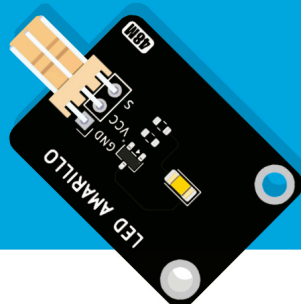
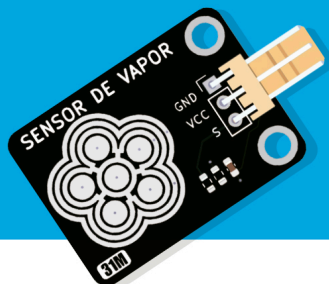
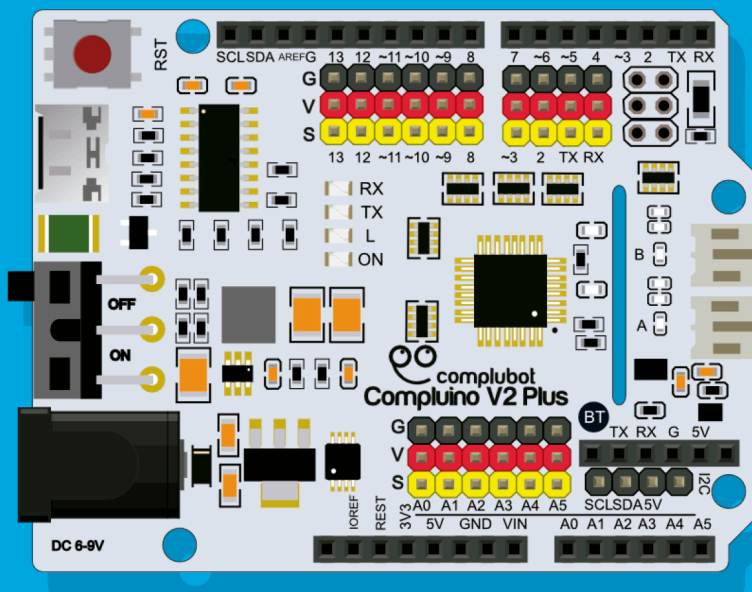
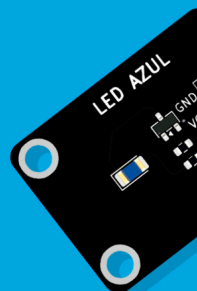


GUÍA DE INICIACIÓN

KIT DE DOMÓTICA Y CIENCIAS CON COMPLUINO



1- Kit de Robótica para Compuino

El kit de robótica para Compuino está pensado para poder implementar diferentes proyectos de robótica, así como proyectos STEAM. Con este material podrás realizar numerosas actividades trabajando programación, electricidad y electrónica. El kit se compone de los siguientes componentes:

- | | |
|---|---|
| <ul style="list-style-type: none">1 - Placa Compuino V2 Plus, con control de motores de CC5 - Módulos led de colores (01M,45M,46M,47M y 48M)1 - Módulo zumbador activo (10M)1 - Módulo zumbador pasivo (09M)1 - Módulo motor vibrador (49M)1 - Ventilador1 - Bomba de agua sumergible2 - Miniservos de 180°1 - Módulo pantalla OLED (36M)1 - Módulo relé (30M)3 - Módulo pulsador (44M)1 - Módulo sensor de presión atmosférica (37M)1 - Módulo sensor de temperatura y humedad (38M)1 - Módulo sensor de conductividad (40M)1 - Módulo sensor de vibración (25M)1 - Módulo sensor de inclinación (41M)1 - Módulo sensor magnético (14M)1 - Módulo sensor de sonido (07M)1 - Módulo potenciómetro (08M)1 - Módulo acelerómetro (43M)2 - Módulo sensor PIR (17M) | <ul style="list-style-type: none">1 - Módulo sensor de vapor (31M)1 - Sonda de temperatura para líquidos1 - Módulo sensor ultravioleta (27M)1 - Módulo sensor LDR (06M)1 - Módulo sensor MQ2 (20M)1 - Módulo sensor de humedad del suelo (23M)1 - Módulo sensor de frecuencia cardíaca (39M)1 - Módulo sensor de nivel de agua (22M)1 - Módulo Bluetooth JDY-31 (V3.0)1 - Placa de prototipado de 400 contactos2 - Pulsador para placa de prototipado1 - Portapilas 5 x AA con conector jack5 - Pilas alcalinas tipo AA10 - Cables macho-hembra de 200mm8 - Cables Header a Molex de 3 pines2 - Cables Header a Molex de 4 pines2 - Cables con cocodrilos1 - Cable USB-C |
|---|---|

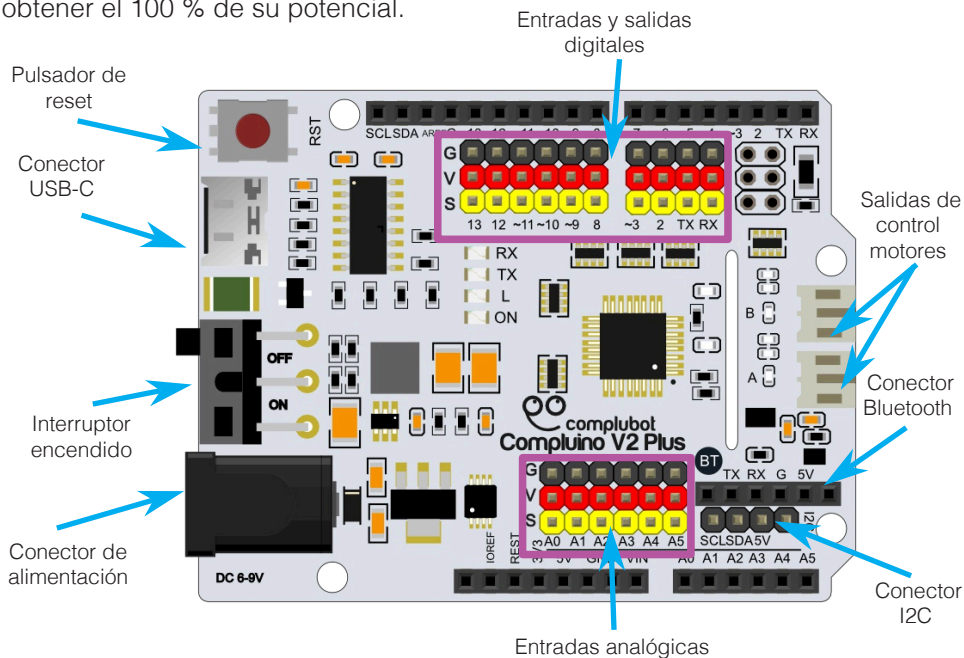
Todas las marcas registradas y nombres comerciales aquí recogidos son propiedad de sus legítimos dueños.



Última versión de esta guía en formato digital

2 - Cómo usar la placa Compluino

La placa Compluino V2 Plus 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.

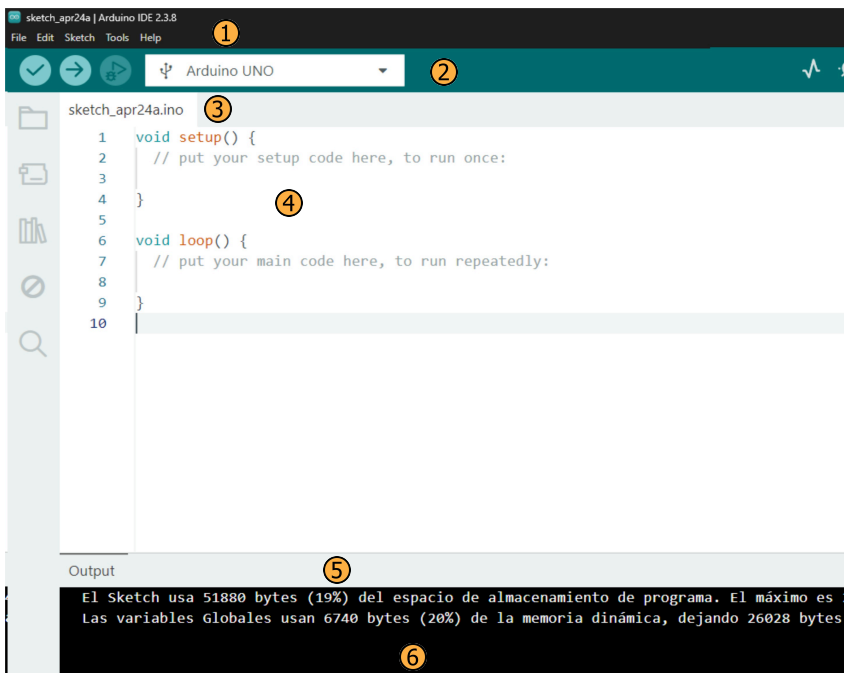
3 - El entorno de programación de Arduino

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

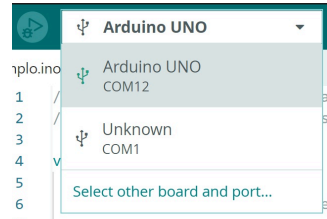
- ① **Menú textual.** Acceso a todas las funcionalidades del IDE.
- ② **Menú de acceso rápido.** Acceso a las funciones más habituales.
- ③ **Pestañas de programa.** Cuando dividimos el programa en varios archivos.
- ④ **Área de programación.** Editor donde escribir los programas.
- ⑤ **Ventana de mensajes.**
- ⑥ **Barra de estado.**



Robótica con Compluino

3.1 Configuración de la placa y del puerto de comunicaciones

Para usar el IDE de Arduino debemos configurar este tipo de placa así como el puerto en el que está conectada. Una vez conectado el cable USB-C, puedes seleccionar la placa y el COM (puerto serie virtual) desde el desplegable del Menú de acceso rápido.



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:

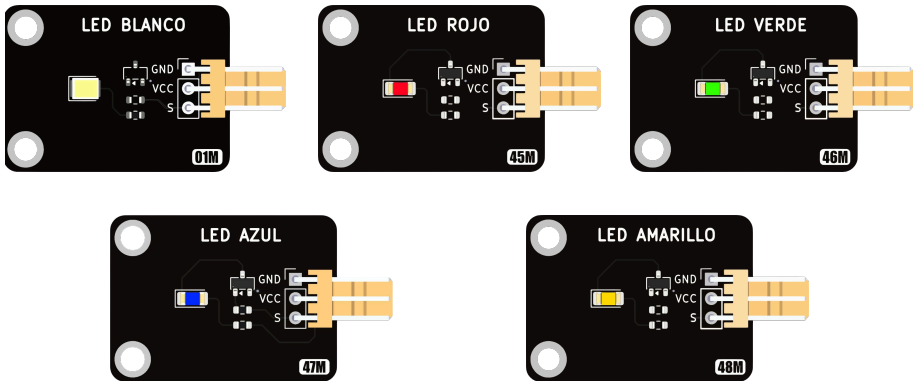
- ① **Definiciones:** al inicio de un programa se describe en qué consiste y se definen las constantes, las variables globales y las bibliotecas que operarán en él.
- ② **void setup():** es la función de inicialización o configuración. Incluye las funciones de configuración del programa y las acciones que deban ejecutarse una sola vez y al comienzo del programa.
- ③ **void loop():** es la función principal y actúa como un bucle infinito. 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 Módulos led de colores (01M,45M,46M,47M y 48M)

El módulo led es un actuador, es decir, recibe una señal de control de la placa Compluino para funcionar. Este módulo incluye un dispositivo electrónico (led) que emite luz visible cuando se activa. En este kit tienes módulos led de 5 colores diferentes:



4.1.1 Control del módulo led desde la placa Compluino

Este módulo led se controla desde una salida digital de la placa Compluino: 2, 3, 8, 9, 10, 12 o 13.

De esta forma el led se enciende cuando activamos la salida digital (HIGH) y se apaga cuando la desactivamos (LOW).

Este módulo es compatible con las salidas de control de brillo (PWM) que tienes en los pines 3, 9, 10 y 11.

Todos los módulos led se conectan y controlan de la misma manera, la única diferencia entre ellos es el color de la luz emitida.

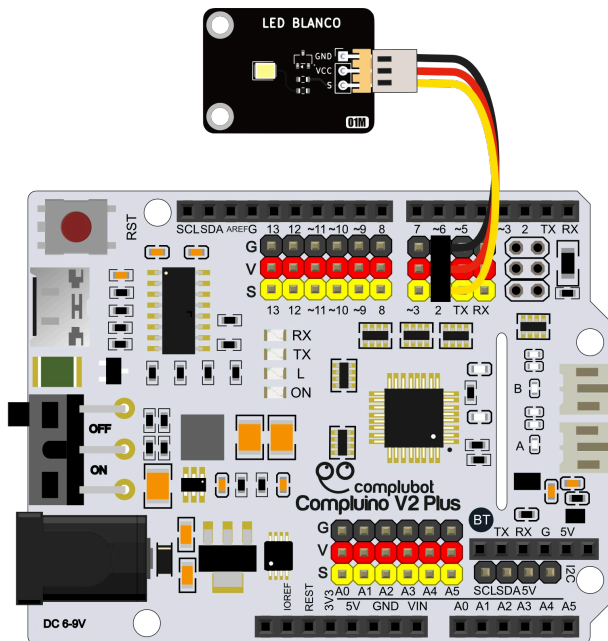
4.1.2 Conexión módulo led a la placa Compluino

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).

