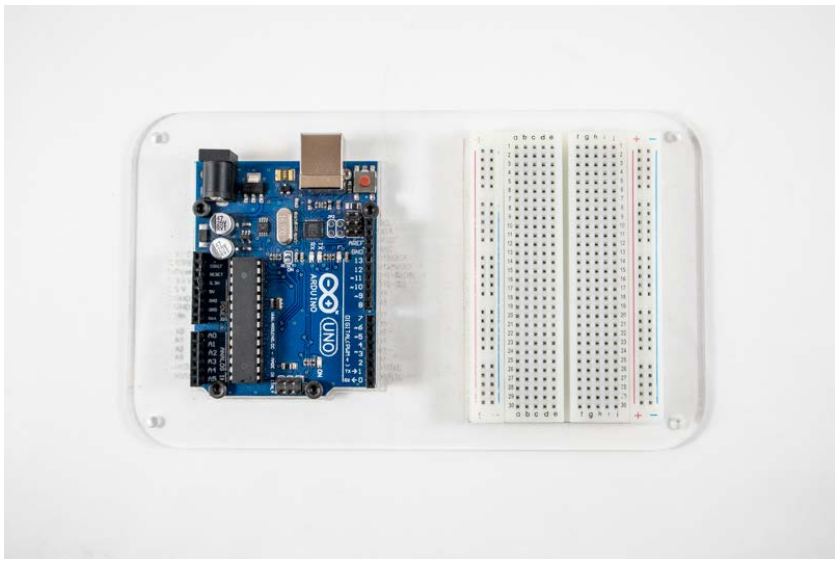
MAKER UNO

GUÍA DE APRENDIZAJE

ÍNDICE

Introducción: Preparación del Hardware y Software

Parte 1: Preparación de la UNO R3 y protoboard

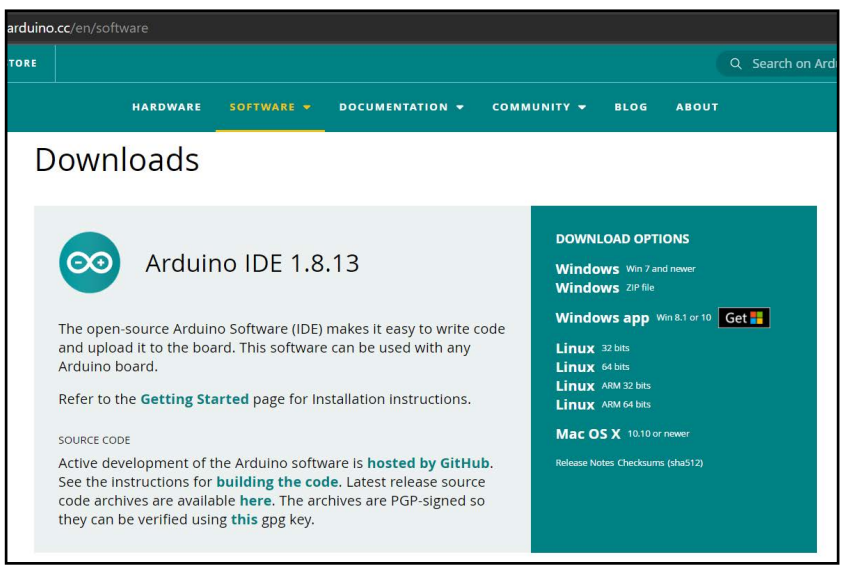


Parte 2: Descargar e instalar Arduino IDE

Ir a www.arduino.cc/en/software

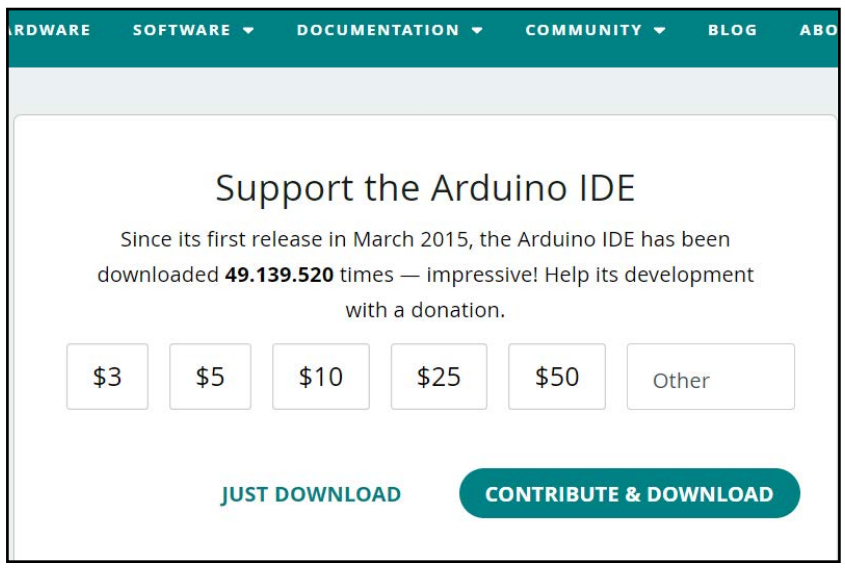
1. Selecciona el sistema operativo correspondiente a tu ordenador.

Nota: Para usuarios de Windows se recomienda la opción de "Win 7 and newer"



2. Arduino IDE es un software de código abierto. Sólo hay que hacer click en "Just Download" y podrás usarlo gratis.

Nota: Si el usuario desea, puede hacer una donación para que se pueda seguir financiando el desarrollo



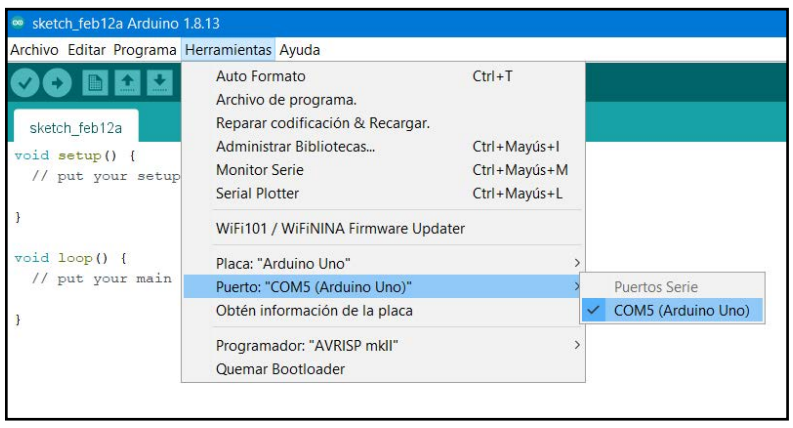
3. Doble click en el archivo descargado para instalarlo.

4. Una vez que se instale aparecerá el icono de Arduino, sólo hay que hacer doble click para abrir Arduino IDE.

3. Archivo descargado

4. Icono de Arduino IDE

5. Ahora hay que abrir el arduino IDE, ir a la sección Herramientas>Puertos>COM# y seleccionar el puerto correspondiente.

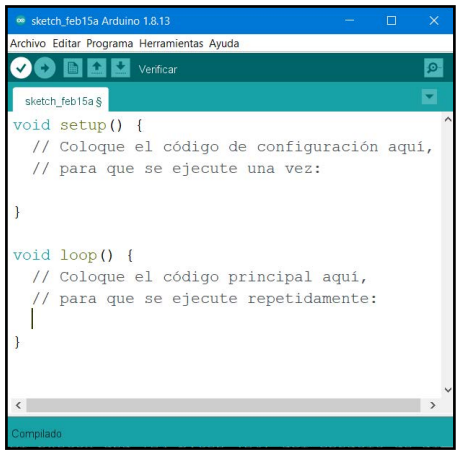


Lección 1 Salida digital

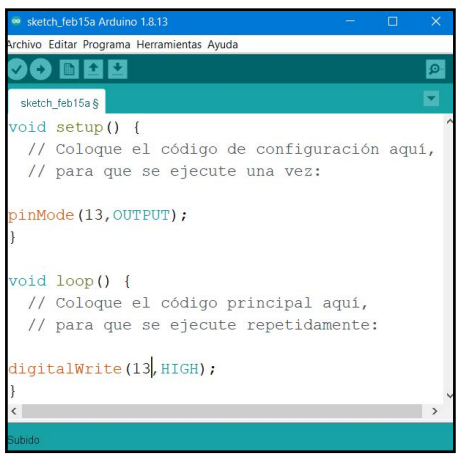
Proyecto 1: Encender un LED

1. Conecta el UNO R3 a tu computadora con el cable USB.

2. Abrir el Arduino IDE, te aparecerá un Sketch para empezar a programar.



3. Escribe el siguiente código en el IDE de Arduino y después haz click en el botón  para compilar el código. Espera unos segundos hasta que aparezca el mensaje “Compilado” que aparecerá en la parte inferior.



SOLUCIÓN DE PROBLEMAS

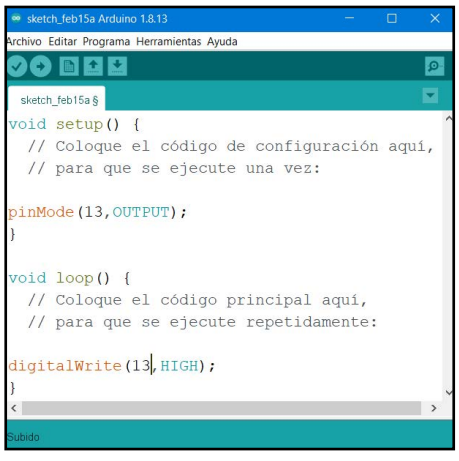
Si hay un error, deberás revisar y corregir el código línea por línea y luego compilarlo.

Arduino reconoce las mayúsculas y minúsculas, asegúrate de usar las mayúsculas para OUTPUT y usar la mayúscula marcada para pinMode y para digitalWrite.

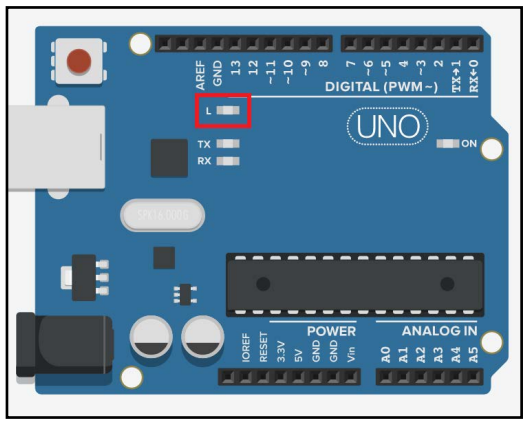
Revisa que tengas todos los signos colocados correctamente y no olvidarás alguno - ; , () y {}.

