Raspberry Pi OS

Introduction

Edit this on GitHub

Raspberry Pi OS is a free operating system based on Debian, optimised for the Raspberry Pi hardware, and is the recommended operating system for normal use on a Raspberry Pi. The OS comes with over 35,000 packages: precompiled software bundled in a nice format for easy installation on your Raspberry Pi.

Raspberry Pi OS is under active development, with an emphasis on improving the stability and performance of as many Debian packages as possible on Raspberry Pi.

Updating and Upgrading Raspberry Pi OS

Edit this on GitHub

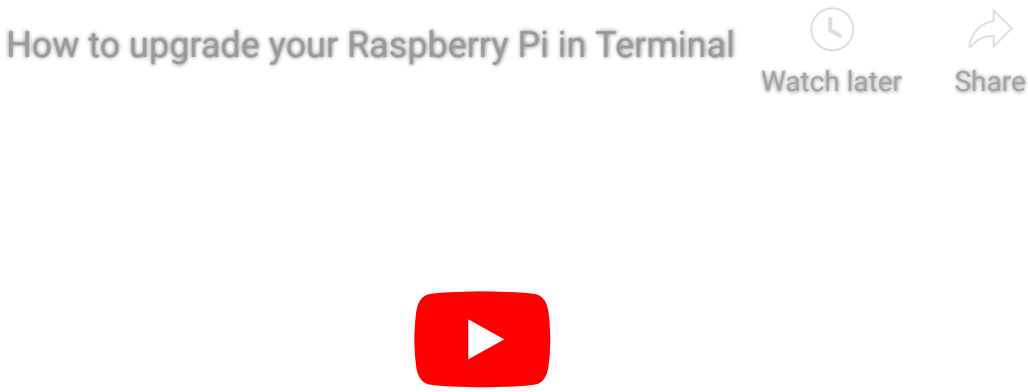
It's important to keep your Raspberry Pi up to date. The first and probably the most important reason is security. A device running Raspberry Pi OS contains millions of lines of code that you rely on. Over time, these millions of lines of code will expose well-known vulnerabilities, which are documented in publicly available databases meaning that they are easy to exploit. The only way to mitigate these exploits as a user of Raspberry Pi OS is to keep your software up to date, as the upstream repositories track CVEs closely and try to mitigate them quickly.

The second reason, related to the first, is that the software you are running on your device most certainly contains bugs. Some bugs are CVEs, but bugs could also be affecting the desired functionality without being related to security. By keeping your software up to date, you are lowering the chances of hitting these bugs.

Using APT

The easiest way to manage installing, upgrading, and removing software is using APT (Advanced Packaging Tool) from Debian. To update software in Raspberry Pi OS, you can use the `apt` tool from a Terminal window.

Keeping your Operating System up to Date



APT keeps a list of software sources on your Raspberry Pi in a file at `/etc/apt/sources.list`. Before installing software, you should update your package list with `apt update`. Go ahead and open a Terminal window and type:

```
sudo apt update
```

Next, **upgrade** all your installed packages to their latest versions with the following command:

```
sudo apt full-upgrade
```

Note that `full-upgrade` is used in preference to a simple `upgrade`, as it also picks up any dependency changes that may have been made.

Generally speaking, doing this regularly will keep your installation up to date for the particular major Raspberry Pi OS release you are using (e.g. Buster). It will not update from one major release to another, for example, Stretch to Buster or Buster to Bullseye.

However, there are occasional changes made in the Raspberry Pi OS image that require manual intervention, for example a newly introduced package. These are not installed with an upgrade, as this command only updates the packages you already have installed.

NOTE

The kernel and firmware are installed as a Debian package, and so will also get updates when using the procedure above. These packages are updated infrequently and after extensive testing.

If moving an existing SD card to a new Raspberry Pi model (for example the Raspberry Pi Zero 2 W), you may also need to update the kernel and the firmware first using the instructions above.

Running Out of Space

When running `sudo apt full-upgrade`, it will show how much data will be downloaded and how much space it will take up on the SD card. It's worth checking with `df -h` that you have enough free disk space, as unfortunately `apt` will not do this for you. Also be aware that downloaded package files (`.deb` files) are kept in `/var/cache/apt/archives`. You can remove these in order to free up space with `sudo apt clean` (`sudo apt-get clean` in older releases of apt).