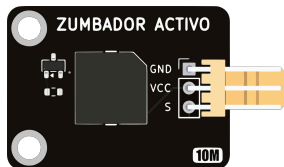
Robótica con Compluino



4.1.3 Programa de ejemplo para el control del módulo led

```
// =====  
// Parpadeo de un módulo LED  
// El LED conectado al pin 2 se enciende  
// y se apaga cada segundo.  
// =====  
  
#define LED 2 // LED conectado al pin digital 2  
  
void setup() {  
  // Configuramos el pin del LED como salida  
  pinMode(LED, OUTPUT);  
}  
  
void loop() {  
  // Encendemos el LED  
  digitalWrite(LED, HIGH);  
  delay(1000);  
  
  // Apagamos el LED  
  digitalWrite(LED, LOW);  
  delay(1000);  
}
```

4.2 Módulo zumbador activo (10M)



Este módulo es un actuador acústico que dispone de un zumbador activo para emitir sonidos de una frecuencia fija, típicamente de unos 2 KHz.

Este tipo de módulos son la forma más sencilla de emitir sonidos sencillos, como señales de aviso, de una forma programada desde la placa Compluino.

Atención: no usar este módulo zumbador activo para generar notas musicales o melodías. En caso de hacerlo, se mezclaría la frecuencia propia del zumbador con la frecuencia que intentemos generar, produciendo sonidos no deseados.

Para generar notas musicales y melodías tenemos que usar el zumbador pasivo (09M).

4.2.1 Control del módulo zumbador activo desde la placa Compluino

Este módulo zumbador activo se controla desde una salida digital de la placa Compluino: 2, 3, 8, 9, 10, 12 o 13.

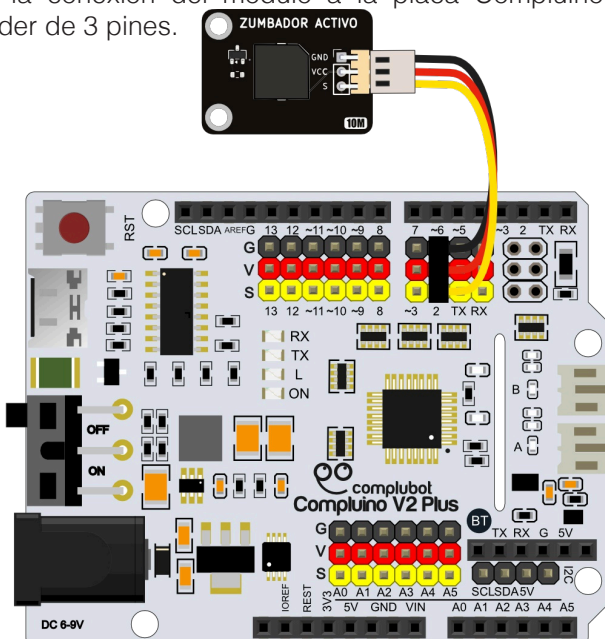
El zumbador emite sonido cuando activamos la salida digital (HIGH) y deja de sonar cuando la desactivamos (LOW).

4.2.2 Conexión módulo zumbador activo 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.

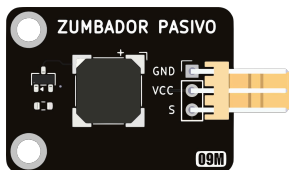
En la placa Compluino, es importante asegurarse de que el cable negro coincida con el terminal negativo (GND).



4.2.3 Programa de ejemplo para el zumbador activo

```
// =====  
// Ejemplo con zumbador activo  
// =====  
// Este programa hace sonar el zumbador 5 veces.  
// Cada pitido dura 200 milisegundos.  
// Entre un pitido y otro hay una pequeña pausa.  
// Después de los 5 pitidos, el programa espera  
// 5 segundos antes de repetir la secuencia.  
// =====  
#define ZUM 2    // Zumbador activo conectado al pin digital 2  
  
void setup() {  
    // Configuramos el pin del zumbador como salida  
    pinMode(ZUM, OUTPUT);  
}  
  
void loop() {  
    // Repetimos el pitido 5 veces  
    for (int i = 0; i < 5; i++) {  
        digitalWrite(ZUM, HIGH); // Encendemos el zumbador  
        delay(200);               // El sonido dura 200 ms  
        digitalWrite(ZUM, LOW);  // Apagamos el zumbador  
        delay(200);              // Pausa de 200 ms entre pitidos  
    }  
    // Cuando termina la secuencia de 5 pitidos,  
    // el zumbador permanece en silencio durante 5 segundos  
    delay(5000);  
}
```

4.3 Módulo zumbador pasivo (09M)



Un zumbador pasivo es en esencia un altavoz, esto es, un actuador electromecánico capaz de reproducir sonidos de distintas frecuencias.

En el caso de este módulo el zumbador pasivo dispone además de un pequeño amplificador para convertir en audibles las señales eléctricas generadas por la placa Compluino.

4.3.1 Control del módulo zumbador pasivo desde la placa Compluino

Este módulo zumbador pasivo se controla desde una salida digital de la placa Compluino: 2, 3, 8, 9, 10, 12 o 13.

Para controlarlo se usa la función:

```
tone(pin, frecuencia)
```

Esta función genera una onda cuadrada al 50 % en el pin seleccionado y de la frecuencia indicada.

La función tone configura de forma automática el pin indicado como salida digital, así que no es necesario hacerlo previamente en el programa.

En la siguiente tabla se muestra la correspondencia entre las distintas notas musicales de la cuarta octava y la frecuencia que hay que emitir para producirlas.

Nota	DO	RE	MI	FA	SOL	LA	SI
Cuarta octava							
Frecuencia (Hz)	262	294	330	349	392	440	494

Tabla de equivalencias entre las notas principales de la cuarta octava y sus frecuencias correspondientes.

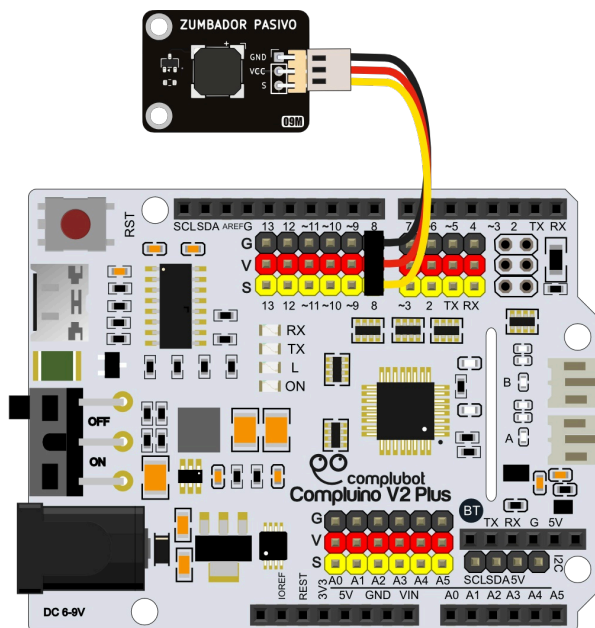
4.3.2 Conexión zumbador pasivo a la placa Compluino

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).

Robótica con Compluino

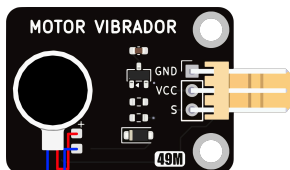


4.3.3 Programa de ejemplo para el zumbador pasivo

Este programa hace sonar las notas musicales DO, RE y MI, de forma repetida, en el zumbador pasivo conectado a la salida 8.

```
#define ZUM 8 // Zumbador pasivo conectado al pin 8
void setup() {
}
void loop() {
  tone(ZUM, 262); // Frecuencia de DO
  delay(400);     // Duración de la nota
  tone(ZUM, 294); // Frecuencia de RE
  delay(400);
  tone(ZUM, 330); // Frecuencia de MI
  delay(400);
  noTone(ZUM); // Parar el sonido
  delay(1000);
}
```


4.4 Módulo motor vibrador (49M)



Este módulo dispone de un pequeño motor que genera una vibración cuando se alimenta de forma correcta. Dispositivos similares a este tienen los mandos a distancia o los teléfonos móviles para interactuar con el usuario.

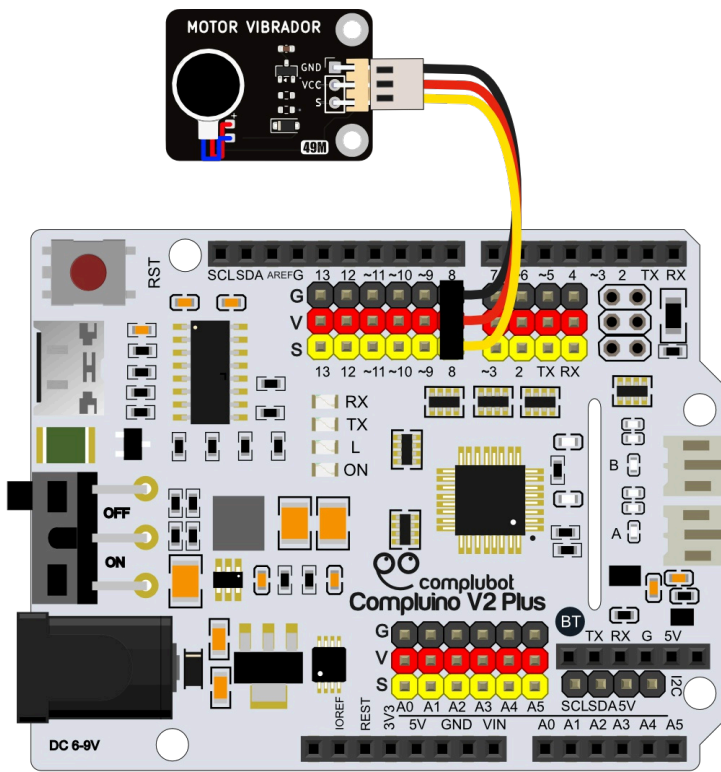
4.4.1 Control del módulo motor vibrador desde la placa Compluino

El módulo motor vibrador es un tipo de actuador que podemos controlar directamente con una salida digital. En el caso de la placa Compluino podemos usar uno de los pines 2, 3, 8, 9, 10, 12 o 13.

4.4.2 Conexión del motor vibrador 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).



4.4.3 Programa de ejemplo para el motor vibrador

El programa de este ejemplo hace que el motor de este módulo realice dos vibraciones cortas y una larga, de forma cíclica.

```
#define VIBRADOR 8 // Motor vibrador en el pin 8
void setup() {
  pinMode(VIBRADOR, OUTPUT);
}
void loop() {
  // --- Dos vibraciones rápidas ---
  digitalWrite(VIBRADOR, HIGH);
  delay(150);
  digitalWrite(VIBRADOR, LOW);
  delay(100);
  digitalWrite(VIBRADOR, HIGH);
  delay(150);
  digitalWrite(VIBRADOR, LOW);
  delay(200);
  // --- Vibración más larga ---
  digitalWrite(VIBRADOR, HIGH);
  delay(600);
  digitalWrite(VIBRADOR, LOW);
  delay(1000); // Pausa antes de repetir
}
```

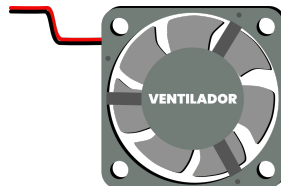
4.4.4 Observaciones sobre el control de motores con salidas digitales

Puede parecer que, igual que ocurre con este módulo, es posible controlar el funcionamiento de un pequeño motor con una salida digital. Pero en general esto solo es posible si, como en este caso, el módulo incorpora su propio driver para el control de motores.

De forma general, las salidas digitales solo proporcionan señales de control, para mover un motor, aunque sea pequeño, se necesitan señales de potencia. Esto se consigue con un driver de control de motores, como el que lleva este módulo, o conectando el motor directamente a las salidas de control de potencia (A y B) de la placa Compluino.

4.5 Ventilador

Se trata de un ventilador de 5 V de CC que podemos usar como sistema de refrigeración controlado.

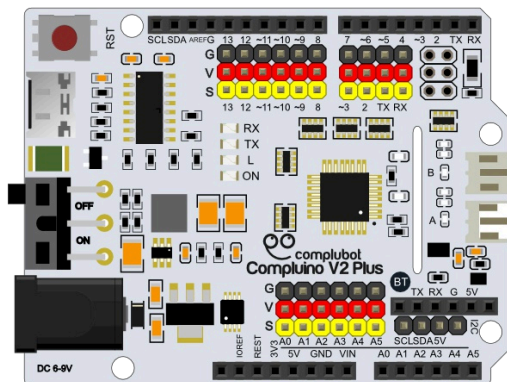


4.5.1 Control del ventilador con la placa Compluino

El ventilador se controla como cualquier otro motor que podemos conectar al puerto A o B de la placa Compluino.

	Dirección	Velocidad
Motor A	D4	D5
Motor B	D7	D6

4.5.2 Conexión del ventilador a la placa Compluino



El ventilador se conecta a la placa Compluino a una de las dos salidas de control de motores. En el ejemplo a la salida A.



4.5.3 Programa de ejemplo para el ventilador

```
#define DIR_M1 4 // Pin de dirección del motor
#define PWM_M1 5 // Pin de velocidad (PWM)

void setup() {
  pinMode(DIR_M1, OUTPUT); // Configurar pin de dirección
  pinMode(PWM_M1, OUTPUT); // Configurar pin de velocidad
}

void loop() {
  // Encender ventilador
  digitalWrite(DIR_M1, HIGH); // Dirección de giro
  analogWrite(PWM_M1, 200); // Velocidad del motor (0 - 255)
  delay(5000); // Funciona durante 5 segundos
  // Apagar ventilador
  analogWrite(PWM_M1, 0); // Velocidad 0 = motor parado
  delay(5000); // Espera 5 segundos
}
```

4.6 Bomba de agua sumergible

Esta bomba de agua tiene un pequeño motor de corriente continua que impulsa agua a través de la salida. Puede utilizarse para mover líquidos.