No confunda los paréntesis () con las llaves {}

4. Haz click en  para cargar el programa, este se cargará inmediatamente. Una vez que se complete la carga aparecerá “Subido” en la parte inferior.



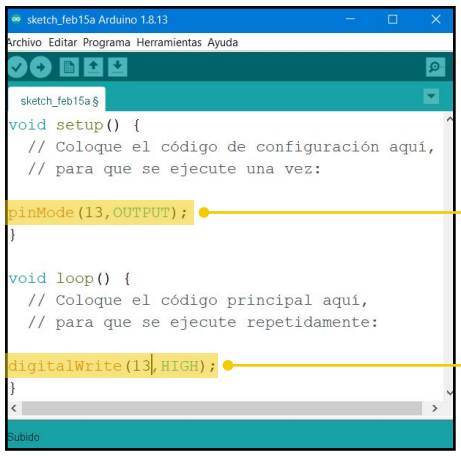
5. Ahora verifica los resultados en tu UNO R3. Observa el led que se encuentra al lado del pin 13 y verás que se encuentra encendido, si lo reseteas se apagará y volverá a encender



SOLUCIÓN DE PROBLEMAS

Si sucede un error al momento de cargar el programa, cerciorarse que el UNO R3 está conectado a tu computadora y que tengas seleccionado el puerto correcto al que está conectado tu UNO R3. A través de Herramientas > Puerto > COM#.

¿CÓMO FUNCIONA?



Esta línea le dice a la placa que el pin 13 está establecido como una salida.

Esta línea le dice a la placa que el pin 13 está establecido en *High* o Alto, esto significa Encendido

DATOS IMPORTANTES

Función de Arduino

1. Para declarar un pin como salida o entrada:
pinMode(pin,mode)
pin: El número de pin que se quiere configurar.
mode: INPUT(Entrada), OUTPUT(Salida) o INPUT_PULLUP

2. Para declarar el pin en HIGH (Encendido) o LOW (Apagado)
digitalWrite(pin,value)
pin: El número de pin que se quiere configurar.
value: HIGH or LOW

Proyecto 2: Parpadeo de un LED

1. Modifica tu código del proyecto 1 con el siguiente código:

```
void setup() {
  // Coloque el código de configuración aquí,
  // para que se ejecute una vez:
pinMode(13,OUTPUT);
}

void loop() {
  // Coloque el código principal aquí,
  // para que se ejecute repetidamente:
digitalWrite(13,HIGH);
delay(100);
digitalWrite(13,LOW);
delay(100);
}
```

2. Compila y carga el código a tu tarjeta UNO R3.
3. Verifica tu resultado.

Da click aquí para ver el video demostrativo

El LED del pin 13 estará parpadeando continuamente

4. Cambia el valor del Delay a 1000 y después carga el código a la placa.

```
void setup() {
  // Coloque el código de configuración aquí,
  // para que se ejecute una vez:
pinMode(13,OUTPUT);
}

void loop() {
  // Coloque el código principal aquí,
  // para que se ejecute repetidamente:
digitalWrite(13,HIGH);
delay(1000);
digitalWrite(13,LOW);
delay(1000);
}
```

¿CÓMO FUNCIONA?

```
void setup() {
  // Coloque el código de configuración aquí,
  // para que se ejecute una vez:
pinMode(13,OUTPUT);
}

void loop() {
  // Coloque el código principal aquí,
  // para que se ejecute repetidamente:
digitalWrite(13,HIGH);
delay(1000);
digitalWrite(13,LOW);
delay(1000);
}
```

Esta línea configura el pin 13 de la placa como una salida.

Enciende el Led del pin 13
Lo mantiene por 1000ms (microsegundos

Apaga el Led del pin 13
Lo mantiene por 1000ms (microsegundos)

El programa continuará en bucle sin fin

DATOS IMPORTANTES

Función de Arduino

1. Para mantener el programa usar:

delay(ms)

ms: el número en milisegundos que se va a pausar

DESAFÍO

Tarea: Programar el LED 13 para que haga diferentes parpadeos en diferente cantidad de tiempo como en el video mostrado.

Da click aquí para ver el video demostrativo

¡Felicidades! Has completado la lección 1 y esto es lo que has aprendido:

- Cómo configurar un pin para que sea una salida digital.
- Cómo configurar la salida digital en HIGH o LOW.
- Cómo se usa la función de Delay.

Lección 2 Entrada digital

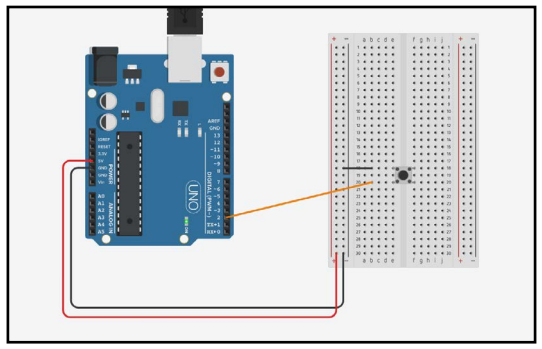
Proyecto 3: Push Button en modo Pull Up

En este proyecto, controlarás el encendido y apagado del LED que tiene la placa usando un push button en modo pull up.

1. Consigue estos componentes:

Cable jumper Macho-Macho
Push button

2. Construye el circuito que se muestra:



3. Abre el IDE de Arduino, crea un nuevo sketch y escribe el siguiente código:

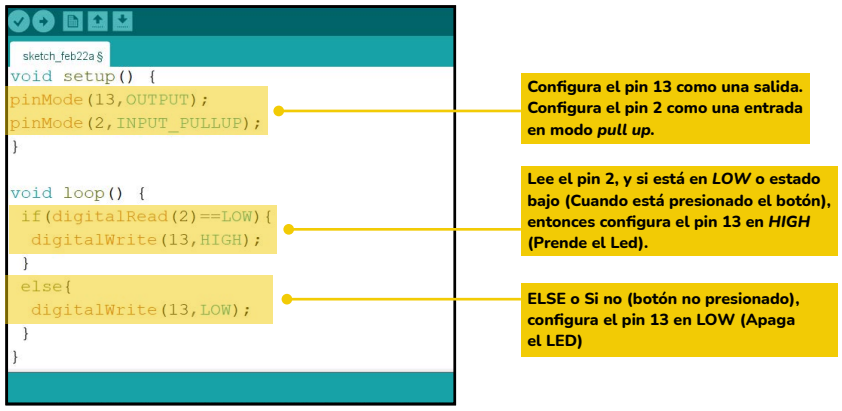
```
void setup() {
pinMode(13,OUTPUT);
pinMode(2,INPUT_PULLUP);
}

void loop() {
 if(digitalRead(2)==LOW){
  digitalWrite(13,HIGH);
 }
 else{
  digitalWrite(13,LOW);
 }
}
```

4. Compila y carga el programa, después checa tu resultado.

Da click aquí para ver el video demostrativo

El LED del pin 13 se encenderá cuando presiones el botón

¿CÓMO FUNCIONA?



DATOS IMPORTANTES

Función de Arduino

1. Para usar un push button sin resistencia tenemos que configurar el pin en modo pullup para que use la resistencia que tiene el UNO R3.

2. El LED integrado en el pin 13 puede actuar como indicador de entrada cuándo se configura de este modo. El LED 13 se apagará cada vez que se presione el push button.

Sintaxis de codificación

1. Para usar el condicional "if-else"

```
if (condición 1){
        // haz la cosa A
    }
    else if (condición 2){
      // haz la cosa B
    }
    else {
      // haz la cosa C
    }
```

2. Puedes usar "//" (dos barras juntas) para dejar un recordatorio o comentario mientras programas. Todo lo escrito después de este signo será ignorado por el programa y no será ejecutado.

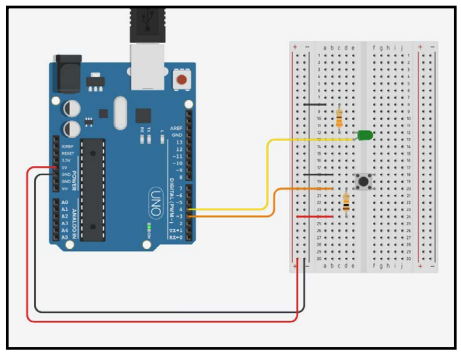
```
// Tus comentarios aquí.
```

Proyecto 4: Construcción de un circuito LED

En este proyecto construirás un circuito que use un push button con resistencia y que encienda y apague un LED.

1. Consigue estos componentes:

Cable Jumper
Push button
Resistencia de 10k Ohms
Resistencia de 330 Ohms
LED



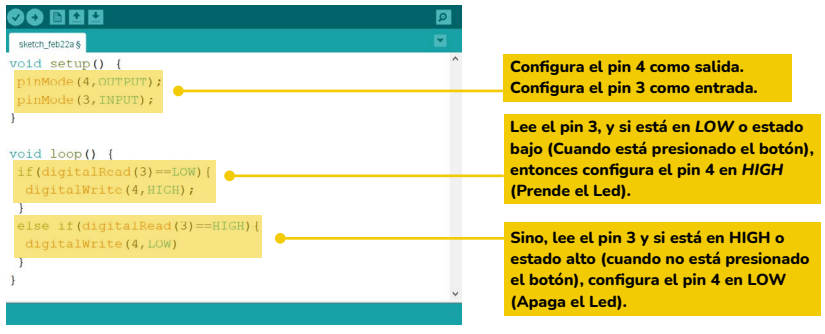
3. Modifica el código del proyecto 3 con el siguiente y cárgalo.

4. Verifica tu resultado.

Da click aquí para ver el video demostrativo

Cuando presiones el push button el LED que se conectó al pin 13 se encenderá

¿CÓMO FUNCIONA?



