CEC Continuing Education Center

DesignNews

Understanding Industrial Controls with an ESP32

Day 2: Digital Output Signal Conditioning

Sponsored by

DigiKey

informa markets

Webinar Logistics

- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Attendee Chat’ by maximizing the chat widget in your dock.

Dr. Don Wilcher

Visit ‘Lecturer Profile’ in your console for more details.

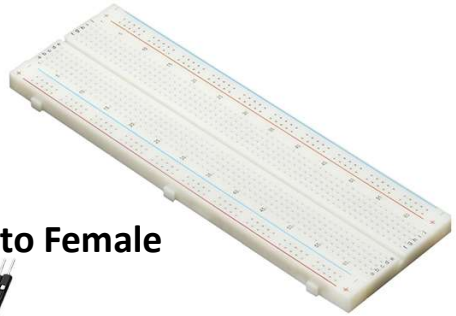
Course Kit and Materials

ESP32-DEVKITC-VIE

Adafruit Parts Pal Kit

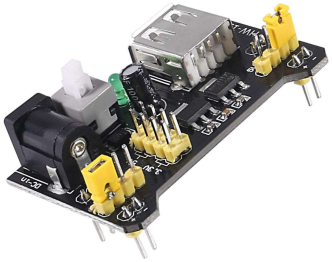
Jumper Wires: Male to Male

Solderless Breadboard x2



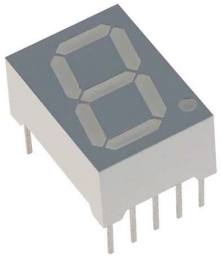
Jumper Wires: Male to Female

Solderless Breadboard Power Supply



5V Relay Module, 5V Indicator Light LED

7 Segment LED Display, Common Cathode



Standard Motor, 9100 RPM 6VDC

Agenda:

- MicroPython Introduction
- MicroPython REPL Programming Examples
- Digital Output Signal Conditioning
- Lab: Build An ESP32 DC Motor Controller

Research Perspective

"Programmable logic controllers (PLCs) provide an ecosystem of relatively simple software logic, robust and ruggedized hardware, networks with controllable real-time behaviors, and extensive availability of interoperable components such as sensors and actuators" (Sehr et al., 2021).

Course Question
Can an ESP32 microcontroller contribute to the Industrial Controls field?

MicroPython Introduction

- MicroPython is a lean and efficient Python 3 programming language implementation.
- Python 3 includes a small subset of the Python standard library.
- Python 3 is optimized to run on microcontrollers and in constrained environments.
- MicroPython is written in the C language.
- MicroPython includes advanced features such as:
  a) interactive prompt (REPL)
  b) precision integers
  c) and exception handling.
- MicroPython aims to be as compatible with the traditional Python.

MicroPython Introduction. . .

- With such compatibility, MicroPython allows the transfer of code from a desktop development machine to a microcontroller or embedded system.
- This code transfer is based on MicroPython's ability to run with 256K of programming space and 16K of RAM.
- MicroPython is bare metal where the code runs on the target microcontroller without an operating system (OS).
- The creator of MicroPython is Damien George.
- MicroPython's initial release date was released May 3rd, 2014 (10 years ago).

Question 1

What was the initial release date for MicroPython?

a) May 3rd, 2015
b) May 3rd, 2014
c) May 3rd, 2012
d) May 3rd, 2016

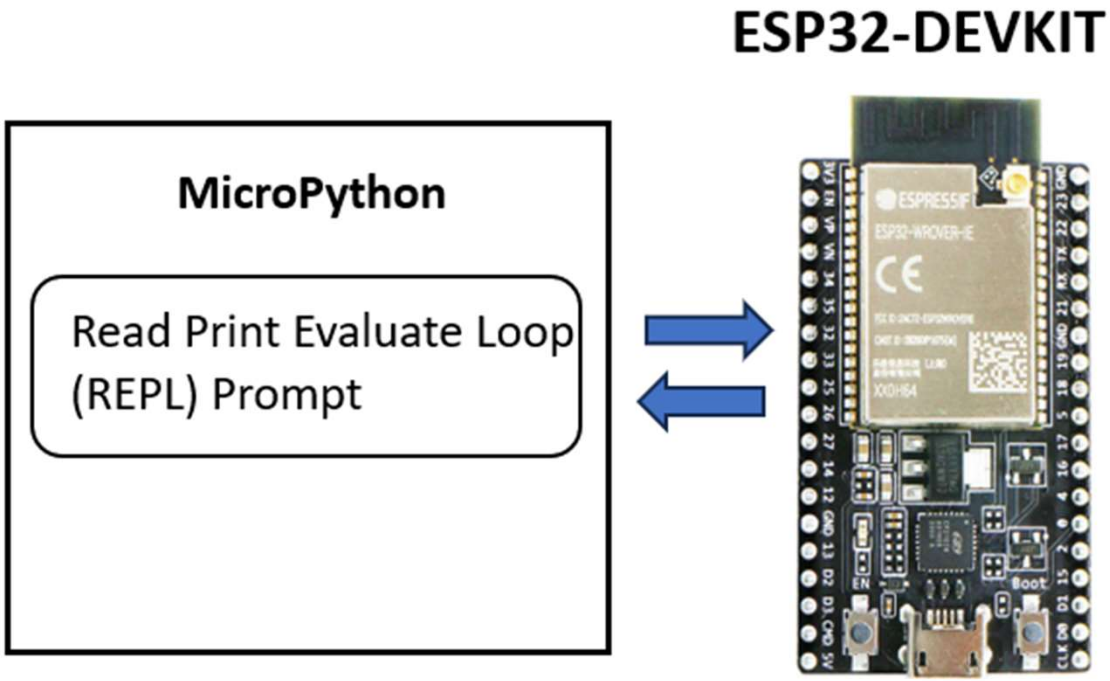
MicroPython REPL Programming Examples

What is REPL?

- REPL stands for Read Evaluate Print Loop.
- REPL is the interactive MicroPython prompt that you can access on the ESP32 microcontroller.
- Using the REPL is by far the easiest way to test out your code and run commands.

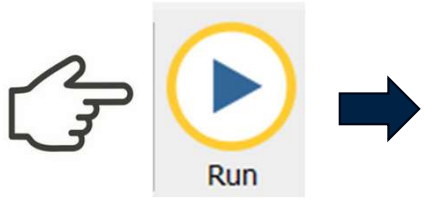
MicroPython REPL Programming Examples. . .

What is REPL?



MicroPython REPL Programming Examples. . .