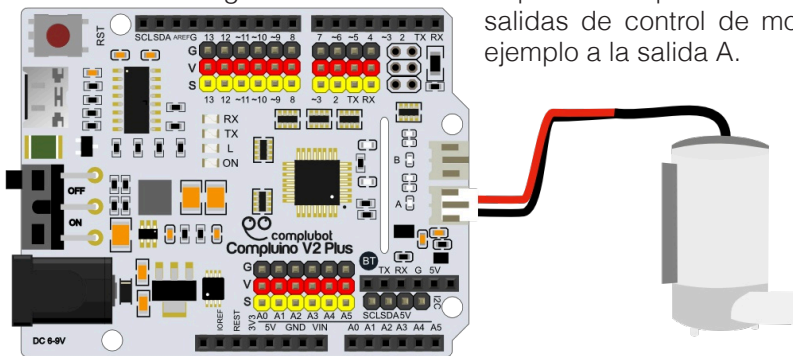
4.6.1 Control de la bomba sumergible con la placa Compluino

Esta bomba se controla como cualquier otro motor que podemos conectar al puerto A o B de la placa Compluino.

	Dirección	Velocidad
Motor A	D4	D5
Motor B	D7	D6

4.6.2 Conexión de la bomba de agua a la placa Compluino

La bomba sumergible se conecta a la placa Compluino a una de las dos salidas de control de motores. En este ejemplo a la salida A.



4.6.3 Programa de ejemplo para la bomba de agua

```
#define DIR_BOMBA    4    // Dirección de la bomba
#define VEL_BOMBA    5    // Velocidad (PWM)

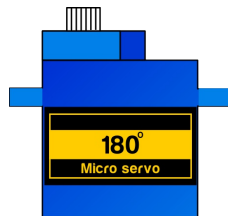
void setup() {
    pinMode(DIR_BOMBA, OUTPUT);
    pinMode(VEL_BOMBA, OUTPUT);
}

void loop() {
    // Encendemos la bomba
    digitalWrite(DIR_BOMBA, HIGH);    // Sentido de giro
    analogWrite(VEL_BOMBA, 255);    // Máxima velocidad
    delay(3000);    // 3 segundos

    // Apagamos la bomba
    analogWrite(VEL_BOMBA, 0);    // Velocidad 0
    delay(5000);    // 5 segundos
}
```

4.7 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°.

4.7.1 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, en este ejemplo:

```
#define SERVO 9  
Servo servo;
```

A continuación establecemos la vinculación del servo con:

```
servo.attach(SERVO);
```

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

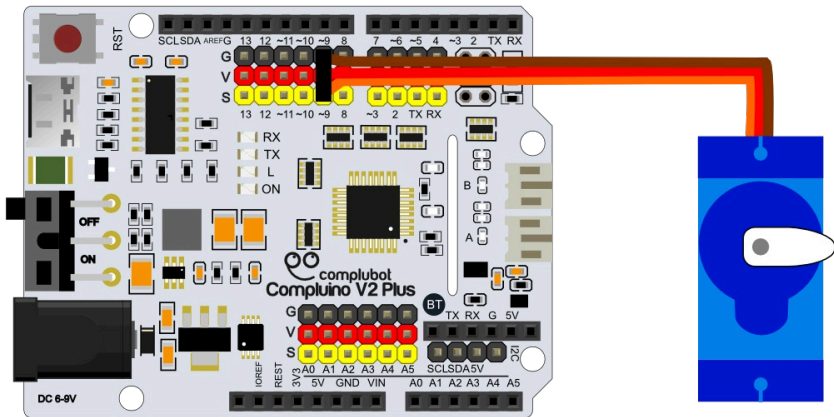
4.7.2 Conexión del miniservo de 180° a la placa Compluino

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.

Robótica con Compluino



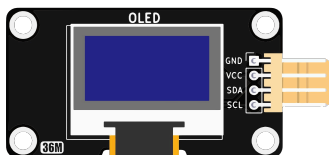
4.7.3 Programa de ejemplo para el miniservo de 180°

```
#include <Servo.h> // Librería necesaria para controlar servomotores
#define SERVO 9     // El servo está conectado al pin 9
Servo servo;        // Creamos un objeto llamado "servo" para manejar el motor
int posicion = 0;   // Variable donde guardaremos el ángulo del servo
void setup() {
  servo.attach(SERVO); // Conectamos el objeto servo al pin definido
}
void loop() {
  // El servo se moverá desde 0° hasta 180° poco a poco
  for (posicion = 0; posicion <= 180; posicion++) {
    servo.write(posicion); // Enviamos al servo el ángulo actual
    delay(10);             // Pequeña pausa para que el movimiento sea suave
  }
  // Ahora el servo vuelve desde 180° hasta 0°
  for (posicion = 180; posicion >= 0; posicion--) {
    servo.write(posicion); // Movemos el servo al ángulo indicado
    delay(10);
  }
}
```

4.7.4 Conexión y alimentación de varios servos a la placa Compluino

Es posible conectar varios servos a la placa Compluino, en teoría a cada salida digital, pero dado el consumo de energía de estos dispositivos, 4 sería un límite razonable. En cualquier caso para alimentar más de un servo necesitaremos poner el portapilas de 5 x AA incluido en el kit. De esta manera además podemos controlar la puesta en marcha de nuestro proyecto con el interruptor de la placa Compluino.

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

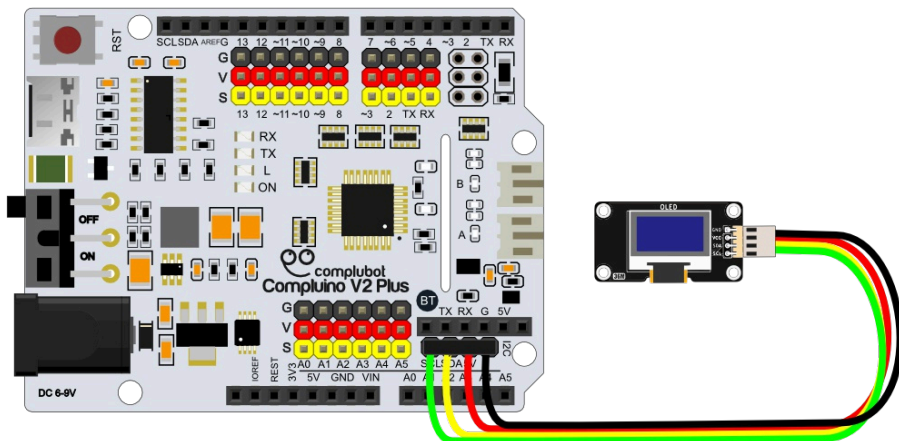
Este módulo OLED utiliza un controlador tipo SSD1306 que se comunica con la placa Compluino mediante un bus del tipo I2C. En este bus cada dispositivo tiene una dirección y la de este módulo es la 0x3C.

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

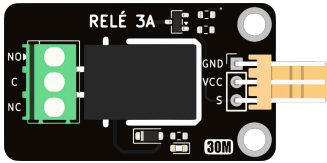
En esta imagen puedes ver la conexión del módulo a la placa Compluino mediante un cable Molex-Header de 4 pines. En el módulo solo existe una posición posible de conexión. En la placa Compluino, es importante asegurarse de que el cable negro coincida con el terminal negativo (GND).



4.8.3 Programa de ejemplo para la pantalla OLED

```
// =====  
// Programa de ejemplo para pantalla OLED  
// Muestra "Hola" y "Adios" alternando cada 2 segundos.  
// La pantalla se comunica mediante I2C.  
// =====  
  
#include <Wire.h>           // Librería necesaria para la 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 OLED  
Adafruit_SSD1306 oled(ANCHO, ALTO, &Wire, -1);  
  
void setup() {  
    // Inicializamos la pantalla OLED  
    oled.begin(SSD1306_SWITCHCAPVCC, OLED_DIR);  
  
    // Establecemos el tamaño del texto  
    oled.setTextSize(2);  
  
    // Indicamos que el texto se muestre en blanco  
    oled.setTextColor(SSD1306_WHITE);  
}  
  
void loop() {  
    oled.clearDisplay(); // Borramos la pantalla antes de escribir el primer mensaje  
    oled.setCursor(15, 25); // Colocamos el cursor en la posición donde aparecerá el texto  
    oled.println("Hola"); // Escribimos el texto "Hola"  
    oled.display();       // Actualizamos la pantalla para mostrar el mensaje  
    delay(2000);          // Esperamos 2 segundos  
  
    oled.clearDisplay(); // Borramos la pantalla antes de escribir el segundo mensaje  
    oled.setCursor(20, 25); // Colocamos el cursor en una nueva posición  
    oled.println("Adios"); // Escribimos el texto "Adios"  
    oled.display();       // Actualizamos la pantalla para mostrar el mensaje  
    delay(2000);          // Esperamos 2 segundos  
}
```

4.9 Módulo relé (30M)

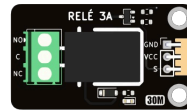


Un relé es un interruptor electromecánico que podemos controlar digitalmente para controlar el encendido y apagado de dispositivos de potencia aislando de la propia placa Compuino. Los contactos de este relé pueden controlar una intensidad máxima de 3 A y un

voltaje máximo de 250 V AC. Por ello es importante manejarlo con la debida seguridad eléctrica.

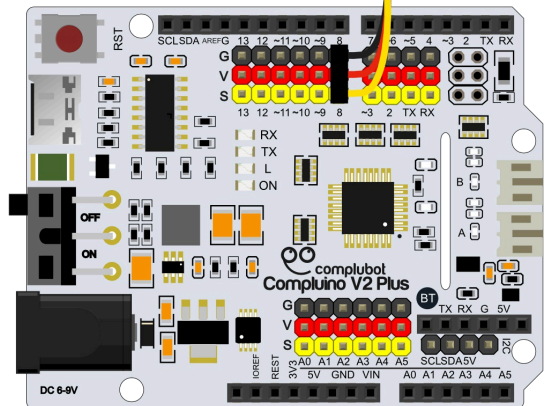
4.9.1 Control del módulo relé con la placa Compuino

Además del relé, este módulo incluye un pequeño driver para manejarlo, de esta manera el encendido y apagado del relé se puede realizar directamente desde una salida digital 2, 3, 8, 9, 10, 12 o 13.



4.9.2 Conexión relé a la placa

En esta imagen puedes ver la conexión del módulo a la placa Compuino mediante un cable Molex-Header de 3 pines. En el módulo solo existe una posición posible de conexión. En la placa Compuino, es importante asegurarse de que el cable negro coincida con el terminal negativo (GND).



4.9.3 Programa de ejemplo para el relé

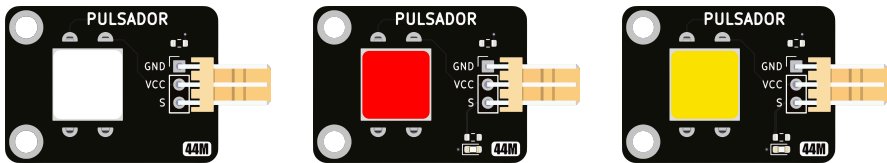
```
#define RELE 8 // Relé de 3V conectado al pin digital 8
int estado = 0;
void setup() {
  pinMode(RELE, OUTPUT); // Configuramos el relé como salida
}
void loop() {
  digitalWrite(RELE, HIGH); // Activamos el relé
  delay(1000);
  digitalWrite(RELE, LOW); // Desactivamos el relé
  delay(1000);
}
```

4.10 Módulo pulsador (44M)

Un pulsador es un tipo de interruptor momentáneo que cierra el circuito mientras se aplica una fuerza sobre él. Al conectar este módulo a la placa Compluino, podemos detectar (leer) si el pulsador está pulsado o no. Este dispositivo es ideal para controlar acciones, iniciar programas o activar luces o zumbadores.

Este módulo incorpora, además, un led de estado que indica si el pulsador se ha presionado o no de forma local, con tan solo conectar este módulo a la placa Compluino.

En el kit puedes encontrar 3 modelos de pulsador distintos, cada uno con una tapa de color distinto: blanco, rojo y amarillo.



4.10.1 Lectura del módulo pulsador desde la placa Compluino

Este módulo es un sensor digital, por lo tanto la forma adecuada de leerlo es conectarlo a uno de los pines de entrada/salida digital (2, 3, 8, 9, 10, 12 o 13) y configurarlo como entrada digital, con la función `pinMode` desde el programa.

Podemos implementar dos maneras de leer este tipo de pulsadores:

- Esperar hasta que el pulsador cambie de estado
- Comprobar de forma periódica el estado del pulsador

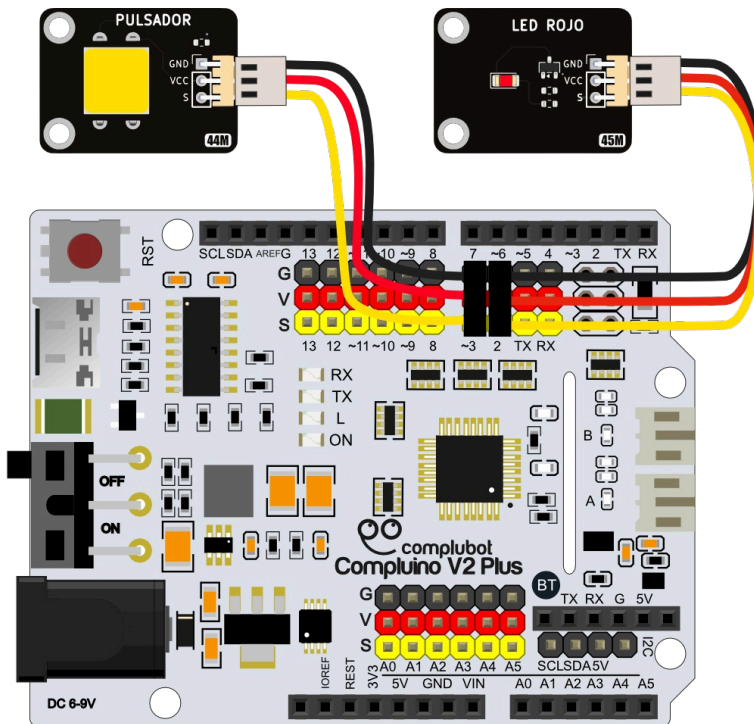
4.10.2 Conexión de los pulsadores a la placa Compluino

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).