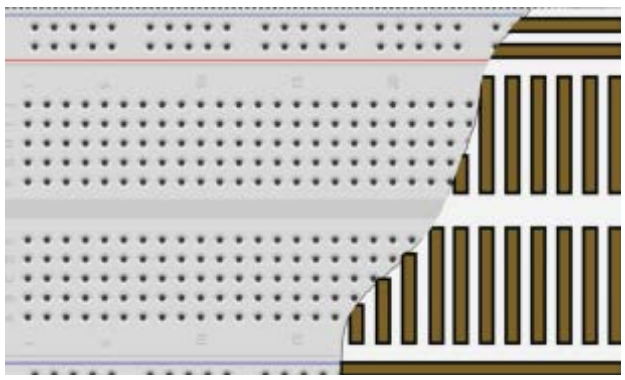
DATOS IMPORTANTES

Función de Arduino

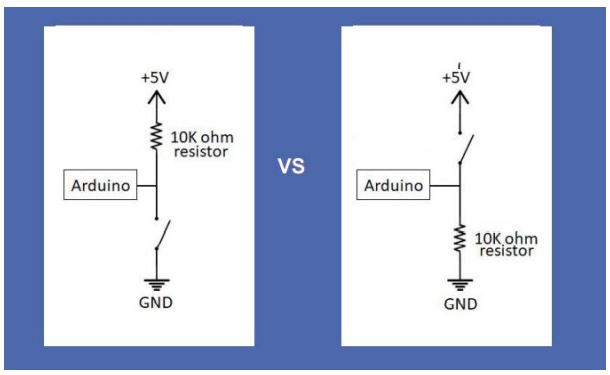
1. El push button es un dispositivo de entrada, necesitas definir el pin al que esté conectado como una entrada antes de ser conectado.
pinMode(pin,INPUT);
2. El push button necesita una resistencia para poder funcionar de manera correcta en caso de no tener la resistencia, se deberá configurar el pin al que esté conectado como pull up.
pinMode(pin,INPUT_PULLUP);

Electrónica Básica

1. Conectividad interna de la tabla de conexiones



2.Para cualquier entrada digital, puedes hacerla como un circuito “pull-up o pull-down”.



Se mantendrá en HIGH mientras el switch no esté presionado.

Se mantendrá en LOW mientras el switch no esté presionado

En este proyecto usamos el circuito pull-up. Puedes elegir entre cualquiera de las configuraciones mostradas para tus proyectos futuros, solo recuerda que debes cambiar el código al tipo de configuración que elijas.

Proyecto 5: Encendido y apagado continuo de un LED

En este proyecto lo que harás será encender y apagar el LED de forma automática pero solamente presionando un botón, también usaremos la función While que explicaremos más adelante.

1.Utilizarás el circuito realizado en el proyecto 4, lo único que tendrás que hacer será modificar el código con el siguiente y cargarlo

```
sketch_feb23a §
void setup() {
 pinMode(4,OUTPUT);
 pinMode(3,INPUT);
}

void loop() {
 if(digitalRead(3)==LOW)
  while(1){
  digitalWrite(4,HIGH);
  delay(200);
  digitalWrite(4,LOW);
  delay(200);
  }
 else digitalWrite(5,LOW);
}
```

2. Revisa tu resultado.

Da click aquí para ver el video demostrativo

El led del pin 4 parpadeara después de que se active el interruptor presionado y no dejará de parpadear a menos que se presiona el botón de reinicio.

¿CÓMO FUNCIONA?

```
sketch_feb23a §
void setup() {
 pinMode(4,OUTPUT);
 pinMode(3,INPUT);
}

void loop() {
 if(digitalRead(3)==LOW)
  while(1){
  digitalWrite(4,HIGH);
  delay(200);
  digitalWrite(4,LOW);
  delay(200);
  }
 else digitalWrite(5,LOW);
}
```

Configura el pin 4 como salida, conecta un LED externo a ese pin.
Configura el pin 3 como entrada para el push button.

Lee el pin 3. Si está BAJO (interruptor presionado), establezca el Pin 4 en HIGH durante 200ms y luego BAJO durante 200 ms.
***Una vez que el push button es presionado, el led encenderá y apagará continuamente y no se detendrá hasta que se presione el botón de reset.**

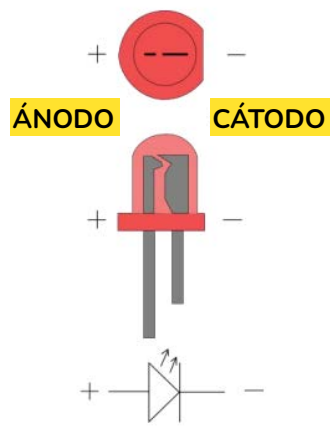
De lo contrario, si el push button no está presionado, configure el Pin 4 en LOW (el LED conectado al Pin 4 no se encenderá).

DATOS IMPORTANTES

Electrónica Básica

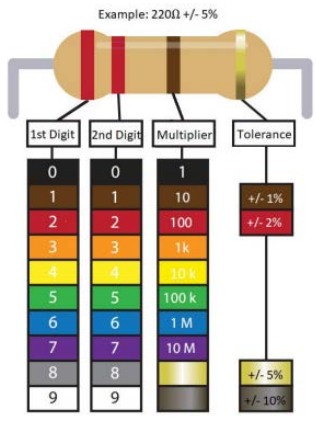
1. El Diodo Emisor de Luz (LED) cuenta con 2 patas. Se debe conectar la pata más larga a la terminal positiva (+) y la pata más corta a la terminal negativa (-). También se puede diferenciar cual es el negativo ya que el led tiene un lado plano para indicar que esa es la terminal negativa.

Así se polariza el led de manera correcta, el LED debe estar conectado de manera correcta ya que si se invierte no funcionara o podría dañarse:



2. Las resistencias no tienen polaridad, por lo cual se pueden conectar los pines en cualquier dirección, sin que afecte su funcionamiento.

3. El color en las bandas que tiene la resistencia nos dicen cuál es el valor que tiene. Puedes usar la tabla mostrada como referencia para entender cómo el valor de las resistencia y cual es el valor de cada color.



Sintaxis de codificación

Un ciclo While se repetirá continuamente e indefinidamente, hasta que la expresión dentro del paréntesis () sea falsa. While = mientras.

```
while(condición)
{
// haz algo
}
```

Ejemplo:
Haz algo 200 veces

```
var=0;
while(var<200)
{
// haz algo
var++;
}
```

DESAFÍO

Tarea: Usa 2 push button y 2 LEDs para realizarlo. Se deberá realizar lo siguiente, cuando los dos botones no estén presionados, el LED A y el LED B (tu defines en que pines los conectaras) deberán estar encendidos, cuando se presione el Switch A, el LED A se deberá apagar y cuando se presione el switch B, el LED B se deberá apagar.

Recuerda definir bien los pines de entrada y salida en tu código, para que no tengas ningún error al momento de cargarlo.

Da click aquí para ver el video demostrativo

¡Felicidades! Has completado la lección 2 y esto es lo que has aprendido:

- Cómo leer la señal de una entrada digital.
- Cómo controlar un LED usando un switch.
- Cómo construir un circuito LED simple.
- Cómo construir un circuito pull-up o pull-down simple.
- Cómo utilizar el condicional if - else.

Lección 3 Salidas analógicas

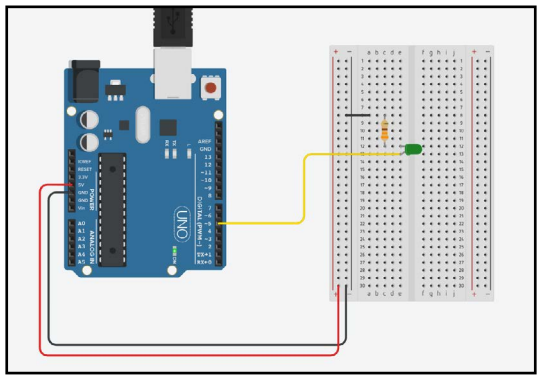
Proyecto 6: Apagado gradual de un LED

En este proyecto harás un circuito simple con un LED.

1. Consigue los siguientes componentes:

Cable Jumper
LED
Resistencia 330 Ohms

2. Realiza el circuito que se presenta a continuación.



3. Escribe y carga el siguiente código a tu placa.

```
sketch_feb23a §
void setup() {
 pinMode(5,OUTPUT);
}

void loop() {
analogWrite(5,60);
delay(200);
analogWrite(5,50);
delay(200);
analogWrite(5,40);
delay(200);
analogWrite(5,30);
delay(200);
analogWrite(5,20);
delay(200);
analogWrite(5,10);
delay(200);
analogWrite(5,0);
delay(1000);
}
```

4. Verifica tu resultado.

Da click aquí para ver el video demostrativo

Vas a notar que el LED se enciende y después va disminuyendo el brillo gradualmente hasta que se apague completamente. Este ciclo se repetirá hasta que desconectes tu UNO R3 de la luz eléctrica.

¿CÓMO FUNCIONA?

```
sketch_feb23a §
void setup() {
 pinMode(5,OUTPUT);
}

void loop() {
analogWrite(5,60);
delay(200);
analogWrite(5,50);
delay(200);
analogWrite(5,40);
delay(200);
analogWrite(5,30);
delay(200);
analogWrite(5,20);
delay(200);
analogWrite(5,10);
delay(200);
analogWrite(5,0);
delay(1000);
}
```

- `pinMode(5,OUTPUT);` — Configura el pin 5 como salida (Led conectado en el pin 5)
- `analogWrite(5,60); delay(200);` — Enciende el LED a un 60% de su brillo durante 200 milisegundos
- `analogWrite(5,50); delay(200);` — Enciende el LED a un 50% de su brillo durante 200 milisegundos
- `analogWrite(5,40); delay(200);` — Enciende el LED a un 40% de su brillo durante 200 milisegundos
- `analogWrite(5,30); delay(200);` — Enciende el LED a un 30% de su brillo durante 200 milisegundos
- `analogWrite(5,20); delay(200);` — Enciende el LED a un 20% de su brillo durante 200 milisegundos
- `analogWrite(5,10); delay(200);` — Enciende el LED a un 10% de su brillo durante 200 milisegundos
- `analogWrite(5,0); delay(1000);` — Apaga el Led (brillo al 0%) durante 1 segundo

5. Existe una forma más sencilla de lograr el mismo resultado. Modifica tu código con el siguiente y después cárgalo a la placa.

```
sketch_feb23a §
int brillo = 60;

void setup() {
 pinMode(5,OUTPUT);
}

void loop() {
analogWrite(5,brillo);
delay(200);
 if (brillo==0){
  delay(1000);
  brillo=60;
 }
 else{
  brillo = brillo - 10;
 }
}
```

6. Verifica tu resultado. ¿Obtuviste el mismo de antes?

Da click aquí para ver el video demostrativo

¿CÓMO FUNCIONA?

```
int brillo = 60;

void setup() {
 pinMode(5,OUTPUT);
}

void loop() {
analogWrite(5,brillo);
delay(200);
 if (brillo==0){
  delay(1000);
  brillo=60;
 }
 else{
  brillo = brillo - 10;
 }
}
```

Define una variable (brillo) y asignarle un valor de 60 a esta misma.