Starting a REPL Session:



ESP MicroPython REPL

```
MicroPython v1.24.0 on 2024-10-25; Generic ESP32 module with ESP32
Type "help()" for more information.
>>>
>>>
>>>
>>>
raw REPL; CTRL-B to exit
>OK
MPY: soft reboot
raw REPL; CTRL-B to exit
>OK

◆◆>
MicroPython v1.24.0 on 2024-10-25; Generic ESP32 module with ESP32
Type "help()" for more information.
>>>
```

MicroPython REPL Programming Examples. . .

Starting a REPL Session:

Type help ( )

```
MicroPython v1.24.0 on 2024-10-25; Generic ESP32 module with ESP32
Type "help()" for more information.
>>> help()
Welcome to MicroPython on the ESP32!

For online docs please visit http://docs.micropython.org/

For access to the hardware use the 'machine' module:

import machine
pin12 = machine.Pin(12, machine.Pin.OUT)
pin12.value(1)
pin13 = machine.Pin(13, machine.Pin.IN, machine.Pin.PULL_UP)
print(pin13.value())
i2c = machine.I2C(scl=machine.Pin(21), sda=machine.Pin(22))
i2c.scan()
i2c.writeto(addr, b'1234')
i2c.readfrom(addr, 4)
```

MicroPython – ESP32 Commands will be listed!

MicroPython REPL Programming Examples. . .

Starting a REPL Session:

Additional MicroPython – ESP32 Commands will be listed!

```
Basic WiFi configuration:

import network
sta_if = network.WLAN(network.STA_IF); sta_if.active(True)
sta_if.scan()                             # Scan for available access points
sta_if.connect("<AP_name>", "<password>") # Connect to an AP
sta_if.isconnected()                      # Check for successful connection

Control commands:
  CTRL-A        -- on a blank line, enter raw REPL mode
  CTRL-B        -- on a blank line, enter normal REPL mode
  CTRL-C        -- interrupt a running program
  CTRL-D        -- on a blank line, do a soft reset of the board
  CTRL-E        -- on a blank line, enter paste mode

For further help on a specific object, type help(obj)
For a list of available modules, type help('modules')
>>> is connected
```

Question 2

On slide 15, which MicroPython command allows for scanning available Bluetooth Access Points?

a) sta_if=network.WLAN(network.STA_IF)
b) sta_if.scan()
c) sta_if.isconnected ()
d) none of the above

MicroPython REPL Programming Examples. . .

Starting a REPL Session:

Type help ('modules')

```
>>> help('modules')
__main__          bluetooth         heapq             select
_asyncio          btree             inisetup          socket
_boot             builtins          io                ssl
_espnow           cmath             json              struct
_onewire          collections       machine           sys
_thread           cryptolib         math              time
_webrepl          deflate           micropython       tls
aioespnow         dht               mip/__init__      uasyncio
apa106            ds18x20           neopixel          uctypes
array             errno             network           umqtt/robust
asyncio/__init__  esp               ntptime           umqtt/simple
asyncio/core      esp32             onewire           upysh
asyncio/event     espnow            os                urequests
asyncio/funcs     flashbdev         platform          vfs
asyncio/lock      framebuf          random            webrepl
asyncio/stream    gc                re                webrepl_setup
binascii          hashlib           requests/__init__ websocket
Plus any modules on the filesystem
>>>
```

MicroPython – ESP32 modules will be listed!

MicroPython REPL Programming Examples. . .

Starting a REPL Session:

Storing and Printing numeric data in variables.

```
>>> a=5
>>> b=6
>>> print a, b
Traceback (most recent call last):
  File "<stdin>", line 1
SyntaxError: invalid syntax
>>> print (a,b)
5 6
>>>
```

MicroPython REPL Programming Examples. . .

Starting a REPL Session:

Storing and Printing string data in variable.

```
>>> word = "Hello World!!"
>>> print(word)
Hello World!!
>>>
```

MicroPython REPL Programming Examples. . .

Starting a REPL Session:

Basic Math Functions

Note:

```
>>> a=5
>>> b=6
```

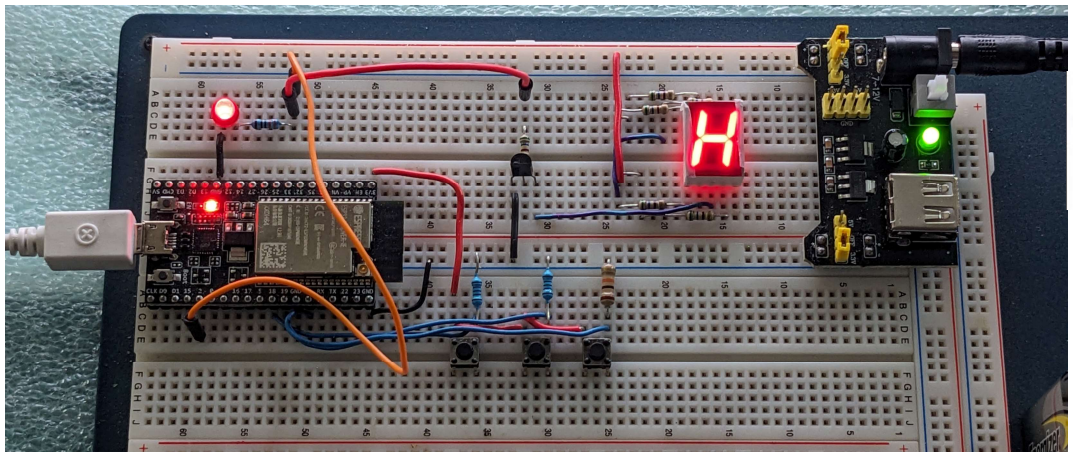
```
>>> print(a+b)
11
>>> print(a*b)
30
>>> print(a-b)
-1
>>> print(a/b)
0.8333333
>>>
```

MicroPython REPL Programming Examples. . .

