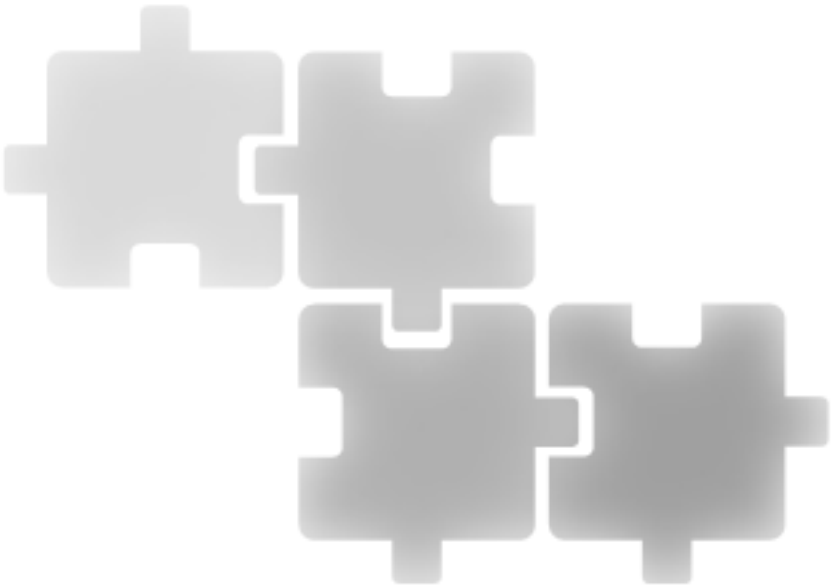
Proyecto final de Tecnología

Sistema Robótico Antropomórfico

ÍNDICE TEMÁTICO



BASES DEL PROYECTO FINAL DE TECNOLOGÍA ARDUINO - ROBOT

Se requiere el desarrollo e implementación de un **sistema robótico antropomórfico** multifuncional y de propósito general.

La finalidad del mismo sería la ejecución de diferentes funciones programadas tales como desplazamiento multidireccional por motores, simulación de lenguaje mediante emisión de señales acústicas y visuales según patrones diversos, movimiento de brazo mecánico, emisión de melodías musicales, etc. siendo posible su control total mediante mando infrarrojo y asignando cada una de las funciones a una tecla.

ELEMENTOS MATERIALES EMPLEADOS EN LA PRÁCTICA

Componente	Cantidad
Placa Arduino MEGA	1
Placa protoboard para conexiones	1
Pantalla LCD	1
Leds	12
Buzzer - KY 006	1
Motor DC (ruedas motrices delanteras)	2
Servo motor	1
Kit resistencias	1
Kit cables conectores	1
Estructura de mecano, poleas y cartón	1
Spray de pintura (verde y negro)	2
Pila - AA	8
Receptor infrarrojos	1
Mando de control remoto	1
Rueda de apoyo trasera (no motriz)	1
Interruptor	1

ENTORNO DE DESARROLLO (IDE) EMPLEADO PARA EL PROYECTO

Para el desarrollo e implementación del software en el sistema Arduino se ha utilizado el software *Arduino Integrated Development Environment*, también denominado *Arduino Software (IDE),* en su versión 1.8.5. para Microsoft Windows®.

Arduino Software (IDE) se trata de un proyecto abierto desarrollado, depurado y soportado por *ARDUINO.CC.*

Una vez conectado el sistema ARDUINO con el PC mediante el cable de datos tipo USB **A-B**, el propio entorno de desarrollo se encarga de compilar el código C del programa y de subirlo seguidamente a la memoria de la placa ARDUINO para su ejecución.

El programa de control del sistema robótico antropomórfico ha sido diseñado de forma modular, es decir, se han creado y depurado los códigos de cada uno de los sistemas (buzzer, leds, servor, etc) y funciones por separado, ensamblándose finalmente todos los módulos para su compilado conjunto.

NOTAS SOBRE EL CÓDIGO Y PROBLEMAS OBSERVADOS

Para el desarrollo del código utilizado en el sistema robótico se han utilizado 3 librerías públicas disponibles en Internet y que permiten el uso de los distintos dispositivos del sistema. Estas librerías son:

```
#include <Wire.h>                // LIBRERÍA PARA CONTROL DE MANDO INFRARROJO
#include <LiquidCrystal_I2C.h>   // LIBRERÍA PARA CONTROL DEL DISPLAY LCD
#include <Servo.h>               // LIBRERÍA PARA CONTROL DEL SERVOMOTOR
```

En las funciones que ejecutan un giro del robot, se ha optado por implementar una función que active ambos motores en sentido opuesto para conseguir mejorar el movimiento del robot, no obstante, se observa cierta falta de potencia en los motores DC que impide un giro efectivo del mismo sobre su propio eje. Este problema tal vez podría resolverse si se sustituyera la rueda trasera por una rueda multidireccional de tipo bola y aumentando la potencia de alimentación de los motores DC.

