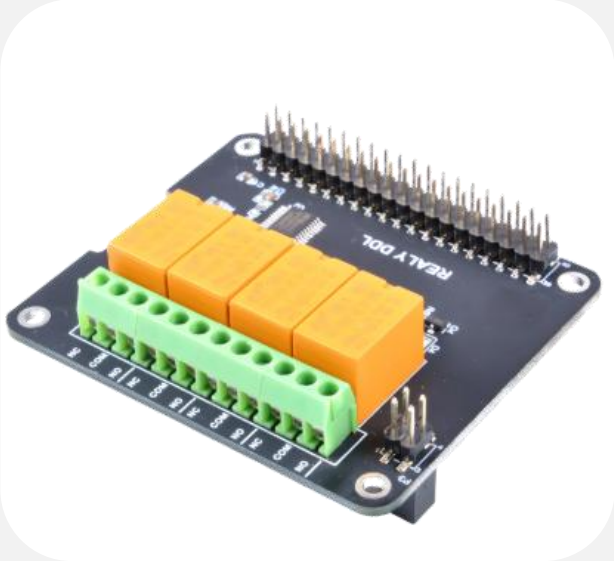
HAT DE 4 RELEVADORES CON BORNERAS – CONTROL DE POTENCIA PARA RASPBERRY PI 5, 4 Y 3B

EP-0099



Productos evaluados por **ingenieros calificados**

Garantía y seguridad en cada producto

Experiencia de compra en la **calidad** como sello distintivo

Descripción

El HAT EP-0099 cuenta con 4 relevadores los cuales puedes integrar en tu proyecto de hogar inteligente. Se pueden utilizar varios módulos apilando hasta cuatro capas. Cambie la dirección I2C del dispositivo mediante el interruptor DIP en la placa de circuito impreso para garantizar que el módulo de cuatro capas obtenga direcciones diferentes, evitando así el problema de duplicación de direcciones. El módulo es muy rápido, fácil de instalar y admite el desarrollo en varios idiomas, lo que lo hace ideal para proyectos de hogares inteligentes.

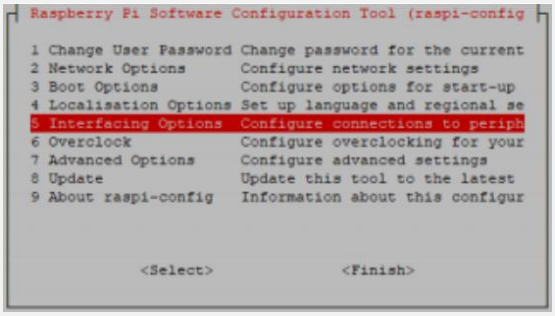
Características

- Serie DockerPi.
- Programable.
- Control directo (sin programación).
- Extensión de los pines GPIO.
- Compatibilidad con 4 direcciones I2C alternativas.
- LED indicador de estado del relevador.
- Admite corriente alterna de hasta 3A y 250VCA.
- Admite corriente directa de hasta 3A y 30VCD.
- Se puede apilar con otros HATs compatibles.
- Independiente del hardware de la placa base (requiere soporte I2C).
- Compatibilidad con cables de 24 AWG a 30 AWG.

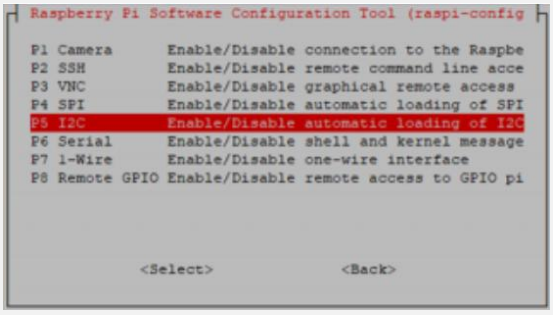
Configuración de I2C (Raspberry Pi)

Ejecute **sudo raspi-config** y siga las instrucciones para instalar el soporte I2C para ARM core y Linux kernel.

