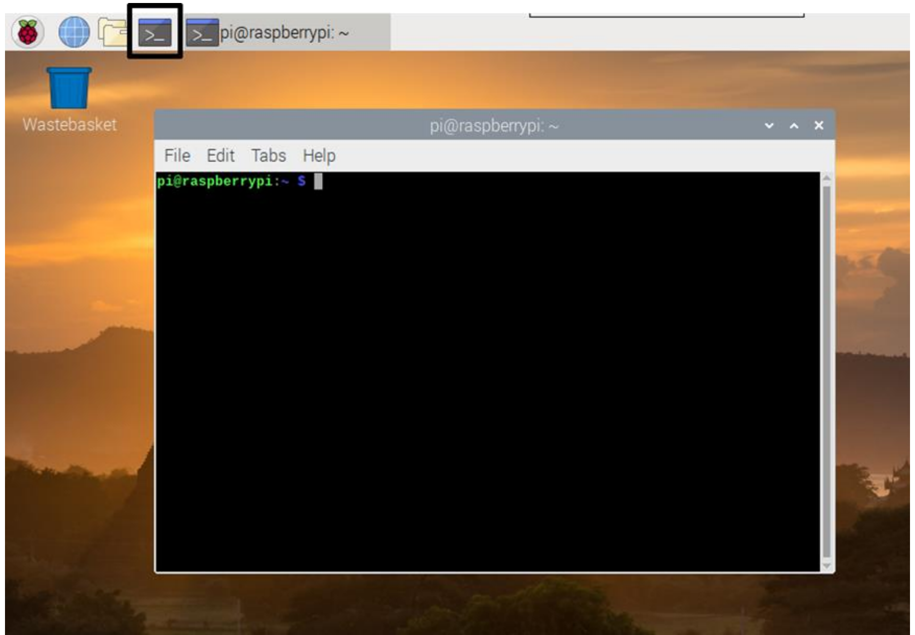
# Fundamental of
# Cognitive Interaction with Robots

**Lecture 3**

# The Terminal

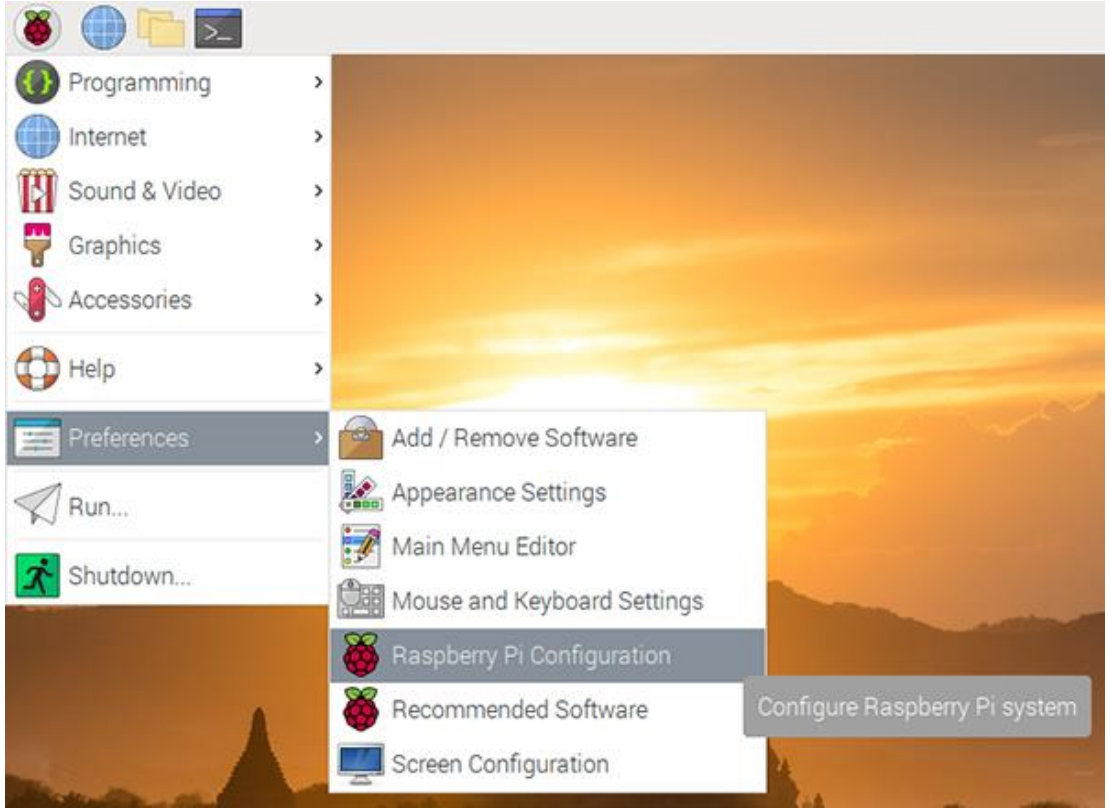
- Raspberry Pi OS has a powerful command line interface (terminal) that gives you a lot more control over the computer than you can get using the GUI.
- Many important tasks are either easier or only possible via commands.
- To open a terminal, either click the Terminal icon or hit CTRL+ ALT + T.
- If you connect to your Pi via SSH or you already booted to the command prompt, you don't need to open terminal, because you're already there.

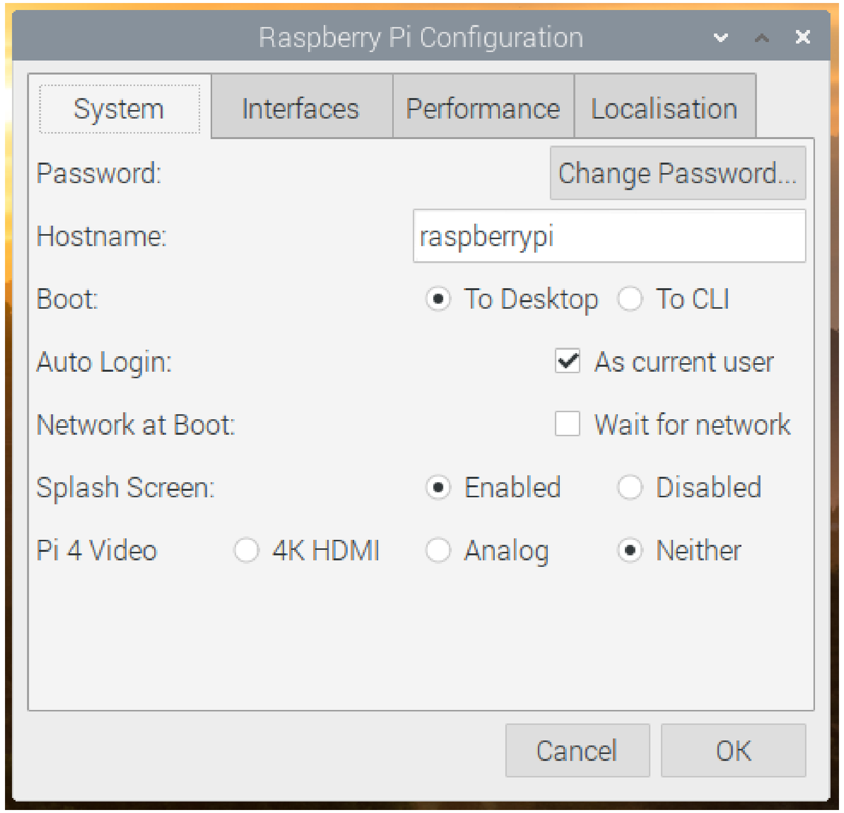


```
pi@raspberrypi:~ $
```

# Configuring the Raspberry Pi

- You can control most of the Raspberry Pi’s settings through the Raspberry Pi Configuration application found in Preferences on the menu, or using the command **`sudo raspi-config`**.
- You can configure System, Interfaces, Performance, and Localisation.




Raspberry Pi Configuration

System | Interfaces | Performance | Localisation

Password: Change Password...

