

1.2.15. Sensor Digital: Ultrasonidos

En este apartado vamos a realizar varias actividades en donde utilizaremos los sensores de ultrasonido. Como veremos más adelante, un sensor de ultrasonido, al igual que hacen los murciélagos y otros animales, mide el tiempo que tarda una señal emitida en ir y rebotar contra un obstáculo. Este tiempo le da una medida de lo lejano que está dicho obstáculo.

Decimos que el sensor de ultrasonidos es un sensor digital porque nos comunicamos con él a través de una señal digital, como puede ser el protocolo TTL. Esto difiere de los sensores analógicos que enviarían un dato a través de una señal analógica (comúnmente un voltaje comprendido entre 0 y 5v) .

1.2.15.1. Medidor de distancias

En esta actividad utilizaremos un sensor de ultrasonidos como medidor de distancias

Componentes

- Sensor ultrasonidos contenido en el kit de robótica de BQ (o cualquier otro sensor digital compatible con el chip HC-SR04)
- Placa ZUMBT o Arduino compatible
- Cable USB para conectar la placa al ordenador

Conexionado

El sensor de ultrasonidos tiene cuatro pines marcados como: GND (masa), ECH (Echo), TRI (Trigger) y VCC (+5v). Conectaremos el TRI a un PIN de Arduino para ordenar al sensor que emita la onda y el ECH (Echo), que lo conectaremos a otro PIN, nos indicará cuando la recibe (por eso se llama eco). Por ejemplo, podemos realizar el siguiente esquema de conexión (ver Figura 1.2.15-1):

- GND sensor – Cualquier pin negro (masa) de Arduino
- VCC sensor – Cualquier pin rojo (+5v) de Arduino
- ECH sensor – pin 4 Arduino
- TRI sensor – pin 5 Arduino
- USB conectando nuestra placa compatible Arduino con el PC

La muestra una imagen de como conectar el sensor de ultrasonido con la ZUMBT en el robot Printbot de BQ. Este conexionado es equivalente para cualquier otro sensor con el chip HC-SR04.

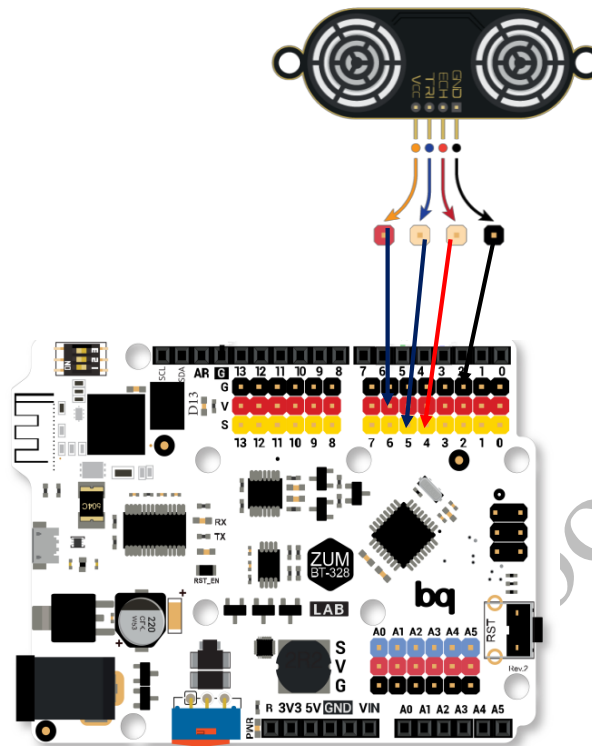


Figura 1.2.15-1 Conexión de un sensor de ultrasonidos a una placa Arduino UNO compatible

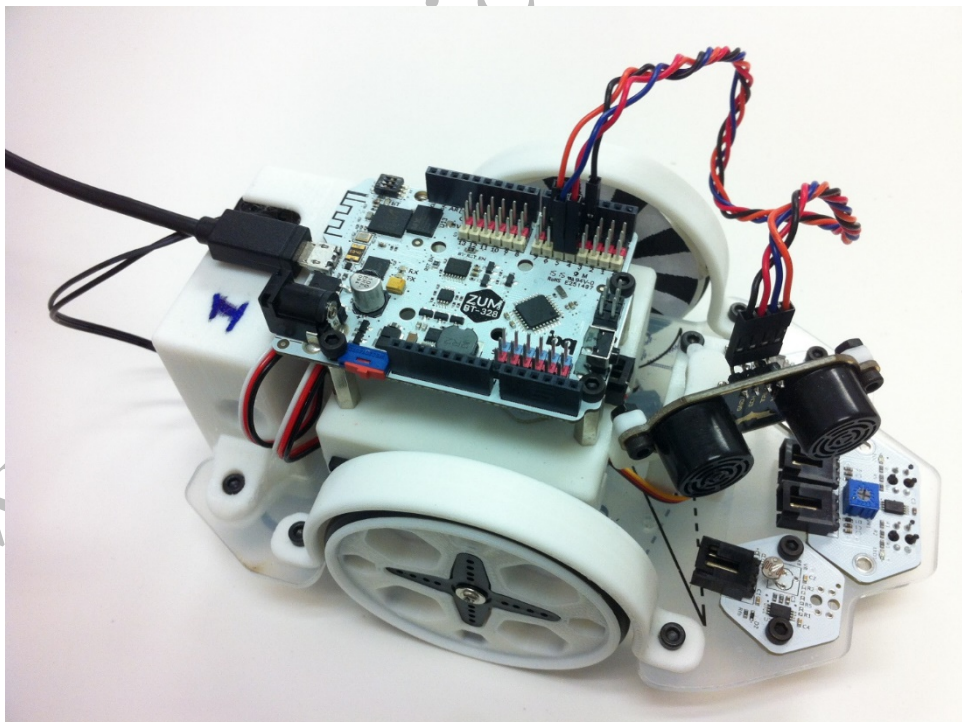


Figura 1.2.15-2 Conexión del ultrasonido en el Printbot

En bitbloq nos crearemos un proyecto en el que añadiremos la placa, el componente ultrasonido y el USB tal y como aparece en la Figura 1.2.15-3.

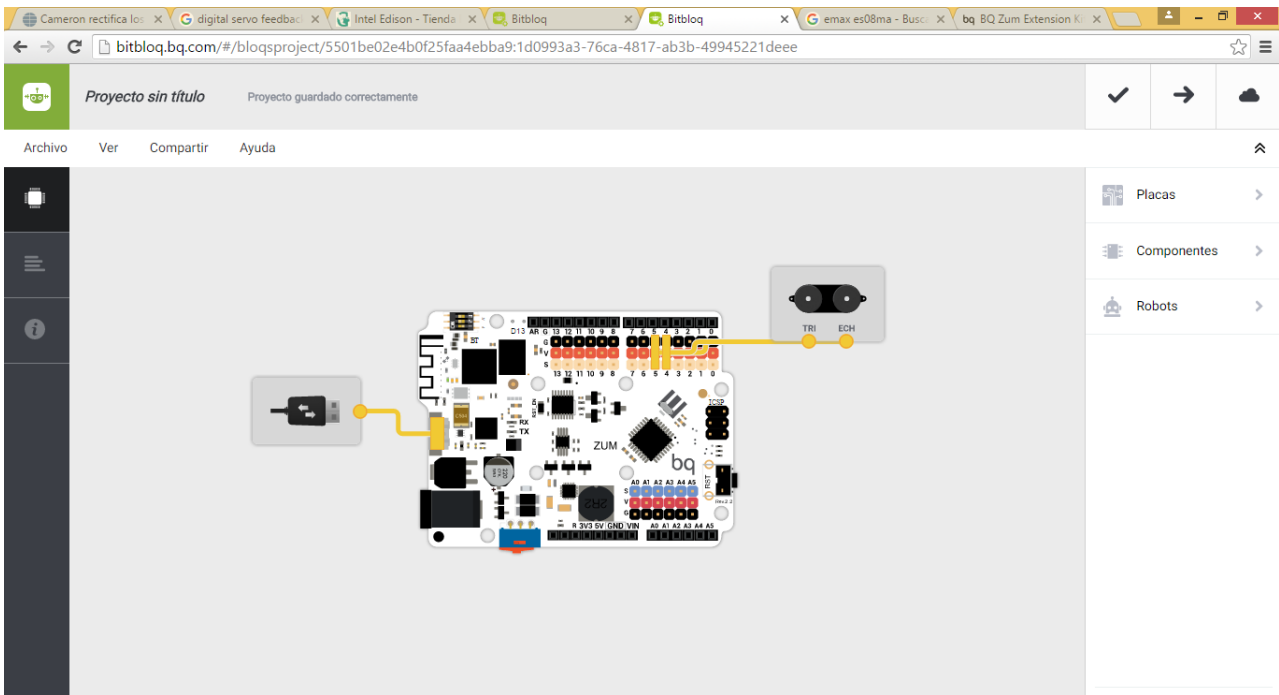




Figura 1.2.15-3 Conexión en bitbloq de los componentes para el medidor de distancias.

