CURSO: ROBÓTICA. CONECTANDO CON EL MUNDO FÍSICO

1 Nombre, apellidos y correo electrónico de EducaMadrid del participante

JAVIER MARTÍN DE LUCAS
javier.martindelucas@educa.madrid.org
JUAN CARLOS NIÑO SÁINZ-MAZA
juancarlos.ninosaimaza@educa.madrid.org

2 Título de la Unidad Didáctica

RADAR DE 180º

3 Curso

3. º DE ESO en el año de implantación (sin base previa de 1º y 2º de ESO

4 Descripción del proyecto

Sistema elemental de radar que barre 180º usando un servomotor. Calcula la distancia a los objetos más cercanos por medio de ultrasonidos e utiliza es información para avisar mediante sonido de la cercanía de los objetos.

5 Competencias que se trabajarán

- Uso de actuadores y sensores estándar.
- Control de movimiento mediante servomotores.
- Iniciación al uso de funciones
- Organización de proyectos.
- Diseño de estrategias para conseguir un objetivo.

6 Objetivos

Describir el proceso de desarrollo de un programa y enumera sus fases principales.
Emplear las herramientas básicas del entorno de programación.
Iniciar y detener la ejecución de un programa.
Manejar los principales grupos de bloques del entorno de programación.
Utilizar los comandos de control de ejecución: condicionales y bucles.
Emplear de manera adecuada variables y listas.
Usar con soltura la interacción entre los elementos de un programa.
Analizar el funcionamiento de un programa a partir de sus bloques.

7 Contenidos

Gestión simultánea de varios dispositivos: servomotor, zumbador y sensor de ultrasonidos. Variables numéricas incrementales y funciones simples. Cálculos numéricos: distancia en función del eco y velocidad del sonido. Funciones.

8 Criterios de evaluación

Utilizar con destreza un entorno de programación gráfica por bloques.

9 Código del programa de Arduino

```
#include <Servo.h>
Servo servos[13];
/***   Global variables   ***/
/***   Function declaration   ***/
//bqBAT
long TP_init(int trigger_pin, int echo_pin);
long Distance(int trigger_pin, int echo_pin);
// La FUNCIÓN "sonar"
// 1. guarda la distancia en cm proporcionada por el emisor/receptor de ultrasonidos.
// 2. emite un pitido si la distancia es menor de 10 cm
// 3. En cualquier caso deja un retardo de 200 ms antes de devolver el control al servo
para que gire otros 5º
void sonar ();
```

```
void setup()
{
  servos[10].attach(10);
  pinMode( 2 , INPUT );
  pinMode( 3 , OUTPUT );
}

void loop()
{
  int i=0;
  // El BUCLE PRINCIPAL  del programa
  // cuenta de 5º en 5º (180º/36)
  // primero en sentido horario y luego antihorario
  // y sitúa en esa posición el servomotor que hace girar el emisor/receptor de
ultrasonidos,
  // de modo que éste hace un barrido de 180º en cada sentido dentro de ese bucle principal
  for (i = 0; i <= 35; i++) {
    servos[10].write(i * 5);
    delay(0);
    sonar();
  }
  for (i = 35; i >= 0; i--) {
    servos[10].write(i * 5);
    delay(0);
    sonar();
  }
}

/***   Function definition   ***/
//bqBAT
long TP_init(int trigger_pin, int echo_pin)
{
  digitalWrite(trigger_pin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigger_pin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigger_pin, LOW);
  long microseconds = pulseIn(echo_pin ,HIGH);
  return microseconds;
}
long Distance(int trigger_pin, int echo_pin)
{
  long microseconds = TP_init(trigger_pin, echo_pin);
  long distance;
  distance = microseconds/29/2;
  if (distance == 0){
    distance = 999;
  }
  return distance;
}
// La FUNCIÓN "sonar"
// 1. guarda la distancia en cm proporcionada por el emisor/receptor de ultrasonidos.
// 2. emite un pitido si la distancia es menor de 10 cm
// 3. En cualquier caso deja un retardo de 200 ms antes de devolver el control al servo
para que gire otros 5º
void sonar () {
  int distancia=Distance(3,2);
  if (distancia < 10) {
    tone(9,494,200);
    delay(200);
   }else {
    delay(200);
   }
 }
```