Starting a REPL Session:

Turning ON the LED and 7-Segment LED Display

```
>>> import machine
>>> pin2 = machine.Pin(2,machine.Pin.OUT)
>>> pin2.value(1)
```



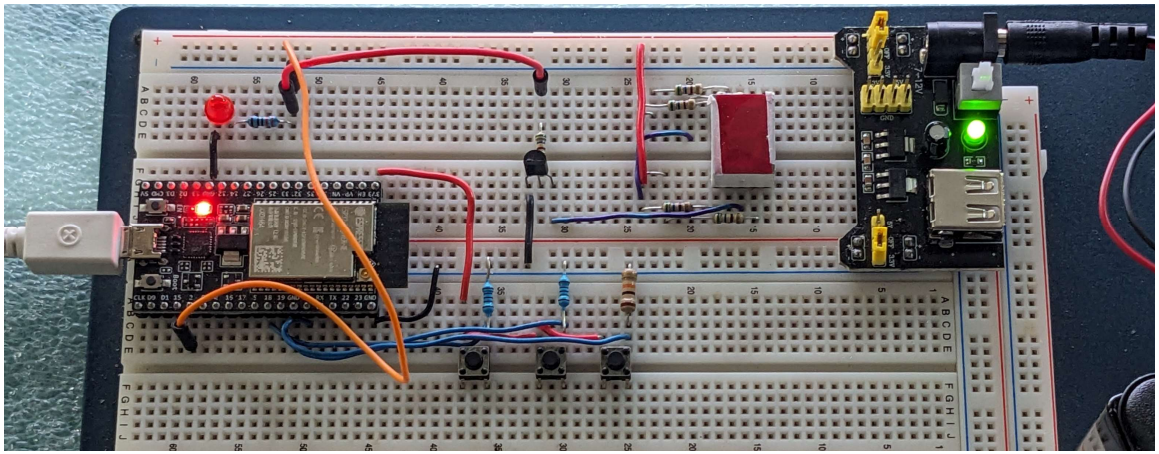
Attach the 9V Battery and turn ON the Breadboard power supply!

MicroPython REPL Programming Examples. . .

Starting a REPL Session:

Turning OFF the LED and 7-Segment LED Display

```
>>> pin2.value(0)
```



Attach the 9V Battery and turn ON the Breadboard power supply!

Question 3

Which MicroPython command is used to turn ON the LED and the 7-Segment LED Display?

a) pin2.value(0)
b) pin2.value(1)
c) pin2.value(ON)
d) pin2.value(OFF)

Digital Output Signal Conditioning

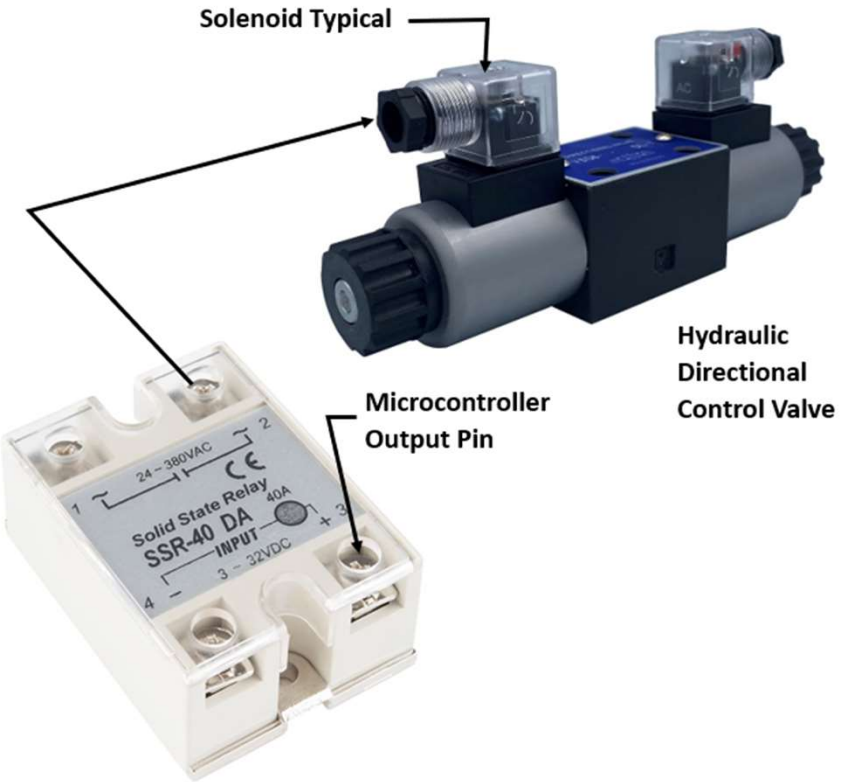
- The outputs of microcontrollers can be used to control the status of output field devices.
- Output devices perform the work or provide motion in an industrial or process control application.
- Another word used to describe an output device is an actuator.
- Examples of actuators are
  a) motors
  b) solenoids
  c) valves
  d) electromechanical relays.

Digital Output Signal Conditioning. . .

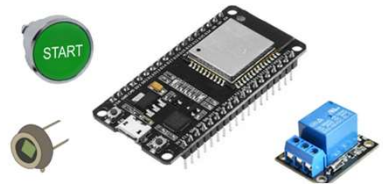
Combining solenoids with pneumatic or hydraulic valves allows precise control of the hybrid actuator using a microcontroller.

Digital Output Signal Conditioning. . .

Controlling an AC Electro-Hydraulic Directional Control Valve (DCV) (Actuator) with a Microcontroller.



Digital Output Signal Conditioning. . .



Appropriate signal conditioning design begins with a review of the characteristics and limitations of the ESP32 outputs.

VOH = 0.8 x VDD
= 2.64 V
VOL = 0.1 x VDD
= 0.1 x 3.3V
= 0.33V

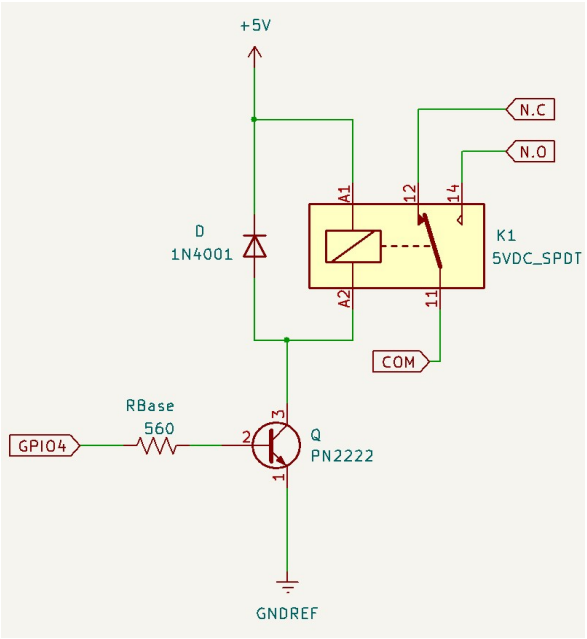
Parameter	Description		Min	Typ	Max	Unit
V_{OH}	High-level output voltage		0.8×VDD[1]	—	—	V
V_{OL}	Low-level output voltage		—	—	0.1×VDD[1]	V
I_{OH}	High-level source current (VDD[1] = 3.3 V, V_{OH} >= 2.64 V, output drive strength set to the maximum)	VDD3P3_CPU power domain [1, 2]	—	40	—	mA
		VDD3P3_RTC power domain [1, 2]	—	40	—	mA
		VDD_SDIO power domain [1, 3]	—	20	—	mA

2. For VDD3P3_CPU and VDD3P3_RTC power domain, per-pin current sourced in the same domain is gradually reduced from around 40 mA to around 29 mA, V_{OH}>=2.64 V, as the number of current-source pins increases.
3. For VDD_SDIO power domain, per-pin current sourced in the same domain is gradually reduced from around 30 mA to around 10 mA, V_{OH}>=2.64 V, as the number of current-source pins increases.

