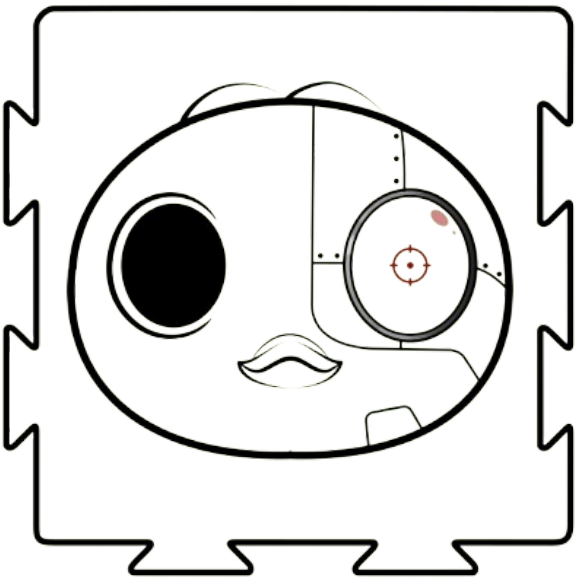
Manual de prácticas

Paluma Pro



PALUMBA

1 Octubre 2024
Versión 0.1

Índice

Conceptos básicos

Definiciones

Para empezar a trabajar en los ejercicios, es importante que tengas algunas nociones básicas. A continuación, te explicamos algunos conceptos que te serán útiles. Si ya tienes experiencia en electrónica digital, Arduino IDE para ESP32 y conoces las partes de un PLC, puedes saltar directamente a los ejercicios. Si no es así, sigue leyendo.

Botones:

Los botones son interruptores mecánicos que funcionan de manera momentánea, permitiendo realizar acciones o encender/apagar circuitos cuando los presionas. Son componentes esenciales en muchos sistemas electrónicos, ya que permiten a los usuarios interactuar de manera simple y directa con los dispositivos.

Entradas:

Una entrada es un punto en el que un dispositivo o sistema recibe información o señales externas, como datos de sensores o dispositivos conectados.

Entradas digitales:

Estas entradas solo tienen dos estados posibles: alto (1) o bajo (0). Estos valores representan los niveles de voltaje típicos en los circuitos digitales.

Entradas analógicas:

Son señales continuas que pueden tomar cualquier valor dentro de un rango específico (por ejemplo, entre 0 y 5V), permitiendo una mayor precisión en la medición de variables como la temperatura o la luz.

ESP32:

El ESP32 es un microcontrolador económico y de alto rendimiento, conocido por poder manejar varias tareas al mismo tiempo. Algunas de sus principales características son:

Wi-Fi y Bluetooth: Tiene conectividad integrada de Wi-Fi y Bluetooth (clásico y BLE), lo que lo hace ideal para proyectos de IoT (Internet de las Cosas).
Múltiples GPIOs: Dispone de muchos pines GPIO (entrada/salida general) para conectar sensores, actuadores y otros periféricos.
Protocolos de comunicación: Soporta varios protocolos como I2C, SPI, UART, PWM, ADC y DAC, haciéndolo flexible para distintas aplicaciones.
Bajo consumo de energía: Está diseñado para consumir poca energía, lo que lo hace ideal para proyectos con baterías de larga duración.

Fotorresistencia:

Una fotorresistencia es un componente que cambia su resistencia dependiendo de la cantidad de luz que recibe. Cuando la luz aumenta, su resistencia disminuye, y cuando la luz baja, su resistencia aumenta.

Relé:

Un relé es un dispositivo electromecánico que funciona como un interruptor controlado eléctricamente. Permite que una señal de baja potencia controle un circuito de alta potencia, actuando como un puente entre ambos.

Salidas:

Una salida es un punto en el que un dispositivo o sistema envía señales o datos hacia el exterior, como para encender un motor o enviar información a una pantalla.

Salidas digitales:

Estas salidas tienen solo dos estados posibles: alto (1) o bajo (0). Un estado alto suele representar 5V y un estado bajo representa 0V.

Salidas analógicas:

Las salidas analógicas proporcionan un rango continuo de valores, permitiendo que el sistema envíe señales de diferente intensidad para controlar, por ejemplo, la velocidad de un motor.

Programa y su configuración necesarios

El programa que utilizaremos para los ejercicios de este manual es **Arduino IDE**.

