CAPÍTULO 7:

ULTRASONIDO + RGB

Ultrasonido + RGB

Cuando definimos que es un robot hablamos de sistemas de Entrada - Procesamiento - Salida. En este capítulo usaremos el sensor de ultrasonido como un sistema de entrada, emitiendo y recibiendo ondas de sonido; estas van a ser procesadas por el Arduino y traduciendo el tiempo de emisión/receptor en una distancia a través de un cálculo matemático; para que finalmente, el led RGB sea el actuador que encienda diferentes colores en relación a la distancia registrada.

Consejo:

Podés realizar una consulta por monitor serie para ver bien los valores del sensor y realizar una programación más precisa.

Estructuras condicionales

También hay estructuras condicionales con las que podemos trabajar dentro del código para darle instrucciones a nuestro robot y controlar así el flujo de información que recibe.

A continuación veremos que hay dos maneras de declarar una estructura condicional:

La estructura "if"

La primera estructura condicional que veremos es **"if"**, que significa "Si..." en inglés. Dentro de un programa se usa para corroborar si se ha alcanzado una determinada condición. Si ésta se cumple (**true**) el programa ejecutará las instrucciones correspondientes que se encuentran entre llaves. Si la condición no se cumple (**false**) el programa ignorará las instrucciones que le siguen y saltará al siguiente paso.

Su sintaxis correcta es:

```
if (CONDICIÓN) {
        //Instrucciones que quiero ejecutar si la condición es "true"
        }
```

La estructura "if ... else"

La segunda opción es **"if... else"**, que significa "Si... Sino" en inglés.

Esta alternativa de la función if funciona como una bifurcación ya que el robot tendrá la instrucción de que si se cumple la condición haga una cosa y, si no la cumple, haga otra. Esto la diferencia del **if** a secas, ya que en lugar de continuar con el programa en caso de no cumplir la condición (**false**) el **else** le indica entre llaves qué hacer a continuación antes de seguir con el resto del código.

Su sintaxis correcta es:

```
if (CONDICIÓN) {
        //Instrucciones que quiero ejecutar si la condición es "true"
}
else {
//Instrucciones que quiero ejecutar si la condición es "false"
}
```

Otras funciones útiles para tu robot

A continuación encontrarás otras funciones que serán de gran ayuda al momento de darle instrucciones al robot.

constrain (restringir)

Fuerza a un número a permanecer dentro de ciertos límites. Es decir, si el valor es mayor o menor a alguno de los límites especificado por parámetro (a y b), entonces se igualará el valor al límite que haya sobrepasado. Si por el contrario, el número está dentro de los límites, la función devolverá el mismo valor.

Su sintaxis correcta es:

constrain(x, a, b);

Donde sus parémtros son:
x: el número a limitar **a:** el límite inferior **b**: el límite superior

¿Cómo trabaja entonces la función constrain?
Si el valor de **x** se encuentra entre el rango determinado por **a** y **b**, **x** conserva su valor. Sin embargo, si el valor de **x** es menor a **a** (**x < a**) o mayor a **b** (**x > b**), **x** adoptará el valor del límite más cercano; es decir que, si el valor de **x** es menor a **a**, adoptará el valor de **a**, y si su valor es mayor a **b**, pasará a tener el valor de **b**.

Dicho de otro modo:

x > a && x < b x conserva su valor.
x < a x adopta el valor de a, ya que es menor al mínimo especificado.
x > b x adopta el valor de b, ya que es mayor al máximo especificado.

Ejemplo:
tVal = constrain(tiempoPulso, 1140, 5700);

si **tiempoPulso** es **> 1140 y < 5700**, entonces **tVal = tiempoPulso;**
si **tiempoPulso < 1140** , entonces, **tVal = 1140;**
si **tiempoPulso > 5700**, entonces **tVal = 5700;**

map (mapeo de valores)

Esta función asigna a un número que está dentro del **rango "a"** (definido por los parámetros **a1** y **a2**) un valor proporcional o equivalente dentro de un **rango "b"** (definido por **b1** y **b2**).