Hostname: raspberrypi

Boot: ● To Desktop ○ To CLI

Auto Login: ☑ As current user

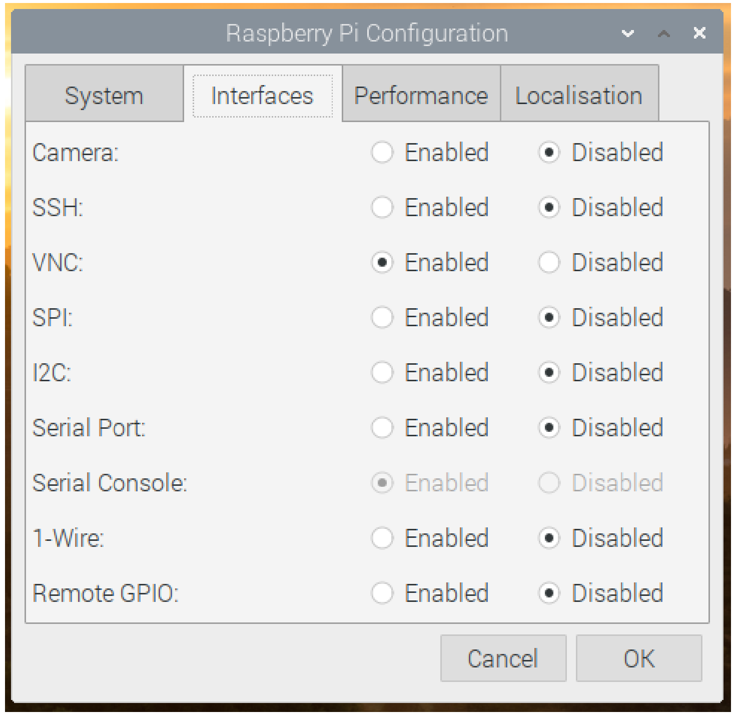
Network at Boot: □ Wait for network

Splash Screen: ● Enabled ○ Disabled

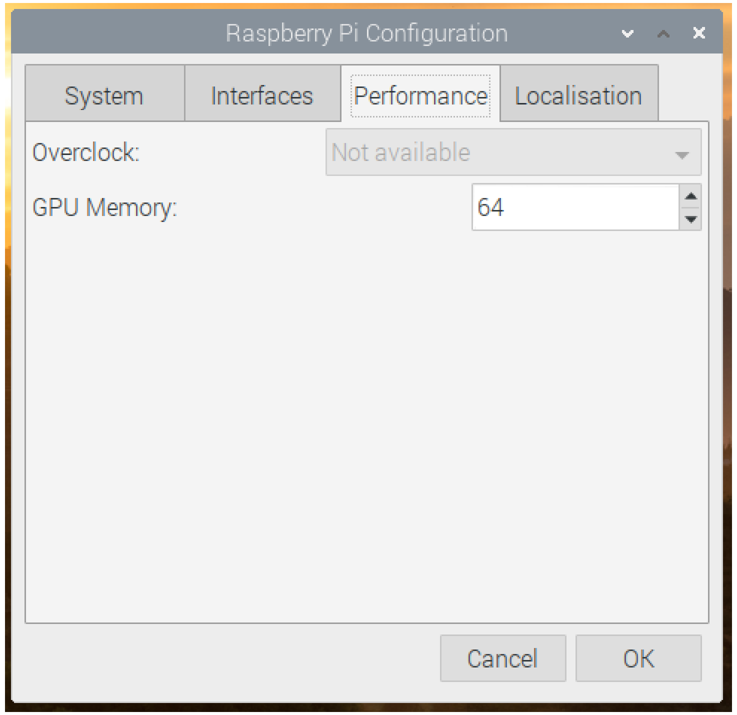
Pi 4 Video ○ 4K HDMI ○ Analog ● Neither

Cancel OK

# Configuring the Raspberry Pi

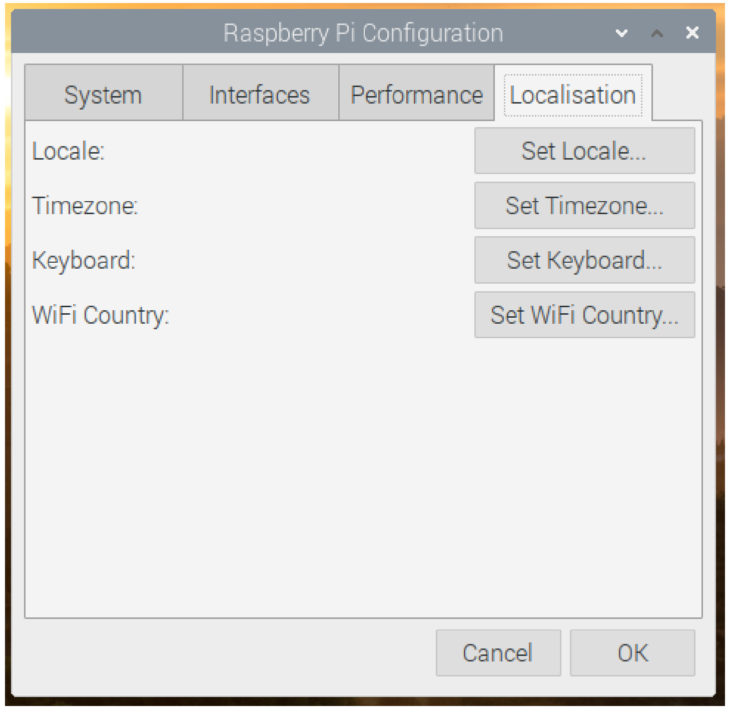


| | | |
|---|---|---|
| Camera: | ○ Enabled | ● Disabled |
| SSH: | ○ Enabled | ● Disabled |
| VNC: | ● Enabled | ○ Disabled |
| SPI: | ○ Enabled | ● Disabled |
| I2C: | ○ Enabled | ● Disabled |
| Serial Port: | ○ Enabled | ● Disabled |
| Serial Console: | ● Enabled | ○ Disabled |
| 1-Wire: | ○ Enabled | ● Disabled |
| Remote GPIO: | ○ Enabled | ● Disabled |



Overclock: Not available

GPU Memory: 64



Locale: Set Locale...

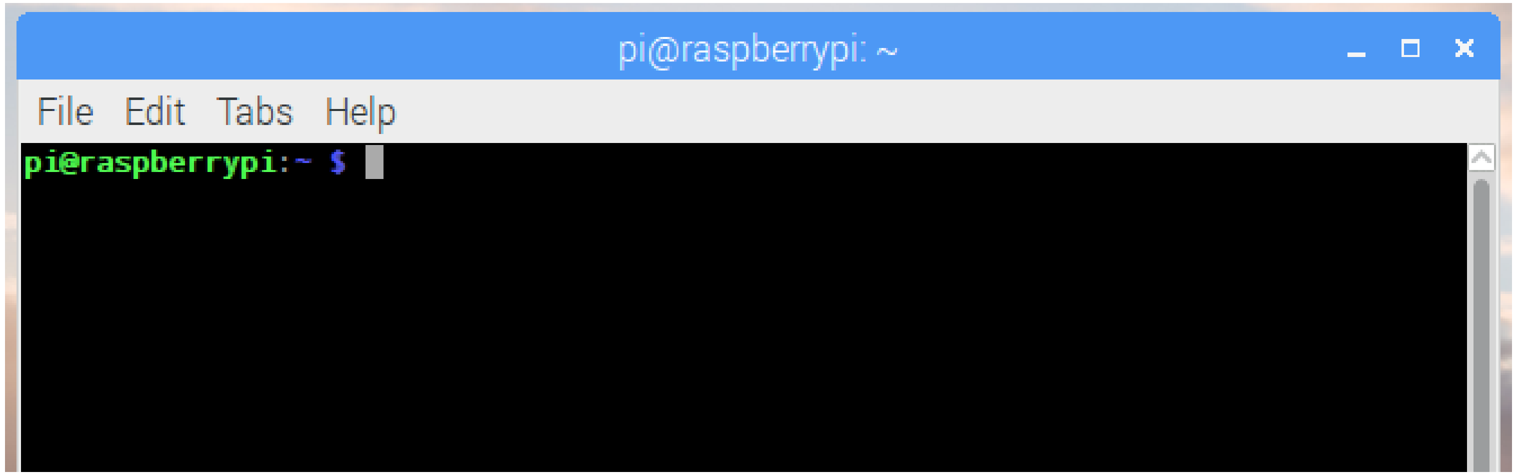
Timezone: Set Timezone...

Keyboard: Set Keyboard...

WiFi Country: Set WiFi Country...

# The Terminal

The Prompt:



```
pi@raspberrypi:~ $
```

```
pi@raspberrypi:~ $
```

- The prompt shows the username and the hostname (machine name) of the Pi.
- Here we are logged in as a user called pi and the machine is called raspberrypi.
- The user pi has permission to edit any file in the home directory, which is /home/pi/.
- But we cannot change the underlying filesystem or modifying outside the home directory as we do not have permission to do so.
- To make wide changes we either need to be a user called “root” which is similar to administrator on Windows, or to use sudo to temporarily give us extra permissions.

# The Terminal

## Login as root:

- Be aware that some commands executed from root account can harm the filesystem
- The root account is not active by default, you need to do:

```
pi@raspberrypi:~ $ sudo passwd root
```

- Then you are prompted to set the password.

- Then you can change to the root user using

```
pi@raspberrypi:~ $ su root
```

- After that you are logged in as the root user

```
root@raspberrypi:/home/pi#
```

**Linux commands are case sensitive**

# The Terminal

## sudo - Super User Do

- To perform any core tasks like installing/removing software from your current “pi” user, you have to write the word “sudo” before any core command.
- By doing so, you have admin rights for that execution.
- To use “sudo” or change from a user account to the “root” account, you have to be in the "sudoers" permission group.
- Fortunately, the default Raspberry Pi user “pi” is already in this group.
- For example,

```
sudo apt update
```

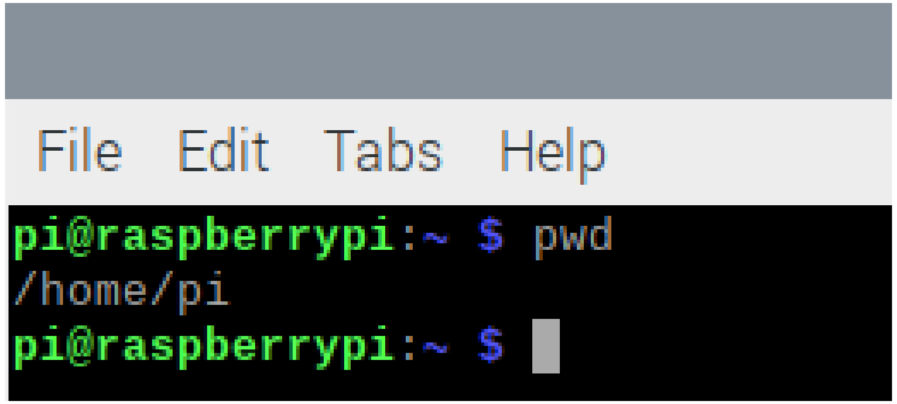
## passwd – change the password for the current user

```
passwd
```

# Navigating the file system

## pwd - Print working directory

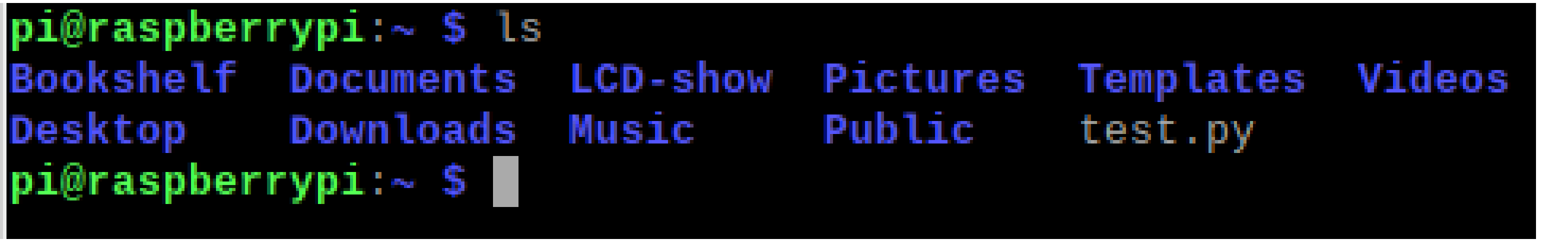
- This command will show the full path to the directory we are in.



```
pi@raspberrypi:~ $ pwd
/home/pi
pi@raspberrypi:~ $
```

## ls - List directory content

- This command is used to list the contents of a directory.