Upgrading from Previous Operating System Versions

WARNING

Upgrading an existing image is possible, but is not guaranteed to work in every circumstance and we do not recommend it. If you do wish to try upgrading your operating system version, we strongly suggest making a backup first — we can accept no responsibility for loss of data from a failed update.

The latest version of Raspberry Pi OS is based on Debian Bullseye. The previous version was based on Buster. If you want to perform an in-place upgrade from Buster to Bullseye (and you're aware of the risks) see the instructions in the forums.

Searching for Software

You can search the archives for a package with a given keyword with `apt-cache search`:

```
apt-cache search locomotive
sl - Correct you if you type `sl' by mistake
```

You can view more information about a package before installing it with `apt-cache show`:

```
apt-cache show sl
Package: sl
Version: 3.03-17
Architecture: armhf
Maintainer: Hiroyuki Yamamoto <yama1066@gmail.com>
```

```
Installed-Size: 114
Depends: libc6 (>= 2.4), libncurses5 (>= 5.5-5~), libtinfo5
Homepage: http://www.tkl.iis.u-tokyo.ac.jp/~toyoda/index_e.html
Priority: optional
Section: games
Filename: pool/main/s/sl/sl_3.03-17_armhf.deb
Size: 26246
SHA256: 42dea9d7c618af8fe9f3c810b3d551102832bf217a5bcdba310f119f62117dfb
SHA1: b08039acccecd721fc3e6faf264fe59e56118e74
MD5sum: 450b21cc998dc9026313f72b4bd9807b
Description: Correct you if you type `sl' by mistake
 Sl is a program that can display animations aimed to correct you
 if you type 'sl' by mistake.
 SL stands for Steam Locomotive.
```

Installing a Package with APT

```
sudo apt install tree
```

Typing this command should inform the user how much disk space the package will take up and asks for confirmation of the package installation. Entering `Y` (or just pressing `Enter`, as yes is the default action) will allow the installation to occur. This can be bypassed by adding the `-y` flag to the command:

```
sudo apt install tree -y
```

Installing this package makes `tree` available for the user.

Uninstalling a Package with APT

You can uninstall a package with `apt remove`:

```
sudo apt remove tree
```

The user is prompted to confirm the removal. Again, the `-y` flag will auto-confirm.

You can also choose to completely remove the package and its associated configuration files with `apt purge`:

```
sudo apt purge tree
```

Using `rpi-update`

`rpi-update` is a command line application that will update your Raspberry Pi OS kernel and VideoCore firmware to the latest pre-release versions.

WARNING

> Pre-release versions of software are not guaranteed to work. You should not use `rpi-update` on any system unless recommended to do so by a Raspberry Pi engineer. It may leave your system unreliable or even completely broken. It should not be used as part of any regular update process.

The `rpi-update` script was originally written by Hexxeh, but is now supported by Raspberry Pi engineers. The script source is in the rpi-update repository.

What it does

`rpi-update` will download the latest pre-release version of the linux kernel, its matching modules, device tree files, along with the latest versions of the VideoCore firmware. It will then install these files to relevant locations on the SD card, overwriting any previous versions.

All the source data used by `rpi-update` comes from the rpi-firmware repository. This repository simply contains a subset of the data from the official firmware repository, as not all the data from that repo is required.

Running `rpi-update`

If you are sure that you need to use `rpi-update`, it is advisable to take a backup of your current system first as running `rpi-update` could result in a non-booting system.

`rpi-update` needs to be run as root. Once the update is complete you will need to reboot.

```
sudo rpi-update
sudo reboot
```

It has a number of options documented in the rpi-update repository.

How to get back to safety

If you have done an `rpi-update` and things are not working as you wish, if your Raspberry Pi is still bootable you can return to the stable release using:

```
sudo apt-get update
sudo apt install --reinstall libraspberrypi0 libraspberrypi-{bin,dev,doc} raspb
errypi-bootloader raspberrypi-kernel
```

You will need to reboot your Raspberry Pi for these changes to take effect.

