Raspberry Pi Quick Manual: OS Install + Python I/O (Digital & Analog)

Works with Raspberry Pi 4/400/5; Raspberry Pi OS (64-bit).

---

1) What you need

- Raspberry Pi + 5V USB-C/µUSB power supply (3A for Pi 4/5 recommended)
- micro-SD card (≥16 GB, Class 10 or better)
- micro-SD card reader
- Optional: HDMI display + keyboard/mouse (headless is fine)
- Breadboard, LEDs + 330Ω resistors, push button, jumper wires
- For analog input: **MCP3008** 10-bit ADC (SPI) + a potentiometer/sensor
- For analog output (optional): **MCP4725** I^2C DAC or simply PWM on a GPIO

> **Voltage note:** Raspberry Pi GPIO is **3.3 V only** (not 5 V). Absolute max per pin ~16 mA; total across all pins ≤50 mA. Always use series resistors with LEDs (≈220–470 Ω).

---

2) Install Raspberry Pi OS using Raspberry Pi Imager

1. Download **Raspberry Pi Imager** (Windows/macOS/Linux) from raspberrypi.com.
2. Insert the micro-SD card and open Imager.
3. **Choose OS** → *Raspberry Pi OS (64-bit)*.
4. **Choose Storage** → your micro-SD.
5. Click the ⚙️**gear icon (Advanced options)** and set:
6. Hostname (e.g., `raspberrypi`)
7. Username/password
8. Wi-Fi SSID + password + country (for headless)
9. Enable **SSH**
10. Locale/timezone/keyboard
11. **Write** → wait for verify → eject SD → insert into Pi.
12. First boot (screen or headless via SSH `ssh <user>@<hostname>.local`).

> If you skipped Advanced options: after first boot run `sudo raspi-config` → *Interface Options* to enable **SSH**, **SPI**, **I2C**; set Wi-Fi in *System Options*.

---

3) First-time setup (terminal)

```
sudo apt update && sudo apt -y full-upgrade
sudo apt install -y python3-pip python3-gpiozero python3-rpi.gpio
```

```
    python3-spidev python3-smbus i2c-tools
# Optional tools
sudo apt install -y git nano
# Reboot after enabling interfaces
sudo raspi-config  # enable SPI/I2C if needed
sudo reboot
```

Check buses:

```
i2cdetect -y 1   # should show addresses if I²C devices are connected
ls /dev/spidev*  # should show /dev/spidev0.0 and/or 0.1
```

---

4) Pinout (quick)

- **3.3 V** supply: Pin 1 or 17
- **5 V** supply: Pin 2 or 4 (do **not** feed GPIOs with 5 V)
- **Ground**: Pins 6,9,14,20,25,30,34,39
- **GPIO examples used below:**
- LED on **GPIO 17** (Pin 11)
- Button on **GPIO 23** (Pin 16)
- SPI0: CE0 **GPIO8**, MISO **GPIO9**, MOSI **GPIO10**, SCLK **GPIO11**
- I^{2}C1: SDA **GPIO2**, SCL **GPIO3**

Use `pinout` command in terminal to view the full map.

---

5) Digital Output: Blink an LED (gpiozero)

Wiring: LED anode → GPIO 17; LED cathode → 330Ω → GND.

```
# blink_led.py
from gpiozero import LED
from time import sleep

led = LED(17)

while True:
    led.on()
    sleep(0.5)
    led.off()
    sleep(0.5)
```

Run: `python3 blink_led.py`

PWM fade ("analog-like" brightness)

```
# pwm_fade.py
from gpiozero import PWMLED
from time import sleep

led = PWMLED(17)

while True:
    for v in [i/100 for i in range(0,101)]:
        led.value = v  # 0.0..1.0
        sleep(0.01)
    for v in [i/100 for i in range(100,-1,-1)]:
        led.value = v
        sleep(0.01)
```

---

6) Digital Input: Button with internal pull-up

Wiring: Button between **GPIO 23** and **GND** (no external resistor needed).

```
# button_read.py
from gpiozero import Button
from signal import pause

btn = Button(23, pull_up=True)  # pressed = GPIO23 pulled to GND

btn.when_pressed = lambda: print("Pressed")
btn.when_released = lambda: print("Released")

print("Waiting for button events...")
pause()
```

Debounced read (manual loop):

```
from gpiozero import Button
from time import sleep

btn = Button(23, pull_up=True, bounce_time=0.05)

while True:
    if btn.is_pressed:
        print("Pressed")
    sleep(0.1)
```

---

7) Analog Input (ADC) with MCP3008 (SPI)

> Raspberry Pi has **no built-in ADC**. Use **MCP3008** (10-bit). You can also use MCP3208 (12-bit) similarly.

