

Piensa
en la vida
como un
experimento

hackid

GUÍA DE EXPERIMENTOS

2

Kit
programable

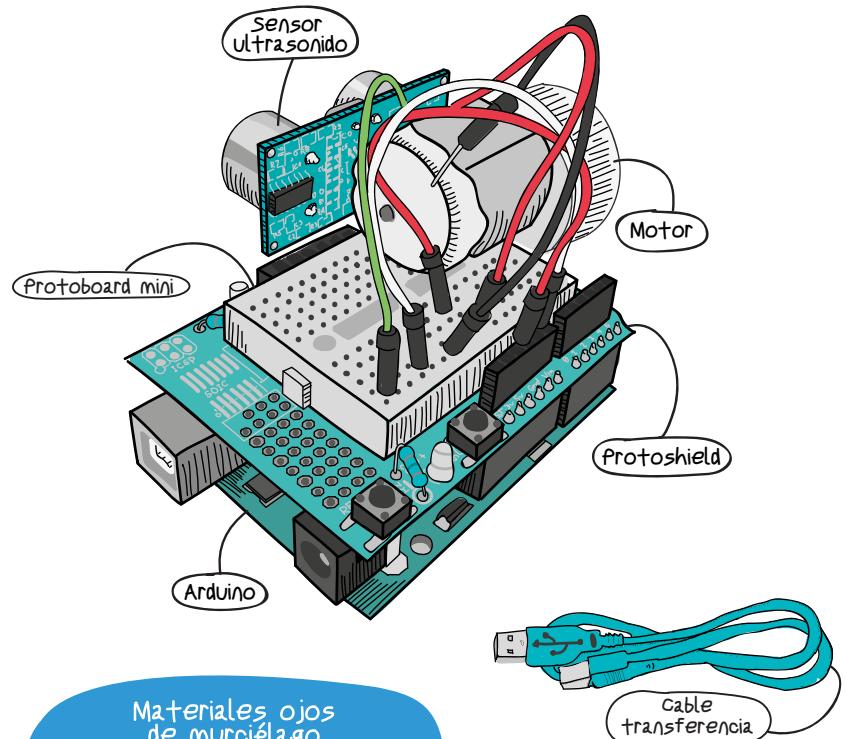
2 Ojos de murciélagos

¡Sabías que los murciélagos emiten ondas de sonido que, al rebotar sobre los objetos, producen ecos de retorno que les permiten ubicarse en el espacio? Así es como vuelan en la oscuridad.

El sensor de ultra sonido que vamos a usar para este proyecto es nuestro murciélagos; uno de los bafles emite una onda de sonido y el otro recibe el eco que, según el tiempo en el que se demora en retornar la onda, calcula la distancia a la que se encuentra el objeto. Si eres como un murciélagos y no te gustan

las sorpresas, si te asusta hasta tu sombra, si eres una de esas personas que siempre está alerta, este sensor de proximidad será tu mejor aliado. En este proyecto vamos a crear un sistema que te avisará si alguien se encuentra a menos de un metro ochenta de distancia. ¡Vamos!

