GPIO
(General Purpose Input/Output, Entrada/Salida de Propósito General)

En este capítulo veremos las características de voltaje,corriente y recursos a nivel de librerías que dan soporte al DevKit.

1. Descripción de los GPIO

En la organización de los canales de la ESP32 se aprecia que muchos de los pines están multiplexados para distintas funcionalidades (compartido entre varios periféricos), dependiendo de la versión la cantidad de pines puede variar.

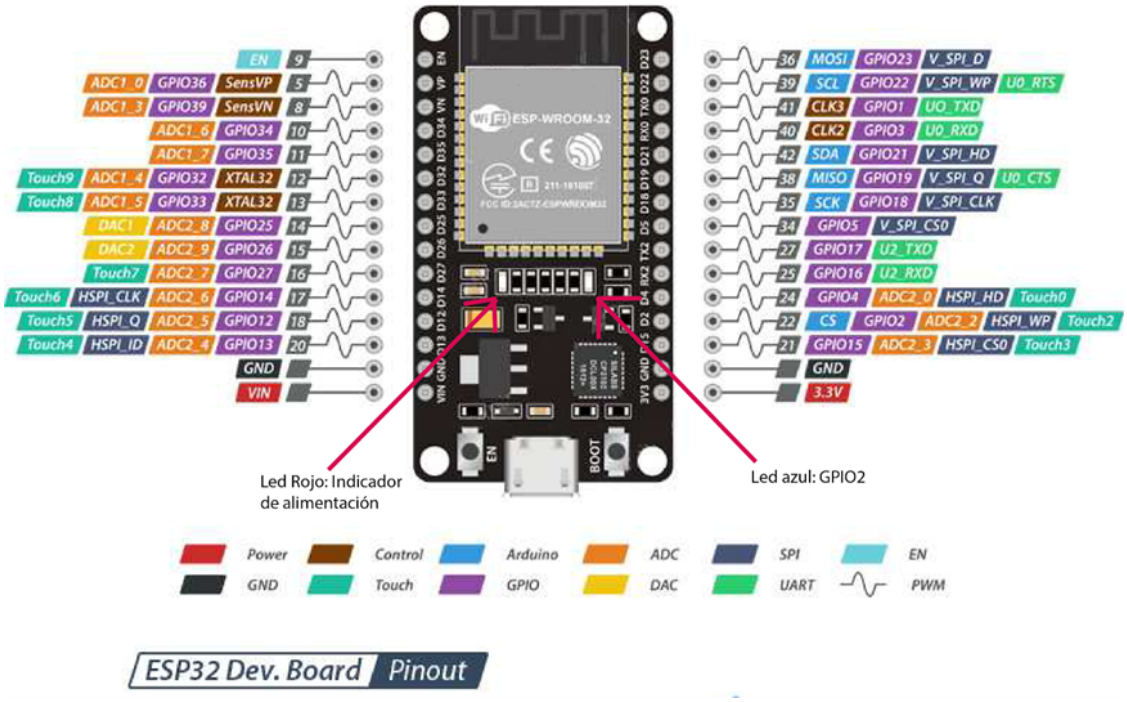


Figure 1: ESP32 Devkit V1

El ESP32 Devkit V1 tiene una cantidad total de 30 pines GPIO disponibles para su uso.

