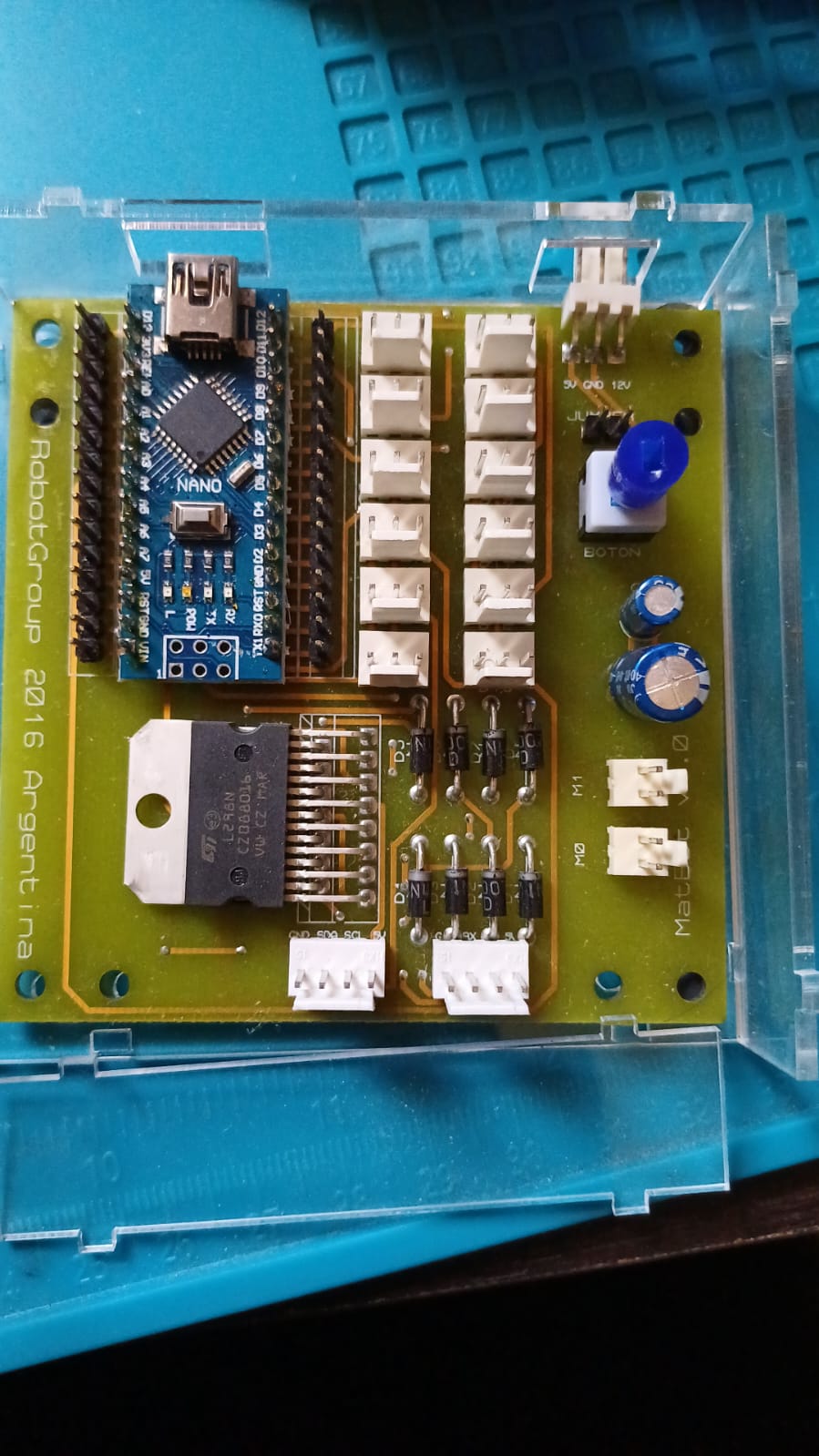
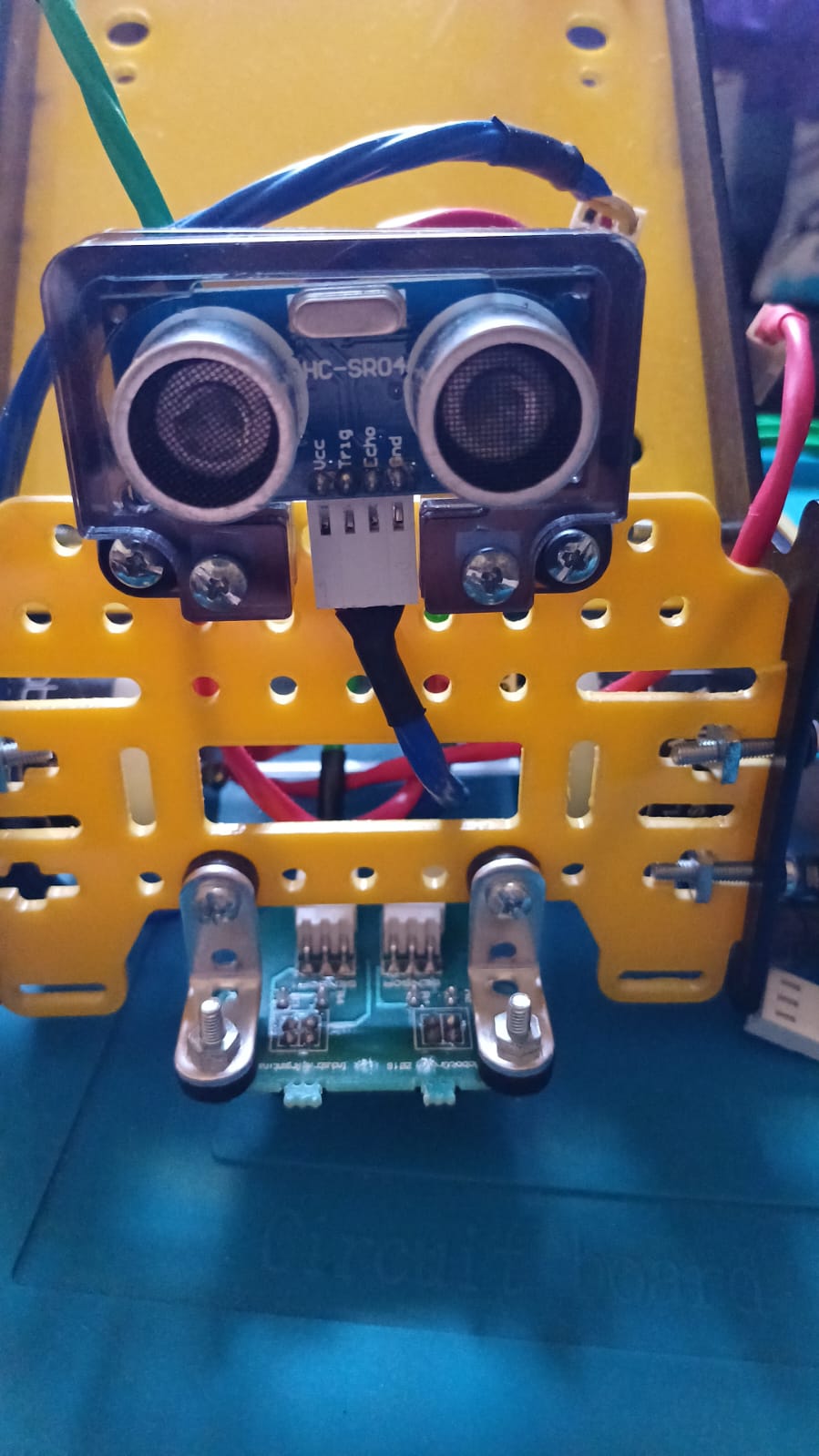