Wiring (MCP3008 ↔ Pi): - VDD → 3.3 V, VREF → 3.3 V, AGND → GND, DGND → GND - CLK → GPIO11 (SCLK), DOUT → GPIO9 (MISO), DIN → GPIO10 (MOSI) - CS/SHDN → GPIO8 (CE0) - CH0 → wiper of potentiometer (sides to 3.3 V & GND) or sensor output

Option A: High-level `gpiozero.MCP3008`

```
# mcp3008_gpiozero.py
from gpiozero import MCP3008
from time import sleep

# channel=0 uses CE0 by default; value is 0.0..1.0 (fraction of Vref)
adc = MCP3008(channel=0)

while True:
    volts = adc.value * 3.3
    print(f"CH0 = {adc.value:.4f} ({volts:.3f} V)")
    sleep(0.2)
```

Option B: Low-level `spidev`

```
# mcp3008_spidev.py
import spidev, time

spi = spidev.SpiDev()
spi.open(0, 0)            # bus 0, device CE0
spi.max_speed_hz = 1350000

def read_ch(ch):
    assert 0 <= ch <= 7
    # MCP3008 protocol: start bit=1, single-ended=1, channel bits, null bit
    cmd1 = 1
    cmd2 = 0b1000_0000 | (ch << 4)
    resp = spi.xfer2([cmd1, cmd2, 0])
    value = ((resp[1] & 3) << 8) | resp[2]
    return value

while True:
    raw = read_ch(0)
    volts = raw * 3.3 / 1023
    print(f"CH0 raw={raw:4d}  {volts:.3f} V")
    time.sleep(0.2)
```

---

8) Analog Output (DAC) options

The Pi has no true analog output. Use: - **PWM on a GPIO** filtered with RC (simple/cheap; good for LED/servo/approx analog) - **MCP4725** I²C DAC for real analog (12-bit)

A) PWM as analog (duty cycle 0.0–1.0)

```
# pwm_as_dac.py
from gpiozero import PWMLED
from time import sleep

vout = PWMLED(18, frequency=1000)  # GPIO18 supports hardware PWM

for duty in [i/20 for i in range(0,21)]:
    vout.value = duty
    print("Duty:", duty)
    sleep(0.3)
```

Add an **RC low-pass** (e.g., R=1 kΩ, C=10 µF) from GPIO18 to GND to smooth.

B) MCP4725 DAC (I²C)

Wiring: MCP4725 VCC→3.3 V, GND→GND, SDA→GPIO2, SCL→GPIO3. Address typically 0x60.

```
# mcp4725_set.py
import smbus, time

bus = smbus.SMBus(1)
ADDR = 0x60

def set_dac(val):
    # 12-bit value 0..4095
    val = max(0, min(4095, val))
    high = (val >> 4) & 0xFF
    low  = (val & 0xF) << 4
    bus.write_i2c_block_data(ADDR, 0x40, [high, low])  # fast write to DAC

for v in range(0, 4096, 256):
    set_dac(v)
    print("DAC:", v)
    time.sleep(0.3)
```

9) Combining Digital + Analog Example

Goal: Press button → read potentiometer (MCP3008 CH0) → scale and set PWM brightness.

```
# button_adc_pwm.py
from gpiozero import Button, MCP3008, PWMLED
from signal import pause

btn = Button(23, pull_up=True)
adc = MCP3008(channel=0)
led = PWMLED(17)

def update_brightness():
    level = adc.value  # 0..1
    led.value = level
    print(f"Button press → LED brightness set to {level:.2f}")

btn.when_pressed = update_brightness
print("Press the button to sample the potentiometer and set brightness.")
pause()
```

10) Troubleshooting Quick Tips

- **No I²C device shown?** Check `sudo raspi-config` → Interface Options → I2C = Enabled; confirm pull-ups (many breakout boards include them). Run `i2cdetect -y 1`.
- **SPI not working?** Enable SPI; confirm wiring (MISO/MOSI/SCLK/CE). Use short wires.
- **Permission errors on GPIO/I²C/SPI?** Use Python 3; most libs run as user. If needed, `sudo adduser $USER i2c` and `sudo adduser $USER spi`, then reboot.
- **Random resets/undervoltage icon?** Use a solid 5 V power supply and good USB-C cable.
- **LED not lighting?** Flip LED orientation; confirm resistor; verify correct pin number (BCM vs BOARD).

11) Safe Shutdown

```
sudo poweroff   # or: sudo shutdown -h now
```

Always shut down before removing power to avoid SD card corruption.

---

12) Next Steps

- Explore `gpiozero` docs for sensors/actuators (servos, distance sensors)
- Log sensor data with `pandas` / `matplotlib`
- Build a Flask/FastAPI web dashboard to control GPIOs over the network

---

Appendix: Using RPi.GPIO instead of gpiozero (optional)

```
# rpi_gpio_blink.py
import RPi.GPIO as GPIO, time
GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.OUT)
try:
    while True:
        GPIO.output(17, 1); time.sleep(0.5)
        GPIO.output(17, 0); time.sleep(0.5)
except KeyboardInterrupt:
    pass
finally:
    GPIO.cleanup()
```