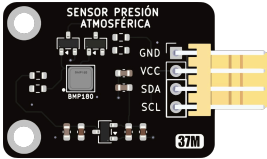
Guía de actividades



4.10.3 Programa de ejemplo para los pulsadores

```
#define LED 2 // LED conectado al pin digital 2
#define PULSADOR 3 // Pulsador conectado 3
void setup() {
  pinMode(LED, OUTPUT); // Configuramos el LED como salida
  pinMode(PULSADOR, INPUT); // Pulsador como entrada
}
void loop() {
  // Si el pulsador está pulsado, encendemos el LED
  if (digitalRead (PULSADOR) == HIGH) {
    digitalWrite(LED, HIGH);
  }
  else {
    // Si NO está pulsado, apagamos el LED
    digitalWrite(LED, LOW);
  }
}
```

4.11 Módulo sensor de presión atmosférica (37M)



Este módulo incluye un sensor digital del tipo BMP180 que es capaz de medir la presión atmosférica y la temperatura del aire. Este tipo de módulo se puede usar para realizar proyectos para estudiar el clima, calcular altitud aproximada y registrar cambios ambientales. Capacidades de medición:

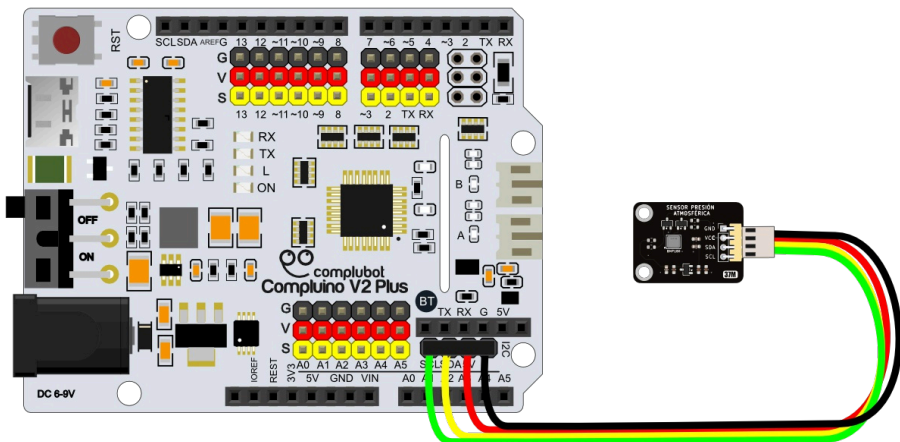
- Presión atmosférica: Mide en un rango de 300 a 1100 hPa. Gracias a su alta resolución (hasta 0,16 Pa), permite estimar la altitud con una precisión de ± 1 metro
- Temperatura: Opera en un rango de $-40\text{ }^{\circ}\text{C}$ a $+85\text{ }^{\circ}\text{C}$ con una precisión típica de $\pm 1,0\text{ }^{\circ}\text{C}$.

4.11.1 Lectura del módulo sensor de presión atmosférica con la placa Compluino

Este módulo se comunica con la placa Compluino mediante una comunicación I2C. En este bus cada dispositivo tiene su dirección y la de este tipo de módulos es la 0x76. Para programar este sensor necesitamos añadir dos librerías externas: Adafruit_Sensor.h y Adafruit_BMP280.h.

4.11.2 Conexión del sensor de presión atmosférica a la placa Compluino

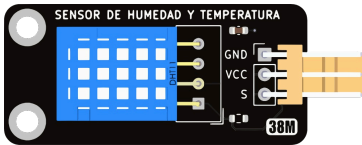
En esta imagen puedes ver la conexión del módulo a la placa Compluino mediante un cable Molex-Header de 4 pines. En el módulo solo existe una posición posible de conexión. En la placa Compluino, es importante asegurarse de que el cable negro coincida con el terminal negativo (GND).



4.11.3 Programa de ejemplo para el sensor de presión atmosférica

```
// =====  
// Lectura de temperatura y presión  
// con sensor barométrico BMP280  
// =====  
// El sensor BMP280 se comunica mediante I2C.  
// Permite medir la temperatura ambiente  
// y la presión atmosférica.  
// Los datos se muestran en el monitor serie.  
// =====  
  
#include <Wire.h>           // Librería para comunicación I2C  
#include <Adafruit_Sensor.h> // Librería base para sensores Adafruit  
#include <Adafruit_BMP280.h> // Librería específica del sensor BMP280  
  
Adafruit_BMP280 bmp; // Creamos el objeto del sensor BMP280  
  
int temperatura = 0; // Variable para guardar la temperatura en grados Celsius  
int presion = 0;     // Variable para guardar la presión atmosférica en hPa  
  
void setup() {  
  Serial.begin(9600); // Iniciamos la comunicación con el monitor serie  
  
  Wire.begin(); // Iniciamos el bus I2C para comunicarnos con el sensor  
  
  bmp.begin(0x76); // Inicializamos el sensor BMP280 con su dirección I2C  
  
  Serial.println("Sensor barometrico listo"); // Mensaje de confirmación  
  Serial.println("Iniciando lectura de datos..."); // Aviso antes de comenzar  
  Serial.println("-----"); // Línea separadora  
}  
  
void loop() {  
  float t = bmp.readTemperature(); // Leemos la temperatura en grados Celsius  
  
  float p = bmp.readPressure(); // Leemos la presión atmosférica en Pascales  
  
  temperatura = (int)t; // Convertimos la temperatura a número entero  
  
  presion = (int)(p / 100); // Convertimos la presión de Pascales a hectoPascales  
  
  Serial.print("Temperatura: "); // Mostramos el texto de temperatura  
  Serial.print(temperatura);     // Mostramos el valor de la temperatura  
  Serial.println(" C");          // Mostramos la unidad de medida  
  
  Serial.print("Presion: "); // Mostramos el texto de presión  
  Serial.print(presion);     // Mostramos el valor de la presión  
  Serial.println(" hPa");    // Mostramos la unidad de presión atmosférica  
  
  Serial.println("-----"); // Separador entre lecturas  
  
  delay(1000); // Esperamos 1 segundo antes de hacer una nueva lectura  
}
```


4.12 Módulo sensor de temperatura y humedad (38M)

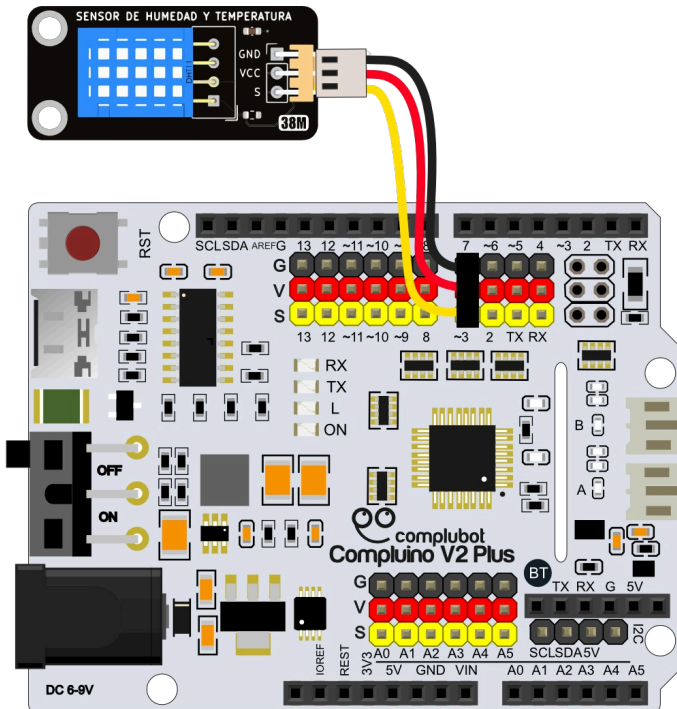


Este módulo utiliza un sensor de temperatura y humedad ambiente del tipo DHT11. Podemos usarlo junto con la placa Compluino para construir estaciones meteorológicas, controlar invernaderos o registrar condiciones ambientales. Este sensor mide temperaturas entre 0 °C y 50 °C (± 2 °C) y humedad relativa entre el 20 % y el 90 % (± 5 %).

4.12.1 Lectura del módulo sensor de temperatura y humedad con la placa Compluino

Este sensor utiliza un protocolo de comunicación propio de un bit. Esto permite establecer la conexión entre el módulo sensor mediante un único puerto digital de la placa Compluino.

Para programarlo se necesita una librería externa llamada DHT.h, hay varias versiones de esta librería, lo mejor es buscar en el gestor de librerías: DHT sensor library, de Adafruit.



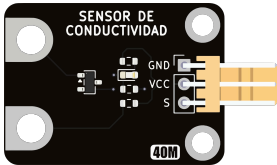
4.12.2 Conexión del sensor de temperatura y humedad a la placa Compuino

En esta imagen puedes ver la conexión del módulo a la placa Compuino mediante un cable Molex-Header de 3 pines. En el módulo solo existe una posición posible de conexión. En la placa Compuino, es importante asegurarse de que el cable negro coincida con el terminal negativo (GND).

4.12.3 Programa de ejemplo para el sensor de temperatura y humedad

```
// =====  
// Lectura de temperatura y humedad  
// con sensor DHT11  
// =====  
// El sensor DHT11 permite medir la temperatura  
// y la humedad relativa del ambiente.  
// En este ejemplo se conecta al pin digital 3  
// y los valores se muestran en el monitor serie.  
// =====  
  
#include <DHT.h> // Librería para controlar el sensor DHT  
  
#define PIN_DHT 3      // Pin donde está conectado el DHT11  
#define TIPO_DHT DHT11 // Tipo de sensor utilizado  
  
DHT dht(PIN_DHT, TIPO_DHT); // Creamos el objeto del sensor  
  
float temperatura; // Variable para guardar la temperatura  
float humedad;     // Variable para guardar la humedad relativa  
  
void setup() {  
    Serial.begin(9600); // Iniciamos la comunicación con el monitor seri  
  
    dht.begin(); // Inicializamos el sensor DHT11  
  
    Serial.println("Sensor DHT11 listo"); // Mensaje de confirmación  
    Serial.println("Iniciando lectura de temperatura y humedad...");  
    Serial.println("-----");  
}  
  
void loop() {  
    humedad = dht.readHumidity(); // Leemos la humedad relativa en %  
    // Leemos la temperatura en grados Celsius  
    temperatura = dht.readTemperature();  
  
    Serial.print("Temperatura: "); // Mostramos el texto de temperatura  
    Serial.print(temperatura);    // Mostramos el valor leído  
    Serial.println(" C");        // Mostramos la unidad de medida  
  
    Serial.print("Humedad: "); // Mostramos el texto de humedad  
    Serial.print(humedad);     // Mostramos el valor leído  
    Serial.println(" %");      // Mostramos la unidad de medida  
  
    Serial.println("-----"); // Separador entre lecturas  
    // Esperamos 2 segundos antes de realizar una nueva lectura  
    delay(2000);  
}
```

4.13 Módulo sensor de conductividad (40M)



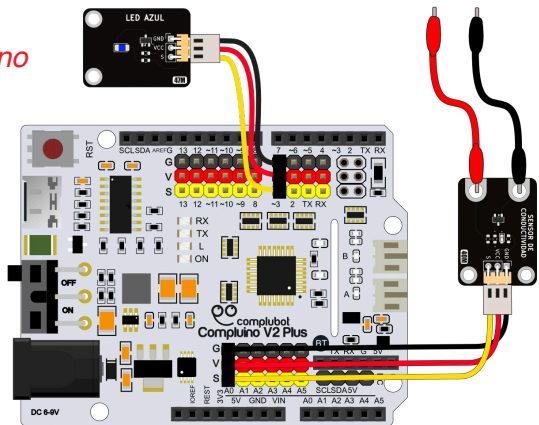
Este es un sensor analógico que mide la conductividad de diferentes materiales. Con él podemos establecer si un determinado material es conductor e incluso qué materiales son más conductores que otros. Siempre se tienen que realizar las medidas sobre materiales libres de tensión, esto es, que no tengan ningún tipo de alimentación eléctrica.

4.13.1 Lectura del módulo sensor de conductividad

Al tratarse de un sensor analógico, la lectura de este sensor se realiza de forma directa desde una de las entradas analógicas de la placa Compluino.

4.13.2 Conexión del sensor de conductividad a la placa Compluino

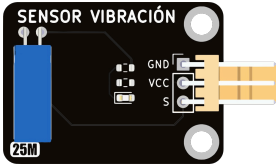
En esta imagen puedes ver la conexión del módulo a la placa Compluino mediante un cable Molex-Header de 3 pines. En el módulo solo existe una posición posible de conexión. En la placa Compluino, es importante asegurarse de que el cable negro coincida con el terminal negativo (GND).