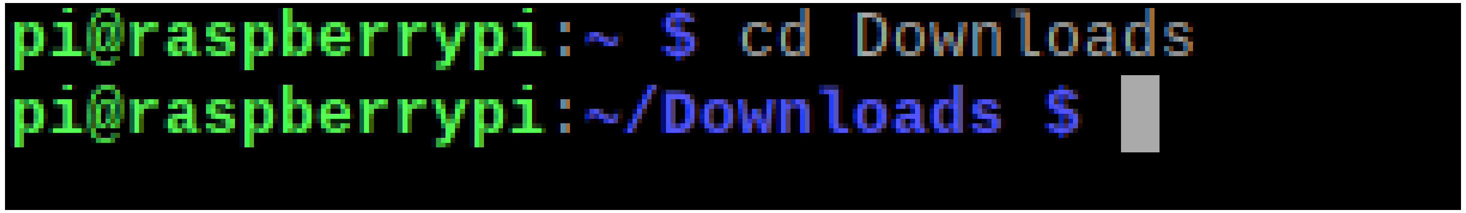


```
pi@raspberrypi:~ $ ls
Bookshelf  Documents  LCD-show  Pictures  Templates  Videos
Desktop    Downloads  Music     Public    test.py
pi@raspberrypi:~ $
```

# Navigating the file system

## cd - Change directory

- This command is used to change the current directory.
- For example, to move from our home directory to Downloads directory:



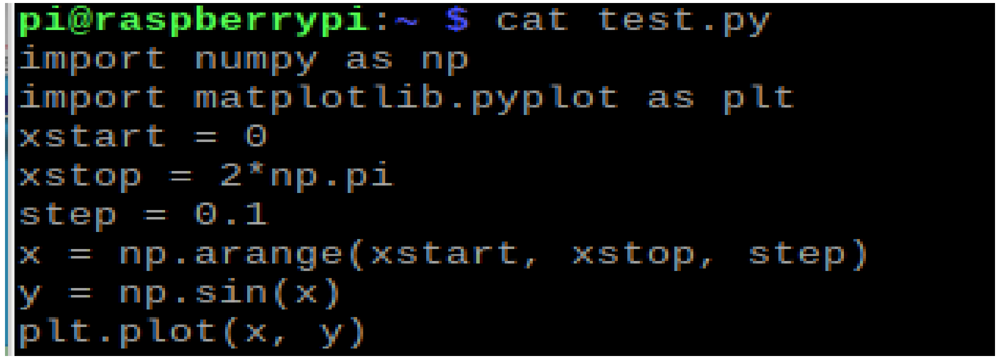
```
pi@raspberrypi:~ $ cd Downloads
pi@raspberrypi:~/Downloads $
```

- To go back to the previous directory that we were in:
  **`cd -`**
- To go back to the directory immediately above the current directory:
  **`cd ..`**
- To go back to our home directory:
  **`cd ~`**

# Working with files

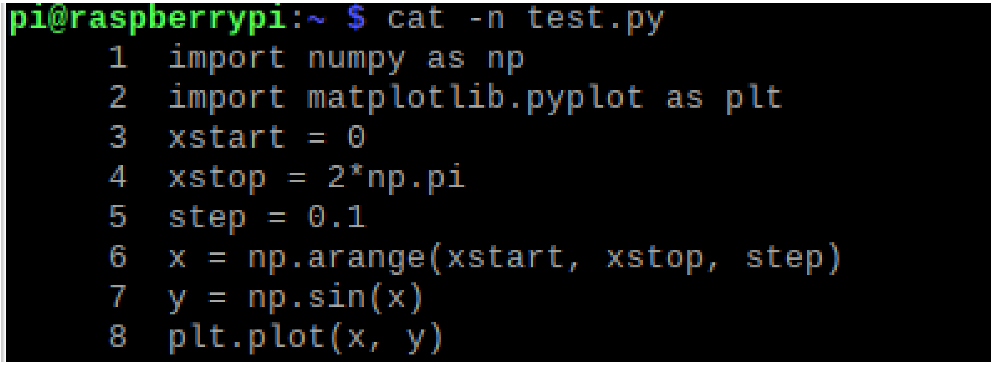
## cat – display (concatenate) the lines of a file to the terminal

- Print the contents of a file to the terminal,
- For example, a Python file: **`cat test.py`**



```
pi@raspberrypi:~ $ cat test.py
import numpy as np
import matplotlib.pyplot as plt
xstart = 0
xstop = 2*np.pi
step = 0.1
x = np.arange(xstart, xstop, step)
y = np.sin(x)
plt.plot(x, y)
```

- Print the contents of a file to the terminal with line numbers: **`cat –n test.py`**



```
pi@raspberrypi:~ $ cat -n test.py
     1  import numpy as np
     2  import matplotlib.pyplot as plt
     3  xstart = 0
     4  xstop = 2*np.pi
     5  step = 0.1
     6  x = np.arange(xstart, xstop, step)
     7  y = np.sin(x)
     8  plt.plot(x, y)
```

# Edit a file

## nano

- Nano is a command-line editor.
- Create a new file, for example newfile.txt.

```
nano newfile.txt
```

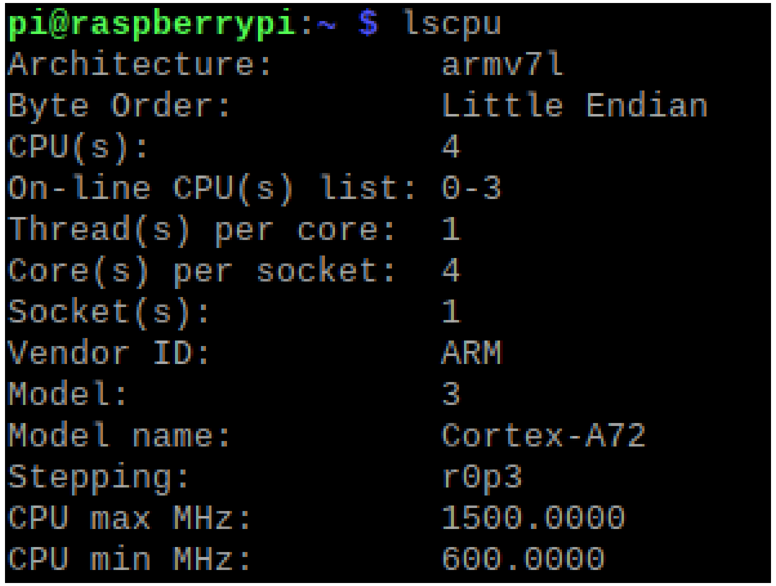
- Edit an existing file, for example test.py.

```
nano test.py
```

- Inside nano we navigate using the arrow keys and it works just like a regular text editor.

# System Information

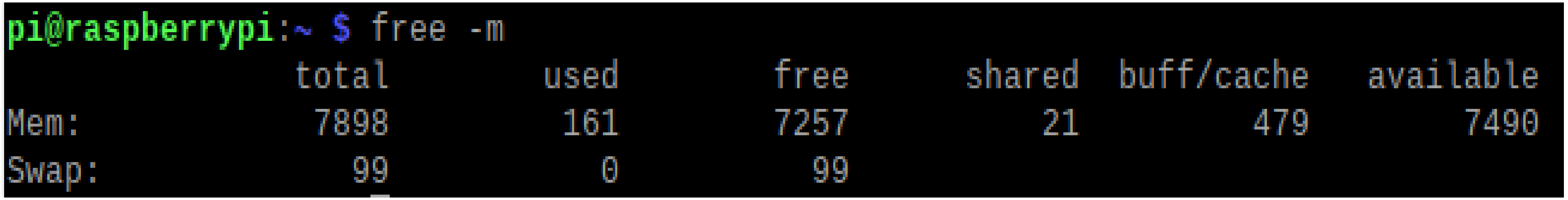
## lscpu – display cpu information



```
pi@raspberrypi:~ $ lscpu
Architecture:        armv7l
Byte Order:          Little Endian
CPU(s):              4
On-line CPU(s) list: 0-3
Thread(s) per core:  1
Core(s) per socket:  4
Socket(s):           1
Vendor ID:           ARM
Model:               3
Model name:          Cortex-A72
Stepping:            r0p3
CPU max MHz:         1500.0000
CPU min MHz:         600.0000
```

## free - Show amount of free and used RAM

- Using the -m option we can set the values in MB.



```
pi@raspberrypi:~ $ free -m
              total        used        free      shared  buff/cache   available
Mem:           7898         161        7257          21         479        7490
Swap:            99           0          99
```

