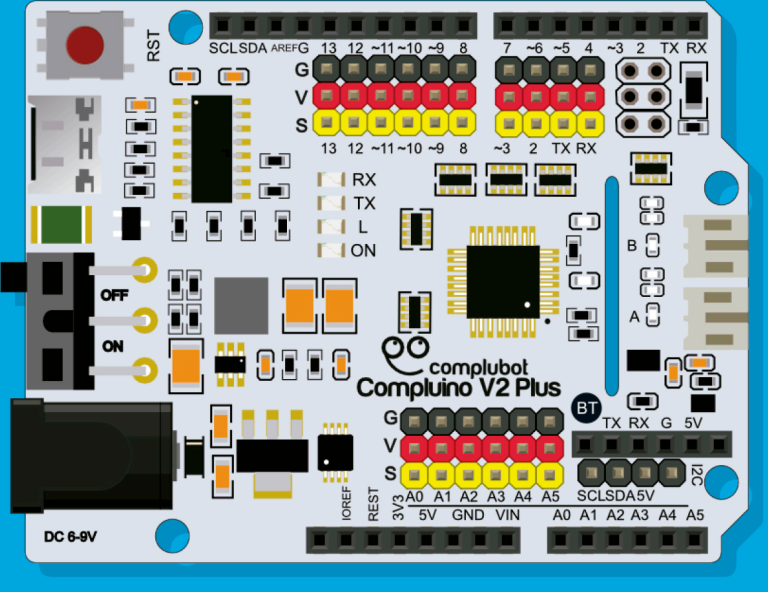
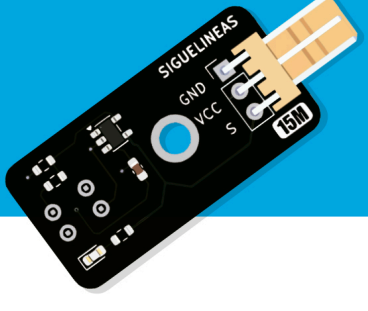
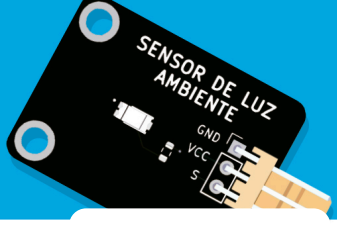
GUÍA DE INICIACIÓN
KIT DE ROBÓTICA PARA COMPLUINO







complubot

1- Contenido del Kit de Robótica para Compluino

El kit de robótica para Compluino está pensado para desarrollar diferentes proyectos de robótica y propuestas STEAM. Con este material podrás realizar numerosas actividades relacionadas con la programación, la electricidad y la electrónica. El kit se compone de múltiples componentes con los que podrás llevar a cabo los montajes propuestos en esta guía y muchos más.

1 - Placa Compluino V2 Plus, con control de motores de CC

2 - Miniservos de 180°

1 - Módulo sensor de distancia por ultrasonidos (34M)

3 - Módulo sensor siguelíneas (15M)

1 - Módulo sensor final de carrera (19M)

1 - Módulo sensor de humedad del suelo (23M)

1 - Módulo sensor de luz ambiente (18M)

1 - Módulo tira RGB tipo NeoPixel (35M)

1 - Módulo pantalla OLED I2C (36M)

2 - Micromotores de CC con reductora (60 RPM a 6 V)

2 - Ruedas multifunción de 40 mm x 3 mm

2 - Ruedas de 43 mm x 19 mm

1 - Rueda loca

1 - Portapilas 5 x AA con conector jack

5 - Pilas alcalinas AA

1 - Conjunto de chasis de metacrilato

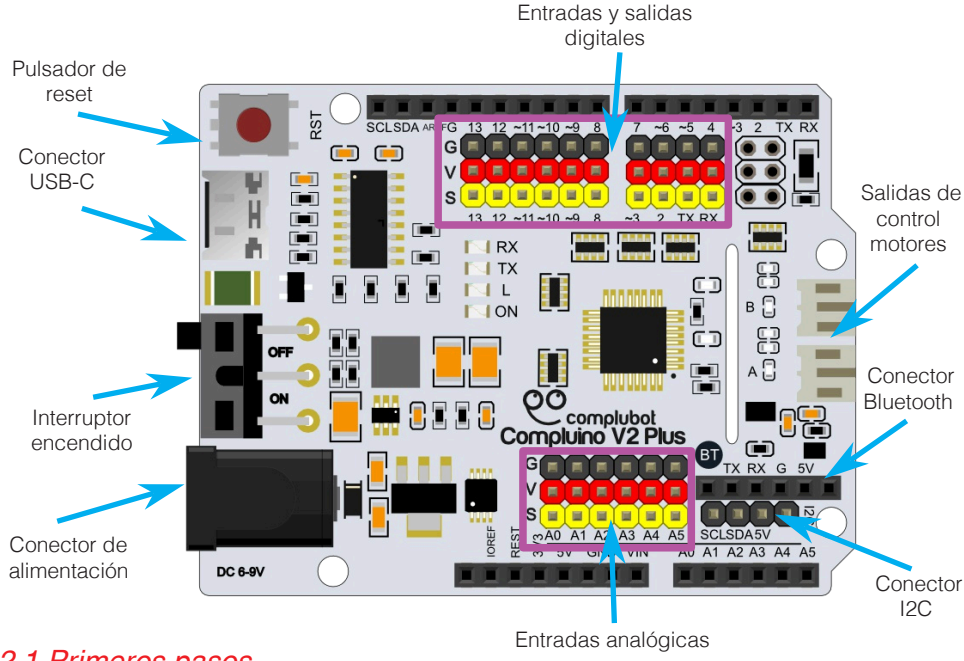
1 - Bolsa con tornillería

Última versión de esta guía en formato digital

Fecha de revisión: 23/04/2026

2 - Placa Compluino V2 Plus

La placa Compluino es una placa compatible multifuncional que incluye importantes mejoras respecto de modelos habituales. Esta placa es compatible con el entorno de programación de Arduino, siendo esta la mejor manera de obtener el 100 % de su potencial.



2.1 Primeros pasos

1. Instalar el IDE de Arduino: Descarga e instala el IDE de Arduino desde la página oficial. Abre el IDE y selecciona tu placa Arduino UNO desde el menú "Herramientas" > "Placa".

2. Escribir tu primer programa: En el IDE, abre el ejemplo "Blink" desde "Archivo" > "Ejemplos" > "01.Basics" > "Blink". Este programa hace que un led en la placa parpadee.

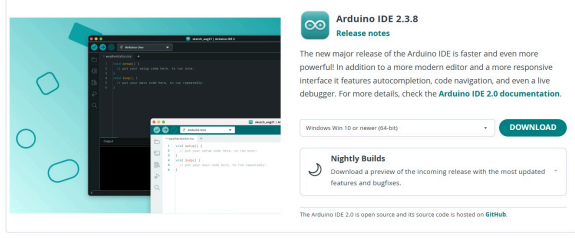
3. Conectar la placa Compluino: Conecta la placa Compluino a tu ordenador usando un cable USB-C.

4. Subir el programa: Haz clic en el botón "Subir" en el IDE para cargar el programa en la placa.

5. Verificar el resultado: El led en la placa debería comenzar a parpadear, confirmando que todo funciona correctamente.

3 - El entorno de programación de Arduino

Este entorno de programación es libre, multiplataforma (Windows, Linux y Mac) y de descarga gratuita. Para instalarlo en tu ordenador, puedes acceder a la página principal del proyecto Arduino, www.arduino.cc, y entrar en la sección “Software”.



La ventana de trabajo del IDE

① **Menú textual.** Acceso a todas las funcionalidades del IDE.

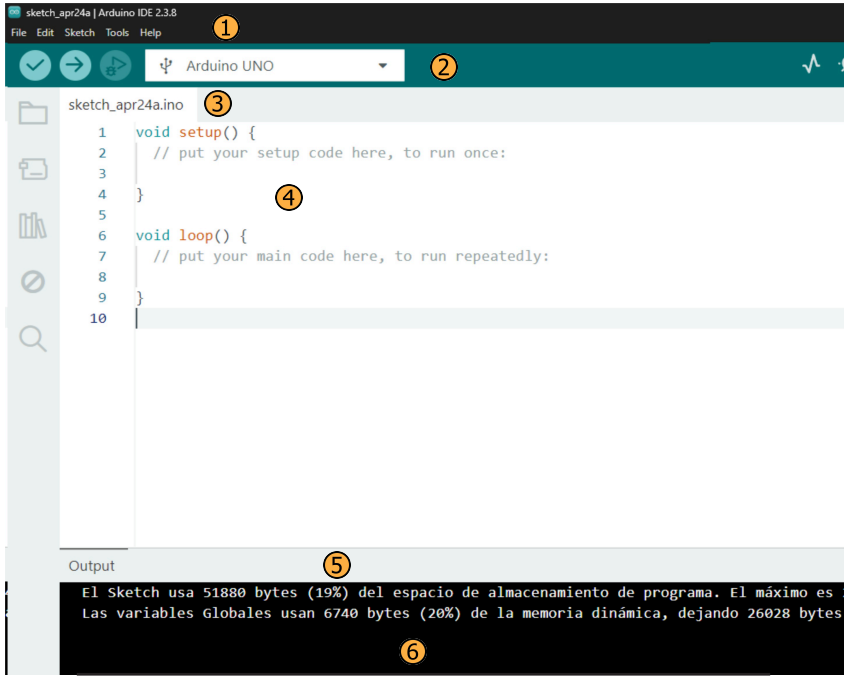
② **Menú de acceso rápido.** Acceso a las funciones más habituales.