Playing Audio and Video

Edit this on GitHub

WARNING

> The following documentation refers to Raspberry Pi OS Buster and earlier versions. OMXPlayer has been deprecated in the latest OS release. If you are running Bullseye, VLC is now the recommended alternative.

The simplest way of playing audio and video on Raspberry Pi is to use the installed OMXPlayer application.

This is hardware accelerated, and can play back many popular audio and video file formats. OMXPlayer uses the OpenMAX (`omx`) hardware acceleration interface (API) which is the officially supported media API on Raspberry Pi. OMXPlayer was developed by the Kodi Project's Edgar Hucek.

The OMXPlayer Application

The simplest command line is `omxplayer <name of media file>`. The media file can be audio or video or both. For the examples below, we used an H264 video file that is included with the standard Raspberry Pi OS installation.

```
omxplayer /opt/vc/src/hello_pi/hello_video/test.h264
```

By default the audio is sent to the analog port. If you are using a HDMI-equipped display device with speakers, you need to tell omxplayer to send the audio signal over the HDMI link.

```
omxplayer --adev hdmi /opt/vc/src/hello_pi/hello_video/test.h264
```

When displaying video, the whole display will be used as output. You can specify which part of the display you want the video to be on using the window option.

```
omxplayer --win 0,0,640,480 /opt/vc/src/hello_pi/hello_video/test.h264
```

You can also specify which part of the video you want to be displayed: this is called a crop window. This portion of the video will be scaled up to match the display, unless you also use the window option.

```
omxplayer --crop 100,100,300,300 /opt/vc/src/hello_pi/hello_video/test.h264
```

If you are using the Raspberry Pi Touch Display, and you want to use it for video output, use the display option to specify which display to use. `n` is 5 for HDMI, 4 for the touchscreen. With the Raspberry Pi 4 you have two options for HDMI output. `n` is 2 for HDMI0 and 7 for HDMI1.

```
omxplayer --display n /opt/vc/src/hello_pi/hello_video/test.h264
```

How to Play Audio

To play an MP3 file, navigate to the location of the `.mp3` file in the terminal using `cd` and then type the following command:

```
omxplayer example.mp3
```

This will play the audio file `example.mp3` through either your monitor's built-in speakers or your headphones, connected via the headphone jack.

If you need an example file you can download one from here using the following command:

```
wget https://raw.githubusercontent.com/raspberrypilearning/burping-jelly-baby/m
aster/data/1a.mp3 -O example.mp3 --no-check-certificate
```

If you cannot hear anything, make sure your headphones or speakers are connected correctly. Note that omxplayer doesn't use ALSA and so ignores the audio configuration set by `raspi-config` or `amixer`.

If omxplayer's auto-detection of the correct audio output device fails, you can force output over HDMI with:

```
omxplayer -o hdmi example.mp3
```

Alternatively, you can force output over the headphone jack with:

```
omxplayer -o local example.mp3
```

You can even force output over both the headphone jack and HDMI with:

```
omxplayer -o both example.mp3
```

How to Play Video

To play a video, navigate to the location of your video file in the terminal using `cd`, then type the following command:

```
omxplayer example.mp4
```

This will play the `example.mp4` in full screen. Hit `Ctrl + C` to exit.

On the Raspberry Pi 4, hardware support for MPEG2 and VC-1 codecs has been removed, so we recommend the use of the VLC application, which supports these formats in software. In addition, VLC has hardware support for H264 and the new HEVC codec.

An Example Video

A video sample of the animated film *Big Buck Bunny* is available on your Raspberry Pi. To play it enter the following command into a terminal window:

```
omxplayer /opt/vc/src/hello_pi/hello_video/test.h264
```

On a Raspberry Pi 4, use the following command for H264 files:

```
omxplayer /opt/vc/src/hello_pi/hello_video/test.h264
```

or for H264, VC1, or MPEG2

```
vlc /opt/vc/src/hello_pi/hello_video/test.h264
```

When using VLC, you can improve playback performance by encapsulating the raw H264 stream, for example from the Raspberry Pi Camera Module. This is easily done using `ffmpeg`. Playback is also improved if VLC is run full screen; either select fullscreen from the user interface, or you can add the `--fullscreen` options to the `vlc` command line.