# File Management

## mv - Move / rename a file

- This command offers two functions. We can move a file from one location to another. For example here we move test.py to the Documents directory.

```
mv test.py Documents/
```

- The command can also be used to rename a file or directory. Here we rename test.py to test2.py.

```
mv test.py test2.py
```

## rm - Delete a file

- With this command we can delete files and directories. In this example we delete the file test.py.

```
rm test.py
```

# File Management

## cp - Copy a file

- To copy a file, for example test.py to our Documents directory.

```
cp test.py Documents/
```

- To copy a directory, we need to use the -r option.

```
cp -r test2/ Documents/
```

## mkdir - Create a directory

- Create a new directory to store work. For example let's create a directory called Work in our home directory.

```
mkdir Work
```

## clear: Clear the Terminal Window

```
clear
```

# Software Installation

## apt - Install and manage software

- apt, the **A**dvanced **P**ackaging **T**ool. The app store of Linux. To use apt, we will need to use sudo as it will make changes to the operating system.
- First we update the list of installable software.

```
sudo apt update
```

- Then we can install a specific package/application:

```
sudo apt install <package-name>
```

- Or we can upgrade all of the software on our Raspberry Pi. Note that for this command we pass the -y option to automatically agree to install every package. But this is optional.

```
sudo apt upgrade -y
```

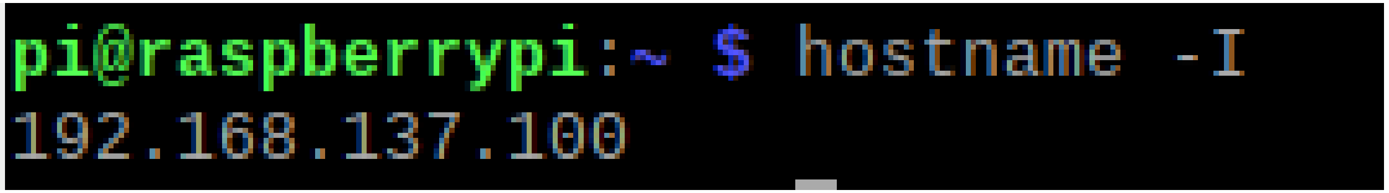
# Network Connectivity & Internet

## ping - Check that we are connected

- The ping command is used to test that our Raspberry Pi is connected to the Internet/local network.
- Ping a website: `ping google.com`
- Ping an IP address: `ping 8.8.8.8`
- Ping a local IP: `ping 192.168.1.1`

## hostname - Get the IP address of the Raspberry Pi

- Use the – I (uppercase i) option to show all IP addresses (WiFi and Ethernet)

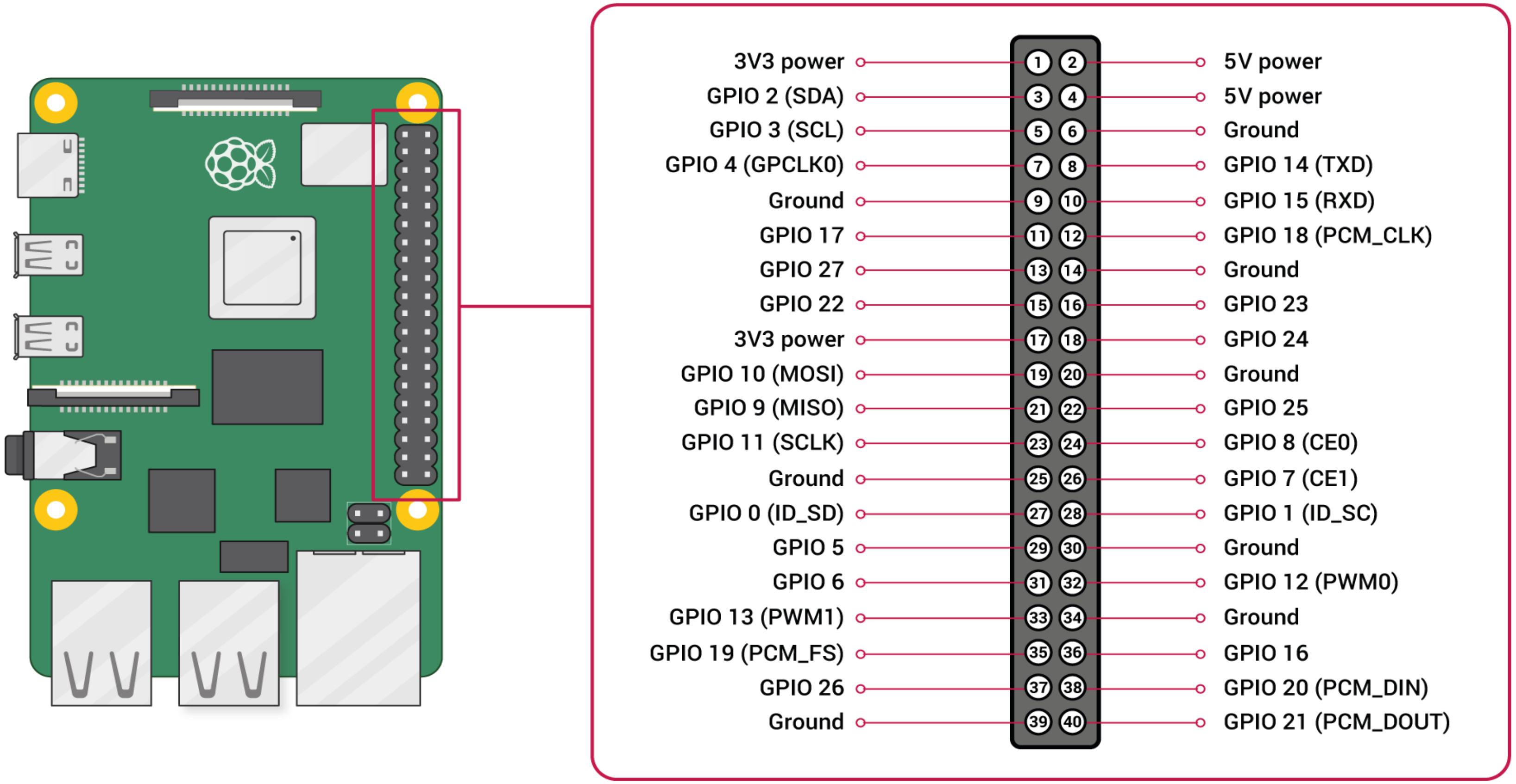


```
pi@raspberrypi:~ $ hostname -I
192.168.137.100
```

- Or you can use **`ifconfig`**

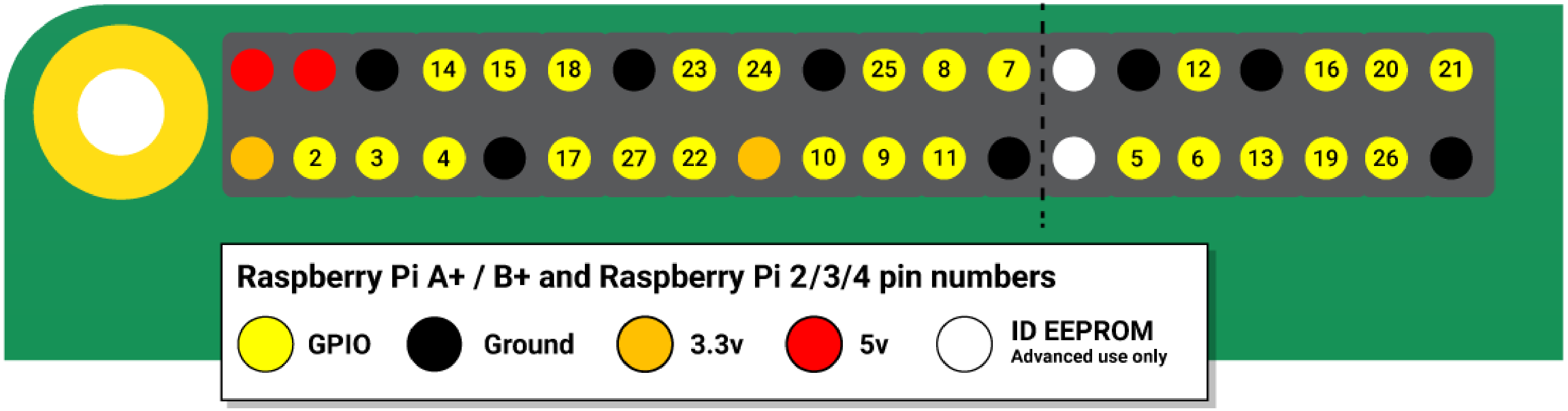
# GPIO pins in Raspberry Pi

- Raspberry Pi boards contain 40 GPIO (General Purpose Input/Output) pins.
- GPIO pins can be used to interface with the physical world whether by reading data from sensors, using components such as LEDs and displays, or controlling motors.



| | | | |
|---|---|---|---|
| 3V3 power | 1 | 2 | 5V power |
| GPIO 2 (SDA) | 3 | 4 | 5V power |
| GPIO 3 (SCL) | 5 | 6 | Ground |
| GPIO 4 (GPCLK0) | 7 | 8 | GPIO 14 (TXD) |
| Ground | 9 | 10 | GPIO 15 (RXD) |
| GPIO 17 | 11 | 12 | GPIO 18 (PCM_CLK) |
| GPIO 27 | 13 | 14 | Ground |
| GPIO 22 | 15 | 16 | GPIO 23 |
| 3V3 power | 17 | 18 | GPIO 24 |
| GPIO 10 (MOSI) | 19 | 20 | Ground |
| GPIO 9 (MISO) | 21 | 22 | GPIO 25 |
| GPIO 11 (SCLK) | 23 | 24 | GPIO 8 (CE0) |
| Ground | 25 | 26 | GPIO 7 (CE1) |
| GPIO 0 (ID_SD) | 27 | 28 | GPIO 1 (ID_SC) |
| GPIO 5 | 29 | 30 | Ground |
| GPIO 6 | 31 | 32 | GPIO 12 (PWM0) |
| GPIO 13 (PWM1) | 33 | 34 | Ground |
| GPIO 19 (PCM_FS) | 35 | 36 | GPIO 16 |
| GPIO 26 | 37 | 38 | GPIO 20 (PCM_DIN) |
| Ground | 39 | 40 | GPIO 21 (PCM_DOUT) |