La librería estándar para el uso del mando de control remoto *(Irremote.h)* que se encuentra disponible en los *repositorios públicos de código y librerías de ARDUINO* y que fue utilizada al inicio del proyecto, fue descartada finalmente por presentar problemas de compilado del código al usarla simultáneamente con las otras librerías.

Dada la disposición enfrentada de los motores DC cuyas poleas arrastran a las ruedas motrices del robot, es posible que debamos combinar el uso de las funciones `MoveBackward y MoveForward` hasta obtener el resultado y giro deseado de los motores.

Todo el código implementado en el sistema robótico antropomórfico ha sido diseñado y/o adaptado expresamente para el presente proyecto, siendo documentado en su totalidad al objeto de facilitar posibles ampliaciones posteriores y/o mejoras por parte de cualquier usuario, pudiendo considerarse por ello como un **proyecto de código abierto** también conocido como *OPENSOURCE*.

TRASCRIPCIÓN LITERAL DEL **CÓDIGO C** DEL PROYECTO

```
// *************** PRÁCTICA DE TECNOLOGÍA ARDUINO  -  3a EVALUACIÓN ***************************
// *************** PROYECTO FIN DE CURSO - ROBOT ARDUINO    2º BACH ***************************
// *************** Autores: ROBIN DYSELINCK  &  RAFAEL LOMEÑA BUSTA ***************************
// *************** IES GAIA - 06/05/2018 ******************************************************

// LIBRERÍAS INCLUIDAS EN EL PROGRAMA ##########################################################
#include <Wire.h>                 // LIBRERÍA PARA CONTROL DE MANDO INFRARROJO
#include <LiquidCrystal_I2C.h>    // LIBRERÍA PARA CONTROL DEL DISPLAY LCD
#include <Servo.h>                // LIBRERÍA PARA CONTROL DEL SERVOMOTOR

// DECLARACIÓN VARIABLES CONTROL MANDO INFRARROJO ----------------------------------------------
// Al presentar problemas de compatibilidad las librerías disponibles para control del mando
// infrarrojo se opta por implementar el código contenido en la FUNCIÓN lecturaCodigoIr()
long duracion[32];  // Array que contiene la duración de cada pulso en pico-segundos (uS)
int x=0;            // Contador para moverse por las distintas variables de los arrays
byte bits[32];      // Array de bits tras la conversión de tiempos a bits.
int pulso_comienzo; // En esta variable se almacena el valor del pulso de inicio de 4,5mS
int codigo_tecla=0; // Valor de la tecla pulsada convertido de binario a decimal
const int IR=13;    // Pin al que va conectado nuestro receptor de infrarrojos

// DECLARACIÓN VARIABLES MOTOR SERVO (BRAZO) ----------------------------------------------------
Servo myservo;      // crea el objeto myservo para rutina servo()
int pos = 0;        // posicion inicial del servo

// DECLARACIÓN VARIABLE PARA CONTROL DEL ZUMBADOR -----------------------------------------------
const int pinBuzzer = 18; // Pin al que se conecta el zumbador

// DECLARACIÓN VARIABLES MOTORES DC -------------------------------------------------------------
const int pinIN1 = 47;    // PINES MOTORES DC
const int pinIN2 = 49;    // ID.
const int pinIN3 = 51;    // ID.
const int pinIN4 = 53;    // ID.
int waitTime=2000;        // ESTABLECE VALOR DE TIEMPO PARA ACCIONAMIENTO DE MOTORES DC's
const int pinMotorA[2] = {pinIN1, pinIN2 }; // MATRIZ DE VALORES PINES MOTOR DC-A (HIGH/LOW)
const int pinMotorB[2] = {pinIN3, pinIN4 }; // MATRIZ DE VALORES PINES MOTOR DC-B (HIGH/LOW)

// DECLARACIÓN VARIABLES PARA CONTROL PANEL DE LEDs  --------------------------------------------
const int led_1 = 31;
const int led_2 = 32;
const int led_3 = 33;
const int led_4 = 34;
const int led_5 = 35;
const int led_6 = 36;
const int led_7 = 37;
const int led_8 = 38;
const int led_9 = 39;
const int led_10 = 40;
const int led_11 = 41;
const int led_12 = 42;