Su sintaxis es: **map (valor, a1,a2,b1,b2);**

Donde sus parámetros son:
valor: el número a mapear
a1: límite inferior del rango actual del valor **a2**: límite superior del rango actual del valor
b1: límite inferior del rango deseado **b2**: límite superior del rango deseado

Es decir que si el valor inicial coincide con el límite inferior del rango **"a" (a1)** , entonces la función devolverá el mismo límite del rango **"b" (b1)**; si coincide con el límite superior del rango **"a" (a2)**, devolverá también el límite superior del rango **"b" (b2)**; y si el valor es algún número cualquiera entre **a1** y **a2**, devolverá un valor equivalente que esté entre **b1** y **b2**. Esto es lo que se denomina mapear un número.

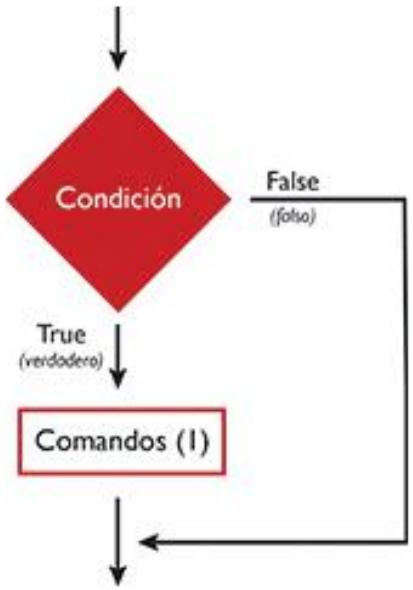
¡IMPORTANTE! Cabe aclarar que la función no restringe los valores si estos no están dentro de los rangos, ya que estos valores a veces pueden ser útiles. La función constrain () se puede usar antes o después de esta función, si se desean limitar los valores para estar dentro de los intervalos.

Ejemplo:
intensidad = map(tVal, 1140, 5700, 0, 255);

si **tval = 1140**, entonces **intensidad = 0** // límites inferiores de ambos rangos
si **tval = 5700**, entonces **intensidad = 255** // límites superiores de ambos rangos
si tval = **3420**, entonces **intensidad = 127** // valores medios de ambos rangos

Tipos de bifurcación

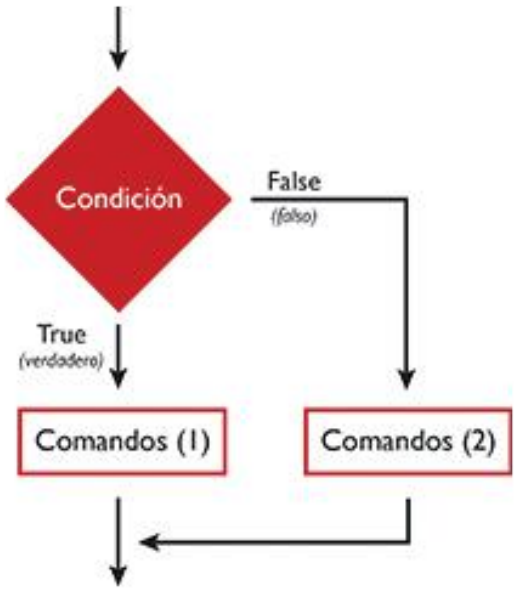
Hacer una bifurcación condicional significa que si se cumple una determinada condición vamos a hacer una cosa, si no se cumple haremos otra cosa. Existen de 3 tipos:



Bifurcación Simple: Ejecutamos un comando si la condición es verdadera y en caso contrario no realizamos nada.

