ROBOT

Juan Antonio Martínez Sánchez
Pablo Huertas Arroyo

```
// Infrarrojos
#define TIMEOUT_ECO 1000000 // <anchura máxima del pulso generado por el módulo en µs>
const int irPinDel = 2;     // Pin del sensor de infrarrojos delantero
const int irPinTra = 3;     // Pin del sensor de infrarrojos trasero
int irValueDel;             // Valor del sensor de infrarrojos delantero
int irValueTra;             // Valor del sensor de infrarrojos trasero

// Ultrasonidos
long duration;         // Duración del pulso de eco
int distance;          // Distancia medida en centímetros
const int trigPin = 4; // Pin de activación del sensor ultrasónico (TRIGGER)
const int echoPin = 5; // Pin de recepción de eco del sensor ultrasónico (ECHO)

// Puente h
const int pwmPin1 = 13; // E1 del puente H conectado al pin 13 del Arduino
const int dirPin1 = 12; // E2 del puente H conectado al pin 12 del Arduino
const int pwmPin2 = 11; // E3 del puente H conectado al pin 11 del Arduino
const int dirPin2 = 10; // E4 del puente H conectado al pin 10 del Arduino

// Variable para el temporizador de 3 segundos
const unsigned long initialDelay = 3000;

int vecesNegro = 0;

void setup()
{
   // Iniciar comunicación serial
   Serial.begin(9600);

   // INFRARROJOS
   pinMode(irPinDel, INPUT);
   pinMode(irPinTra, INPUT);

   // ULTRASONIDOS
   //  Inicializar los pines de TRIGGER y ECHO
   pinMode(trigPin, OUTPUT);
   pinMode(echoPin, INPUT);

   digitalWrite(trigPin, LOW);

   // Esperar 3 segundos antes de iniciar las acciones
   delay(initialDelay);

   pinMode(pwmPin1, OUTPUT);
   pinMode(pwmPin2, OUTPUT);
   pinMode(dirPin1, OUTPUT);
   pinMode(dirPin2, OUTPUT);

   //parar();
```

```
    haciaDelante();
    delay(4000);

}

void loop()
{
    // Actualizar el valor de infrarrojos
    irValueDel = digitalRead(irPinDel);
    irValueTra = digitalRead(irPinTra);

    Serial.println(irValueDel);

    vecesNegro = 0;

    // Si el robot detecta la línea negra por el sensor delantero, realizar maniobra para evitar salirse
    while (irValueDel == 0)
    {
        vecesNegro++;
        if (vecesNegro > 2)
        {
            haciaAtras();
            delay(500);

            // Elegir aleatoriamente la dirección de giro para no salirse del tatami
            if (random(2) == 0)
            {
                girarIzquierdaMarchaAtras();
            }
            else
            {
                girarDerechaMarchaAtras();
            }

            delay(1000);
        }

        irValueDel = digitalRead(irPinDel);
    }

    if (irValueDel == 1)
    { // Si detecta que estamos en lo blanco, avanzar hacia delante
        haciaDelante();
    }

    // Si el sensor trasero detecta la línea negra, realizar maniobra para escapar de la situación
    if (irValueTra == 0)
    {
        girarDerecha();
        delay(500);
        haciaDelante();
        delay(1000);
    }

    getDistancia();
```

```
    if(distance < 30)
    {

          haciaDelante();
          delay(100);
          getDistancia();

    }


}

void getDistancia()
{
    // Generar un pulso de 10 microsegundos en el pin TRIGGER
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    // Medir la duración del pulso de eco
    duration = pulseIn(echoPin, HIGH);

    // Calcular la distancia en centímetros
    distance = duration * 0.034 / 2;

    // Imprimir la distancia en el monitor serial
    Serial.print("Distancia: ");
    Serial.print(distance);
    Serial.println(" cm");
}

// Función para mover hacia atrás
void haciaAtras()
{
    digitalWrite(dirPin1, LOW);
    digitalWrite(pwmPin1, HIGH);

    digitalWrite(dirPin2, LOW);
    digitalWrite(pwmPin2, HIGH);
}

// Función para detener el movimiento
void parar()
{
    digitalWrite(dirPin1, LOW);
    digitalWrite(pwmPin1, LOW);

    digitalWrite(dirPin2, LOW);
    digitalWrite(pwmPin2, LOW);
}

// Funcion para avanzar hacia delante
```

```
void haciaDelante()
{
  digitalWrite(dirPin1, HIGH);
  digitalWrite(pwmPin1, LOW);

  digitalWrite(dirPin2, HIGH);
  digitalWrite(pwmPin2, LOW);
}

// Funcion para girar a la izquierda
void girarIzquierda()
{
  digitalWrite(dirPin1, HIGH);
  digitalWrite(pwmPin1, LOW);

  digitalWrite(dirPin2, HIGH);
  digitalWrite(pwmPin2, HIGH);
}

// Funcion para girar a la derecha
void girarDerecha()
{
  digitalWrite(dirPin1, HIGH);
  digitalWrite(pwmPin1, HIGH);

  digitalWrite(dirPin2, HIGH);
  digitalWrite(pwmPin2, LOW);
}

// Función para girar a la derecha marcha atrás
void girarDerechaMarchaAtras()
{
  digitalWrite(dirPin1, LOW);
  digitalWrite(pwmPin1, HIGH);

  digitalWrite(dirPin2, HIGH);
  digitalWrite(pwmPin2, LOW);
}

// Función para girar a la izquierda marcha atrás
void girarIzquierdaMarchaAtras()
{
  digitalWrite(dirPin1, HIGH);
  digitalWrite(pwmPin1, LOW);

  digitalWrite(dirPin2, LOW);
  digitalWrite(pwmPin2, HIGH);
}
```