bps
}

void loop(){
  if (miBT.available())       // si hay informacion disponible desde modulo
    Serial.write(miBT.read()); // lee BT y envia a monitor de Arduino

  if (Serial.available())     //si hay informacion disponible desde el monitor serial
    miBT.write(Serial.read()); // lee monitor serial y envia a Bluetooth
}
```

A continuación, para entrar la placa HC-05 en el modo configuración ([ver en vídeo](#)):

- Desconectar el módulo de la alimentación (Vcc)
- Mantener presionado el pulsador del módulo HC-05 mientras se reconecta.
- Soltamos el pulsador una vez que los LEDs parpadeen de forma lenta, entonces estaremos en el modo configuración y podremos enviar por el monitor serie los comandos AT.

Listado de comandos AT que más vamos a utilizar ([ver en vídeo ejemplo de uso](#)):

- **AT** Comando de prueba, debería responder con OK
- **AT+ROLE=1** Comando para colocar el módulo en modo Maestro (Master)
- **AT+ROLE=0** Comando para colocar el módulo en modo Esclavo (Slave)
- **AT+NAME=<Nombre>** configura el nuevo nombre que llevará el módulo HC-05
- **AT+PSWD=<número 4 dígitos>** cambiar la contraseña (PIN) del módulo bluetooth
- **AT+RESET** vuelve al modo usuario