Programación

Una vez realizada la conexión nos vamos a la pestaña Software (icono ). Pulsamos en componentes (icono ) apareciendo los bloques de la Figura 1.2.15-4. La programación con Bitbloq de esta actividad es muy sencilla. EL bloque „leer sensor“ nos proporciona el valor de la distancia medida por el sensor. Enviaremos dicha información a través del puerto serie utilizando el bloque enviar.

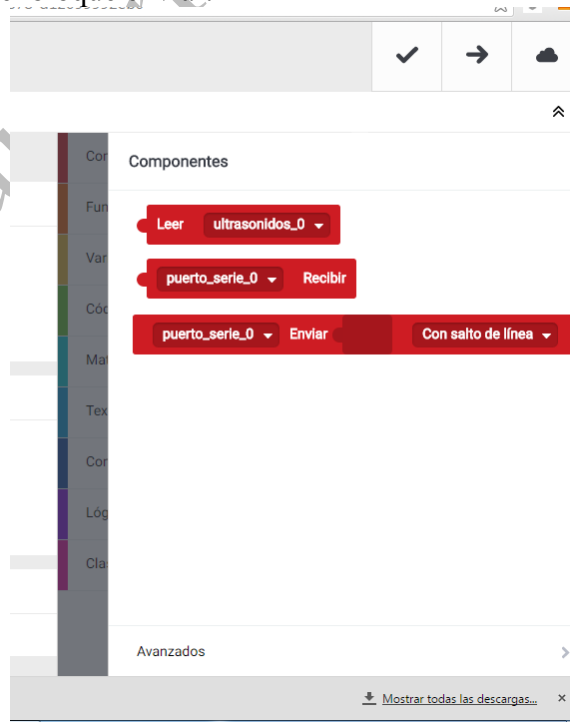


Figura 1.2.15-4 Bloques de los componentes de la actividad

Libro de Actividades de Robótica Educativa

El código de bloques resultante aparece en la Figura 1.2.15-5. Hemos añadido una espera (del bloque Control) para que el envío de información no sea demasiado rápido.

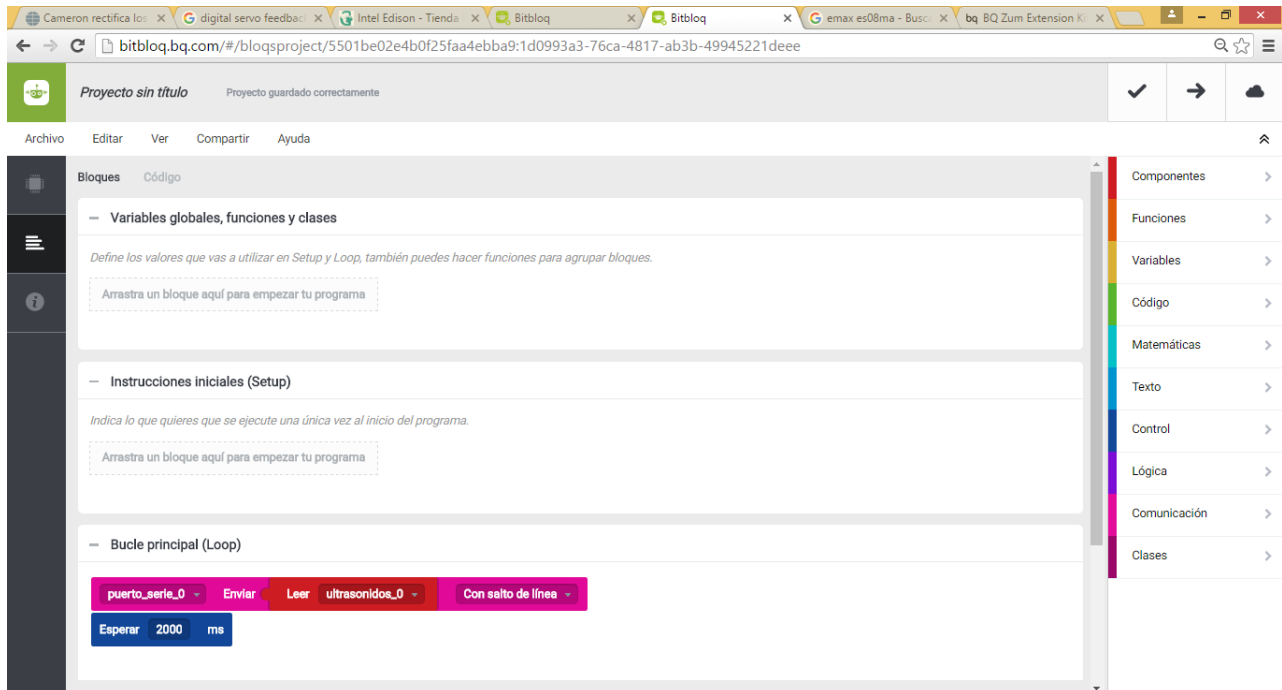


Figura 1.2.15-5 Programación por bloques de la actividad de medidor de distancias.

La traducción de estos bloques al lenguaje de programación de Arduino es el siguiente (Figura 1.2.15-6):

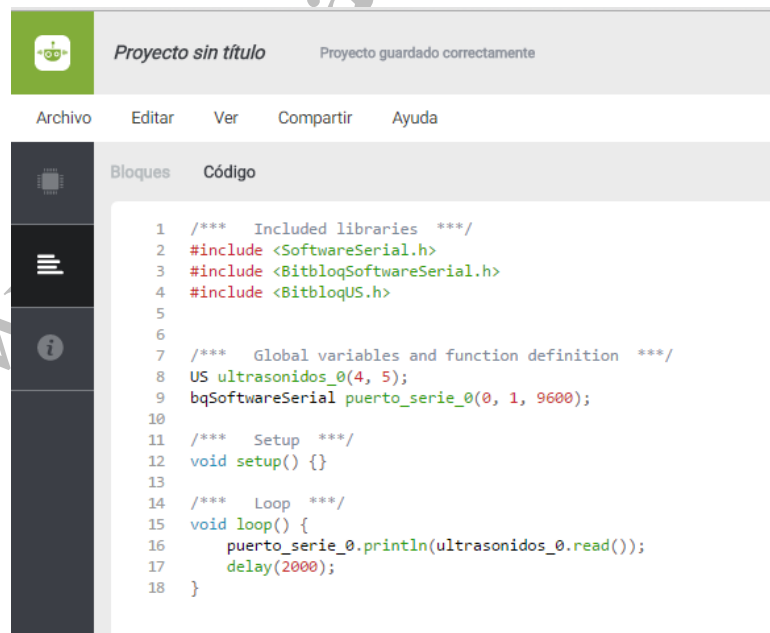



Figura 1.2.15-6 Código arduino de la actividad

Lo más importante de dicho código es que hay una librería (bitbloqUS) que se encargará del ultrasonido. En la línea 8 inicializamos dicha librería indicando los pines a los que tenemos conectado el sensor. Tenemos otra librería (BitbloqSoftwareSerial) que, como en actividades

Libro de Actividades de Robótica Educativa

anteriores, nos permite comunicarnos con nuestro PC. En la línea 16 vemos como la distancia medida del ultrasonido es enviada por el puerto serie.