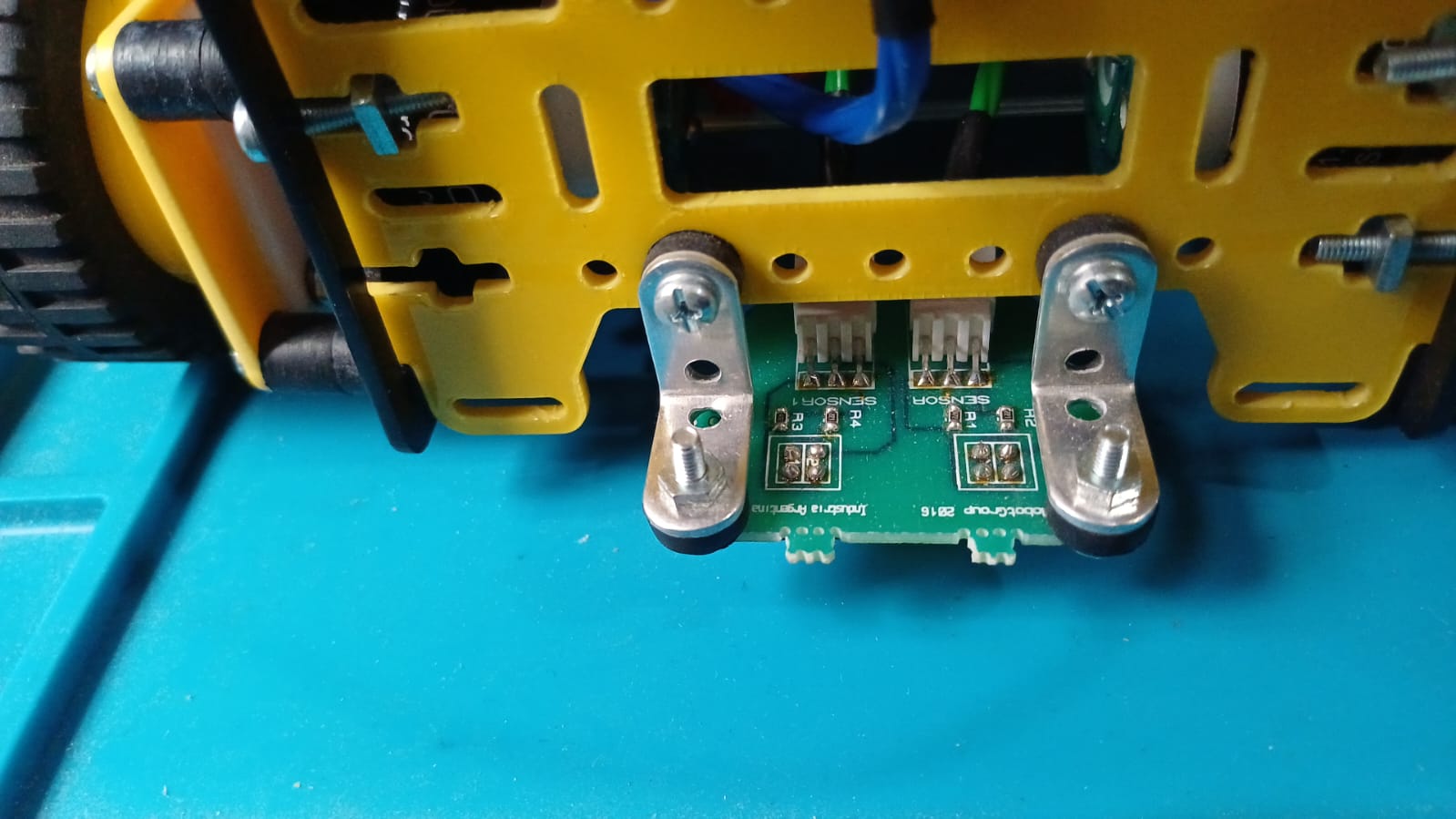
# **Robot MatBot 1.0**