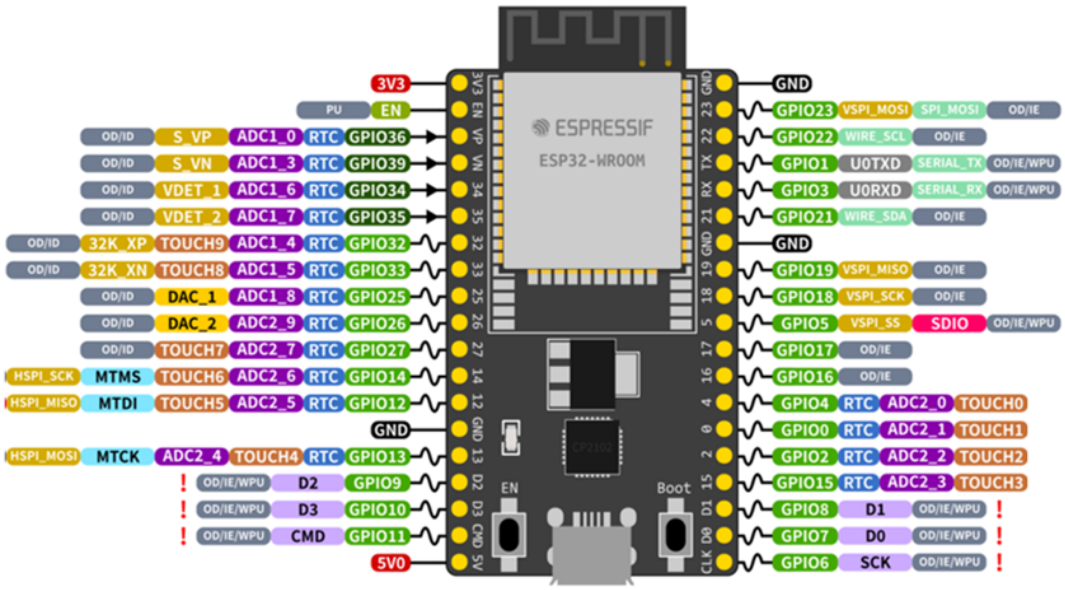


Figure 2: ESP32 Devkit V4

El ESP32 Devkit V4 tiene una cantidad total de 38 pines GPIO disponibles para su uso.

1.1. Niveles de tensión GPIO

Los niveles de tensión de los pines GPIO del ESP32-DevKitC son los siguientes:

- Alimentación (VCC): La alimentación de la tarjeta se va a 5V por el puerto USB y también por el canal marcado como 5V o VIN (Dependiendo de la versión).
- Pines de entrada y salida (GPIO): Los pines GPIO del ESP32 son compatibles con niveles de tensión de 3.3V. Esto significa que los pines pueden ser configurados como entradas o salidas digitales y funcionar con señales de 0V (lógica LOW) a 3.3V (lógica HIGH).
- Pines analógicos (ADC): El ESP32 tiene pines analógicos que también son compatibles con niveles de tensión de 0V a 3.3V. La conversión analógica a digital (ADC) del ESP32 mapea los valores analógicos a un rango digital de 0 a 4095 (12 bits), lo que corresponde a la tensión de 0 a 3.3V.

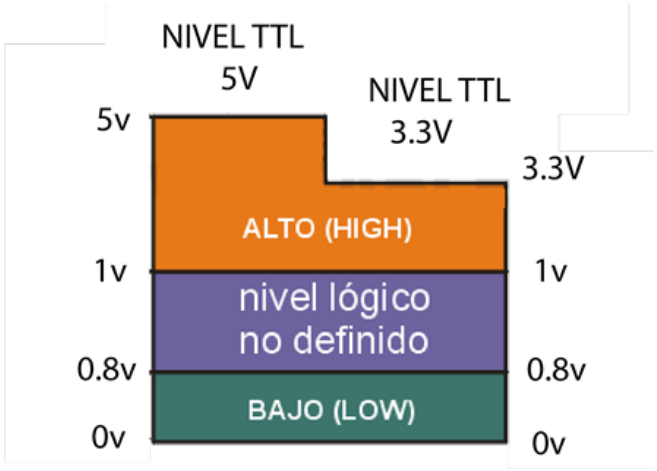


Figure 3: Niveles lógicos de tensión TTL

1.2. Corriente GPIO

En general, los pines GPIO del ESP32 están diseñados para proporcionar y tolerar una corriente máxima de alrededor de 12 mA en modo de salida. Es importante tener en cuenta que el suministro o consumo de corriente en los pines GPIO también está relacionado con la tensión de alimentación del chip.
Algunos pines GPIO también pueden configurarse como pines de corriente alta (High-Current GPIOs), que pueden suministrar una corriente máxima más alta, generalmente alrededor de 40 mA. Estos pines están destinados para aplicaciones que requieran mayor capacidad de corriente, como encender y apagar LEDs o manejar otros dispositivos de potencia.
Además, el suministro de corriente total del ESP32, incluyendo los pines GPIO, no debe exceder las especificaciones del regulador de voltaje o la fuente de alimentación utilizada para alimentar el dispositivo.

2. Descripción de las funciones para GPIO

API GPIO: Aquí están las funciones comunes utilizadas para el periférico GPIO.

2.1. pinMode

pinMode: La pinModefunción se usa para definir el modo de operación GPIO para un pin específico.

- void pinMode(uint8_t pin, uint8_t mode);
  - pin define el número de pin GPIO.
  - mode establece el modo de funcionamiento.
- Los siguientes modos son compatibles con la entrada y salida básicas:
  - INPUT establece el GPIO como entrada sin pullup o pulldown (alta impedancia).
  - OUTPUT configura el GPIO como modo de salida/lectura.
  - INPUT_PULLDOWN establece el GPIO como entrada con el menú desplegable interno.
  - INPUT_PULLUP establece el GPIO como entrada con el pullup interno.