Ahora es el momento de comprobar el funcionamiento de nuestro programa. Primero cargaremos el programa en nuestra placa con el icono . Una vez cargado, abriremos un monitor serie (en el menu editar-mostra monitor serie)

Se nos abrirá una ventana, como la de la Figura 1.2.15-7 donde se irán mostrando valores que corresponden a las distancias medidas del sensor (probar a acercar y alejar un objeto al sensor de ultrasonidos para comprobar que estos valores cambian)

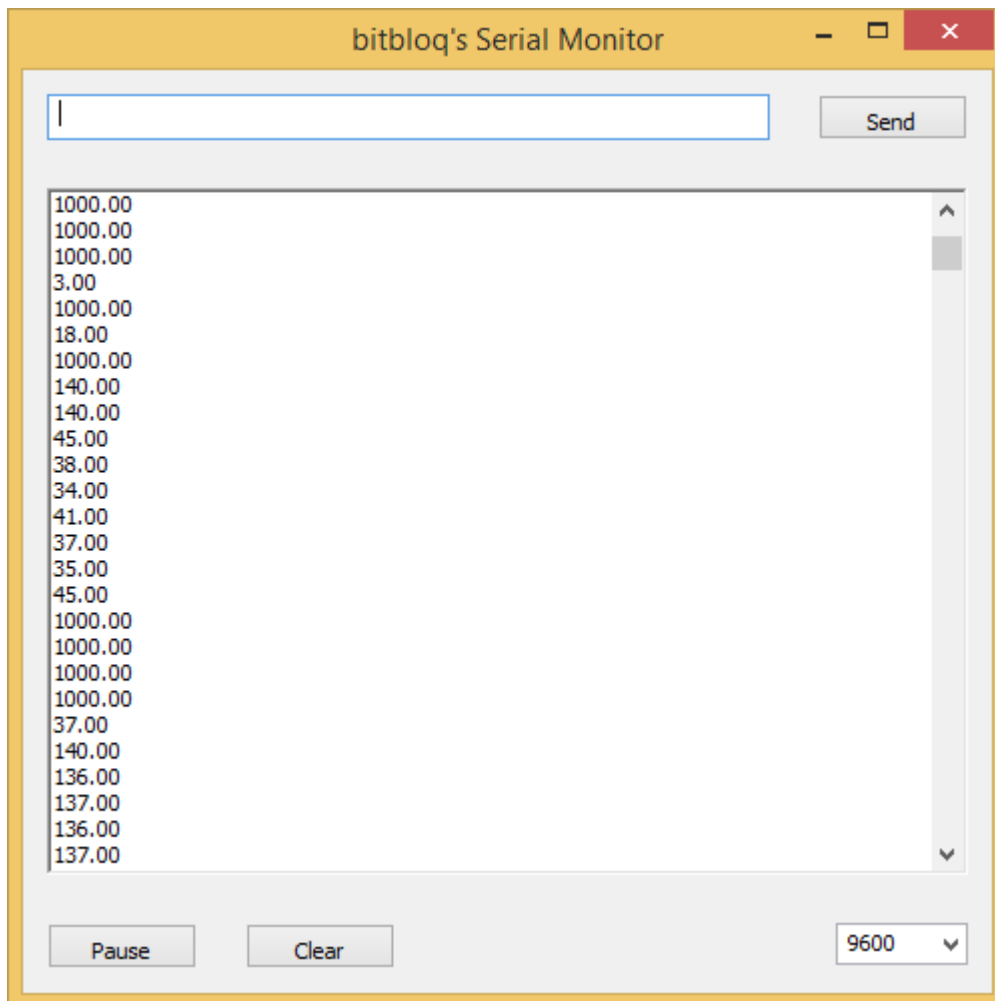


Figura 1.2.15-7 Valores de distancia mostrados en el monitor serie

Un poquito de teoría:

Los sensores de ultrasonidos basados en el chip HC-SR04 tienen dos transductores: un altavoz (que emite el ultrasonido) y un micrófono (que recibe el eco). Su principio de funcionamiento, resumido en la Figura 1.2.15-8, es el siguiente:

Libro de Actividades de Robótica Educativa

1. El envío de un Pulso "1" de al menos de 10uS por el Pin Trigger (Disparador) le dice al sensor que debe empezar a emitir ultrasonido.
2. El sensor enviará 8 Pulsos de 40KHz (Ultrasonido), colocando su salida Echo a alto. Este evento es detectado por un programa en la placa de Arduino (en nuestro caso la librería BitbloqUS) para iniciar el conteo del tiempo.
3. La salida Echo se mantiene en alto hasta recibir el eco reflejado por el obstáculo a lo cual el sensor pondrá su pin Echo a bajo. En este momento se termina de contar el tiempo.
4. La distancia es proporcional a la duración del pulso contada y se puede calcular con las siguiente formula (utilizando la velocidad del sonido = 340m/s):
Distancia (centímetros) = Tiempo medido x 0.017
(ya que la velocidad del sonido es 343 metros/segundo y es ida y vuelta -> distancia = tiempo / 343 * 1/2)

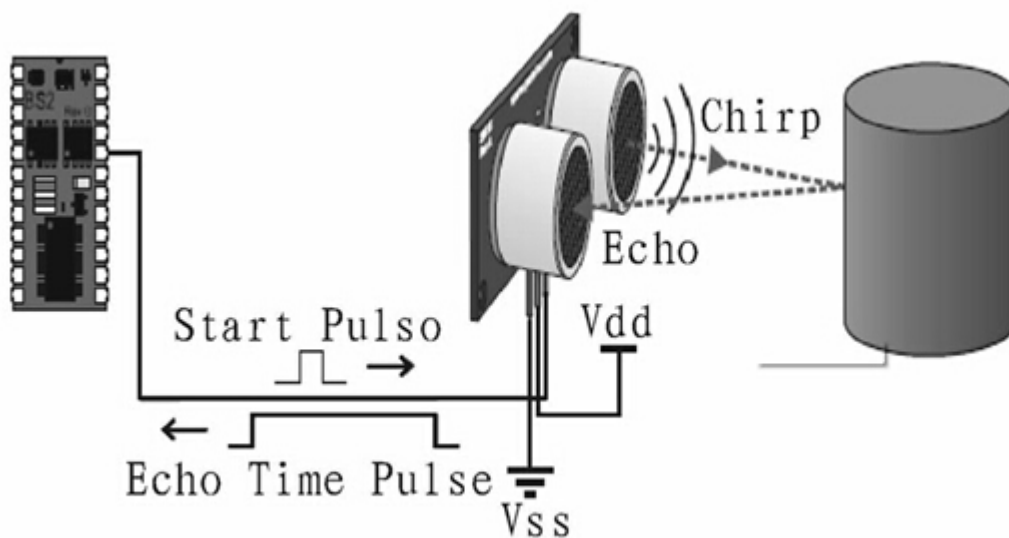
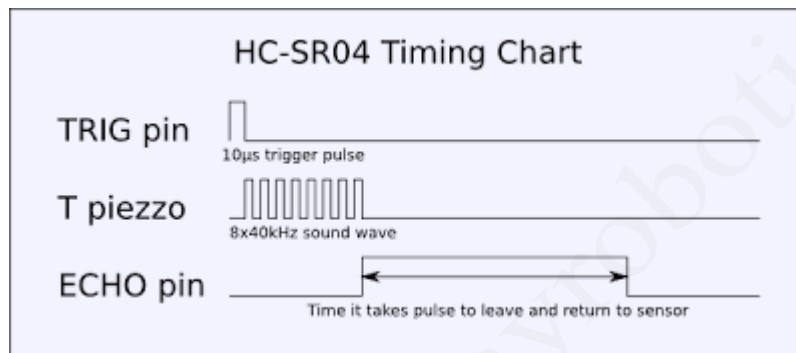


Figura 1.2.15-8 Principio de Funcionamiento de un sensor de ultrasonidos (de <http://www.electronics.com/wiki/index.php?title=BAT>)

Hay varias cosas que se tienen en cuenta a la hora de obtener la distancia:

- La velocidad del sonido en el aire (a una temperatura de 20 °C) es de 343 m/s. (por cada grado centígrado que sube la temperatura, la velocidad del sonido aumenta en 0,6 m/s)
- Estos sensores no se ven afectados por la luz solar o por el color de los materiales como ocurre con los sensores infrarrojos (aunque acústicamente materiales suaves como telas

pueden ser difíciles de detectar).

1.2.15.2. Robot evita obstáculos

En esta actividad vamos a utilizar los ultrasonidos como sensor para que un robot evite obstáculos a medida que se mueve (si ve un obstáculo delante se girará para encontrar un camino libre). Para ello utilizaremos el robot Printbot que estamos siguiendo en este libro, pero recuerda que puedes utilizar cualquier robot que tenga un Arduino compatible y un sensor de ultrasonidos.

Componentes.