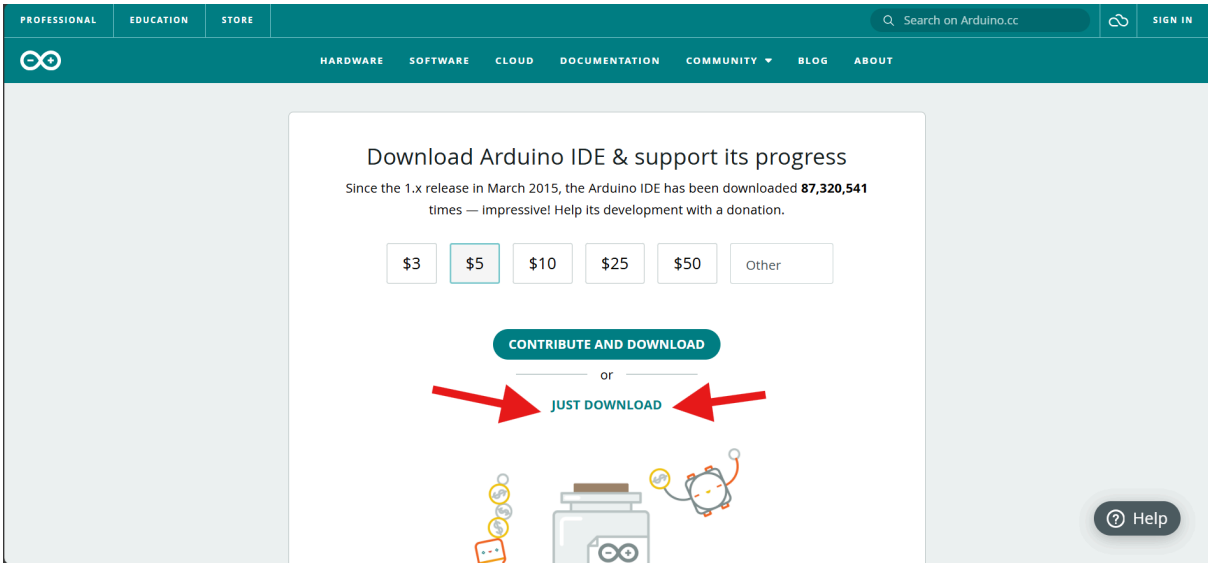
A continuación, te explicamos cómo configurarlo para poder establecer la conexión con el Palumba Pro.

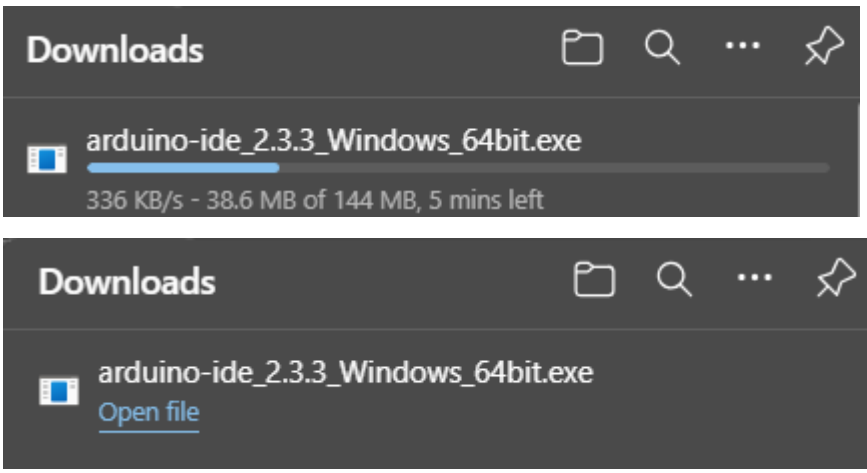
Instalación del Arduino IDE
Si aún no tienes instalado Arduino IDE, sigue estos pasos para descargarlo e instalarlo desde la página oficial de Arduino.

Dirígete a la página oficial de Arduino.
https://www.arduino.cc/en/software

más versátil pequeño y accesible
Desplázate hacia abajo hasta la sección de descargas y selecciona la versión que corresponde a tu sistema operativo (Windows, macOS o Linux).

Una vez que hagas clic en el sistema operativo correspondiente, serás redirigido a una nueva página. Aquí puedes hacer una donación opcional o hacer clic en "Just Download" para continuar con la descarga gratuita.





Cuando se haya descargado el archivo, sigue los pasos de instalación específicos para tu sistema operativo:

Windows: Ejecuta el archivo .exe descargado y sigue el asistente de instalación. Asegúrate de marcar la opción "Instalar drivers" cuando se te solicite.
macOS: Abre el archivo .dmg descargado y arrastra el ícono de Arduino a la carpeta "Aplicaciones".
Linux: Extrae el archivo descargado y ejecuta el script de instalación desde la terminal.

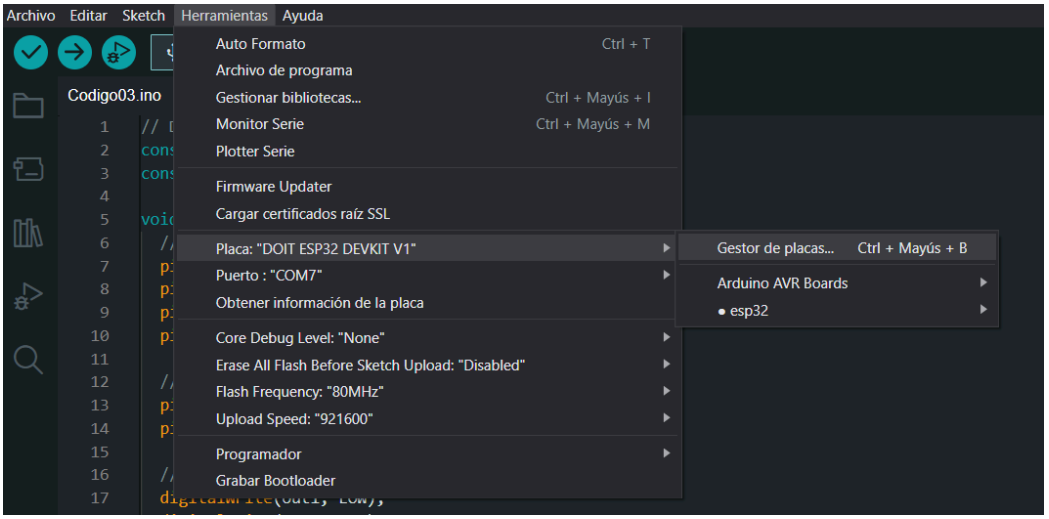
Una vez que la instalación haya finalizado, abre Arduino IDE para proceder con la configuración del ESP32.

