CEC Continuing Education Center

DesignNews

Understanding Industrial Controls with an ESP32

Day 1: Digital Input 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.

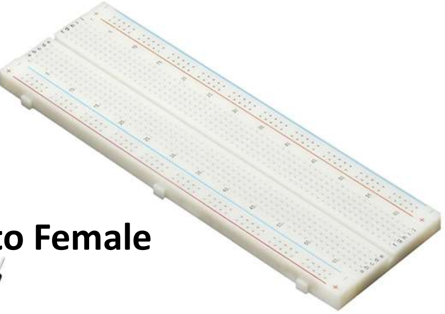
Course Kit and Materials

ESP32-DEVKITC-VIE

Adafruit Parts Pal Kit

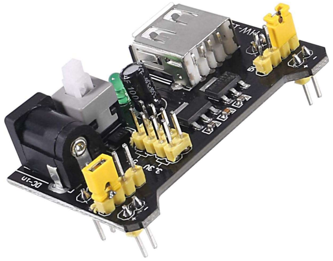
Jumper Wires: Male to Male

Solderless Breadboard x2



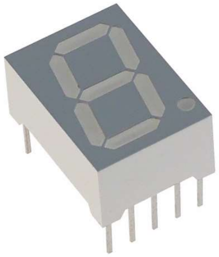
Jumper Wires: Male to Female

Solderless Breadboard Power Supply



5V Relay Module, 5V Indicator Light LED

7 Segment LED Display, Common Cathode



Standard Motor, 9100 RPM 6VDC

Agenda:

- Industrial Controls: Historical Perspective
- Automation Pyramid
- Industrial Switches
- Digital Input Signal Conditioning
- Lab: Build an ESP32 Micro Trainer

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).

Industrial Control: Historical Perspectives

- Industrial Controls automate and regulate industrial processes, ensuring three critical elements:
  a) efficiency,
  b) precision,
  c) and safety.
- Their evolution spans centuries, reflecting advances in technology and engineering principles.

Industrial Control: Historical Perspectives. . .

Pre-Industrial Era (Before 1750)

- Industrial Controls trace their roots to simple mechanisms like clocks, windmills, and early steam engines.
- Feedback Systems: The concept of feedback appeared as early as 270 BC with the invention of the water clock. The feedback mechanism was used to maintain a steady water flow.

Industrial Revolution (1750-1850)

- James Watt's steam engine invented in 1760 opened the field of mechanical feedback systems to regulate engine speed.
- This event marked a milestone in automatic controls.

Industrial Control: Historical Perspectives. . .

Early 20th Century: The Age Of Electrification

- The widespread adoption of electricity enabled the development of electromechanical systems.
- Relays, timers, and switches became standard for controlling industrial equipment.
- Engineers like Nyquist, Bode, and Black laid the foundation for control theory, focusing on system stability, and performance.
- Process Industries: The rise of chemical and petroleum industries necessitated the invention of basic controllers like thermostats, pressure regulators, and flow meters.

Question 1

What are the three critical elements of Industrial Controls?

a) efficiency, accuracy, and safety
b) energy, precision, and safety
c) efficiency, precision, and safety
d) none of the above

Industrial Control: Historical Perspectives. . .

Electromechanical Switching Devices and Basic Controllers

Electromechanical Relay

Flow meter

Timer Relay

Electrical Pushbutton Switches

Pressure Regulator

Thermostat

Industrial Control: Historical Perspectives. . .

Mid 20th Century: The Era of Automation

- The invention of the PLC was created by Dick Morley in 1968.
- The PLC replaced bulky relay-based systems, thereby providing flexibility and programmability for diverse applications.
- Analog controls transition to digital systems, improving precision and integration.
- Cybernetics and advanced control theories (e.g. PID controllers) allow widespread application in automation

Question 2

Who created the PLC?

- **a) Rufus P. Turner**
- **b) Richard Morley**
- **c) Hugo Gernsback**
- **d) none of the above**

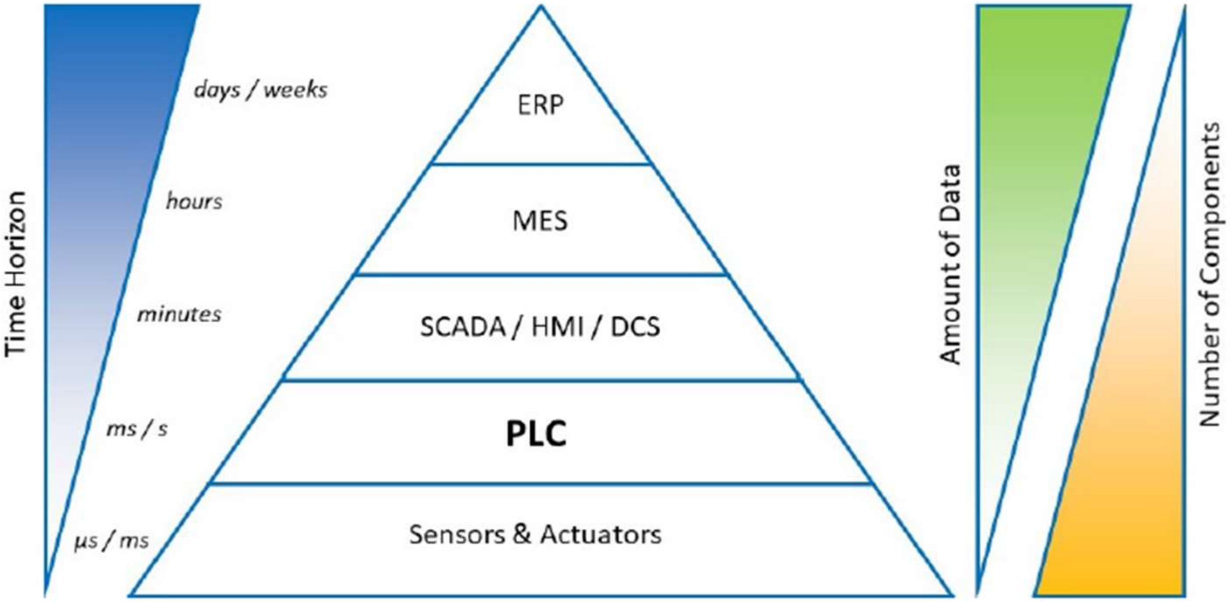
Automation Pyramid

Automation Pyramid

- A hierarchical model is used to represent the various levels of control and information flow within industrial automation systems.
- It provides a structured framework to understand how data moves from physical processes on the factory floor to higher-level business management systems.
- Each layer focuses on specific functions, technologies, and responsibilities.

Automation Pyramid. . .