- Robot Printbot
 - Tarjeta Arduino compatible ZumBT
 - 2 servos de rotación continua encargados del movimiento del robot
 - Sensor de ultrasonidos

Conexionado

El conexionado para esta actividad es el siguiente (ver Figura 1.2.15-9):

- Servo izquierdo → Pin 10
- Servo derecho → Pin 12
- Ultrasonidos
 - GND sensor – Cualquier pin negro (masa) de Arduino
 - VCC sensor – Cualquier pin rojo (+5v) de Arduino
 - ECH sensor – pin 4 Arduino
 - TRI sensor – pin 5 Arduino

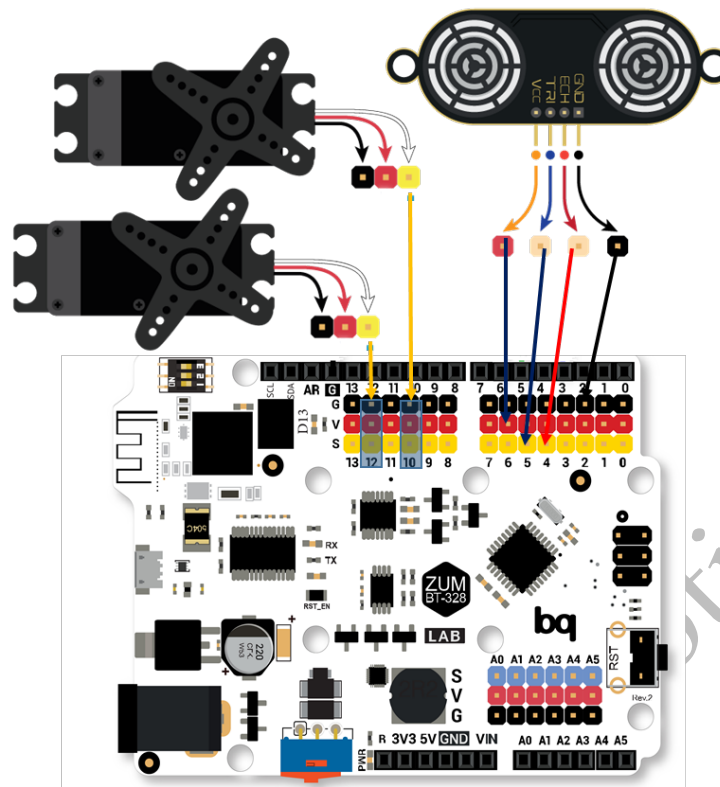


Figura 1.2.15-9 Conexión para un robot evita obstáculos

El mismo conexionado en el robot real puede observarse en la imagen de la Figura 1.2.15-10

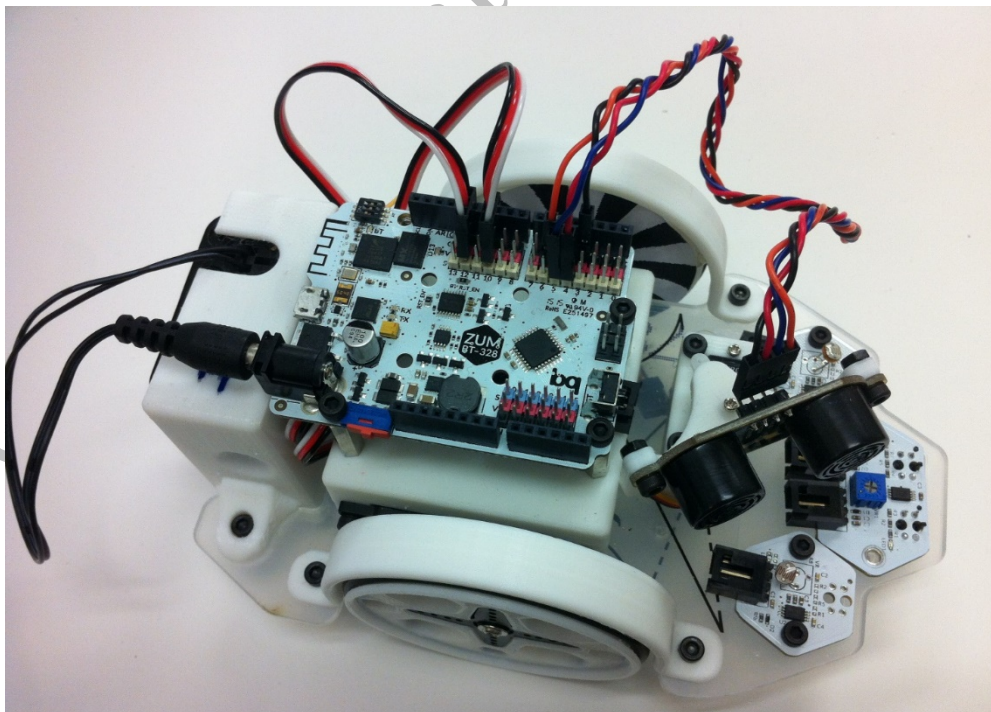


Figura 1.2.15-10 Robot Printbot con los componentes conectados para evitar obstáculos

Libro de Actividades de Robótica Educativa

En bitbloq procederemos a añadir la placa, el sensor y los dos servos en ventana de componentes tal y como se muestra en la Figura 1.2.15-11. Recuerda realizar el conexionado como lo hemos hecho en el robot real.

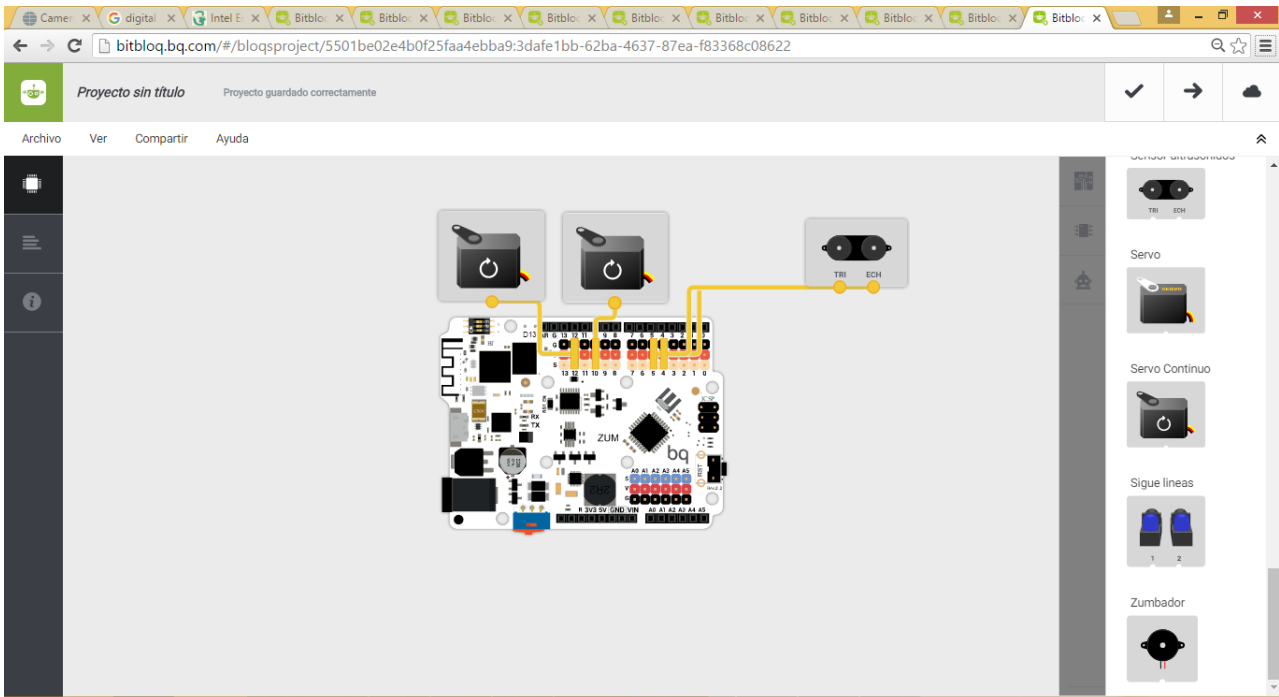


Figura 1.2.15-11 Conexionado en bitbloq de los componentes del robot evita obstáculos

Programación

Vamos a programar el siguiente comportamiento del robot:

- Si el sensor no hay nada a menos de 100 cm (es decir si el sensor devuelve una lectura mayor a 100) el robot sigue en línea recta
- Si el sensor detecta algo a menos de 100cm (caso contrario a lo anterior) el robot gira (hasta que no haya nada)

Ese comportamiento, programado en bloques queda así (Figura 1.2.15-12):