El ESP32 no viene incluido en el IDE de Arduino por defecto, pero podemos agregarlo fácilmente utilizando el “Gestor de tarjetas”.

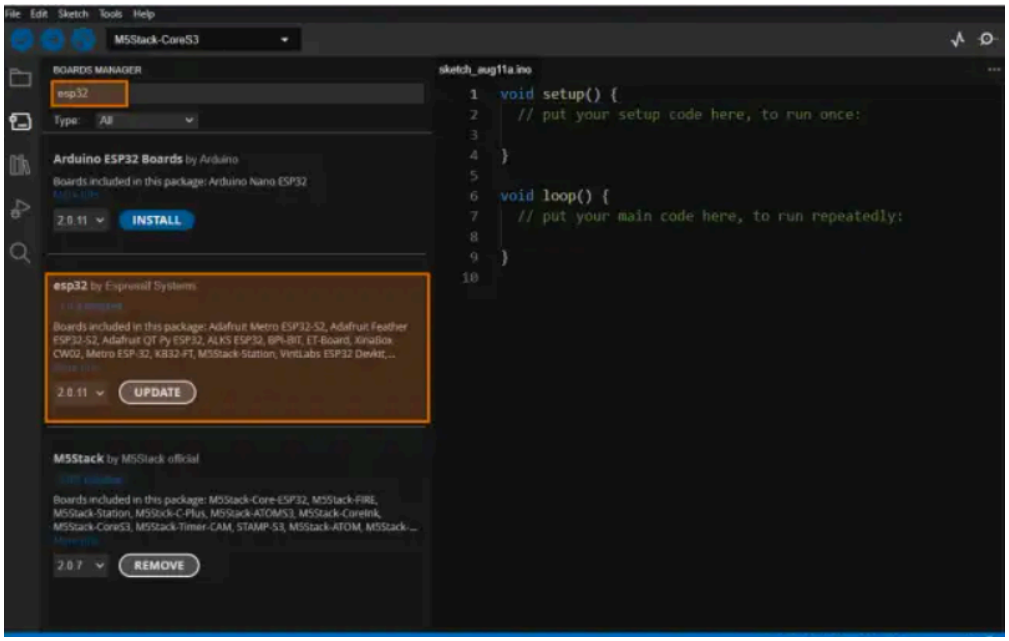
Abre el IDE de Arduino y ve a Archivo > Preferencias. En el campo “URLs adicionales de gestor de tarjetas”, añade la siguiente URL:

https://espressif.github.io/arduino-esp32/package_esp32_index.json

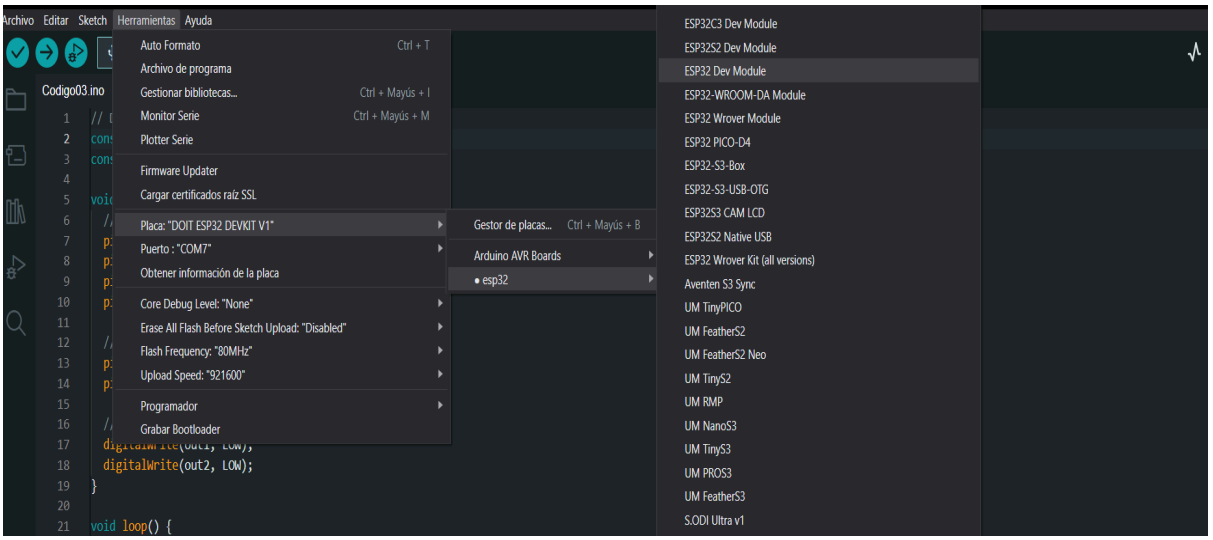
–Luego, vamos a Herramientas > Placa > Gestor de tarjetas.