Automation Pyramid



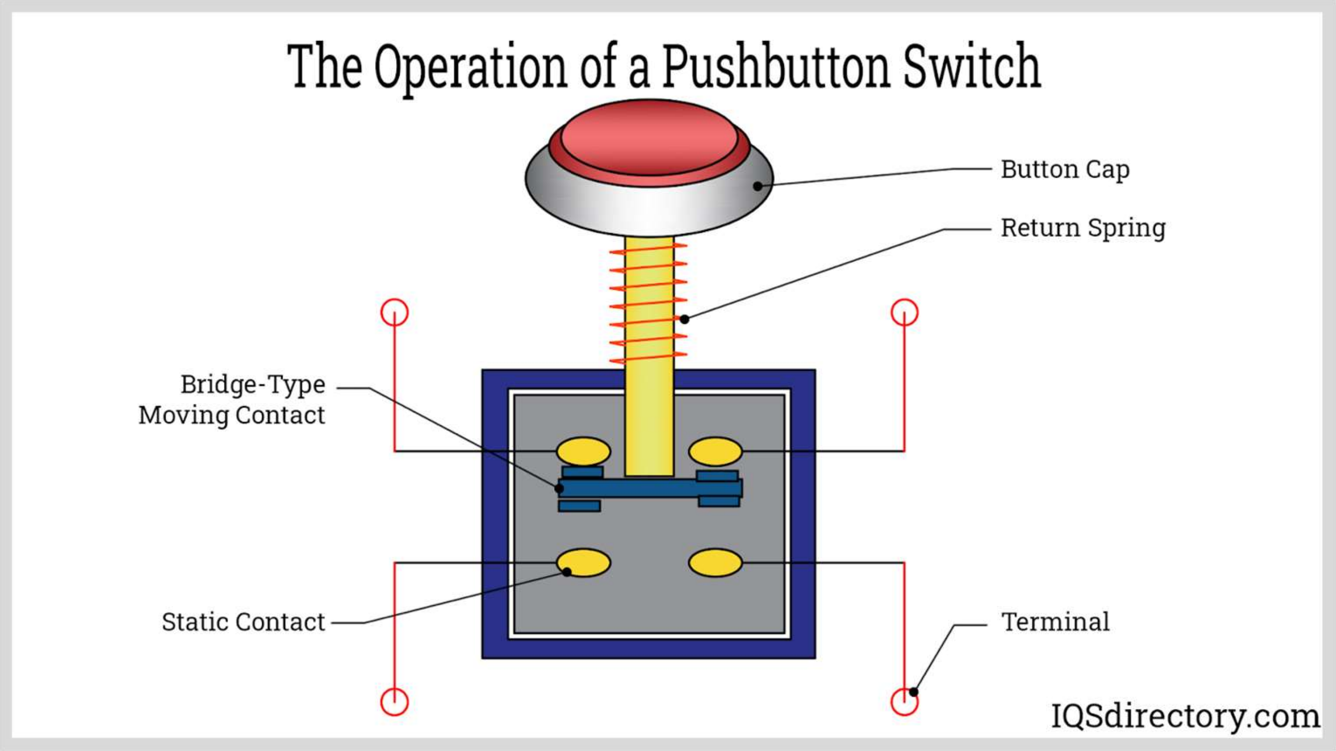
Sehr et al., 2021

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

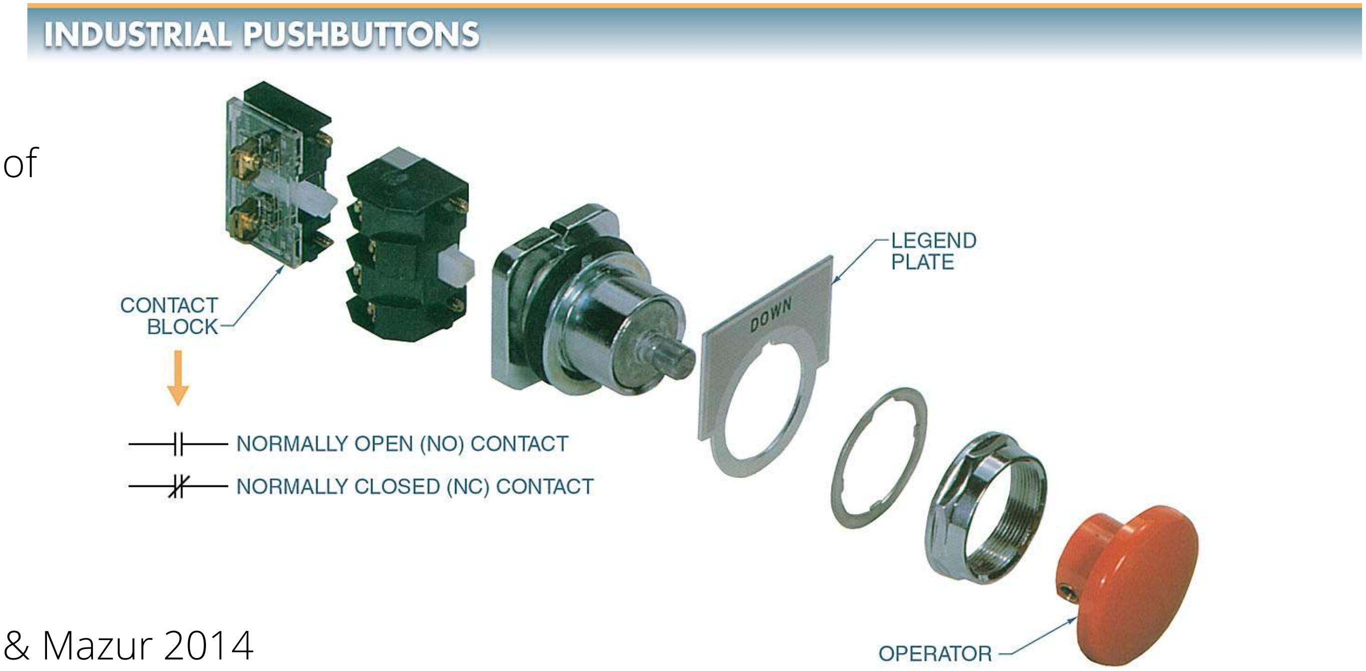
Industrial Switches

- An industrial switch is a robust switching device designed for controlling:
  a) electrical circuits
  b) machinery
  c) processes in industrial environments.
- Industrial switches are built to handle high currents and voltages.
- Industrial switches play an important role in the safe and reliable operation of industrial systems.

Industrial Switches. . .


The Operation of a Pushbutton Switch
Button Cap
Return Spring
Bridge-Type Moving Contact
Static Contact
Terminal
IQSdirectory.com

Industrial Switches. . .

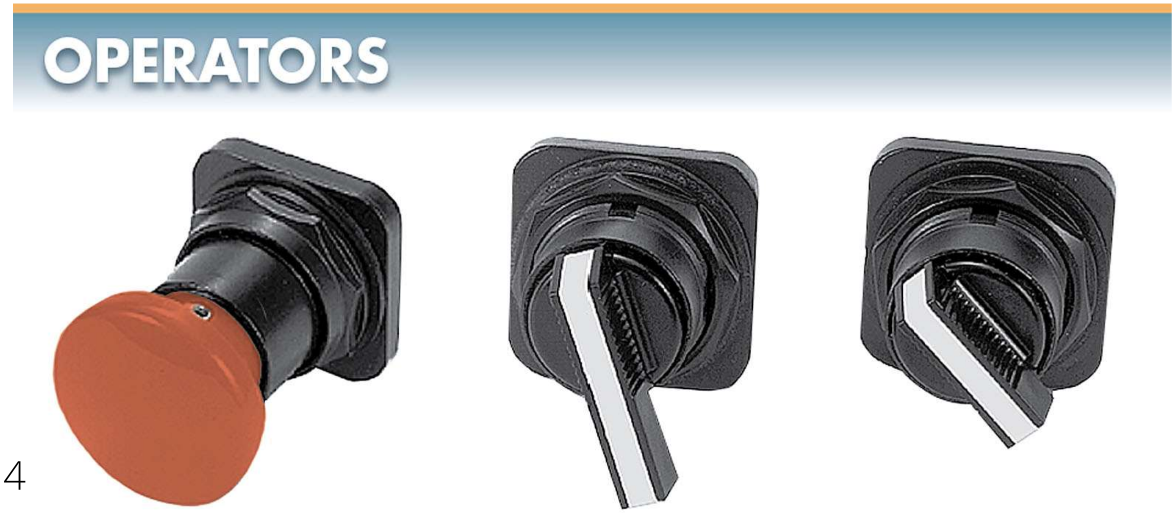


Typical construction of an industrial pushbutton switch

Rockis & Mazur 2014

Industrial Switches. . .

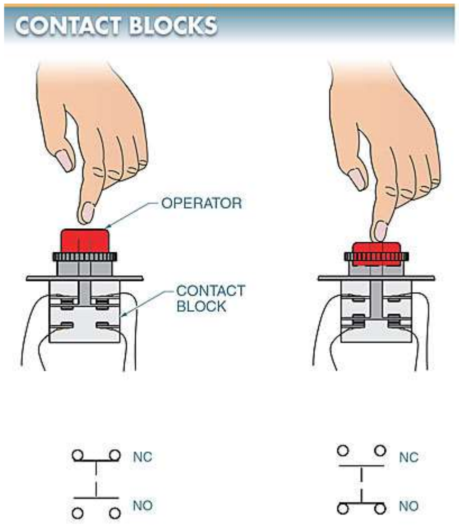
An operator is a device that is pressed, pulled, or rotated by the individual operating the circuit.

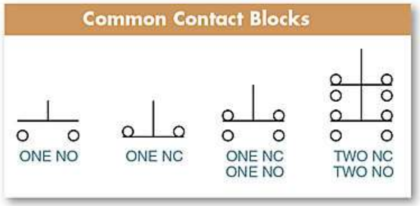


Rockis & Mazur 2014

Industrial Switches. . .

Contact blocks include normally open (NO), normally closed (NC), or both NO and NC contacts.





Rockis & Mazur 2014

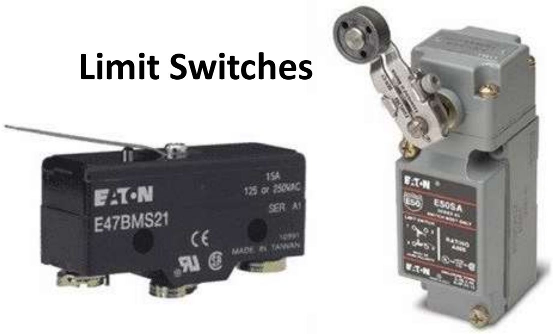
Question 3

What device's name is pressed, pulled, or rotated by the individual operating the circuit?

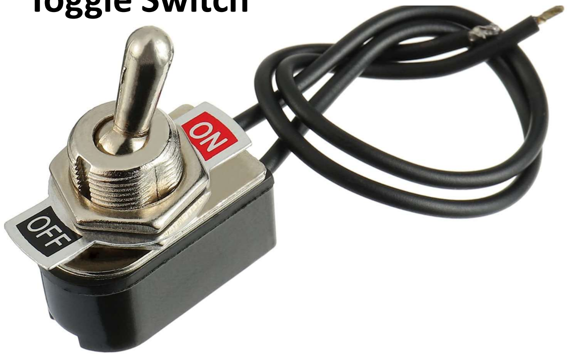
a) toggle
b) plunger
c) operator
d) none of the above

Industrial Switches. . .

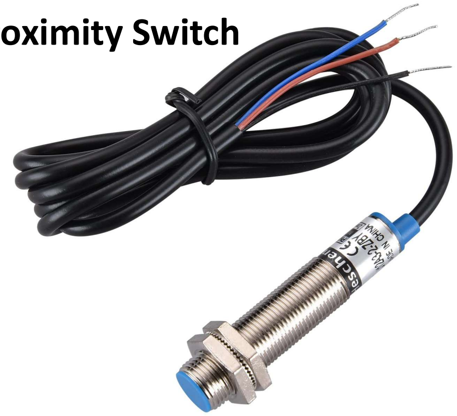
Limit Switches



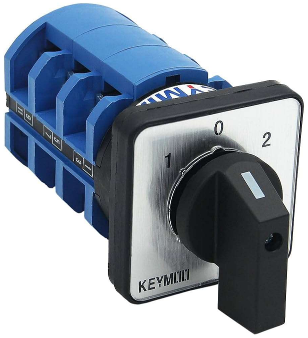
Toggle Switch



Proximity Switch



Rotary Switch



Digital Input Signal Conditioning

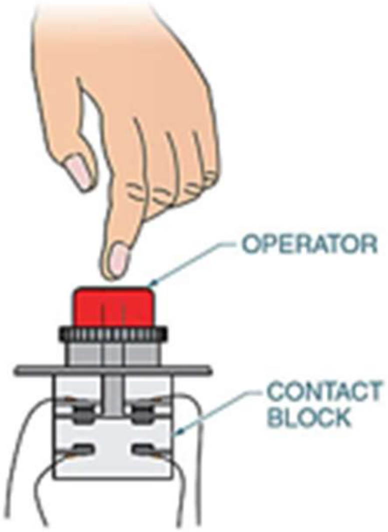
- An industrial control relies on gathering input information, evaluating it, and initiating action.
- In industrial control, input information often involves monitoring field devices whose outputs are one of two possible states:

  a) open

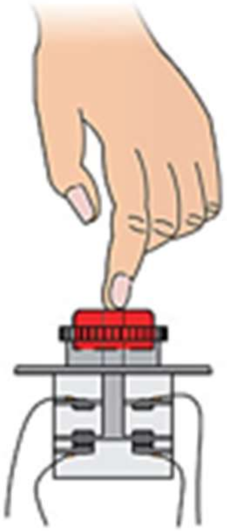
  b) closed.
- An electrical switch is the most common device that allows an open or closed state to occur.
- Another name used to describe this open and closed operation is called a *bi-state* device.