# GPIO pins in Raspberry Pi



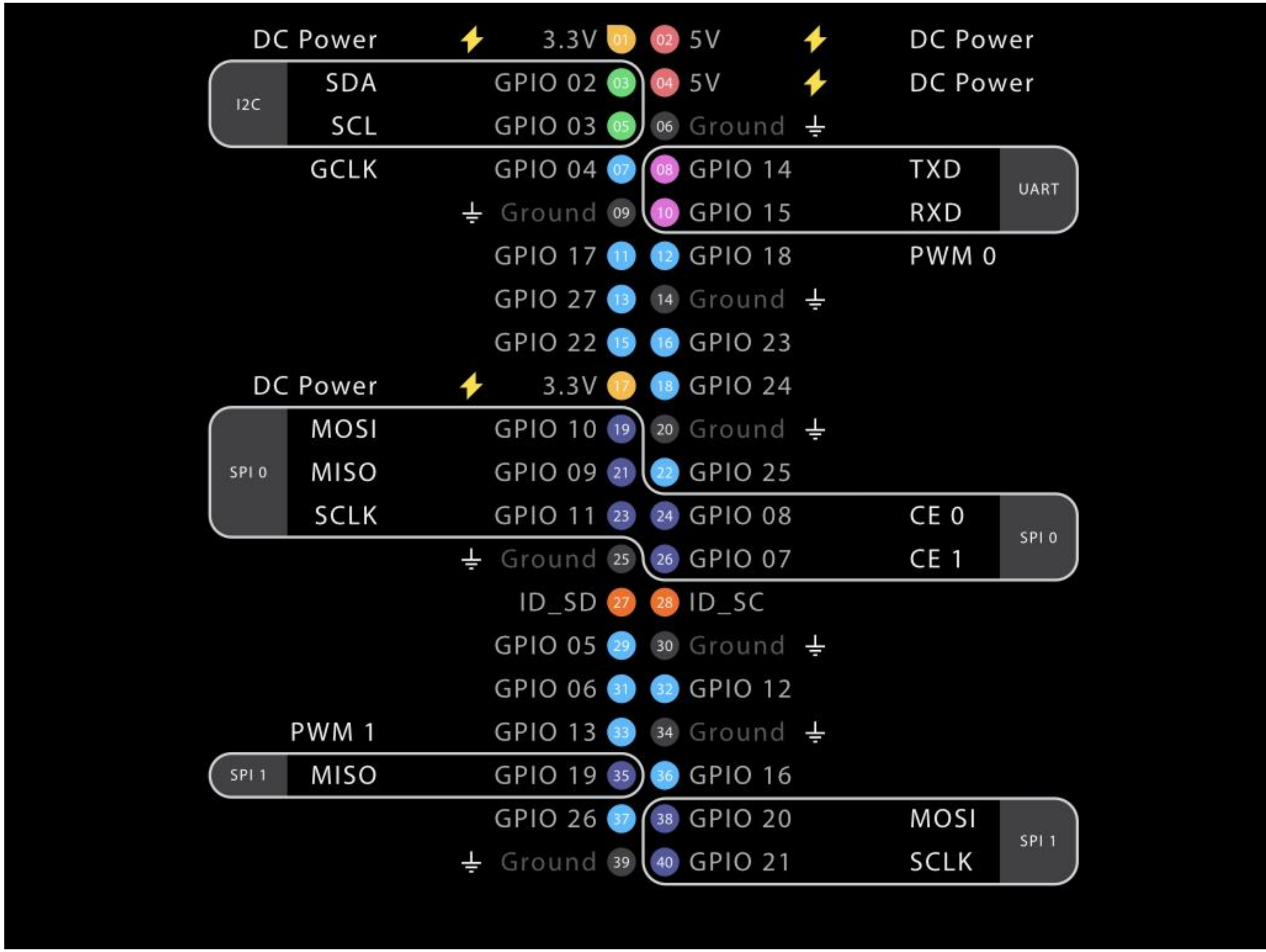
- The numbering of the GPIO pins is not in numerical order; GPIO pins 0 and 1 are present on the board (physical pins 27 and 28) but are reserved for advanced use.
- Any of the GPIO pins can be designated (in software) as an input or output pin.
- **Voltages:**

  Two 5V pins and two 3.3V pins are present on the board, as well as a number of ground pins.

  The remaining pins are all general purpose 3.3V pins, meaning outputs are set to 3.3V and inputs are 3.3V-tolerant.

# More !

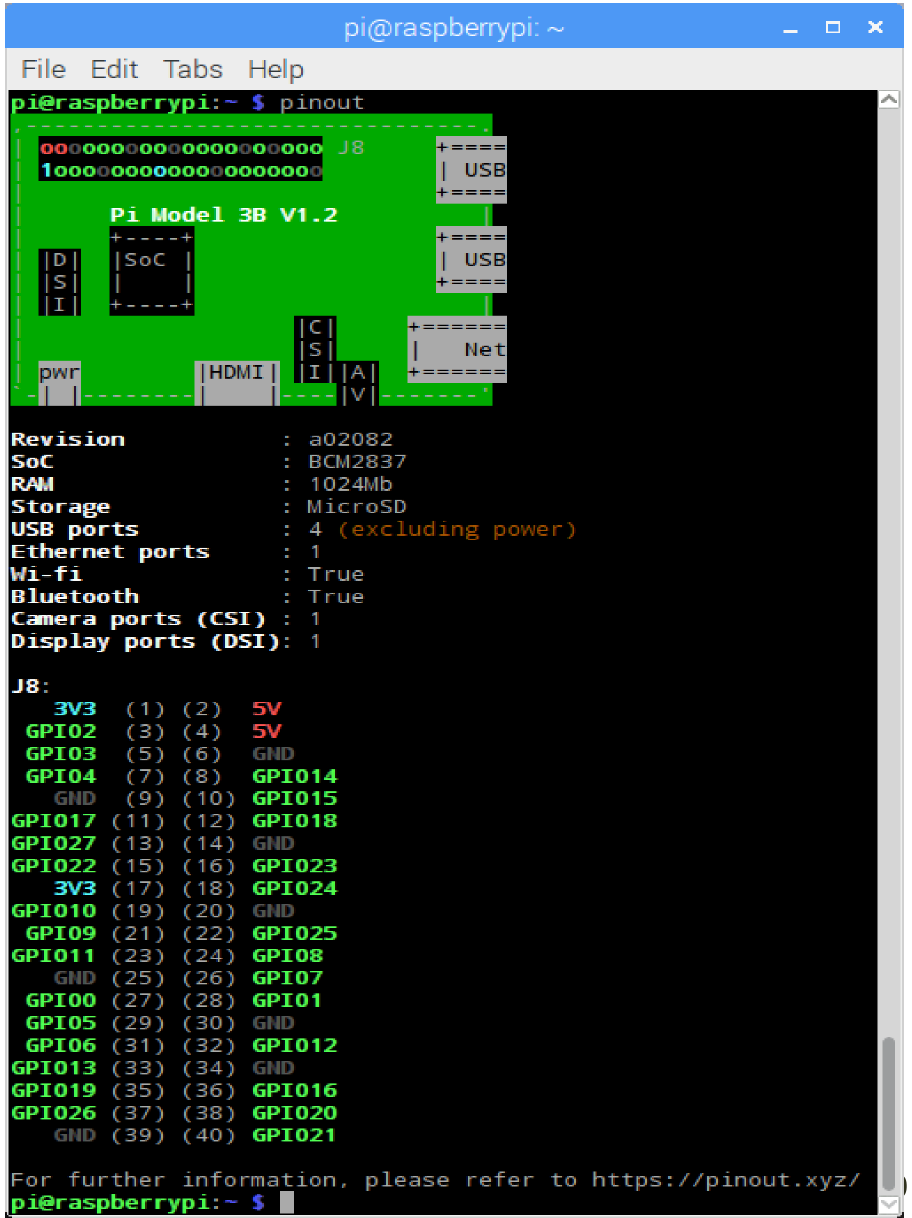
- As well as input and output devices, the GPIO pins can be used with a variety of alternative functions:
  - PWM (all pins)
  - I2C
  - SPI
  - Serial



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | DC Power | 3.3V | 01 | 02 | 5V | DC Power | |
| I2C | SDA | GPIO 02 | 03 | 04 | 5V | DC Power | |
| | SCL | GPIO 03 | 05 | 06 | Ground | | |
| | GCLK | GPIO 04 | 07 | 08 | GPIO 14 | TXD | UART |
| | | Ground | 09 | 10 | GPIO 15 | RXD | |
| | | GPIO 17 | 11 | 12 | GPIO 18 | PWM 0 | |
| | | GPIO 27 | 13 | 14 | Ground | | |
| | | GPIO 22 | 15 | 16 | GPIO 23 | | |
| | DC Power | 3.3V | 17 | 18 | GPIO 24 | | |
| SPI 0 | MOSI | GPIO 10 | 19 | 20 | Ground | | |
| | MISO | GPIO 09 | 21 | 22 | GPIO 25 | | |
| | SCLK | GPIO 11 | 23 | 24 | GPIO 08 | CE 0 | SPI 0 |
| | | Ground | 25 | 26 | GPIO 07 | CE 1 | |
| | | ID_SD | 27 | 28 | ID_SC | | |
| | | GPIO 05 | 29 | 30 | Ground | | |
| | | GPIO 06 | 31 | 32 | GPIO 12 | | |
| | PWM 1 | GPIO 13 | 33 | 34 | Ground | | |
| SPI 1 | MISO | GPIO 19 | 35 | 36 | GPIO 16 | | |
| | | GPIO 26 | 37 | 38 | GPIO 20 | MOSI | SPI 1 |
| | | Ground | 39 | 40 | GPIO 21 | SCLK | |