③ **Pestañas de programa.** Cuando dividimos el programa en varios archivos.

④ **Área de programación**. Editor donde escribir los programas.

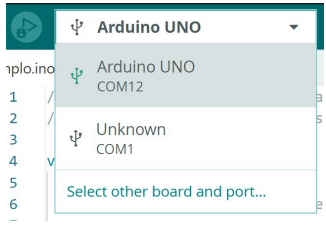
⑤ **Ventana de mensajes.**

⑥ **Barra de estado.**



3.1 Configuración de la placa y del puerto de comunicaciones

Para programar la placa Compluino con el IDE de Arduino hay que conectarla al ordenador con el cable USB-C y seleccionar la placa "Arduino UNO" en el menú desplegable.



3.2 La estructura de un programa en Arduino

El IDE de Arduino utiliza un lenguaje de programación textual basado en el lenguaje C/C++, enriquecido con funciones propias que simplifican la programación de dispositivos. Todo programa presentará las siguientes partes:

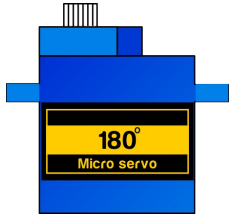
Definiciones: al inicio de un programa se describe en qué consiste y se ① definen las constantes, las variables globales y las bibliotecas que operarán en él.

`void setup()`: es la función de inicialización o configuración. Incluye las ② funciones de configuración del programa y las acciones que deban ejecutarse una sola vez y al comienzo del programa.

`void loop()`: es la función principal y actúa como un bucle infinito. ③ Incluye todas las funciones que queramos que el programa repita de forma secuencial una y otra vez después de **`void setup()`**.

4 - Componentes del kit

4.1 Miniservo de 180°



Un miniservo de 180° es un pequeño motor controlado electrónicamente que puede girar hasta 180 grados (medio círculo). Este dispositivo incluye el driver para controlar el motor interno y un sistema sensor que determina la posición del eje de rotación.

Este dispositivo es muy común en proyectos de robótica debido a su capacidad de moverse de manera precisa. Puede ser posicionado en cualquier ángulo entre 0° y 180°.

4.1.1 Control del miniservo con la placa Compluino

A pesar de su complejidad interna, este tipo de servos se controla de una forma sencilla. Para conectarlo podemos usar cualquiera de las salidas digitales 2, 3, 8, 9, 10, 12 o 13. Ahora bien, para controlarlo necesitamos hacer uso de la librería de control de servos escribiendo al principio del programa:

```
#include <Servo.h>
```

Esta librería forma parte del núcleo de Arduino. Para usarla, solo tenemos que incluirla al inicio del programa.

En este manual también veremos el uso de librerías externas. Para utilizarlas tendremos que hacer uso del gestor de librerías del IDE de Arduino.

A continuación, tenemos que asociar un nombre al miniservo y relacionarlo con el pin al que está conectado.

```
#define SERVO 8
Servo servo180;
```

A continuación, establecemos la vinculación del servo con:

```
servo180.attach(SERVO);
```

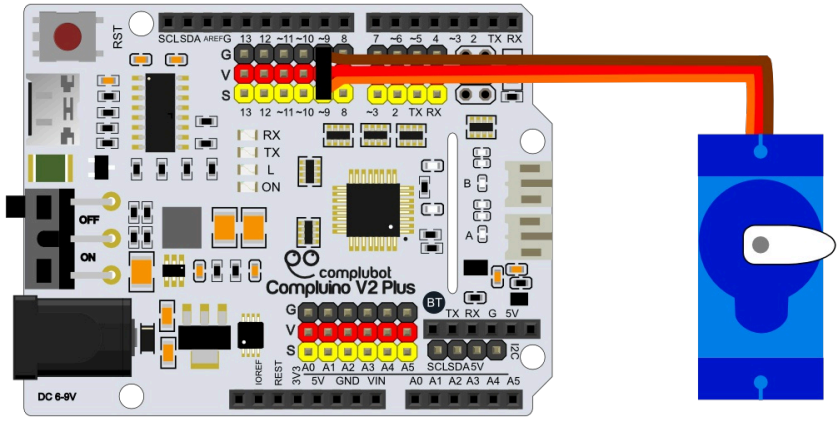
En el caso de conectar más de un servo a la placa Compluino, hay que repetir este proceso para cada servo.

4.1.2 Conexión del miniservo de 180° a la placa Compluino

En esta imagen puedes ver cómo debes conectar los 3 pines del servo a un puerto de salida digital 2, 3, 8, 9, 10, 12 o 13.

Presta especial atención en hacer coincidir el cable marrón con el terminal G, el cable rojo con el terminal V y el cable naranja con el terminal S.

Importante: conectar el miniservo de forma inadecuada puede ocasionar daños al propio servo o a la placa Compluino.



4.1.3 Programa de ejemplo para el miniservo de 180°

```
//para programar servos es necesario instalar la biblioteca <Servo.h>
#include <Servo.h>
//el servo está conectado al PIN 8
#define SERVO 8
//clase y nombre que le queramos poner al servo
Servo servo180;

void setup() {
  servo180.attach(SERVO);
}
void loop() {
//el servo se moverá de un extremo al otro
//pasando por la posición central
  servo180.write (0);
  delay(1000);
  servo180.write (90);
  delay(1000);
  servo180.write (180);
  delay(1000);
  servo180.write (90);
  delay(1000);
}
```

4.1.4 Los comentarios del programa

```
1   // Este es un comentario de una línea
2   // Podemos poner varios seguidos
3
4   /* Este es un comentario
5   de varias líneas
6   que no finaliza hasta que no pongamos
7   los símbolos de terminación */
```

Gracias a los comentarios podemos entender mejor el funcionamiento de los programas. Esto es especialmente interesante tanto a la hora de escribirlo como cuando lo estamos actualizando o manteniendo. En lenguaje C, el que usamos con Arduino, tenemos comentarios de una línea (//) o de múltiples líneas (/* ... */).

4.2 Módulo sensor de distancia por ultrasonidos (34M)

Un sensor de distancia por ultrasonidos es un dispositivo que se utiliza para medir la distancia entre el sensor y un objeto o para detectar la presencia de objetos en su entorno.

Funciona emitiendo ondas de sonido a una frecuencia ultrasónica (más allá del rango audible para los humanos, generalmente por encima de 20 kHz) y midiendo el tiempo que tarda en regresar el eco de estas ondas después de reflejarse en un objeto.

4.2.1 Rango de medida

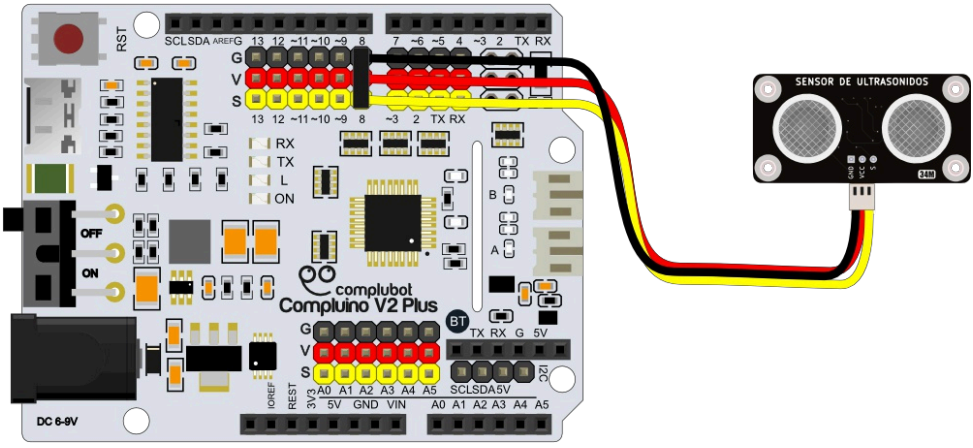
Este sensor de medida de distancia por ultrasonidos tiene un rango de medida de distancias de 2 cm a más de 2 m. Al valor mínimo se le llama "zona muerta" y está determinado por la física del sensor y por el tiempo necesario para el comienzo de la medida.

4.2.2 Conexión módulo sensor de distancia por ultrasonidos a la placa Compluino

En esta imagen puedes ver la conexión del módulo a la placa Compluino mediante un cable Molex-Header de 3 pines.

En el módulo solo existe una posición posible de conexión.

En la placa Compluino, es importante asegurarse de que el cable negro coincida con el terminal negativo (GND).