// DECLARACIÓN VARIABLES PARA FUNCIONES VARIAS ---------------------------------------------------
int rand1;                // VARIABLE ALEATORIA USADA PARA SIMULAR PARPADEO DE OJOS EN loop()
int rand2;                // VARIABLE ALEATORIA USADA PARA SIMULAR LENGUA ROBOT CODIFICADA
int tm=40;                // TIEMPO DE ENCENDIDO DEL LED PARA EFECTO CIRCULAR ELED_1()

// PINES DE CONTROL DEL DISPLAY LCD
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);

// MATRIZ BINARIA PARA DEFINIR CARACTERES GRÁFICOS BOCA [5 x 8] EN LCD
    byte boca_1[8] = {
      0b00010000,
```

```
    0b00001100,
    0b00000111,
    0b00000001,
    0b00000000,
    0b00000000,
    0b00000000,
    0b00000000,
  };
  byte boca_2[8] = {
    0b00000000,
    0b00000000,
    0b00000000,
    0b00011100,
    0b00011111,
    0b00001111,
    0b00000011,
    0b00000000,
  };
  byte boca_3[8] = {
    0b00000000,
    0b00000000,
    0b00000000,
    0b00000000,
    0b00011000,
    0b00011111,
    0b00011111,
    0b00000111,
  };
  byte boca_4[8] = {
    0b00000000,
    0b00000000,
    0b00000000,
    0b00000000,
    0b00000000,
    0b00011111,
    0b00011111,
    0b00011111,
  };
  byte boca_5[8] = {
    0b00000000,
    0b00000000,
    0b00000000,
    0b00000000,
    0b00000000,
    0b00011111,
    0b00011111,
    0b00011111,
  };
  byte boca_6[8] = {
    0b00000000,
    0b00000000,
    0b00000000,
    0b00000000,
    0b00000011,
    0b00011111,
    0b00011111,
    0b00011100,
  };
  byte boca_7[8] = {
    0b00000000,
    0b00000000,
    0b00000000,
    0b00000111,
```

```
      0b00011111,
      0b00011110,
      0b00011000,
      0b00000000,
    };
    byte boca_8[8] = {
      0b00000001,
      0b00000110,
      0b00011100,
      0b00010000,
      0b00000000,
      0b00000000,
      0b00000000,
      0b00000000,
    };

// ********************************************************************************
// ******************** INICIO RUTINA setup ***********************************
// ********************************************************************************
void setup()
{
  myservo.attach(12);        // ESTABLECE PIN DEL SERVOMOTOR PARA BRAZO MECÁNICO
  pinMode(pinBuzzer,OUTPUT); // BUZZER
  pinMode(pinIN1, OUTPUT);   // MOTOR DC A
  pinMode(pinIN2, OUTPUT);   // MOTOR DC A
  pinMode(pinIN3, OUTPUT);   // MOTOR DC B
  pinMode(pinIN4, OUTPUT);   // MOTOR DC B
  pinMode(led_1, OUTPUT);    // PINES LEDs 1-12
  pinMode(led_2, OUTPUT);
  pinMode(led_3, OUTPUT);
  pinMode(led_4, OUTPUT);
  pinMode(led_5, OUTPUT);
  pinMode(led_6, OUTPUT);
  pinMode(led_7, OUTPUT);
  pinMode(led_8, OUTPUT);
  pinMode(led_9, OUTPUT);
  pinMode(led_10, OUTPUT);
  pinMode(led_11, OUTPUT);
  pinMode(led_12, OUTPUT);
// REINICIA LCD DE 16 CARACTERES x 2 LÍNEAS
  lcd.begin(16, 2);
// CREA CARACTERES GRÁFICOS PERSONALIZADOS DE LA BOCA
  lcd.createChar (2, boca_1);
  lcd.createChar (3, boca_2);
  lcd.createChar (4, boca_3);
  lcd.createChar (5, boca_4);
  lcd.createChar (6, boca_5);
  lcd.createChar (7, boca_6);
  lcd.createChar (8, boca_7);
  lcd.createChar (9, boca_8);
// SE LIMPIA EL DISPLAY
  lcd.clear();

// CONFIGURACIÓN DE LA CARA MEDIANTE EL USO DE CARACTERES SEMIGRÁFICOS
  // SE USA CARACTERES ASCII PARA OJOS Y NARIZ EN LÍNEA SUPERIOR
  // CARACTERES ASCII PARA DEFINIR DIFERENTES ESTADOS OJOS
    // ASCII 111 = o .. o  >>>>>>>>>>>> TRANSICIÓN OJOS ABIERTOS/CERRADOS
    // ASCII 79  = O .. O  >>>>>>>>>>>> OJOS ABIERTOS
    // ASCII 95  = _ .. _  >>>>>>>>>>>> DORMIDO/OJOS CERRADOS
    // ASCII 94  = ^ .. ^  >>>>>>>>>>>> FELIZ
    // ASCII 126 = ~ .. ~  >>>>>>>>>>>> DISTRAÍDO

      // POSICION/ESTADO INICIAL DE LOS OJOS Y NARIZ
```

```
      lcd.setCursor(5,0 );
      lcd.print("O .. O");

      // SE ESCRIBE CARACTERES GRÁFICOS DE LA BOCA EN LÍNEA INFERIOR DISPLAY
      lcd.setCursor(4,1 );
      lcd.write((byte)2);
      lcd.setCursor(5,1 );