4.13.3 Programa de ejemplo para el sensor de conductividad

```
#define CONDUCTIVIDAD A0 // Sensor de conductividad en A0
#define LED 2 // LED en pin 2
int valor = 0;
int umbral = 600; // Ajustable a otro valor
void setup() {
  pinMode(LED, OUTPUT); // LED como salida
  Serial.begin(9600); // Monitor serie
}
void loop() {
  valor = analogRead(CONDUCTIVIDAD); // Leemos la conductividad (0-1023)
  Serial.print("Conductividad: ");
  Serial.println(valor);
  if (valor > umbral) {
    digitalWrite(LED, HIGH); // Si supera el umbral → LED encendido
  }
  else {
    digitalWrite(LED, LOW); // Si no → LED apagado
  }
  delay(100);
}
```

4.14 Módulo sensor de vibración (25M)



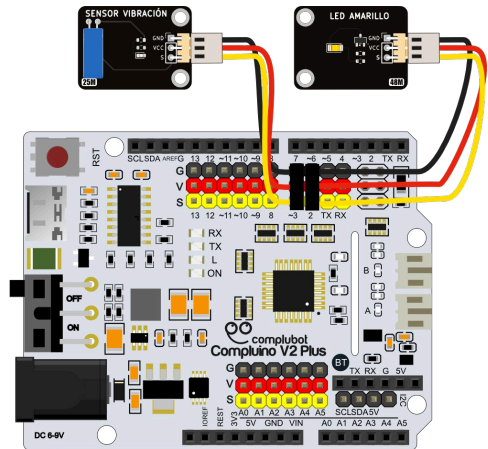
Este sensor de vibración detecta golpes o movimientos bruscos mediante un sensor interno sensible al movimiento. Con la placa Compluino, permite activar luces, alarmas o registrar impactos cuando se agita u ocurre una vibración.

4.14.1 Lectura del sensor de vibración con la placa Compluino

Se trata de un sensor digital que podemos leer en cualquiera de los pines 2, 3, 8, 9, 10, 12 o 13.

4.14.2 Conexión del sensor de vibración a la placa Compluino

En esta imagen puedes ver la conexión del módulo a la placa Compuino mediante un cable Molex-Header de 3 pines. En el módulo solo existe una posición posible de conexión. En la placa Compuino, es importante asegurarse de que el cable negro coincida con el terminal negativo (GND).



4.14.3 Programa de ejemplo para el sensor de vibración

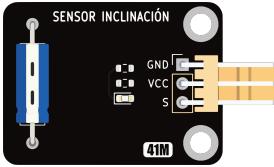
```
#define VIBRACION 3 // Sensor de vibración en pin 3
#define LED_AMAR 2

int valor = 0;

void setup() {
    pinMode(VIBRACION, INPUT); // Sensor como entrada
    pinMode(LED_AMAR, OUTPUT);
}

void loop() {
    valor = digitalRead(VIBRACION); // Leemos la vibración
    if (valor == HIGH) { // Hay vibración → LED encendido
        digitalWrite(LED_AMAR, HIGH);
    }
    else { // No hay vibración → LED apagado
        digitalWrite(LED_AMAR, LOW);
    }
}
```

4.15 Módulo sensor de inclinación (41M)



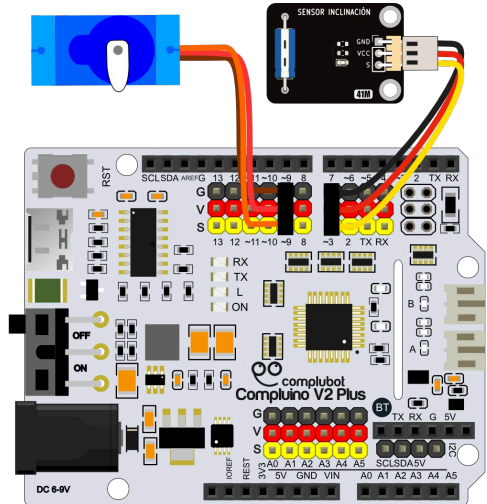
Este sensor de inclinación detecta cambios de posición mediante un pequeño interruptor interno que se activa al inclinarse. Con la placa Compluino puedes responder a movimientos, por ejemplo, girando servos o encendiendo ledes.

4.15.1 Lectura del sensor de inclinación con la placa Compluino

Se trata de un sensor digital que podemos leer en cualquiera de los pines 2, 3, 8, 9, 10, 12 o 13.

4.15.2 Conexión del sensor de inclinación a la placa Compluino

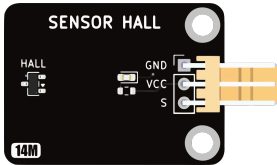
En esta imagen puedes ver la conexión del módulo a la placa Compluino mediante un cable Molex-Header de 3 pines. En el módulo solo existe una posición posible de conexión. En la placa Compluino, es importante asegurarse de que el cable negro coincida con el terminal negativo (GND).



4.15.3 Programa de ejemplo para el sensor de inclinación

```
#include <Servo.h>
#define INCLINACION 3 // Sensor de inclinación
#define SERVO 9 // Servo en pin 9
Servo servo;
int valor = 0;
void setup() {
    pinMode(INCLINACION, INPUT);
    servo.attach(SERVO); // Activamos el servo
}
void loop() {
    valor = digitalRead(INCLINACION); // Leemos el tilt
    if (valor == HIGH) {
        servo.write(180); // Si está inclinado → servo a 180º
    }
    else {
        servo.write(0); // Si está recto → servo a 0º
    }
    delay(50);
}
```

4.16 Módulo sensor Hall (14M)



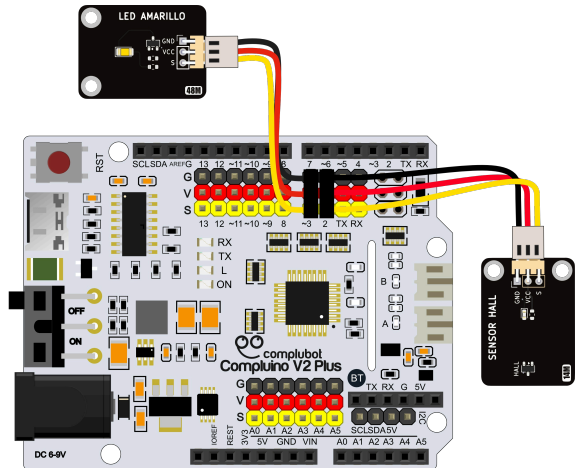
El sensor Hall detecta la presencia de un campo magnético generado por un imán. Este es un tipo de sensor magnético que controla una salida digital que se activa cuando detecta un campo magnético fuerte.

4.16.1 Lectura del sensor Hall con la placa Compluino

Se trata de un sensor digital que podemos leer en cualquiera de los pines 2, 3, 8, 9, 10, 12 o 13.

4.16.2 Conexión del sensor Hall a la placa Compluino

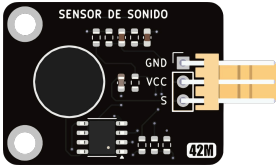
En esta imagen puedes ver la conexión del módulo a la placa Compluino mediante un cable Molex-Header de 3 pines. En el módulo solo existe una posición posible de conexión. En la placa Compluino, es importante asegurarse de que el cable negro coincida con el terminal negativo (GND).



4.16.3 Programa de ejemplo para el sensor Hall

```
#define SENSOR_HALL 2
#define LED_AMAR 3
int estadoSensor = 0; // para guardar el estado del sensor
void setup() {
  pinMode(SENSOR_HALL, INPUT);
  pinMode(LED_AMAR, OUTPUT);
}
void loop() {
  // Leemos el estado del sensor Hall
  estadoSensor = digitalRead(SENSOR_HALL);
  // Si detecta un campo magnético (imán), encendemos el LED
  if (estadoSensor == HIGH) {
    digitalWrite(LED_AMAR, HIGH);
  }
  else {
    digitalWrite(LED_AMAR, LOW);
  }
}
```


4.17 Módulo sensor de sonido (07M)



Un sensor de sonido detecta variaciones en el nivel acústico del entorno mediante un micrófono integrado y un circuito que convierte esas ondas en una señal eléctrica. Con la placa Compluino, permite activar ledes, zumbadores o acciones programadas cuando el sonido supera un determinado umbral configurado

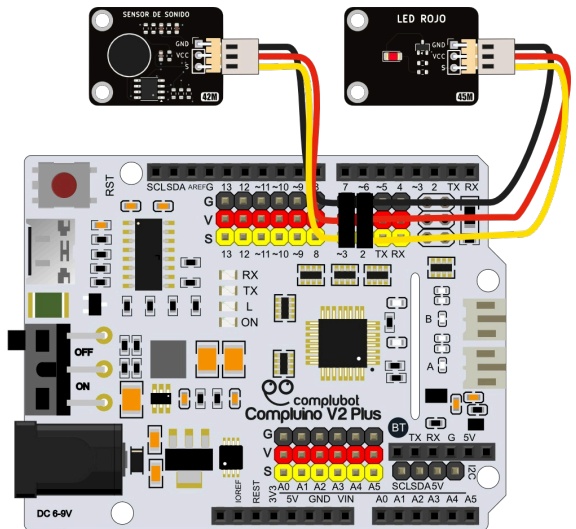
en el código.

4.17.1 Lectura del módulo sensor de sonido con la placa Compluino

Es un sensor digital que podemos leer en los pines 2, 3, 8, 9, 10, 12 o 13.

4.17.2 Conexión del sensor de sonido a la placa Compluino

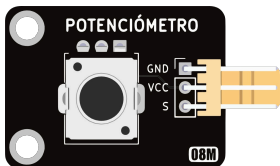
En esta imagen puedes ver la conexión del módulo a la placa Compluino mediante un cable Molex-Header de 3 pines. En el módulo solo existe una posición posible de conexión. En la placa Compluino, es importante asegurarse de que el cable negro coincida con el terminal negativo (GND).



4.17.3 Programa de ejemplo para el sensor de sonido

```
#define SONIDO 3
#define LED_ROJO 2
int valor = 0;
void setup() {
  pinMode(SONIDO, INPUT);
  pinMode(LED_ROJO, OUTPUT);
}
void loop() {
  // Leemos el estado del sensor de sonido
  valor = digitalRead(SONIDO);
  // Si el sensor detecta sonido, encendemos el LED rojo
  if (valor == HIGH) {
    digitalWrite(LED_ROJO, HIGH);
  }
  else {
    digitalWrite(LED_ROJO, LOW);
  }
}
```


4.18 Módulo potenciómetro (08M)



El potenciómetro es un sensor angular que proporciona un valor analógico en función de su posición. Al girar su eje genera una señal de control que se puede usar para variar el nivel de iluminación de un led o la velocidad de giro de un motor.

4.18.1 Lectura del módulo potenciómetro con la placa Compluino

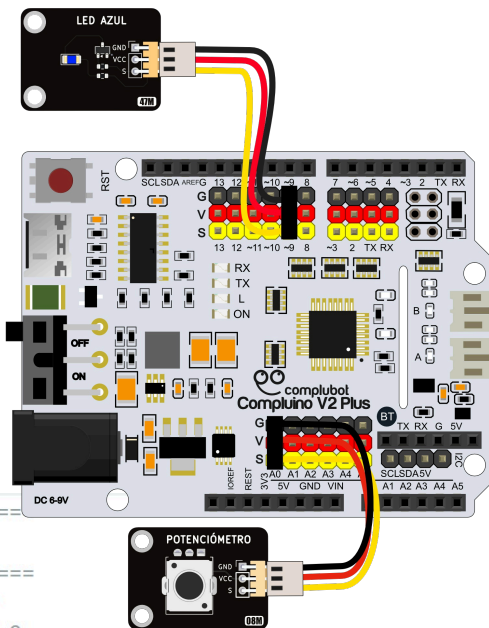
Es un sensor analógico que podemos leer en las entradas A0 a A5

4.18.2 Conexión del potenciómetro a la placa Compluino

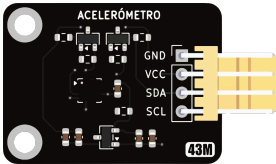
En esta imagen puedes ver la conexión del módulo a la placa Compluino mediante un cable Molex-Header de 3 pines. En el módulo solo existe una posición posible de conexión. En la placa Compluino, es importante asegurarse de que el cable negro coincida con el terminal negativo (GND).

4.18.3 Programa de ejemplo para el potenciómetro

```
// =====  
// Definición de componentes  
// =====  
#define POT A0 // Potenciómetro en A0  
#define LED_AZUL 9 // LED azul en PWM 9  
int valorPot = 0;  
int brillo = 0;  
void setup() {  
  pinMode(LED_AZUL, OUTPUT); // LED azul como salida  
}  
void loop() {  
  valorPot = analogRead(POT);  
  // Leemos el potenciómetro (0-1023)  
  brillo = map(valorPot, 0, 1023, 0, 255);  
  // Convertimos el valor con map()  
  analogWrite(LED_AZUL, brillo);  
  // Ajustamos el brillo  
  delay(10);  
}
```



4.19 Módulo acelerómetro (43M)



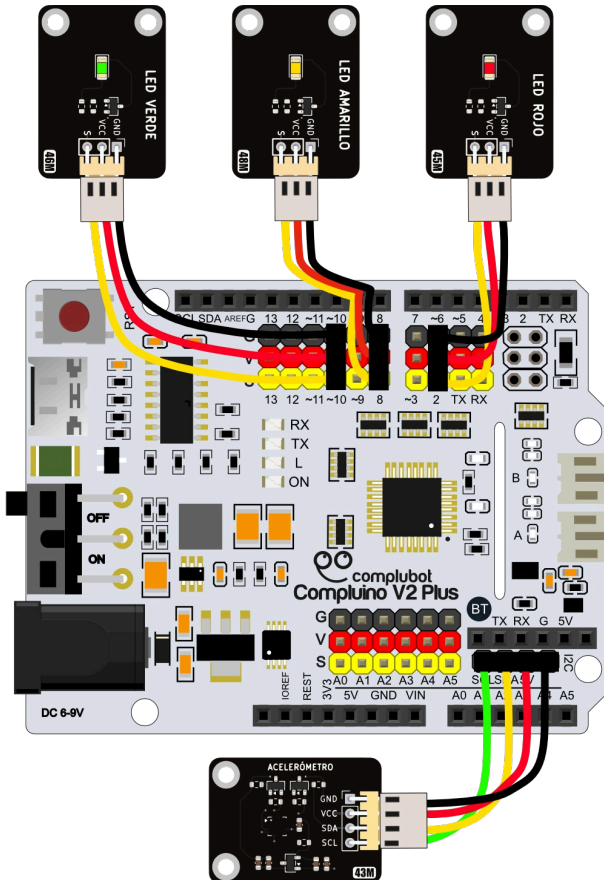
Este módulo incorpora un sensor del tipo LIS3DH. Este es un acelerómetro de tres ejes X, Y y Z, por lo que puede detectar inclinaciones, movimientos, vibraciones, golpes, orientación y caída libre.

