----------VERSIÓN FINAL----------

```
#include <R3D3.h>

// Creamos un objeto de la clase r3d3.
R3D3 myR3D3(11);

// define los números de los pins
const int trigPin_1 = 12 ;
const int echoPin_1 = 13 ;
int microfono = 8;

// define variables
float duracion ;
float distancia;
float distancia1 ;
float distancia2 ;
float distancia3 ;
byte distancia10 ;
byte distancia20 ;
byte distancia30 ;
float distanciaMedia ;
int valorMic;
int empezar = 0;

////////////////////////////////////////////////////////////////// Funcion de preparacion de modulos
void setup ( ) {
  pinMode( trigPin_1 , OUTPUT ) ;   // Establece el trigPin como salida
  pinMode( echoPin_1 , INPUT ) ;  // Establece el echoPin como entrada
  pinMode( microfono , INPUT) ; //Establece el microfono como entrada
  valorMic = 0;

//  headServo.attach(12);
myR3D3.Init();
  Serial.begin( 9600 ) ;    // Inicia la comunicación serial para debug

  while (!myR3D3.getPulsador(pulsador_A));  // Espera a que se pulse
}

//////////////////////////////////////////////////////// FUNCION PRINCIPAL ////////////////////////////////////////////////////////////

void loop ( ) {

      //myR3D3.beep((byte)2);
      //while(1);

      // myR3D3.testServo();
      /*myR3D3.headPosition(HeadLeft);*/

      // myR3D3.stop(); while(1);
      // muevo es robot hacia delante
```

```
      //myR3D3.moveForward(100);
      //delay(100);

   valorMic = digitalRead(microfono); //Dar a valorMic el valor del microfono
   while ((valorMic != 1)&&(empezar<=0)){ //Establecer empezar como 1 si se detecta una palmada
     valorMic = digitalRead(microfono); //Dar a valorMic el valor del microfono
     myR3D3.stop();
   }
   if (empezar<=0){
     empezar=4;
   }
// Juego de cabeza mirando
// 1. Colocar la cabeza mirando al frente y avanzar
      myR3D3.headPosition(112);
      myR3D3.headPosition(90);
      delay(400);

      myR3D3.moveForward(100);

// 2. Mirar si hay objeto delante. Si lo hay, giro
      distanciaMedia = ultrasonido_medio();
      if(distanciaMedia < 50){
        myR3D3.moveBack(100);
        myR3D3.beep(100,2);
        delay(1000);
        myR3D3.turnRight(180, 100);
        myR3D3.moveForward(100);
        empezar--;
      }

// 3. Colocar la cabeza mirando a la izquierda y mirar si hay objeto
      myR3D3.headPosition(72);
      myR3D3.headPosition(45);
      delay(400);

      distanciaMedia = ultrasonido_medio();
      if(distanciaMedia < 50){
        myR3D3.moveBack(100);
        myR3D3.beep(100,2);
        delay(1000);
        myR3D3.turnRight(180, 100);
        myR3D3.moveForward(100);
         empezar--;
      }
// 3.1. Colocar la cabeza mirando al frente y mirar
      myR3D3.headPosition(72);
      myR3D3.headPosition(90);
      delay(400);
      distanciaMedia = ultrasonido_medio();
      if(distanciaMedia < 50){
        myR3D3.moveBack(100);
```

```
          myR3D3.beep(100,2);
          delay(1000);
          myR3D3.turnRight(180, 100);
          myR3D3.moveForward(100);
           empezar--;
        }

// 4. Colocar la cabeza mirando a la izquierda y repetir
        myR3D3.headPosition(112);
        myR3D3.headPosition(135);
        delay(400);
        distanciaMedia = ultrasonido_medio();
        if(distanciaMedia < 50){
          myR3D3.moveBack(100);
          myR3D3.beep(100,2);
          delay(1000);
          myR3D3.turnRight(180, 100);
          myR3D3.moveForward(100);
           empezar--;
        }
       // if ( pulsador == ON)  myR3D3.stop();
}//loop

////////////////////////////////////////////// Función para calcular distancia desde un sensor de ultraSonido

float  ultrasonicDistance( byte sensorNum) {

  float dist=0;

        // Borra el trigPin
        digitalWrite ( trigPin_1 , LOW ) ;
        delayMicroseconds ( 20 ) ;
        // Establece el trigPin en estado ALTO durante 10 micro segundos
        // 1.- Lanzamos el pulso de sonido
        digitalWrite( trigPin_1 , HIGH ) ;
        delayMicroseconds( 10 ) ;
        digitalWrite ( trigPin_1 , LOW ) ;
        // 2.- Lee el echoPin, devuelve el tiempo de viaje de la onda de sonido en uSg
        duracion = pulseIn( echoPin_1 , HIGH ) ;
        // 3.- Calculando la distancia en centimetros ??