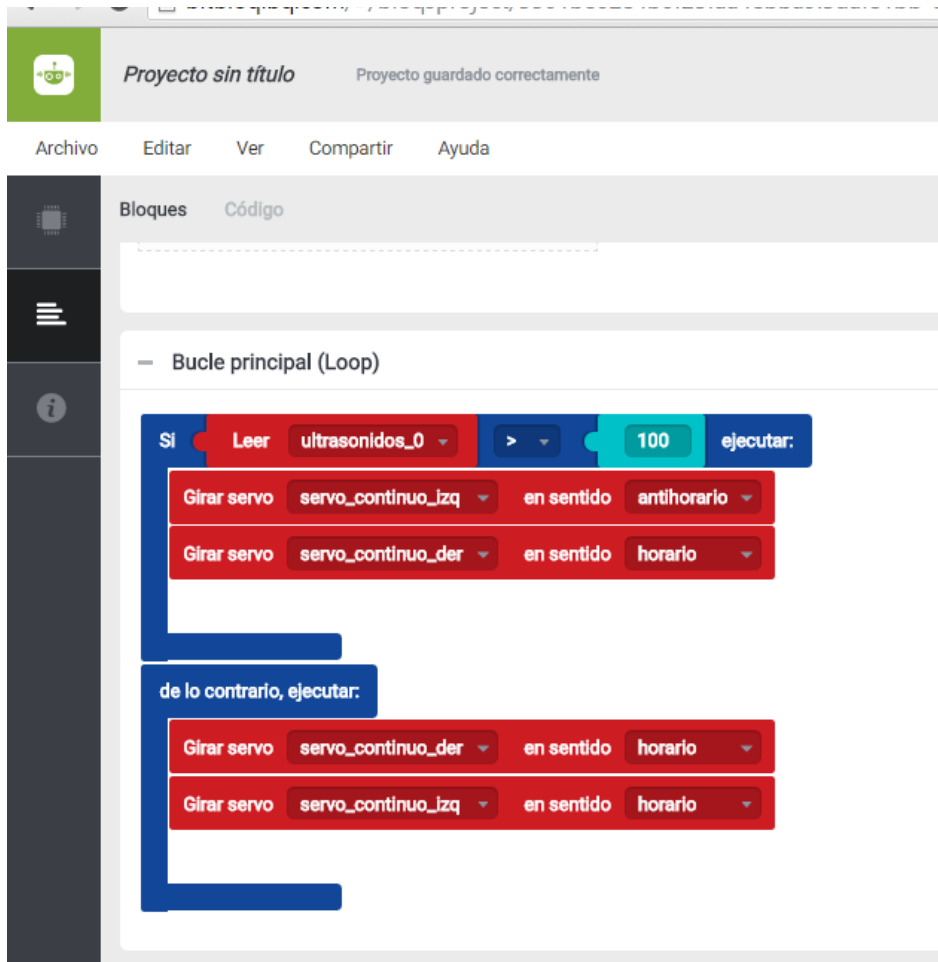


Figura 1.2.15-12 Programación por bloques de un robot evita obstáculos

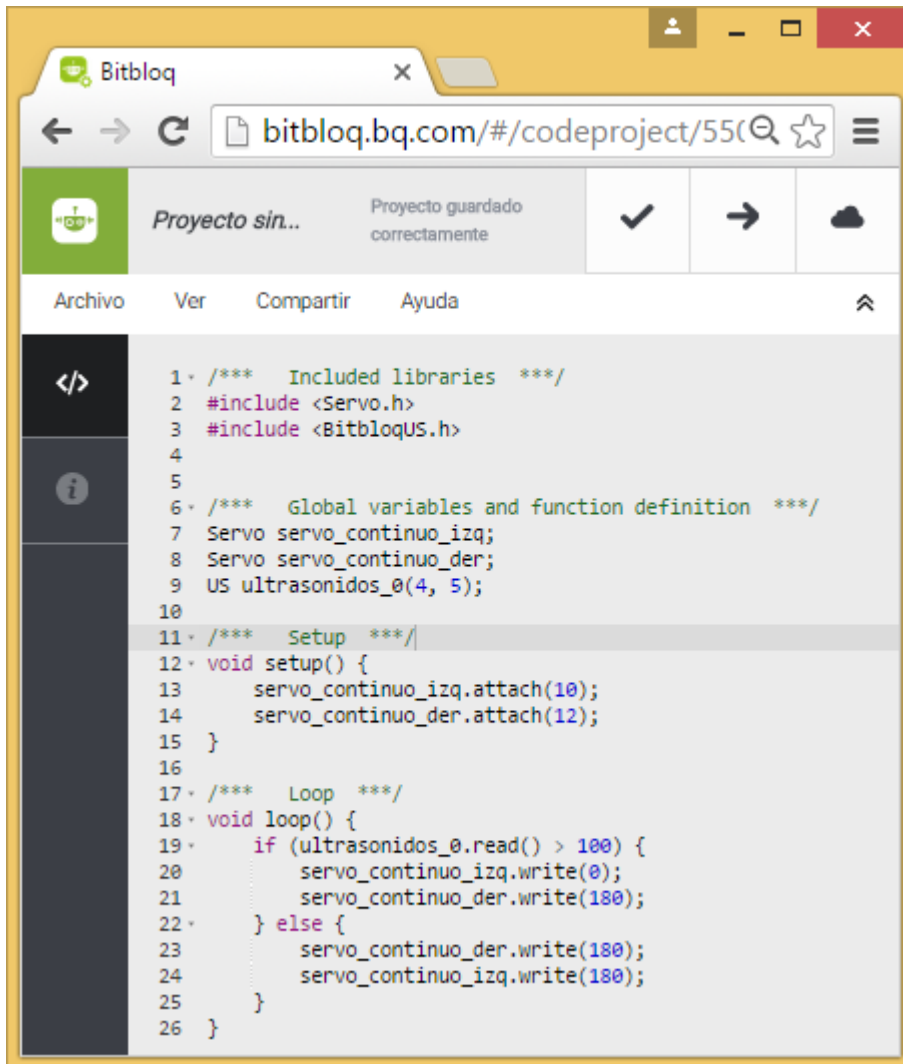
Como podemos ver, tenemos una condición inicial (si... ejecutar) que comprueba si el sensor está dando un valor superior a 100, si es así gira los motores de manera que el robot siga recto. Si no se cumple esa condición, es decir, si hay algo a menos de 100cm, los motores giran en la misma dirección haciendo que el robot gire sobre su eje. Como estas condiciones están en el bucle principal (loop), que se ejecuta continuamente, el robot estará continuamente leyendo del sensor y haciendo una cosa u otra en función de dicha lectura.

El código en el lenguaje de programación de Arduino es el siguiente:

```
/** Included libraries */  
#include <Servo.h>  
#include <BitbloqUS.h>  
  
/** Global variables and function definition */  
Servo servo_continuo_izq;  
Servo servo_continuo_der;  
US ultrasonidos_0(4, 5);  
  
/** Setup */  
void setup() {  
  servo_continuo_izq.attach(10);  
  servo_continuo_der.attach(12);  
}
```

Libro de Actividades de Robótica Educativa

```
/** Loop **/  
void loop() {  
  if (ultrasonidos_0.read() > 100) {  
    servo_continuo_izq.write(0);  
    servo_continuo_der.write(180);  
  } else {  
    servo_continuo_der.write(180);  
    servo_continuo_izq.write(180);  
  }  
}
```



```
1· /** Included libraries **/  
2· #include <Servo.h>  
3· #include <BitbloqUS.h>  
4  
5  
6· /** Global variables and function definition **/  
7· Servo servo_continuo_izq;  
8· Servo servo_continuo_der;  
9· US ultrasonidos_0(4, 5);  
10  
11· /** Setup **/  
12· void setup() {  
13·   servo_continuo_izq.attach(10);  
14·   servo_continuo_der.attach(12);  
15· }  
16  
17· /** Loop **/  
18· void loop() {  
19·   if (ultrasonidos_0.read() > 100) {  
20·     servo_continuo_izq.write(0);  
21·     servo_continuo_der.write(180);  
22·   } else {  
23·     servo_continuo_der.write(180);  
24·     servo_continuo_izq.write(180);  
25·   }  
26· }
```

Como vemos, utilizamos dos librerías, una para los servos y otra para el ultrasonido. Simplemente asignamos cada servo a los pines correspondientes con el método attach (líneas 13 y 14) y el sensor a sus pines (línea 9). En el loop comprobamos si el sensor nos da una lectura mayor que 100 con una condición if (línea 19). Si es así, y para que el robot se mueva en línea recta, tal como vimos en la actividad X, movemos un servo en una dirección y el otro en la dirección contraria (líneas 20 y 21). En el caso de que haya algo a menos de 100 movemos ambos motores en la misma dirección para que el robot gire sobre su eje (líneas 23 y 24).

1.2.15.3. Evita obstáculo con giro de cabeza

Libro de Actividades de Robótica Educativa

En esta actividad añadiremos un servo de posición para girar el sensor de ultrasonidos a izquierda y derecha. Esto permitirá que el robot mire a los lados a medida que anda y no solo mire hacia en frente.

Componentes.

- Robot Printbot
 - Tarjeta Arduino compatible ZumBT
 - 2 servos de rotación continua encargados del movimiento del robot
 - Sensor de ultrasonidos
 - 1 miniservo de posición encargado del movimiento del sensor de ultrasonidos

Conexionado

El conexionado para esta actividad es el siguiente:

- Servo izquierdo → Pin 10
- Servo derecho → Pin 12
- Miniservo → Pin 7
- Ultrasonidos
 - GND sensor – Cualquier pin negro (masa) de Arduino
 - VCC sensor – Cualquier pin rojo (+5v) de Arduino
 - ECH sensor – pin 4 Arduino
 - TRI senso – pin 5 Arduino

En la Figura 1.2.15-13 mostramos un esquema del conexionado y en la Figura 1.2.15-14 una imagen del printbot con los servos y el ultrasonido conectados.