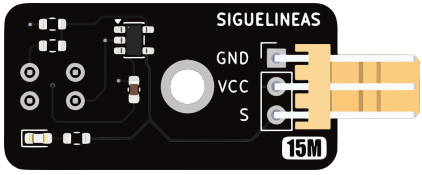


4.2.3 Programa de ejemplo para el sensor de distancia por ultrasonidos

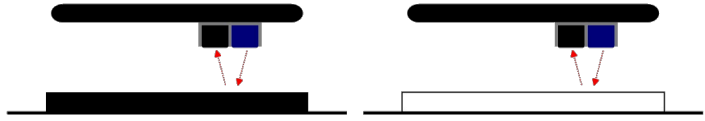
Aquí tienes un ejemplo básico de cómo usar este sensor ultrasónico con la placa Compluino.

```
// ==================================
// Sensor de distancia por ultrasonidos
// Usando un solo pin
// ==================================

#define US 3  // Sensor de ultrasonidos conectado al pin 8

float tiempo = 0;     // Variable para el tiempo
float distancia = 0;  // Variable para la distancia

void setup() {
  Serial.begin(9600);  // Comunicación serie con el monitor
}

void loop() {
  pinMode (US, OUTPUT);  // Preparamos el pin como salida
  digitalWrite(US, LOW);
  delayMicroseconds(5);

  digitalWrite(US, HIGH);  // Iniciamos la medida
  delayMicroseconds(10);
  digitalWrite(US, LOW);

  pinMode (US, INPUT);        // Leemos la respuesta
  tiempo = pulseIn(US, HIGH);
  distancia = tiempo / 58.3;

  Serial.print("La distancia es: ");  // Muestra la medida
  Serial.print(distancia);
  Serial.println(" cm");

  pinMode (US, OUTPUT);  // Preparamos para la siguiente medida
  digitalWrite(US, LOW);

  delay (50);
}
```

4.3 Módulo sensor siguelíneas (15M)



Un sensor siguelíneas, también conocido como sensor de línea o sensor reflexivo, es un dispositivo utilizado en robótica y sistemas de automatización. Estos sensores detectan la cantidad de luz infrarroja que se refleja sobre la superficie que tengan debajo y se utiliza para detectar líneas o marcas negras en un campo de pruebas.



Superficie negra Superficie blanca

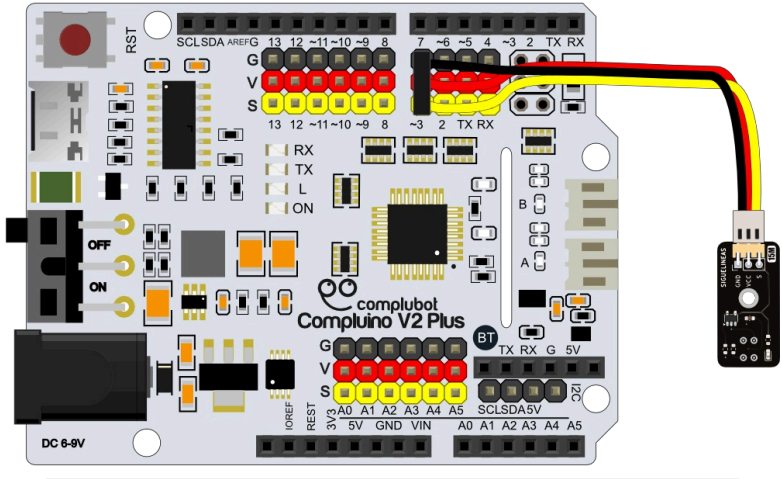
4.3.1 Distancia de funcionamiento

Este sensor está diseñado para conseguir los mejores resultados con una distancia de 1 mm a 3 mm respecto de la superficie. Las medidas pueden ser alteradas por fuentes de luz fuertes (como el sol) que incidan de forma oblicua respecto del sensor, por ello suele ser mejor poner el sensor lo más cerca posible del suelo.

4.3.2 Conexión del módulo sensor siguelíneas a la placa Compluino

En esta imagen puedes ver la conexión del módulo a la placa Compluino mediante un cable Molex-Header de 3 pines.

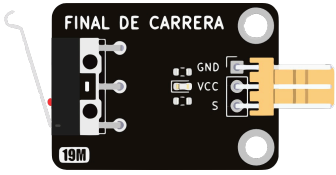
En el módulo solo existe una posición posible de conexión. En la placa Compluino, es importante asegurarse de que el cable negro coincida con el terminal negativo (GND).



4.3.3 Programa de ejemplo para el módulo sensor siguelíneas

```
// ====================================
// Sensor siguelíneas
// Detecta líneas negras sobre fondo claro
// ====================================

// Sensor conectado al pin digital 3
#define SIGUELINEAS 3

void setup() {
  // Configuramos el pin del sensor como entrada
  pinMode(SIGUELINEAS, INPUT);
  // Iniciamos la comunicación con el monitor serie
  Serial.begin(9600);
}
void loop() {
  // Leemos el estado del sensor siguelíneas
  int linea = digitalRead(SIGUELINEAS);
  /* Si el sensor detecta una línea negra,
     devolverá un valor 0 (LOW).Si el sensor
     detecta una superficie clara o blanca,
     devolverá un valor 1 (HIGH). */

// Mostramos el valor leído en el monitor serie
  Serial.println(linea);

  // Pequeña pausa antes de la siguiente lectura
  delay(100);
}
```

4.4 Módulo sensor final de carrera (19M)



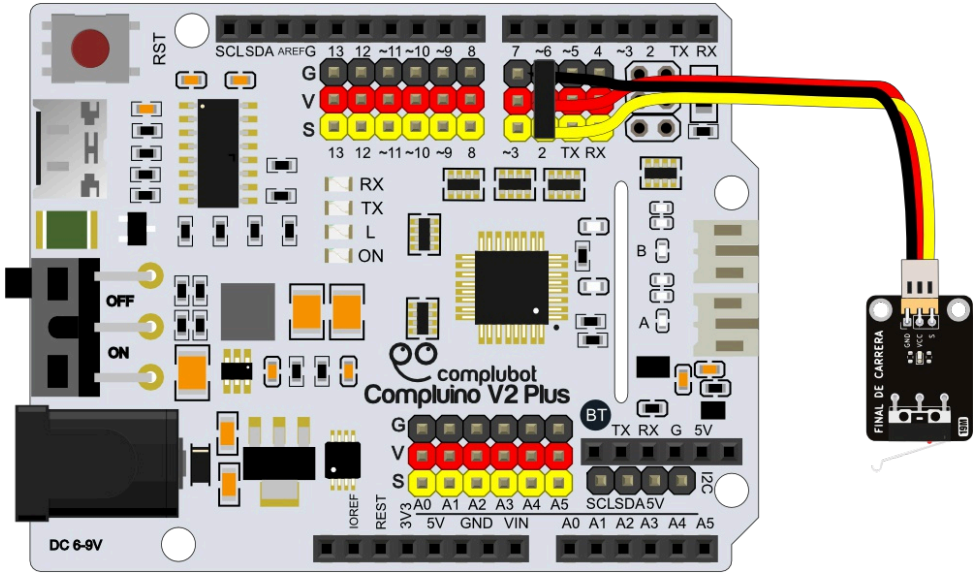
Un sensor final de carrera es un dispositivo electromecánico que se utiliza para detectar la presencia, posición o fin de movimiento de un objeto en sistemas automáticos o robots. Funciona como un interruptor que se activa cuando un objeto presiona su palanca o botón, enviando una señal al controlador del sistema. Es útil para limitar el movimiento de motores, como en impresoras 3D, cintas transportadoras o sistemas de control de acceso, asegurando que el mecanismo no exceda sus límites físicos.

4.4.1 Conexión del módulo sensor final de carrera a la placa Compluino

En esta imagen puedes ver la conexión del módulo a la placa Compluino mediante un cable Molex-Header de 3 pines.

En el módulo solo existe una posición posible de conexión.

En la placa Compluino, es importante asegurarse de que el cable negro coincida con el terminal negativo (GND).

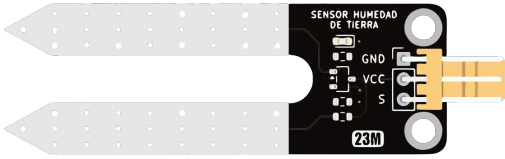


4.4.2 Programa de ejemplo para el sensor final de carrera

Aquí tienes un ejemplo básico de cómo podrías utilizar un módulo sensor final de carrera con la placa Compluino.

```
//
// Programa de ejemplo del funcionamiento del
// final de carrera
//

// Final de carrera conectado al pin 2
#define FINAL_C 2

// LED integrado en la placa
#define LED_USUARIO 13

void setup() {
  // Configuramos el final de carrera como entrada normal
  pinMode(FINAL_C, INPUT);

  // LED como salida
  pinMode(LED_USUARIO, OUTPUT);
}

void loop() {
  // Si el final de carrera se pulsa
  if (digitalRead(FINAL_C) == HIGH) {
    // Interruptor activado → encendemos LED
    digitalWrite(LED_USUARIO, HIGH);
  }
  else {
    // Interruptor no activado → apagamos LED
    digitalWrite(LED_USUARIO, LOW);
  }
}
```

4.5 Módulo sensor de humedad del suelo (23M)



Este módulo es un dispositivo sensor que se utiliza para medir el contenido de humedad en el suelo. Este tipo de sensores son útiles en aplicaciones de agricultura, jardinería y sistemas de riego automatizados para garantizar que las plantas reciban la cantidad adecuada de agua.