      lcd.write((byte)3);
      lcd.setCursor(6,1 );
      lcd.write((byte)4);
      lcd.setCursor(7,1 );
      lcd.write((byte)5);
      lcd.setCursor(8,1 );
      lcd.write((byte)6);
      lcd.setCursor(9,1 );
      lcd.write((byte)7);
      lcd.setCursor(10,1 );
      lcd.write((byte)8);
      lcd.setCursor(11,1 );
      lcd.write((byte)9);
}
// ********************* FIN RUTINA setup ************************************

// ******************************************************************************
// ********************* INICIO RUTINA PRINCIPAL loop ************************
// ******************************************************************************
void loop (){
  rand1 = random(1,30000);  // VARIABLE DE VALOR ALEATORIO DE 1 A 30 MIL QUE SIMULA PARPADEO
  ELED_ALL ();              // LLAMADA A FUNCIÓN QUE ENCIENDE PANEL DE LEDs
  if (rand1<=3) {           // PARPADEA SI rand1 ES INFERIOR A 4
       Parpadea (1);       // LLAMADA A FUNCIÓN QUE PARPADEA (1) VEZ
      }
  if(digitalRead(IR)==0)    // ESPERA QUE SE PULSE TECLA EN MANDO INFRARROJO
    {
    // LLAMA A FUNCIÓN lecturaCodigoIr QUE LEE TECLA PULSADA Y GUARDA VALOR EN codigo_tecla
    lecturaCodigoIr();

    // ACCIÓN A REALIZAR EN FUNCIÓN DE LA TECLA PULSADA (VARIABLE codigo_tecla)
    switch(codigo_tecla)
      {
      case 41:       // TECLA 1
            HABLA(); // LLAMA A FUNCIÓN QUE SIMULA LENGUA ROBOT CODIFICADA
            break;
      case 25:       // TECLA 2
            ELED_1 (16,300);  // LLAMA A FUNCIÓN EFECTO CIRCULAR LED pasando Num.veces,velocidad
            // ESTOS VALORES PASADOS INDICAN NUMERO DE VUELTAS Y TIEMPO DE LUZ EN ms
            Parpadea (2);  // LLAMADA A FUNCIÓN QUE PARPADEA (2) VECES
            break;
      case 57:      // TECLA 3
          // libre
          break;
      case 9:       // TECLA 4
          zumba_1();          // LLAMA A FUNCIÓN MÚSICA
          break;
      case 1:       // TECLA 5
          Guino_izqu ();      // LLAMADA A FUNCIÓN GUIÑO OJO IZQUIERDO
          break;
      case 49:      // TECLA 6
          Guino_dcho ();      // LLAMADA A FUNCIÓN GUIÑO OJO DERECHO
          break;
      case 56:      // TECLA 7
          servo();            // LLAMADA A FUNCIÓN servo QUE ACCIONA BRAZO ROBOT
          break;
```

```
    case 42:      // TECLA 8    // EFECTO DECODIFICA CLAVE DE ACCESO CON BUCLES ANIDADOS
        ELEDOFF_ALL();      // APAGA LEDs
        delay (1000);
        Despiste ();        // CAMBIA OJOS A ESTADO DE PROCESAMIENTO MASIVO
        // SE LANZAN 3 BUCLES ANIDADOS PARA SIMULAR EFECTO BÚSQUEDA CLAVE SECRETA
        for (int d=100;d>=20;d=d-20){
          for (int n2=1;n2<=5;n2++){
            for (int n=1;n<=7*n2;n++){
              ELED_ALE();     // LLAMADA A FUNCIÓN QUE ILUMINA LED AL AZAR Y EMITE SONIDO
              delay (d/3);    // VELOCIDAD DEL EFECTO EN AUMENTO PROGRESIVO
              ELEDOFF_ALL();  // APAGA LEDs
            }
          }
        }
         // SIMULA LOCALIZACION DE CLAVE DE ACCESO POSITIVA
        for (int n=1;n<=6;n++){
              lcd.setCursor(0,0 );    // FIJA POSICIÓN LCD EN ESQUINA SUPERIOR IZQUIERDA
              lcd.print("  Open System ! "); // SE MUESTRA MENSAJE DE ADVERTENCIA
              tone(pinBuzzer, 1000);  // SE EMITE SEÑAL ACÚSTICA DE ATENCIÓN
              delay (600);
              lcd.setCursor(0,0 );  // FIJA POSICIÓN LCD EN ESQUINA SUPERIOR IZQUIERDA
              lcd.print(" PASSWORD FOUND!"); // SE MUESTRA MENSAJE DE ADVERTENCIA
              tone(pinBuzzer, 500);   // SE EMITE SEÑAL ACÚSTICA DE ATENCIÓN
              delay (300);
              }
        lcd.setCursor(0,0 );           // FIJA POSICIÓN LCD EN ESQUINA SUPERIOR IZQUIERDA
        lcd.print("               "); // SE BORRA LA LÍNEA SUPERIOR DEL LCD CON ESPACIOS
        noTone(pinBuzzer);             // SE SILENCIA EL ZUMBADOR