En el panel que aparece, buscamos “ESP32” en la barra de búsqueda, y hacemos click en “Instalar” en el paquete que aparece.



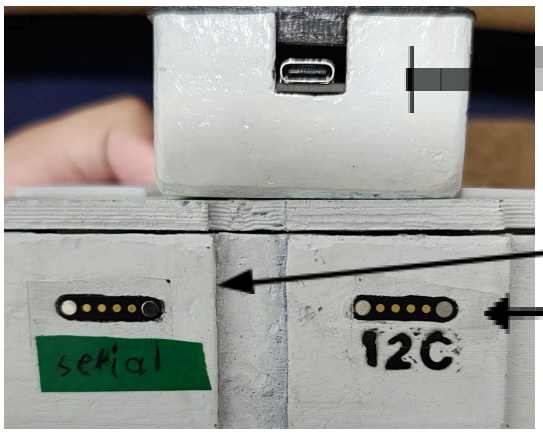
Por último, seleccionas el esp32 que necesitaremos desde Herramientas > Placa > esp32 y escogemos ESP32 Dev Module.



Entradas azules de la parte derecha- Distribución de la fuente externa(Positivos)

Entradas verdes de la parte izquierda- Positivo y negativo de la fuente externa.

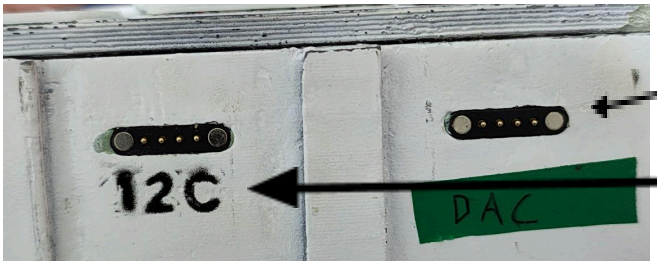
Partes del palumba Pro



Puerto de entrada y programación

Comunicación por Serial

Comunicación por I2C

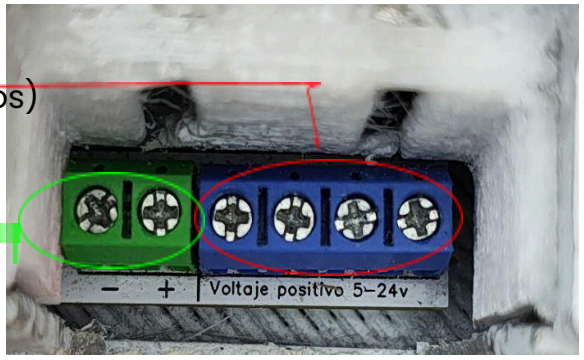


Comunicación por DAC

Comunicación por I2C

Distribución de fuente externa(positivos)

Positivo y negativo de la fuente externa.



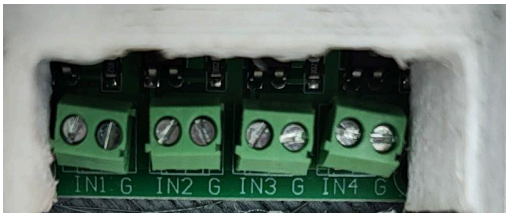
IN1: es la entrada 1

IN2: es la entrada 2

IN3: es la entrada 3

IN4: es la entrada 4

G: es la tierra común



P.C: pin común(aquí se conecta al positivo de la fuente externa).

N.C: circuito normalmente cerrado.

N.A: circuito normalmente abierto.

Tomando de referencia una de las secciones de 3 entradas, la parte izquierda es,
N.A: circuito normalmente abierto.

la parte central es P.C: pin común(aquí se conecta al positivo de la fuente externa).
Y la parte derecha es, N.C: circuito normalmente cerrado.

Ejercicios

Activación de un motor mediante un sensor de temperatura

Materiales a ocupar:

- 1 termistor. (Especificar)
- Jumpers.
- 2 fuentes de voltaje(baterías, eliminadores de voltaje, etc…)(Cuales?)
- Un motor DC(5V, 12V o 24V).