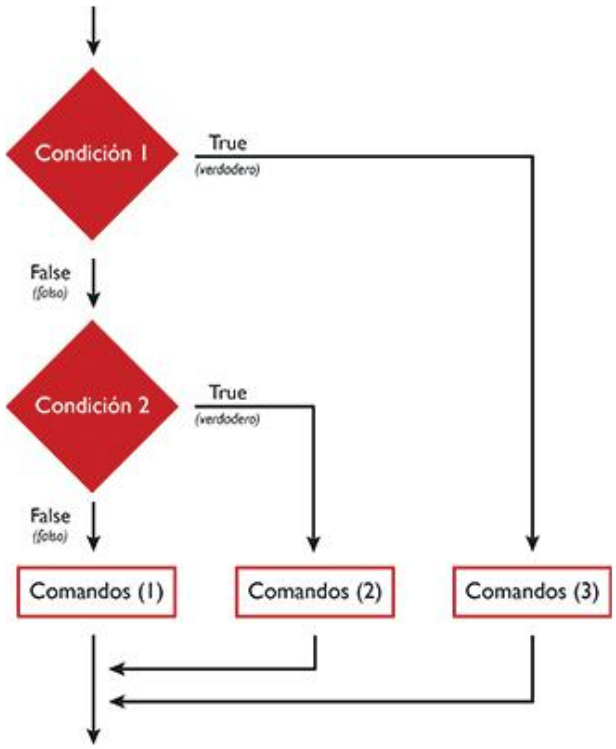
Su programacion es:

```
if(condición){
Mis comandos....
}
```



Bifurcación Completa: Si la condición es verdadera realizamos una serie de comandos y si es falsa otra serie de comandos diferentes.

```
if(condición){
accion / comandos 1
}
else{
accion / comandos 1
}
```

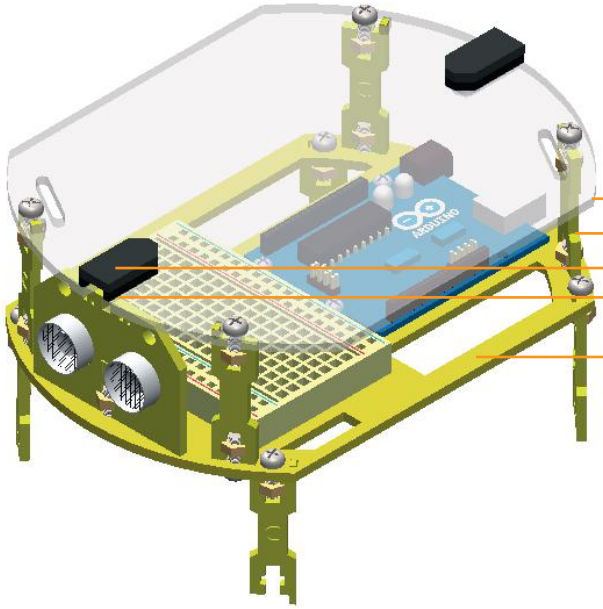


Bifurcación Múltiple: En este caso hacemos lo siguiente, si la condición 1 es verdadera realizamos una determinada acción, en el caso de no cumplirse pero si se cumple una condición 2 realizamos una segunda acción, caso contrario, es decir ninguna de las condiciones se cumple realizamos otra acción.