2.2. Pullup y Pulldown interno

Las familias de SoC ESP32 admiten el pullup y pulldown interno a través de una resistencia de 45kR, que se puede habilitar al configurar el modo GPIO como INPUTmodo. Si el modo pullup o pulldown no está definido, el pin permanecerá en el modo de alta impedancia.

2.3. Escritura digital

La función digitalWrite establece el estado del GPIO seleccionado en HIGHo LOW. Esta función solo se usa si se pinMode configuró como OUTPUT.

- void digitalWrite(uint8_t pin, uint8_t val);
  - pin define el número de pin GPIO.
  - val establezca el estado digital de salida en HIGHo LOW.

2.4. Lectura digital

Para leer el estado de un pin determinado configurado como, se utiliza INPUT la función digitalRead

- int digitalRead(uint8_t pin);
  - pin seleccione GPIO

Esta función devolverá el estado lógico del pin seleccionado como HIGHo LOW.

3. Descripción de actuadores, sensores y transductores

3.1. Actuadores

Los actuadores son componentes o dispositivos que convierten una señal o instrucción en un movimiento físico o una acción en el mundo real. Estos dispositivos son fundamentales en sistemas de automatización y control, ya que permiten que los sistemas electrónicos interactúen con el entorno físico.
Los actuadores son esenciales en numerosos campos, desde la robótica y la industria manufacturera hasta los sistemas de automatización en el hogar. Permiten la automatización de tareas, mejoran la eficiencia y aumentan la precisión en diversas aplicaciones. Además, son un componente crucial en la integración entre sistemas electrónicos y el mundo físico.
Existen varios tipos de actuadores, cada uno diseñado para realizar una acción específica:

- Actuadores eléctricos: Convierten una señal eléctrica en movimiento mecánico. Algunos ejemplos son los motores eléctricos, los solenoides y los relés.
- Actuadores hidráulicos: Utilizan un fluido, generalmente aceite, para generar fuerza y movimiento. Se emplean en maquinaria pesada, sistemas de dirección hidráulica, entre otros.
- Actuadores piezoeléctricos: Se basan en materiales piezoeléctricos que generan movimiento cuando se aplica una tensión eléctrica. Son utilizados en dispositivos de precisión y sistemas microelectromecánicos (MEMS).
- Actuadores electromagnéticos: Emplean campos magnéticos para producir movimiento en una bobina o imán. Los altavoces y relés son ejemplos de actuadores electromagnéticos.
- Actuadores térmicos: Utilizan cambios de temperatura para producir movimiento. Un ejemplo común es el actuador bimetálico presente en termostatos y sistemas de control de temperatura.

Figure 4: Motor paso a paso

Figure 5: Motor DC

Figure 6: Micro Servo SG90

Figure 7: Relé electromecánico SRD-05VDC-SL-C 5V

Figure 8: Luces LED

Figure 9: LCD 16x2

3.2. Sensores

Los sensores son dispositivos o elementos que capturan información del entorno físico y la convierten en señales eléctricas o digitales que pueden ser interpretadas y utilizadas por sistemas electrónicos o computadoras. Estos dispositivos son fundamentales para adquirir datos del mundo real y proporcionar información esencial para diversas aplicaciones.
Existen muchos tipos de sensores, cada uno diseñado para detectar y medir diferentes variables físicas o ambientales.

- Sensores digitales:
  - Sensores de proximidad: Detectan la presencia o ausencia de objetos cercanos sin necesidad de contacto físico. Ejemplo: sensores de proximidad infrarrojos.
  - Sensores de movimiento: Detectan el movimiento o cambios en la posición de un objeto o persona. Ejemplo: sensores de movimiento por infrarrojos (PIR).
  - Sensores de luz: Detectan la intensidad de la luz ambiente y generan una señal digital correspondiente. Ejemplo: sensores de luz ambiental.
  - Sensores de sonido: Convierten las vibraciones del sonido en señales digitales. Ejemplo: micrófonos digitales.
  - Sensores de humedad digital: Miden la cantidad de humedad presente en el aire o en un medio específico y proporcionan lecturas digitales. Ejemplo: sensores de humedad y temperatura DHT11 o DHT22.
  - Sensores de gas digital: Detectan la presencia y concentración de gases específicos en el aire y proporcionan salidas digitales. Ejemplo: sensores de monóxido de carbono (CO) y dióxido de carbono (CO2) digitales.
- Sensores analógicos:
  - Sensores de temperatura analógicos: Detectan la temperatura ambiente y generan una señal analógica proporcional. Ejemplo: sensores de temperatura termistores o termopares.
  - Sensores de presión analógicos: Miden la presión del aire o líquidos y producen una señal analógica correspondiente. Ejemplo: sensores de presión piezorresistivos.
  - Sensores de luz analógicos: Detectan la intensidad de la luz ambiente y generan una señal analógica proporcional. Ejemplo: fotodiodos o LDR (resistencias dependientes de la luz).
  - Sensores de humedad analógicos: Miden la cantidad de humedad presente en el aire o en un medio específico y proporcionan una señal analógica. Ejemplo: sensores de humedad capacitivos.
  - Sensores de aceleración analógicos: Detectan la aceleración lineal en una o más direcciones y producen una señal analógica proporcional. Ejemplo: acelerómetros analógicos.