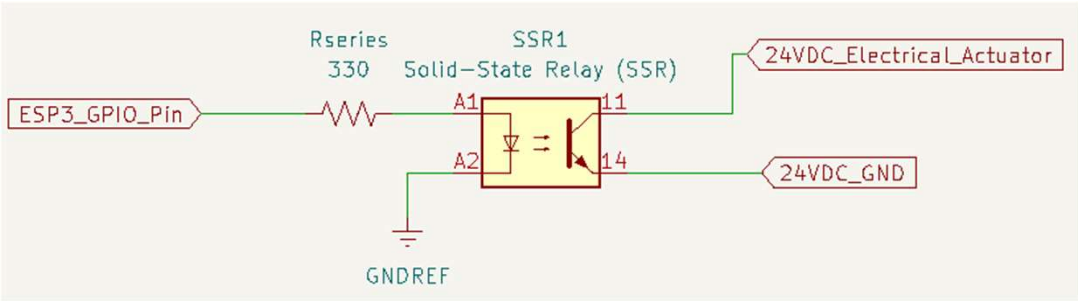
Digital Output Signal Conditioning. . .

Typical Digital Output Signal Conditioned Switching Circuits for Actuators.

Transistor Relay Driver Circuit



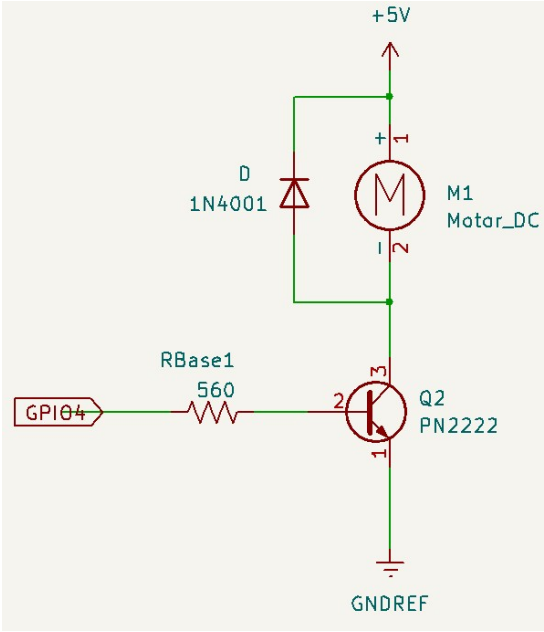
Solid State Relay



Digital Output Signal Conditioning. . .

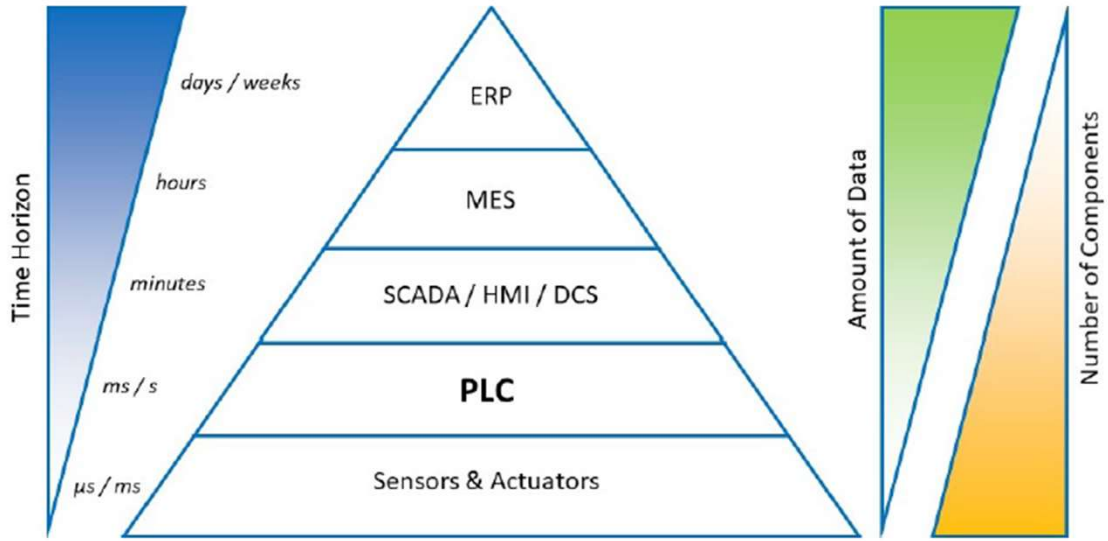
Typical Digital Output Signal Conditioned Switching Circuits for Actuators.