Configurar el pin 5 como una salida

Enciende el LED en el valor actual asignado a la variable 'brillo'. Mantener durante 200 ms

De lo contrario, si el valor de brillo no es 0, actualiza el valor de la variable 'brillo' con valor actual menos 10 (atenúe el brillo del Led)

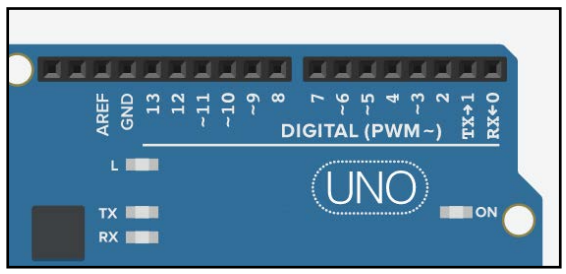
DATOS IMPORTANTES

Funciones de Arduino

Hay 2 tipos de salidas, digital y analógica. Una salida digital solo tiene dos valores posibles, los cuales son 0 (LOW) o 1 (HIGH). El valor de las salidas análogas pueden estar en un rango de 0 a 250. Puedes usar la siguiente función para controlar la salida analogica.

analogWrite(pin,value);

Sin embargo, no todos los pines del arduino se pueden usar como salidas analogicas. Solo los pines 3, 5, 6, 9, 10 y 11 (Marcadas con el signo ~) tienen funcionalidad de salida analógica.



Sintaxis de codificación

1. Lo bello de codificar es que permite al programador simplificar una instrucción larga en unas cuantas líneas de código y que realicen la misma tarea.

2. Reducir el número de líneas también ayuda a reducir el tiempo de procesamiento, por lo que es muy importante optimizar siempre tu código.

3. En este proyecto, queremos reducir el brillo del LED cada 200ms. Entonces la definimos como una variable entera al comienzo del programa. "brillo" es solo un nombre de variable; puedes elegir ponerle el nombre que desees.

4. Los siguientes son comandos de comparación que puedes usar en tu código:

x == y (x es igual a y)
x != y (x es diferente a y)
x < y (x es menor que y)
x > y (x es mayot que y)
x <= y (x es menor o igual que y)
x >= y (x es mayor o igual que y)

DESAFÍO

Tarea: Hacer el apagado gradual de un led pero solo cuando se presione un push button.

Da click aquí para ver el video demostrativo

¡Felicidades! Has completado la lección 3 y esto es lo que has aprendido:

- Cómo usar el bucle while
- La diferencia entre una salida digital y una analógica
- La importancia de reducir el número de líneas de código

Lección 4 Tonos de melodía

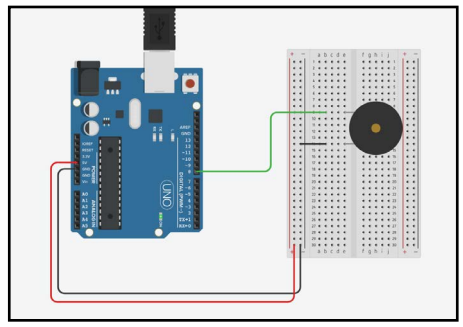
Proyecto 7: Componer tonos básicos

Lo que harás será programar tonos básicos en el UNO R3 para que salgan por medio de un buzzer y así entender cómo es que podemos producir tonos o sonidos con la placa.

1. Consigue los siguientes componentes:

Cable Jumper
Buzzer pasivo

2. Realizar el circuito que se presenta a continuación.



3. Abre un nuevo proyecto, escribe el siguiente código y después cárgalo a la placa.

```
void setup() {
 pinMode(8,OUTPUT);
}

void loop() {
 tone(8,262,250);
 delay(325);
 tone(8,294,250);
 delay(325);
 tone(8,330,250);
 delay(325);
 tone(8,349,250);
 delay(325);
 tone(8,392,250);
 delay(325);
 tone(8,440,250);
 delay(325);
 tone(8,494,250);
 delay(325);
 tone(8,523,250);
 delay(1000);
}
```

4. Verifica tu resultado.

Da click aquí para ver el video demostrativo

El buzzer seguirá reproduciendo los tonos básicos "Do Re Mi Fa So La Ti Do" repetidamente

¿CÓMO FUNCIONA?

```
void setup() {
 pinMode(8,OUTPUT);
}

void loop() {
 tone(8,262,250);
 delay(325);
 tone(8,294,250);
 delay(325);
 tone(8,330,250);
 delay(325);
 tone(8,349,250);
 delay(325);
 tone(8,392,250);
 delay(325);
 tone(8,440,250);
 delay(325);
 tone(8,494,250);
 delay(325);
 tone(8,523,250);
 delay(1000);
}
```

Configura el pin 8 como salida (Buzzer conectado a este pin).

Reproduce el tono "Do" de 262Hz durante 250 milisegundos

Reproduce el tono "Re" de 294Hz durante 250 milisegundos

Reproduce el tono "Mi" de 330Hz durante 250 milisegundos

Reproduce el tono "Fa" de 349Hz durante 250 milisegundos

Reproduce el tono "Sol" de 392Hz durante 250 milisegundos

Reproduce el tono "La" de 440Hz durante 250 milisegundos

Reproduce el tono "Do" de 494Hz durante 250 milisegundos

Reproduce el tono "Do" de 523Hz durante 1000 milisegundos

DATOS IMPORTANTES

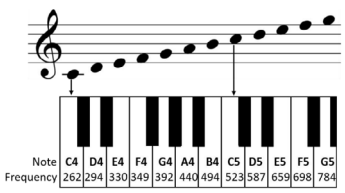
Función de Arduino

1. Para que el buzzer reproduzca un tono.
tone(pin,frecuencia,duración);
Pin: el número de pin al que está conectado el buzzer.
Frecuencia: frecuencia de la nota en hertz (Hz).
Duración: el tiempo que durará la nota en milisegundos (ms).

2. Para distinguir las diferentes notas en una melodía, necesitamos agregar un retraso entre las notas. El retraso tiene que ser un 30% mayor que la duración de la nota para reproducirla por completo. Por ejemplo, si la duración de la nota es de 250 ms, deberíamos agregar un retraso de 325 ms (250 ms + 75 ms).

Leyendo la hoja de música

1. La posición de una nota musical en el pentagrama (es decir, las cinco líneas horizontales) determina su tono. Cuanto más alta sea la nota en el pentagrama, más alta será la frecuencia del sonido y viceversa. Puedes consultar el siguiente enlace para obtener las frecuencias de las 88 teclas del piano: https://www.arduino.cc/en/Tutorial/ToneMelody



2. Se utilizan diferentes notaciones musicales para indicarnos la duración (es decir, cuánto tiempo) que debe tocarse una nota. En la biblioteca de tonos de Arduino, la duración para tocar una negra (1 tiempo) es de 250 ms; asumiendo que se toca una nota completa durante 1 segundo.

Notes	𝅝	𝅗𝅥	♩	♪	𝅘𝅥𝅯
Rests	𝄻	𝄼	𝄽	𝄾	𝄿
Relative Length	Whole Note	Half Note	Quarter Note	Eighth Note	Sixteenth Note
Beat(s)	4	2	1	1/2	1/4
Duration(ms)	1000	500	250	125	63

Proyecto 8: Componer "Happy Birthday"

Vamos a componer la primer línea de la melodía "Happy Birthday''

1. Usa el circuito del proyecto 7, modifica el código con el siguiente y cárgalo a tu placa.

```
void setup() {
 pinMode(8,OUTPUT);
  tone(8,392,125);
 delay(163);
 tone(8,392,125);
 delay(163);
 tone(8,440,250);
 delay(325);
 tone(8,392,250);
 delay(325);
 tone(8,523,250);
 delay(325);
 tone(8,494,500);
 delay(650);
}

void loop() {
}
```

2. Verifica tu resultado.

Da click aquí para ver el video demostrativo

El timbre reproducirá la primera línea de la melodía "Happy Birthday" una vez

¿CÓMO FUNCIONA?

```
void setup() {
pinMode(8,OUTPUT);          // Configura el pin 8 como salida
 tone(8,392,125);           // Reproduce el tono 'G4' de 392 Hz durante 125 milisegundos
delay(163);
tone(8,392,125);            // Reproduce el tono 'G4' de 392 Hz durante 125 milisegundos
delay(163);
tone(8,440,250);            // Reproduce el tono 'A4' de 440 Hz durante 250 milisegundos
delay(325);
tone(8,392,250);            // Reproduce el tono 'G4' de 392 Hz durante 250 milisegundos
delay(325);
tone(8,523,250);            // Reproduce el tono "La" de 440Hz durante 250 milisegundos
delay(325);
tone(8,494,500);            // Reproduce el tono 'C5' de 523 Hz durante 250 milisegundos
delay(650);                 // Reproduce el tono 'B4' de 494 Hz durante 500 milisegundos
}

void loop() {
}
```

DATOS IMPORTANTES

¿Notaste que la melodía del proyecto 8 sólo sonó una vez y no se repite continuamente como los proyectos anteriores?
Esto se debe a que todo el código está escrito en *void setup ()* en lugar de *void loop ()*. En otras palabras, el código se ejecutará línea por línea al encender el Arduino y no se repetirá nuevamente a menos que se presione el botón de reinicio.

```
void setup() {
  //ponga su código de configuración aquí,
    para que se ejecute una vez
}

void loop() {
  // ponga su código principal aquí,
    para que se ejecute repetidamente

}
```

Inicio

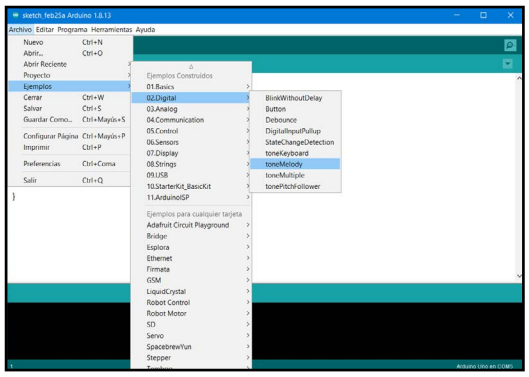
Haz esto una vez (al encender o cuando se presione el botón de reinicio)