Digital Input Signal Conditioning . . .

An electrical pushbutton switch is an example of a Bi-State device.



Open Contact



Closed Contact

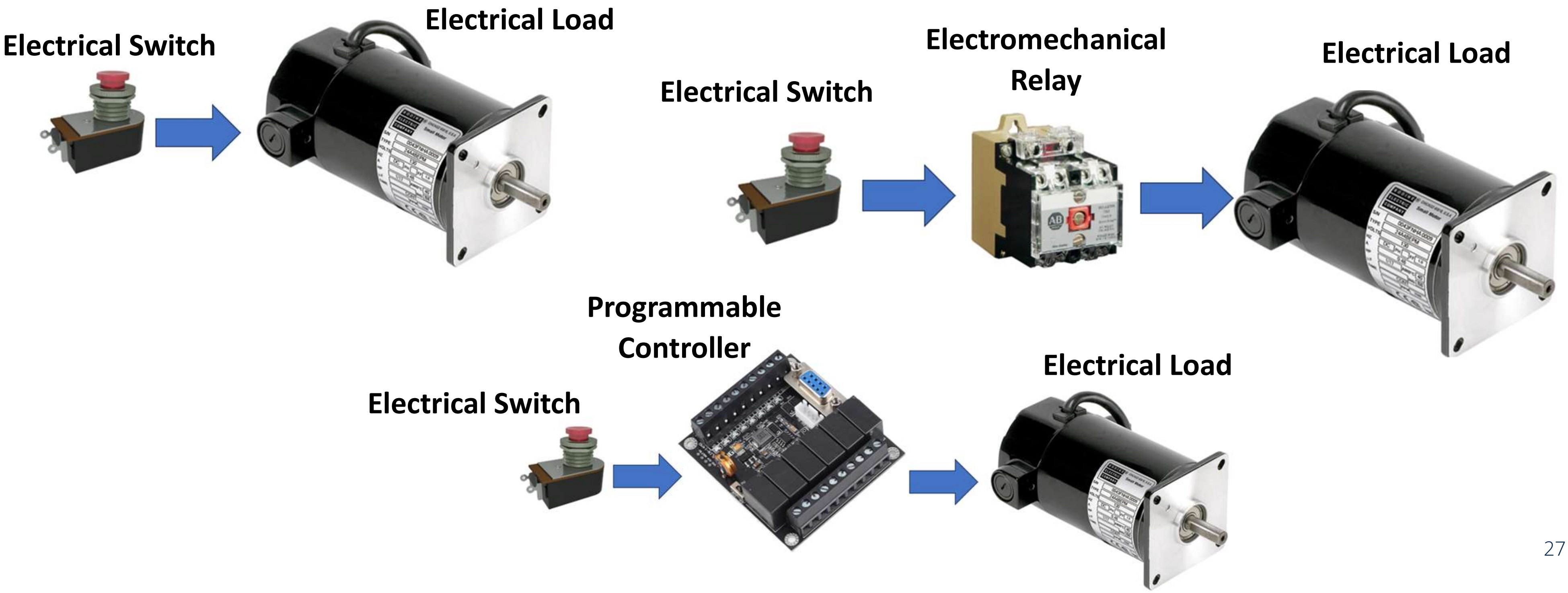
Digital Input Signal Conditioning . . .

Electrical Switch can provide control of an operation in three ways.

- An electrical switch wired directly to the load.
- An electrical switch can be wired into the input circuit of an electromechanical relay.
- The on/off status of an electrical switch may provide a digital input to a programmable controller.

Digital Input Signal Conditioning . . .

Electrical Switch can provide control of an operation in three ways.



Electrical Switch

Electrical Load

Electrical Switch

Electromechanical Relay

Electrical Load

Electrical Switch

Programmable Controller

Electrical Load

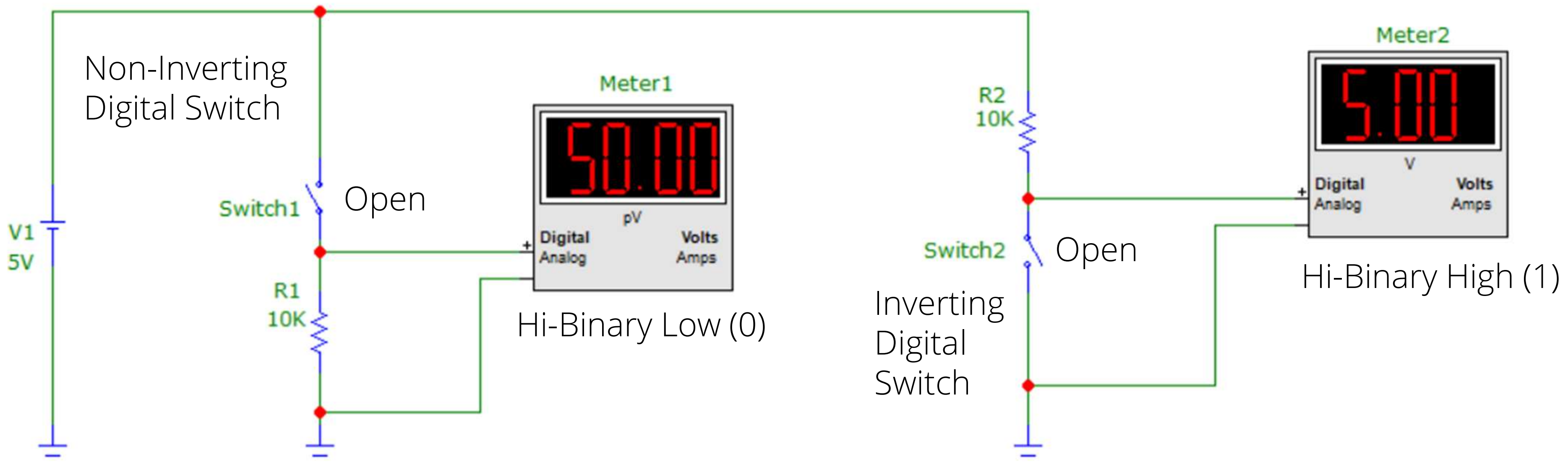
Digital Input Signal Conditioning . . .

The Digital Switch

- A digital switch is an electronic device or circuit that operates in two distinct states:
  - a) ON (logic high, binary 1).
  - b) OFF (logic low, binary 0).
- It is used to control the flow of current or signals in a digital circuit.

Digital Input Signal Conditioning

The operation of a Digital Switch



Non-Inverting Digital Switch

Open

Hi-Binary Low (0)

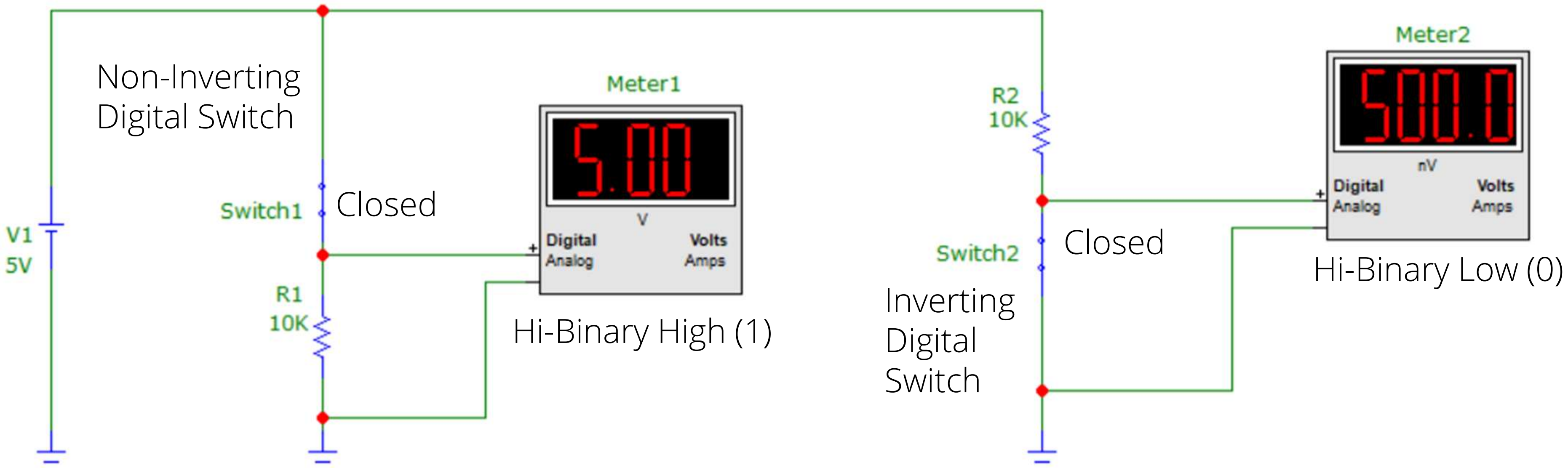
Inverting Digital Switch

Open

Hi-Binary High (1)

Digital Input Signal Conditioning

The operation of a Digital Switch



Non-Inverting Digital Switch

Closed

Hi-Binary High (1)