# GPIO pinout

- A pinout reference for your Raspberry Pi can be accessed by opening a terminal window and running the command `pinout`.
- This tool is provided by the GPIO Zero Python library, which is installed by default on the Raspberry Pi OS, but not on Raspberry Pi OS Lite.



```
pi@raspberrypi:~ $ pinout
,--------------------------------.
| oooooooooooooooooooo J8     +====
| 1ooooooooooooooooooo        | USB
|                             +====
|      Pi Model 3B V1.2          |
|      +----+                 +====
| |D|  |SoC |                 | USB
| |S|  |    |                 +====
| |I|  +----+                    |
|                   |C|     +======
|                   |S|     |   Net
| pwr        |HDMI| |I||A|  +======
`-| |--------|    |----|V|-------'

Revision           : a02082
SoC                : BCM2837
RAM                : 1024Mb
Storage            : MicroSD
USB ports          : 4 (excluding power)
Ethernet ports     : 1
Wi-fi              : True
Bluetooth          : True
Camera ports (CSI) : 1
Display ports (DSI): 1

J8:
   3V3  (1) (2)  5V
 GPIO2  (3) (4)  5V
 GPIO3  (5) (6)  GND
 GPIO4  (7) (8)  GPIO14
   GND  (9) (10) GPIO15
GPIO17 (11) (12) GPIO18
GPIO27 (13) (14) GND
GPIO22 (15) (16) GPIO23
   3V3 (17) (18) GPIO24
GPIO10 (19) (20) GND
 GPIO9 (21) (22) GPIO25
GPIO11 (23) (24) GPIO8
   GND (25) (26) GPIO7
 GPIO0 (27) (28) GPIO1
 GPIO5 (29) (30) GND
 GPIO6 (31) (32) GPIO12
GPIO13 (33) (34) GND
GPIO19 (35) (36) GPIO16
GPIO26 (37) (38) GPIO20
   GND (39) (40) GPIO21

For further information, please refer to https://pinout.xyz/
pi@raspberrypi:~ $ 
```

# GPIO Pins

## Permissions

- In order to use the GPIO ports your user must be a member of the gpio group. The pi user is a member by default, other users need to be added manually.

```
sudo usermod -a -G gpio <username>
```

## WARNING

- LEDs should have resistors to limit the current passing through them.
- Do not use 5V for 3.3V components.
- Do not connect motors directly to the GPIO pins, instead use an H-bridge circuit or a motor controller board.

# GPIO in Python

## GPIO Zero Library

- Controlling the GPIO ports requires many more lines of code, but this is already written for you and made easy to use with the GPIO Zero Library.
- The library is comprehensively documented at gpiozero.readthedocs.io.
- GPIO Zero library is installed by default on Raspberry Pi.

## Importing GPIO Zero

- In Python, libraries used in a script must be imported by name at the top of the file.
- For example, to use the Button from GPIO Zero:

  ```
  from gpiozero import Button
  ```

  Now Button is available directly in your script:

  ```
  button = Button(2)
  ```

- Alternatively, the whole GPIO Zero library can be imported:

  ```
  import gpiozero
  ```

  In this case, all references to items within GPIO Zero must be prefixed:

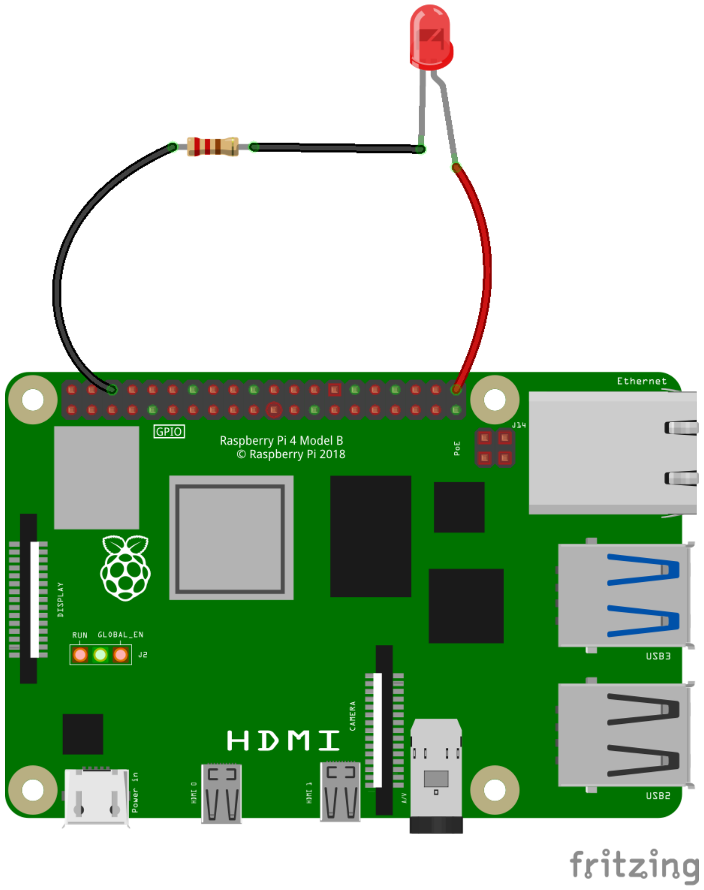
  ```
  button = gpiozero.Button(2)
  ```

# GPIO Zero Library

## LED

To control an LED connected to GPIO21, you can use this code:

```
from gpiozero import LED
from time import sleep

red = LED(21)

while True:
    red.on()
    sleep(1)
    red.off()
    sleep(1)
```



Run this in an IDE like Thonny, and the LED will blink on and off repeatedly.

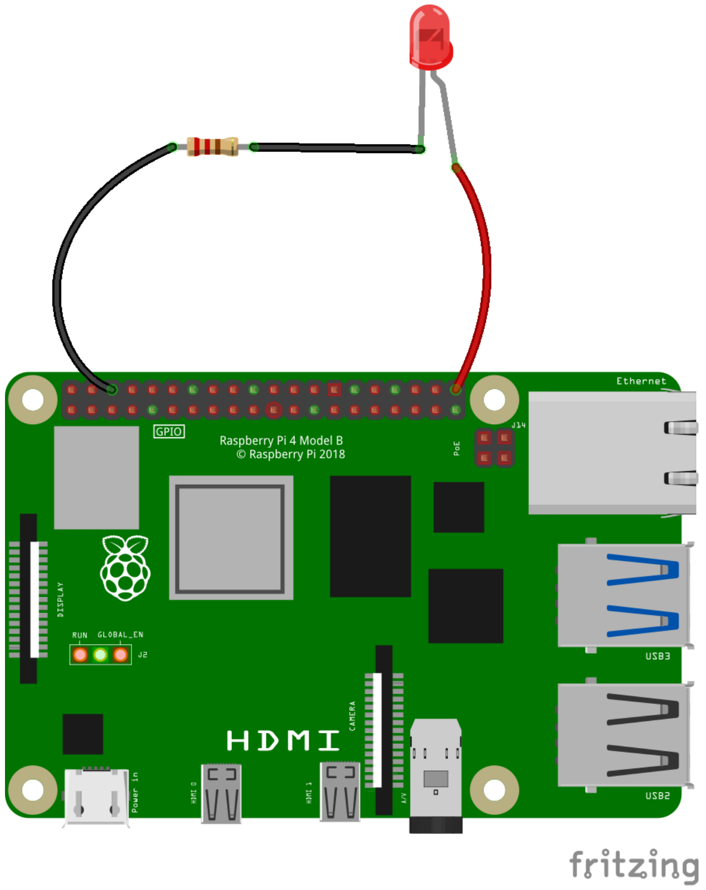
LED methods include on(), off(), toggle(), blink(), and value()

# GPIO Zero Library

## LED with variable brightness (PWM)

To change the brightness of an LED, PWMLED is used using values between 0 and 1:

```
from gpiozero import PWMLED
from time import sleep

led = PWMLED(21)

while True:
    led.value = 0  # off
    sleep(1)
    led.value = 0.5  # half brightness
    sleep(1)
    led.value = 1  # full brightness
    sleep(1)
```