Transistor Motor Driver Circuit



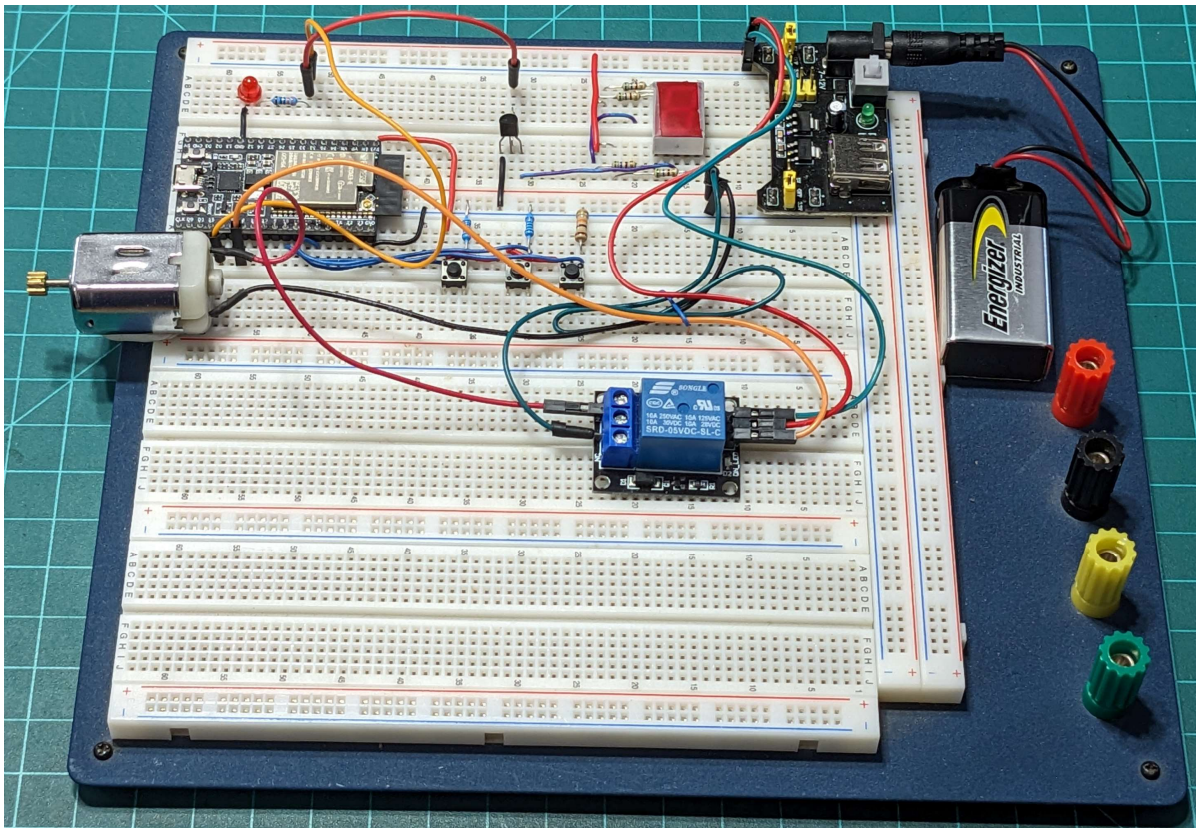
Digital Output Signal Conditioning. . .

Digital Output Signal Conditioned Switching Circuits for Actuators align with the Automation Pyramid's layer of Sensors & Actuators.



Sehr et al., 2021

Lab: Build An ESP32 DC Motor Controller

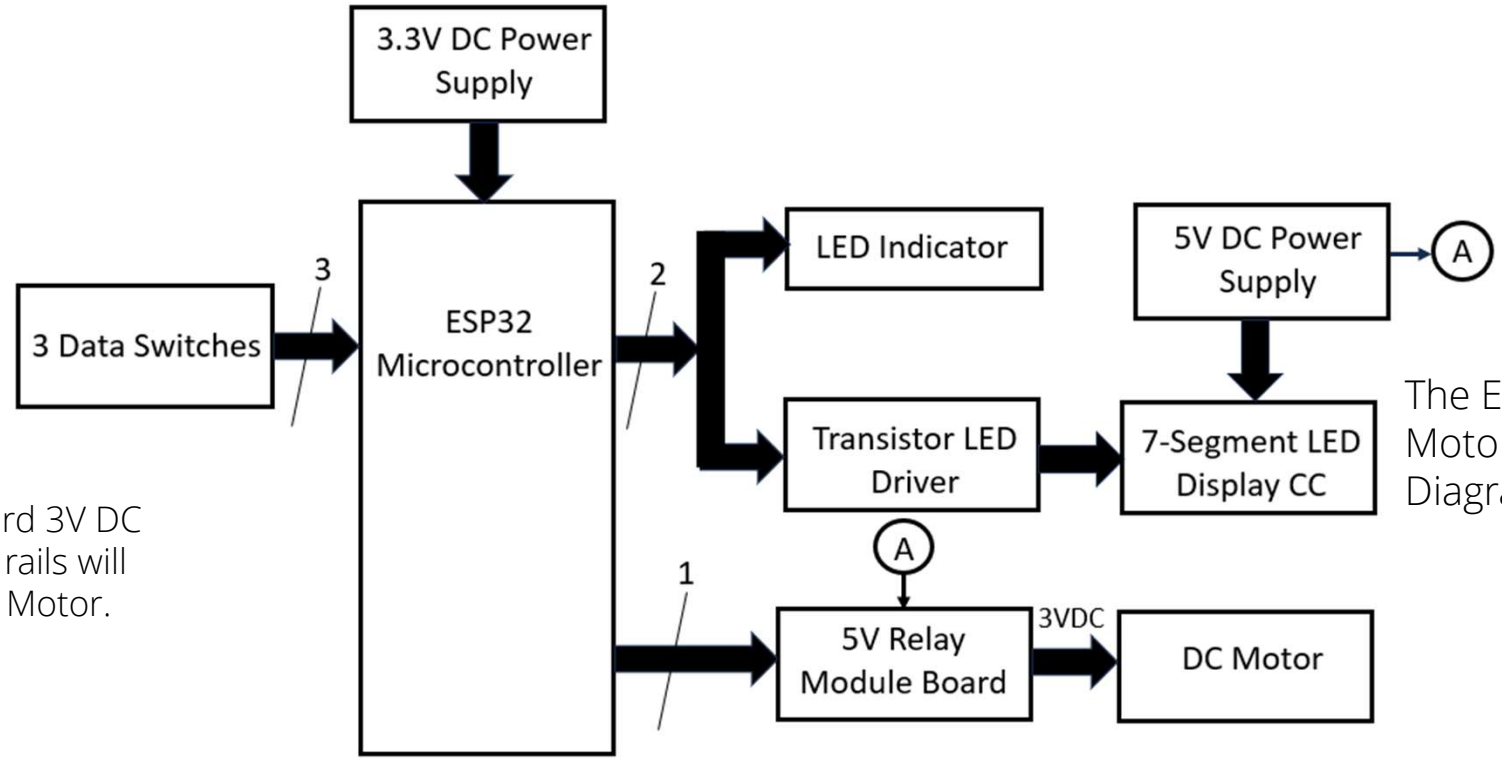


Lab: Build An ESP32 DC Motor Controller. . .

Participant Learning Objectives:

- Participants will learn to wire a DC Motor Controller using an ESP32 Micro Trainer, off-the-shelf electronic components, and a solderless breadboard.
- Participants will learn to install and set up the Mu programming platform.
- Participants will learn to program and test their ESP32 Micro Trainer using the MicroPython language.
- Participants will learn to operate a DC motor using the ESP32 Micro Trainer's tactile pushbutton switch.