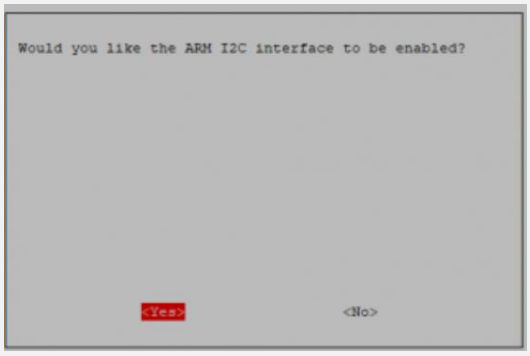
1. Seleccione **Interfacing Options:**



2. Depues seleccione la opción P5 **I2C:**



3. Mostrará el siguiente mensaje, seleccione la opción **Yes:**



4. Finalmente mostrará el siguiente mensaje y quedará abilitado:

Control directo sin programación (Raspberry Pi)

Para canal 1:

- Encendido: **`i2cset -y 1 0x10 0x01 0xFF`**
- Apagado: **`i2cset -y 1 0x10 0x01 0x00`**

Para canal 2:

- Encendido: **`i2cset -y 1 0x10 0x02 0xFF`**
- Apagado: **`i2cset -y 1 0x10 0x02 0x00`**

Para canal 3:

- Encendido: **`i2cset -y 1 0x10 0x03 0xFF`**
- Apagado: **`i2cset -y 1 0x10 0x03 0x00`**

Para canal 4:

- Encendido: **`i2cset -y 1 0x10 0x04 0xFF`**
- Apagado: **`i2cset -y 1 0x10 0x04 0x00`**

Programación en lenguaje C (Raspberry Pi)

```
#include <stdio.h>
#include <wiringPi.h>
#include <wiringPiI2C.h>

#define DEVCIE_ADDR  0x10
#define RELAY1  0x01
#define RELAY2  0x02
#define RELAY3  0x03
#define RELAY4  0x04
#define ON      0xFF
#define OFF     0x00

int main(void){
    printf("Turn on Relays in C\n");
    int fd;
    int i = 0;
    fd = wiringPiI2CSetup(DEVICE_ADDR);
    for(;;){
       for (i=1; i<=4; i++){
          printf("turn on relay No.%d\n", i);
          wiringPiI2CWriteReg8(fd, i, ON);
          sleep(200);
          printf("turn off relay No.%d\n", i);
          wiringPiI2CWriteReg8(fd, i, OFF);
          sleep(200);
       }
    }
    return 0;
}
```

Programación en Python (Raspberry Pi)

```
import time as t
import smbus
import sys

DEVICE_BUS = 1
DEVICE_ADDR = 0x10
bus = smbus.SMBus(DEVICE_BUS)

while True:
    try:
        for i in range(1,5):
            bus.write_byte_data(DEVICE_ADDR, i, 0xFF)
            t.sleep(1)
            bus.write_byte_data(DEVICE_ADDR, i, 0x00)
            t.sleep(1)
    except KeyboardInterrupt as e:
        print("Quit the Loop")
        sys.exit()
```

Programación en Java (Raspberry Pi)

```
import java.io.IOException;
import java.util.Arrays;

import com.pi4j.io.i2c.I2CBus;
import com.pi4j.io.i2c.I2CDevice;
import com.pi4j.io.i2c.I2CFactory;
import com.pi4j.io.i2c.I2CFactory.UnsupportedBusNumberException;
import com.pi4j.platform.PlatformAlreadyAssignedException;
import com.pi4j.util.Console;

public class I2CRelay {

    // relay's register address.
    public static final int DOCKER_PI_RELAY_ADDR = 0x10;

    // channel of relay.
    public static final byte DOCKER_PI_RELAY_1 = (byte)0x01;
    public static final byte DOCKER_PI_RELAY_2 = (byte)0x02;
    public static final byte DOCKER_PI_RELAY_3 = (byte)0x03;
    public static final byte DOCKER_PI_RELAY_4 = (byte)0x04;

    // Relay status
    public static final byte DOCKER_PI_RELAY_ON = (byte)0xFF;
    public static final byte DOCKER_PI_RELAY_OFF = (byte)0x00;

    public static void main(String[] args) throws InterruptedException, PlatformAlreadyAssignedException, IOException, UnsupportedBusNumberException {

        final Console console = new Console();

        I2CBus i2c = I2CFactory.getInstance(I2CBus.BUS_1);
        I2CDevice device = i2c.getDevice(DOCKER_PI_RELAY_ADDR);

        console.println("Turn on Relay!");
        device.write(DOCKER_PI_RELAY_1, DOCKER_PI_RELAY_ON);

        Thread.sleep(500);

        console.println("Turn off Relay!");
        device.write(DOCKER_PI_RELAY_1, DOCKER_PI_RELAY_OFF);
    }
}
```

Contenido

- 1 Docker Pi de 4 canales.
- 1 Instrucciones.
- 4 Pilares de cobre.
- 4 Tuercas M2.5*6.
- 4 Tornillos de cabeza semicircular M2.5*6.

Notas adicionales

El conector de 4 pines es para una fuente de alimentación de respaldo para futuros productos. Puede ignorar estos pines. No conecte ningún dispositivo a ellos.