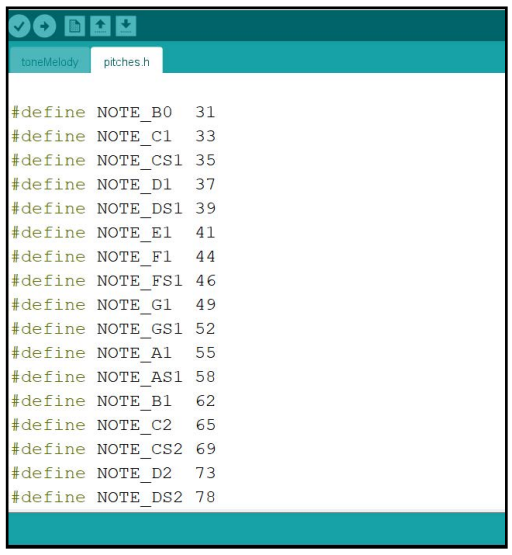
Haz esto repetidamente (hasta que se presione el botón de apagado o reinicio)

Proyecto 9: Optimiza tu código

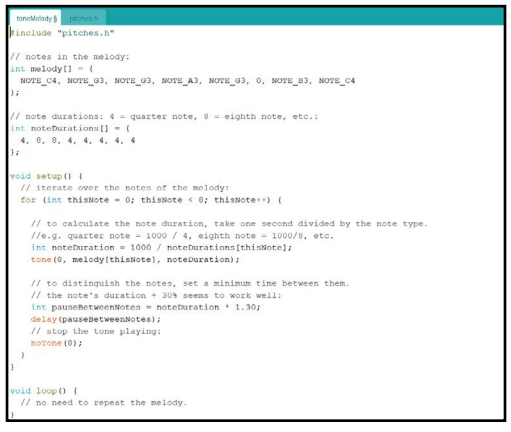
En lugar de introducir manualmente las frecuencias y duraciones de las notas línea por línea, podemos optimizar nuestro código usando un código de muestra dado. Esto nos permite componer una melodía larga con facilidad.

1. Para abrir el código de muestra *"toneMelody"* ve a **Archivo > Ejemplos > 02.Digital > toneMelody**





En el ejemplo dado, la frecuencia de cada nota está predefinida desde la nota B0 hasta DS8. Puedes hacer clic en la pestaña pitches.h para ver las frecuencias de notas predefinidas.

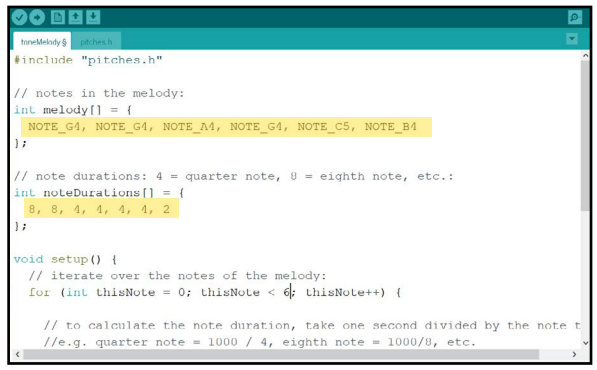


```
#include "pitches.h"

// notes in the melody:
int melody[] = {
  NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4
};

// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {
  4, 8, 8, 4, 4, 4, 4, 4
};

void setup() {
  // iterate over the notes of the melody:
  for (int thisNote = 0; thisNote < 8; thisNote++) {

    // to calculate the note duration, take one second divided by the note type.
    //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
    int noteDuration = 1000 / noteDurations[thisNote];
    tone(8, melody[thisNote], noteDuration);

    // to distinguish the notes, set a minimum time between them.
    // the note's duration + 30% seems to work well:
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    // stop the tone playing:
    noTone(8);
  }
}

void loop() {
  // no need to repeat the melody.
}
```

Este es el boceto de muestra toneMelody creado por Tom Igoe. Sube este código a tu Arduino y escucharás el final pegadizo de la canción *"Shave and a Haircut, Two Bits"*.

Modifique el boceto de muestra para reproducir la primera línea de la melodía "Feliz cumpleaños": reemplace las notas en la melodía, sus correspondientes duraciones de notas y el número total de notas para tocar, de la siguiente manera:

NOTE_	Hap- G4	-py G4	Birth- A4	-day G4	To C5	You B4
noteDurations	8	8	4	4	4	2



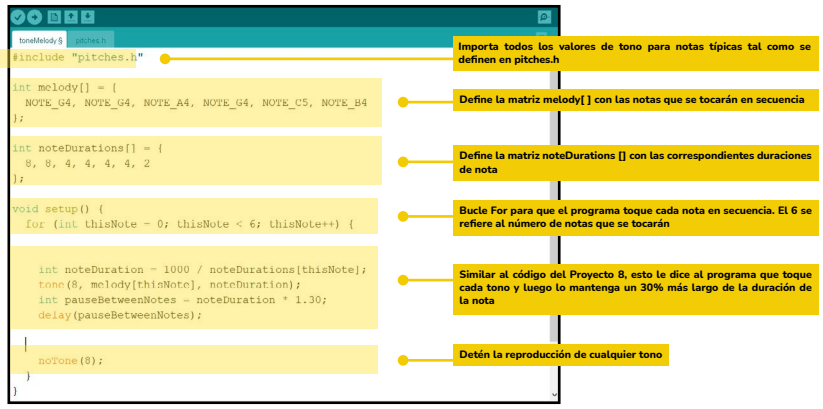
```
#include "pitches.h"

// notes in the melody:
int melody[] = {
  NOTE_G4, NOTE_G4, NOTE_A4, NOTE_G4, NOTE_C5, NOTE_B4
};

// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {
  8, 8, 4, 4, 4, 4
};

void setup() {
  // iterate over the notes of the melody:
  for (int thisNote = 0; thisNote < 6; thisNote++) {

    // to calculate the note duration, take one second divided by the note t
    //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
```

2. Compila y carga tu código. Checa el resultado.

Da click aquí para ver el video demostrativo

Similar al Proyecto 8, el timbre reproducirá la primera línea de la melodía *"Feliz cumpleaños"* una vez

¿CÓMO FUNCIONA?



DATOS IMPORTANTES

Sintaxis de codificación

1. Para utilizar la declaración "for":

for(inicializacion; condicion; incremento)

{

//declaración

}

Ejemplo:

[Imagen 63]

Estas líneas de código se pueden acortar en solo tres líneas usando la instrucción for, como se muestra a continuación:

```
for(int i=2; i<8; i++)
{
        digitalWrite(i,HIGH);
        delay(1000);
}
```

```
void loop() {
digitalWrite(2,HIGH);
delay (1000);
digitalWrite(3,HIGH);
delay (1000);
digitalWrite(4,HIGH);
delay (1000);
digitalWrite(5,HIGH);
delay (1000);
digitalWrite(6,HIGH);
delay (1000);
digitalWrite(7,HIGH);
delay (1000);
}
```

Estas líneas de código se pueden acortar en solo tres líneas usando la instrucción for, como se muestra a continuación:

```
for(int i=2; i<8; i++)
{
        digitalWrite(i,HIGH);
        delay(1000);
}
```

Cuando tenemos muchas variables del mismo tipo de datos, podemos usar una matriz e introducir todas las variables usando solo una línea de código. Así puedes usar una matriz o array:

type arrayName [] = { Lista de variables]

type: tipo de dato (int, char, float...)

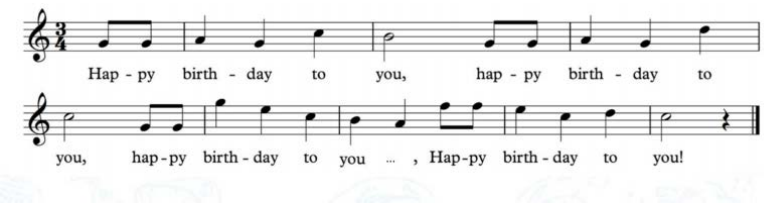
arrayName = cualquier nombre que ayude a identificar la matriz.

Ejemplo: int melody[ ] = { NOTE_C4, NOTE_D4, NOTE_E4}

DESAFÍO

Tarea: Programar el UNO R3 para que reproduzca una melodía completa de *"Feliz cumpleaños"* cuando se presiona un interruptor.

Da click aquí para ver el video demostrativo



¡Felicidades! Has completado la lección 4 y esto es lo que has aprendido:

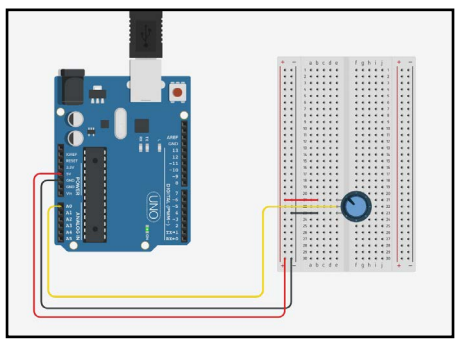
- Cómo programar el UNO R3 para que reproduzca tonos.
- Cómo componer una melodía simple usando UNO R3.
- Cómo cargar un ejemplo del programa de la librería de Arduino.
- Cómo usar la declaración "for".

Lección 5 Entrada analógica

Proyecto 10: Mostrar valor analógico en el monitor en serie

1. Consigue estos componentes.
Cables jumper
Potenciómetro

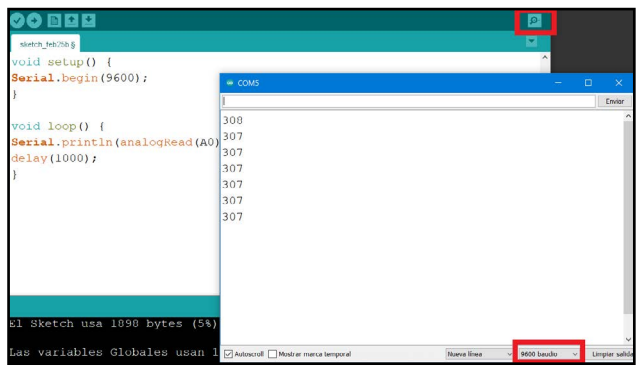
2. Construye el circuito que se muestra.