Lab: Build An ESP32 DC Motor Controller. . .

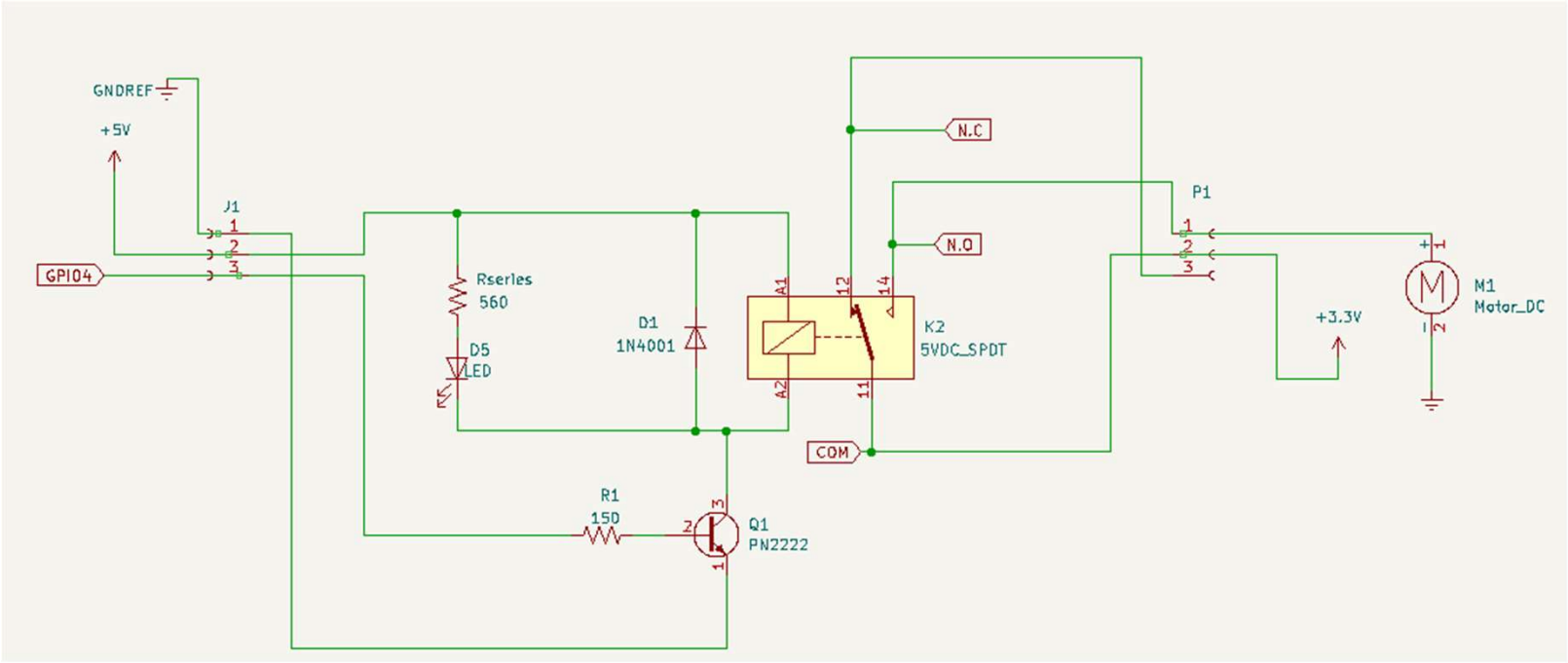


The ESP32 Micro DC Motor Controller Block Diagram

Note:
The Breadboard 3V DC Power Supply rails will power the DC Motor.

Lab: Build An ESP32 DC Motor Controller. . .

5V Relay Module with LED Indicator Board Electronic Circuit Schematic Diagram



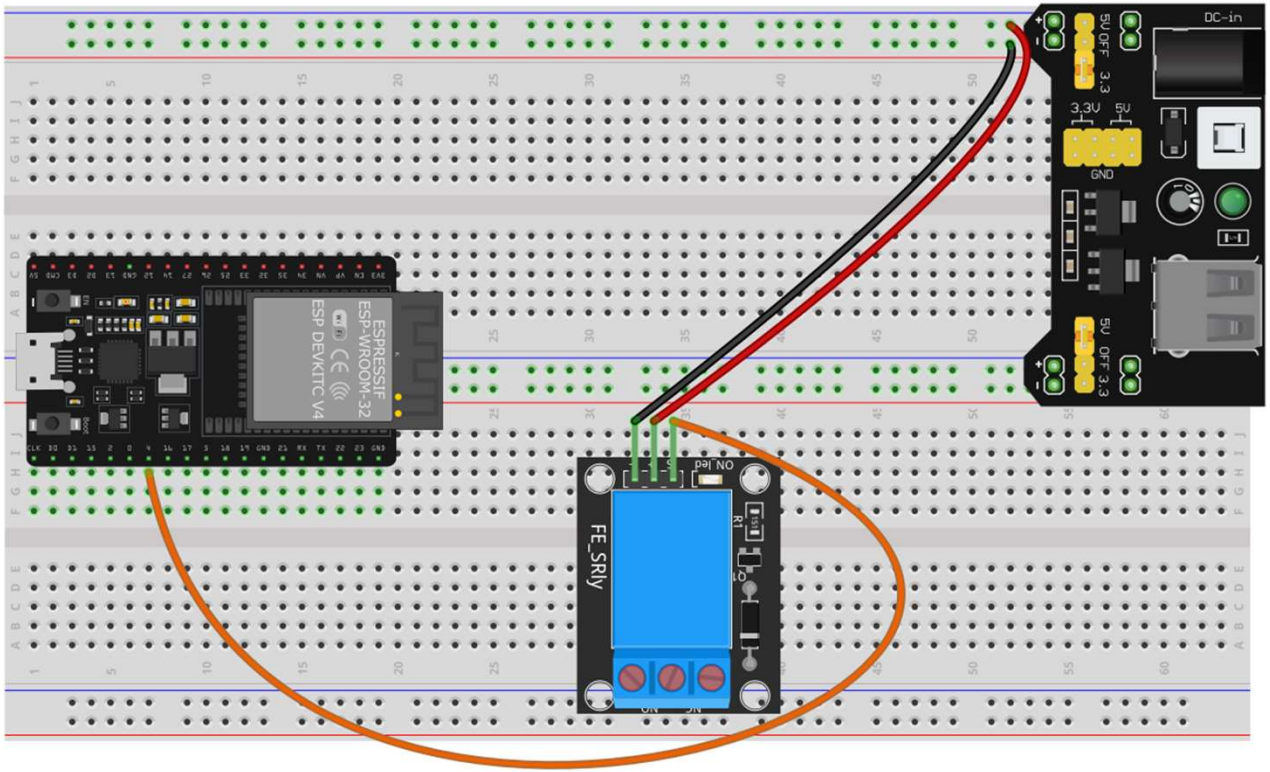
Question 4

In reviewing slide 34, what is the function of LED D5?