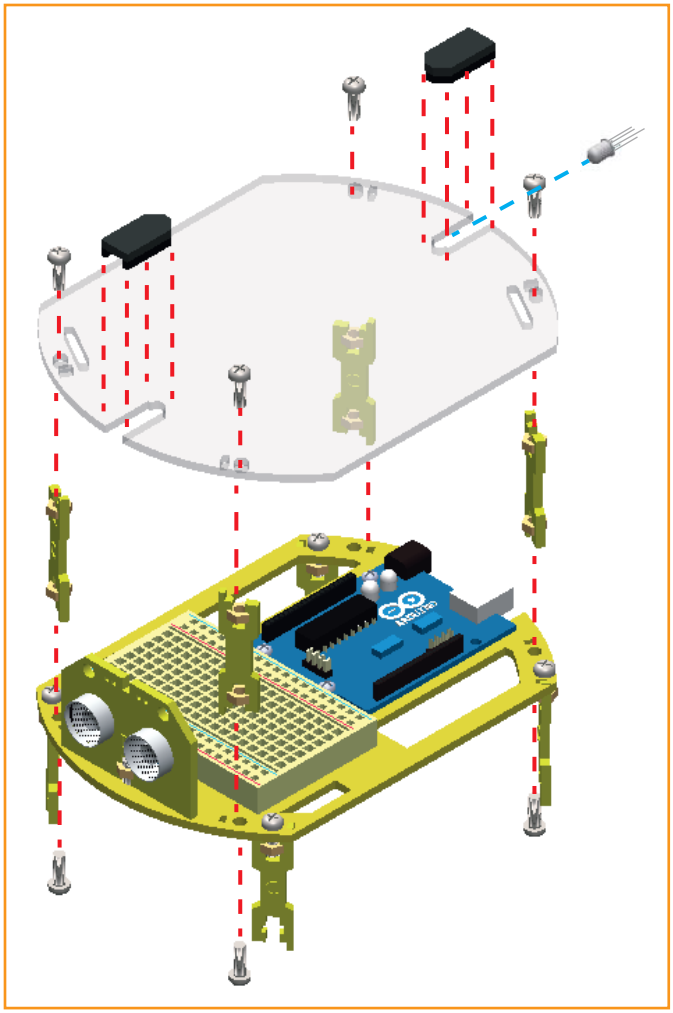
```
if(condicion1){
acción / comandos 1
}
else if(condición 2){
acción / comandos 2
}
else{
acción / comandos 3
}
```

Ensamble

A continuación veremos la estructura