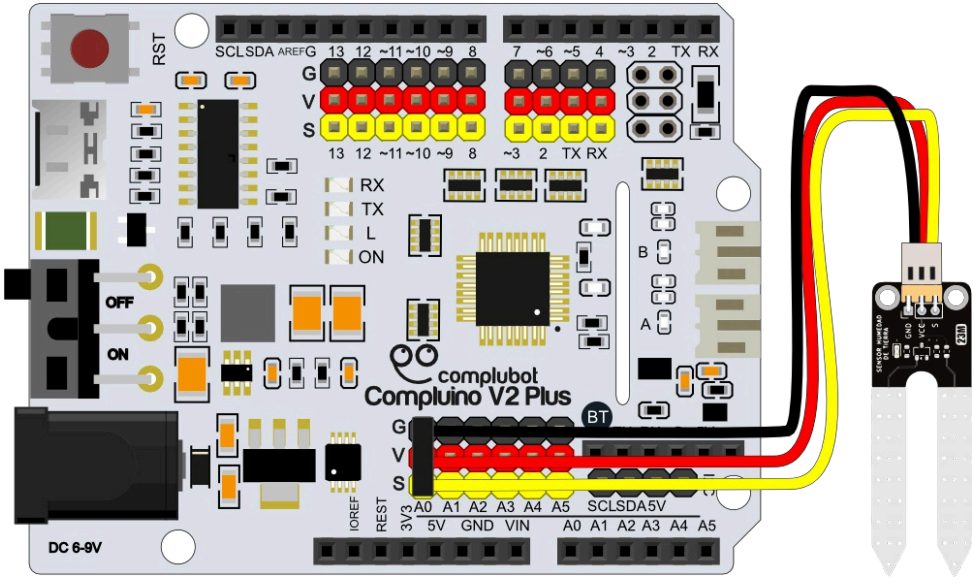
Este sensor está pensado para uso escolar. Una exposición prolongada a terreno húmedo puede disminuir su vida útil.

4.5.1 Conexión del módulo sensor de humedad del suelo a la placa Compluino

En esta imagen puedes ver la conexión del módulo a la placa Compluino mediante un cable Molex-Header de 3 pines.

En el módulo solo existe una posición posible de conexión.

En la placa Compluino, es importante asegurarse de que el cable negro coincida con el terminal negativo (GND).

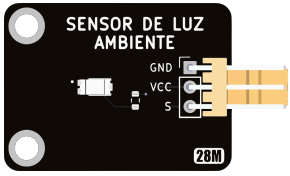


4.5.2 Programa de ejemplo para el sensor de humedad del suelo

Aquí tienes un ejemplo básico de cómo podrías utilizar un sensor de humedad de suelo en un programa utilizando la placa Compluino.

```
// ====================================
// Lectura de un sensor de humedad
// conectado a una entrada analógica
// ====================================
// Este programa permite comprobar el
// funcionamiento del sensor de humedad del suelo.
// Cuanto más alto sea el valor leído,
// menor será la humedad del suelo.
// Los valores se muestran en el monitor serie
// para poder observar los cambios.

#define S_HUMEDAD A0 // Conectamos el sensor al pin A0

void setup() {
  // Iniciamos la comunicación serie para ver los valores
  Serial.begin(9600);
}

void loop() {
  // Leemos el valor analógico del sensor
  // El resultado estará entre 0 y 1023
  int valorHumedad = analogRead(S_HUMEDAD);

  // Mostramos el valor en el monitor serie
  Serial.print("Valor de humedad: ");
  Serial.println(valorHumedad);

  // Pequeña pausa para que la lectura sea estable y legible
  delay(500);
}
```

4.6 Sensor de luz ambiente (18M)



Un sensor de luz es un dispositivo que detecta la intensidad de la luz en su entorno y convierte esta información en una señal eléctrica que puede ser utilizada por un sistema electrónico para tomar decisiones.

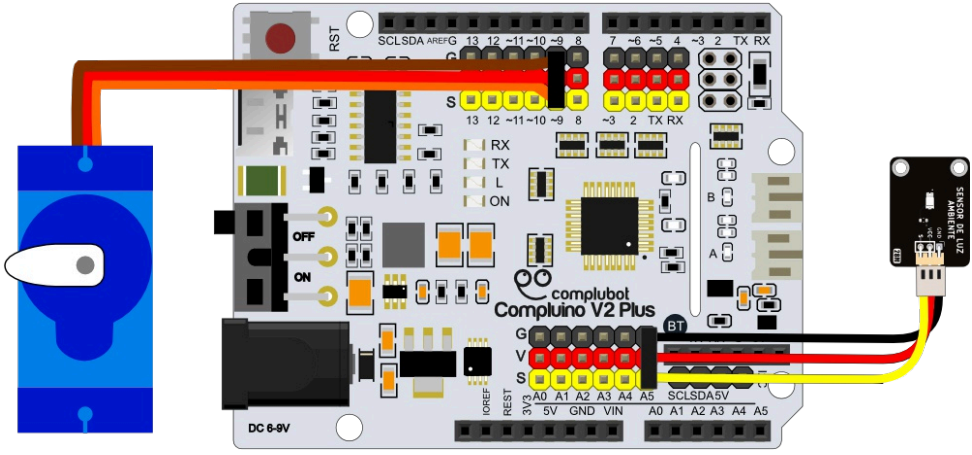
Los sensores de luz se utilizan en una amplia variedad de aplicaciones, desde dispositivos simples hasta sistemas complejos de automatización.

4.6.1 Conexión del módulo sensor de luz ambiente a la placa Compluino

En esta imagen puedes ver la conexión del módulo a la placa Compluino mediante un cable Molex-Header de 3 pines.

En el módulo solo existe una posición posible de conexión.

En la placa Compluino, es importante asegurarse de que el cable negro coincida con el terminal negativo (GND).

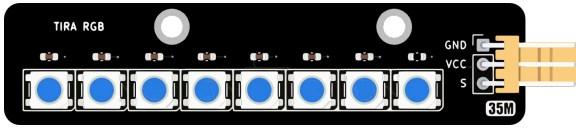


4.6.2 Programa de ejemplo para el sensor de luz ambiente

Control de un servo con la luz ambiente

```
// ====================================
// Control de un servo con un sensor de luz
// El servo se mantiene quieto con poca luz
// y se mueve entre 0º y 180º cuando hay más luz.
// ====================================

#include <Servo.h>
#define S_AMBIENTE A5 // Sensor de luz conectado al A5
#define SERVO 9  // Servo conectado al pin digital 9
Servo miServo;  // Objeto servo
void setup() {
  Serial.begin(9600);
  miServo.attach(SERVO);
  miServo.write(90);
  delay(500);
}

void loop() {
  int luz = analogRead(S_AMBIENTE);
  if (luz < 512) {
    miServo.write(90);
  }
  else {
    miServo.write(0);
    delay(3000);

    miServo.write(180);
    delay(3000);
  }
}
```

4.7 Tira de ledes RGB tipo NeoPixel (35M)



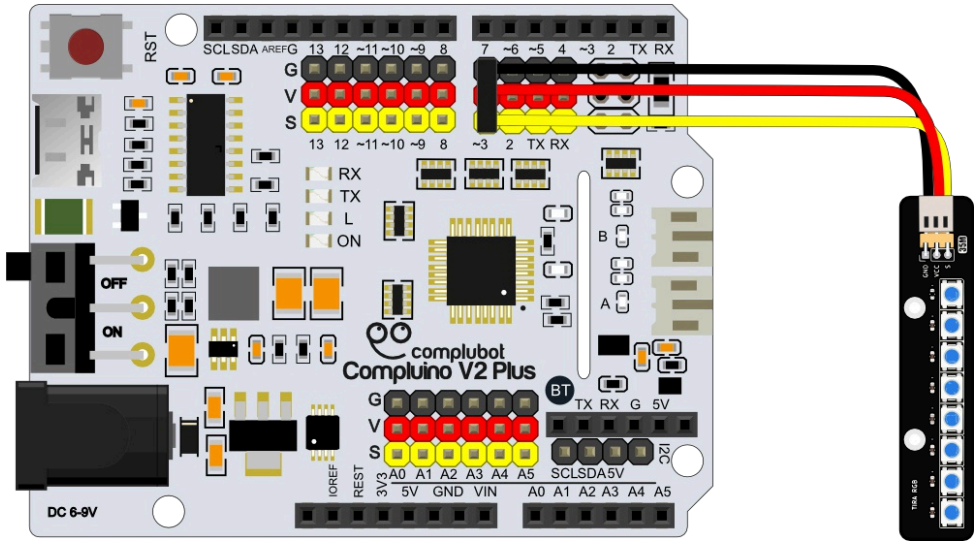
Una tira de ledes RGB es un módulo formado por 8 ledes RGB capaces de generar distintos colores combinando rojo, verde y azul. Cada led puede controlarse de manera independiente, permitiendo crear efectos visuales dinámicos y personalizados. Con la placa Compluino, estas tiras pueden conectarse y programarse fácilmente para usarlas en proyectos de iluminación, decoración, robótica o sistemas interactivos, añadiendo efectos avanzados de forma sencilla y educativa.

4.7.1 Conexión de la tira de ledes RGB a la placa Compluino

En esta imagen puedes ver la conexión del módulo a la placa Compluino mediante un cable Molex-Header de 3 pines.

En el módulo solo existe una posición posible de conexión.

En la placa Compluino, es importante asegurarse de que el cable negro coincida con el terminal negativo (GND).