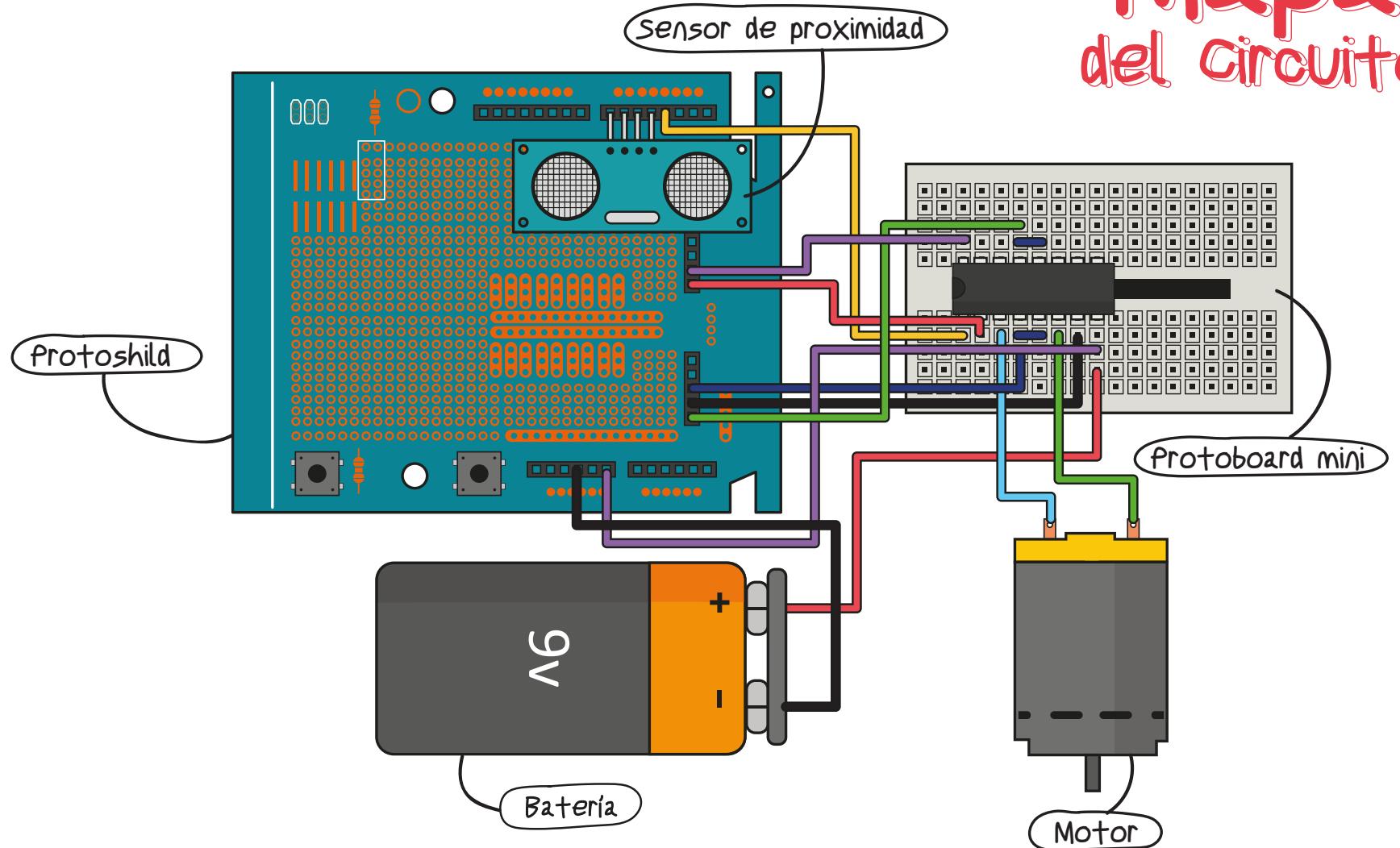
1 Lo primero que debemos hacer es alistar los materiales que vamos a usar para este proyecto. Recomendación: Intenta seguir el mapa del circuito para hacer las conexiones.



Materiales ojos de murciélagos	
Componentes	Cantidad
Arduino	1
Shield	1
Sensor de ultra sonido	1
Cable de transferencia	1
Motor	1
chip L293D	1
Pila 9v	1
Cables	1

2 Copia el código en el programa. (Ve a la hoja de códigos al final del proyecto y transcribe el código que corresponde a Ojos de murciélagos). Ten cuidado con los espacios, los signos y las mayúsculas, ya que cualquier error que se pase afecta la programación y las órdenes que recibe el Arduino.

Mapa del Circuito

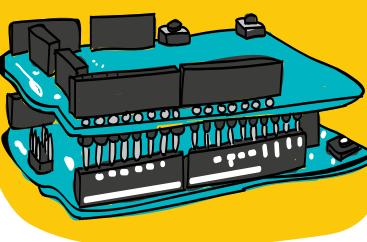


3

Comprueba que el código haya quedado bien copiado y pásalo al módulo.

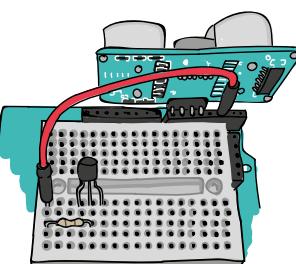
4

Ahora hagamos las conexiones físicas. Conecta el Shield con el Arduino, solo debes encajar el Shield encima del módulo.



5

Conecta el sensor de ultrasonido HC-SR04 directamente al protoshield, con los parlantes apuntando afuera y entre los pines 4-7, asegurándote que se respete el siguiente orden: pin 4 - Vcc, pin 5 - Trig, pin 6 - Echo, pin 7 Gnd.

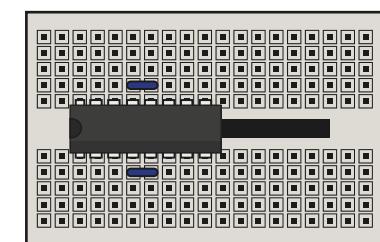


6

Pon una protoboard pequeña sobre el protoshield y conecta CI L293D (circuito integrado) en la mitad de La protoboard como lo muestra la imagen.