y componentes. Combinaremos el sensor ultrasónico con el led RGB.



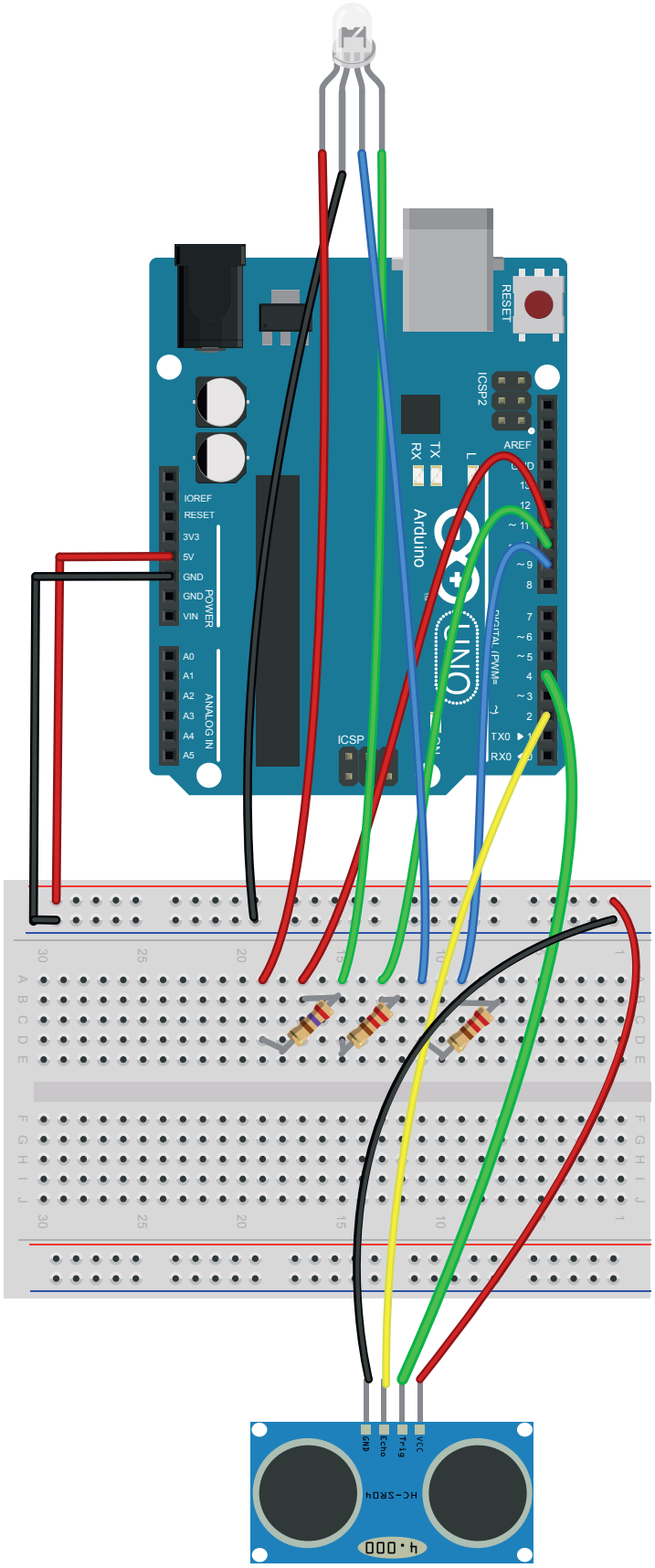
Materiales:
-Estructura de Ultrasonido
-Piezas C x 4 unidades
-Piezas K
-Led RGB
-Chasis RGB