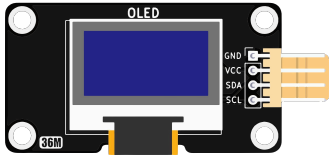


4.7.2 Programa de ejemplo para la tira de ledes RGB

Aquí tienes un ejemplo básico de cómo podrías utilizar una tira led RGB en un programa con la placa Compluino.

```
// ====================================
// Tira LED RGB direccionable (NeoPixel)
// Ejemplo de cambio de colores
// ====================================
// Librería para controlar LEDs RGB tipo NeoPixel
#include <Adafruit_NeoPixel.h>
#define NUM_LEDS 8 // Número total de LEDs en la tira
#define PIN_TIRA 3 // Pin donde está conectada la tira LED
// Creamos el objeto "tira" indicando: número de LEDs, pin de conexión y tipo de LED
Adafruit_NeoPixel tira(NUM_LEDS, PIN_TIRA, NEO_GRB + NEO_KHZ800);
void setup() {
  tira.begin(); // Inicializamos la tira LED
  tira.show();  // Apagamos todos los LEDs al iniciar
}
void loop() {
  // PRIMER COLOR: ROJO
  // Encendemos todos los LEDs en color rojo
  for (int i = 0; i < NUM_LEDS; i++) {
    tira.setPixelColor(i, tira.Color(255, 0, 0));
  }
  tira.show(); // Actualizamos la tira para mostrar el color
  delay(1000);  // Esperamos 1 segundo
  // SEGUNDO COLOR: VERDE
  for (int i = 0; i < NUM_LEDS; i++) {
    tira.setPixelColor(i, tira.Color(0, 255, 0));
  }
  tira.show(); // Actualizamos la tira
  delay(1000); // Esperamos 1 segundo
  // TERCER COLOR: AZUL
  for (int i = 0; i < NUM_LEDS; i++) {
    tira.setPixelColor(i, tira.Color(0, 0, 255));
  }
  tira.show();
  delay(1000);
}
```

4.8 Módulo pantalla OLED (36M)



La pantalla OLED de este módulo es gráfica, tiene una resolución de 128 x 64 y un tamaño de 0,96 pulgadas. Es monocromo de color blanco y en ella se pueden mostrar gráficos, texto, iconos e imágenes. Con este módulo podemos crear indicadores, menús, animaciones simples y mensajes de estado.

4.8.1 Control del módulo pantalla OLED con la placa Compluino

Este módulo OLED utiliza un controlador tipo SSD1306 que se comunica con la placa Compluino mediante un bus del tipo I2C. En este bus cada dispositivo tiene una dirección y la de este módulo es la 0x3C.

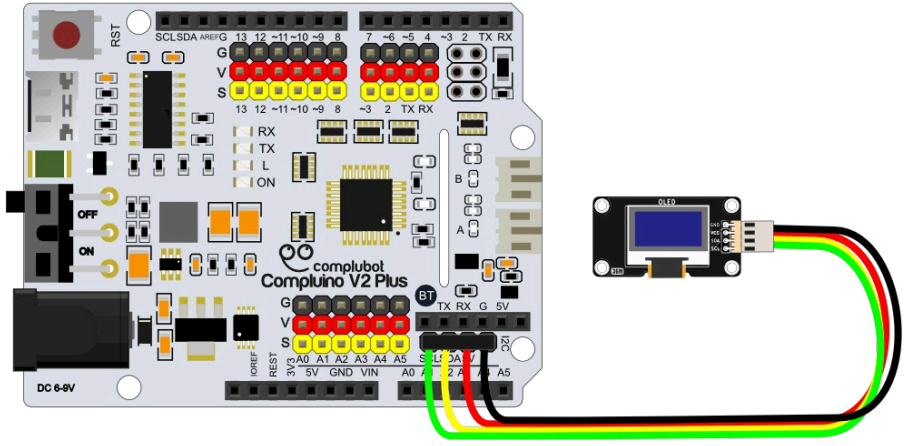
Para controlar la pantalla necesitamos dos librerías:

- Wire.h, es una librería interna de Arduino (no hay que instalarla) y sirve para poder usar el puerto I2C en la placa Compluino.

- Adafruit_SSD1306.h, es una librería externa (hay que instalarla con el gestor de librerías) y sirve para usar este módulo con pantalla OLED.

4.8.2 Conexión del módulo pantalla OLED a la placa Compluino

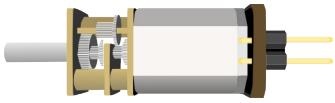
En esta imagen puedes ver la conexión del módulo a la placa Compluino mediante un cable Molex-Header de 4 pines. En el módulo solo existe una posición posible de conexión. En la placa Compluino, es importante asegurarse de que el cable negro coincida con el terminal negativo (GND).



4.8.3 Programa de ejemplo para la pantalla OLED

```
// ====================================
// Programa de ejemplo para pantalla OLED
// Muestra "Hola" y "Adios"
// alternando cada 2 segundos.
// ====================================
#include <Wire.h> // Librería para comunicación I2C
#include <Adafruit_SSD1306.h> // Librería para controlar la pantalla OLED
#define OLED_DIR 0x3C // Dirección I2C de la pantalla OLED
#define ANCHO 128 // Número de columnas de la pantalla
#define ALTO 64   // Número de filas de la pantalla
// Creamos el objeto de la pantalla indicando tamaño y comunicación I2C
Adafruit_SSD1306 oled(ANCHO, ALTO, &Wire, -1);
void setup() {
  // Inicializamos la pantalla OLED
  oled.begin(SSD1306_SWITCHCAPVCC, OLED_DIR);
  oled.setTextSize(2); // Tamaño del texto
  oled.setTextColor(SSD1306_WHITE); // Color del texto
}
void loop() {
  oled.clearDisplay(); // Borramos la pantalla antes de escribir
  oled.setCursor(15, 25); // Posición del cursor (x, y)
  oled.println("Hola"); // Escribimos el texto "Hola"
  oled.display();  // Enviamos el contenido a la pantalla
  delay(2000);
  oled.clearDisplay();  // Volvemos a limpiar la pantalla
  oled.setCursor(20, 25); // Colocamos el cursor en otra posición
  oled.println("Adios");
  oled.display(); // Actualizamos la pantalla
  delay(2000);
}
```

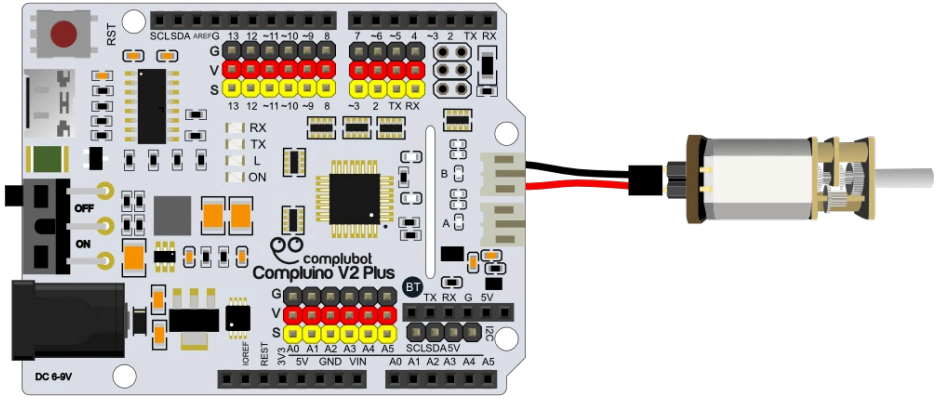
4.9 Micromotores de CC con reductora



Este kit incorpora dos micromotores de corriente continua, del tipo N20, con reductora. Además cada motor dispone de un circuito adaptador para facilitar la conexión a la placa y de dos ledes (uno rojo y otro verde) que muestran en todo momento el sentido de giro del motor.

4.9.1 Conexión de los micromotores de CC a la placa Compluino

La conexión de los micromotores se realiza mediante un cable Header-JST de dos pines. El extremo Header se conecta al motor y el JST a la placa Compluino.



La placa Compluino dispone de dos salidas directas para control de motores gracias al driver que se encuentra integrado en la propia placa. Estos motores se controlan mediante 4 pines de la placa, de acuerdo a la siguiente tabla:

	DIR Dirección del motor	EN Velocidad del motor
Motor A	D4	D5
Motor B	D7	D6

4.9.2 Programa de ejemplo para dos micromotores de CC

A continuación se muestra un ejemplo básico de uso de los motores con la placa Compluino.