Esta guía es nuestro manual de referencia para el robot MatBot 1.0. Aquí aprenderemos a identificar cada uno de sus componentes, entender cómo están conectados y, lo más importante, programarlos para dar vida a nuestros proyectos.

## **1. Análisis de Componentes**

Cada robot MatBot se compone de varias partes clave que trabajan juntas.

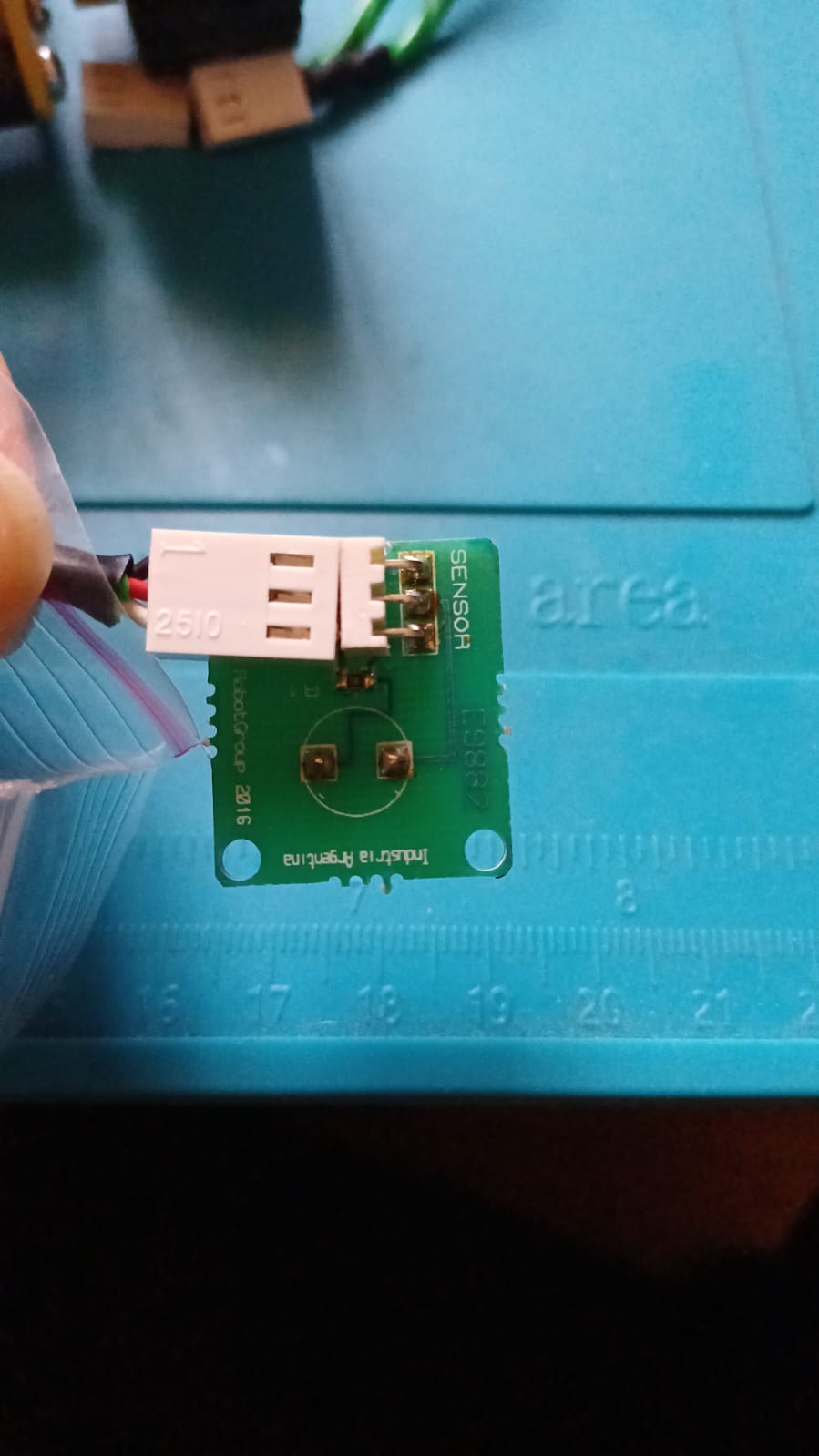
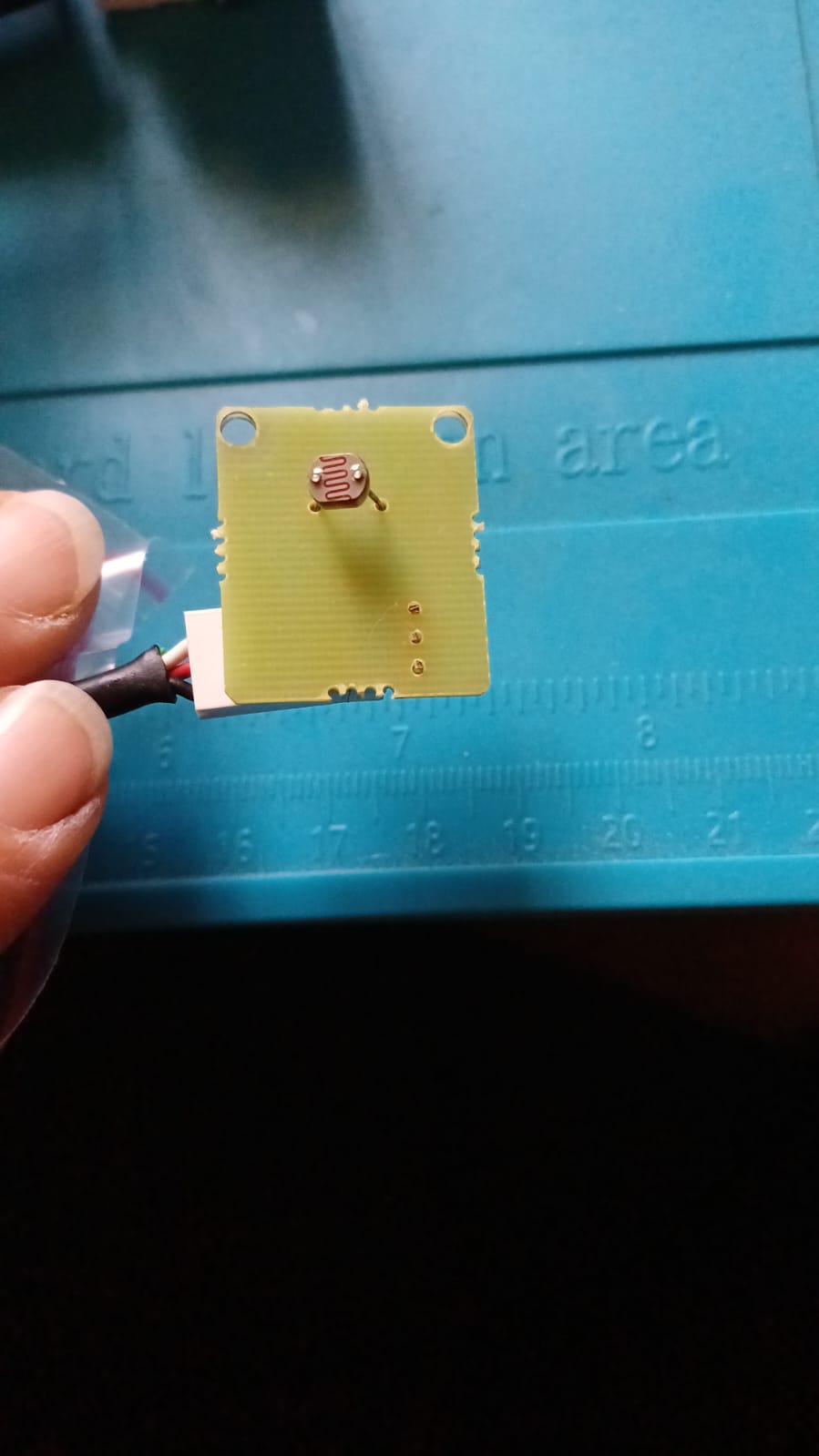
* **Placa Controladora (PCB):** Es la placa verde grande, la "MatBot V1.0". Sirve como una "placa madre" que conecta todos los componentes de forma ordenada.
* **Cerebro (Microcontrolador):** El **Arduino Nano** (la placa azul pequeña). Aquí es donde cargamos nuestros programas (sketches) y se toman todas las decisiones.
* **Controlador de Motores (Puente H):** El chip grande **L298N**. Es el "músculo" del robot. Recibe señales de bajo voltaje del Arduino (5V) y las usa para controlar los motores con mayor voltaje (ej. 9V-12V). Nos permite controlar la **velocidad y dirección** de dos motores.
* **Actuadores (Movimiento):**
  + **Motores y Ruedas:** Hacen que el robot se mueva. Se conectan a los puertos **M0** y **M1**.
* **Periféricos Integrados (En la placa):**
  + **LED Integrado (Pin D13):** Un LED de prueba conectado directamente al pin D13 del Nano. Nota: En tu placa puede ser de color **ROJO**, en otras puede ser azul).
  + **Botones:** La placa solo tiene un interruptor de **Encendido (Power)** y el botón de **Reset** (en el Nano).
* **Sensores (Los "Sentidos" del Robot):**
  + **Sensor de Ultrasonido (HC-SR04):** Mide distancias usando sonido (como un murciélago). Se usa para **esquivar obstáculos**.
  + 
  + **Módulo Seguidor de Línea:** Usa luz infrarroja (IR) para detectar una línea negra sobre fondo blanco (o viceversa).