  Es importante tener en cuenta que algunos sensores pueden tener versiones tanto digitales como analógicas, y la elección entre uno u otro dependerá de la aplicación específica y las necesidades del proyecto.

Figure 10: Sensor de humedad DHT11

Figure 11: Sensor de movimiento PIR HC-SR501

Figure 12: Sensor de presion BMP280

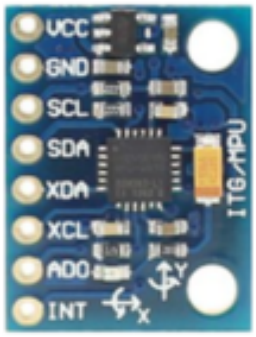


Figure 13: Sensor de movimiento y orientacion MPU6050

Figure 14: Sensor infrarrojo FC-51

Figure 15: Sensor de ultrasónico HC-SR04

3.3. Transductor

Los transductores son dispositivos que convierten una forma de energía en otra. Estos dispositivos son esenciales para la adquisición y conversión de diferentes tipos de señales o variables físicas en formas que puedan ser más útiles o comprensibles para los sistemas electrónicos o computadoras. En esencia, los transductores actúan como "traductores" que permiten que los sistemas electrónicos comprendan y respondan al mundo físico que les rodea.
Existen varios tipos de transductores, cada uno diseñado para convertir una forma específica de energía en otra. Algunos ejemplos comunes de transductores incluyen:

- Transductores eléctricos: Convierten una señal física en una señal eléctrica. Por ejemplo, los sensores piezoeléctricos convierten la presión mecánica en una señal eléctrica.
- Transductores mecánicos: Transforman una señal eléctrica en una señal mecánica. Un ejemplo es el actuador piezoeléctrico, que convierte una señal eléctrica en un movimiento mecánico.
- Transductores ópticos: Convierten una señal eléctrica en una señal óptica o viceversa. Los LED y los fotodiodos son ejemplos de transductores ópticos.
- Transductores acústicos: Transforman una señal eléctrica en una señal acústica o viceversa. Los micrófonos y los altavoces son transductores acústicos.
- Transductores térmicos: Convierten una señal eléctrica en una señal térmica o viceversa. Los termopares son ejemplos de transductores térmicos.