a) Power OFF Indicator
b) Power ON Indicator
c) Low voltage detection
d) none of the above

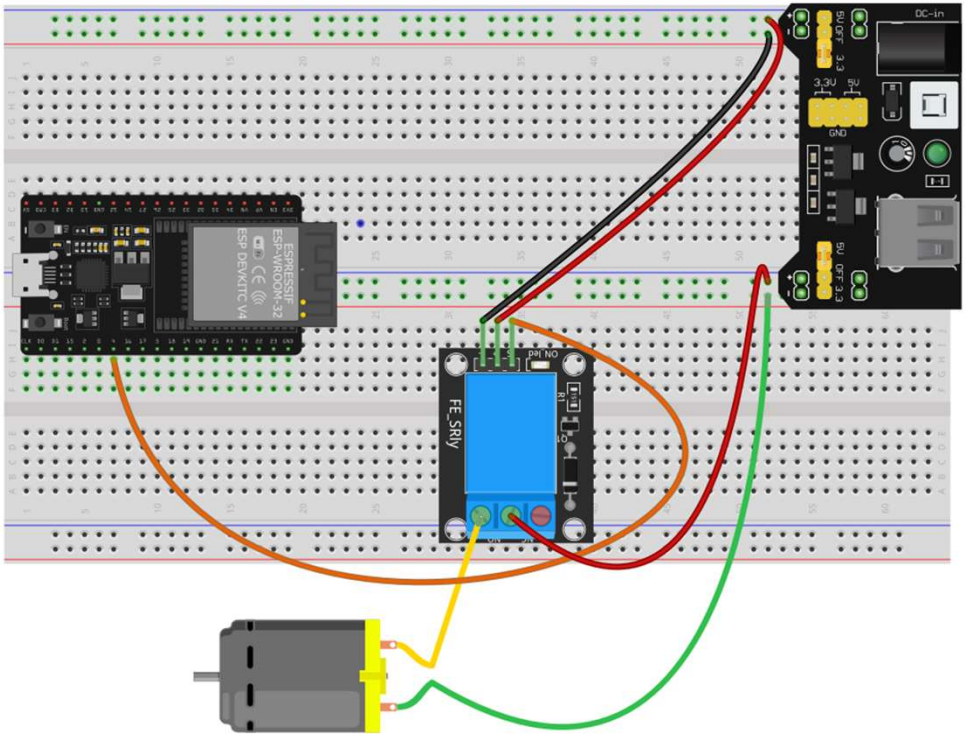
Lab: Build An ESP32 DC Motor Controller…

ESP32 To 5V Relay Module with LED Indicator Wiring Diagram



Lab: Build An ESP32 DC Motor Controller…

DC Motor to 5V Relay Module with LED Indicator Wiring Diagram

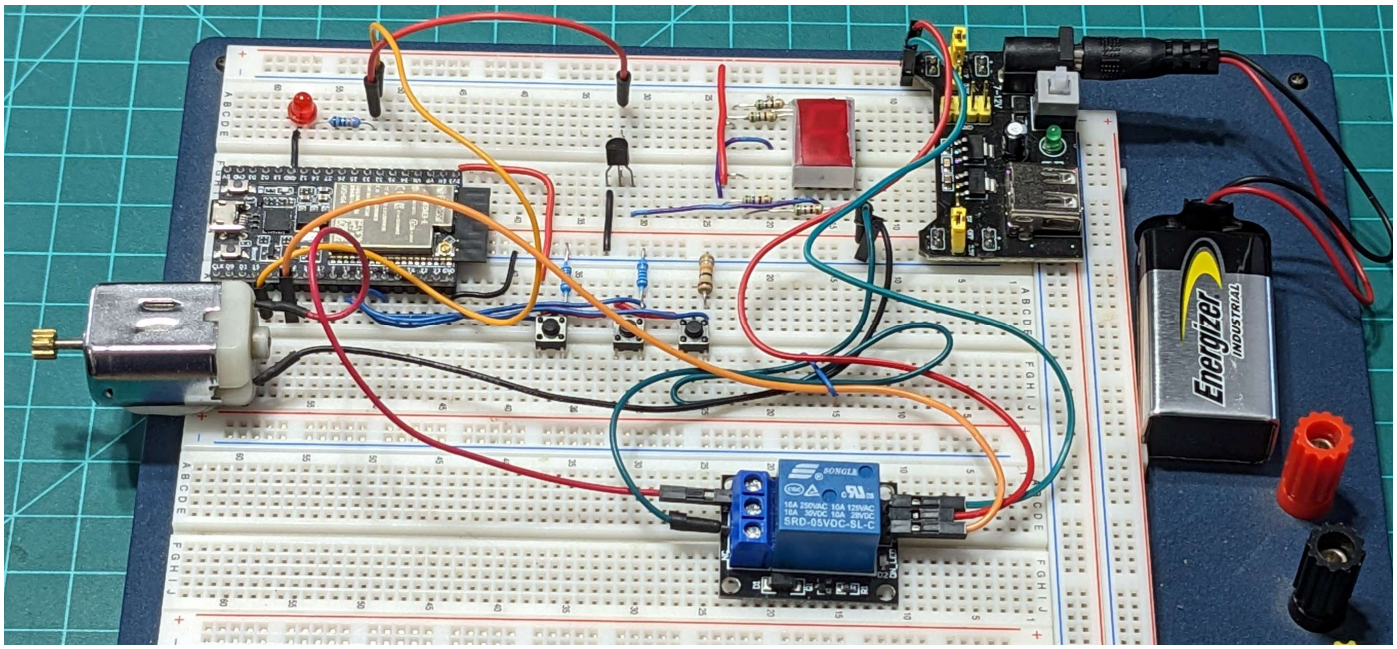


Lab: Build An ESP32 DC Motor Controller…

ESP32 DC Motor Controller: MicroPython Code

```
#ESP32_DC Motor_Controller

from machine import Pin
import time

# Define GPIO pins
button_pin = Pin(18, Pin.IN)  # Pushbutton with pull-up resistor
led_pin = Pin(2, Pin.OUT)  # LED pin
motor_pin = Pin(4, Pin.OUT)  # Motor pin

# Initialize variables
previous_state = button_pin.value()

# Main loop
try:
    while True:
        # Read the current state of the button
        button_state = button_pin.value()

        # Detect a press event
        if button_state == 1 and previous_state == 0:  # Button pressed
            led_pin.value(1)  # Turn on LED
            motor_pin.value(1) # Turn on Motor

        # Detect a release event
        if button_state == 0 and previous_state == 1:  # Button released
            led_pin.value(0)  # Turn off LED
            motor_pin.value(0) # Turn on Motor

        # Update the previous state
        previous_state = button_state

        time.sleep(0.05)  # Debounce delay
except KeyboardInterrupt:
    print("Program terminated.")