* + **Receptor Infrarrojo (TSOP):** Es el "oído" que recibe señales del **control remoto**.

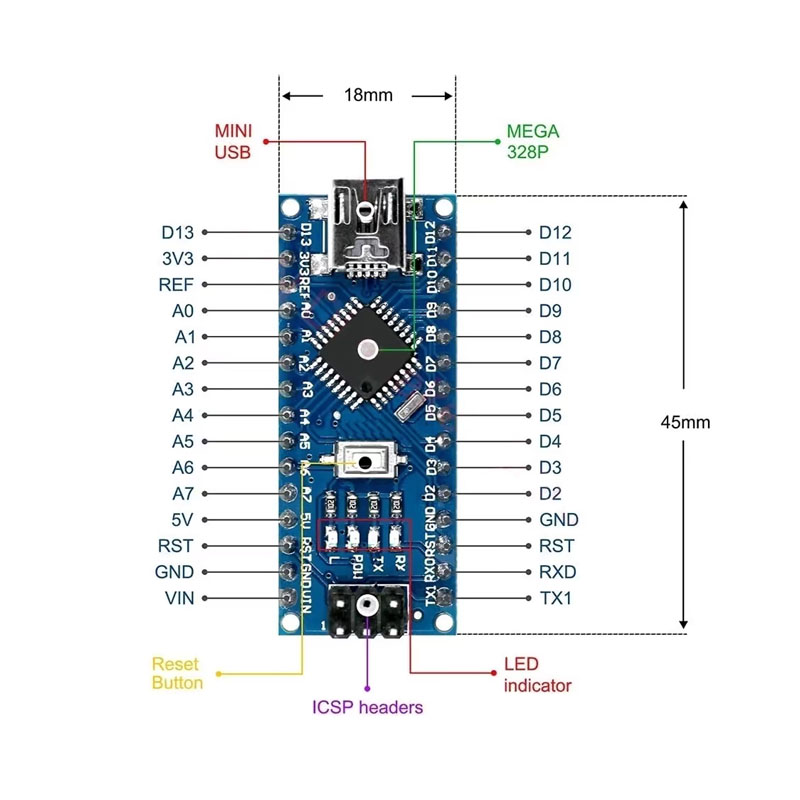


* + **Sensor de Luz (LDR):** Es una fotorresistencia que mide el nivel de luz ambiental.



## **2. Mapeo de Pines: El Cerebro y sus Conexiones**

Para que nuestros programas funcionen, necesitamos saber qué pin del Arduino controla qué cosa.



### **Conexiones Integradas (Componentes Fijos)**

Estas conexiones son fijas en la placa y no podemos cambiarlas:

* **LED Integrado:** Pin D13
* **Motor Derecho (M0):**
  + Velocidad (PWM): Pin **D6**
  + Input 1 (Dirección): Pin **D5**
  + Input 2 (Dirección): Pin **D7**
* **Motor Izquierdo (M1):**
  + Velocidad (PWM): Pin **D3**
  + Input 1 (Dirección): Pin **D2**
  + Input 2 (Dirección): Pin **D4**

### **Puertos de Sensores (Los 12 conectores blancos)**

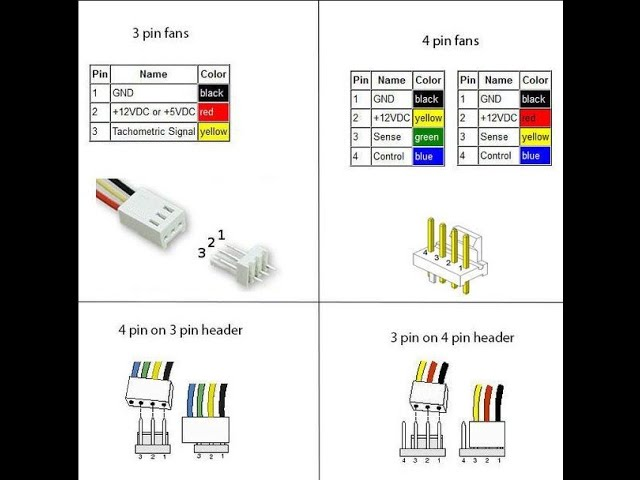
Estos son los puertos donde conectamos nuestros sensores. Todos tienen 3 pines:

* **Pin Externo** (cerca del borde): **GND** (Negativo / Tierra / Cable Negro)
* **Pin Medio:** **VCC** (Positivo / Alimentación 5V / Cable Rojo)
* **Pin Interno** (cerca del Nano/Centro): **SEÑAL** (Datos / Cable de Color)

**Conector de 3 Pines**

Para Sensores de Arduino/Robot

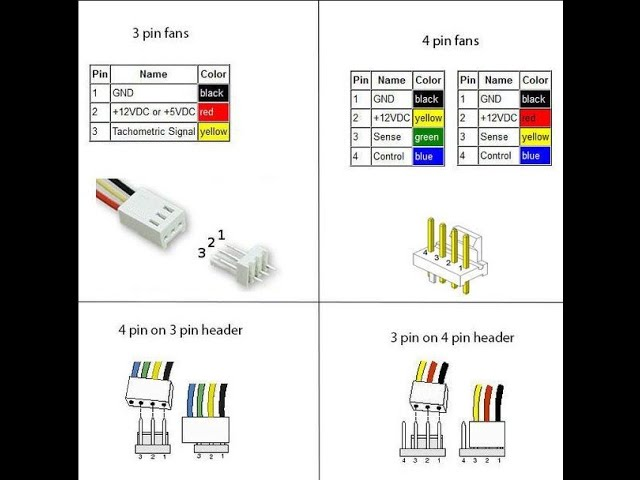
| **Pin** | **Nombre** | **Color del Cable Típico** |
| --- | --- | --- |
| 1 | **GND** | Negro |
| 2 | **VCC (+5V)** | Rojo |
| 3 | **SEÑAL** | Amarillo (o Blanco/Verde) |



**Conector de 4 Pines**

Ejemplo: Ventiladores de PC o Módulos Especiales

| **Pin** | **Nombre** | **Color del Cable Típico** |
| --- | --- | --- |
| 1 | **GND** | Negro |
| 2 | **+12VDC** | Rojo o Amarillo |
| 3 | **Sense** | Verde o Amarillo |
| 4 | **Control (PWM)** | Azul |



| **Puerto (Tapa)** | **Pin de SEÑAL Arduino Nano** | **Tipo de Pin** |  | **Puerto (Tapa)** | **Pin de SEÑAL Arduino Nano** | **Tipo de Pin** |
| --- | --- | --- | --- | --- | --- | --- |
| **PUERTO S0** | **A0** | Analógico | **PUERTO D8** | **D8** | Digital |
| **PUERTO S1** | **A1** | Analógico | **PUERTO D9** | **D9** | Digital |
| **PUERTO S2** | **A2** | Analógico | **PUERTO D10** | **D10** | Digital |
| **PUERTO S3** | **A3** | Analógico | **PUERTO D11** | **D11** | Digital |
| **PUERTO S4** | **A4** | Analógico | **PUERTO D12** | **D12** | Digital |
| **PUERTO S5** | **A5** | Analógico | **PUERTO D13** | **D13** | Digital |

**D3 ¡Compartido con Motor M1!**

**D2 ¡Compartido con Motor M1**

**D7 ¡Compartido con Motor M0!**

Si conectas algo en el puerto **S0** (arriba a la izquierda), en tu código usas A0. Si conectas algo en el puerto de **abajo a la derecha D13**, en tu código usas 13.