Figure 16: Sensor de temperatura termistor

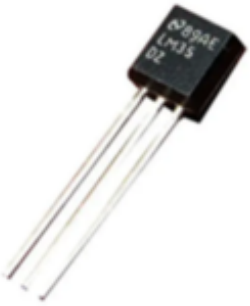


Figure 17: Sensor de temperatura LM35

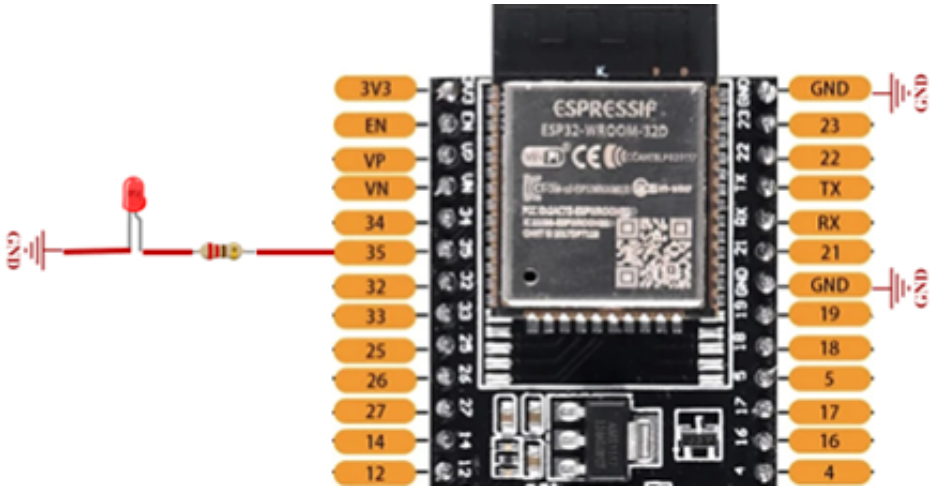
Figure 18: Sensor de luz fotodiodo

Figure 19: Micrófono

4. Ejemplos(Salidas digitales)

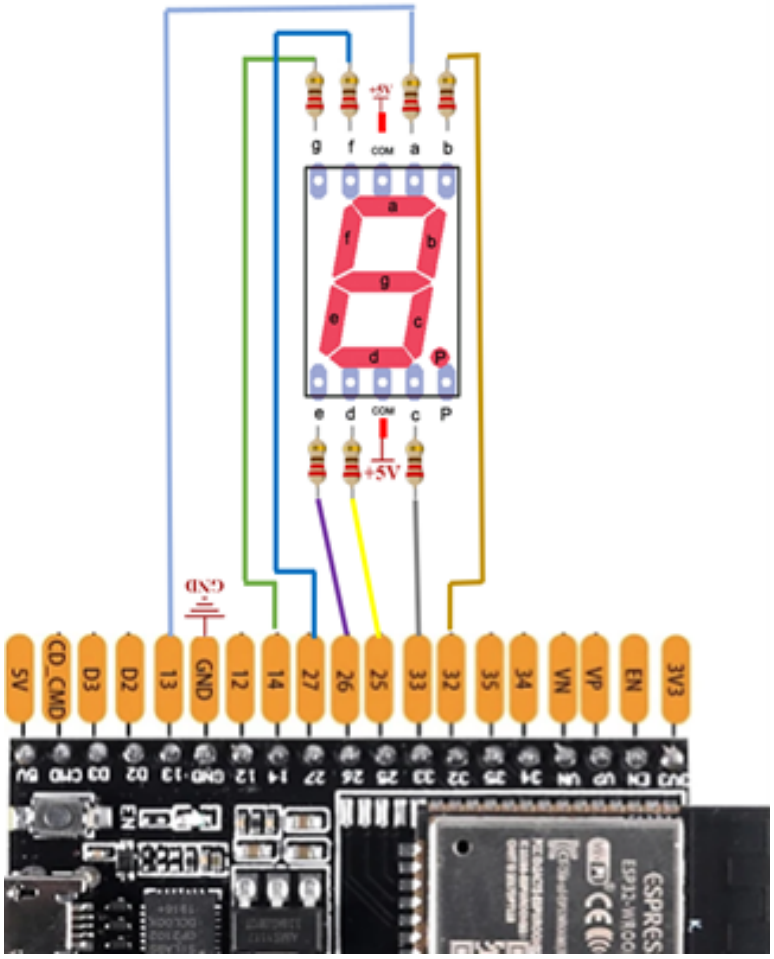
4.1. Prender y apagar un led

- Esquemático:



4.2. Contar del 0 al 9 en un display de 7 segmentos ánodo común

- Esquemático:



umaker

- Simulación:

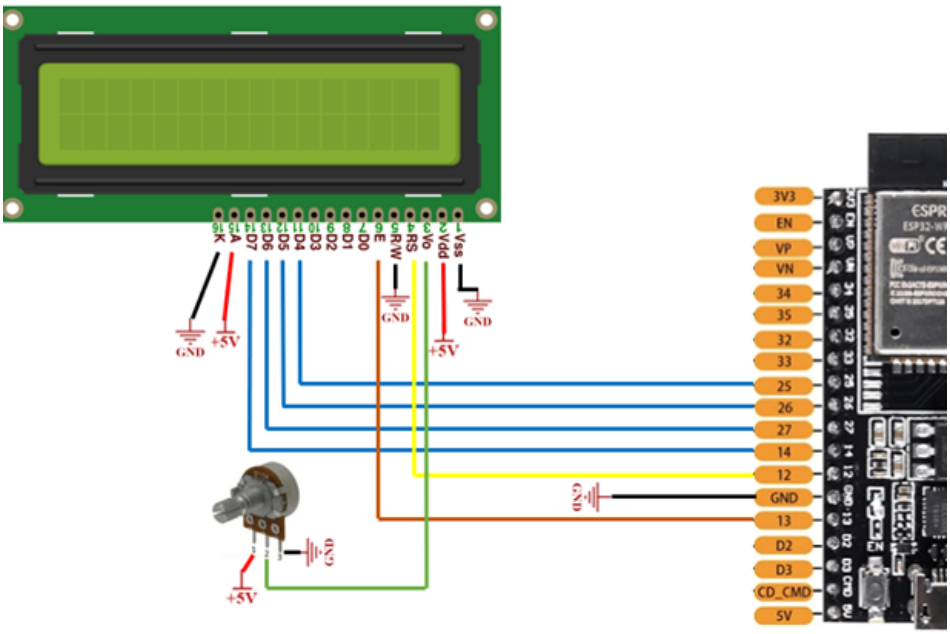
https://wokwi.com/projects/356223905049693185

