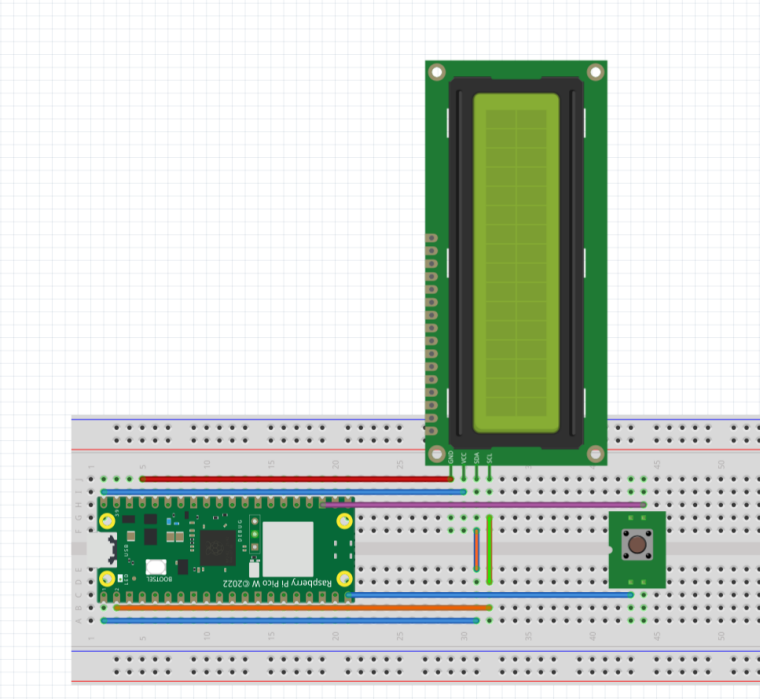
MICROPYTHON • PHYSICAL COMPUTING

Turn On an LCD with a Button

A practical Raspberry Pi Pico W classroom lesson



LEVEL	DURATION	OUTCOME
Beginner	45-60 minutes	Press a button to display two lines of text

1. Lesson overview

In this activity, a Raspberry Pi Pico W reads a push button on GP15. When the button is pressed, a callback function clears a 16 x 2 I2C LCD and displays a two-line message.

Learning objectives

By the end of the lesson, students can...	Evidence of learning
Connect a push button to a digital GPIO pin	Button input is detected reliably
Connect an I2C LCD using SDA and SCL	The LCD address appears during an I2C scan
Define and assign an event callback	The function runs only when the button is pressed
Position text on both LCD rows	Text appears on row 1 and row 2 without overlap

What students need

Component	Quantity	Purpose
Raspberry Pi Pico or Pico W	1	Microcontroller
16 x 2 LCD with I2C backpack	1	Displays the message
Momentary push button	1	Provides user input
Breadboard and jumper wires	1 set	Builds the circuit
USB data cable	1	Power and programming

> Important pin correction
> The original comment says GP20, but the code uses `Button(15)`. Therefore, connect the button to GP15. GP15 is located at physical pin 20 on the Pico header.

Key vocabulary: GPIO - a programmable input/output pin; I2C - a two-wire communication system; callback - a function that runs when an event occurs.

2. Circuit and wiring

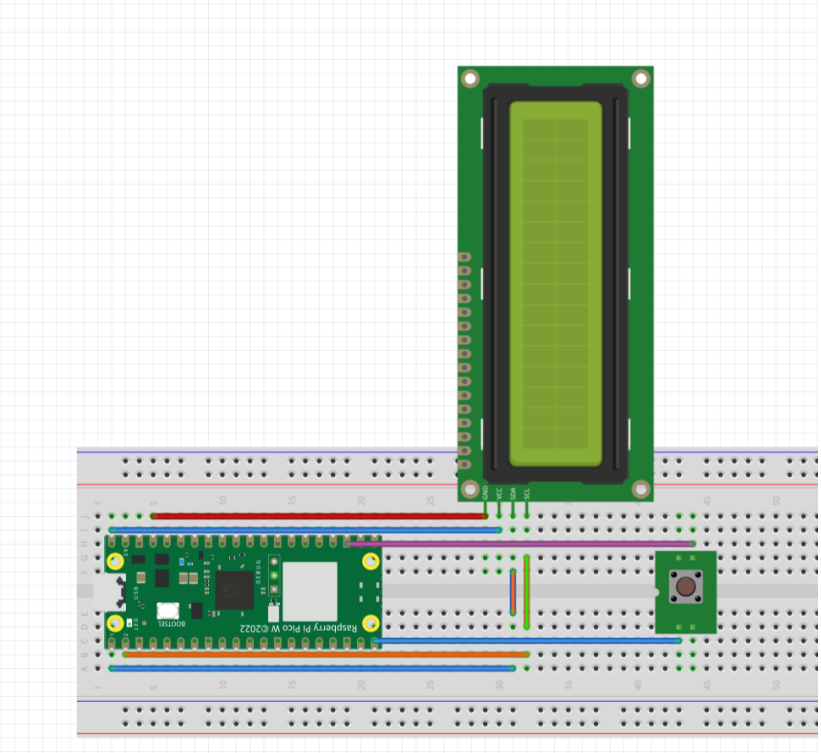


Figure 1. Breadboard layout for the Pico W, I2C LCD and push button.