### **Analógico vs. Digital: ¿Cuál usar?**

Como puedes ver en la tabla, la placa está organizada de forma muy inteligente:

* **Puertos Analógicos (S0 al S5):** Toda la columna izquierda está conectada a los pines **S**. Son perfectos para sensores que dan un **rango de valores**:
  + Sensor de Luz (LDR)
  + Sensor Seguidor de Línea
  + Potenciómetros
* **Puertos Digitales (D8 al D13):** Toda la columna derecha está conectada a pines **D**. Son perfectos para sensores o actuadores que están **encendidos o apagados (ON/OFF)** o que usan comunicación digital:
  + Sensor de Ultrasonido (Trig en D8, Echo en D9)
  + Receptor Infrarrojo (IR) (Ej: en D11)
  + Botones o pulsadores externos
  + LEDs externos
  + Zumbadores (Buzzers)

## **3. Proyectos de Laboratorio ¿Qué podemos hacer?**

Aquí hay una lista de proyectos, de menor a mayor dificultad, que podemos construir. El código para cada uno está en la siguiente sección.

1. **Nivel 1: "Hola Mundo" y Actuadores**
   1. **LED (Blink):** Probar los componentes integrados.
   2. **Prueba de Motores:** Hacer que el robot se mueva en una secuencia.
   3. **Conectar un LED Externo:** Aprender a usar los puertos de 3 pines.
2. **Nivel 2: Sensores Individuales**4. **Radar de Proximidad con LED Externo**

5. **Robot Esquiva-Obstáculos:** El robot avanza y esquiva lo que encuentra.  
6. **Robot Sigue-Luz**: El robot persigue una fuente de luz (con 2 LDR).

1. **Nivel 3: Proyectos Avanzados**7. **Robot Seguidor de Línea**: El robot sigue una línea negra en el suelo.  
   8. **Robot por Control Remoto:** Controlar el robot con el control IR.  
   9. **Robot Sigue-Personas:** El robot mantiene una distancia fija con un objeto.

## **4. Biblioteca de Códigos**

### **Notas de Configuración**

1. ¡CONECTA LAS PILAS!  
   El cable USB solo da energía al "cerebro" (Nano) y a los sensores. Los motores NO se moverán (o lo harán muy despacio) si no pones las 4 pilas AA en el portapilas para alimentar el chip L298D.
2. ¡CUIDADO CON EL "RUIDO ELÉCTRICO"!  
   Cuando los motores funcionan (con pilas) y el USB está conectado, el "ruido" de los motores puede hacer que el sensor de ultrasonido falle y reporte "0 cm".  
   Para probar el robot, es mejor usarlo solo con pilas y desconectado del USB.
3. CONFIGURACIÓN DEL IDE DE ARDUINO  
   Para cargar el código, la configuración que te funcionó fue:
   * **Placa:** Herramientas > Placa > Arduino AVR Boards > Arduino Nano
   * **Procesador:** Herramientas > Procesador > ATmega328P (Old Bootloader)

### **Módulo Base: Movimiento**

Este bloque de código define los pines y las funciones de movimiento. **Casi todos los proyectos de movimiento lo necesitan.**

**Código:**

/\*  
 \* ==========================================================  
 \* MÓDULO BASE DE MOVIMIENTO - MATBOT 1.0   
 \*

Incluir este bloque en todos los proyectos con motores.

\* Mapeo M0 (Derecho): {D5, D6, D7}

\* Mapeo M1 (Izquierdo): {D2, D3, D4}

\* ==========================================================  
 \*/  
  
// --- Pines Motor Derecho (M0) ---

#define M0\_EN\_PIN 6 // Pin D6 (Enable/Velocidad)

#define M0\_IN1\_PIN 5 // Pin D5 (Input 1)

#define M0\_IN2\_PIN 7 // Pin D7 (Input 2)

// // --- Pines Motor Izquierdo (M1) ---

#define M1\_EN\_PIN 3 // Pin D3 (Enable/Velocidad)

#define M1\_IN1\_PIN 2 // Pin D2 (Input 1)

#define M1\_IN2\_PIN 4 // Pin D4 (Input 2)

void setupMovimiento() {  
 pinMode(M0\_EN\_PIN, OUTPUT);  
 pinMode(M0\_IN1\_PIN, OUTPUT);  
 pinMode(M0\_IN2\_PIN, OUTPUT);  
 pinMode(M1\_EN\_PIN, OUTPUT);  
 pinMode(M1\_IN1\_PIN, OUTPUT);  
 pinMode(M1\_IN2\_PIN, OUTPUT);  
}  
  
// --- Funciones Básicas de Movimiento ---  
void avanzar(int velocidad) {  
 digitalWrite(M0\_IN1\_PIN, HIGH);

digitalWrite(M0\_IN2\_PIN, LOW);

analogWrite(M0\_EN\_PIN, velocidad);  
 digitalWrite(M1\_IN1\_PIN, HIGH);

digitalWrite(M1\_IN2\_PIN, LOW);

analogWrite(M1\_EN\_PIN, velocidad);  
}  
void retroceder(int velocidad) {  
 digitalWrite(M0\_IN1\_PIN, LOW);

digitalWrite(M0\_IN2\_PIN, HIGH);

analogWrite(M0\_EN\_PIN, velocidad);  
 digitalWrite(M1\_IN1\_PIN, LOW);

digitalWrite(M1\_IN2\_PIN, HIGH);

analogWrite(M1\_EN\_PIN, velocidad);  
}  
void girarDerecha(int velocidad) {  
 digitalWrite(M1\_IN1\_PIN, HIGH);

digitalWrite(M1\_IN2\_PIN, LOW);

analogWrite(M1\_EN\_PIN, velocidad);  
 digitalWrite(M0\_IN1\_PIN, LOW);

digitalWrite(M0\_IN2\_PIN, HIGH);

analogWrite(M0\_EN\_PIN, velocidad);  
}  
void girarIzquierda(int velocidad) {  
 digitalWrite(M1\_IN1\_PIN, LOW);

digitalWrite(M1\_IN2\_PIN, HIGH);

analogWrite(M1\_EN\_PIN, velocidad);  
 digitalWrite(M0\_IN1\_PIN, HIGH);

digitalWrite(M0\_IN2\_PIN, LOW);

analogWrite(M0\_EN\_PIN, velocidad);  
}  
void detener() {  
 analogWrite(M0\_EN\_PIN, 0);

analogWrite(M1\_EN\_PIN, 0);  
}

#### **Nota sobre Sensores de 3 Pines (NewPing)**

Los proyectos 4, 5 y 9 usan tu sensor de ultrasonido de 3 pines. Para que funcione, primero debes instalar una biblioteca:

1. Ve a Herramientas > Administrar Bibliotecas...
2. Busca e instala **NewPing** (de Tim Eckel).

### **Proyecto 1: LED**

Objetivo: Verificar que el LED integrado (Pin D13) funciona, haciéndolo parpadear.