Closed

Inverting Digital Switch

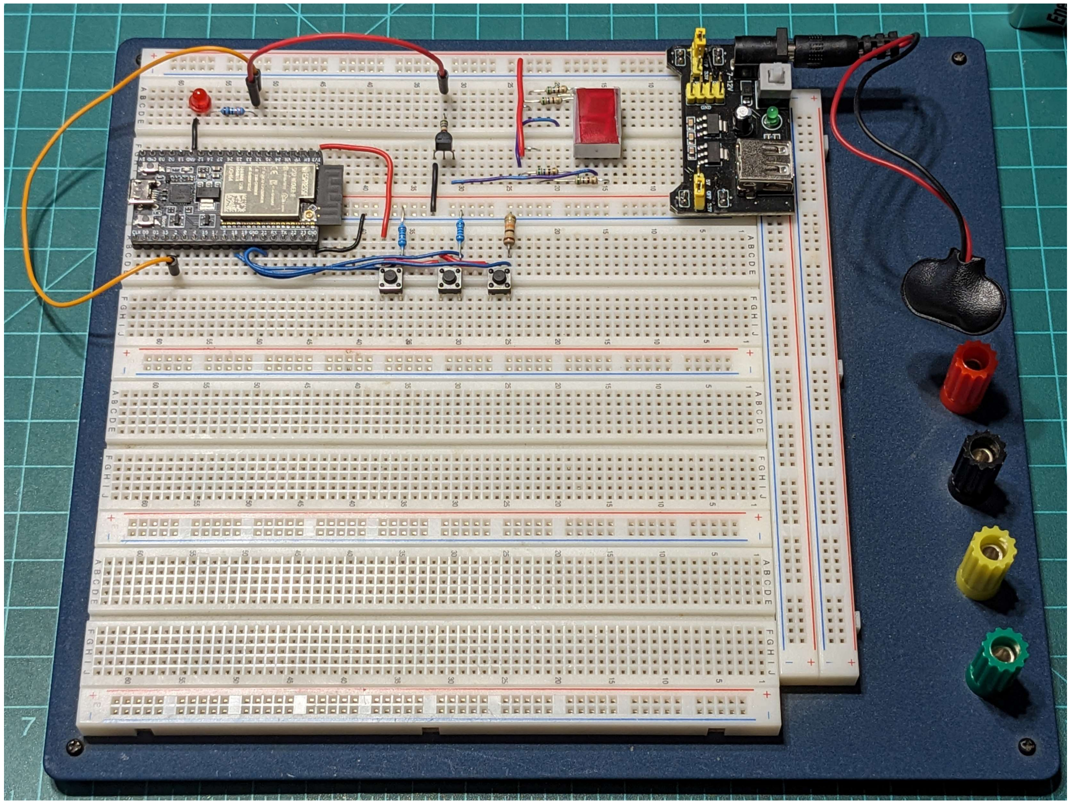
Hi-Binary Low (0)

Question 4

How many distinct states does a digital switch operate in?

a) 2
b) 4
c) 3
d) none of the above

Lab: Build an ESP32 Micro Trainer



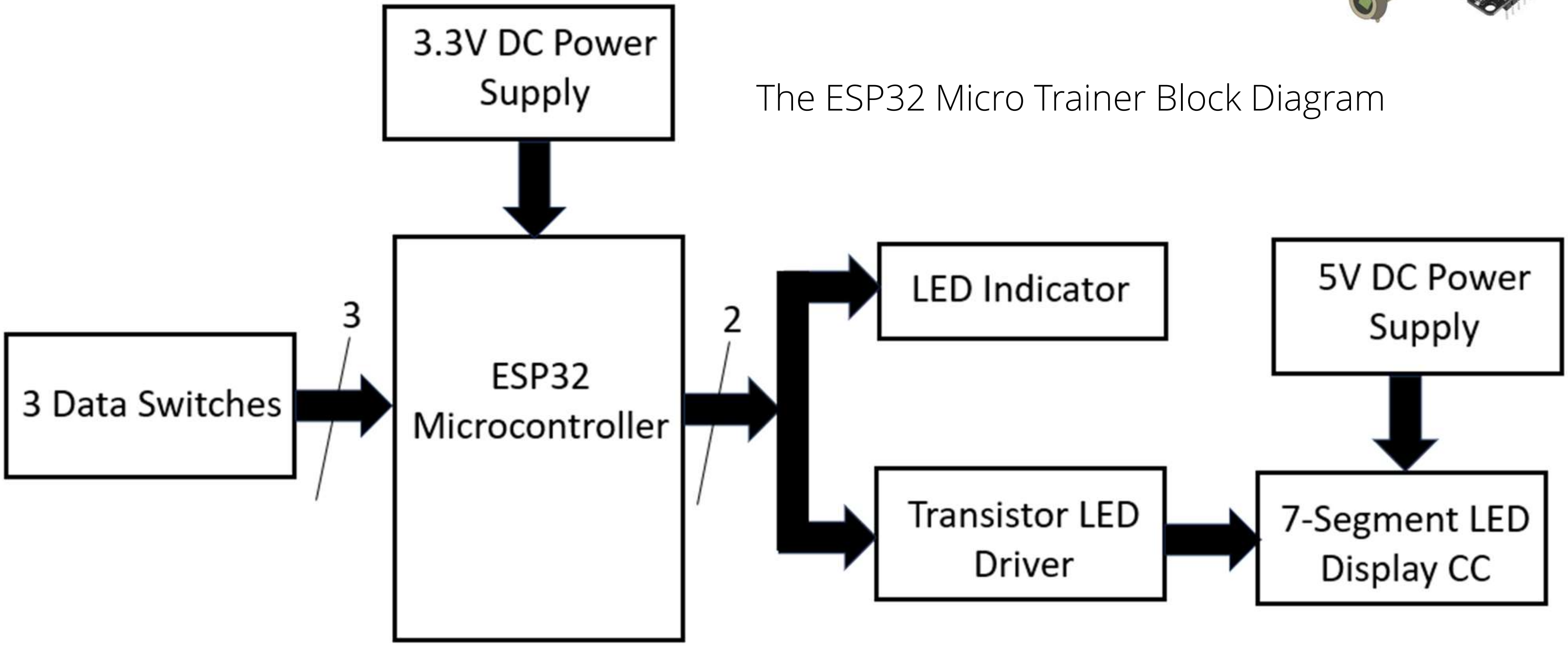
Lab: Build an ESP32 Micro Trainer. . .

Participant Learning Objectives:

- Participants will learn to wire the ESP32 Micro Trainer using off-the-shelf electronic components and a solderless breadboard.
- Participants will learn to install the MicroPython firmware onto an ESP32 Microcontroller.
- 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 display the tactile switch status using the ESP32 Mu plotter.

Lab: Build an ESP32 Micro Trainer. . .

The ESP32 Micro Trainer Block Diagram

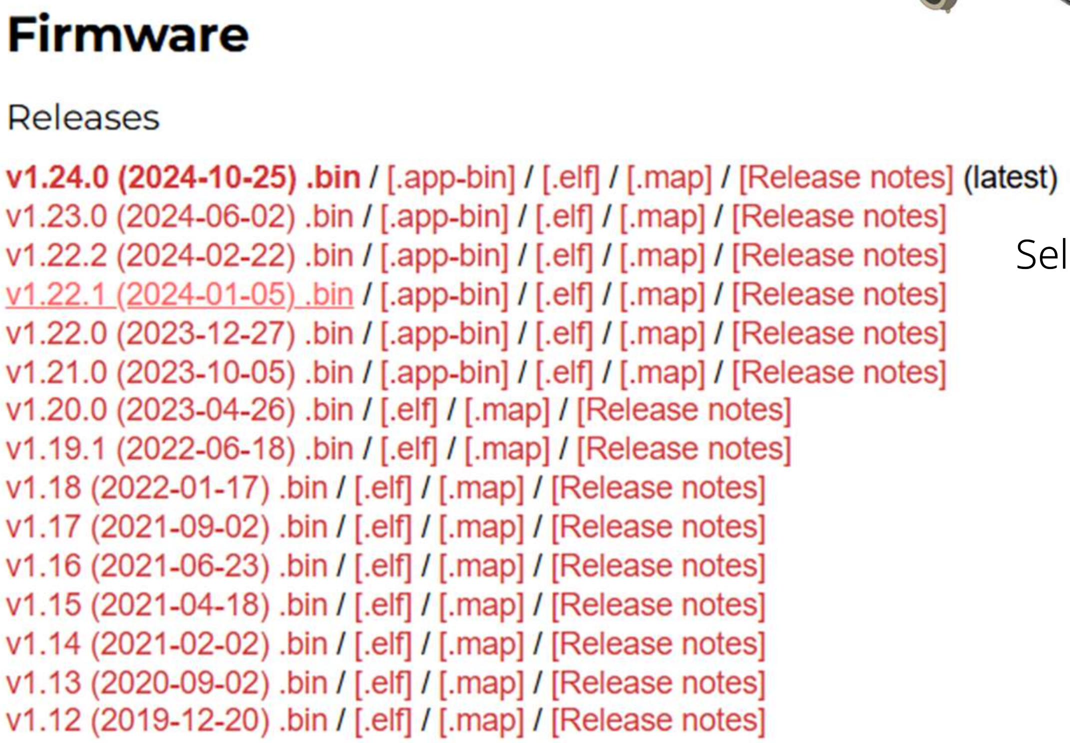


Lab: Build an ESP32 Micro Trainer. . .

Selecting the MicroPython Firmware for the ESP32 microcontroller.