SONAR 180º
CON ALARMA DE PROXIMIDAD

SOY HARDWARE

SENSOR ULTRASONIDOS

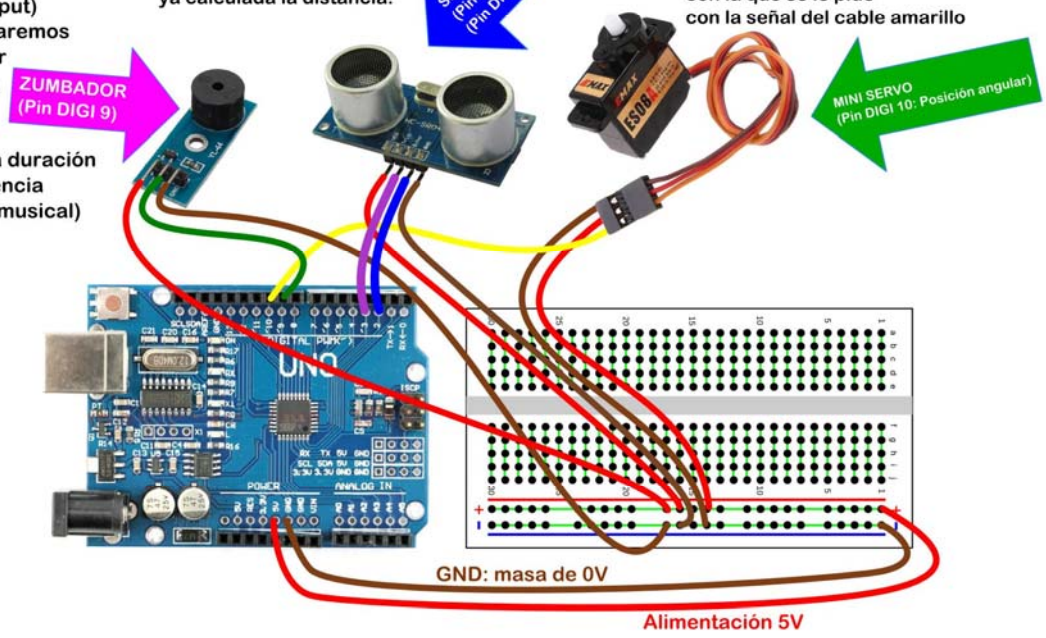
Sirve para calcular la distancia a la que está un objeto del mismo modo que lo hacen los murciélagos: Emite ultrasonidos y es capaz de escuchar los ecos. Quien da las órdenes de emitir es la placa arduino y se lo dice al pin "Trigger".
Quien recibe el momento del eco es también la placa y lo escucha del pin "Echo"
Con esos datos la placa podrá calcular la distancia pues tiene la velocidad del sonido (340 m/s) y el tiempo transcurrido entre el disparo del sonido y la recepción de su eco.
Como es una tarea repetitiva se define una función dentro del código de la placa: le suministramos los datos del "trigger" y del "echo" y la función nos devolverá ya calculada la distancia.

MINI SERVO

Interiormente es un simple motor de CC con un juego de engranajes que desmultiplica mucho la velocidad de giro, dándole, en cambio, más potencia en el par de giro.
El cable rojo (5V) y el marrón de masa le dan energía para el giro.
El tercer cable (amarillo) sirve para consignarle la posición deseada, que puede ir de 0 a 180º.
Conectado al eje de salida gira solidario un potenciómetro circular que va leyendo la posición de giro de ese eje.
En su base hay electrónica propia que procesa y compara esa posición con la que se le pide con la señal del cable amarillo

ZUMBADOR

Se puede usar como actuador (suena), pero también como sensor (micrófono). Por eso su pin central, el de la señal, está etiquetado como IO (Input/Output)
Aquí lo usaremos para emitir un sonido, del que podemos escoger la duración y la frecuencia (la "nota" musical)



Todos los sensores y actuadores están alimentados con 5V y una masa (GND) que actúa como 0V para esa alimentación y como masa de señal o referencia 0 para comparar las señales que entran y/o salen.
Si hay varios componentes es conveniente usar el BUS de + y - de la Protoboard para que todos tomen voltaje y masa de allí.
Si algunos de los actuadores necesita más potencia de la que sale de la placa Arduino, voltaje y masa se suministran aparte. Es el caso, por ejemplo, de los motores de CC, que necesitan, además, un componente específico: el puente en H

Hacer conexiones de prueba en "Protoboard"

También se llama a este tipo de placas "Breadboard" y las hay de distintos tamaños. Ésta que empleamos aquí es corta y, además, le hemos suprimido arriba dos líneas de conexiones largas, idénticas a las que están abajo marcadas con "+" y "-".
Sirven para hacer conexiones sin soldar para probar un circuito con componentes.
Por dentro lleva conexiones ocultas cada 5 huecos en vertical (30+30=60 nodos) y dos nodos largos de 30 huecos a modo de "bus": uno para el positivo de voltaje y otro para el negativo (o masa o GND).
Si pincho algo en el "1a" y otro componente en el "1e" es como si esos terminales estuvieran soldados. En el dibujo se han marcado en verde las conexiones ocultas.