        break;
    case 36:      // TECLA 9
        // libre (SIN FUNCIÓN ASIGNADA)
        break;
    case 26:      // TECLA *
        // libre (SIN FUNCIÓN ASIGNADA)
        break;
    case 38:      // TECLA 0
       // libre (SIN FUNCIÓN ASIGNADA)
        break;
    case 44:      // TECLA #
        break;
        case 6:   // TECLA ^ ARRIBA - AVANZA ROBOT
        MoveBackwardA(pinMotorA);     // ACTIVA MOTOR DC-A LLAMANDO A FUNCIÓN MoveBackwardA
        MoveBackwardB(pinMotorB);     // ACTIVA MOTOR DC-B LLAMANDO A FUNCIÓN MoveBackwardB
        delay(waitTime);              // TIEMPO DE ACCIONAMIENTO DE LOS MOTORES
        fullStopA(pinMotorA);         // DETIENE MOTOR DC-A LLAMANDO A FUNCIÓN fullStopA
        fullStopB(pinMotorB);         // DETIENE MOTOR DC-B LLAMANDO A FUNCIÓN fullStopB
        break;
    case 4:       // TECLA < IZDA     // GIRO DEL ROBOT A LA IZDA.
        MoveForwardA(pinMotorA);      // ACCIONA MOTOR DC-A
        MoveBackwardB(pinMotorB);     // ACCIONA MOTOR DC-B (EN SENTIDO CONTRARIO)
        delay(waitTime);              // TIEMPO DEL ACCIONAMIENTO DE LOS MOTORES
        fullStopA(pinMotorA);         // DETIENE MOTOR DC-A
        fullStopB(pinMotorB);         // DETIENE MOTOR DC-B
        break;
    case 14:      // TELA OK
        // libre (SIN FUNCIÓN ASIGNADA)
        break;
    case 23:      // TECLA > DCHA     // GIRO DEL ROBOT A LA DCHA.
        MoveBackwardA(pinMotorA);     // ACCIONA MOTOR DC-A
        MoveForwardB(pinMotorB);      // ACCIONA MOTOR DC-B (EN SENTIDO CONTRARIO)
        delay(waitTime);              // TIEMPO DE ACCIONAMIENTO DE LOS MOTORES
        fullStopA(pinMotorA);         // DETIENE MOTOR DC-A
        fullStopB(pinMotorB);         // DETIENE MOTOR DC-B
```

```
          break;
      case 19:      // TECLA V ABAJO - RETROCEDE ROBOT
          MoveForwardA(pinMotorA);      // ACTIVA MOTOR DC-A LLAMANDO A FUNCIÓN MoveForwardA
          MoveForwardB(pinMotorB);      // ACTIVA MOTOR DC-B LLAMANDO A FUNCIÓN MoveForwardB
          delay(waitTime);              // TIEMPO DE ACCIONAMIENTO DE LOS MOTORES
          fullStopA(pinMotorA);         // DETIENE MOTOR DC-A
          fullStopB(pinMotorB);         // DETIENE MOTOR DC-B
          break;
      }   // cierra switch ()
    }     // cierra condicional if
}         // cierra FUNCIÓN loop()
// ********************** FIN RUTINA PRINCIPAL loop ***************************************

// ************************************************************************************************
// ********************** INICIO RUTINAS/FUNCIONES ************************************************
// ************************************************************************************************

// RUTINA DE SIMULACIÓN DE HABLA EL LENGUA ROBOT CODIFICADA--------------------------------
void HABLA()
{
   Despiste();      // LLAMA A FUNCIÓN QUE CAMBIA OJOS A ESTADO DE PROCESAMIENTO MASIVO
   for (int n=1;n<30;n++)
       {
       ELED_ALE();
       rand2= random(1000,3000);
       tone(pinBuzzer, rand2);
       delay(.5*200);
       noTone(pinBuzzer);
       ELEDOFF_ALL ();
       }
   Guino_izqu ();
   Guino_dcho ();
   Feliz();
   delay(500);
   Parpadea(3);
}

// RUTINA DE CONTROL DEL BRAZO ------------------------------------------------------------
void servo()
{
   // VARÍA LA POSICIÓN DEL SERVO DE 0 A 90 Y DE 90 A 0 CON ESPERAS DE 15 ms
   for (pos = 0; pos <= 90; pos += 1)
   {
       myservo.write(pos);
       delay(15);
   }
   for (pos = 90; pos >= 0; pos -= 1)
   {
      myservo.write(pos);
      delay(15);
   }
}

// RUTINA DE MÚSICA STARWARS --------------------------------------------------------------
void zumba_1()
{
 tone(pinBuzzer, 392.00);               // TONO DE LA SEÑAL SONORA
 delay(.5*1000);                        // DURACIÓN