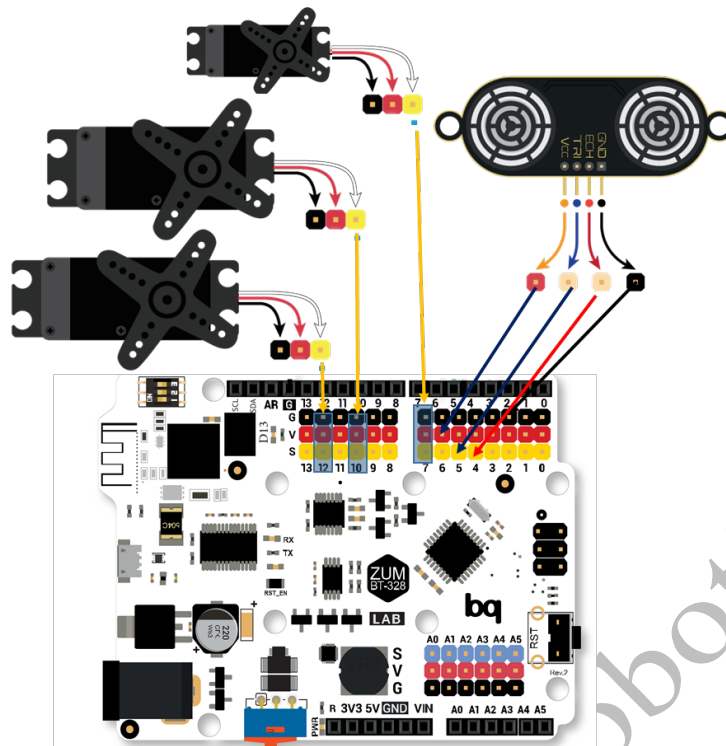


Figura 1.2.15-13 Conexionado de los componentes para un robot que evite obstáculos con cabeza rotante.

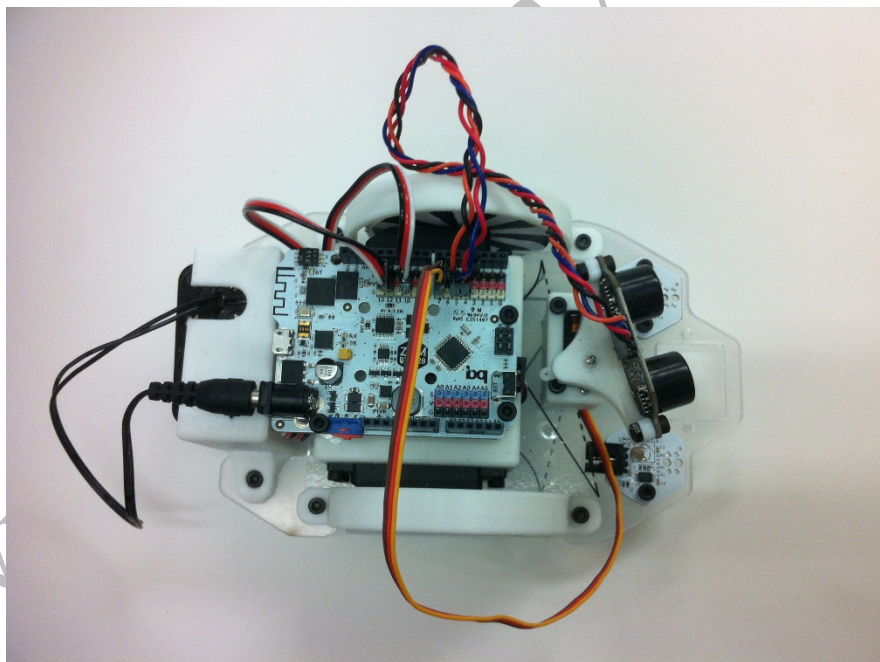


Figura 1.2.15-14 Robot Printbot con las conexiones para evitar obstáculos con cabeza rotante.

En bitbloq abriremos un proyecto nuevo, y en la pestaña de componentes añadiremos todos los componentes de esta actividad (placa Arduino compatible, servos y ultrasonido) tal como aparece en la figura X. Ten mucho cuidado de conectar los diferentes componentes a los pines que hemos asignado en el conexionado.

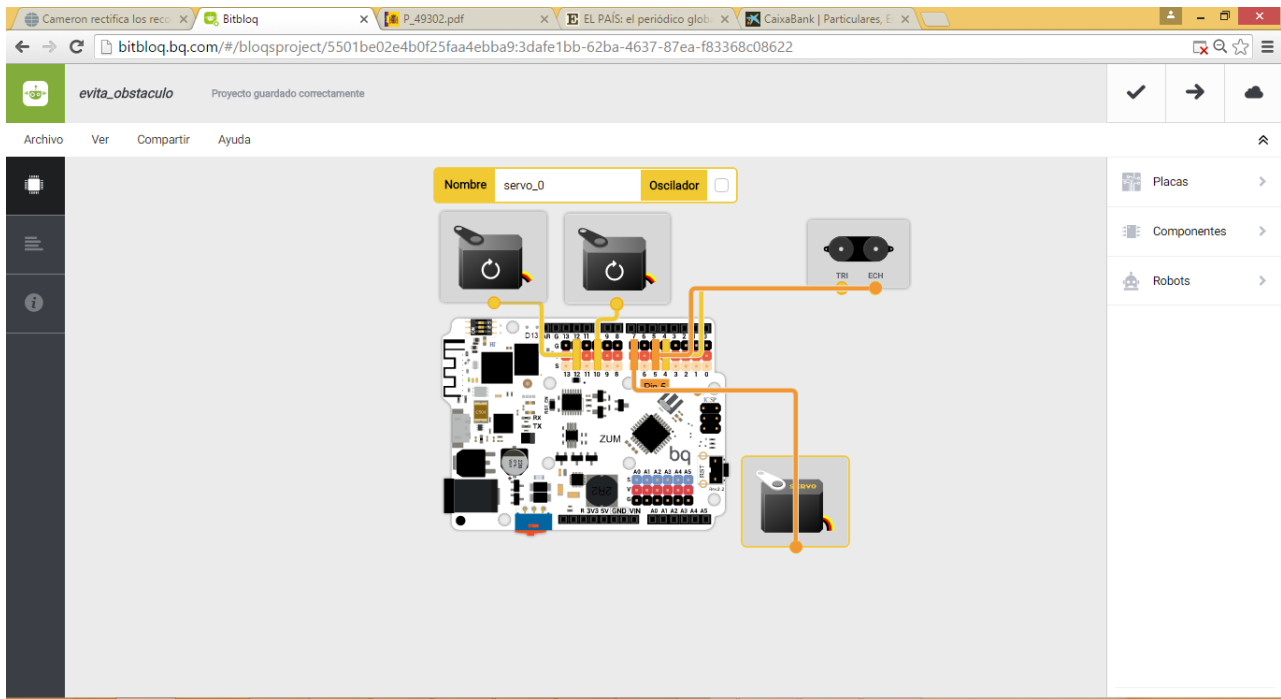


Figura 1.2.15-15 Conexión en Bitbloq de los componentes de la actividad

Programación

Vamos a programar el robot de manera que este girando la cabeza del sensor de un lado a otro y cuando encuentre un obstáculo el robot se mueva en dirección contraria de donde apunta la cabeza. Podemos resumir este comportamiento de esta manera:

- Si el sensor no detecta obstáculo (en cualquier posición de la cabeza) seguir en línea recta y va moviendo la cabeza de izquierda a derecha
- Si el sensor detecta un obstáculo el robot gira hacia el lado contrario de donde apunte la cabeza en ese momento. Es decir, si cuando detecta un obstáculo la cabeza está en alguna posición del centro a la izquierda, el robot gira a la derecha. Si la cabeza está en alguna posición del centro a la derecha, el robot gira hacia la izquierda.

Para realizar este comportamiento tendremos una variable donde guardamos la posición del servo de la cabeza. Le iremos dando valores a esta variable desde 0 a 180 para que el servo vaya girando, siendo de 0 a 90 los ángulos que hacen que la cabeza este mirando hacia la derecha y de 90 a 180 los ángulos que hacen que la cabeza mire hacia la izquierda.

Cuando detectemos un obstáculo comprobaremos la variable de posición del servo de la cabeza y moveremos los servos de rotación continua de los motores para girar hacia un lado u hacia otro.

El código de bloques resultante es el siguiente:

Libro de Actividades de Robótica Educativa

— Variables globales, funciones y clases

Declarar variable `servo_posicion` = `0`

— Instrucciones iniciales (Setup)

Indica lo que quieres que se ejecute una única vez al inicio del programa.