Website Address for MicroPython Firmware:

https://micropython.org/download/
2_GENERIC/



Firmware

Releases

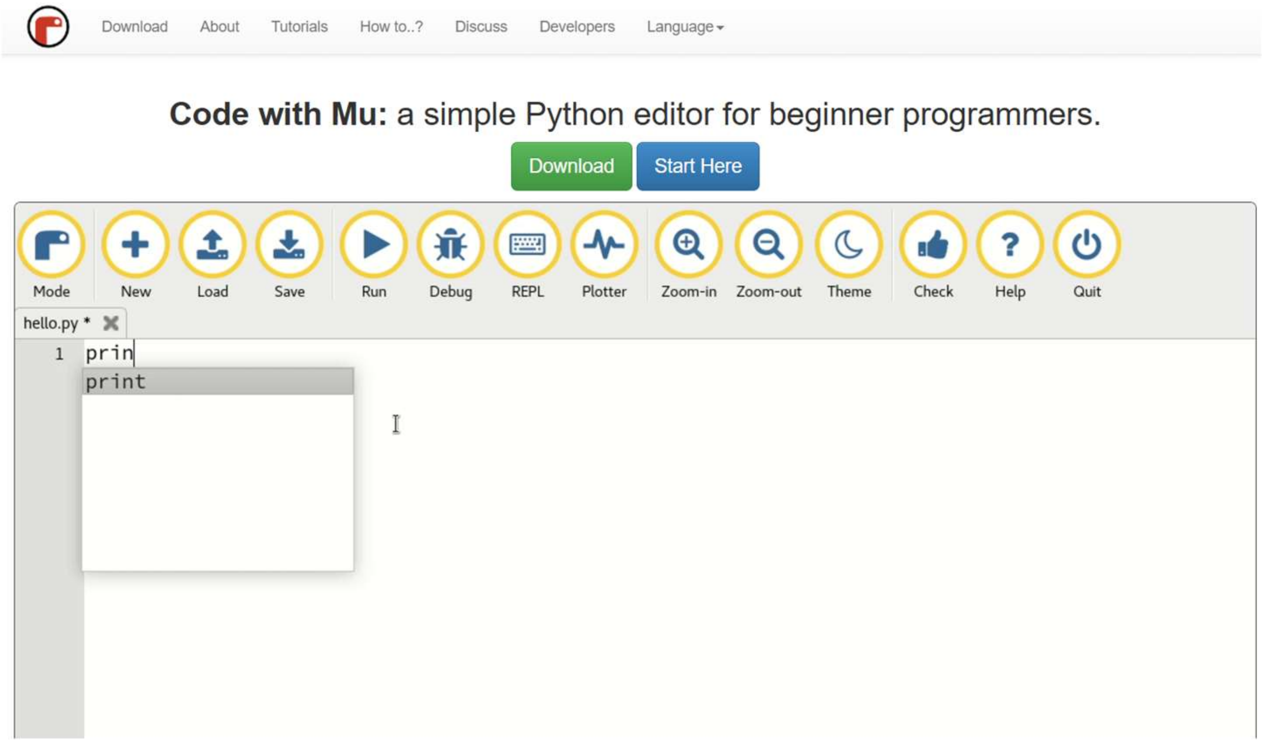
v1.24.0 (2024-10-25) .bin / [.app-bin] / [.elf] / [.map] / [Release notes] (latest)
v1.23.0 (2024-06-02) .bin / [.app-bin] / [.elf] / [.map] / [Release notes]
v1.22.2 (2024-02-22) .bin / [.app-bin] / [.elf] / [.map] / [Release notes]
v1.22.1 (2024-01-05) .bin / [.app-bin] / [.elf] / [.map] / [Release notes]
v1.22.0 (2023-12-27) .bin / [.app-bin] / [.elf] / [.map] / [Release notes]
v1.21.0 (2023-10-05) .bin / [.app-bin] / [.elf] / [.map] / [Release notes]
v1.20.0 (2023-04-26) .bin / [.elf] / [.map] / [Release notes]
v1.19.1 (2022-06-18) .bin / [.elf] / [.map] / [Release notes]
v1.18 (2022-01-17) .bin / [.elf] / [.map] / [Release notes]
v1.17 (2021-09-02) .bin / [.elf] / [.map] / [Release notes]
v1.16 (2021-06-23) .bin / [.elf] / [.map] / [Release notes]
v1.15 (2021-04-18) .bin / [.elf] / [.map] / [Release notes]
v1.14 (2021-02-02) .bin / [.elf] / [.map] / [Release notes]
v1.13 (2020-09-02) .bin / [.elf] / [.map] / [Release notes]
v1.12 (2019-12-20) .bin / [.elf] / [.map] / [Release notes]

Select this one!

Lab: Build an ESP32 Micro Trainer. . .

Mu Editor and Firmware Installation

Mu Home Page



Code with Mu: a simple Python editor for beginner programmers.

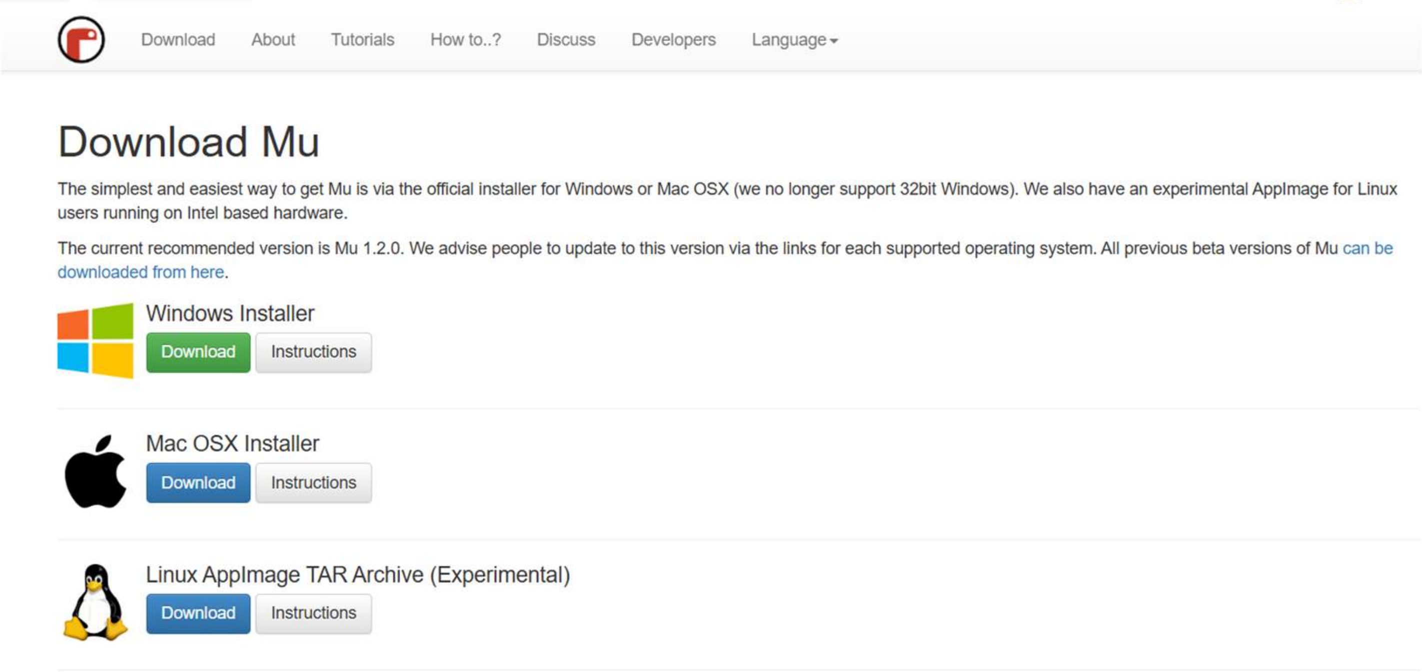
```
prin
```

https://codewith.mu/

Lab: Build an ESP32 Micro Trainer. . .

Mu Editor and Firmware Installation

Mu Download Page



Download Mu

The simplest and easiest way to get Mu is via the official installer for Windows or Mac OSX (we no longer support 32bit Windows). We also have an experimental AppImage for Linux users running on Intel based hardware.

The current recommended version is Mu 1.2.0. We advise people to update to this version via the links for each supported operating system. All previous beta versions of Mu can be downloaded from here.

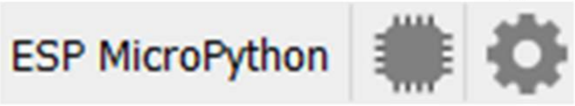
Windows Installer

Mac OSX Installer

Linux AppImage TAR Archive (Experimental)

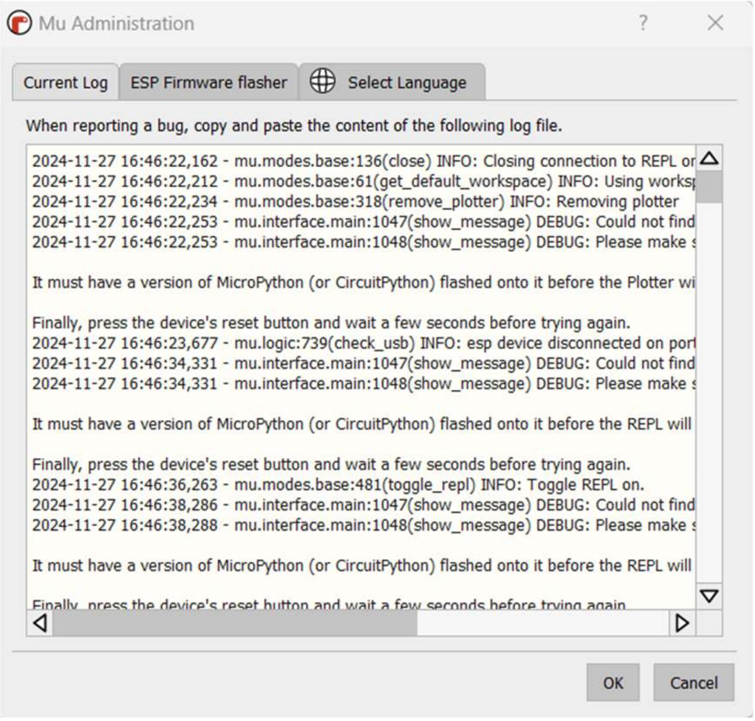
Lab: Build an ESP32 Micro Trainer. . .

Click this icon!