- Programación:

```
//Colocamos los pines conectados al display como salida de la ESP32
uint8_t Display_7Seg_A[10] = {0XC0,0XF9,0XA4,0XB0,0X99,0X92,0X82,0XF8,0X80,0X90};

void Write_Display7seg(uint8_t val){
    uint8_t data=Display_7Seg_A[val];
    digitalWrite(13, LOW);
    digitalWrite(32, LOW);
    digitalWrite(33, LOW);
    digitalWrite(25, LOW);
    digitalWrite(26, LOW);
    digitalWrite(27, LOW);
    digitalWrite(14, LOW);

    if(data & 1<<0) digitalWrite(13, HIGH);//a
    if(data & 1<<1) digitalWrite(32, HIGH);//b
    if(data & 1<<2) digitalWrite(33, HIGH);//c
    if(data & 1<<3) digitalWrite(25, HIGH);//d
    if(data & 1<<4) digitalWrite(26, HIGH);//e
    if(data & 1<<5) digitalWrite(27, HIGH);//f
    if(data & 1<<6) digitalWrite(14, HIGH);//g
    return;
}


void setup() {
  pinMode (27, OUTPUT);
  pinMode (14, OUTPUT);
  pinMode (26, OUTPUT);
  pinMode (25, OUTPUT);
  pinMode (33, OUTPUT);
  pinMode (32, OUTPUT);
  pinMode (13, OUTPUT);
}
void loop()
{
  for(int i=0; i<=9; i++)
  {
    Write_Display7seg(i);
    delay(2000);
  }
}
```

4.3. "Hola mundo" en LCD 16x2

- Esquemático:



- Simulación:

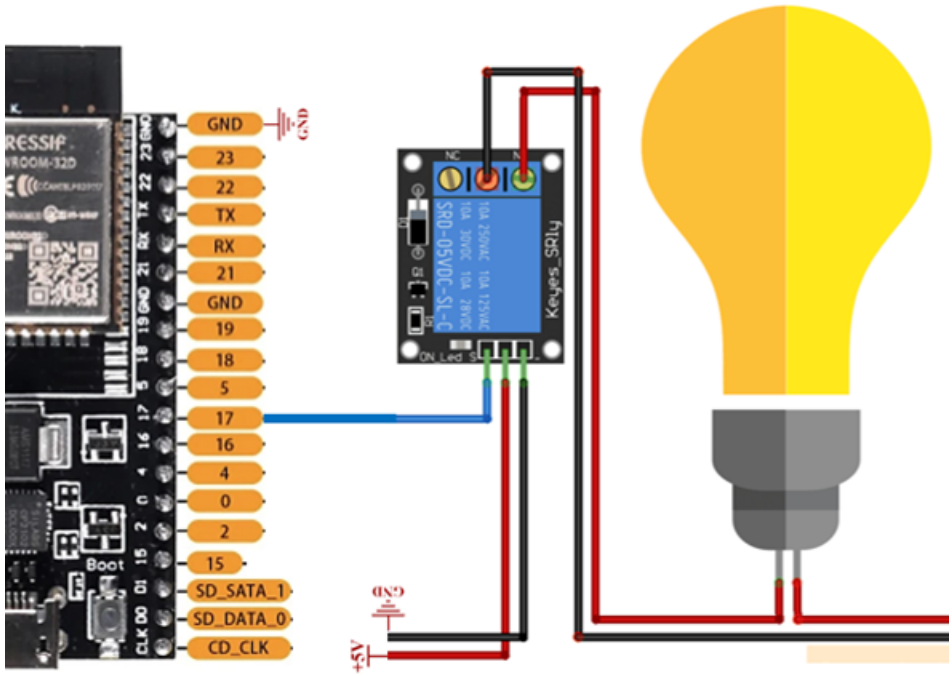
https://wokwi.com/projects/372180336724944897