Conexiones: Ninguna.

**Código:**

#**define** LED\_INTEGRADO 13 // (En el Nano, también se puede usar LED\_BUILTIN)  
  
void setup() {  
 pinMode(LED\_INTEGRADO, OUTPUT);  
}  
  
void loop() {  
 digitalWrite(LED\_INTEGRADO, HIGH); // Encender LED  
 delay(1000); // Esperar 1 segundo  
 digitalWrite(LED\_INTEGRADO, LOW); // Apagar LED  
 delay(1000); // Esperar 1 segundo  
}

### **Proyecto 2: Prueba de Motores**

Objetivo: Probar que los motores funcionan. La secuencia se iniciará automáticamente.

Conexiones: Motores en M1 y M2.

**Código:**

// --- COPIAR Y PEGAR EL MÓDULO BASE DE MOVIMIENTO COMPLETO AQUÍ ---  
// ... (copia todo el bloque de "Módulo Base" de arriba)  
// ...  
// --- FIN DEL MÓDULO BASE ---  
  
void setup() {  
 setupMovimiento(); // Llama a la configuración de pines  
}  
  
void loop() {  
   
 // Secuencia de prueba (velocidad 200 de 255)  
 avanzar(200);  
 delay(2000); // Avanza por 2 segundos  
  
 retroceder(200);  
 delay(2000); // Retrocede por 2 segundos  
  
 girarDerecha(200);  
 delay(1000); // Gira a la derecha por 1 segundo  
  
 girarIzquierda(200);  
 delay(1000); // Gira a la izquierda por 1 segundo  
  
 detener();  
 delay(3000); // Detenido por 3 segundos  
}

### **Proyecto 3: Conectar un LED Externo (Blink)**

Objetivo: Aprender a usar uno de los puertos de 3 pines como una salida digital simple para encender y apagar un LED.

Conexiones:

* Necesitarás: 1 LED (cualquier color) y 1 Resistencia (valor entre 220 Ohm y 330 Ohm).
* **Pata CORTA (-) del LED** -> Conectar al pin **GND**  del **PUERTO S5**.
* **Pata LARGA (+) del LED** -> Conectar a una pata de la **resistencia**.
* La **OTRA pata de la resistencia** -> Conectar al pin **SEÑAL** del **PUERTO S5 (Pin A5)**.

**Código:**

/\*  
 \* ==========================================================  
 \* Proyecto 3: Blink con LED Externo  
 \* ==========================================================  
 \*/  
  
// Usaremos el pin A5 (PUERTO S5) como si fuera un pin digital.  
// (Los pines Analógicos A0-A5 también pueden usarse como Digitales)

#**define** LED\_EXTERNO\_PIN A5 // Pin de señal del Puerto S5  
  
void setup() {  
 // Configurar el pin A5 como SALIDA  
 pinMode(LED\_EXTERNO\_PIN, OUTPUT);  
}  
  
void loop() {  
 // Parpadeo clásico (Blink)  
 digitalWrite(LED\_EXTERNO\_PIN, HIGH); // Encender  
 delay(1000); // Esperar 1 segundo  
 digitalWrite(LED\_EXTERNO\_PIN, LOW); // Apagar  
 delay(1000); // Esperar 1 segundo  
}

### 

### **Proyecto 4: Radar de Proximidad con LED Externo**

**Objetivo:** El LED **externo** parpadeará más rápido cuanto más cerca esté un objeto del sensor de ultrasonido.

**Conexiones:**

* **Sensor HC-SR04:**
  + **VCC** -> Pin VCC (Medio) del **PUERTO D9**.
  + **GND** -> Pin GND (Externo) del **PUERTO D9**.
  + **SIG (Señal)** -> Pin SEÑAL (Interno) del **PUERTO D9** (corresponde al Pin D9 del Nano).
* **LED Externo (con Resistencia 220-330 Ohm):**
  + Pata CORTA (-) -> Pin GND del **PUERTO S5**
  + Pata LARGA (+) -> Resistencia -> Pin SEÑAL del **PUERTO S5**

**Código:**

/\*

\* ==========================================================

\* Proyecto 4: Radar (Versión para sensor de 3 pines)

\*

\* Este código usa la biblioteca "NewPing", que es necesaria

\* porque nuestro cable-adaptador usa 1 solo pin para

\* Trig (enviar) y Echo (recibir).

\* ==========================================================

\*/

// 1. INCLUIR LA BIBLIOTECA

// Primero, le decimos al programa que usaremos las herramientas

// que descargamos (la biblioteca NewPing).

#include <NewPing.h>

// 2. DEFINIR LOS PINES

// Definimos los "apodos" para los pines que usaremos.

// (Todos son puertos seguros, ¡sin conflictos!)

#define SENSOR\_PIN 9 // Pin D9 (Puerto D9), donde conectamos el sensor

#define LED\_RADAR\_PIN A5 // Pin A5 (Puerto S5), donde conectamos el LED

#define MAX\_DISTANCE 200 // Le decimos al sensor que no mida más allá de 200cm

// 3. CREAR EL OBJETO SENSOR (¡La "Función" Especial!)

// Aquí es donde ocurre la magia de NewPing.

// Creamos una "variable" especial (un "objeto") de tipo NewPing

// y la llamamos "sonar".

//

// NewPing sonar(TRIGGER\_PIN, ECHO\_PIN, MAX\_DISTANCE);

//

// Como nuestro sensor usa el MISMO pin para ambos, le pasamos

// SENSOR\_PIN (o sea, 8) en los dos primeros lugares.

NewPing sonar(SENSOR\_PIN, SENSOR\_PIN, MAX\_DISTANCE);

void setup() {  
 // Configura el pin A5 como salida para el LED  
 pinMode(LED\_RADAR\_PIN, OUTPUT);   
 Serial.begin(9600); // Opcional: para ver la distancia  
}  
  
void loop() {

// 4. LLAMAR A LA "FUNCIÓN" DE NEWPING

// A diferencia de la versión anterior, no necesitamos

// una función "leerDistancia()" separada.

// La biblioteca NewPing ya la tiene adentro.

//

// Para obtener la distancia, simplemente "llamamos"

// a la función .ping\_cm() de nuestro objeto "sonar".

// La biblioteca se encarga de todo (el pulso, la escucha,

// el cálculo) y nos devuelve un número.

int distancia = sonar.ping\_cm();  
 Serial.print("Distancia: "); // Opcional: para debug

Serial.print(distancia);

Serial.println("cm");  
 // Si el sensor no detecta nada, devuelve 0.

// Lo cambiamos a 100 para que el LED no parpadee súper rápido.  
 if (distancia <= 0 || distancia > 100) {  
 distancia = 100;  
 }  
  
 // Mapear la distancia (1-100cm) a un tiempo de parpadeo (10-1000ms)  
 int tiempoDelay = map(distancia, 1, 100, 10, 1000);  
  
 // Hacer el parpadeo con el LED EXTERNO  
 digitalWrite(LED\_RADAR\_PIN, HIGH);  
 delay(tiempoDelay);  
 digitalWrite(LED\_RADAR\_PIN, LOW);  
 delay(tiempoDelay);  
}

### **Proyecto 5: Robot Esquiva-Obstáculos**

Objetivo: El robot avanza y esquiva.