Located at the lower right-hand corner of the Mu Editor window

Firmware Installation



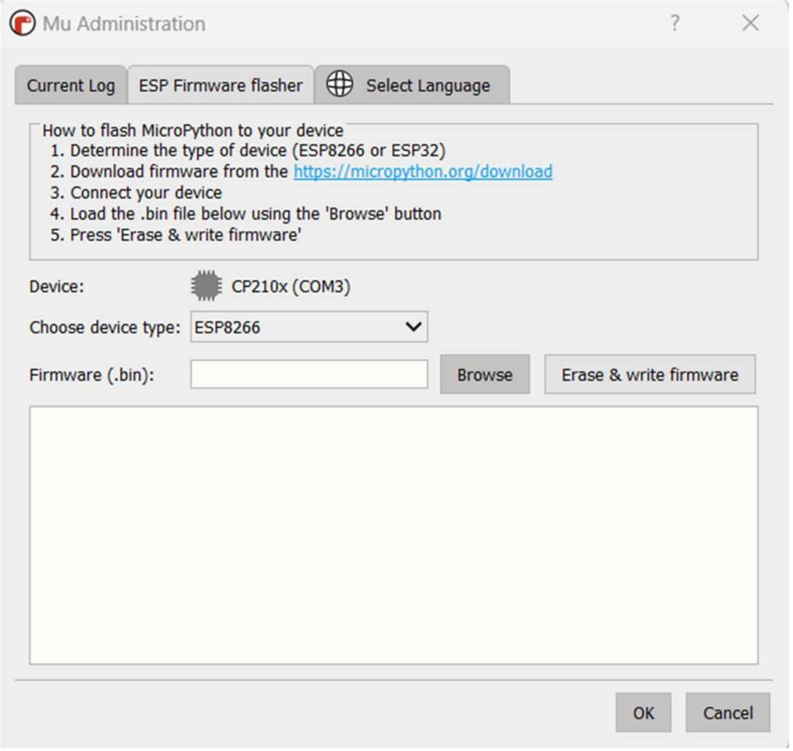
When reporting a bug, copy and paste the content of the following log file.

```
2024-11-27 16:46:22,162 - mu.modes.base:136(close) INFO: Closing connection to REPL or
2024-11-27 16:46:22,212 - mu.modes.base:61(get_default_workspace) INFO: Using worksp
2024-11-27 16:46:22,234 - mu.modes.base:318(remove_plotter) INFO: Removing plotter
2024-11-27 16:46:22,253 - mu.interface.main:1047(show_message) DEBUG: Could not find
2024-11-27 16:46:22,253 - mu.interface.main:1048(show_message) DEBUG: Please make s

It must have a version of MicroPython (or CircuitPython) flashed onto it before the Plotter wi

Finally, press the device's reset button and wait a few seconds before trying again.
2024-11-27 16:46:23,677 - mu.logic:739(check_usb) INFO: esp device disconnected on port
2024-11-27 16:46:34,331 - mu.interface.main:1047(show_message) DEBUG: Could not find
2024-11-27 16:46:34,331 - mu.interface.main:1048(show_message) DEBUG: Please make s

It must have a version of MicroPython (or CircuitPython) flashed onto it before the REPL will

Finally, press the device's reset button and wait a few seconds before trying again.
2024-11-27 16:46:36,263 - mu.modes.base:481(toggle_repl) INFO: Toggle REPL on.
2024-11-27 16:46:38,286 - mu.interface.main:1047(show_message) DEBUG: Could not find
2024-11-27 16:46:38,288 - mu.interface.main:1048(show_message) DEBUG: Please make s

It must have a version of MicroPython (or CircuitPython) flashed onto it before the REPL will

Finally, press the device's reset button and wait a few seconds before trying again.
```

Select the ESP Firmware Flasher tab

Lab: Build an ESP32 Micro Trainer. . .

Firmware Installation



How to flash MicroPython to your device

1. Determine the type of device (ESP8266 or ESP32)
2. Download firmware from the https://micropython.org/download
3. Connect your device
4. Load the .bin file below using the 'Browse' button
5. Press 'Erase & write firmware'

Device: CP210x (COM3)

Choose device type: ESP8266

Firmware (.bin):

Click Browse button to search for the MicroPython firmware bin file.

Click the Erase & write firmware button.

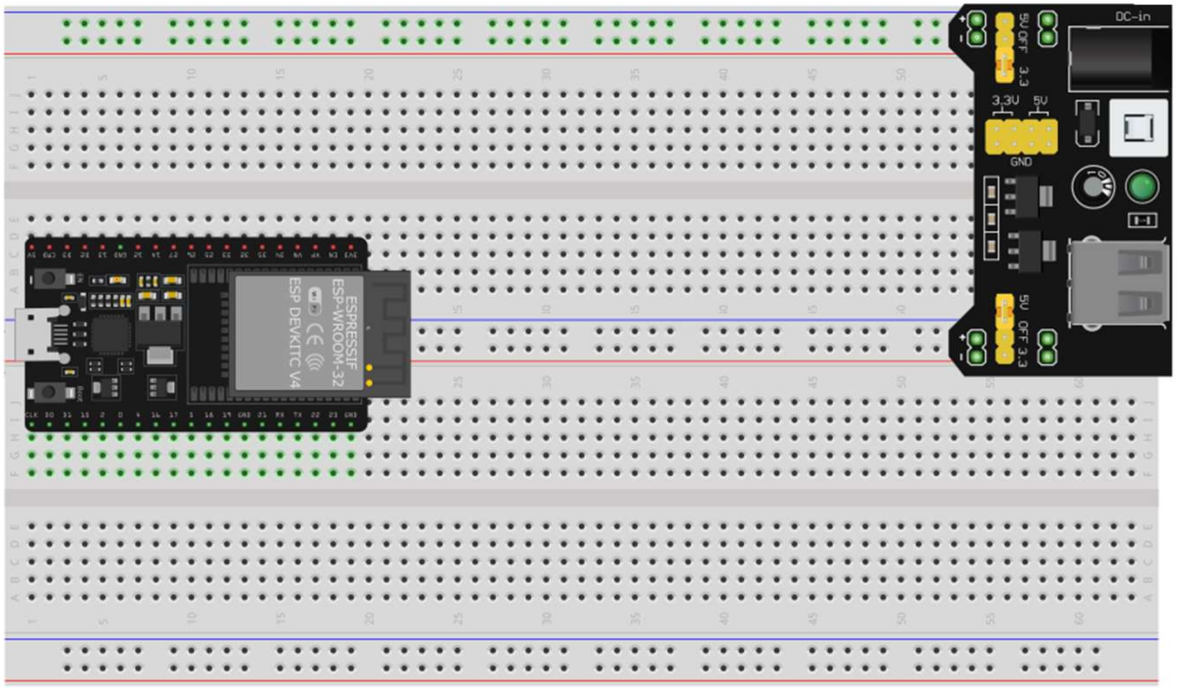
After firmware installation is completed, Click Ok.

Lab: Build an ESP32 Micro Trainer. . .

The ESP32- Solderless breadboard power supply layout diagram:

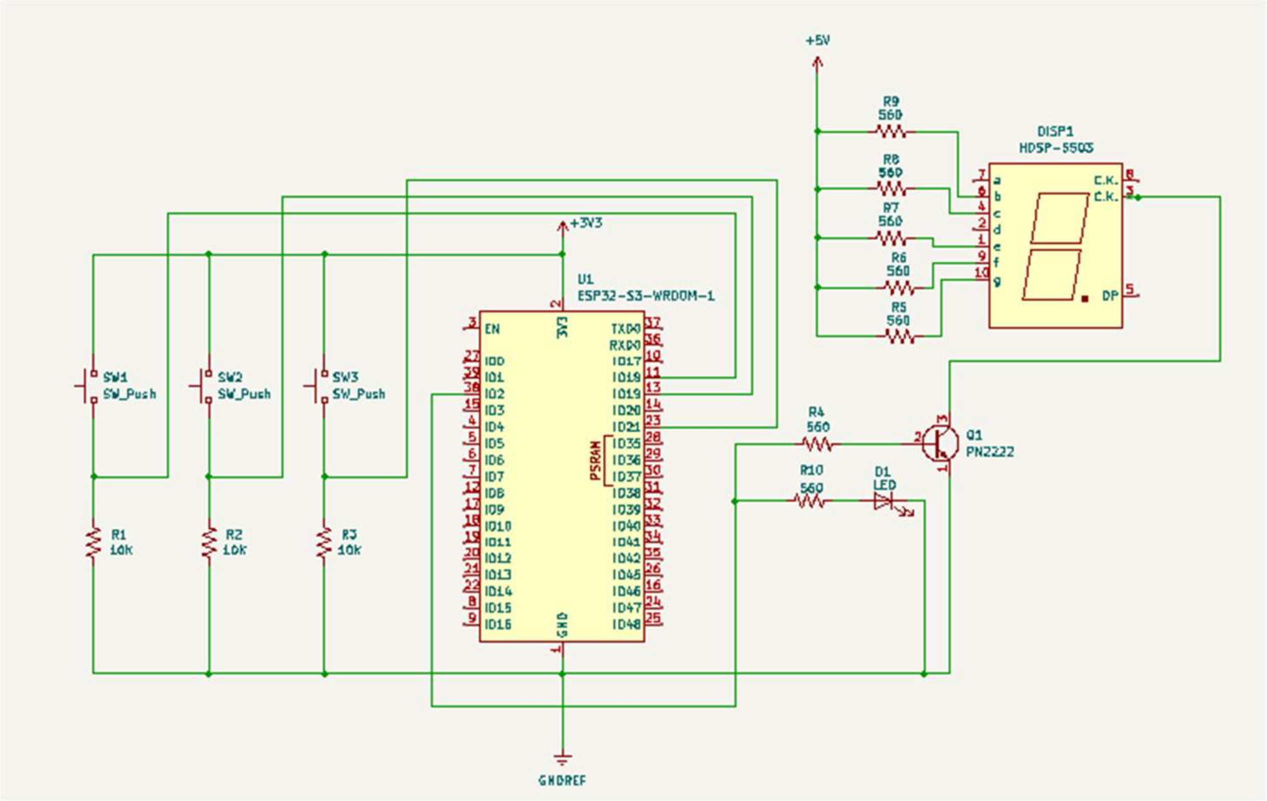
Notes:

- Two solderless breadboards are required for proper mounting of the ESP32 devkit board.
- The red and blue rails on the real solderless breadboards will properly align with the 5V and 3.3V polarity pins.