Pasos:
1- Pegar piezas K en chasis RGB
2- Colocar 4 columnas C en estructura de ultrasonico.
3- Unir el chasis RGB a las 4 columnas C.
4- Ubicar el led RGB en la ranura del chasis RGB, por debajo de la pieza K.

Circuito electrónico

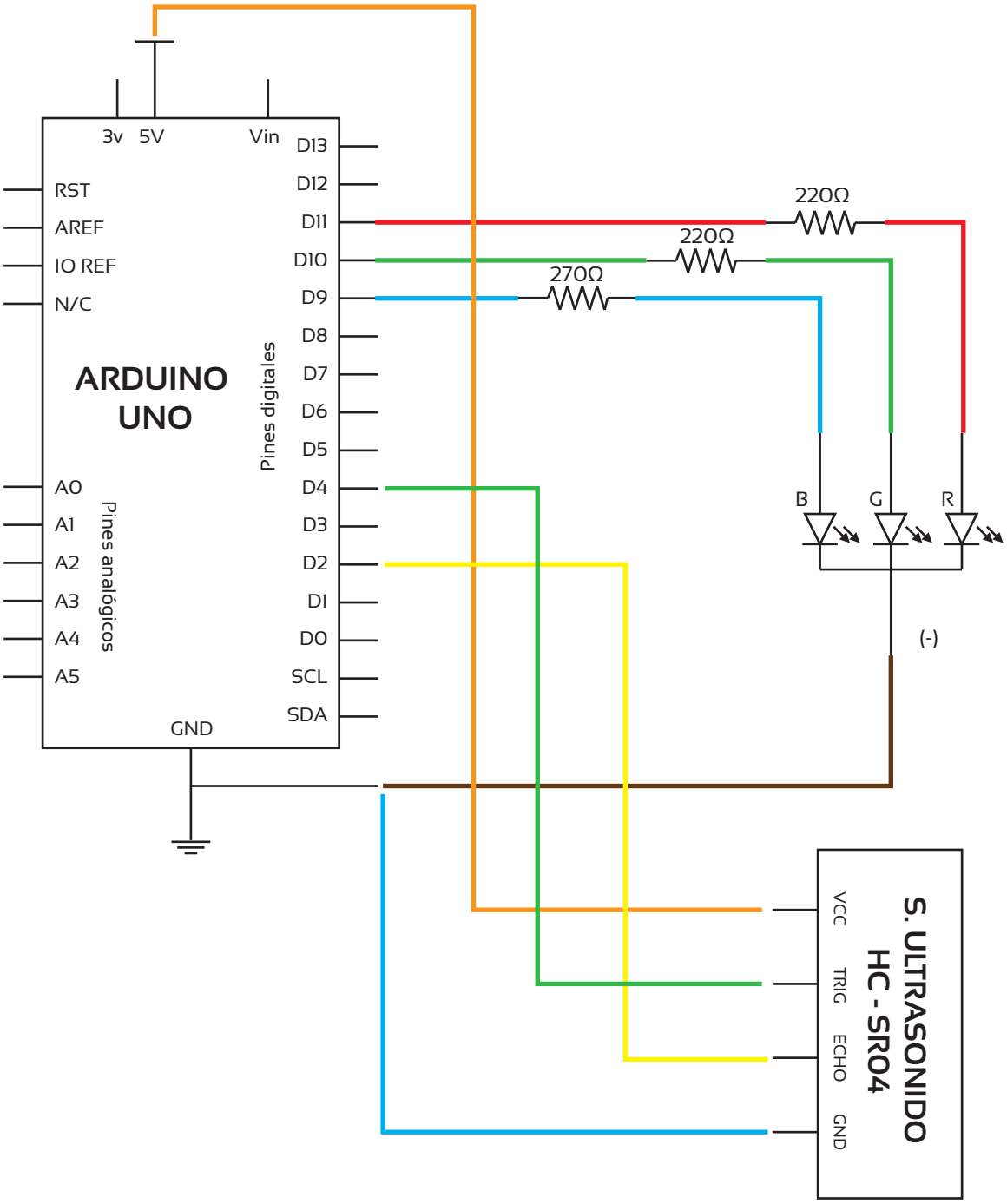
Finalmente observamos las conexiones de nuestro circuito electrónico.