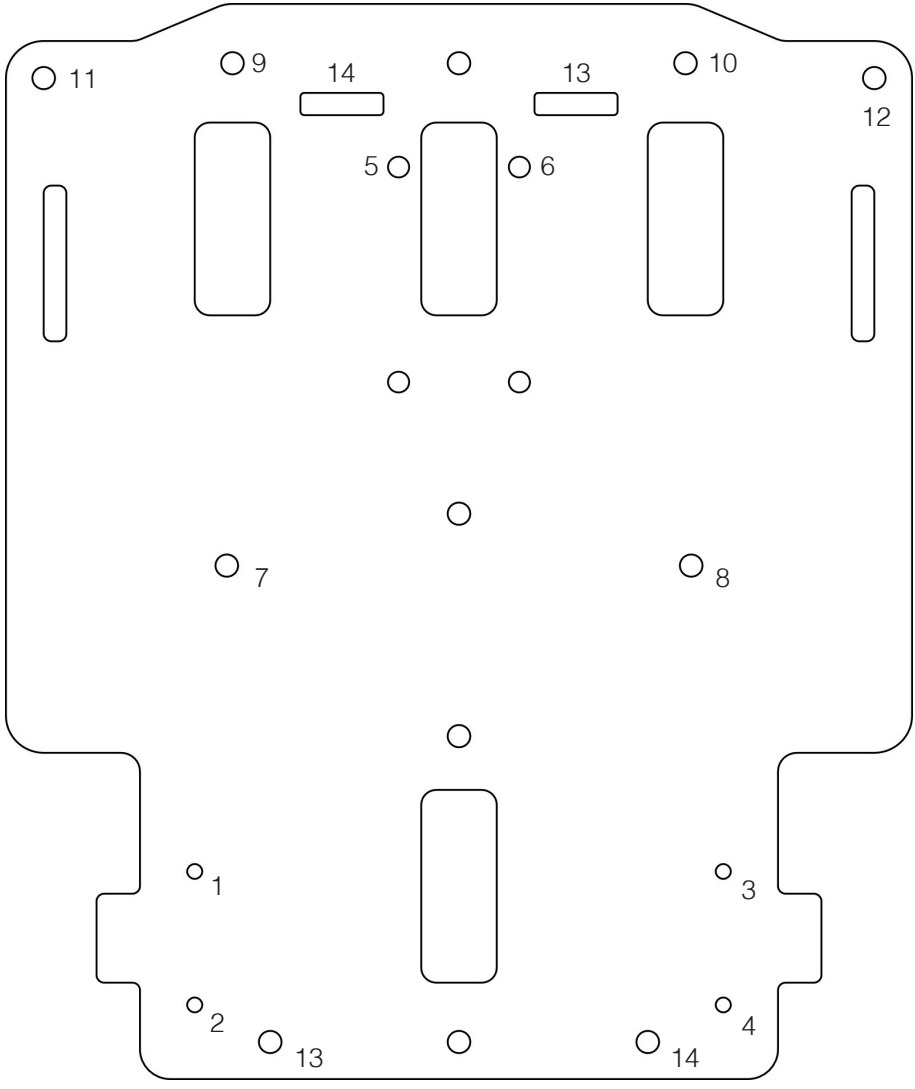
```
#define DIR_A 4 // Controla la dirección del motor A
#define EN_A 5  // Controla la velocidad del motor A
#define DIR_B 7 // Controla la dirección del motor B
#define EN_B 6  // Controla la velocidad del motor B

int velocidad = 0; // Almacenar la velocidad del motor
void setup() {
  pinMode(DIR_A, OUTPUT); // Configura el pin DIR_A como salida
  pinMode(EN_A, OUTPUT); // Configura el pin EN_A como salida
  pinMode(DIR_B, OUTPUT); // Configura el pin DIR_B como salida
  pinMode(EN_B, OUTPUT); // Configura el pin EN_B como salida
}

void loop() {
  digitalWrite(DIR_A, HIGH); // Motor A en sentido de avance
  digitalWrite(DIR_B, HIGH); // Motor B en sentido de avance
  for (velocidad = 0; velocidad <= 255; velocidad += 5) {
    analogWrite(EN_A, velocidad); // Ajusta velocidad del motor A
    analogWrite(EN_B, velocidad); // Ajusta velocidad del motor B
    delay(100); // Espera 100 ms antes de aumentar la velocidad
  }

  analogWrite(EN_A, 0); // Para el motor A
  analogWrite(EN_B, 0); // Para el motor B
  delay(1000); // Espera 1 segundo antes de cambiar la dirección

  digitalWrite(DIR_A, LOW); // Motor A en sentido de retroceso
  digitalWrite(DIR_B, LOW); // Motor B en sentido de retroceso
  for (velocidad = 0; velocidad <= 255; velocidad += 5) {
    analogWrite(EN_A, velocidad); // Ajusta velocidad del motor A
    analogWrite(EN_B, velocidad); // Ajusta velocidad del motor B
    delay(200); // Espera 200 ms antes de aumentar la velocidad
  }

  analogWrite(EN_A, 0); // Para el motor A
  analogWrite(EN_B, 0); // Para el motor B
  delay(2000); // Espera 2 segundos antes de repetir el ciclo
}
```

5 - Robot siguelíneas con Compluino

5.1 Chasis

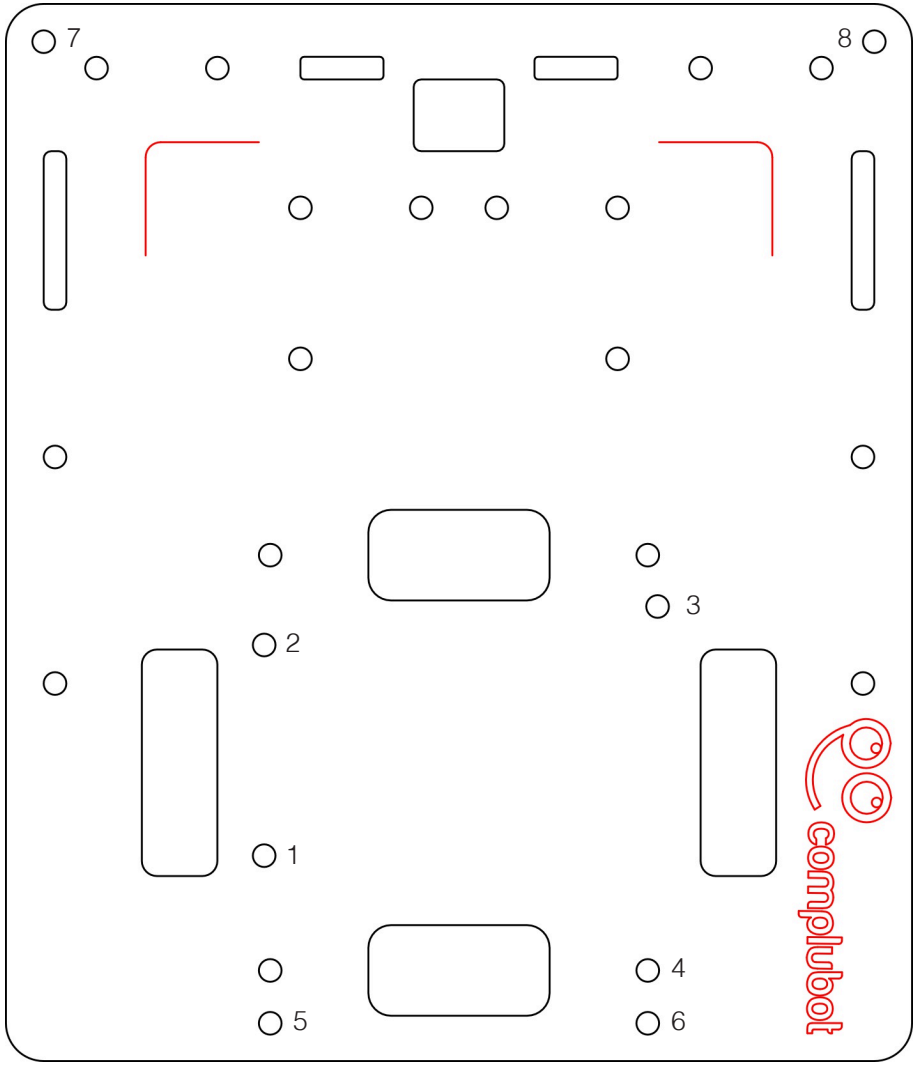
5.1.1 Chasis inferior (escala 1:1)

En esta imagen se muestra ver a escala real una numeración de todos los agujeros de la parte inferior del chasis.



5.1.2 Chasis superior (escala 1:1)

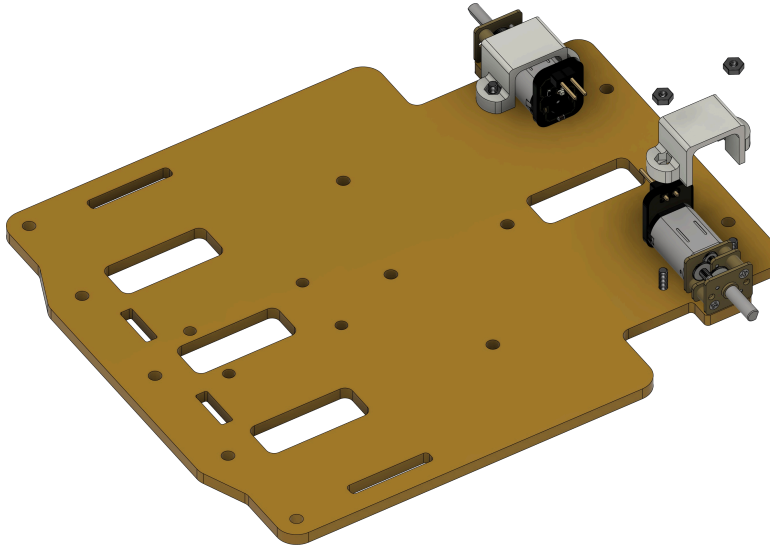
En esta imagen se muestra ver a escala real una numeración de todos los agujeros de la parte superior del chasis.



5.2 Montaje del chasis del robot

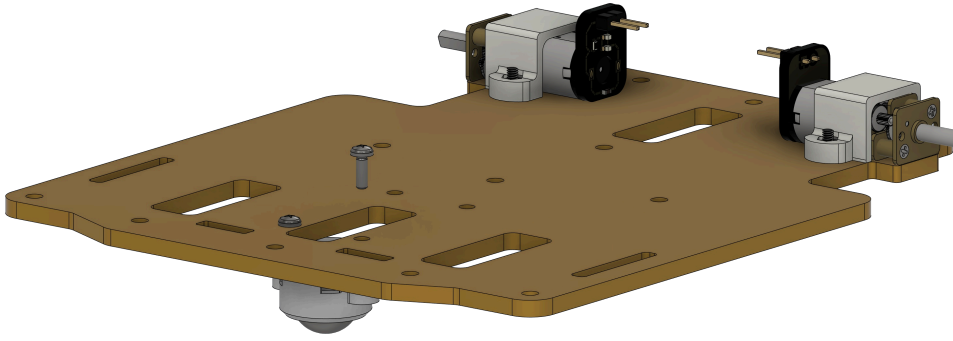
5.2.1 Paso 1 - motores

Coloca los motores con los soportes sobre el chasis. Encaja los agujeros de los soportes con los agujeros 1 y 2 en el lado derecho del chasis y con el 3 y 4 en el lado izquierdo. Utiliza tornillos y tuercas de M2 para fijarlos al chasis.