        dist = duracion * 0.034 / 2 ;

        /*
        // DEBUG:: Imprime la distancia en el monitor serie
        Serial.print ( "Distancia:" ) ;
        Serial.println ( dist ) ;
        */

        return (dist);
}// fin ultasonicDistance
```

```
////////////////////////////////////////////// Función para calcular la muestra de la mitad de 3 muestras

float  ultrasonido_medio(){

  //Reiniciar los valores
  distancia10 =1;
  distancia20 =1;
  distancia30 =1;

  //Tomar 3 muestras
  distancia1 = ultrasonicDistance(1);
  delayMicroseconds(50);
  distancia2 = ultrasonicDistance(1);
  delayMicroseconds(50);
  distancia3 = ultrasonicDistance(1);

  //Hallar el más alto
  if ((distancia1 > distancia2) && (distancia1 > distancia3)){
    distancia10 = 0;
  }
  if ((distancia2 > distancia1) && (distancia2 > distancia3)){
    distancia20 = 0;
  }
  if ((distancia3 > distancia1)&&(distancia3 > distancia2)){
    distancia30 = 0;
  }

  //Hallar el más bajo
  if ((distancia1 < distancia2) && (distancia1 < distancia3)){
    distancia10 = 0;
  }
  if ((distancia2 < distancia1) && (distancia2 < distancia3)){
    distancia20 = 0;
  }
  if ((distancia3 < distancia1)&&(distancia3 < distancia2)){
    distancia30 = 0;
  }

/*
  //DEBUG
  Serial.print ("Distancia1: ");
  Serial.print (distancia1);
  Serial.print (" ");
  Serial.println (distancia10);
  Serial.print ("Distancia2: ");
  Serial.print (distancia2);
  Serial.print (" ");
  Serial.println (distancia20);
  Serial.print ("Distancia3: ");
  Serial.print (distancia3);
  Serial.print (" ");
```

```
  Serial.println (distancia30);
*/

  //Devolver el valor medio
  if (distancia10 != 0){
    return(distancia1);
  }
  if (distancia20 != 0){
    return(distancia2);
  }
  if (distancia30 != 0){
    return(distancia3);
  }
}//fin ultrasonidos_medio
```

----------Libreria .cpp----------

```
/*
  R3D3
  Libreria para controlar el robot de Kiddybot R3D3
    Created by J3Viton [Kiddybot], November, 2017.


    Hardware:
        DC motor Driver:
        L911O h bridge  http://www.gie.com.my/UploadFiles/robotics/drivers/motor_waveform.jpg
*/

#include "Arduino.h"
#include "R3D3.h"

#define CORRECION_ANGULO  4


Servo headServo;   /* inhabilita pines 9 y 10 */


R3D3::R3D3(int pin)
{
  // Preparo los pines de salida
  pinMode(MotorA_IA, OUTPUT);
  pinMode(MotorA_IB, OUTPUT);
  pinMode(MotorB_IA, OUTPUT);
  pinMode(MotorB_IB, OUTPUT);

  //pinMode(11, OUTPUT);
  _pin = pin;

  //  pinMode(LED_RED, OUTPUT);
  //  pinMode(LED_GREEN, OUTPUT);
}
```

```
/*******************************************************
Init. funciÃ³n para inicializar el robot

*******************************************************/
void R3D3::Init(void){
  headServo.attach(12);
   pinMode(pulsador_A, INPUT);
   pinMode(zumb, OUTPUT);
}

/*********************************************************************
test_bridgeH
Funcion para testeo y debugg del resto de la clase y sus metodos.
toDo:
 .- aÃ±adir el control PWM
 .- controlar motor + velocidad + sentido
 .- ajustar angulos de giro.