Componentes electrónicos:
-1 led RGB
-2 resistencias de 220Ω
-1 resistencia de 270Ω
-Sensor ultrsónico
-5 cables M-M
-8 cables M-H

NOTAS:
Recuerden que su led puede ser de ánodo común y en ese caso debe ir conectado al positivo, y por otro lado, las conexiones de trigger y echo están invertidas con respecto al capítulo anterior, siempre observar bien los pines del gráfico.

Circuito electrónico en símbolos



Código de programación

Desarrollaremos el código de programación con el cual vamos a configurar las diferentes salidas de nuestro arduino para lograr diferentes colores, utilizando la tecnica de adición, en relación a los parámetros que mida nuestro sensor ultrasónico.

```
1  int disparo = 4;                          //Arrancamos definiendo las variables:
2  int echo = 2;
3  int ledTest = 13;
4  int ledRojo = 11;
5  int ledVerde = 10;
6  int ledAzul = 9;
7  int intensidad = 0;
8  long tiempoPulso;
9  long tVal;
10
11 void setup() {
12
13   pinMode(ledTest, OUTPUT);
14   pinMode(disparo, OUTPUT);               //Configuramos las entradas y salidas
15   pinMode(echo, INPUT);
16
17   digitalWrite(disparo, LOW);
18   analogWrite(ledRojo, 0);
19   analogWrite(ledVerde, 0);
20   analogWrite(ledAzul, 0);
21
22   digitalWrite(ledTest, HIGH);            //Saludo inicial
23   delay(500);
24   digitalWrite(ledTest, LOW);
25   delay(500);
26   digitalWrite(ledTest, HIGH);
27   delay(500);
28   digitalWrite(ledTest, LOW);
29   delay(500);
30   digitalWrite(ledTest, HIGH);
31   delay(500);
32   digitalWrite(ledTest, LOW);
33   delay(500);
34 }
35
36
37 void loop() {
38
39   digitalWrite(disparo, HIGH);            //INICIO: Medimos la longitud del puslo entrante la
40   delayMicroseconds(10);                  utilizamos para medir el tiempo que ha transcurrido
41   digitalWrite(disparo, LOW);             entre el envío  del pulso ultrasónico y la recepción
42                                           del rebote,es decir: desde que el pin "echo" empieza
43   tiempoPulso = pulseIn(echo, HIGH);      a recibir el rebote, HIGH, hasta que deja de hacerlo,
44                                           LOW, la longitud del pulso entrante. El resultado lo
45                                           da en microsegundos.
46
47
48
49
50
51
52
53
54
```

```
54                                                    //Ahora dependiendo del tiempo del
55                                                    pulso vamos a utilizar los leds RGB
56                                                    para señalar si el objeto se encuentra
57                                                    cerca o lejos, usando diferentes condi-
58                                                    cionantes.
59
60  if (tiempoPulso < 550) {                          //Cuando el objeto se encuentre a
61    analogWrite(ledRojo, 255);                      menos de 10cm mostraremos una luz
62    analogWrite(ledVerde, 0);                       roja.
63    analogWrite(ledAzul, 0);
64   }
65
66
67  else if ( tiempoPulso >= 550 &&  tiempoPulso < 860 ) {   //Cuando esté entre 20cm y 15cm una
68    analogWrite(ledRojo, 255);                              luz amarilla.
69    analogWrite(ledVerde, 255);
70    analogWrite(ledAzul, 0);
71   }
72   else if (tiempoPulso >= 860 && tiempoPulso < 1140) {     //Cuando está a entre 15cm y 20cm la
73    analogWrite(ledRojo, 0);                                luz verde (1140µs segun las pruebas en
74    analogWrite(ledVerde, 255);                             el capitulo anterior)
75    analogWrite(ledAzul, 0);
76   }
77   else {                                                   //Cuando esté mas lejos luz blanca
78    tVal = constrain(tiempoPulso, 1140, 5700);              proporcional a la distancia
79    intensidad = map(tVal, 1140, 5700, 0, 255);
80
81    analogWrite(ledRojo, intensidad);
82    analogWrite(ledVerde, intensidad);
83    analogWrite(ledAzul, intensidad);
84   }
85  }
```