1- Para empezar, se definen los pines:

```
2   const int pinTermistor = 36; // Pin donde se conecta el termistor
3   const int pinRelay = 27;     // Pin donde se conecta el relé
```

2- se proponen variables:

```
6   float umbralTemperatura = 30.0;
7   int valorADC = 0;
8   float voltaje = 0.0;
9   float temperatura = 0.0;
```

3- Se definen las variables:

```
12  const float R_Balance = 10000.0;   // Resistencia balance en Ohms
13  const float B_Constante = 3950.0;  // Constante B del termistor
14  const float R_25 = 10000.0;        // Resistencia del termistor a 25°C
15  const float T_Referencia = 298.15; // Temperatura de referencia en Kelvin (25°C)
```

4- Se configura el inicio de como empezar a funcionar el esp32:

```
17  void setup() {
18    // Iniciamos el monitor serial para depuración
19    Serial.begin(115200);
20
21    // Configuramos el pin del relé como salida
22    pinMode(pinRelay, OUTPUT);
23
24    // Iniciamos el relé en estado apagado
25    digitalWrite(pinRelay, LOW);
26  }
```

5- Se programa el bucle principal para leer y convertir los datos en bucle principal:

```
28  void loop() {
29    // Leer el valor ADC del termistor
30    valorADC = analogRead(pinTermistor);
31
32    // Convertimos el valor ADC a voltaje (ESP32 es de 12 bits, 0-4095)
33    voltaje = valorADC * (3.3 / 4095.0);
34
35    // Calculamos la resistencia del termistor
36    float resistenciaTermistor = (R_Balance * (3.3 / voltaje - 1));
37
38    // Aplicamos la ecuación de Steinhart-Hart para calcular la temperatura
39    temperatura = 1.0 / ((log(resistenciaTermistor / R_25) / B_Constante) + (1.0 / T_Referencia));
40    temperatura = temperatura - 273.15; // Convertir de Kelvin a Celsius
```

6- en el mismo bucle principal se programa para ver los datos recibidos desde la función de monitor serial en el programa arduino IDE:

```
42    // Mostramos los valores en el monitor serie
43    Serial.print("Voltaje: ");
44    Serial.print(voltaje);
45    Serial.print(" V - Temperatura: ");
46    Serial.print(temperatura);
47    Serial.println(" °C");
```

7- Por último, se programa en el mismo bucle principal la parte de activar la salida 1 cuando la temperatura deseada se haya conseguido y se cierra el bucle principal:

```
  // Si la temperatura supera el umbral, activamos el relé
  if (temperatura > umbralTemperatura) {
    digitalWrite(pinRelay, HIGH); // Encender relé
  } else {
    digitalWrite(pinRelay, LOW);  // Apagar relé
  }

  // Esperamos un tiempo antes de la siguiente lectura
  delay(1000);
}
```

Código completo:

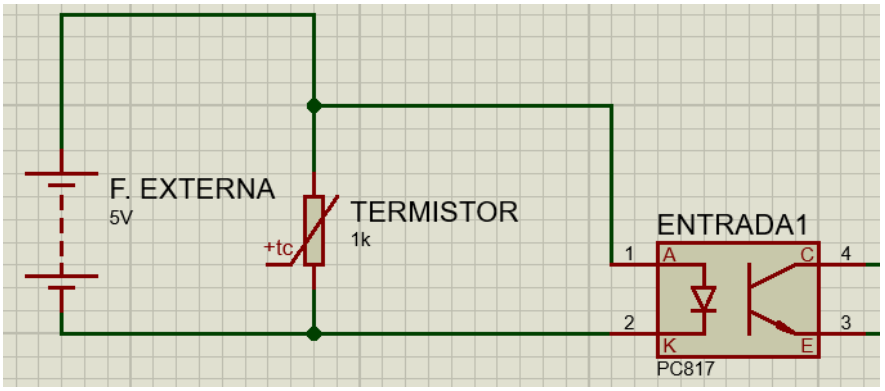
```
// Definimos los pines
const int pinTermistor = 36; // Pin donde se conecta el termistor
const int pinRelay = 27;     // Pin donde se conecta el relé

// Variables para controlar el umbral de temperatura
float umbralTemperatura = 30.0;  // Umbral en grados Celsius (ajusta
según necesites)
int valorADC = 0;
float voltaje = 0.0;
float temperatura = 0.0;
// Parámetros del termistor
const float R_Balance = 10000.0;   // Resistencia balance en Ohms
const float B_Constante = 3950.0;  // Constante B del termistor
const float R_25 = 10000.0;        // Resistencia del termistor a 25°C
const float T_Referencia = 298.15; // Temperatura de referencia en
Kelvin (25°C)
void setup() {
  // Iniciamos el monitor serial para depuración
  Serial.begin(115200);
  // Configuramos el pin del relé como salida
  pinMode(pinRelay, OUTPUT);
  // Iniciamos el relé en estado apagado
  digitalWrite(pinRelay, LOW);
}
```

```
void loop() {
  // Leer el valor ADC del termistor
  valorADC = analogRead(pinTermistor);
  // Convertimos el valor ADC a voltaje (ESP32 es de 12 bits, 0-4095)
  voltaje = valorADC * (3.3 / 4095.0);
  // Calculamos la resistencia del termistor
  float resistenciaTermistor = (R_Balance * (3.3 / voltaje - 1));
  // Aplicamos la ecuación de Steinhart-Hart para calcular la
temperatura
  temperatura = 1.0 / ((log(resistenciaTermistor / R_25) / B_Constante)
+ (1.0 / T_Referencia));
  temperatura = temperatura - 273.15; // Convertir de Kelvin a Celsius
  // Mostramos los valores en el monitor serie
  Serial.print("Voltaje: ");
  Serial.print(voltaje);
  Serial.print(" V - Temperatura: ");
  Serial.print(temperatura);
  Serial.println(" °C");
  // Si la temperatura supera el umbral, activamos el relé
  if (temperatura > umbralTemperatura) {
    digitalWrite(pinRelay, HIGH); // Encender relé
  } else {
    digitalWrite(pinRelay, LOW);  // Apagar relé
  }
  // Esperamos un tiempo antes de la siguiente lectura
  delay(1000);
}
```