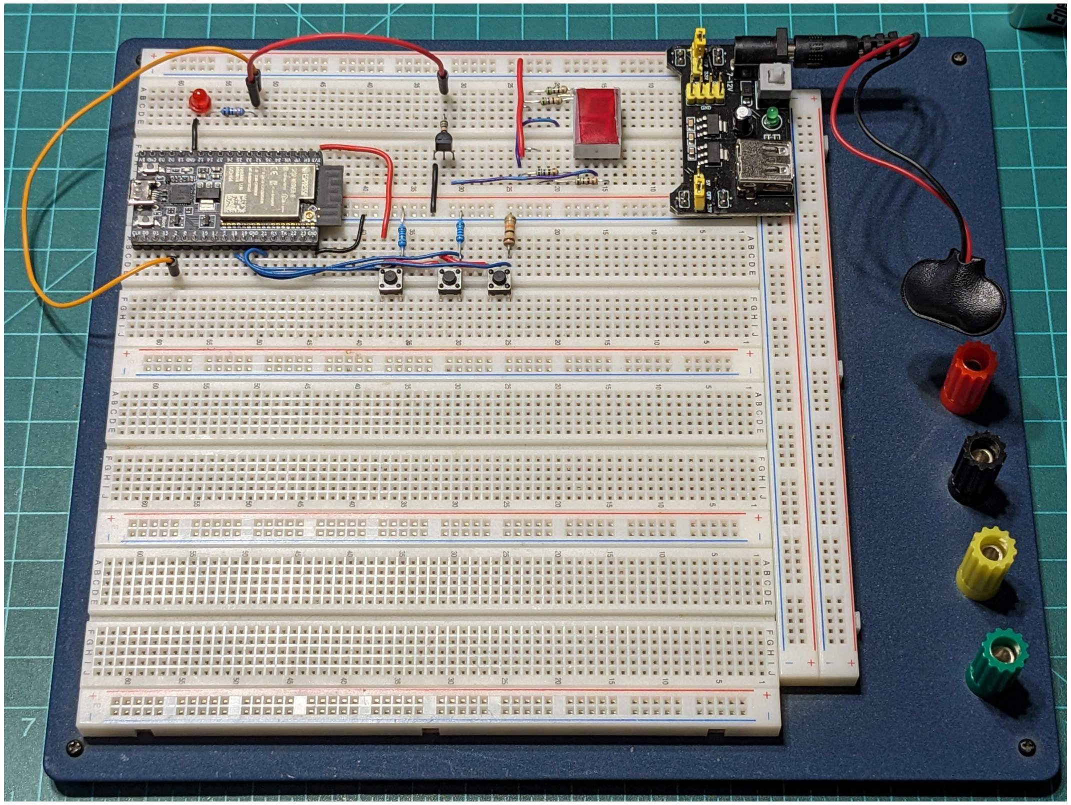
Lab: Build an ESP32 Micro Trainer. . .

The ESP32 Micro Trainer Electronic Circuit Schematic Diagram



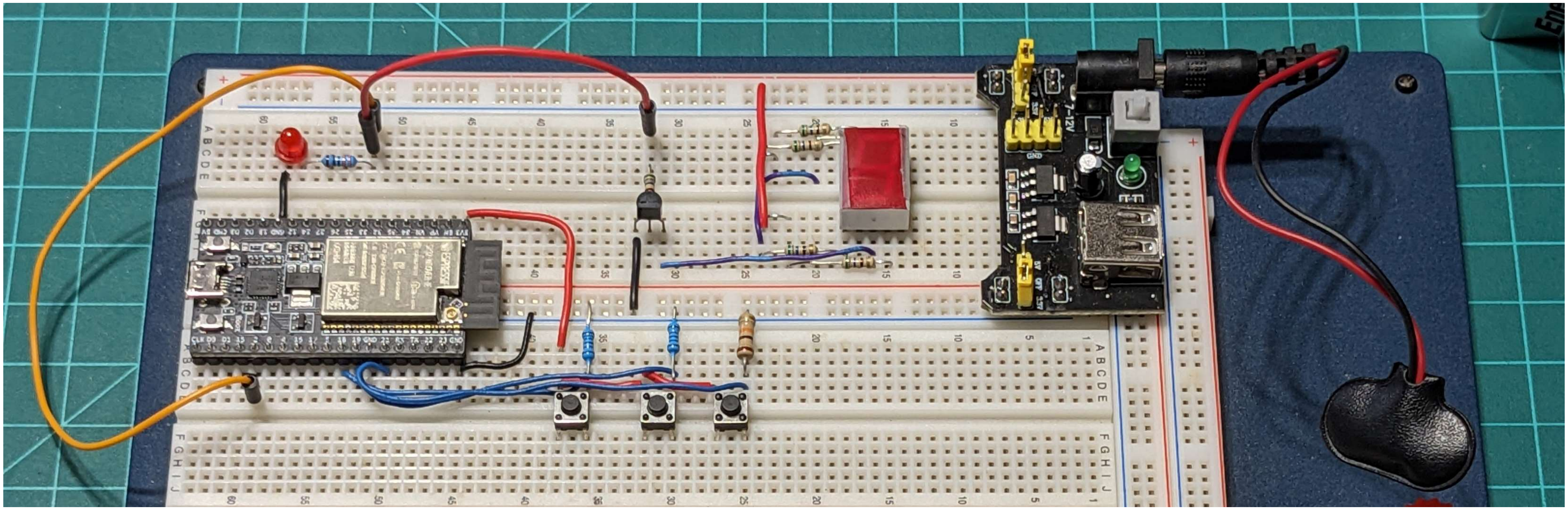
Lab: Build an ESP32 Micro Trainer. . .

Completed The ESP32 Micro Trainer



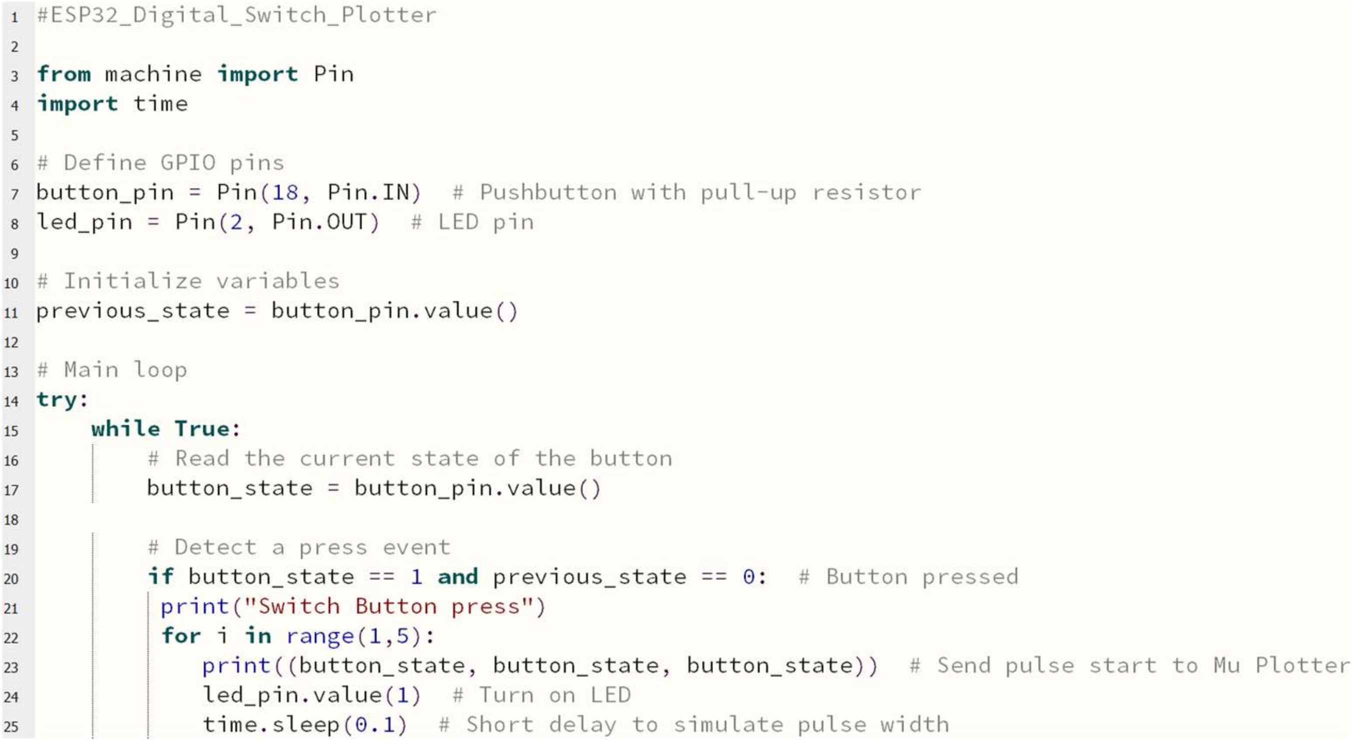
Lab: Build an ESP32 Micro Trainer. . .

Completed The ESP32
Micro Trainer: Zoomed-In



Lab: Build an ESP32 Micro Trainer. . .