```

Lab: Build An ESP32 DC Motor Controller…



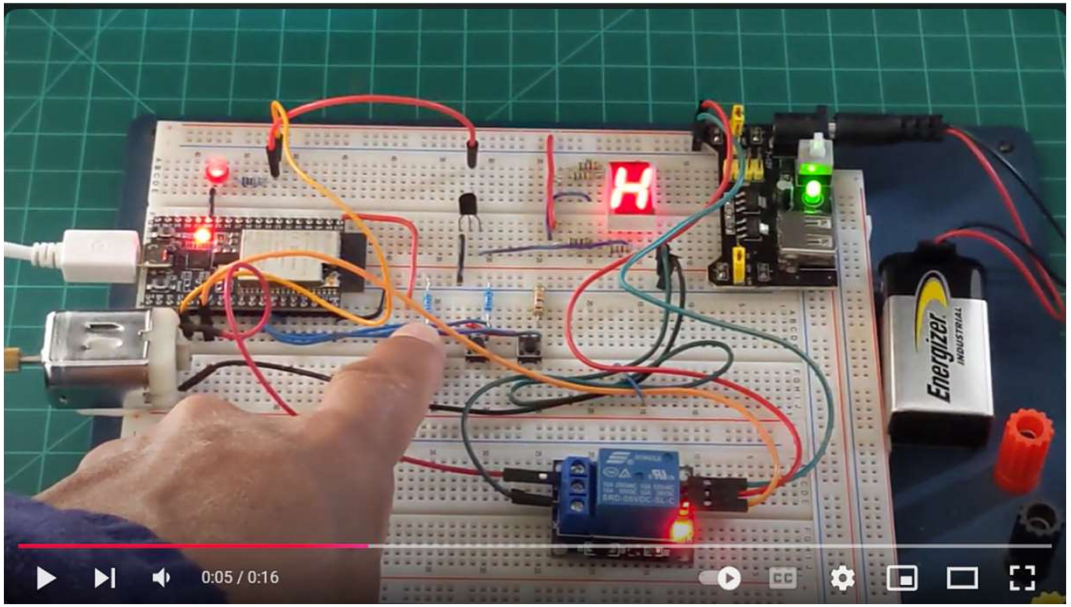
Completed Build ESP32 DC Motor Controller

Lab: Build An ESP32 DC Motor Controller. . .

Functional ESP32-
DC Motor Controller

Watch the Video Clip!

https://youtu.be/E2adQiSrKyg

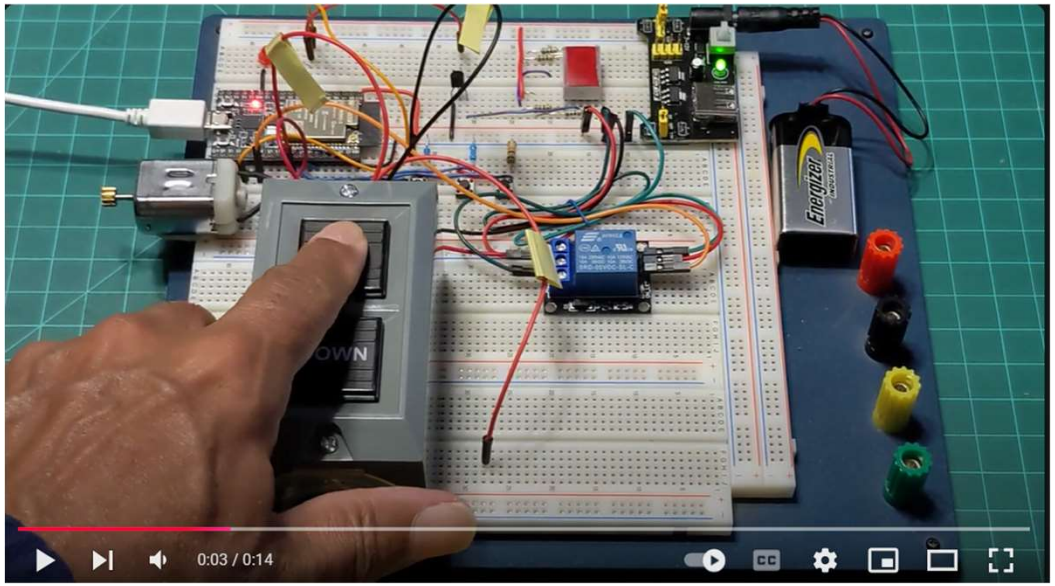


Lab: Build An ESP32 DC Motor Controller. . .

An Enhanced ESP32 DC Motor Controller With An Industrial Pushbutton Station: Bonus Video

Watch the Video Clip!

https://youtu.be/OyRUZyQX5_c



Question 5

What ESP32 GPIO pin is used to operate the 5V Relay with LED indicator?

a) 2
b) 18
c) 19
d) 4

Thank you for attending

Please consider the resources below:

Sehr, M.A, Lohstroh, M., Weber, M., Ugaide, I., Witte, M., Neidig, J., Hoeme, S., Niknami, M., & Lee, E.A. (2021). Programmable logic controllers in the context of industry 4.0. *IEEE Transactions On Industrial Informatics* 17(5), 3523 – 3535. https://ieeexplore.ieee.org/document/9134804

Rockis, G.J., & Mazur, G.A. (2014). *Electrical motor controls: For integrated systems* (5th ed.). American Technical Publishers.

Wilcher, D. (2024). *Understanding industrial controls with an esp32*. GitHub. https://github.com/DWilcher/DesignNews-WebinarCode/blob/main/December_24_Webinar_Code.zip

CEC Continuing Education Center

DesignNews

Thank You

Sponsored by

DigiKey

informa markets