3. Escribe y carga el siguiente código a tu placa.

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

void loop() {
Serial.println(analogRead(A0));
delay(1000);
}
```

4. Una vez cargado, haga clic en el botón del monitor en serie en la barra de herramientas. Aparecerá una nueva ventana en su pantalla. Asegúrese de que la velocidad en baudios esté configurada en 9600 baudios.



5. Observe el valor que se muestra en la ventana del monitor en serie mientras gira el potenciómetro en sentido horario y luego en sentido antihorario

6. Verifica tu resultado.

Da click aquí para ver el video demostrativo

Los valores que se muestran en el monitor en serie son las lecturas analógicas proporcionadas por el potenciómetro.

¿CÓMO FUNCIONA?

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

void loop() {
Serial.println(analogRead(A0));
delay(1000);
}
```

Configura la comunicación serial

Lee el valor análogo del pin A0 y muestra el valor en la ventana de monitor serial

Espera por 1 segundo

DATOS IMPORTANTES

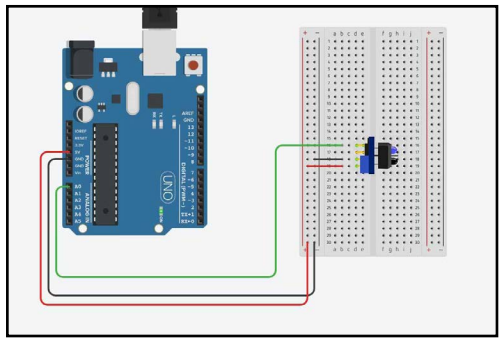
1. Solo los pines A0, A1, A2, A3, A4 y A5 tienen función de entrada analógica. Por lo tanto, necesitamos conectar sensores analógicos a estos pines si queremos obtener valores de entrada analógica.

2. La entrada digital tiene solo 2 valores posibles; 0 (LOW) o 1 (HIGH). Los valores de entrada analógica van en un rango de 0 a 1023.

Proyecto 11: Leer sensor infrarrojo analógico

1. Consigue estos componentes.
Cables jumper
Sensor IR tcrt5000

2. Construye el circuito que se muestra.



IR Sensor	VCC	GND	DO	A0
Maker UNO X	5V	GND	--	A0

3. Carga el mismo programa usado en el proyecto 10.

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

void loop() {
Serial.println(analogRead(A0));
delay(1000);
}
```

4. Una vez cargado, abriremos la ventana del monitor serial. Coloca tu mano frente al sensor. Observe el valor que se muestra en el monitor serial mientras acercas la palma de la mano hacia el sensor y luego alejala del sensor.

5. Verifica tu resultado

Da click aquí para ver el video demostrativo

¿Qué observas? ¿La lectura de infrarrojos aumenta o disminuye a medida que se mueve la mano hacia el sensor? ¿Cuál es la lectura cuando quitas la mano?

DATOS IMPORTANTES

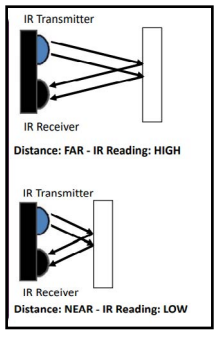
Sensor infrarrojo (IR)

1. Un sensor de infrarrojos (IR) consta de dos partes: transmisor de infrarrojos (LED de infrarrojos) y receptor de infrarrojos (fotodiodo).

2. Cuando se enciende, el transmisor emitirá luz infrarroja. Si hay un objeto colocado frente al sensor, la luz IR se reflejará de regreso al receptor.

3. La intensidad de la luz IR detectada por el receptor se convierte en un valor analógico que se puede observar en la ventana del monitor serial.

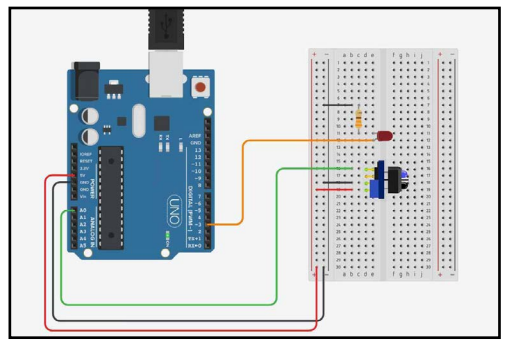
4. La lectura de infrarrojos cambia en proporción a la distancia entre el objeto y el sensor infrarrojo.



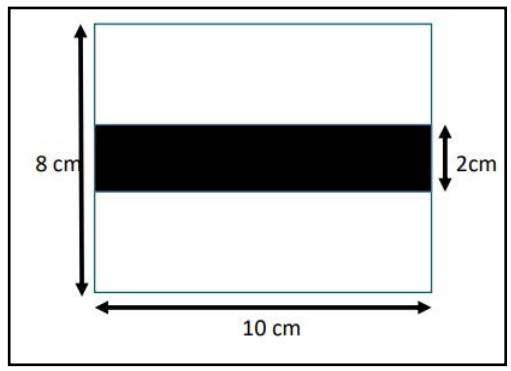
Proyecto 12: Sensor infrarrojo para detectar líneas negras

1. Consigue estos componentes.
Cables jumper
Sensor IR
LED
Resistencia 330 ohms

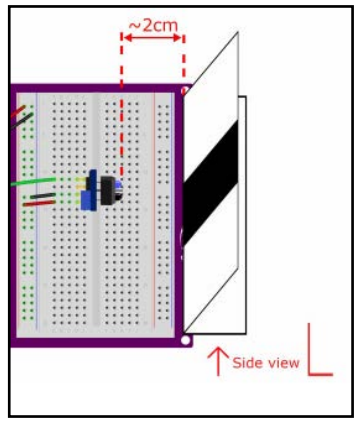
2. Construye el circuito que se muestra.



3. Prepara un trozo de cartón blanco y un rotulador negro. Utilice el rotulador negro para dibujar una línea negra gruesa (de unos 2 cm de ancho) en el centro del cartón, como se muestra a continuación.



4. Dobla el cartón y colóquelo frente al sensor de infrarrojos como se muestra a continuación. Fije la distancia entre el cartón y el sensor en aproximadamente 2 cm.



5. Usando el código del proyecto 11, abre el monitor serial y después, deslice el cartón horizontalmente, de un extremo al otro. Observe y registre las lecturas de infrarrojos del monitor serial cuando el sensor de infrarrojos esté orientado hacia la superficie blanca y luego hacia la línea negra.

Condición	Lectura IR
Superficie Blanca	
Linea Negra	

6. Cambia el código anterior por este y cárgalo a tu placa.

```
int irValue;

void setup() {
 Serial.begin(9600);
 pinMode(3,OUTPUT);
}

void loop() {
 Serial.println(analogRead(A0));
 irValue = analogRead(A0);
 if (irValue > 500){
  digitalWrite (3,HIGH);
  delay(200);
 }
 else{
  digitalWrite(3,LOW);
  delay(200);
 }
}
```

7. Verifica tu resultado.

Da click aquí para ver el video demostrativo

¿Se iluminó el LED cuando se detectó la línea negra? ¿Y se apago cuando se detecta la superficie blanca?

¿CÓMO FUNCIONA?

```
int irValue;

void setup() {
 Serial.begin(9600);
 pinMode(3,OUTPUT);
}

void loop() {
 Serial.println(analogRead(A0));
 irValue = analogRead(A0);
 if (irValue > 500){
  digitalWrite (3,HIGH);
  delay(200);
 }
 else{
  digitalWrite(3,LOW);
  delay(200);
 }
}
```

Define la variable "irValue" como entero

Configura la comunicación Serial
Configura el pin 3 como salida

Lee el valor analogico del pin A0 y muéstralo en el monitor Serial

Asignar la entrada analógica del pin A0 a la variable

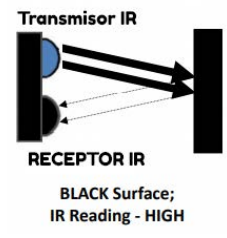
Checa irValue:
Si irValue > 500 (por ejemplo, cuando se detecta la línea negra), enciende el Led.
Mantén por 200 microsegundos

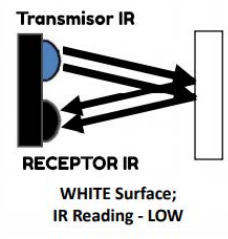
Si no, apaga el Led.
Mantén por 200 microsegundos

DATOS IMPORTANTES

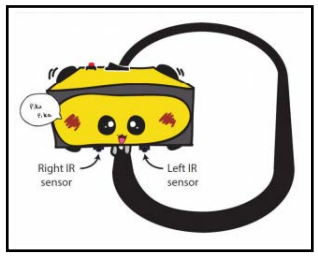
Sensor infrarrojo (IR)

Los sensores infrarrojos se utilizan comúnmente para detectar obstáculos y medir distancias. Además de eso, también podemos utilizar sensores de infrarrojos para diferenciar superficies en blanco y negro.
Esto se debe a que una superficie blanca (o de color claro) puede reflejar la mayor parte de la luz del transmisor de infrarrojos, mientras que una superficie negra (o de color oscuro) tiende a absorber la luz. Por lo tanto, el receptor de infrarrojos detectará comparativamente más luz reflejada de una superficie blanca que de una negra.





Aplicando este concepto, podemos utilizar varios sensores infrarrojos puestos en línea recta para construir un robot seguidor de línea.



Sintaxis de codificación

En este proyecto, el valor de entrada analógica del sensor infrarrojo sigue cambiando según el color del objeto al que se enfrenta actualmente. Necesitamos almacenar el valor en un lugar determinado y solo invocarlo cuando sea necesario. Para hacer eso, podemos asignar una variable.

int NombreVariable = value; **o** **int NombreVariable;**

NombreVariable puede ser cualquier palabra EXCEPTO las que ya se utilizan como palabras clave en Arduino IDE. Te recomiendo utilizar nombres descriptivos, como "irValue" o "distancia", para que puedas entender fácilmente lo que representa la variable.

DESAFÍO

Tarea: Utiliza los valores analógicos de un potenciómetro para cambiar la velocidad de parpadeo del LED 3.

Da click aquí para ver el video demostrativo

¡Felicidades! Has completado la lección 4 y esto es lo que has aprendido:

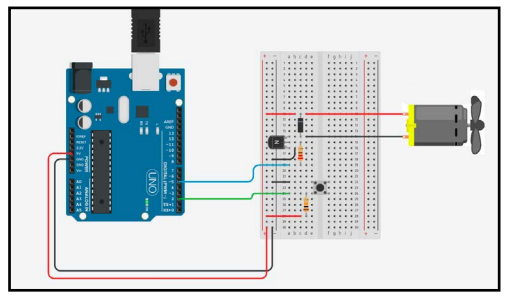
- Cómo leer una entrada analógica.
- Cómo usar el monitor serial.
- Sensor infrarrojo y cómo funciona.
- Como asignar valores a las variables.

Lección 6 Motor corriente directa (CD)

Proyecto 13: Controlar un motor CD

1. Consigue los siguientes componentes:
Cables jumper
Motor con aspas
Diodo **(La terminal con una banda blanca será conectado al cable positivo)**
Resistencia 220 Ohms
Transistor 2N2222
Push button

2. Construye el circuito como se muestra a continuación.



3. Escribe y carga el siguiente código en tu placa.