10

Conectemos la pila de 9V de la siguiente manera, negativo (negro, Gnd) a cualquier Gnd del Arduino, y rojo (línea) a la pata 8 del L293D.



7

Conecta los jumper del motor a las patas 3 y 6 del L293D. Recuerda que antes debes haber ajustado los cables a los terminales del motor. Una buena forma de lograr que no se desajusten es doblar un poco la puntas del cables.

8

A los pines Gnd del protoshield vamos a conectar las patas 5,7 y 12 del L293D.

9

Ahora vamos a conectar las patas 4 y 5 y 12 y 13 del L293D entre sí usando un jumper. (Apóyate en la imagen para realizar este paso).

códigos

NOTA: El código está escrito en amarillo. Las palabras que están en blanco sirven para explicar cada línea del código, así que no son necesarias que las copies en el programa.

* Este código está escrito para controlar la velocidad de un motor de vibración por medio de un puente HL293D. La velocidad del motor aumenta en la medida que se acerque un obstáculo para lo cual se cuenta con un sensor de ultrasonido SR04. El código está diseñado para que conectes el sensor de ultrasonido directamente a la Arduino.

```
//Empezamos por definir que pines vamos a usar y para que.
```

```
const int motorPin = 3; // este es el pin por el cual vamos a controlar el motor.
```

```
const int EchoPin = 6; //Estos son los pines de donde se va a conectar el sensor.
```

```
const int TriggerPin = 5;
```

```
const int Vcc = 4;
```

```
const int Gnd = 7;
```

```
long Tiempo, Distancia, DistanciaC; // estas son las variables que vamos a usar.
```

```
void setup() { // Esta es la fracción de código que se ejecutará una sola vez.
```

```
Serial.begin (9600); // gracias a esta línea, podremos acceder a las lecturas del sensor.
```

```
pinMode(motorPin,OUTPUT); // aquí definimos los canales de entrada y salida.
```

```
pinMode(Gnd,OUTPUT);
```

```
pinMode(Vcc,OUTPUT);
```

```
pinMode(TriggerPin,OUTPUT);
```

```
pinMode(EchoPin,INPUT); // este es el pin que necesitaremos para interpretar la distancia.
```

```
digitalWrite(Gnd,LOW); // aquí activaremos definitivamente la tierra y la fuente del sensor.
```

```
digitalWrite(Vcc,HIGH);
```

```
}
```

```
void loop() { // Esta es la fracción de código que se repite continuamente.
```

```
digitalWrite(TriggerPin, LOW); // Estas líneas describen una secuencia de pulsos
```

```
delayMicroseconds(2); // que emite el sensor en ondas de sonido, para luego recibirlas.
```

```
digitalWrite(TriggerPin, HIGH);
```

```
delayMicroseconds(10);
```

```
digitalWrite(TriggerPin, LOW);
```

Tiempo = pulseIn (EchoPin, HIGH); // El sensor recibe las ondas y mide cuánto demoraron en llegar.
Distancia = (Tiempo/2) / 29; // con esta ecuación se deduce la distancia a la que se encuentra el obstáculo.

DistanciaC = Distancia/2; // definimos una variable que divide la distancia en 2,

// Esta se creó para simplificar la relación que se quiere establecer entre la distancia y la velocidad. El valor máximo que se le puede dar a la velocidad es de 255, y la distancia máxima a la que puede trabajar el sensor es de 500 cm. // dividir la anterior en 2 nos da valores Homogéneos.

```
Serial.print(DistanciaC); // Instrucción que imprime la información solicitada
Serial.println("DistanciaC"); // por medio de la comunicación serial.
```

```
int speed= (-1*DistanciaC) + 250; /* Dado que la relación que queremos entre distancia y velocidad corresponde a una función lineal - inversamente proporcional, usamos una ecuación de la forma: f(x) = -1x + 250. Donde -1 es la pendiente y 250 la intersección. */
```

```
Serial.print(speed);
Serial.println("vel");
```

```
if (speed >= 250){ // Los siguientes if definen los rangos de velocidad y distancia.
analogWrite(motorPin,255);
}
```

```
if (speed <= 30){
analogWrite(motorPin,0);
}
```

```
else {
analogWrite(motorPin, speed); // pedimos que controle el motor según la velocidad.
}
delay(300); // da un espacio de 300 milisegundos entre lectura y lectura.
```

```
} // juega con los valores descritos en el código y decide como quieras que el motor reaccione.
```