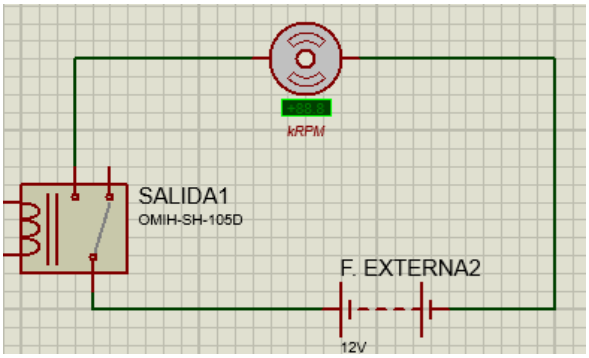
Diagrama del circuito y como conectar:

Para empezar se conectará el termistor En la entrada 1:



Donde se conectará una fuente externa de 5V al termistor y al mismo tiempo al IN1 y G del PALUMBA.

Por otro lado, la conexión de la salida:



Se conectará una fuente correspondiente al motor que se usará, el positivo de la corriente irá conectado al pin común(P.C) de la salida 1 mientras el negativo de la fuente al negativo del motor y finalmente el positivo del motor a la salida normalmente abierta.

Activación de un foco mediante una fotoresistencia

Materiales a ocupar:

- 1 Fotorresistencia.
- Jumpers.
- 1 foco. (especificar)
- 1 socket.
- 1 clavija.
- 2 fuentes de voltaje(baterías, eliminadores de voltaje, etc…)

1- Para empezar, se definen los pines:

```
2   const int pinFotoresistencia = 36; // Pin de entrada para la fotoresistencia
3   const int pinRelevador = 27;       // Pin de salida para el relevador
```

2- definir el umbral de la fotorresistencia:

```
6   const int umbral = 2000;  // Ajusta este valor según tu fotoresistencia y condiciones de luz
```

3- Se configura el inicio de como empezar a funcionar el esp32:

```
8  void setup() {
9    // Inicializar el pin del relevador como salida
10   pinMode(pinRelevador, OUTPUT);
11
12   // Inicializar el pin de la fotoresistencia como entrada (opcional porque el ADC es automático)
13   pinMode(pinFotoresistencia, INPUT);
14
15   // Configurar el relevador en estado apagado inicialmente
16   digitalWrite(pinRelevador, LOW);
17
18   // Inicializar comunicación serie para depuración
19   Serial.begin(115200);
20 }
```

4- Se programa el bucle principal para leer y convertir los datos en bucle principal:

```
22  void loop() {
23    // Leer el valor analógico de la fotoresistencia
24    int valorLuz = analogRead(pinFotoresistencia);
```

5- en el mismo bucle principal se programa para ver los datos recibidos desde la función de monitor serial en el programa arduino IDE:

```
26    // Mostrar el valor en el monitor serie
27    Serial.print("Valor de la fotoresistencia: ");
28    Serial.println(valorLuz);
```

6- Por último, se programa en el mismo bucle principal la parte de activar la salida 1 cuando la intensidad de luz deseada se haya conseguido y se cierra el bucle principal:

```
30   // Verificar si el valor de luz está por debajo del umbral
31   if (valorLuz < umbral) {
32     // Encender el relevador si está oscuro
33     digitalWrite(pinRelevador, HIGH);
34     Serial.println("Relevador encendido");
35   } else {
36     // Apagar el relevador si hay suficiente luz
37     digitalWrite(pinRelevador, LOW);
38     Serial.println("Relevador apagado");
39   }
40
41   // Pequeña pausa para evitar lecturas demasiado rápidas
42   delay(500);
43 }
```