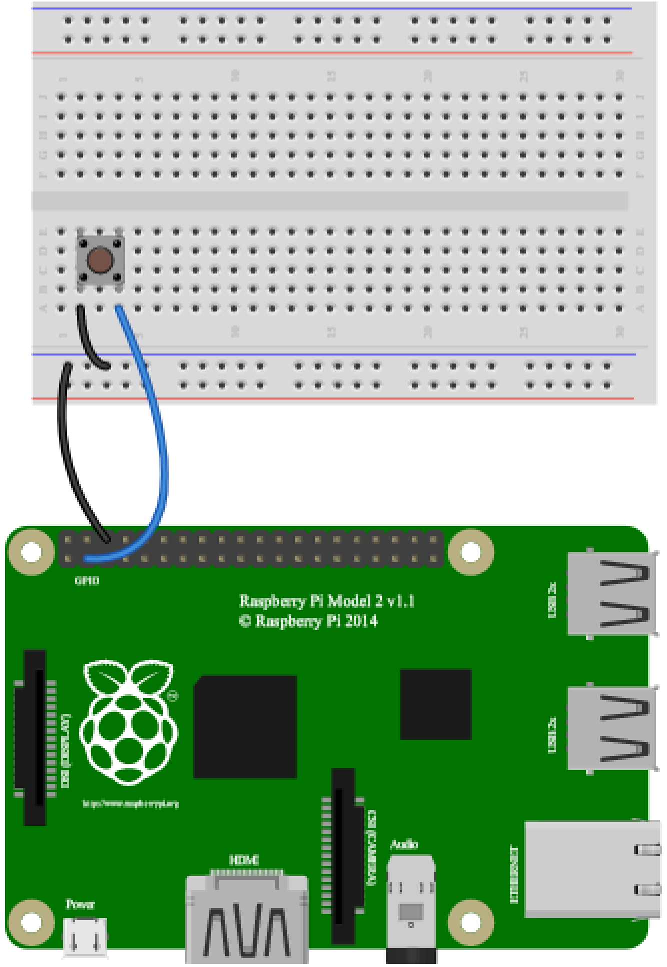
# Button

## Check if a Button is pressed:

```
from gpiozero import Button

button = Button(2)
while True:
    if button.is_pressed:
        print("Button is pressed")
    else:
        print("Button is not pressed")
```

## Run a function every time the button is pressed:

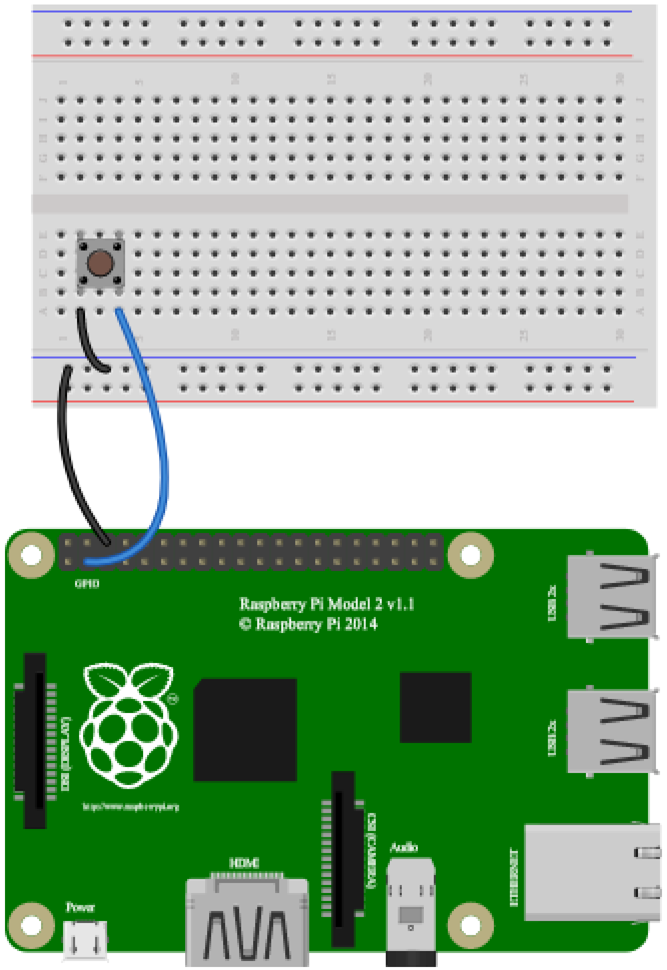
```
from gpiozero import Button
from signal import pause

def say_hello():
    print("Hello!")

button = Button(2)
button.when_pressed = say_hello
pause()
```



# Shutdown button

- The Button class also provides the ability to run a function when the button has been held for a given length of time.
- This example will shut down the Raspberry Pi when the button is held for 2 seconds:

```
from gpiozero import Button
from subprocess import check_call
from signal import pause

def shutdown():
    check_call(['sudo', 'poweroff'])

shutdown_btn = Button(2, hold_time=2)
shutdown_btn.when_held = shutdown

pause()
```



# Keyboard controlled LED

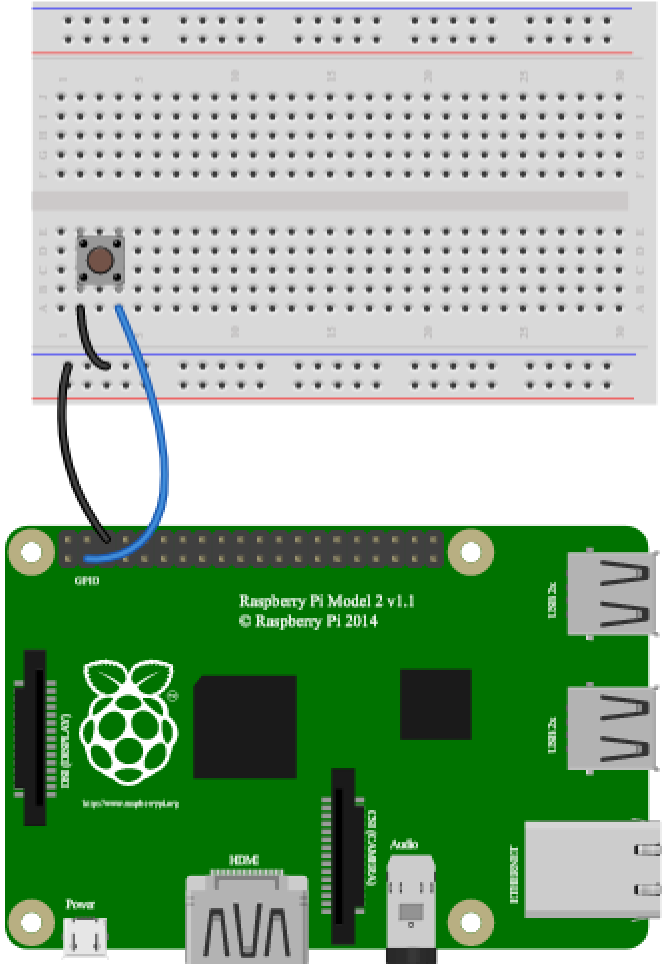
```
import curses
from gpiozero import LED

led = LED(17)
actions = {
    curses.KEY_UP:  led.on,
    curses.KEY_DOWN:  led.off
}

def main(window):
    next_key = None
    while True:
        if next_key is None:
            key = window.getch()
        else:
            key = next_key
            next_key = None
        if key != -1: # KEY PRESSED
            action = actions.get(key)
            if action is not None:
                action()
            next_key = key
            while next_key == key:
                next_key = window.getch()

curses.wrapper(main)
```

- You can control an LED using a keyboard
- Up_arrow: led on
- Down_arrow: led off

# Button controlled camera

- Using a button to take a picture from camera

```
from gpiozero import Button
from picamera import PiCamera
from datetime import datetime
from signal import pause

button = Button(2)
camera = PiCamera()

def capture():
    timestamp = datetime.now().isoformat()
    camera.capture('/home/pi/%s.jpg' % timestamp)

button.when_pressed = capture

pause()
```



# Button controlled camera

- Another example uses one button to start and stop the camera preview, and another to capture:

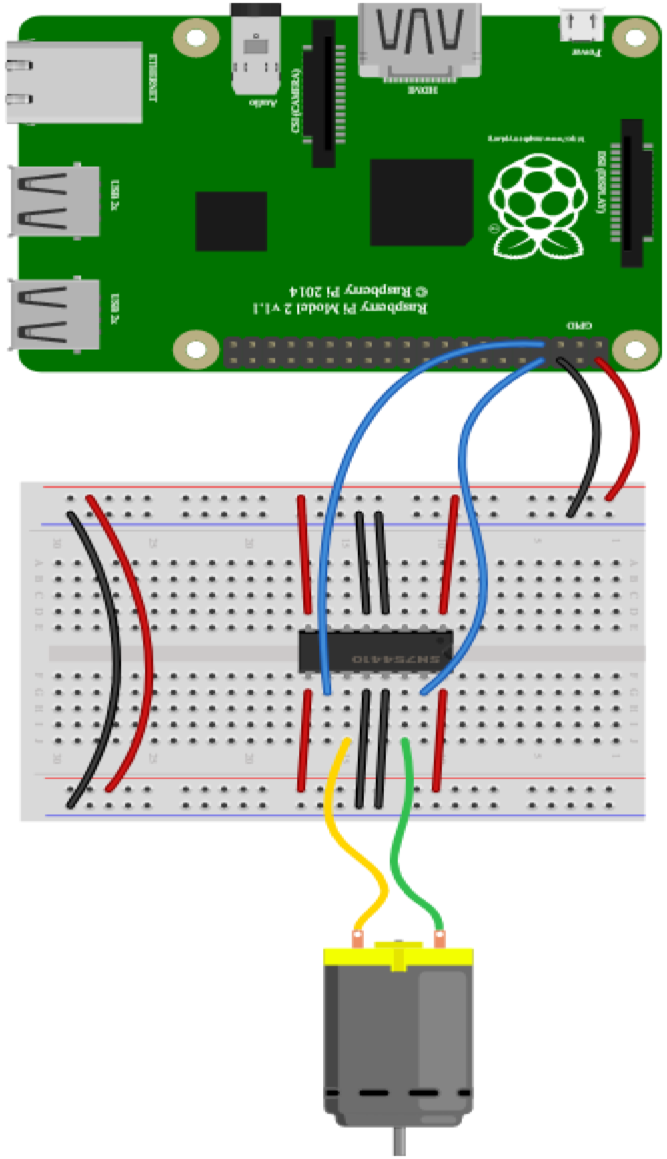
```
from gpiozero import Button
from picamera import PiCamera
from datetime import datetime
from signal import pause

left_button = Button(2)
right_button = Button(3)
camera = PiCamera()

def capture():
    timestamp = datetime.now().isoformat()
    camera.capture('/home/pi/%s.jpg' % timestamp)

left_button.when_pressed = camera.start_preview
left_button.when_released = camera.stop_preview
right_button.when_pressed = capture

pause()
```