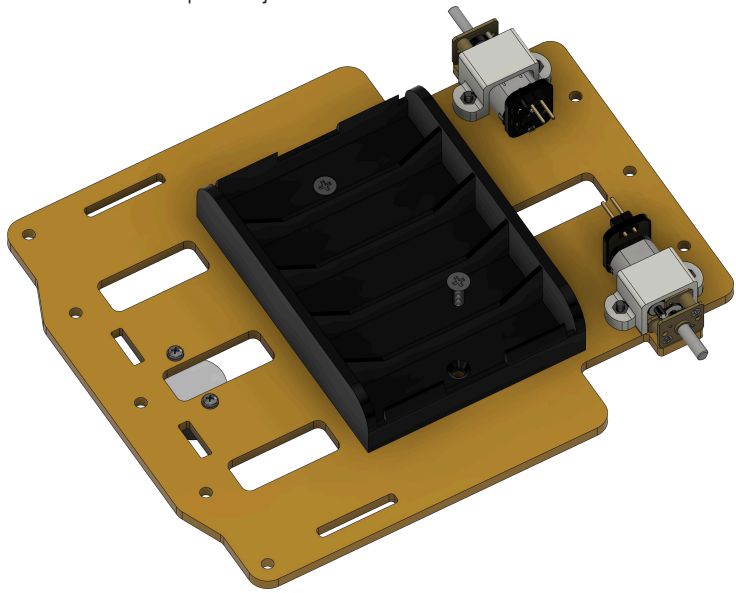
5.2.2 Paso 2 - rueda loca

Coloca la rueda loca en la parte inferior del chasis en los agujeros 5 y 6. Utiliza los tornillos que vienen montados en la rueda loca.



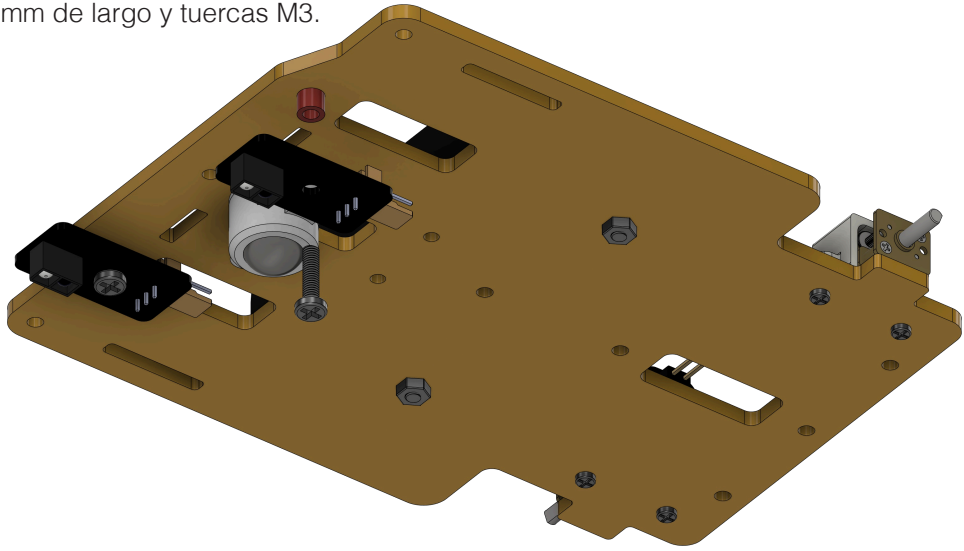
5.2.3 Paso 3 - portapilas

Coloca el portapilas encajándolo en los agujeros 7 y 8 del chasis. Utiliza tornillos y tuercas de M3 para fijarlo al chasis.



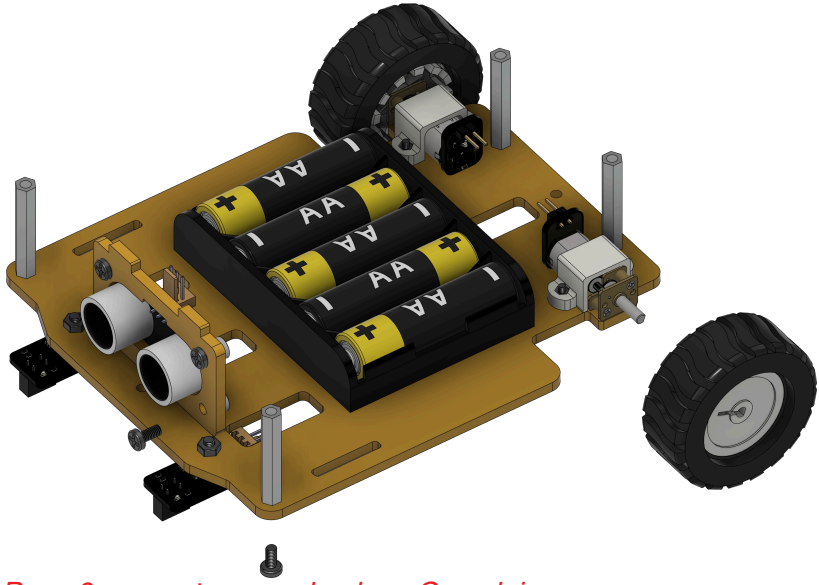
5.2.4 Paso 4 - módulos siguelíneas

Coloca los módulos siguelíneas en la parte inferior del chasis encajándolos en los agujeros 9 y 10. Utiliza separadores más largos para mantener los sensores lo más próximos posible de la superficie. Después, fíjalos con tornillos de 16 mm de largo y tuercas M3.



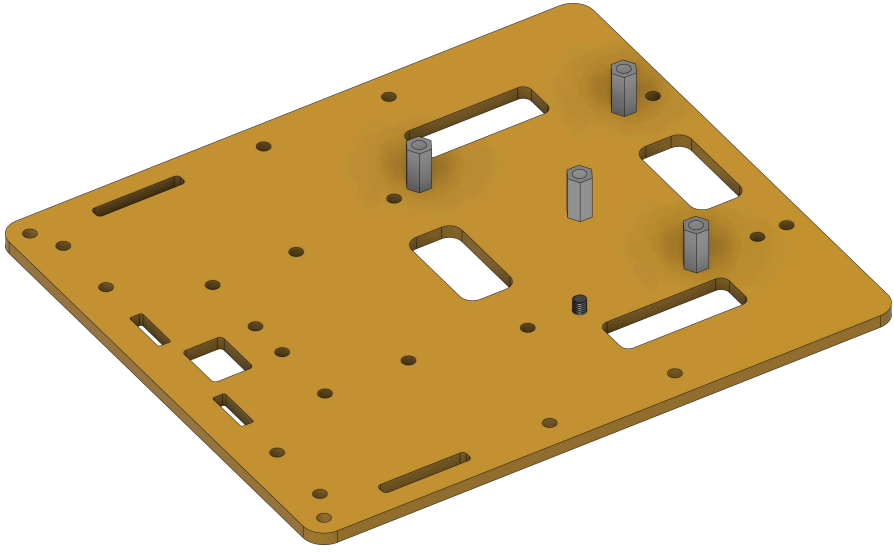
5.2.5 Paso 5 - ruedas y sensor de ultrasonidos

Coloca los separadores largos en los agujeros 11, 12, 13 y 14. En los agujeros 13 y 14 coloca el soporte del sensor de ultrasonidos y sujeta el sensor con tornillos de M3 cortos. Coloca las ruedas en los ejes de los motores.



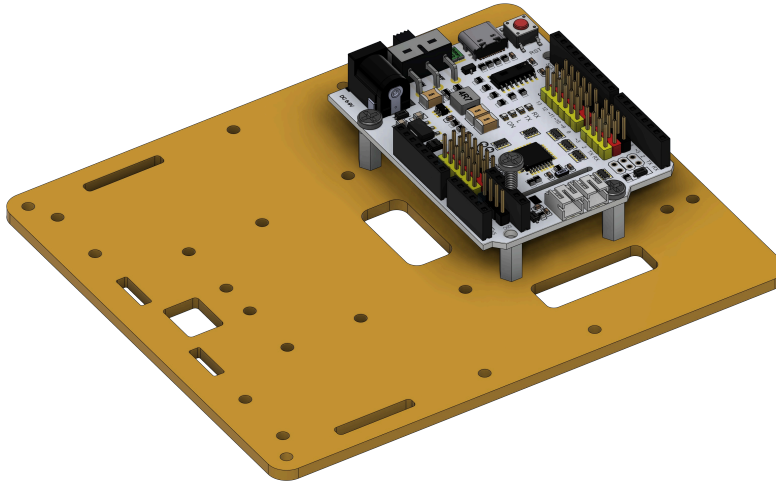
5.2.6 Paso 6 - soportes para la placa Compluino

Coge la parte superior del chasis y coloca separadores cortos en los agujeros 1, 2, 3 y 4.