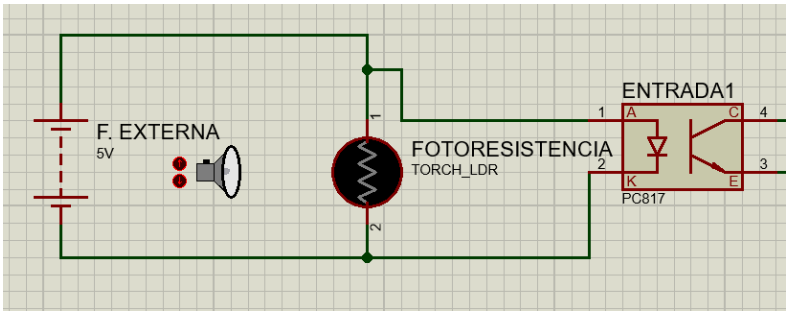
Código completo:

```
// Definir pines
const int pinFotoresistencia = 36; // Pin de entrada para la
fotoresistencia
```

```
const int pinRelevador = 27;       // Pin de salida para el relevador
// Definir el umbral para la activación del relevador
const int umbral = 2000;  // Ajusta este valor según tu fotoresistencia
y condiciones de luz
void setup() {
  // Inicializar el pin del relevador como salida
  pinMode(pinRelevador, OUTPUT);
  // Inicializar el pin de la fotoresistencia como entrada (opcional
porque el ADC es automático)
  pinMode(pinFotoresistencia, INPUT);
  // Configurar el relevador en estado apagado inicialmente
  digitalWrite(pinRelevador, LOW);
  // Inicializar comunicación serie para depuración
  Serial.begin(115200);
}
void loop() {
  // Leer el valor analógico de la fotoresistencia
  int valorLuz = analogRead(pinFotoresistencia);
  // Mostrar el valor en el monitor serie
  Serial.print("Valor de la fotoresistencia: ");
  Serial.println(valorLuz);
  // Verificar si el valor de luz está por debajo del umbral
  if (valorLuz < umbral) {
    // Encender el relevador si está oscuro
    digitalWrite(pinRelevador, HIGH);
    Serial.println("Relevador encendido");
  } else {
    // Apagar el relevador si hay suficiente luz
    digitalWrite(pinRelevador, LOW);
    Serial.println("Relevador apagado");
  }
  // Pequeña pausa para evitar lecturas demasiado rápidas
  delay(500);
}
```

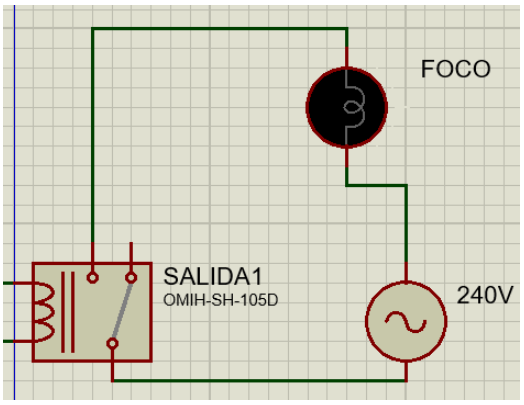
Diagrama del circuito y como conectar:

Para empezar, se conectará la fotorresistencia a la entrada 1:



Se conectará el positivo de la fuente a una pata de la fotorresistencia y ahí se conecta otro cable al (IN1), mientras que el negativo de la fuente irá a la otra pata de la fotorresistencia y conecta a (G).

Por último, conectar la salida 1:



127

En la clavija se conecta 2 cables, uno de ellos se conecta al pin común(P.C), mientras que el otro a un lado de socket y del otro lado de socket ira un cable conectado a la salida normalmente abierta, por último, se conecta el foco al socket y la clavija a la luz.

Activación y desactivación de 2 motores mediante botones

Lista de materiales:

- 4 Push boton.
- 4 resistencia de 220 ohmios.
- 2 fuentes de voltaje(baterías, eliminadores de voltaje, etc…).
- 2 motores DC (5V, 12V o 24V).

1- Para empezar, se definen los pines:

```
2   const int btn1 = 36, btn2 = 39, btn3 = 34, btn4 = 35;
3   const int out1 = 27, out2 = 13;
```

2-Se configura el inicio de como empezar a funcionar el esp32:

```
5   void setup() {
6     // Configurar botones como entradas
7     pinMode(btn1, INPUT);
8     pinMode(btn2, INPUT);
9     pinMode(btn3, INPUT);
10    pinMode(btn4, INPUT);
11
12    // Configurar salidas
13    pinMode(out1, OUTPUT);
14    pinMode(out2, OUTPUT);
15
16    // Inicializar salidas en LOW
17    digitalWrite(out1, LOW);
18    digitalWrite(out2, LOW);
19  }
```

3- Se programa el bucle principal para leer y convertir los datos en bucle principal, empezando la programación para controlar la salida 1 mediante las entradas 1 y 2, donde el botón 1 activara el motor y la entrada 2 lo desactiva:

```
21  void loop() {
22    // Controlar salida 1 (pin 27) con botones 1 y 2
23    if (digitalRead(btn1) == HIGH && digitalRead(btn2) == LOW) {
24      digitalWrite(out1, HIGH); // Activar salida 1 solo si botón 1 está presionado y botón 2 no
25    } else if (digitalRead(btn2) == HIGH && digitalRead(btn1) == LOW) {
26      digitalWrite(out1, LOW);  // Apagar salida 1 solo si botón 2 está presionado y botón 1 no
27    }
```