- Note that the camera preview is not sent over VNC by default.
- To enable this option, go to VNC options on RP > troubleshooting > enable "experimental direct capture mode"

# Motors

- Turn a Motor forwards and backwards:

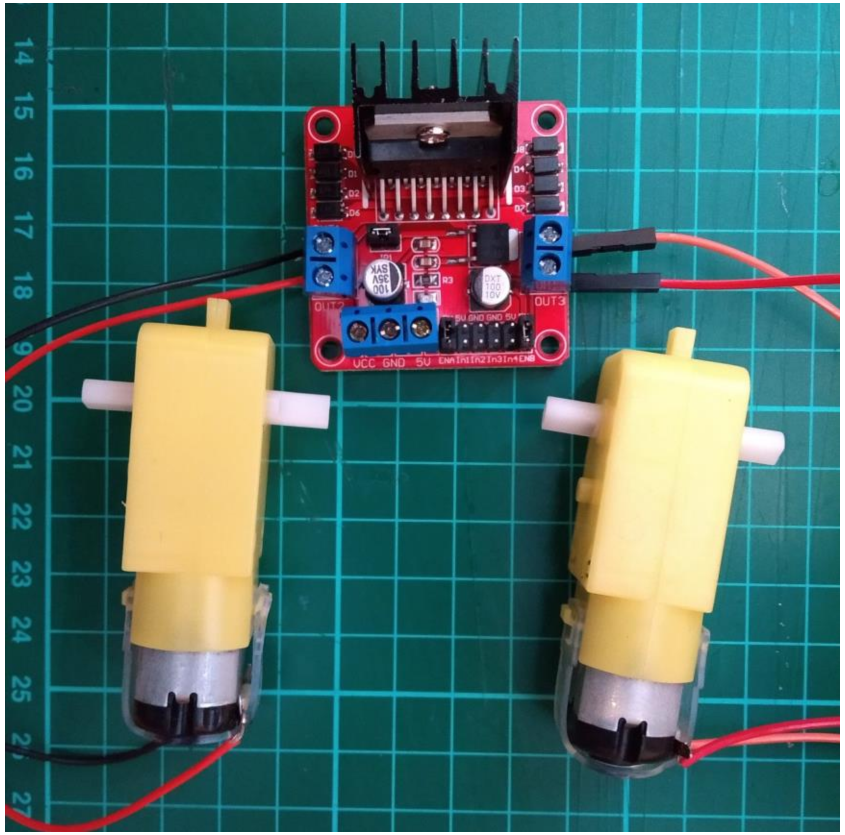
```
from gpiozero import Motor
from time import sleep

motor = Motor(forward=4, backward=14)

while True:
    motor.forward()
    sleep(5)
    motor.backward()
    sleep(5)
```



# Robot

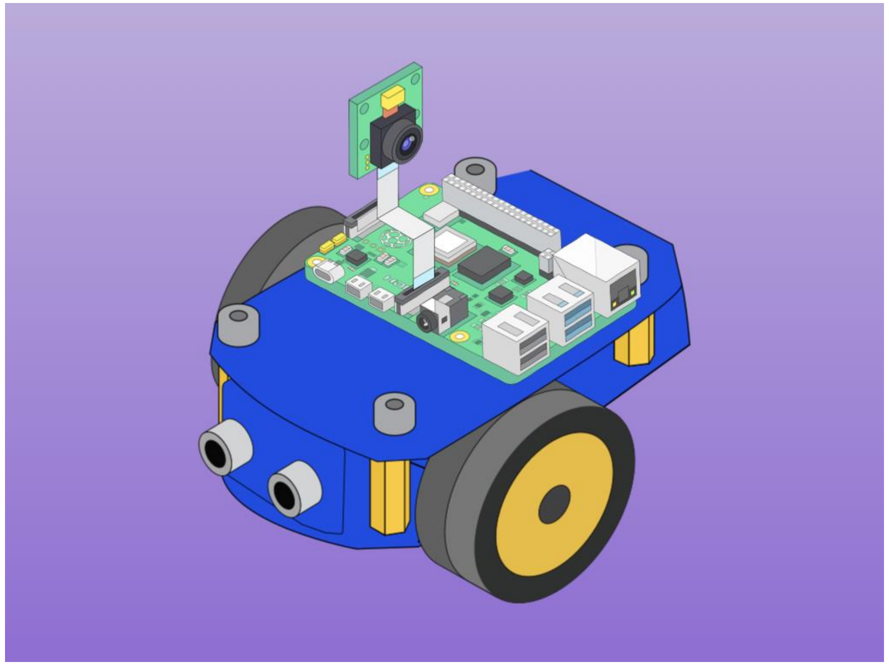
- Turn a Motor forwards and backwards:

```
from gpiozero import Robot
from time import sleep

robot = Robot(left=(4, 14), right=(17, 18))

for i in range(4):
    robot.forward()
    sleep(10)
    robot.right()
    sleep(1)
```



# Keyboard Controlled Robot

- Build a keyboard-controlled robot add a camera to it so you can see where it is heading, and use a Wi-Fi device to view it remotely through VNC!



# Run a Raspberry Pi Program on Startup

## Using rc.local file

- You will need root-level access to modify rc.local, so do so with sudo:

```
sudo nano /etc/rc.local
```

- Scroll down, and just before the exit 0 line, enter the following:

```
python /home/pi/program_name.py &
```

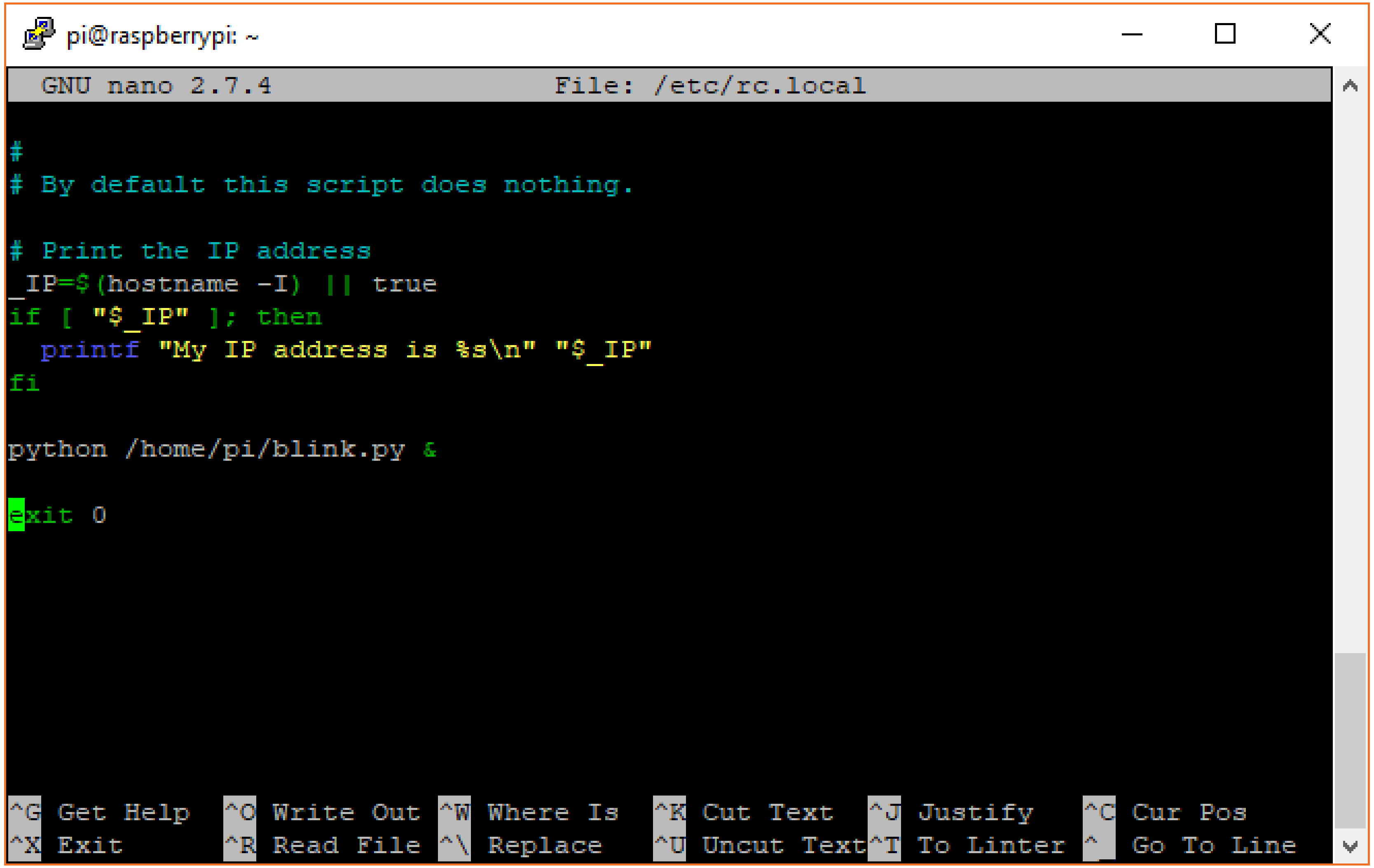
  where `program_name.py` is the program that you want to run at startup.

  Don’t forget ‘&’ at the end of the line.

- Save and exit with ctrl + x, followed by y when prompted to save, and then enter.
- Test it by restarting your Pi with `sudo reboot`.

# Run a Raspberry Pi Program on Startup

## The rc.local file will look as:



```
  GNU nano 2.7.4                File: /etc/rc.local

#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi

python /home/pi/blink.py &

exit 0
```

Any Questions