```
void setup() {
 pinMode(2,INPUT);
 pinMode(5,OUTPUT);
}

void loop() {
 if(digitalRead(2)==LOW){
  while(1){
    analogWrite(5,255);
    delay(3000);
    analogWrite(5,80);
    delay(3000);
  }
 }
 else digitalWrite(3,LOW);
}
```

4. Una vez cargado a la placa, presiona el push button y observa el resultado.

Da click aquí para ver el video demostrativo

Cuando presionas el interruptor, ¿ves que el motor comienza a girar? Continuará girando continuamente, a máxima velocidad y luego a velocidad media alternativamente, hasta que apagues o presiones el botón de reinicio.

¿CÓMO FUNCIONA?

```
void setup() {
 pinMode(2,INPUT);
 pinMode(5,OUTPUT);
}

void loop() {
 if(digitalRead(2)==LOW){
  while(1){
    analogWrite(5,255);
    delay(3000);
    analogWrite(5,80);
    delay(3000);
  }
 }
 else digitalWrite(3,LOW);
}
```

Configura el pin 2 como entrada (push button)
Configura el pin 5 como una salida

Verifique el valor del Pin 2. Si el valor del Pin 2 = LOW (es decir, se presiona el interruptor), siempre configura el Pin 5 en 255 (el motor gira a velocidad máxima) durante 3 segundos, luego configure el Pin 5 en 80 (el motor gira a velocidad media) durante 3 segundos

De lo contrario, establezca el Pin 5 en LOW (el motor está apagado; no gira)

SOLUCIÓN DE PROBLEMAS

1. Asegurate que el motor CD sea compatible. Es posible que el motor no tenga suficiente torque para girar si las cuchillas están apoyadas sobre una superficie.

2. Revisa que el programa tenga bien definidos los pines de salida y de entrada.

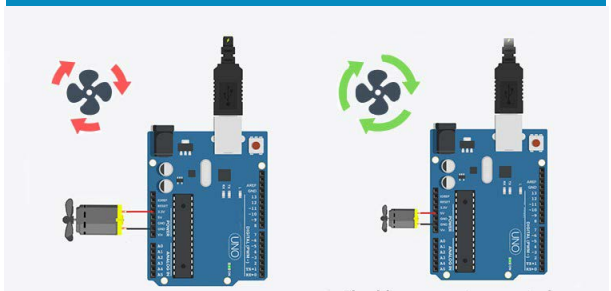
3. Si el n. ° 2 anterior está funcionando pero el motor aún no gira, debe verificar su circuito y asegurarse de que todos los componentes estén conectados correctamente.

DATOS IMPORTANTES

¿Cómo controlar la velocidad de un motor de corriente directa (DC)?

Para que un motor CD gire, debes aplicarle el voltaje de entrada adecuado. Los voltajes de entrada típicos para motores DC son 3V, 6V y 12V. El fabricante del motor generalmente indicará el voltaje de entrada recomendado en las especificaciones del producto.

Sin embargo, los motores aún pueden funcionar incluso cuando se aplica un voltaje mayor o menor que el recomendado. Para el motor de CC que estamos usando en este proyecto, el voltaje recomendado es de 3-6V. Intentemos aplicar 3.3V y 5V y veamos qué sucede.



El cable negro esta conectado a GND y el cable rojo al pin de 3.3V.

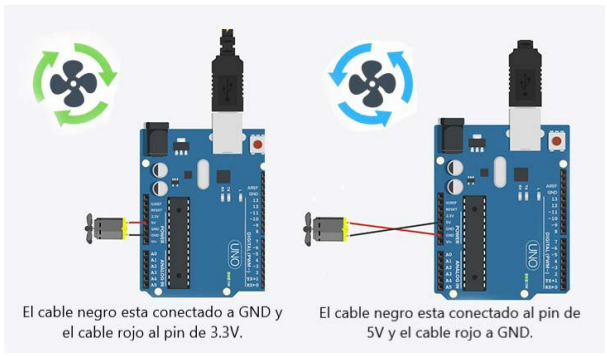
El cable negro esta conectado a GND y el rojo al pin de 5V.

Podemos observar que la velocidad del motor CD cambia con la tensión de entrada. Cuanto mayor sea el voltaje de entrada, más rápido gira el motor. Por lo tanto, en el Proyecto 13, cuando establecemos el valor del pin 5 en 255, el motor gira a toda velocidad y luego se ralentiza cuando el valor del pin 5 se establece en 80.

¡ADVERTENCIA! La aplicación de alto voltaje a un motor (más allá del voltaje recomendado) acortará el ciclo de vida del motor a largo plazo.

¿Cómo controlar la dirección de giro de un motor CD?

Cambiar la dirección de giro es fácil: simplemente invierta la polaridad del motor cambiando las conexiones del cable rojo y del cable negro.



El cable negro esta conectado a GND y el cable rojo al pin de 3.3V.

El cable negro esta conectado al pin de 5V y el cable rojo a GND.

Pudiste observar que el motor gira en sentido contrario cuando cambiamos la conexión del cable.

Sin embargo, no es práctico cambiar manualmente las conexiones de cables cada vez que desee cambiar la dirección de giro. Para controlar fácilmente tanto la dirección de giro como la velocidad de un motor CD (sin modificar el circuito), podemos usar un controlador de motor, como los siguientes:

Proyecto 14: Controlar la velocidad de un motor mediante un botón

1. Usando el proyecto anterior solo cambia el código por el siguiente y súbelo a tu placa.

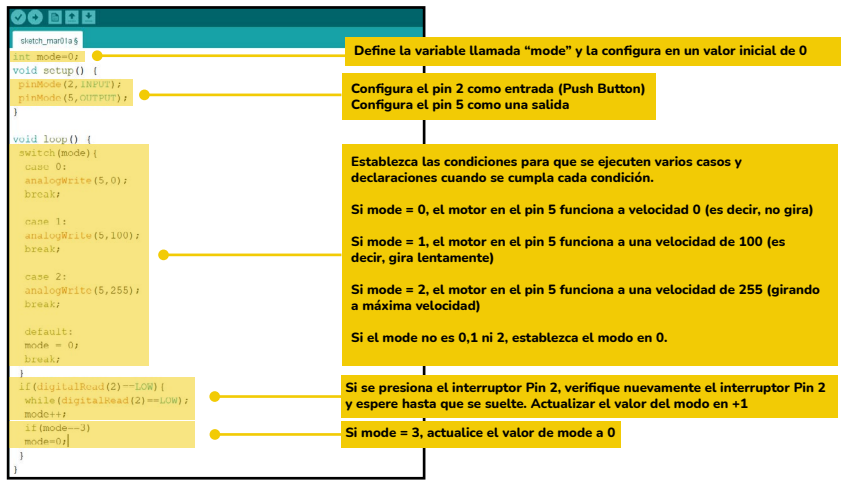
```
int mode=0;
void setup() {
 pinMode(2,INPUT);
 pinMode(5,OUTPUT);
}

void loop() {
 switch(mode){
  case 0:
  analogWrite(5,0);
  break;

  case 1:
  analogWrite(5,100);
  break;

  case 2:
  analogWrite(5,255);
  break;

  default:
  mode = 0;
  break;
 }
 if(digitalRead(2)==LOW){
  while(digitalRead(2)==LOW);
  mode++;
  if(mode==3)
  mode=0;
 }
}
```

2. Una vez subido, presiona el push button y observa el resultado. Repítelo varias veces y observa los cambios que tiene la velocidad del motor cada vez que se presiona el botón.

Da click aquí para ver el video demostrativo

Cuando presionas el interruptor, el motor comienza a girar. Y si presionas contra, el motor gira más rápido (a máxima velocidad). La tercera vez que presionas el interruptor, el motor deja de girar.

¿CÓMO FUNCIONA?



DATOS IMPORTANTES

Sintaxis de codificación

Al igual que las declaraciones if, switch ... case controla el flujo de un programa al permitir a los programadores especificar qué líneas de código se ejecutarán cuando se cumplan las condiciones establecidas. Esto es útil cuando se necesita usar el mismo conmutador para ejecutar diferentes conjuntos de programas.

```
switch(var){
case 1:
// haz esto cuando var sea igual a 1
break;
case 2:
// haz esto cuando var sea igual a 0
break;
default:
// Si nada coincide con lo anterior, haz lo que dice default
// Default es opcional
break;
}
```

Para evitar que el motor gire, puedes escribir cualquiera de estos:

```
digitalWrite(pin,LOW);
        o
analogWrite(pin,0);
```

¿Notaste que no pusimos un punto y coma después de while(1) en nuestros proyectos anteriores, pero agregamos un punto y coma después de while(digitalRead (2) == LOW) en este proyecto?

Sin el ";"

```
while(condición){
// si la condición es verdadera, el programa ejecutará todas las líneas de
código dentro de {}
}
```

Con el ";"

```
while(condición);
//si la condición es verdadera, sólo ejecutará esta línea hasta que la
condición se vuelva falsa.
```

Ejemplo:

```
while(digitalRead(2)==LOW);
//Si se presiona el interruptor 2 (la condición es verdadera), el programa
permanecerá en esta línea de código hasta que se suelte el interruptor 2 (la
condición se vuelve falsa). Entonces, y solo entonces, continuará para ejecutar la siguiente línea de código.
```

DESAFÍO

Tarea: Programa el motor para que funcione a una velocidad constante cuando mantenga presionado el botón. El motor se detiene cuando suelta el botón. Y cuando presione el botón por segunda vez, la velocidad del motor cambiará.

Da click aquí para ver el video demostrativo

¡Felicidades! Has completado la lección 6 y esto es lo que has aprendido:

- Cómo construir un circuito para hacer funcionar un motor CD.
- Cómo controlar la velocidad de un motor CD.
- Cómo utilizar la declaración switch.case.

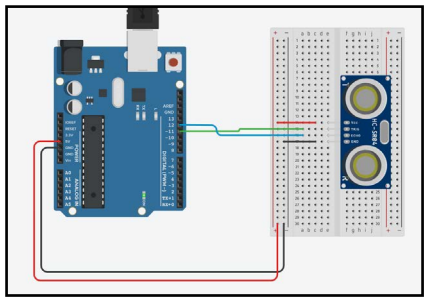
Lección 7: Sensor ultrasónico

Proyecto 15: Configuración del sensor ultrasónico

1. Consigue los siguientes componentes.

Cables jumper
Sensor ultrasónico

2. Construye el circuito que se muestra.



Sensor Ultrasonico	VCC	TRIG	ECHO	GND
UNO R3	5V	PIN 11	PIN 12	GND

Conecta el sensor en el borde de la placa de pruebas y asegúrate de que los cables de puente no bloqueen el sensor.

3. Escribe y carga el siguiente código en tu placa.