- Programación:

```
#include <LiquidCrystal.h>

const int rs = 13, en = 12, d4 = 14, d5 = 27, d6 = 26, d7 = 25;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
  lcd.begin(16, 2);
}

void loop()
{
  lcd.setCursor(3,0);
  lcd.print("Hola mundo");
}
```

4.4. Prender y apagar un foco

- Esquemático:



- Simulación:

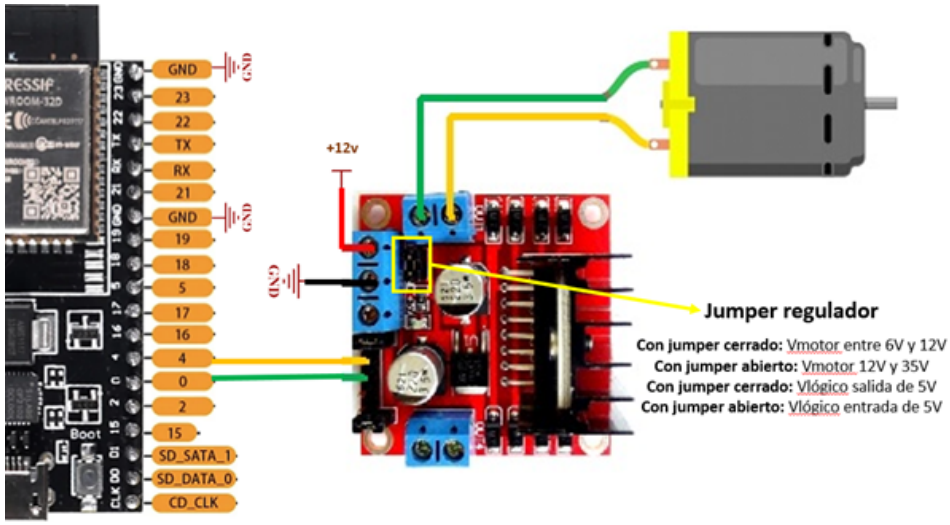
https://wokwi.com/projects/372181325192389633

- Programación:

```
int relay = 17;

void setup(){
  pinMode (relay, OUTPUT);
  Serial.begin(9600);
}
void loop(){
  digitalWrite(relay, HIGH);
  Serial.println("Relay accionado");
  delay(1000);

  digitalWrite(relay, LOW);
  Serial.println("Relay no accionado");
  delay(1000);
}
```

4.5. Girar el eje de un motor DC en sentido horario

- Esquemático:

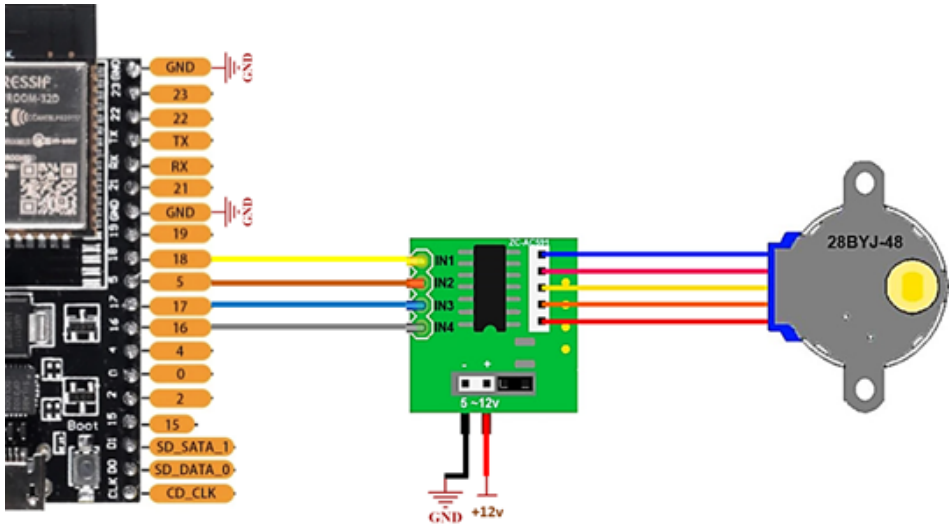


- Programación:

```
int speedPin = 11;
int dirPin1 = 12;
int dirPin2 = 13;
int speedMotor = 80;
int dt = 50;

void setup()
{
  Serial.begin(9600);
  pinMode (speedPin, OUTPUT);
  pinMode (dirPin1, OUTPUT);
  pinMode (dirPin2, OUTPUT);
}

void loop() {
  digitalWrite (dirPin1, 0);
  digitalWrite (dirPin2, 1);
  digitalWrite (speedPin, 255);
}
```