Conexiones Corregidas:

* **Motores:** En M0 y M1.
* **Sensor HC-SR04:**
  + **VCC** -> Pin VCC (Medio) del **PUERTO D9**.
  + **GND** -> Pin GND (Externo) del **PUERTO D9**.
  + **SIG (Señal)** -> Pin SEÑAL (Interno) del **PUERTO D9** (corresponde al Pin D9 del Nano).

**Código:**

/\*  
 \* ==========================================================  
 \* PROYECTO 5: ROBOT ESQUIVA-OBSTÁCULOS  
 \* ==========================================================  
 \*/  
  
// --- PEGAR EL "MÓDULO BASE DE MOVIMIENTO (V4.0)" COMPLETO AQUÍ ---  
// (Este módulo define las funciones: avanzar, retroceder, etc.)  
// ...  
// --- FIN DEL MÓDULO BASE ---  
  
// --- 1. INCLUIR LA BIBLIOTECA ---  
// Le decimos al programa que usaremos la biblioteca "NewPing"  
// que instalamos para nuestro sensor de 3 pines.  
#**include** <NewPing.h>  
  
// --- 2. DEFINICIÓN DE PINES ---  
// Aquí definimos los "apodos" para los pines que usaremos.  
  
#**define** SENSOR\_PIN 9 // Pin D9 (Puerto D9), donde está el sensor  
#**define** MAX\_DISTANCE 200 // Distancia máxima que medirá el sensor (en cm)  
  
// --- 3. CREAR EL OBJETO SENSOR ---  
// Creamos una "variable" especial (un "objeto") de tipo NewPing  
// y la llamamos "sonar".  
// Le decimos que tanto el pin Trig como el Echo están en el  
// MISMO pin (SENSOR\_PIN).  
NewPing sonar(SENSOR\_PIN, SENSOR\_PIN, MAX\_DISTANCE);  
  
// --- 4. CONFIGURACIÓN INICIAL ---  
void setup() {  
 // Prepara los pines de los motores (esta función  
 // viene de nuestro "Módulo Base")  
 setupMovimiento();  
   
 // (Opcional) Inicia el monitor serie para poder ver  
 // las distancias en la computadora.  
 Serial.begin(9600);  
}  
  
// --- 5. BUCLE PRINCIPAL ---  
void loop() {  
 // --- A. MEDIR LA DISTANCIA ---  
 // Usamos la función .ping\_cm() de la biblioteca NewPing.  
 // Esta función hace todo el trabajo (enviar pulso,  
 // escuchar el eco, calcular) y nos devuelve un número  
 // (la distancia en cm).  
 int distancia = sonar.ping\_cm();  
   
 // (Opcional) Imprimimos el valor en el monitor  
 Serial.print("Distancia: ");  
 Serial.print(distancia);  
 Serial.println(" cm");  
  
 // --- B. LÓGICA DE DECISIÓN (EL "CEREBRO") ---  
   
 // Revisamos dos condiciones:  
 // 1. ¿La distancia es menor a 20 cm? (¡Obstáculo!)  
 // 2. ¿La distancia es mayor a 0? (¡Medición válida!)  
 // (NewPing devuelve 0 si hay un error o no detecta nada)  
 if (distancia < 20 && distancia > 0) {  
   
 // --- RUTINA DE "ESQUIVAR" ---  
 // Si ambas condiciones son verdaderas, ejecutamos esta  
 // secuencia de 3 pasos para evitar el choque:  
  
 // a. ¡Frenar!  
 // Detiene los motores inmediatamente.  
 detener();  
 delay(100); // Pausa corta para que el robot se detenga por completo.  
  
 // b. ¡Retroceder!  
 // Damos marcha atrás para tomar distancia del obstáculo.  
 retroceder(150); // (150 es una velocidad media)  
 delay(500); // Lo hace por medio segundo.  
  
 // c. ¡Girar!  
 // Giramos sobre nuestro eje para buscar un nuevo camino.  
 girarDerecha(200); // (200 es una velocidad de giro rápida)  
 delay(500); // Lo hace por medio segundo.  
  
 } else {  
   
 // --- RUTINA DE "AVANZAR" ---  
 // Si no hay ningún obstáculo (distancia > 20 o 0),  
 // simplemente seguimos adelante.  
 avanzar(180); // (180 es una velocidad de avance normal)  
 }  
   
 // Damos una pequeña pausa de 50 milisegundos  
 // antes de volver a medir. Esto da estabilidad  
 // y evita lecturas erráticas del sensor.  
 delay(50);  
}

### 

### **Proyecto 6: Robot Sigue-Luz (con 2 LDR)**

Objetivo: El robot gira y avanza para "perseguir" una fuente de luz brillante, como una linterna.

Conexiones:

* **Motores:** En M0 y M1.
* **LDR Izquierdo:** Enchufar en el **PUERTO S2** (Pin A2).
* **LDR Derecho:** Enchufar en el **PUERTO S3** (Pin A3).

**Código:**

/\*  
 \* ==========================================================  
 \* Proyecto 6: Robot Sigue-Luz  
 \* ==========================================================  
 \*/  
  
// --- PEGAR EL MÓDULO BASE DE MOVIMIENTO AQUÍ ---  
// ...  
// --- FIN DEL MÓDULO BASE ---  
  
// --- Definición de Pines de LDRs ---  
#**define** LDR\_IZQ\_PIN A2  
#**define** LDR\_DER\_PIN A3

// --- Variables de Calibración ---  
// Umbral de luz: qué valor consideramos "suficiente luz" para movernos.  
// ¡Calibra este valor! (Usa el sketch del Proyecto 7 para ver valores)  
int UMBRAL\_LUZ = 400;   
  
// Tolerancia: diferencia mínima de luz para decidir girar.  
// Evita que el robot "vibre" cuando la luz está al frente.  
int TOLERANCIA = 50;   
  
void setup() {  
 setupMovimiento();  
 Serial.begin(9600); // Opcional, para calibrar  
}  
  
void loop() {  
 // Leemos los valores de los LDR.  
 // IMPORTANTE: Dependiendo de cómo esté hecha la placa del LDR,  
 // más luz puede ser un valor MÁS ALTO o MÁS BAJO.  
 // Asumiremos que MÁS LUZ = VALOR MÁS ALTO.  
 // Si tu robot huye de la luz, invierte la lógica.  
 int luzIzq = analogRead(LDR\_IZQ\_PIN);  
 int luzDer = analogRead(LDR\_DER\_PIN);  
  
 // Opcional: imprimir valores para calibrar  
 Serial.print("Izq: "); Serial.print(luzIzq);  
 Serial.print(" | Der: "); Serial.println(luzDer);  
  
 // --- Lógica de decisión ---  
  
 if (luzIzq > (luzDer + TOLERANCIA)) {  
 // 1. Hay más luz a la izquierda  
 girarIzquierda(180);  
 }   
 else if (luzDer > (luzIzq + TOLERANCIA)) {  
 // 2. Hay más luz a la derecha  
 girarDerecha(180);  
 }   
 else {  
 // 3. La luz está al frente (o los valores son parecidos)  
 // Solo avanzamos si hay suficiente luz  
 if (luzIzq > UMBRAL\_LUZ && luzDer > UMBRAL\_LUZ) {  
 avanzar(160);  
 } else {  
 // No hay suficiente luz, nos detenemos  
 detener();  
 }  
 }  
}

### **Proyecto 7: Robot Seguidor de Línea**

Objetivo: El robot sigue una línea negra en el suelo.