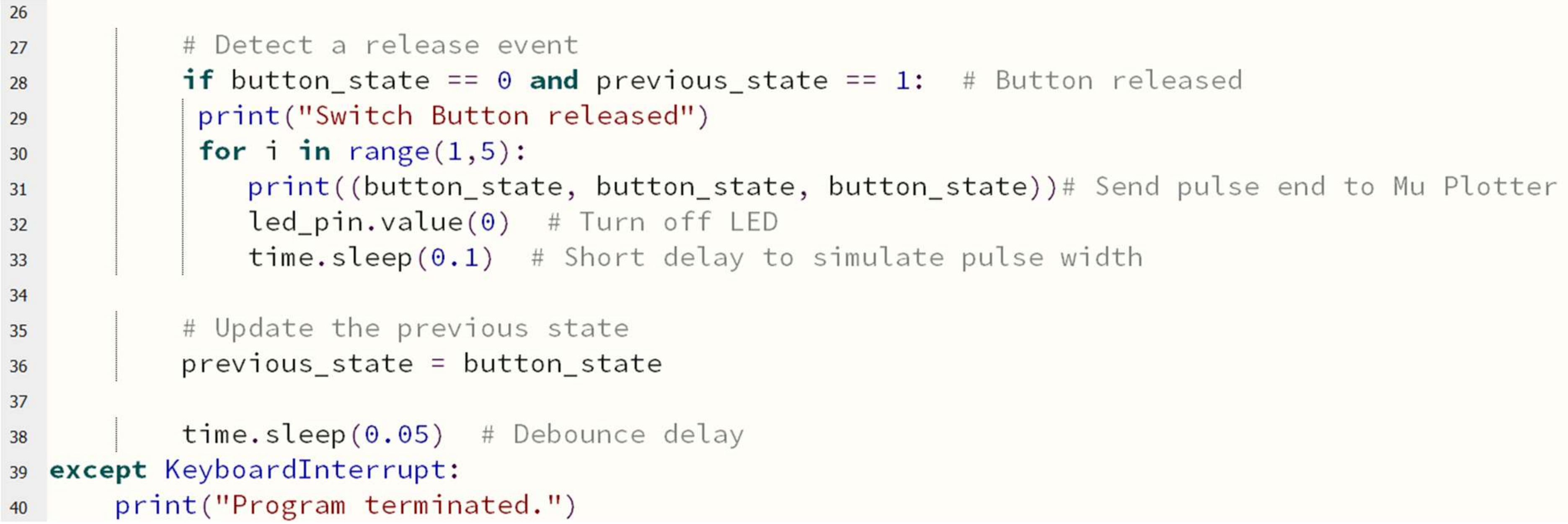
MicroPython Code: Part 1



```
#ESP32_Digital_Switch_Plotter

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

# 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
            print("Switch Button press")
            for i in range(1,5):
                print((button_state, button_state, button_state))  # Send pulse start to Mu Plotter
                led_pin.value(1)  # Turn on LED
                time.sleep(0.1)  # Short delay to simulate pulse width
```

Lab: Build an ESP32 Micro Trainer. . .

MicroPython Code: Part 2



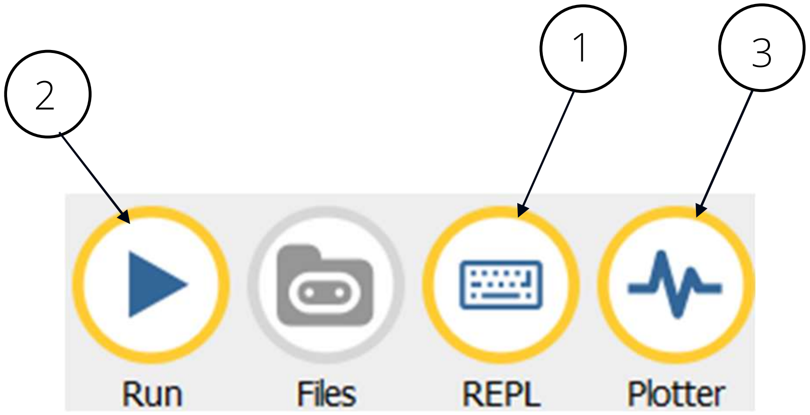
```
        # Detect a release event
        if button_state == 0 and previous_state == 1:   # Button released
         print("Switch Button released")
         for i in range(1,5):
            print((button_state, button_state, button_state))# Send pulse end to Mu Plotter
            led_pin.value(0)   # Turn off LED
            time.sleep(0.1)   # Short delay to simulate pulse width

        # Update the previous state
        previous_state = button_state

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

Lab: Build an ESP32 Micro Trainer. . .

Run The Code Using the Following Steps

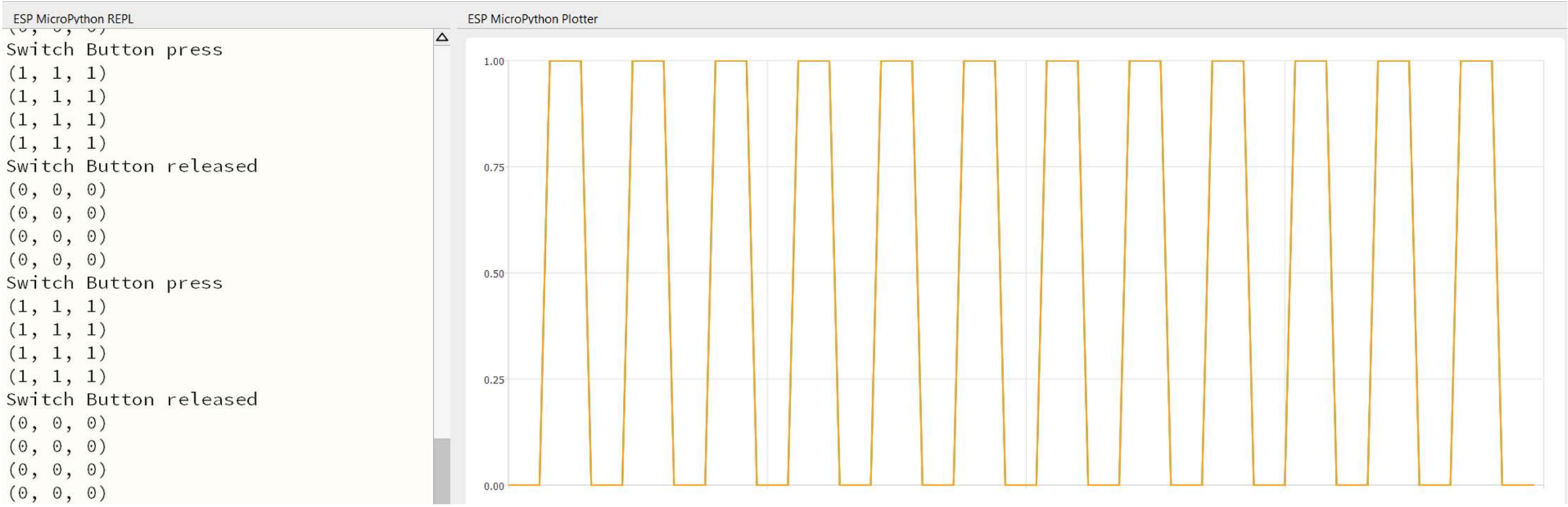
Steps:

1. Click the REPL icon with the mouse.
2. Click the RUN icon with the mouse.
3. Click the Plotter icon with the mouse.



Lab: Build an ESP32 Micro Trainer. . .

Switch Button Actuation Output Results
The LED will blink ON/OFF with each press of the Pushbutton Switch.



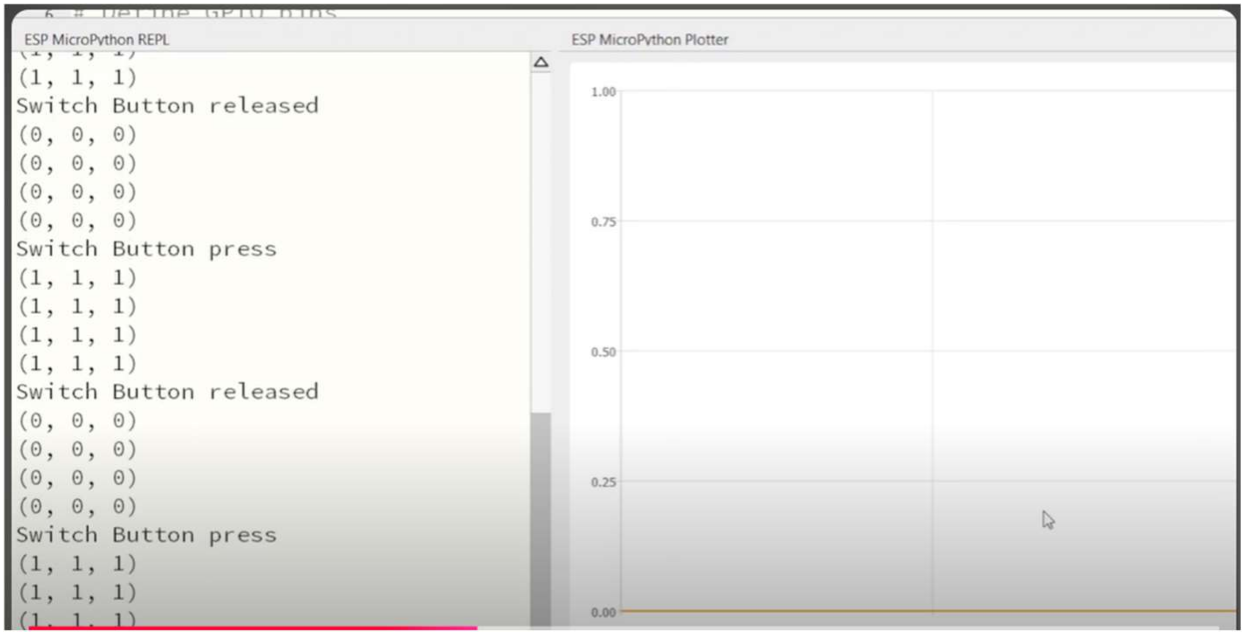
```
Switch Button press
(1, 1, 1)
(1, 1, 1)
(1, 1, 1)
(1, 1, 1)
Switch Button released
(0, 0, 0)
(0, 0, 0)
(0, 0, 0)
(0, 0, 0)
Switch Button press
(1, 1, 1)
(1, 1, 1)
(1, 1, 1)
(1, 1, 1)
Switch Button released
(0, 0, 0)
(0, 0, 0)
(0, 0, 0)
(0, 0, 0)
```

Lab: Build an ESP32 Micro Trainer. . .

Functional ESP32 Micro Trainer: Pushbutton Switch Test with Data Plotter

Watch the Video Clip!

https://youtu.be/GBdoemfBRds



```
6 # Define GPIO pins
(1, 1, 1)
(1, 1, 1)
Switch Button released
(0, 0, 0)
(0, 0, 0)
(0, 0, 0)
(0, 0, 0)
Switch Button press
(1, 1, 1)
(1, 1, 1)
(1, 1, 1)
(1, 1, 1)
Switch Button released
(0, 0, 0)
(0, 0, 0)
(0, 0, 0)
(0, 0, 0)
Switch Button press
(1, 1, 1)
(1, 1, 1)
(1, 1, 1)
```

Question 5

On slide 44, what line number provides a Tuple to the Mu Plotter?

a) 21
b) 23
c) 7
d) 25

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