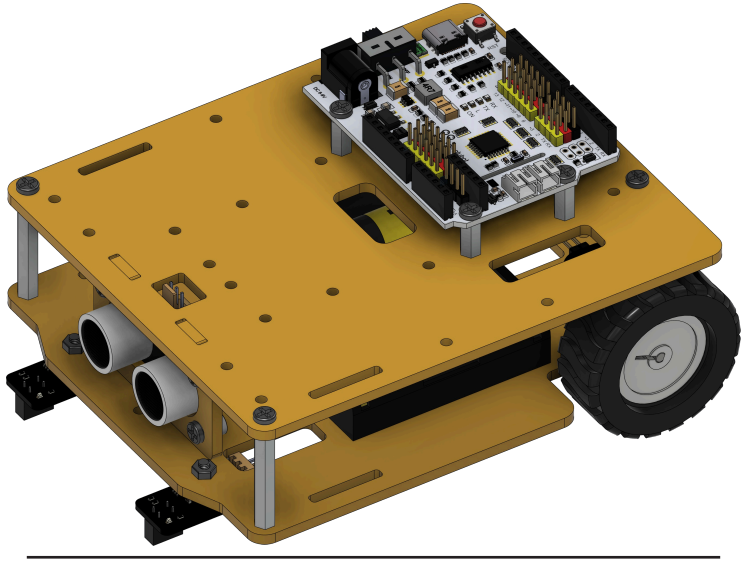
5.2.7 Paso 7 - placa Compluino

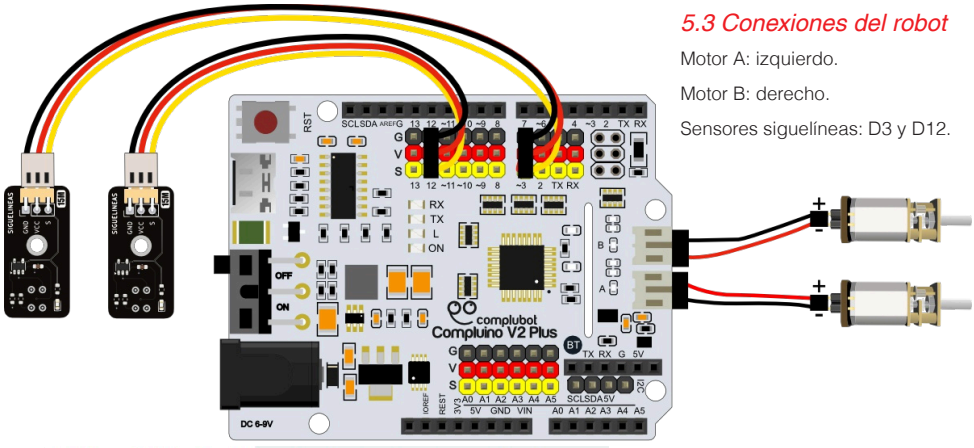
Coloca la placa Compluino sobre los separadores instalados en el paso anterior. Utiliza tornillos M3 para fijarla al chasis.



5.2.8 Paso 8 - montaje final

Une la parte superior del chasis con la parte inferior, colocando los tornillos en los separadores instalados previamente.





5.3 Conexiones del robot

Motor A: izquierdo.

Motor B: derecho.

Sensores siguelíneas: D3 y D12.

```
#define DIR_A 4    // Controla la dirección del motor A
#define EN_A 5     // Controla la velocidad del motor A

#define DIR_B 7    // Controla la dirección del motor B
#define EN_B 6     // Controla la velocidad del motor B

int velocidad = 150;

void setup() {
  pinMode(DIR_A, OUTPUT);
  pinMode(EN_A, OUTPUT);

  pinMode(DIR_B, OUTPUT);
  pinMode(EN_B, OUTPUT);
}

void loop() {
  // AVANZAR 3 SEGUNDOS
  digitalWrite(DIR_A, HIGH);
  analogWrite(EN_A, velocidad);

  digitalWrite(DIR_B, HIGH);
  analogWrite(EN_B, velocidad);

  delay(3000);

  // RETROCEDER 3 SEGUNDOS
  digitalWrite(DIR_A, LOW);
  analogWrite(EN_A, velocidad);

  digitalWrite(DIR_B, LOW);
  analogWrite(EN_B, velocidad);

  delay(3000);

  // PARAR 1 SEGUNDO
  analogWrite(EN_A, 0);
  analogWrite(EN_B, 0);

  delay(1000);
}
```

5.4 Movimientos básicos

Antes de programar el robot siguelíneas, es recomendable probar sus movimientos básicos. Para ello, podemos crear programas sencillos que hagan avanzar el robot, detenerlo, retroceder y girar hacia ambos lados.

El movimiento del robot depende del sentido de giro de cada motor. Si los dos motores giran en el sentido adecuado, el robot avanza. Si ambos giran en sentido contrario, el robot retrocede. Para girar, se puede detener un motor y mantener el otro en movimiento, o hacer que cada motor gire en un sentido diferente.

Estas pruebas permiten comprobar que los motores están conectados correctamente y ajustar la velocidad antes de trabajar con los sensores.

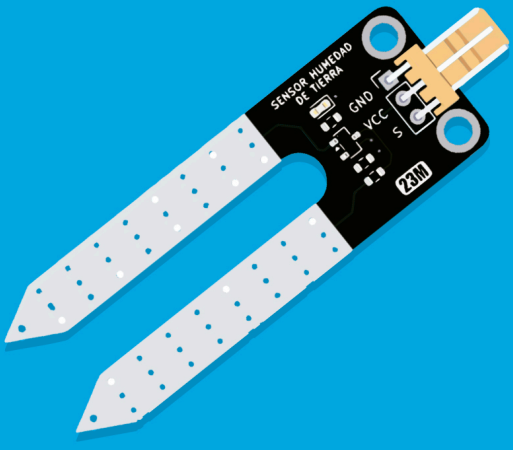
5.5 Robot siguelíneas básico

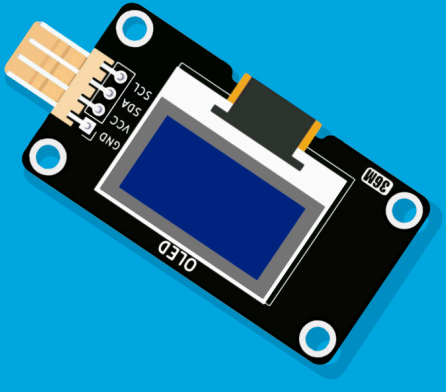
El robot siguelíneas utiliza dos sensores situados en la parte inferior del chasis para detectar la línea negra del campo de trabajo. En este módulo, una superficie blanca devuelve el valor 1 y una línea negra devuelve el valor 0.

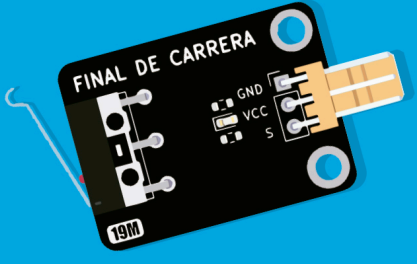
Cuando los dos sensores detectan blanco, el robot avanza. Si el sensor izquierdo detecta negro, el motor izquierdo se detiene y el motor derecho sigue avanzando, corrigiendo la trayectoria. Si el sensor derecho detecta negro, se detiene el motor derecho y avanza el motor izquierdo.

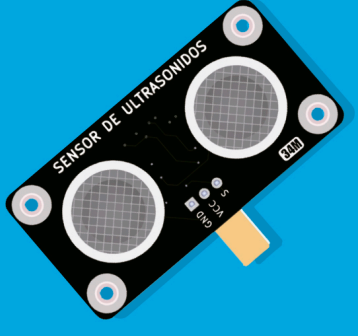
Este comportamiento se repite continuamente dentro del programa. De esta forma, el robot realiza pequeñas correcciones mientras avanza y consigue seguir el recorrido marcado por la línea negra.

```
#define DIR_A 4    // Controla la dirección del motor A
#define EN_A 5     // Controla la velocidad del motor A
#define DIR_B 7    // Controla la dirección del motor B
#define EN_B 6     // Controla la velocidad del motor B
#define SENSOR_IZQUIERDO 3
#define SENSOR_DERECHO 12
int velocidad = 200;
void setup() {
  pinMode(DIR_A, OUTPUT);
  pinMode(EN_A, OUTPUT);
  pinMode(DIR_B, OUTPUT);
  pinMode(EN_B, OUTPUT);
  pinMode(SENSOR_IZQUIERDO, INPUT);
  pinMode(SENSOR_DERECHO, INPUT);
  // Los dos motores preparados para avanzar
  digitalWrite(DIR_A, HIGH);
  digitalWrite(DIR_B, HIGH);
}
void loop() {
  // El sensor derecho controla el motor B
  if (digitalRead(SENSOR_DERECHO) == 0) {
    analogWrite(EN_B, 0);
  } else {
    analogWrite(EN_B, velocidad);
  }
  // El sensor izquierdo controla el motor A
  if (digitalRead(SENSOR_IZQUIERDO) == 0) {
    analogWrite(EN_A, 0);
  } else {
    analogWrite(EN_A, velocidad);
  }
}
```


SENSOR HUMEDAD DE TIERRA
GND
VCC
S
23M


GND
VCC
SDA
SCL
OLED
18B


FINAL DE CARRERA
GND
VCC
S
19M


SENSOR DE ULTRASONIDOS
GND
VCC
S
24M