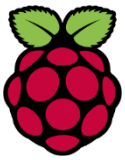
create.withcode.uk

Raspberry Pi Python GPIO Quick start guide

The GPIO pins on a Raspberry Pi allow you to connect your Pi up to all sorts of electronic devices. This guide talks you through how to set up and control the inputs and outputs using python.



- GPIO stands for General Purpose Input and Output.
- GPIO pins are the metal spikes that stick out of a RPi.
- Older RPis have 26 pins and newer ones have 40 pins.
- Some of those pins are for power rather than inputs or outputs

RPi Model	Number of GPIO pins	Number of inputs or outputs
A	26	17
B	26	17
A+	40	28
B+	40	28

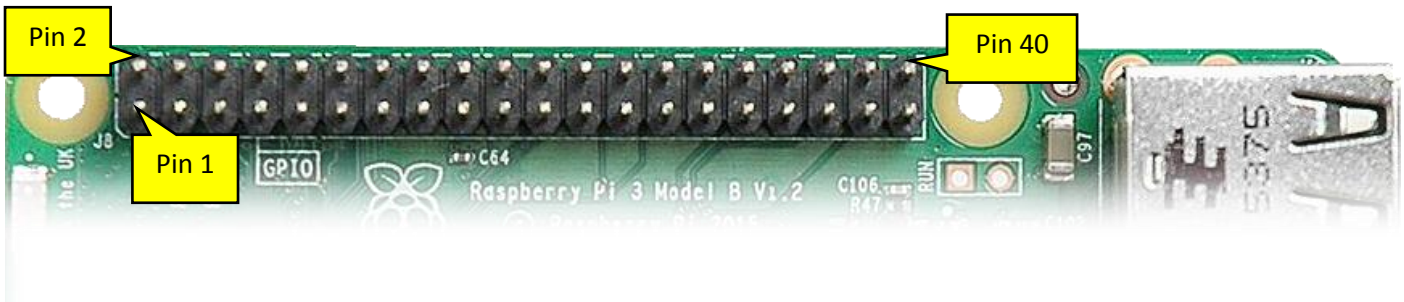
Pins:

- Pins are either power points or I/O pins
- Power pins are hard wired to 0v (ground), +3.3v or +5v. You can't change them.
- I/O pins can be set to either an input or an output
- Inputs read a digital value into the RPi (e.g. has a switch been pressed?)
- Output send a digital value out from the RPI (e.g. switch an LED on or off)

Pins 27-40 not available on older RPi models

Pin numbering:

- There are two numbering systems for GPIO pins.
- Physical numbering is the easiest to understand: it tells you where to find the pin on the physical RPi board (see below)
- BCM numbers are only for I/O pins. They don't match the physical numbers but come from the way that they are connected to the processor on the RPi.
- When writing your code you can choose to use either physical numbering or BCM numbering



Warning:

- You might permanently break your RPi if you're not careful when connecting anything to the GPIO pins.
- Short circuits (connecting +3v or +5 either from a power pin or an output pin directly to ground) will damage your RPi
- Whilst your GPIO pins have power pins, there's a limit to how much power they can provide. You may need an external power supply if you're controlling a circuit that needs anything more than a few LEDs & switches.
- Never connect a motor or speaker directly to a GPIO pin. The electrical feedback can cause damage.

Controlling the GPIO in python

- You can simulate and test these commands without a RPi on create.withcode.uk

Import the GPIO module

```
import RPi.GPIO as GPIO
```

You'll need to do this once at the start of your code so that you can use the GPIO module to access the inputs and outputs.

Setup inputs and outputs

- All I/O pins are set to be inputs unless you write code to make them outputs.

```
GPIO.setmode(GPIO.BOARD)
GPIO.setup(3, GPIO.OUT)
```

This example sets the RPi to use physical pin numbering then sets physical pin 3 to be an output.

- Change **GPIO.BOARD** to **GPIO.BCM** if you want BCM pin numbering instead of physical pin numbering
- Change **GPIO.OUT** to **GPIO.IN** if you want to set a pin back to being an input

Reading inputs and setting outputs

```
1  # import modules
2  import RPi.GPIO as GPIO
3  import time
4
5  # setup pins
6  GPIO.setmode(GPIO.BOARD)
7  GPIO.setup(3, GPIO.OUT)
8  GPIO.setup(5, GPIO.IN)
9
10 # loop 5 times
11 for i in range(5):
12
13     # flash output pin 3
14     GPIO.output(3, GPIO.HIGH)
15     time.sleep(1)
16     GPIO.output(3, GPIO.LOW)
17     time.sleep(1)
18
19     # read input pin 5
20     if GPIO.input(5) == GPIO.HIGH:
21         print("Pin 5 is on")
22     else:
23         print("Pin 5 is off")
```

The **time** module is useful for adding delays to pause your program

We've set up physical pin 3 (BCM2) as an output and physical pin 5 (BCM 3) as an input:

GPIO.output sends a value to an output pin

time.sleep(1) pauses the program for 1 second

GPIO.input reads a value from an input pin

You can run this code online with a simulated Raspberry Pi here;
https://create.withcode.uk/python/A3