*********************************************************************/
void R3D3::test_bridgeH(int motor, int speed, int rotation){


   if(motor == Motor_A){
      // Turn Righ
      analogWrite(MotorA_IA, speed);    //PWM
      digitalWrite(MotorA_IB, LOW);
      //digitalWrite(LED_GREEN, HIGH);
      delay(2000);
      // Stop
      digitalWrite(MotorA_IA, LOW);
      digitalWrite(MotorA_IB, LOW);
      //digitalWrite(LED_GREEN, LOW);
      delay(1000);
      // Turn Left
      analogWrite(MotorA_IA, (255-speed));    //PWM
      digitalWrite(MotorA_IB, HIGH);
      //digitalWrite(LED_RED, HIGH);
      delay(2000);
      // Stop
      digitalWrite(MotorA_IA, LOW);
      digitalWrite(MotorA_IB, LOW);
      //digitalWrite(LED_RED, LOW);
      delay(1000);
    }
    else if(motor=Motor_B){
      // Turn Righ
      analogWrite(MotorB_IA, speed);    //PWM
      digitalWrite(MotorB_IB, LOW);
      //digitalWrite(LED_GREEN, HIGH);
```

```
      delay(2000);
      // Stop
      digitalWrite(MotorB_IA, LOW);
      digitalWrite(MotorB_IB, LOW);
      //digitalWrite(LED_GREEN, LOW);
      delay(1000);
      // Turn Left

      analogWrite(MotorB_IA, (255-speed));    //PWM
      digitalWrite(MotorB_IB, HIGH);
      //digitalWrite(LED_RED, HIGH);
      delay(2000);
      // Stop
      digitalWrite(MotorB_IA, LOW);
      digitalWrite(MotorB_IB, LOW);
      //digitalWrite(LED_RED, LOW);
      delay(1000);
    }

  }//test_bridgeH  FIN

void R3D3::moveForward(int speed){
  analogWrite(MotorA_IA, speed);    //PWM
  digitalWrite(MotorA_IB, LOW);
  analogWrite(MotorB_IA, speed);    //PWM
  digitalWrite(MotorB_IB, LOW);
}//moveForward

void R3D3::moveBack(int speed){
  analogWrite(MotorA_IA, 255-speed);    //PWM
  digitalWrite(MotorA_IB, HIGH);
  analogWrite(MotorB_IA, 255-speed);    //PWM
  digitalWrite(MotorB_IB, HIGH);
}//moveForward

// Metodo para parar los motores
void R3D3::stop(void){

  digitalWrite(MotorA_IA, LOW);
  digitalWrite(MotorA_IB, LOW);
  digitalWrite(MotorB_IA, LOW);
  digitalWrite(MotorB_IB, LOW);
}//stop

void R3D3::turnRight(int angle, int speed){
  // Turn Right,Pivotando.
  analogWrite(MotorA_IA, speed);    //PWM
  digitalWrite(MotorA_IB, LOW);
  analogWrite(MotorB_IA, 255-speed);    //PWM
  digitalWrite(MotorB_IB, HIGH);
  delay((unsigned long) (angle * CORRECION_ANGULO));  // tiempo de giro
```

```
  stop();              // Paramos motores
}//turnRight

void R3D3::turnLeft(int angle, int speed){
  // Turn Left,Pivotando.
  analogWrite(MotorB_IA, speed);    //PWM
  digitalWrite(MotorB_IB, LOW);
  analogWrite(MotorA_IA, 255-speed);    //PWM
  digitalWrite(MotorA_IB, HIGH);
  delay((unsigned long) (angle * CORRECION_ANGULO));  // tiempo de giro
  stop();              // Paramos motores
}//turnLeft


void R3D3::testServo(void){
  int pos;

   for (pos = 30; pos <= 150; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    headServo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  for (pos = 150; pos >= 30; pos -= 1) { // goes from 180 degrees to 0 degrees
    headServo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }


}//testServo

void R3D3::headPosition(int position){
      headServo.write(position);
}//headPosition


byte R3D3::getPulsador(byte reference){
   return (digitalRead(reference));
}


void R3D3::beep(byte duration, byte times){
  for (int i=0; i<times; i++){
    digitalWrite(zumb, 255);
    delay(duration);
    digitalWrite(zumb, 0);
    delay(duration);
  }
}
```

----------Libreria .h----------

```
/*
  R3D3  Library.
  Created by J3Viton [Kiddybot], November, 2017.
*/
#ifndef R3D3_h
#define R3D3_h

#include "Arduino.h"
#include "Servo.h"



#define Motor_A   1
#define MotorB_IA 6    // analog OUT
#define MotorB_IB 7
#define Motor_B   2
#define MotorA_IB 4
#define MotorA_IA 5   // analog OUT

#define HeadCenter 90
#define HeadRight   150
#define HeadLeft  30

#define pulsador_A 3  // pulsador
#define zumb  9


#define ON 1
#define OFF 0


class R3D3
{
  public:
    R3D3(int pin);          // Constructor
    void R3D3::Init(void);  // Inicializo todo

    void test_bridgeH(int motor, int speed, int rotation);
    void moveForward(int speed);
    void moveBack(int speed);
    void turnRight(int angle, int speed);
    void turnLeft(int angle, int speed);
    void stop();

    void testServo();
    void headPosition(int position);

    byte getPulsador(byte reference);  //Devuelve TRUE si pulsado
    void beep(byte duration, byte times);
```

```
  private:
    int _pin;
  //headServo.attach(12);


};

#endif
```