4.19.1 Lectura del módulo acelerómetro con la placa Compluino

Este módulo se conecta mediante una conexión I2C y necesita de dos librerías para poder ser controlado desde un programa:

- Wire.h, librería interna para la comunicación I2C
- Adafruit_LIS3DH.h, librería externa para la comunicación con este módulo

4.19.2 Conexión del acelerómetro a la placa Compluino



En esta imagen puedes ver la conexión de los módulos a la placa Compluino, mediante un cable Molex-Header de 4 pines para el módulo acelerómetro y de 3 pines para los módulos led.

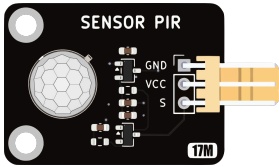
En el módulo solo existe una posición posible de conexión.

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

4.19.3 Programa de ejemplo para el acelerómetro

```
// =====  
// Acelerómetro 43M con ledes indicadores  
// Lee los ejes X, Y y Z del módulo 43M por I2C.  
// Si un eje tiene valor positivo, enciende su LED.  
// Librería necesaria: Adafruit LIS3DH.  
// =====  
  
#include <Wire.h>  
#include <Adafruit_LIS3DH.h>  
#define LED_ROJO 2      // Eje X  
#define LED_VERDE 10    // Eje Y  
#define LED_AMARILLO 8  // Eje Z  
Adafruit_LIS3DH accel;  
  
void setup() {  
  Serial.begin(115200);  
  Wire.begin();  
  pinMode(LED_ROJO, OUTPUT);  
  pinMode(LED_VERDE, OUTPUT);  
  pinMode(LED_AMARILLO, OUTPUT);  
  if (!accel.begin(0x19)) {  
    Serial.println("Acelerometro no detectado");  
    while (1);  
  }  
  Serial.println("Acelerometro listo");  
}  
  
void loop() {  
  accel.read();  
  int x = accel.x;  
  int y = accel.y;  
  int z = accel.z;  
  Serial.print("X: ");  
  Serial.print(x);  
  Serial.print(" Y: ");  
  Serial.print(y);  
  Serial.print(" Z: ");  
  Serial.println(z);  
  digitalWrite(LED_ROJO, x > 0);  
  digitalWrite(LED_VERDE, y > 0);  
  digitalWrite(LED_AMARILLO, z > 0);  
  delay(100);  
}
```

4.20 Módulo sensor PIR (17M)



El sensor PIR (Passive InfraRed) detecta el movimiento de personas o animales midiendo los cambios en la radiación infrarroja del entorno. No emite ninguna señal, solo capta variaciones de calor, por eso se llama pasivo. Cuando el sensor detecta movimiento, envía una señal alta (1) durante unos segundos; después vuelve al estado bajo (0).

estado bajo (0).

4.20.1 Lectura del sensor PIR con la placa Compluino

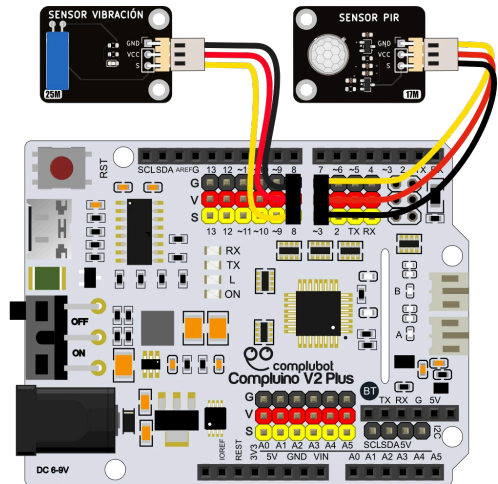
El sensor PIR es un sensor digital. Se puede conectar a cualquiera de los pines digitales 2, 3, 8, 9, 10, 12 o 13. Cuando detecta movimiento, la placa Compluino lee un valor alto (1). Cuando no detecta movimiento, lee un valor bajo (0).

4.20.2 Conexión del sensor PIR a la placa Compluino

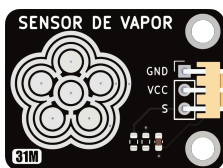
En esta imagen puedes ver la conexión del módulo a la placa Compluino mediante un cable Molex-Header de 3 pines. En el módulo solo existe una posición posible de conexión. En la placa Compluino, es importante asegurarse de que el cable negro coincida con el terminal negativo (GND).

4.20.3 Programa de ejemplo para el sensor PIR

```
#define PIR 3
#define VIBRADOR 8
int movimiento = 0;
void setup() {
  pinMode(PIR, INPUT);
  pinMode(VIBRADOR, OUTPUT);
}
void loop() {
  movimiento = digitalRead(PIR);
  if (movimiento == HIGH) {
    digitalWrite(VIBRADOR, HIGH); // Si hay movimiento, vibra
  }
  else {
    digitalWrite(VIBRADOR, LOW); // Si no hay movimiento, se para
  }
}
```



4.21 Módulo sensor de vapor (31M)



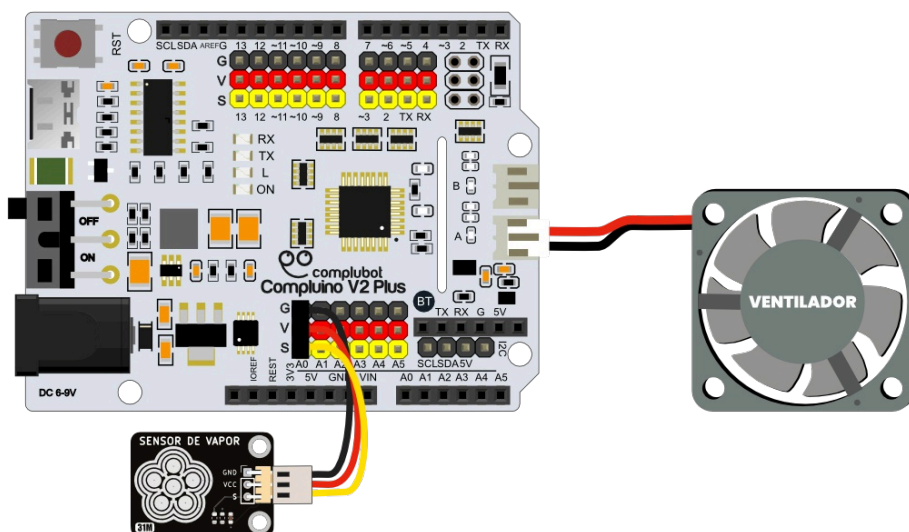
Un sensor de vapor detecta la presencia y concentración de gases o vapores en el aire mediante un elemento sensible que varía su resistencia al contacto con determinadas sustancias. Con la placa Compluino, permite medir niveles de vapor, activar alarmas, encender ledes o registrar cambios ambientales según la intensidad detectada.

4.21.1 Lectura del módulo sensor de vapor con la placa Compluino

Este módulo entrega una señal analógica que cambia según la cantidad de vapor detectada. Para leerlo, se conecta a una entrada analógica de la placa Compluino y se utiliza la función `analogRead()`.

4.21.2 Conexión del sensor de vapor 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).

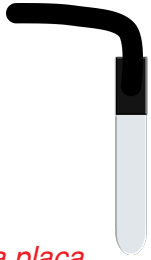


4.21.3 Programa de ejemplo para el sensor de vapor

```
// =====  
// Control automático de ventilador con sensor de vapor  
// =====  
// Este programa lee un sensor de vapor conectado a A0.  
// Si el nivel de vapor supera el umbral establecido,  
// activa un ventilador mediante el driver de motores  
// integrado en la Compuino V2 Plus.  
// Si el nivel de vapor está por debajo del umbral,  
// el ventilador permanece apagado.  
// =====  
  
#define VAPOR A0  
#define DIR_MOTOR 4  
#define VENTILADOR 5  
  
int valor = 0;  
int umbral = 500;  
  
void setup() {  
  pinMode(DIR_MOTOR, OUTPUT);  
  pinMode(VENTILADOR, OUTPUT);  
  Serial.begin(9600);  
}  
  
void loop() {  
  valor = analogRead(VAPOR);  
  
  Serial.print("Nivel de vapor: ");  
  Serial.println(valor);  
  
  if (valor > umbral) {  
    digitalWrite(DIR_MOTOR, HIGH);  
    analogWrite(VENTILADOR, 255);  
  }  
  else {  
    analogWrite(VENTILADOR, 0);  
  }  
  
  delay(100);  
}
```


4.22 Sonda de temperatura para líquidos

Esta sonda de temperatura (DS18B20) para líquidos mide la temperatura dentro de agua u otras soluciones usando un sensor encapsulado en acero inoxidable para evitar daños. Con la placa Compluino, permite obtener lecturas precisas mediante comunicación digital, útil para controlar procesos térmicos, monitorizar recipientes o registrar variaciones en experimentos.



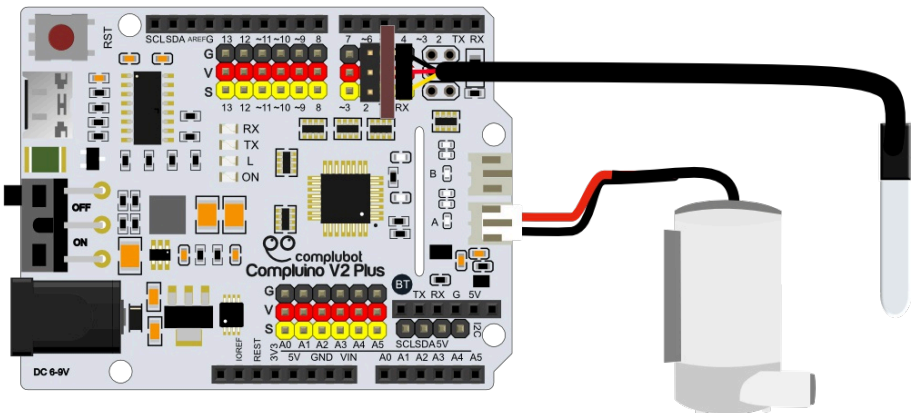
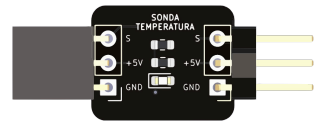
4.22.1 Lectura de la sonda de temperatura para líquidos con la placa Compluino

La sonda DS18B20 es un sensor digital de temperatura encapsulado en acero inoxidable. Esto permite medir la temperatura en líquidos o en entornos húmedos sin dañar el sensor. Para comunicarse con la placa Compluino utiliza el protocolo OneWire, por lo que solo necesita un pin digital de datos.

Para programarla se utilizan dos librerías externas: OneWire.h y DallasTemperature.h. Con estas librerías, la placa Compluino puede solicitar la temperatura a la sonda y obtener el valor directamente en grados Celsius.

4.22.2 Conexión de la sonda de temperatura para líquidos a la placa Compluino

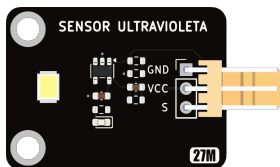
En esta imagen puedes ver la conexión de la sonda a la placa Compluino mediante su cable y el adaptador de conexión.