Connection table

Device terminal	Connect to Pico	Physical pin	Role
LCD GND	GND	Pin 38 or another GND	Common ground
LCD VCC	3V3(OUT)	Pin 36	LCD power*
LCD SDA	GP0	Pin 1	I2C data
LCD SCL	GP1	Pin 2	I2C clock
Button terminal 1	GP15	Pin 20	Digital input
Button terminal 2	GND	Any GND	Completes circuit when pressed

LCD voltage note
Start with 3.3 V if your I2C LCD works at that voltage. Some backpacks require 5 V for a bright, reliable display. Because many boards pull SDA and SCL up to their supply voltage, use a bidirectional I2C level shifter before powering such a backpack from 5 V. Never expose Pico GPIO pins directly to 5 V.

Before powering the circuit

1. Disconnect USB power. 2. Check that VCC and GND are not reversed. 3. Confirm the button crosses the breadboard centre gap. 4. Confirm GP0 goes to SDA and GP1 goes to SCL. 5. Reconnect USB only after inspection.

3. Corrected MicroPython program

Save the program as `main.py` if it should start automatically whenever the Pico receives power.

```
from machine import Pin, I2C
from pico_i2c_lcd import I2cLcd
from picozero import Button

# GP15 is physical pin 20 on the Pico
button = Button(15)

# I2C0: GP0 = SDA and GP1 = SCL
i2c = I2C(
    0,
    sda=Pin(0),
    scl=Pin(1),
    freq=100000
)

# Scan and display detected I2C addresses
devices = i2c.scan()
print("I2C devices:", devices)
for device in devices:
    print("Address:", hex(device))

I2C_ADDR = 0x27
lcd = I2cLcd(i2c, I2C_ADDR, 2, 16)
lcd.clear()

def turnON():
    lcd.clear()
    lcd.move_to(0, 0)
    lcd.putstr("Learn Basic")
    lcd.move_to(0, 1)
    lcd.putstr("IoT RM 650")

# Assign the function; do not add parentheses
button.when_pressed = turnON
```

Why no parentheses?
Write `button.when_pressed = turnON`. This gives the button a reference to the function. Writing `turnON()` would run it immediately during setup.

4. How the program works

1	Import Load GPIO, I2C, LCD and button tools.
2	Configure Set GP15 as the button input and create I2C0 on GP0/GP1.
3	Detect Scan the I2C bus and print any detected device addresses.
4	Prepare Create the LCD object at address 0x27 and clear its screen.
5	Wait Assign turnON as the button-press callback.
6	Display When pressed, clear the LCD and write both text rows.

Event flow

USB supplies power	LCD address is checked	Program remains ready	GP15 becomes active	Callback writes two rows

Expected shell output
A common successful scan prints an address equivalent to `0x27`. The decimal scan list may show `[39]`, because hexadecimal 0x27 equals decimal 39.

5. Testing and troubleshooting

Testing procedure

1. Open the program in Thonny. 2. Select the correct MicroPython interpreter and port. 3. Run the program. 4. Check the Shell for an I2C address. 5. Press and release the button. 6. Confirm both LCD rows appear.

Problem	Likely cause	Recommended check
No I2C devices found	SDA/SCL reversed, no power, or loose wiring	Check GP0-SDA, GP1-SCL, VCC and GND
Address is not 0x27	Backpack uses a different address	Replace I2C_ADDR with the address printed by the scan
Backlight on, no text	Contrast setting or wrong address	Turn the small contrast potentiometer and verify the scan
Text appears immediately	Callback was called during assignment	Use turnON without parentheses
Button does nothing	Connected to GP20 instead of GP15, or wrong button legs	Connect GP15 (physical pin 20) to GND through the button
Repeated/unwanted triggers	Mechanical switch bounce	Use picozero's button handling; add a short software guard if needed

Success checklist

☐ The Pico is detected by the computer.
☐ The LCD address is printed.
☐ No 5 V signal reaches a Pico GPIO pin.
☐ Pressing the button displays “Learn Basic.”
☐ The second row displays “IoT RM 650.”

Safety
Switch off or disconnect USB before changing the wiring. Do not power motors, servos or high-current LED strips directly from the Pico's 3.3 V output.

6. Student challenge and review

Challenge A - Personalise the message

Change the first and second rows to display your name and class. Remember that each LCD row can show a maximum of 16 characters.

Challenge B - Clear the display on release

Create another function called `turn_off()` and assign it to `button.when_released`.

```
def turn_off():
    lcd.clear()

button.when_released = turn_off
```

Review questions

Question	Student answer
1. Which GPIO pin reads the button?	
2. Which pins carry I2C data and clock?	
3. Why is turnON written without parentheses?	
4. What should you change if the scan reports 0x3F?	
5. What is the maximum number of characters on each LCD row?	

Teacher answer key

1. GP15. 2. GP0 is SDA and GP1 is SCL. 3. It assigns the function as a callback instead of running it immediately. 4. Change `I2C_ADDR` to `0x3F`. 5. Sixteen characters.

> Extension idea
> Add a second button that displays a different message, or use a counter so each press cycles through several screens.

End of lesson • Raspberry Pi Pico W • MicroPython • I2C LCD • Digital input