Conexiones:

* **Motores:** En M0 y M1.
* **Sensor Izquierdo:** Enchufar en el **PUERTO S0** (Pin A0).
* **Sensor Derecho:** Enchufar en el **PUERTO S1** (Pin A1).

**Código:**

// --- PEGAR EL MÓDULO BASE DE MOVIMIENTO AQUÍ ---  
// ...  
// --- FIN DEL MÓDULO BASE ---  
  
#**define** SENSOR\_IZQ\_PIN A0  
#**define** SENSOR\_DER\_PIN A1  
  
// --- Variable de Calibración ---  
// ¡DEBES AJUSTAR ESTE VALOR!  
// Carga el "Código de Calibración" de abajo para encontrar tu valor.  
// Es el valor intermedio entre leer "blanco" y leer "negro".  
int UMBRAL = 500;   
  
void setup() {  
 setupMovimiento();  
}  
  
void loop() {  
 int valorIzq = analogRead(SENSOR\_IZQ\_PIN);  
 int valorDer = analogRead(SENSOR\_DER\_PIN);  
  
 // Asumimos que BLANCO < UMBRAL y NEGRO > UMBRAL  
 // (Si tu sensor funciona al revés, invierte los < y >)  
  
 if (valorIzq < UMBRAL && valorDer < UMBRAL) {  
 // 1. Ambos en blanco: Avanzar  
 avanzar(150);  
 }   
 else if (valorIzq > UMBRAL && valorDer < UMBRAL) {  
 // 2. Izquierda ve negro: Girar a la izquierda  
 girarIzquierda(180);  
 }   
 else if (valorIzq < UMBRAL && valorDer > UMBRAL) {  
 // 3. Derecha ve negro: Girar a la derecha  
 girarDerecha(180);  
 }   
 else {  
 // 4. Ambos ven negro (o están en el aire): Detenerse  
 detener();  
 }  
}  
  
/\*  
 \* --- CÓDIGO DE CALIBRACIÓN ---  
 \* Carga este sketch TEMPORALMENTE.  
 \* Abre el "Monitor Serie" (Ctrl+Shift+M).  
 \* Pasa los sensores por el blanco y el negro.  
 \* Anota los valores y elige un UMBRAL intermedio.  
   
void setup() {  
 Serial.begin(9600);  
}  
void loop() {  
 int valorIzq = analogRead(A0);  
 int valorDer = analogRead(A1);  
 Serial.print("Izquierdo: ");  
 Serial.print(valorIzq);  
 Serial.print(" | Derecho: ");  
 Serial.println(valorDer);  
 delay(200);  
}  
\*/

### **Proyecto 8: Robot por Control Remoto**

Objetivo: Controlar el robot con el control IR.

Requisito: Instalar la biblioteca IRremote.

Conexiones Corregidas:

* **Motores:** En M0 y M1.
* **Receptor IR (TSOP):** Enchufar en el **PUERTO D11** (Pin D11).

**Código:**

#**include** <IRremote.h>  
  
// --- PEGAR EL "MÓDULO BASE DE MOVIMIENTO" COMPLETO AQUÍ ---  
// ...  
// --- FIN DEL MÓDULO BASE ---  
  
// --- Pin de Sensor ---  
#**define** IR\_RECEIVE\_PIN 11 // Pin Nano D11 (corresponde al Puerto D11)  
  
// --- ¡REEMPLAZA ESTOS CÓDIGOS POR LOS TUYOS! ---  
#**define** CODE\_AVANZAR 0x46 // Ejemplo  
#**define** CODE\_RETROCEDER 0x15 // Ejemplo  
#**define** CODE\_IZQUIERDA 0x44 // Ejemplo  
#**define** CODE\_DERECHA 0x43 // Ejemplo  
#**define** CODE\_DETENER 0x40 // Ejemplo  
  
void setup() {  
 setupMovimiento();  
 IrReceiver.begin(IR\_RECEIVE\_PIN, ENABLE\_LED\_FEEDBACK);  
}  
  
void loop() {  
 if (IrReceiver.decode()) {  
   
 switch (IrReceiver.decodedIRData.command) {  
 case CODE\_AVANZAR:  
 avanzar(200);  
 break;  
 case CODE\_RETROCEDER:  
 retroceder(200);  
 break;  
 case CODE\_IZQUIERDA:  
 girarIzquierda(200);  
 break;  
 case CODE\_DERECHA:  
 girarDerecha(200);  
 break;  
 case CODE\_DETENER:  
 detener();  
 break;  
 default:  
 detener();   
 }  
   
 IrReceiver.resume(); // Listo para el siguiente  
 }  
}

### **Proyecto 9: Robot Sigue-Personas (Sigue-Objeto)**

Objetivo: El robot intenta mantener una distancia fija (ej. 30cm) con un objeto.

Conexiones: Las mismas que el Proyecto 5 (Esquiva-Obstáculos).

* Sen**Motores:** En M0 y M1.
* **Sensor Ultrasónico (3 pines):** Enchufar en el **PUERTO D9**.

**Código:**

// --- PEGAR EL MÓDULO BASE DE MOVIMIENTO AQUÍ ---  
// ...  
// --- FIN DEL MÓDULO BASE ---  
  
// --- Definición de Pines ---  
#**define** SENSOR\_PIN 9 // Pin D9 (Puerto D9)  
#**define** MAX\_DISTANCE 200 // Distancia máxima a medir (en cm)  
  
// --- Crear el Objeto Sensor ---  
NewPing sonar(SENSOR\_PIN, SENSOR\_PIN, MAX\_DISTANCE);  
  
// --- Variables de Comportamiento ---  
int distanciaIdeal = 30; // Distancia a la que queremos estar (en cm)  
int tolerancia = 5; // Margen de error (para que no "vibre")  
  
void setup() {  
 setupMovimiento();  
}  
  
void loop() {  
 int distanciaActual = sonar.ping\_cm();  
  
 if (distanciaActual == 0) {  
 detener(); // Error de lectura  
 return;  
 }  
  
 if (distanciaActual > (distanciaIdeal + tolerancia)) {  
 avanzar(150); // Caso 1: Muy lejos -> Avanzar  
 }   
 else if (distanciaActual < (distanciaIdeal - tolerancia)) {  
 retroceder(150); // Caso 2: Muy cerca -> Retroceder  
 }   
 else {  
 detener(); // Caso 3: Distancia perfecta -> Detenerse  
 }  
 delay(50); // Pequeña pausa  
}