4.22.3 Programa de ejemplo para la sonda de temperatura para líquidos

```
// =====  
// Control de una bomba según temperatura  
// =====  
// Lee una sonda digital de temperatura. Si la temperatura del agua supera el  
// umbral, activa una bomba de agua.  
// Conexión de la sonda con adaptador:  
// GND -> G, VCC -> V, DATA -> S del pin digital 2  
// Al usar el adaptador de la sonda,  
// no es necesario añadir una resistencia externa.  
// Librerías necesarias:  
// OneWire  
// DallasTemperature  
// =====  
#include <OneWire.h>  
#include <DallasTemperature.h>  
#define TEMP_AGUA 2 // Señal de la sonda DS18B20  
#define DIR_BOMBA 4 // Dirección de la bomba  
#define BOMBA 5 // Velocidad de la bomba (PWM)  
OneWire oneWire(TEMP_AGUA);  
DallasTemperature sensorTemp(&oneWire);  
float temperatura = 0;  
float umbral = 30.0; // Temperatura a partir de la cual se activa la bomba  
void setup() {  
    pinMode(DIR_BOMBA, OUTPUT);  
    pinMode(BOMBA, OUTPUT);  
    Serial.begin(9600);  
    sensorTemp.begin();  
    Serial.println("Sistema de control de bomba iniciado");  
}  
void loop() {  
    sensorTemp.requestTemperatures();  
    temperatura = sensorTemp.getTempCByIndex(0);  
    Serial.print("Temperatura agua: ");  
    Serial.print(temperatura);  
    Serial.println(" C");  
    if (temperatura > umbral) {  
        digitalWrite(DIR_BOMBA, HIGH);  
        analogWrite(BOMBA, 255);  
        Serial.println("Bomba activada");  
    }  
    else {  
        analogWrite(BOMBA, 0);  
        Serial.println("Bomba apagada");  
    }  
    Serial.println("-----");  
    delay(1000);  
}
```

4.23 Módulo sensor ultravioleta (27M)

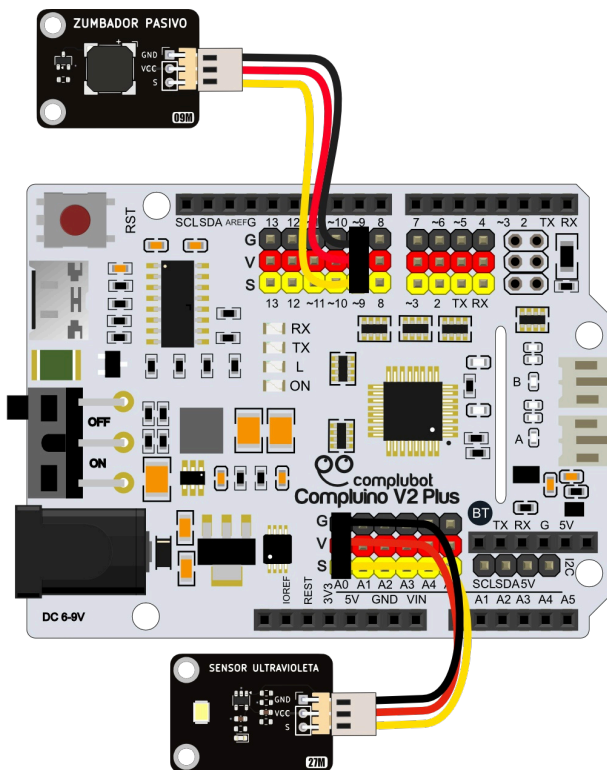


El sensor ultravioleta (UV) mide la intensidad de la radiación ultravioleta presente en la luz solar. Este tipo de radiación, invisible para el ojo humano, es responsable del bronceado y las quemaduras en la piel. El sensor convierte la cantidad de radiación UV en una señal analógica que la placa Compluino puede leer y transformar en un valor numérico entre 0 y 1023.

4.23.1 Lectura del módulo sensor ultravioleta con la placa Compluino

El módulo sensor ultravioleta entrega una señal analógica proporcional a la radiación UV detectada. Para leerlo, se conecta a una entrada analógica de la placa Compluino y se utiliza la función `analogRead()`.

4.23.2 Conexión del sensor ultravioleta a la placa Compluino



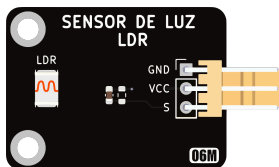
En esta imagen puedes ver la conexión del módulo a la placa Compluino mediante un cable Molex-Header de 3 pines.

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).

```
// =====  
// Avisador de radiación ultravioleta  
// =====  
// Este programa lee un sensor UV conectado a A0.  
// Si el valor medido supera un umbral establecido,  
// el zumbador pasivo emite un sonido de aviso.  
// El valor del sensor se muestra en el monitor serie.  
// =====  
  
// =====  
// Definición de componentes  
// =====  
  
#define UV A0    // Sensor de ultravioleta conectado al pin A0  
#define ZUM 9    // Zumbador pasivo conectado al pin digital 9  
  
int valorUV = 0;    // Variable para guardar la lectura del sensor UV  
int umbralUV = 600; // Umbral de ejemplo a partir del cual suena el zumbador  
  
void setup() {  
  pinMode(ZUM, OUTPUT); // Configuramos el zumbador como salida  
  
  Serial.begin(9600); // Iniciamos el monitor serie para ver los valores  
  Serial.println("Sistema de aviso UV iniciado");  
}  
  
void loop() {  
  valorUV = analogRead(UV); // Leemos el valor del sensor UV entre 0 y 1023  
  
  // Mostramos el valor leído en el monitor serie  
  Serial.print("UV: ");  
  Serial.println(valorUV);  
  
  // Si la radiación UV supera el umbral, activamos el aviso sonoro  
  if (valorUV > umbralUV) {  
    tone(ZUM, 1500); // El zumbador emite un tono agudo  
    Serial.println("Nivel UV alto");  
  }  
  else {  
    noTone(ZUM); // Si el valor es bajo, el zumbador permanece en silencio  
    Serial.println("Nivel UV normal");  
  }  
  
  Serial.println("-----");  
  
  // Pequeña pausa antes de realizar una nueva lectura  
  delay(100);  
}
```

4.24 Módulo sensor LDR (06M)



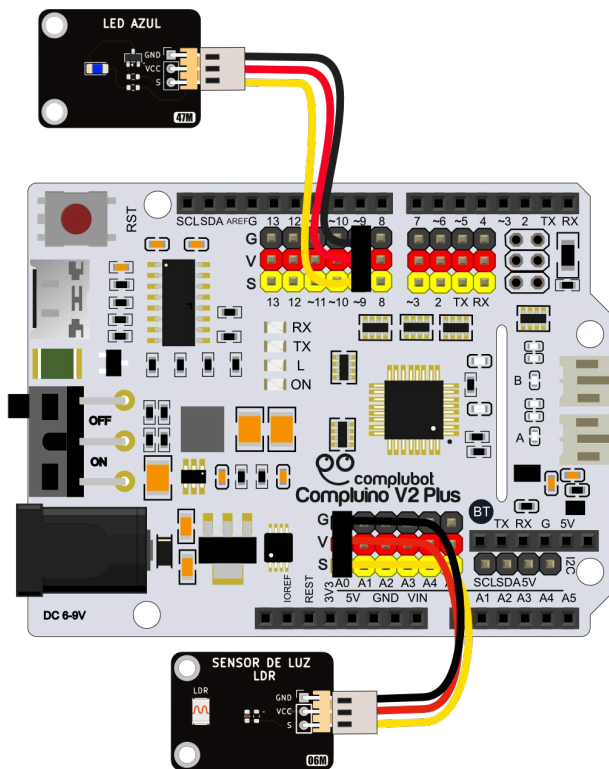
El sensor LDR permite detectar la cantidad de luz ambiente mediante una resistencia dependiente de la luz. Cuando cambia la iluminación que recibe el sensor, también cambia la señal analógica que envía a la placa Compluino. Este módulo se puede utilizar para crear sistemas que reaccionan a la luz, como

luces automáticas, avisos de oscuridad o proyectos de medición de iluminación.

4.24.1 Lectura del módulo sensor LDR con la placa Compluino

El módulo sensor LDR entrega una señal analógica. Para leerlo, se conecta a una entrada analógica de la placa Compluino y se utiliza la función `analogRead()`. El valor obtenido cambia según la cantidad de luz que recibe el sensor.

4.24.2 Conexión del sensor LDR a la placa Compluino



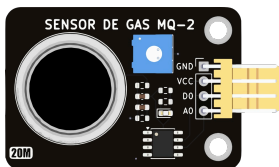
En esta imagen puedes ver la conexión del módulo a la placa Compluino mediante un cable Molex-Header de 3 pines. En el módulo solo existe una posición posible de conexión.

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

4.24.3 Programa de ejemplo para el sensor LDR

```
// =====  
// Control de brillo de un LED con una LDR  
// =====  
// Este programa utiliza un sensor de luz LDR  
// para modificar el brillo de un LED azul.  
//  
// En este ejemplo el brillo está invertido:  
// cuanto más luz detecta la LDR, menos brilla el LED.  
//  
// Si no se quiere invertir el brillo, se puede cambiar  
// la función map() para que más luz produzca más brillo.  
// =====  
  
#define LDR A0          // Sensor de luz LDR conectado al pin analógico A0  
#define LED_AZUL 9      // LED azul conectado al pin PWM 9  
  
int valor = 0;          // Variable para guardar la lectura de la LDR  
int brillo = 0;         // Variable para guardar el brillo del LED  
  
void setup() {  
  pinMode(LED_AZUL, OUTPUT);  
  Serial.begin(9600);  
  Serial.println("Control de brillo con sensor LDR");  
  Serial.println("Cuanta menos luz haya, mas brillara el LED");  
  Serial.println("-----");  
}  
  
void loop() {  
  // Leemos el valor analógico del sensor LDR  
  // El valor estará entre 0 y 1023  
  valor = analogRead(LDR);  
  
  // Convertimos el valor de la LDR en un valor de brillo.  
  // analogWrite() utiliza valores entre 0 y 255.  
  //  
  // En este ejemplo el brillo está invertido:  
  // valor bajo de luz -> brillo alto  
  // valor alto de luz -> brillo bajo  
  brillo = map(valor, 0, 1023, 255, 0);  
  
  // Si no queremos invertir el brillo, usaríamos esta línea:  
  // brillo = map(valor, 0, 1023, 0, 255);  
  // De esta forma:  
  // valor bajo de luz -> brillo bajo  
  // valor alto de luz -> brillo alto  
  analogWrite(LED_AZUL, brillo);  
  Serial.print("Valor LDR: ");  
  Serial.print(valor);  
  Serial.print(" | Brillo LED: ");  
  Serial.println(brillo);  
  delay(200);  
}
```


4.25 Módulo sensor MQ2 (20M)



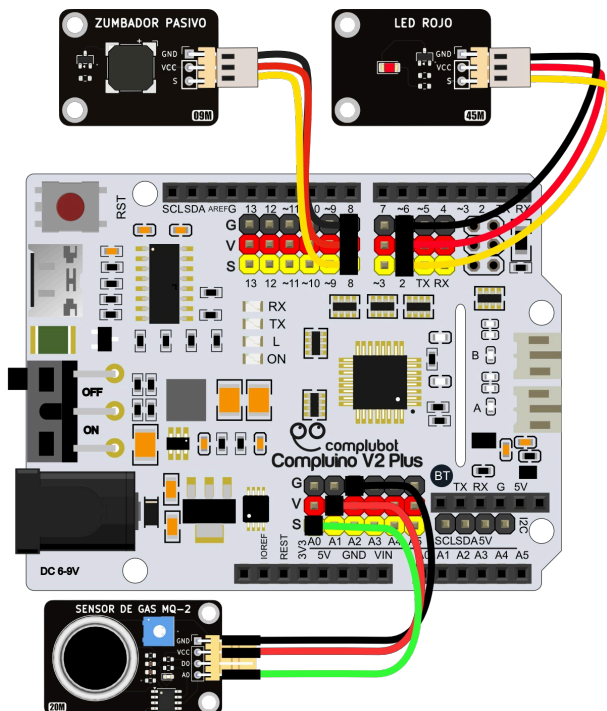
Un sensor MQ2 detecta gases combustibles como humo, propano, hidrógeno o metano mediante un elemento sensible que cambia su resistencia cuando entra en contacto con estos gases. Con la placa Compluino, entrega una señal analógica que permite medir la concentración aproximada y activar respuestas como alarmas, ventiladores o

indicadores visuales.

4.25.1 Lectura del módulo sensor MQ2 con la placa Compluino

El módulo sensor MQ2 entrega una señal analógica que varía según la concentración aproximada de ciertos gases o humo en el aire. Para leerlo, se conecta a una entrada analógica de la placa Compluino y se utiliza la función `analogRead()`. Este sensor necesita un breve tiempo de calentamiento para que sus lecturas sean más estables.

4.25.2 Conexión del sensor MQ2 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.

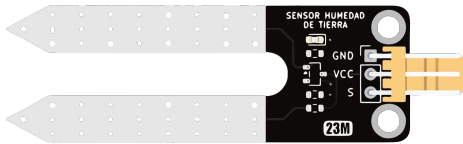
En el módulo solo existe una posición posible de conexión.

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

4.25.3 Programa de ejemplo para el sensor MQ2

```
// =====  
// Detector de gas con sensor MQ2  
// =====  
// Este programa lee el valor del sensor de gas MQ2.  
// Si el valor supera un umbral establecido,  
// se activa una alarma con un LED rojo y un sonido.  
// Si el valor está por debajo del umbral,  
// el LED permanece apagado y el altavoz no suena.  
// =====  
  
#define MQ2 A0  
#define LED_ROJO 2  
#define ALTAVOZ 8  
  
int valorGas = 0;  
int umbral = 500;  
  
void setup() {  
  pinMode(LED_ROJO, OUTPUT);  
  pinMode(ALTAVOZ, OUTPUT);  
  Serial.begin(9600);  
}  
  
void loop() {  
  valorGas = analogRead(MQ2);  
  
  Serial.print("MQ2: ");  
  Serial.println(valorGas);  
  
  if (valorGas > umbral) {  
    digitalWrite(LED_ROJO, HIGH);  
    tone(ALTAVOZ, 1500);  
  }  
  else {  
    digitalWrite(LED_ROJO, LOW);  
    noTone(ALTAVOZ);  
  }  
  
  delay(100);  
}
```

4.26 Módulo sensor de humedad del suelo (23M)



El sensor de humedad del suelo mide la cantidad de agua presente en el suelo. Está formado por dos varillas metálicas que funcionan como electrodos: cuando el suelo está húmedo, conduce mejor la

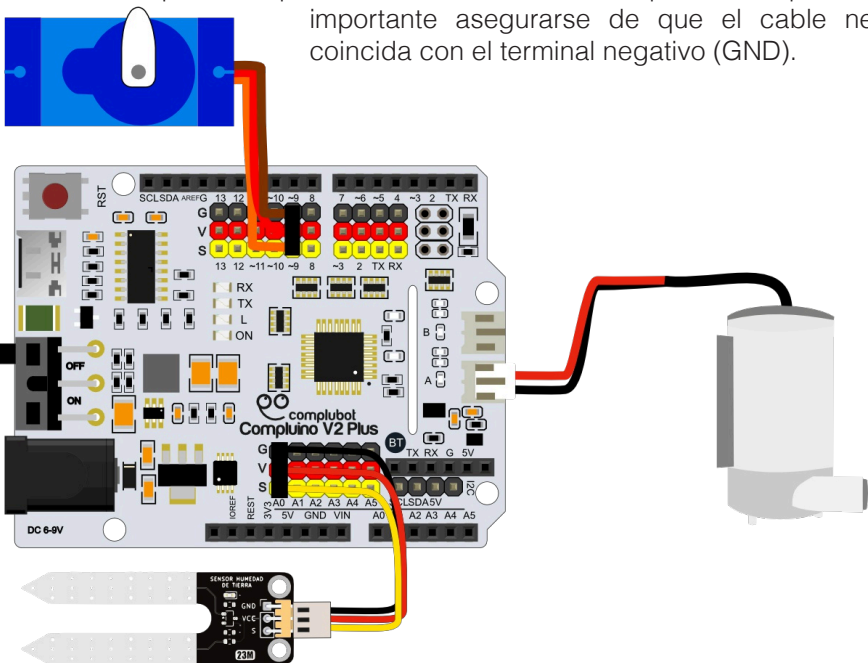
electricidad y la placa Compuino recibe un valor analógico más alto; cuando el suelo está seco, el valor disminuye.