4- Por último, Se programa el mismo bucle principal para leer y convertir los datos, continuando la programación para controlar la salida 4 mediante las entradas 3 y 4, donde el botón 3 activara el motor y la entrada 4 lo desactiva y así cerrando el bucle principal:

```
29    // Controlar salida 2 (pin 13) con botones 3 y 4
30    if (digitalRead(btn3) == HIGH && digitalRead(btn4) == LOW) {
31      digitalWrite(out2, HIGH); // Activar salida 2 solo si botón 3 está presionado y botón 4 no
32    } else if (digitalRead(btn4) == HIGH && digitalRead(btn3) == LOW) {
33      digitalWrite(out2, LOW);  // Apagar salida 2 solo si botón 4 está presionado y botón 3 no
34    }
35  }
```

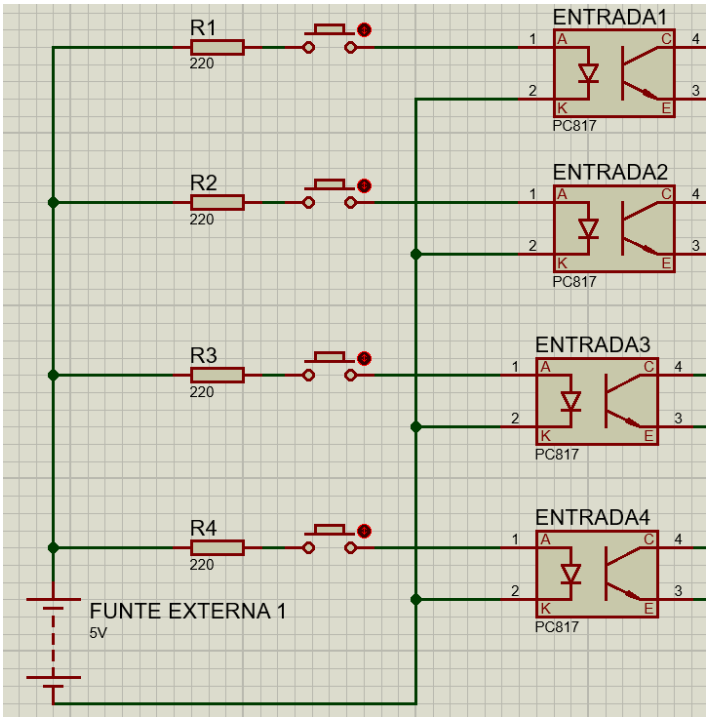
Código completo:

```
// Definir pines de botones y salidas
const int btn1 = 36, btn2 = 39, btn3 = 34, btn4 = 35;
```

```
const int out1 = 27, out2 = 13;
void setup() {
  // Configurar botones como entradas
  pinMode(btn1, INPUT);
  pinMode(btn2, INPUT);
  pinMode(btn3, INPUT);
  pinMode(btn4, INPUT);
  // Configurar salidas
  pinMode(out1, OUTPUT);
  pinMode(out2, OUTPUT);
  // Inicializar salidas en LOW
  digitalWrite(out1, LOW);
  digitalWrite(out2, LOW);
}
void loop() {
  // Controlar salida 1 (pin 27) con botones 1 y 2
  if (digitalRead(btn1) == HIGH && digitalRead(btn2) == LOW) {
    digitalWrite(out1, HIGH);
  } else if (digitalRead(btn2) == HIGH && digitalRead(btn1) == LOW) {
    digitalWrite(out1, LOW);  // Apagar salida 1 solo si botón 2 está
presionado y botón 1 no
  }
  // Controlar salida 2 (pin 13) con botones 3 y 4
  if (digitalRead(btn3) == HIGH && digitalRead(btn4) == LOW) {
    digitalWrite(out2, HIGH);
  } else if (digitalRead(btn4) == HIGH && digitalRead(btn3) == LOW) {
    digitalWrite(out2, LOW);
  }
}
```

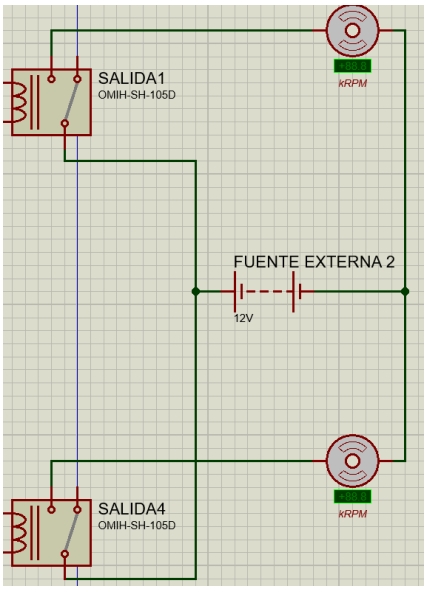
Diagrama del circuito y como conectar:

Para empezar, se conectaran las entradas:



En el positivo de la fuente de 5V se conecta a una pata del botón con su respectiva resistencia cada una mientras que las otras patas de los botones va a IN1, IN2, IN3, IN4, mientras que el negativo de la fuente va a tierra(G) de las entradas.

Por otro lado, En las salidas:



El positivo de la fuente va en el pin común(P.C) de la entradas 1 y 4 y el negativo de la fuente al negativo de los motores, el positivo del primer motor irá conectada a la salida normalmente abierta de la salida 1, mientras que el positivo del segundo motor irá conectada a la salida normalmente abierta de la salida 4