  tone(pinBuzzer, 523.25);
  delay(.5*2000);
   tone(pinBuzzer, 587.33);
   delay(.5*1500);
    tone(pinBuzzer, 622.25);
```

```
    delay(.5*250);
     tone(pinBuzzer, 698.46);
     delay(.5*250);
      tone(pinBuzzer, 622.25);
      delay(.5*2000);
       tone(pinBuzzer, 392.00);
       delay(.5*1500);
        tone(pinBuzzer, 392.00);
        delay(.5*500);
         tone(pinBuzzer, 523.25);
         delay(.5*1500);
          tone(pinBuzzer, 587.33);
          delay(.5*500);
           tone(pinBuzzer, 622.25);
           delay(.5*500);
            tone(pinBuzzer, 392);
            delay(.5*450);
             tone(pinBuzzer, 622.25);
             delay(.5*400);
              tone(pinBuzzer, 523.25);
              delay(.5*350);
               tone(pinBuzzer, 783.99);
               delay(.5*400);
                tone(pinBuzzer, 698.46);
                delay(.5*3000);
                 tone(pinBuzzer, 392);
                 delay(.5*1000);
                  tone(pinBuzzer, 523.25);
                  delay(.5*1500);
                   tone(pinBuzzer, 261.63);
                    tone(pinBuzzer, 261.63);
                    noTone(pinBuzzer);         // APAGA EL ZUMBADOR
}

// RUTINA PARA MOVER HACIA ADELANTE MOTOR DC A ------------------------------------------
  void MoveForwardA(int pinMotorA[2])
    {
    digitalWrite(pinMotorA[0], HIGH);
    digitalWrite(pinMotorA[1], LOW);
    }

// RUTINA PARA MOVER HACIA ADELANTE MOTOR DC B ------------------------------------------
  void MoveForwardB(int pinMotorB[2])
    {
    digitalWrite(pinMotorB[0], LOW);
    digitalWrite(pinMotorB[1], HIGH);
    }

// RUTINA PARA MOVER HACIA ATRÁS MOTOR DC A ------------------------------------------
  void MoveBackwardA(int pinMotorA[2])
    {
    digitalWrite(pinMotorA[0], LOW);
    digitalWrite(pinMotorA[1], HIGH);
    }

// RUTINA PARA MOVER HACIA ATRÁS MOTOR DC B ------------------------------------------
  void MoveBackwardB(int pinMotorB[2])
    {
    digitalWrite(pinMotorB[0], HIGH);
    digitalWrite(pinMotorB[1], LOW);
    }
// RUTINA QUE DETIENE MOTOR DC A ------------------------------------------------------
  void fullStopA(int pinMotorA[2])
```

```
    {
    digitalWrite(pinMotorA[0], LOW);
    digitalWrite(pinMotorA[1], LOW);
    }
// RUTINA QUE DETIENE MOTOR DC B ------------------------------------------------------------
  void fullStopB(int pinMotorB[2])
    {
    digitalWrite(pinMotorB[0], LOW);
    digitalWrite(pinMotorB[1], LOW);
    }
// RUTINA QUE ENCIENDE TODOS LOS LEDS -------------------------------------------------------
  void ELED_ALL ()
    {
    digitalWrite(led_1, 1);digitalWrite(led_2, 1);digitalWrite(led_3, 1);
    digitalWrite(led_4, 1);digitalWrite(led_5, 1);digitalWrite(led_6, 1);
    digitalWrite(led_7, 1);digitalWrite(led_8, 1);digitalWrite(led_9, 1);
    digitalWrite(led_10, 1);digitalWrite(led_11, 1);digitalWrite(led_12, 1);
    }
// RUTINA QUE ENCIENDE LEDS DE FORMA ALEATORIA CON SONIDOS ASOCIADOS ----------------------
  void ELED_ALE()
  {
    int rand2=random(1,12);   // SE USA FUNCIÓN random PARA VARIABLE rand2 (DE 1 A 12)
    switch(rand2)             // SE ENCIENDE UN LED DIFERENTE EN FUNCIÓN DEL VALOR DE rand2
      {
      case 1:
      digitalWrite(led_1, 1);
      tone(pinBuzzer, 2500);
      delay(20);
      noTone(pinBuzzer);
      break;
      case 2:
      digitalWrite(led_2, 1);
      break;
      case 3:
      digitalWrite(led_3, 1);
      tone(pinBuzzer, 1300);
      delay(20);
      noTone(pinBuzzer);
      break;
      case 4:
      digitalWrite(led_4, 1);
      break;
      case 5:
      digitalWrite(led_5, 1);
      tone(pinBuzzer, 1900);
      delay(20);
      noTone(pinBuzzer);
      break;
      case 6:
      digitalWrite(led_6, 1);
      break;
      case 7:
      digitalWrite(led_7, 1);
      break;