ACTIVIDADES Y EJERCITACIÓN

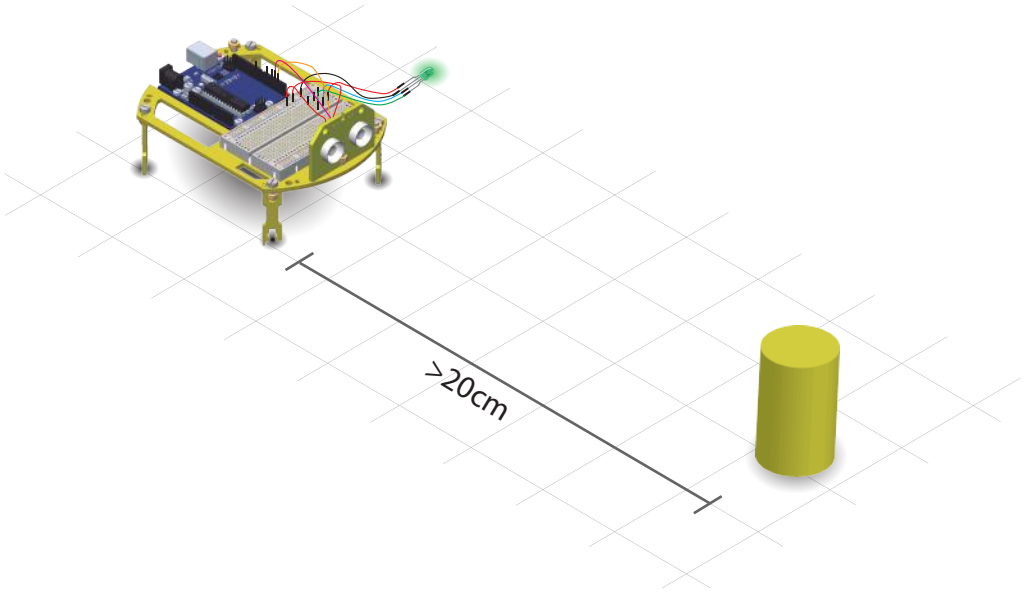
Cap. 7: Actividades complementarias

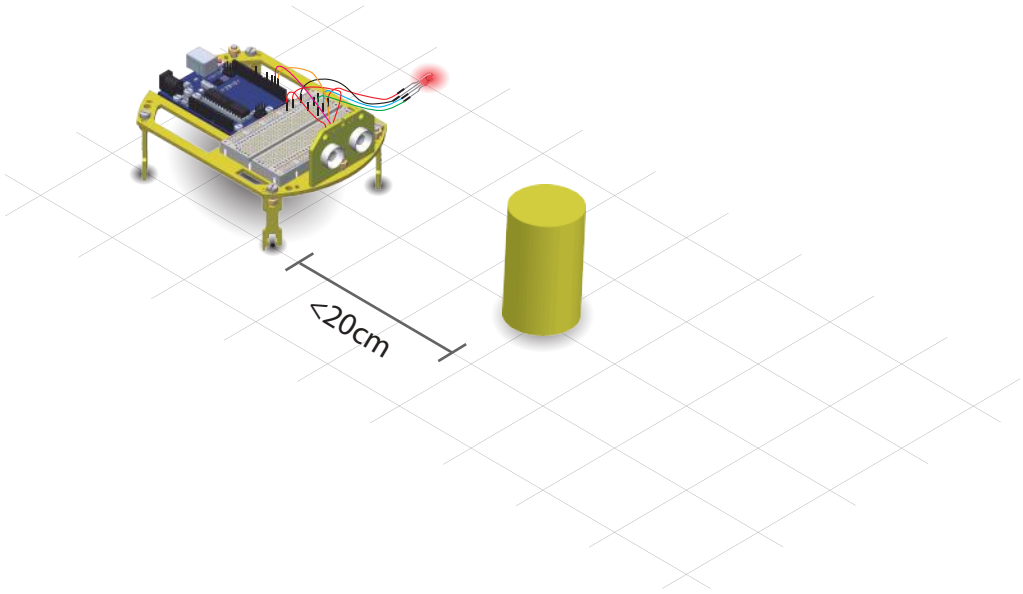
En base a los contenidos vistos en el **"Capítulo 7: Calibración de uso"** realiza las siguientes actividades complementarias.

Actividad 1

Desafío práctico: alarma LED

Crea una "alarma LED" que haga parpadear el led RGB en color rojo cuando detecte algo a menos de 20 cm. y que esté en verde mientras no haya nada dentro de los 20 cm.





* Las ilustraciones son esquemáticas.

Cap. 7: Respuestas

Actividad 1

Desafío práctico: alarma LED

```
int ledRojo = 11;
int ledVerde = 10;
int ledAzul = 9;
int disparo = 4;
int echo =2;
int rojo = 0;
int verde = 0;
int azul = 0;

long tiempoPulso;
float dstMedida;

void setup() {
  pinMode(ledRojo, OUTPUT);
  pinMode(ledVerde, OUTPUT);
  pinMode(ledAzul, OUTPUT);
  pinMode(disparo, OUTPUT);
  pinMode(echo, INPUT);

  analogWrite(ledRojo, 255);
  analogWrite(ledVerde, 255);
  analogWrite(ledAzul, 255);
}

void loop() {
  digitalWrite(disparo, LOW);
  delayMicroseconds(5);
  digitalWrite(disparo, HIGH);
  delayMicroseconds(10);
  digitalWrite(disparo, LOW);

  tiempoPulso = pulseIn(echo, HIGH);
  dstMedida = 0.0340 / 2 * tiempoPulso;
  Serial.println(dstMedida);

  if (dstMedida < 20){
     analogWrite(ledVerde, 255);
     analogWrite(ledAzul, 255);
     analogWrite(ledRojo, 0);
     delay(50);
     analogWrite(ledRojo, 255);
     delay(50);}
  else {
     analogWrite(ledRojo, 255);
     analogWrite(ledVerde, 0);
     analogWrite(ledAzul,255);
     }
  }
```