SOY SOFTWARE

La idea "humana" es:

Simular un radar de sonido (sonar) que barra una zona semicircular y que utilice la información para, por ejemplo, avisar de que algún objeto está demasiado cerca (10 cm en nuestro caso)

Tres tareas = tres elementos:

1. El "sonar" será un emisor-receptor de ultrasonidos.

Una función propia se encarga de procesar las señales que salen y el eco que retorna, de modo que un sencillo cálculo con la velocidad del sonido nos dé el resultado.
Lo almacenamos en la variable "distancia"

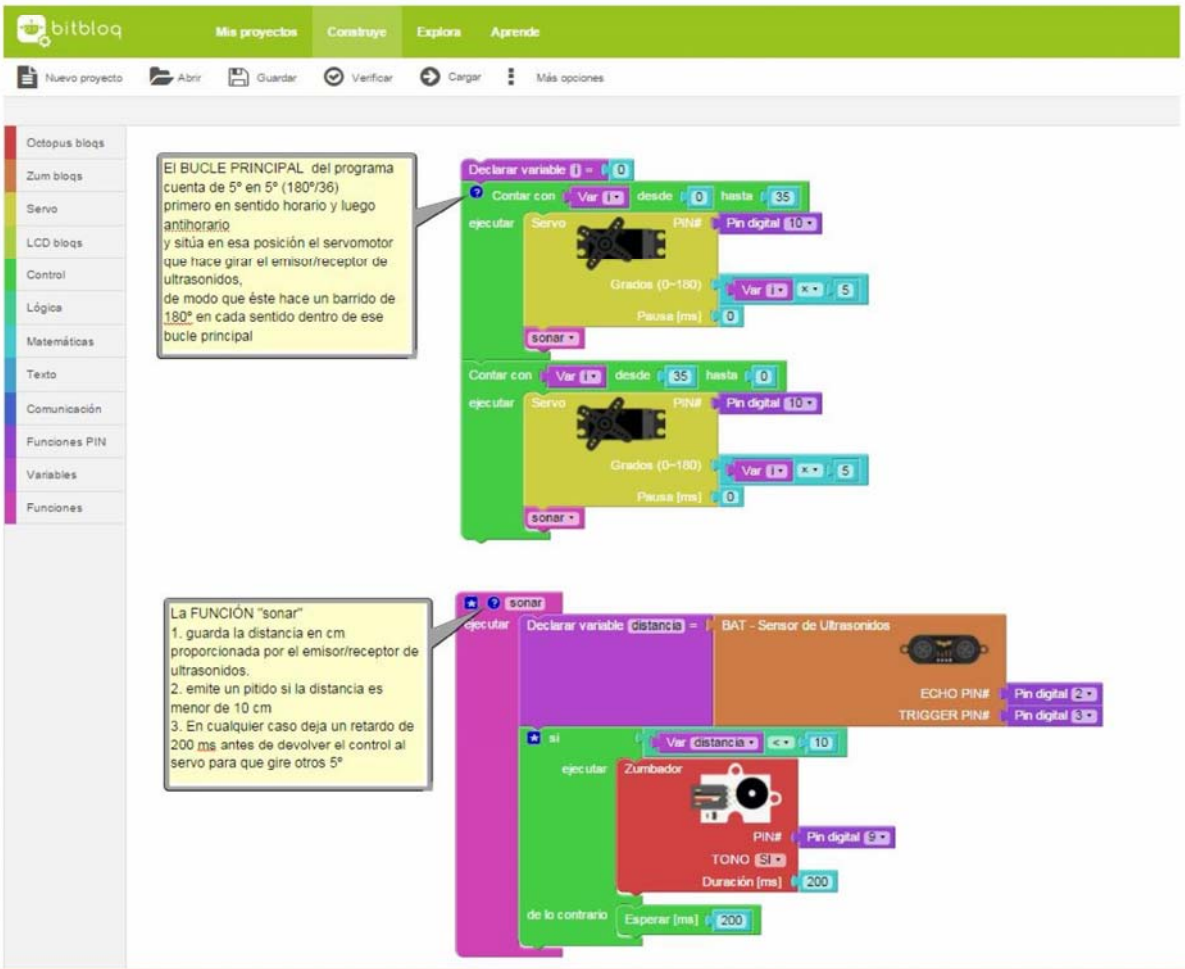
2. El barrido circular lo hacemos con un miniservo de 180º, sobre el que se monta el sonar.

Hay que darle la posción angular en grados.
Una bilioteca propia se incluye y funde con el código que nosotros escribimos.

3. La alarma de proximidad es tarea de un sencillo zumbador

Algo se repite: hay que convertirlo en una "función"

Después de cada micro-giro (5º) hay que hacer siempre lo mismo: recibir la distancia del sensor de ultrasonidos y decidir si hay que hacer sonar la alarma de proximidad o no.
La función se crea y se le da un nombre ("sonar") juntando todo el código necesario.
Sólo hay que llamarla por su nombre cada vez que queremos que se ejecute.
Esta función nuestra es "muda": no dice nada, sólo hace.
Hay otras que devuelven algún dato interesante. El componente de ultrasonidos en realidad se gestiona como una función de esas que retornan información.
Definir funciones aporta claridad a nuestro código y permite cambiar las cosas de un plumazo.
Otra buena costumbre: comentar nuestro código, explicándolo. No sólo es bueno para que otros entiendan lo que hacemos, sino para entender nosotros mismos en el futuro lo que un día hicimos



11 Fotografía del montaje