      case 8:
      digitalWrite(led_8, 1);
      tone(pinBuzzer, 1000);
      delay(20);
      noTone(pinBuzzer);
      break;
      case 9:
      digitalWrite(led_9, 1);
      tone(pinBuzzer, 700);
```

```
      delay(20);
      noTone(pinBuzzer);
      break;
      case 10:
      digitalWrite(led_10, 1);
      break;
      case 11:
      digitalWrite(led_11, 1);
      tone(pinBuzzer, 2000);
      delay(20);
      noTone(pinBuzzer);
      break;
      case 12:
      digitalWrite(led_12, 1);
      break;
      }
  }

// RUTINA QUE ENCIENDE TODOS LOS LEDS EXCEPTO LOS BLANCOS ---------------------------------
  void ELED_MED ()
  {
    digitalWrite(led_1, 1);digitalWrite(led_2, 0);digitalWrite(led_3, 1);
    digitalWrite(led_4, 1);digitalWrite(led_5, 0);digitalWrite(led_6, 1);
    digitalWrite(led_7, 1);digitalWrite(led_8, 0);digitalWrite(led_9, 1);
    digitalWrite(led_10, 1);digitalWrite(led_11, 0);digitalWrite(led_12, 1);
  }

// RUTINA QUE APAGA TODOS LOS LEDS ---------------------------------------------------------
void ELEDOFF_ALL ()
  {
     digitalWrite(led_1, 0); digitalWrite(led_2, 0); digitalWrite(led_3, 0);
     digitalWrite(led_4, 0); digitalWrite(led_5, 0); digitalWrite(led_6, 0);
     digitalWrite(led_7, 0); digitalWrite(led_8, 0); digitalWrite(led_9, 0);
     digitalWrite(led_10, 0); digitalWrite(led_11, 0); digitalWrite(led_12, 0);
  }

// RUTINA QUE PRODUCE EFECTO CIRCULAR EN LOS LEDS ------------------------------------------
  void ELED_1 (int veces,int tm)      // PRIMER VALOR= NUM.VUELTAS  //SEGUNDO= TIEMPO
  {
       for (int n=0; n<veces; n++){
        tm=tm/1.5;                    // REDUCE TIEMPO EN CADA VUELTA DE BUCLE
       digitalWrite(led_1, 255);
        delay(tm);
       digitalWrite(led_1, 0);
        delay(tm);
       digitalWrite(led_2, 255);
        delay(tm);
       digitalWrite(led_2, 0);
        delay(tm);
       digitalWrite(led_3, 255);
        delay(tm);
       digitalWrite(led_3, 0);
        delay(tm);
          lcd.setCursor(5,0 );       // SE INTERCALA UN PARPADEO DE OJOS
          lcd.print("_ .. _");
       digitalWrite(led_4, 255);
        delay(tm);
       digitalWrite(led_4, 0);
        delay(tm);
       digitalWrite(led_5, 255);
        delay(tm);
       digitalWrite(led_5, 0);
        delay(tm);
```

```
          lcd.setCursor(5,0 );      // SE RECUPERA ESTADO NORMAL DE OJOS
          lcd.print("O .. O");
        digitalWrite(led_6, 255);
         delay(tm);
        digitalWrite(led_6, 0);
         delay(tm);
        digitalWrite(led_7, 255);
         delay(tm);
        digitalWrite(led_7, 0);
         delay(tm);
        digitalWrite(led_8, 255);
         delay(tm);
        digitalWrite(led_8, 0);
         delay(tm);
        digitalWrite(led_9, 255);
         delay(tm);
        digitalWrite(led_9, 0);
         delay(tm);
        digitalWrite(led_10, 255);
         delay(tm);
        digitalWrite(led_10, 0);
         delay(tm);
        digitalWrite(led_11, 255);
         delay(tm);
        digitalWrite(led_11, 0);
         delay(tm);
        digitalWrite(led_12, 255);
         delay(tm);
        digitalWrite(led_12, 0);
         delay(tm);
        }
  }

// RUTINA QUE PRODUCE GUIÑO OJO IZQUIERDO ------------------------------------------------
void Guino_izqu ()
  {
      lcd.setCursor(5,0 );  //  _ .. O
      lcd.print("o .. O");
      delay (100);
      lcd.setCursor(5,0 );
      lcd.print("_ .. O");
      delay (300);
      lcd.setCursor(5,0 );
      lcd.print("O .. O");
  }

// RUTINA QUE PRODUCE GUIÑO OJO DERECHO --------------------------------------------------
void Guino_dcho ()
  {
      lcd.setCursor(5,0 );  //  O .. _
      lcd.print("O .. o");
      delay (100);
      lcd.setCursor(5,0 );
      lcd.print("O .. _");
      delay (300);
      lcd.setCursor(5,0 );
      lcd.print("O .. O");
  }