4.6. Motor paso a paso

- Esquemático:



- Simulación:

https://wokwi.com/projects/356228856059735041

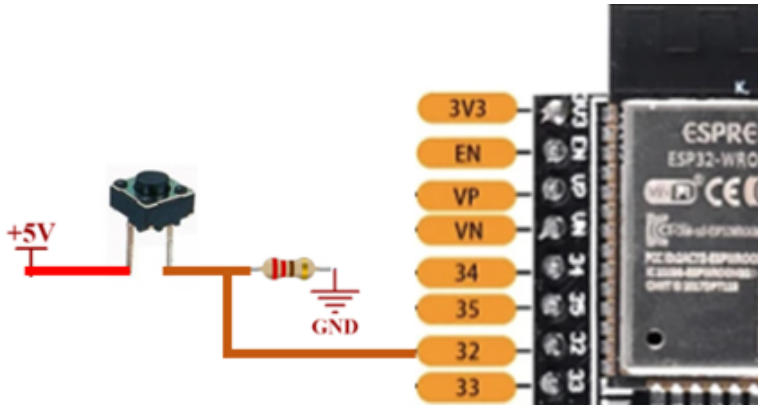
- Programación:

```
#include <Stepper.h>

int stepsPerRevolution = 2048; //Pasos por revolucion
int motSpeed = 10; //Velocidad del motor
Stepper myStepper (stepsPerRevolution,8,10,9,11);
int dt = 500;

void setup()
{
  myStepper.setSpeed(motSpeed);
}

void loop()
{
  myStepper.step(stepsPerRevolution);
  delay(dt);
  myStepper.step(-stepsPerRevolution);
  delay(dt);
}
```

5. Ejemplos(Entradas digitales)

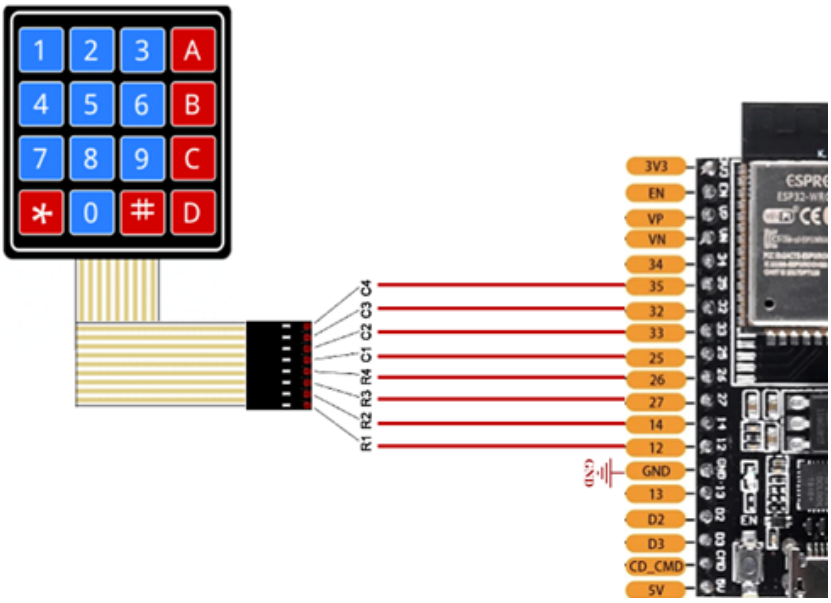
5.1. Mostrar en estado de un pulsador en el monitor serial

- Esquemático:



5.2. Mostrar los valores del Keypad en el monitor serial

- Esquemático:



- Simulación:

https://wokwi.com/projects/356208244800351233

- Programación:

```
#include <Keypad.h>

const byte Filas = 4;
const byte Columnas = 4;

char keys[Filas][Columnas] =
{
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};

byte filasPins[Filas] = {13, 12, 14, 27};
byte colPins[Columnas] = {26, 25, 33, 32};

Keypad teclado=
Keypad(makeKeymap(keys), filasPins, colPins, Filas, Columnas);
char key;


void setup(){
  Serial.begin(9600);
}

void loop()
{
  key = teclado.getKey();
  if(key)
  {
    Serial.println(key);
  }
}
```