4.26.1 Lectura del módulo sensor de humedad del suelo con la placa Compuino

El módulo sensor de humedad del suelo entrega una señal analógica. Para leerlo, se conecta a una entrada analógica de la placa Compuino y se utiliza la función `analogRead()`. El valor obtenido permite comparar el estado del suelo y decidir si está más seco o más húmedo.

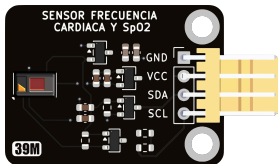
4.26.2 Conexión del montaje a la placa Compuino

En esta imagen puedes ver la conexión de los módulos a la placa Compuino. El servo y la bomba de agua mediante sus propios cables y el módulo sensor de humedad del suelo mediante un cable Molex-Header de 3 pines. En el módulo solo existe una posición posible de conexión. En la placa Compuino, es importante asegurarse de que el cable negro coincida con el terminal negativo (GND).



```
// =====  
// Sistema de riego automático  
// Este programa lee la humedad de la tierra.  
// Si la tierra está seca, mueve un servo para  
// colocar el difusor en posición de riego  
// y activa una bomba de agua durante 10 segundos.  
// Si la tierra tiene suficiente humedad,  
// la bomba permanece apagada y el servo vuelve  
// a su posición inicial.  
// =====  
  
#include <Servo.h>  
  
#define HUMEDAD A0  
#define SERVO_1 9  
#define DIR_BOMBA 4  
#define BOMBA 5  
  
Servo servo1;  
  
int valor = 0;  
int umbral = 600;  
  
void setup() {  
  pinMode(DIR_BOMBA, OUTPUT);  
  pinMode(BOMBA, OUTPUT);  
  servo1.attach(SERVO_1);  
  Serial.begin(9600);  
}  
  
void loop() {  
  valor = analogRead(HUMEDAD);  
  
  Serial.print("Humedad: ");  
  Serial.println(valor);  
  
  if (valor > umbral) {  
    servo1.write(90);  
  
    digitalWrite(DIR_BOMBA, HIGH);  
    analogWrite(BOMBA, 255);  
  
    delay(10000);  
  
    analogWrite(BOMBA, 0);  
    servo1.write(0);  
  }  
  else {  
    analogWrite(BOMBA, 0);  
    servo1.write(0);  
  }  
  delay(500);  
}
```

4.27 Módulo sensor de frecuencia cardíaca (39M)



Un sensor de frecuencia cardíaca mide los latidos detectando cambios en la luz reflejada por la piel. Utiliza un led y un fotodiodo para registrar variaciones en el flujo sanguíneo. Con la placa Compluino, permite obtener el ritmo cardíaco en pulsaciones por minuto y activar acciones según

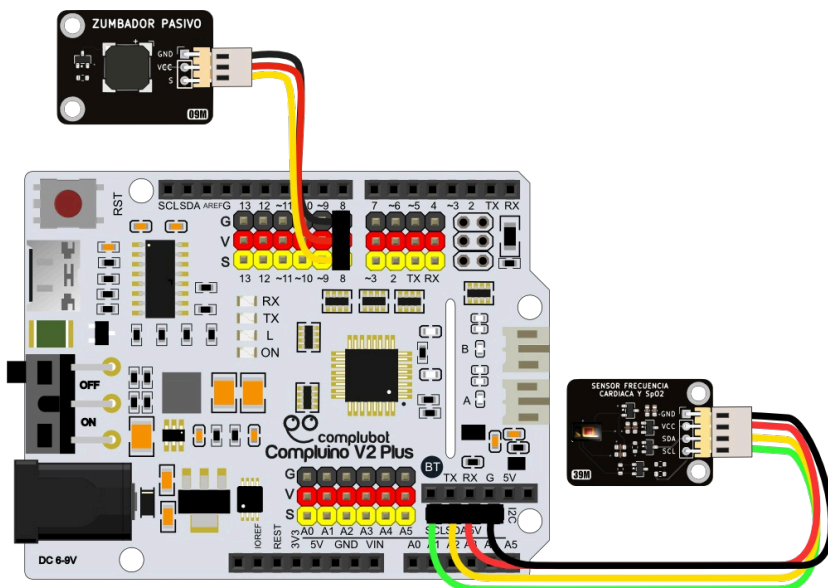
cada latido.

4.27.1 Lectura del módulo sensor de frecuencia cardíaca con la placa Compluino

El módulo sensor de frecuencia cardíaca detecta pequeñas variaciones producidas por el flujo sanguíneo en el dedo. Para utilizarlo, se conecta a la placa Compluino y se leen los cambios de señal desde el programa. Sus valores pueden variar según la colocación del dedo, la presión ejercida y la luz ambiente.

4.27.2 Conexión del sensor de frecuencia cardíaca a la placa Compluino

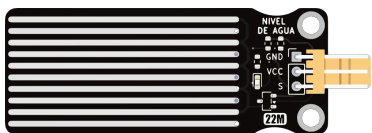
En esta imagen puedes ver la conexión del módulo a la placa Compluino mediante un cable Molex-Header de 4 pines. En el módulo solo existe una posición posible de conexión. En la placa Compluino, es importante asegurarse de que el cable negro coincida con el terminal negativo (GND).



4.27.3 Programa de ejemplo para el sensor de frecuencia cardíaca

```
#include <Wire.h>
#include "MAX30105.h" // Para controlar el sensor de pulso MAX30105/MAX30102
#include "heartRate.h" // Incluye el algoritmo para detectar latidos del corazón
#define DIR_PULSO 0x57 // Dirección I2C típica del sensor MAX3010x
#define ZUM 8
MAX30105 sensorPulso; // Creamos el objeto del sensor de pulso
int bpm = 0; // Pulsaciones por minuto
int valorIR = 0; // Señal infrarroja detectada por el sensor
unsigned long tiempoUltimoLatido = 0; // Momento en que se detectó el último latido
void setup() {
  Serial.begin(9600);
  Wire.begin();
  pinMode(ZUM, OUTPUT);
  // Inicializamos el sensor de pulso
  if (!sensorPulso.begin(Wire, I2C_SPEED_STANDARD)) {
    // Si el sensor no se detecta mostramos un mensaje de error
    Serial.println("Error sensor pulso");
    // Detenemos el programa
    while (1);
  }
  sensorPulso.setup(); // Configuración básica del sensor
  sensorPulso.setPulseAmplitudeRed(0x0A); // Activamos el LED rojo del sensor
  sensorPulso.setPulseAmplitudeGreen(0); // Apagamos el LED verde
  Serial.println("Sensor de frecuencia cardiaca listo");
}
void loop() {
  // Esta señal cambia cuando pasa sangre por el dedo
  valorIR = sensorPulso.getIR();
  // La función checkForBeat detecta si ha ocurrido un latido
  if (checkForBeat(valorIR)) {
    // Guardamos el tiempo actual
    unsigned long tiempoActual = millis();
    // Calculamos el tiempo entre latidos
    unsigned long intervalo = tiempoActual - tiempoUltimoLatido;
    // Si no es el primer latido calculamos las pulsaciones por minuto
    if (tiempoUltimoLatido > 0 && intervalo > 0) {
      // 60000 ms = 1 minuto
      bpm = 60000 / intervalo;
    }
    // Guardamos el momento del último latido
    tiempoUltimoLatido = tiempoActual;
    // Mostramos las pulsaciones en el monitor serie
    Serial.print("BPM: ");
    Serial.println(bpm);
    // Emitimos un pitido corto con el zumbador
    // cada vez que se detecta un latido
    tone(ZUM, 1200);
    delay(50);
    noTone(ZUM);
  }
  // Pequeña pausa para estabilizar las lecturas
  delay(10);
}
```


4.28 Módulo sensor de nivel de agua (22M)



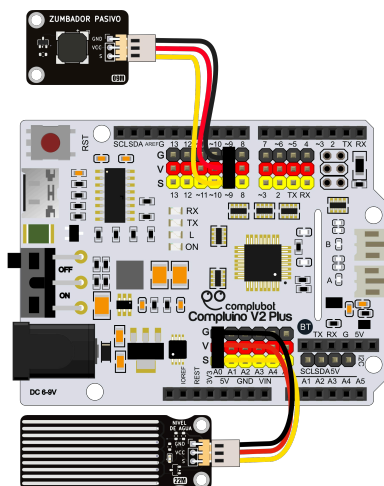
El sensor de nivel de agua detecta cuánta parte del módulo está en contacto con el agua. La placa Compuino lee esa variación como un valor analógico.

4.28.1 Lectura del módulo sensor de nivel de agua con la placa Compuino

Este módulo se conecta a una entrada analógica y se lee con `analogRead()`. Cuanta más agua toca el sensor, mayor suele ser el valor obtenido.

4.28.2 Conexión del sensor de nivel de agua a la placa Compuino

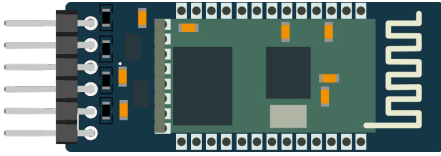
En esta imagen puedes ver la conexión del módulo a la placa Compuino mediante un cable Molex-Header de 3 pines. Comprueba que el cable negro coincida con GND.



4.28.3 Programa de ejemplo para el sensor de nivel de agua

```
#define SENSOR_AGUA A0 // Sensor de nivel de agua
#define ZUMBADOR 8 // Zumbador pasivo conectado al pin 8
int nivel = 0; // Variable para guardar la lectura
void setup() {
  Serial.begin(9600); // Iniciar monitor serie
  pinMode(ZUMBADOR, OUTPUT);
}
void loop() {
  // Leer valor del sensor (0 - 1023)
  nivel = analogRead(SENSOR_AGUA); // Mostrar valor en el monitor serie
  Serial.print("Nivel de agua: ");
  Serial.println(nivel); // Si el nivel es alto, activar alarma
  if (nivel > 600) {
    tone(ZUMBADOR, 1000); // Sonido de 1000 Hz
    delay(200);
    noTone(ZUMBADOR); // Apagar sonido
    delay(200);
  } else {
    noTone(ZUMBADOR); // Si no hay mucha agua, silencio
  }
  delay(300);
}
```

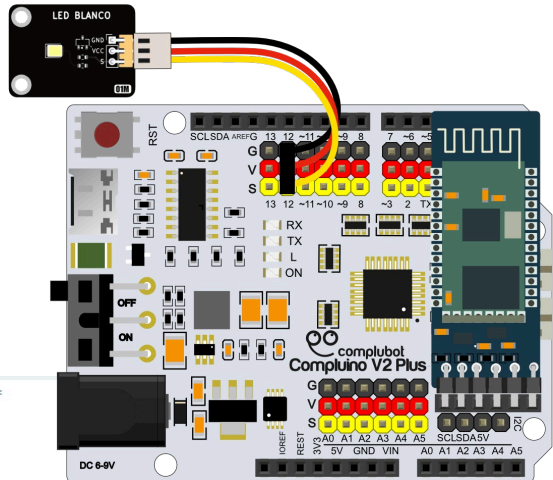
4.29 Módulo Bluetooth JDY-31



dispositivo conectado, y los datos enviados desde ese dispositivo pueden llegar a la placa Compuino.

4.29.1 Comunicación del módulo Bluetooth con la placa Compuino

El módulo se comunica mediante UART, utilizando las señales TX y RX. En la placa Compuino se conecta al conector Bluetooth, respetando la orientación correcta del módulo.



4.29.2 Programa de ejemplo para el módulo Bluetooth

```
// =====
// Control de un LED mediante Bluetooth
// usando el conector Bluetooth
// de la Compuino V2 Plus.
// Conexión del módulo Bluetooth:
// G o GND -> GND
// VCC o V -> V
// TX -> RX
// RX -> TX
// Envía "1" para encender el LED y "0" para apagarlo.
// =====
#define LED 12 // Módulo LED conectado al pin digital 12
void setup() {
  pinMode(LED, OUTPUT);
  // Iniciamos la comunicación serie con el módulo Bluetooth
  Serial.begin(9600);
}
void loop() {
  // Comprobamos si ha llegado algún dato por Bluetooth
  if (Serial.available()) {
    char dato = Serial.read();
    if (dato == '1') {
      digitalWrite(LED, HIGH);
    }
    if (dato == '0') {
      digitalWrite(LED, LOW);
    }
  }
}
```