// RUTINA QUE PRODUCE ESTADO FELICIDAD ---------------------------------------------------
void Feliz ()
  {
      lcd.setCursor(5,0 );  //  ^ .. ^
      lcd.print("^ .. ^");
```

```
      delay (200);
  }
// RUTINA QUE PRODUCE ESTADO DESPISTE ----------------------------------------------------
void Despiste ()
  {
      lcd.setCursor(5,0 );  //  ~ .. ~
      lcd.print("@ .. @");
      delay (400);
  }
// RUTINA QUE PRODUCE PARPADEO N VECES ---------------------------------------------------
void Parpadea(int veces)
  {
    for (int n=0; n<veces; n++){
      lcd.setCursor(5,0 );  // o    o
      lcd.print("o .. o");
      delay (150);
      lcd.setCursor(5,0 );  // _    _
      lcd.print("_ .. _");
      delay (200);
      lcd.setCursor(5,0 );  // o    o
      lcd.print("o .. o");
      delay (150);
      lcd.setCursor(5,0 );  // O    O
      lcd.print("O .. O");
      delay (150);
    }
  }
// RUTINA QUE LEE QUÉ TECLA DEL MANDO A DISTANCIA SE HA PULSADO Y ALMACENA EL CÓDIGO
// EN LA VARIABLE codigo_tecla PARA SU INTERPRETACIÓN
void lecturaCodigoIr()
  {
    //PASO 1: DETECCIÓN DEL PULSO DE INICIO DE SECUENCIA (4,5mS)
    pulso_comienzo=pulseIn(IR, HIGH);
    // SE COMPRUEBA SI VALOR DE PULSO DE COMIENZO ES DE 4.5mS (INICIO DE SECUENCIA)
    if(pulso_comienzo>4000 && pulso_comienzo<5000)
    {
    //PASO 2: CRONOMETRA TIEMPOS DE CADA PULSO(uS) Y ALMACENA VALORES EN MATRIZ duracion[ ]
      for(x=1; x<=32; x++)
        {
        duracion[x]=pulseIn(IR, HIGH); // ALMACENA DURACIÓN DE CADA PULSO (EN uS) DE 1 A 32
        }
    //PASO 3: SEGUN TIEMPO DE CADA PULSO SE DETERMINA SI ES UN 0 ó UN 1 LÓGICO
      for(x=1; x<=32; x++)             // BUCLE DE 1 A 32 PARA RECORRER LA MATRIZ duracion[ ]
        {
        if(duracion[x]>500 && duracion[x]<700)   //SI EL PULSO DURA ENTRE 500 y 700uS
        {
        bits[x]=0;                               //... ES UN 0 LÓGICO
        }
        if(duracion[x]>1500 && duracion[x]<1750) //SI EL PULSO DURA ENTRE 1500 y 1750uS
        {
        bits[x]=1;                               //... ES UN 1 LÓGICO
        }
      }
    //PASO 4: CONVIERTE ARRAY BINARIO A VALOR DECIMAL Y ALMACENA EN codigo_tecla
    //Puesto que muchos de los bits se repiten en todas las teclas, omitimos dichos
    //bits y nos quedamos con los bits 17 al 21 y el 23 (6 bits). Suficientes para
    //distinguir todas las teclas:
    //bits:     17   18   19   20   21   23
    //         ----------------------------
    //valores: 0/1  0/1  0/1  0/1  0/1  0/1  < se convierte valor binario a decimal
    //Estos 6 bits los convertimos a decimal con el método de Potencias de 2. Y el
    //resultado lo almacenamos en la variable codigo_tecla para su interpretación.
    codigo_tecla=0;                     // RESETEAMOS LA ÚLTIMA TECLA PULSADA ANTES
```

```
    if(bits[17]==1)
       {
         codigo_tecla=codigo_tecla+32;   // SI EL BIT ES 1 SE SUMA 2^5 A codigo_tecla
       }
    if(bits[18]==1)
       {
         codigo_tecla=codigo_tecla+16;   // SI EL BIT ES 1 SE SUMA 2^4 A codigo_tecla
       }
    if(bits[19]==1)
       {
         codigo_tecla=codigo_tecla+8;    // SI EL BIT ES 1 SE SUMA 2^3 A codigo_tecla
       }
    if(bits[20]==1)
       {
         codigo_tecla=codigo_tecla+4;    // SI EL BIT ES 1 SE SUMA 2^2 A codigo_tecla
       }
    if(bits[21]==1)
       {
         codigo_tecla=codigo_tecla+2;    // SI EL BIT ES 1 SE SUMA 2^1 A codigo_tecla
       }
    if(bits[23]==1)
       {
         codigo_tecla=codigo_tecla+1;    // SI EL BIT ES 1 SE SUMA 2^0 A codigo_tecla
       }
    }             // CIERRA if QUE COMPRUEBA SI VALOR DE PULSO DE COMIENZO ES DE 4.5mS
    delay(180);  // RETARDO PARA EVITAR REBOTES Y NO LEA DOS PULSACIONES DE LA MISMA TECLA
  }
// FIN DE CÓDIGO DE RUTINAS/FUNCIONES ******************************************************
```