Arrastra un bloque aquí para empezar tu programa

— Bucle principal (Loop)

Mover Variable (componentes) `servo_0` a Variable `servo_posicion` grados

Si Leer `ultrasonidos_0` > `50` ejecutar:

- Girar servo `servo_continuo_izq` en sentido `antihorario`
- Girar servo `servo_continuo_der` en sentido `horario`
- Variable `servo_posicion` = Variable `servo_posicion` + `10`

de lo contrario, ejecutar:

Si Variable `servo_posicion` < `90` ejecutar:

- Girar servo `servo_continuo_der` en sentido `horario`
- Girar servo `servo_continuo_izq` en sentido `horario`

de lo contrario, ejecutar:

- Girar servo `servo_continuo_der` en sentido `antihorario`
- Girar servo `servo_continuo_izq` en sentido `antihorario`

Si Variable `servo_posicion` > `160` ejecutar:

- Variable `servo_posicion` = `0`

Y el código de Arduino es:

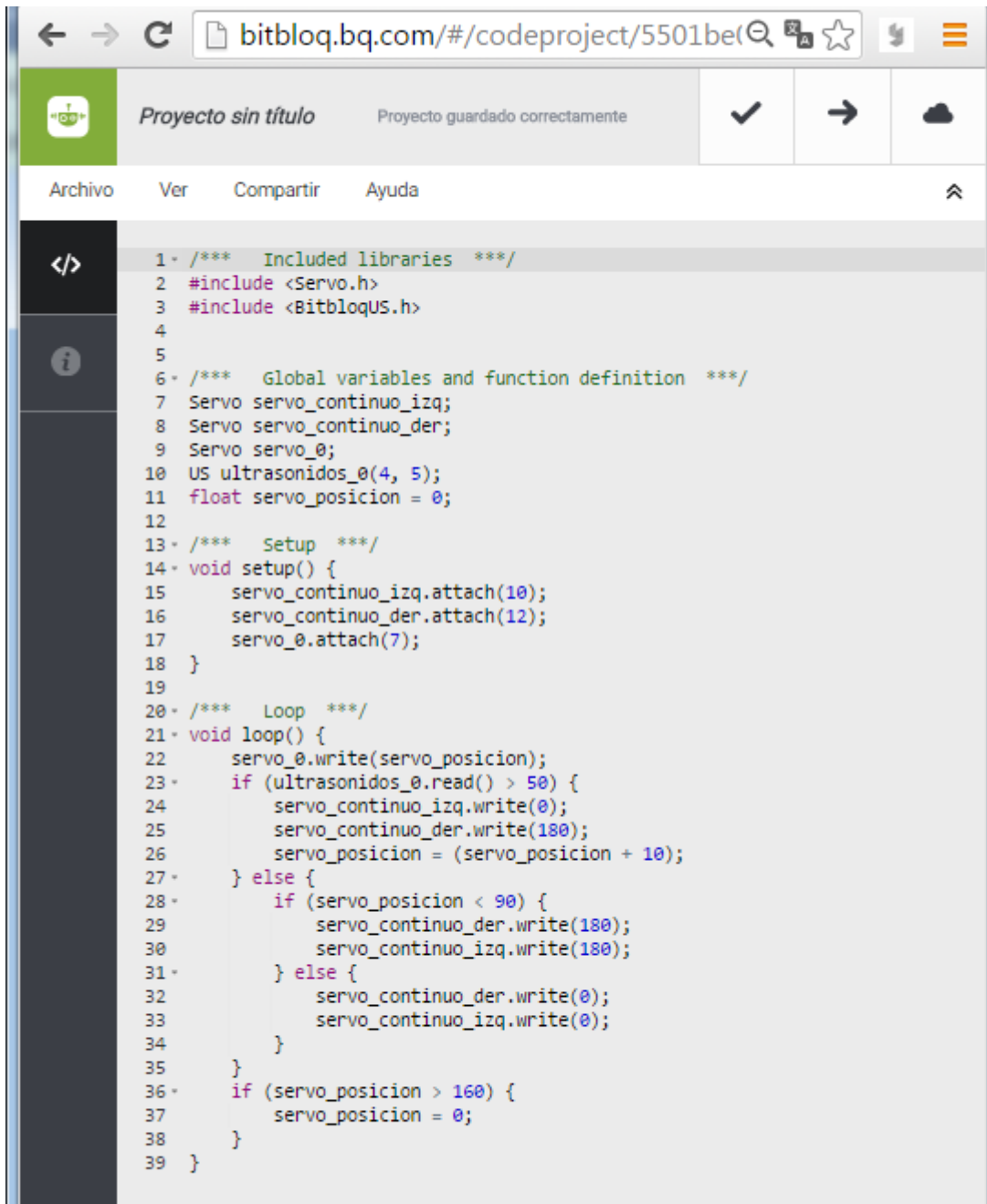
```
/** Included libraries */  
#include <Servo.h>  
#include <BitbloqUS.h>  
  
/** Global variables and function definition */  
Servo servo_continuo_izq;  
Servo servo_continuo_der;  
Servo servo_0;  
US ultrasonidos_0(4, 5);  
float servo_posicion = 0;
```

Libro de Actividades de Robótica Educativa

```
/** Setup ***/
void setup() {
  servo_continuo_izq.attach(10);
  servo_continuo_der.attach(12);
  servo_0.attach(7);
}

/** Loop ***/
void loop() {
  servo_0.write(servo_posicion);
  if (ultrasonidos_0.read() > 50) {
    servo_continuo_izq.write(0);
    servo_continuo_der.write(180);
    servo_posicion = (servo_posicion + 10);
  } else {
    if (servo_posicion < 90) {
      servo_continuo_der.write(180);
      servo_continuo_izq.write(180);
    } else {
      servo_continuo_der.write(0);
      servo_continuo_izq.write(0);
    }
  }
  if (servo_posicion > 160) {
    servo_posicion = 0;
  }
}
```

www.automatiza.com



```
1- /** Included libraries **/  
2 #include <Servo.h>  
3 #include <BitbloqUS.h>  
4  
5  
6- /** Global variables and function definition **/  
7 Servo servo_continuo_izq;  
8 Servo servo_continuo_der;  
9 Servo servo_0;  
10 US ultrasonidos_0(4, 5);  
11 float servo_posicion = 0;  
12  
13- /** Setup **/  
14 void setup() {  
15     servo_continuo_izq.attach(10);  
16     servo_continuo_der.attach(12);  
17     servo_0.attach(7);  
18 }  
19  
20- /** Loop **/  
21 void loop() {  
22     servo_0.write(servo_posicion);  
23     if (ultrasonidos_0.read() > 50) {  
24         servo_continuo_izq.write(0);  
25         servo_continuo_der.write(180);  
26         servo_posicion = (servo_posicion + 10);  
27     } else {  
28         if (servo_posicion < 90) {  
29             servo_continuo_der.write(180);  
30             servo_continuo_izq.write(180);  
31         } else {  
32             servo_continuo_der.write(0);  
33             servo_continuo_izq.write(0);  
34         }  
35     }  
36     if (servo_posicion > 160) {  
37         servo_posicion = 0;  
38     }  
39 }
```

Como vemos, mientras que no detecte a nada menos de 50cm (sentencia if de la línea 23) el robot se moverá en línea recta y estará girando la cabeza ya que iremos incrementando la variable `servo_posicion` de 10 en 10 (si la cabeza llega a 160 grados reiniciamos esa variable a 0). En el caso contrario, es decir que se detecte obstáculo (else de la línea 27), el robot girará hacia la derecha si el obstáculo está en la izquierda, es decir si el servo de la cabeza estaba mirando hacia el lado izquierdo cuando detectó obstáculo (if de la línea 28). Si el servo estaba mirando hacia el lado derecho (else de la línea 31) cuando se detecta obstáculo el robot girará hacia la izquierda.

1.2.15.4. Robot que nos sigue

En esta actividad utilizaremos los mismos componentes que en la actividad anterior, pero programaremos el robot para que nos siga en lugar de esquivarnos. El comportamiento del robot por tanto será prácticamente inverso del comportamiento de la actividad anterior. En concreto

Libro de Actividades de Robótica Educativa

podemos resumir el programa que queremos realizar en los siguientes comportamientos (ver Figura 1.2.15-16):

- Si el robot no detecta nada se queda parado moviendo la cabeza de un lado a otro
- Si detecta algo y el objeto está enfrente del robot (indicado porque la cabeza esté situada entre 70° y 110° , siendo 90° cuando la cabeza mira al frente) el robot sigue hacia delante
- Si detecta algo pero el objeto está a la izquierda del robot (indicado porque la cabeza esté situada entre 0° y 70°) el robot gira hacia la izquierda
- Si detecta algo pero el objeto está a la derecha del robot (indicado porque la cabeza esté situada entre 110° y 180°) el robot gira hacia la derecha.