```
long duration;
int distance;
void setup() {
 pinMode(11,OUTPUT);
 pinMode(12,INPUT);
 Serial.begin(9600);
}

void loop() {
 digitalWrite(11,LOW);
 delayMicroseconds(2);
 digitalWrite(11,HIGH);
 delayMicroseconds(10);
 digitalWrite(11,LOW);
 duration = pulseIn(12,HIGH);
 distance = duration*0.034/2;
 delay(50);
 Serial.print("Distance =");
 Serial.print(distance);
 Serial.println("cm");
}
```

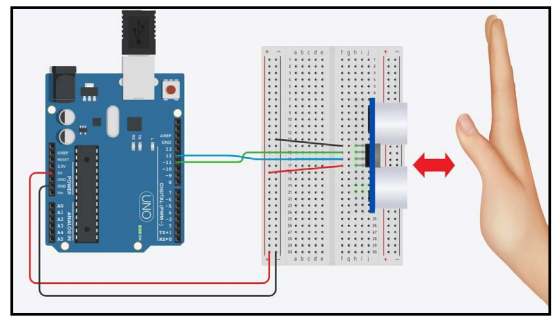
4. Una vez cargado, haz clic en el botón del monitor serial para ver el resultado. Coloca tu mano frente al sensor. Observa el valor que se muestra en la ventana del monitor serial a medida que mueves la palma de la mano hacia el sensor ultrasónico y luego alejándote.

5. Verifica tu resultado.

Da click aquí para ver el video demostrativo

¿Qué observas? ¿La lectura ultrasónica aumenta o disminuye a medida que se mueve la mano hacia el sensor? ¿Cuál es la lectura cuando quita la caja (es decir, sin obstáculos)?

¿CÓMO FUNCIONA?



6. A continuación, modifica el programa anterior con el siguiente y luego cargalo a la placa.

```
long duration;
int distance;
void setup() {
 pinMode(11,OUTPUT);
 pinMode(12,INPUT);
 Serial.begin(9600);
}

void loop() {
 ultrasonic();
 Serial.print("Distance =");
 Serial.print(distance);
 Serial.println("cm");
}

void ultrasonic(){
 digitalWrite(11,LOW);
 delayMicroseconds(2);
 digitalWrite(11,HIGH);
 delayMicroseconds(10);
 digitalWrite(11,LOW);
 duration = pulseIn(12,HIGH);
 distance = duration*0.034/2;
 delay(50);
}
```

Da click aquí para ver el video demostrativo

Deberías obtener el mismo resultado que el programa anterior

¿CÓMO FUNCIONA?

```
long duration;
int distance;
void setup() {
 pinMode(11,OUTPUT);
 pinMode(12,INPUT);
 Serial.begin(9600);
}

void loop() {
 ultrasonic();
 Serial.print("Distance =");
 Serial.print(distance);
 Serial.println("cm");
}

void ultrasonic(){
 digitalWrite(11,LOW);
 delayMicroseconds(2);
 digitalWrite(11,HIGH);
 delayMicroseconds(10);
 digitalWrite(11,LOW);
 duration = pulseIn(12,HIGH);
 distance = duration*0.034/2;
 delay(50);
}
```

Llama a la función “ultrasonic” (es decir, ejecuta todas las líneas de código en la función “ultrasonic”.)

Estas líneas de código se han agrupado en una función denominada “ultrasonic

Sensores ultrasónicos

Al igual que los sensores infrarrojos, los sensores ultrasónicos se utilizan para detectar obstáculos y medir distancias. El mecanismo de trabajo también es similar al de los sensores de infrarrojos. El transmisor emite una onda ultrasónica, que al golpear un objeto en su camino, es reflejada y detectada por el receptor.
Calculando cuánto tiempo tarda en detectarse la onda reflejada, podemos determinar la distancia del objeto al sensor.

t = Time travel of the sound wave in µs

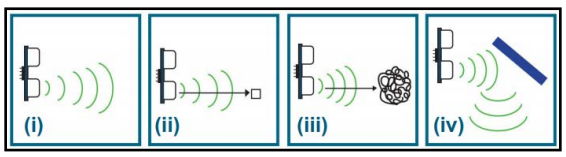
Speed of Sound = 343 m/s = 0.034cm/µs

$$\text{Distance} = \frac{t}{2} \times \text{Speed of Sound}$$

$$= \frac{t}{2} \times 0.034 \text{ cm}$$

$$= t \times 0.017 \text{ cm}$$

Los sensores ultrasónicos se utilizan ampliamente como sensores de retroceso de automóviles para ayudar a los conductores a estimar la distancia y detectar obstáculos. La ventaja de un sensor ultrasónico sobre un sensor infrarrojo es su insensibilidad a la iluminación circundante.
Sin embargo, es posible que los sensores ultrasónicos no funcionen como se esperaba si el objeto está (i) demasiado lejos, (ii) demasiado pequeño, (iii) demasiado blando o (iv) el objeto hace rebotar la onda de sonido lejos del receptor.



Función de arduino

1. Puedes utilizar esta función para leer el ciclo de tiempo de un pulso.
pulseIn(pin,value)
pin: el número de pin por el cual quieres que se haga la lectura del pulso.
value: tipo de pulso a leer; también HIGH o LOW.

Si el valor se establece en HIGH, pulseIn () espera que el pin pase de LOW a HIGH, comienza a cronometrar, luego espera a que el pin vuelva a LOW y detiene el cronometraje. Devuelve la longitud del pulso en microsegundos.

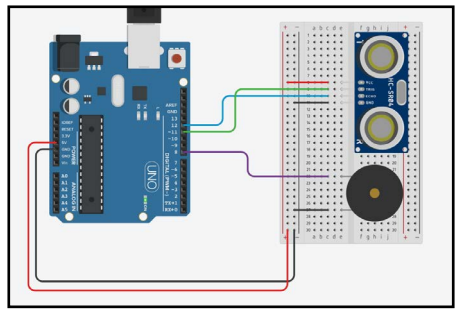
2. Tipos de dato
Te preguntarás por qué la mayoría de las veces usamos “int” (entero) para definir una variable, pero en este proyecto estamos usando “long” en su lugar. Esto se debe a que necesitamos informar a Arduino qué tipo de número queremos almacenar en la variable, ya sea un número pequeño, un número grande o un número con puntos decimales, etc. En este proyecto, el número que necesitamos almacenar para ‘duration’ es grande.

Variable	Number Range
char	-128 to 127
int	-32,768 to 32,767
long	-2,147,483,648 to 2,147,483,647
float	3.4028235E+38 to 3.4028235E+38

Proyecto 16: Construye un sensor de parachoques trasero de automóvil

1. Consigue los siguientes componentes.
Cables Jumper
Sensor ultrasónico
Buzzer pasivo

2. Construye el siguiente circuito.



3. Modifica el código del proyecto 15 con el siguiente y súbelo a tu placa.

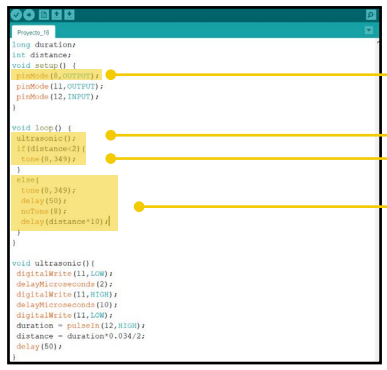
```
long duration;
int distance;
void setup() {
 pinMode(8,OUTPUT);
 pinMode(11,OUTPUT);
 pinMode(12,INPUT);
}

void loop() {
 ultrasonic();
 if(distance<2){
  tone(8,349);
 }
 else{
  tone(8,349);
  delay(50);
  noTone(8);
  delay(distance*10);
 }
}

void ultrasonic(){
 digitalWrite(11,LOW);
 delayMicroseconds(2);
 digitalWrite(11,HIGH);
 delayMicroseconds(10);
 digitalWrite(11,LOW);
 duration = pulseIn(12,HIGH);
 distance = duration*0.034/2;
 delay(50);
}
```

4. Verifica tu resultado.

Da click aquí para ver el video demostrativo

¿Suena el zumbador repetidamente cuando detecta un obstáculo? ¿El pitido se intesifica (es decir, la frecuencia del pitido aumenta) a medida que el sensor se acerca al objeto?

¿CÓMO FUNCIONA?



Configura el pin 8 (buzzer) como salida

Llama a la función “ultrasonic”

Si la distancia es inferior a 2 cm, el zumbador reproducirá la nota F4 (349) continuamente

De lo contrario, el zumbador emitirá una nota F4 (349) en un intervalo de ‘distancia x 10’ milisegundos. Por ejemplo: cuando la distancia = 5 cm, el zumbador suena una vez cada 50 ms. Cuando la distancia = 10 cm, el zumbador suena una vez cada 100 ms

DESAFÍO

¿Sabías que?

“El theremin es un instrumento musical electrónico controlado sin contacto físico por el thereminista (intérprete). Lleva el nombre de su inventor, Léon Theremin, quien patentó el dispositivo en 1928 “.
Fuente: : https://en.wikipedia.org/wiki/Theremin

Tarea: Construye un Theremin usando tu UNO R3 y el sensor ultrasónico. Divida el rango de detección del sensor ultrasónico en algunas zonas para reproducir diferentes tonos.

Da click aquí para ver el video demostrativo

¡Felicidades! Has completado la lección 7 y esto es lo que has aprendido:

- Cómo funciona un sensor ultrasónico.
- Cómo crear funciones autodeclaradas.