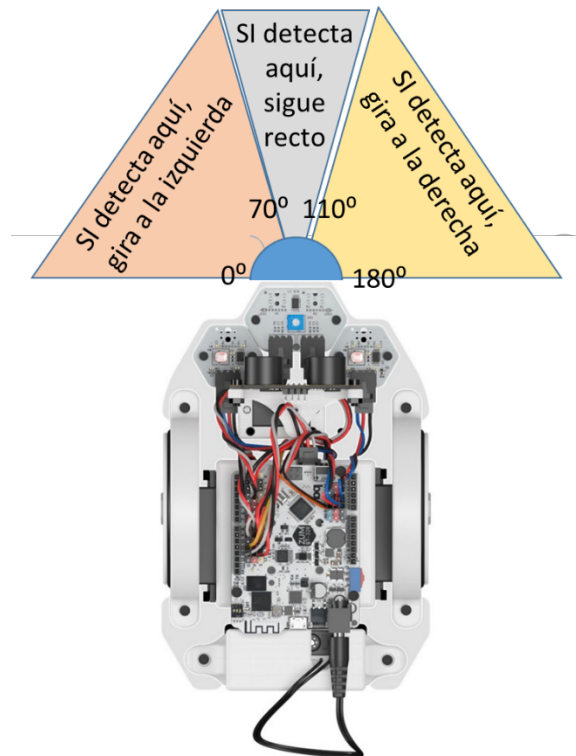


Figura 1.2.15-16 Comportamiento del robot que nos sigue

Componentes.

- Robot Printbot
 - Tarjeta Arduino compatible ZumBT
 - 2 servos de rotación continua encargados del movimiento del robot
 - Sensor de ultrasonidos
 - 1 miniservo de posición encargado del movimiento del sensor de ultrasonidos

Conexionado

El conexionado para esta actividad es el mismo que la actividad anterior (ver figura X):

Libro de Actividades de Robótica Educativa

- Servo izquierdo → Pin 10
- Servo derecho → Pin 12
- Miniservo → Pin 7
- Ultrasonidos
 - GND sensor – Cualquier pin negro (masa) de Arduino
 - VCC sensor – Cualquier pin rojo (+5v) de Arduino
 - ECH sensor – pin 4 Arduino
 - TRI senso – pin 5 Arduino

De nuevo, abriremos en bitbloq un proyecto nuevo, y en la pestaña de componentes añadiremos todos los componentes de esta actividad (placa Arduino compatible, servos y ultrasonido) tal como aparece en la Figura 1.2.15-17.

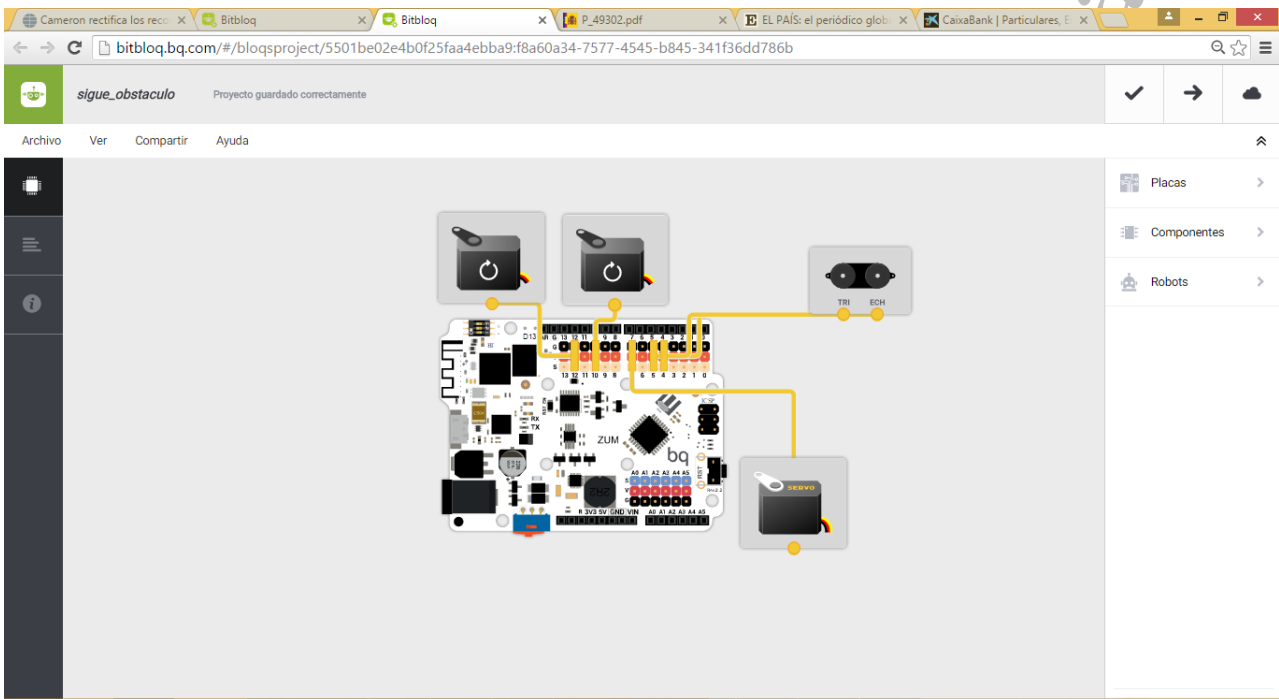


Figura 1.2.15-17 Conexión en bitbloq de los bloques necesarios para esta actividad

Programación

La programación de los comportamientos descritos anteriormente se realiza con los bloques siguientes:

Libro de Actividades de Robótica Educativa

The screenshot shows a programming interface with three main sections:

- Variables globales, funciones y clases:** A block to declare a variable named 'servo_posicion' with a value of 0.
- Instrucciones iniciales (Setup):** A placeholder for a single execution block.
- Bucle principal (Loop):** A main loop containing:
 - A 'Mover' block for 'servo_0' to 'servo_posicion' degrees.
 - A 'Si' block: 'Leer ultrasonidos_0' > 50, 'ejecutar':
 - 'Parar servo' for 'servo_continuo_der' and 'servo_continuo_izq'.
 - 'Variable servo_posicion' = 'Variable servo_posicion' + 10.
 - 'de lo contrario, ejecutar':
 - 'Si' block: 'Variable servo_posicion' < 70, 'ejecutar':
 - 'Girar servo' for 'servo_continuo_der' and 'servo_continuo_izq' in 'antihorario' direction.
 - 'en cambio, si' block: 'Variable servo_posicion' > 110, 'ejecutar':
 - 'Girar servo' for 'servo_continuo_der' and 'servo_continuo_izq' in 'horario' direction.
 - 'de lo contrario, ejecutar':
 - 'Girar servo' for 'servo_continuo_der' and 'servo_continuo_izq' in 'horario' direction.
 - 'Si' block: 'Variable servo_posicion' > 160, 'ejecutar':
 - 'Variable servo_posicion' = 0.

Utilizamos la variable servo_posicion para guardar la posición de la cabeza

Movemos la cabeza a servo_posicion

Si no hay obstáculo nos quedamos parados y aumentamos servo_posición para seguir moviendo la cabeza

Si hay un obstáculo:
Si cabeza entre 0°-70°
Girar izquierda

Si cabeza entre 110°-180°
Girar derecha

Si cabeza entre 70°-110°
Ir recto

Si la posición cabeza es mayor que 160° entonces volvemos a la posición 0° (para girar la cabeza de un

Y el código resultante en Arduino es:

```
/** Included libraries */  
#include <Servo.h>  
#include <BitbloqUS.h>  
  
/** Global variables and function definition */  
Servo servo_continuo_izq;  
Servo servo_continuo_der;  
Servo servo_0;  
US ultrasonidos_0(4, 5);  
float servo_posicion = 0;  
  
/** Setup */  
void setup() {  
  servo_continuo_izq.attach(10);
```

Libro de Actividades de Robótica Educativa

```
servo_continuo_der.attach(12);
servo_0.attach(7);
}

/** Loop */
void loop() {
  servo_0.write(servo_posicion);
  if (ultrasonidos_0.read() > 50) {
    servo_continuo_der.write(90);
    servo_continuo_izq.write(90);
    servo_posicion = (servo_posicion + 10);
  } else {
    if (servo_posicion < 70) {
      servo_continuo_der.write(0);
      servo_continuo_izq.write(0);
    } else if (servo_posicion > 110) {
      servo_continuo_der.write(180);
      servo_continuo_izq.write(180);
    } else {
      servo_continuo_der.write(180);
      servo_continuo_izq.write(0);
    }
  }
  if (servo_posicion > 160) {
    servo_posicion = 0;
  }
}
```

www.automatiza.com