This example command converts `video.h264` to a containerised `video.mp4` at 30 fps:

```
ffmpeg -r 30 -i video.h264 -c:v copy video.mp4
```

Options During Playback

There are a number of options available during playback, actioned by pressing the appropriate key. Not all options will be available on all files. The list of key bindings can be displayed using `omxplayer --keys`:

```
1           decrease speed
2           increase speed
<           rewind
>           fast forward
z           show info
j           previous audio stream
k           next audio stream
i           previous chapter
o           next chapter
n           previous subtitle stream
m           next subtitle stream
s           toggle subtitles
w           show subtitles
x           hide subtitles
d           decrease subtitle delay (- 250 ms)
f           increase subtitle delay (+ 250 ms)
q           exit omxplayer
p / space   pause/resume
-           decrease volume
+ / =       increase volume
left arrow  seek -30 seconds
right arrow seek +30 seconds
down arrow  seek -600 seconds
up arrow    seek +600 seconds
```

Playing in the Background

`omxplayer` will close immediately if run in the background without tty (user input), so to run successfully, you need to tell `omxplayer` not to require any user input using the `--no-keys` option.

```
omxplayer --no-keys example.mp3 &
```

Adding the `&` at the end of the command runs the job in the background. You can then check the status of this background job using the `jobs` command. By default, the job will complete when `omxplayer` finishes playing, but if necessary, you can stop it at any point using the `kill` command.

```
$ jobs
[1]-  Running                 omxplayer --no-keys example.mp3 &
$ kill %1
$
[1]-  Terminated              omxplayer --no-keys example.mp3 &
```

Using a USB webcam

Edit this on GitHub

Rather than using the Raspberry Pi camera module, you can use a standard USB webcam to take pictures and video on your Raspberry Pi.

NOTE

> The quality and configurability of the camera module is highly superior to a standard USB webcam.

First, install the `fswebcam` package:

```
sudo apt install fswebcam
```

If you are not using the default `pi` user account, you need to add your username to the `video` group, otherwise you will see 'permission denied' errors.

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

To check that the user has been added to the group correctly, use the `groups` command.

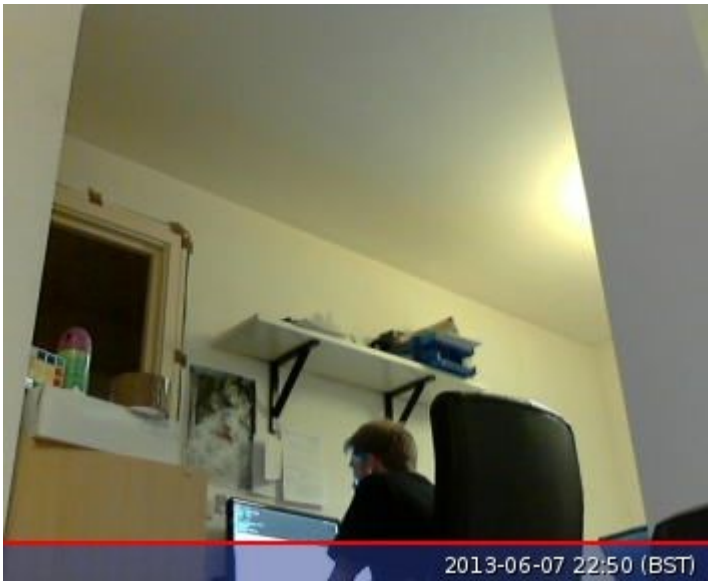
Basic Usage

Enter the command `fswebcam` followed by a filename and a picture will be taken using the webcam, and saved to the filename specified:

```
fswebcam image.jpg
```

This command will show the following information:

```
--- Opening /dev/video0...
Trying source module v4l2...
/dev/video0 opened.
No input was specified, using the first.
Adjusting resolution from 384x288 to 352x288.
--- Capturing frame...
Corrupt JPEG data: 2 extraneous bytes before marker 0xd4
Captured frame in 0.00 seconds.
--- Processing captured image...
Writing JPEG image to 'image.jpg'.
```



NOTE

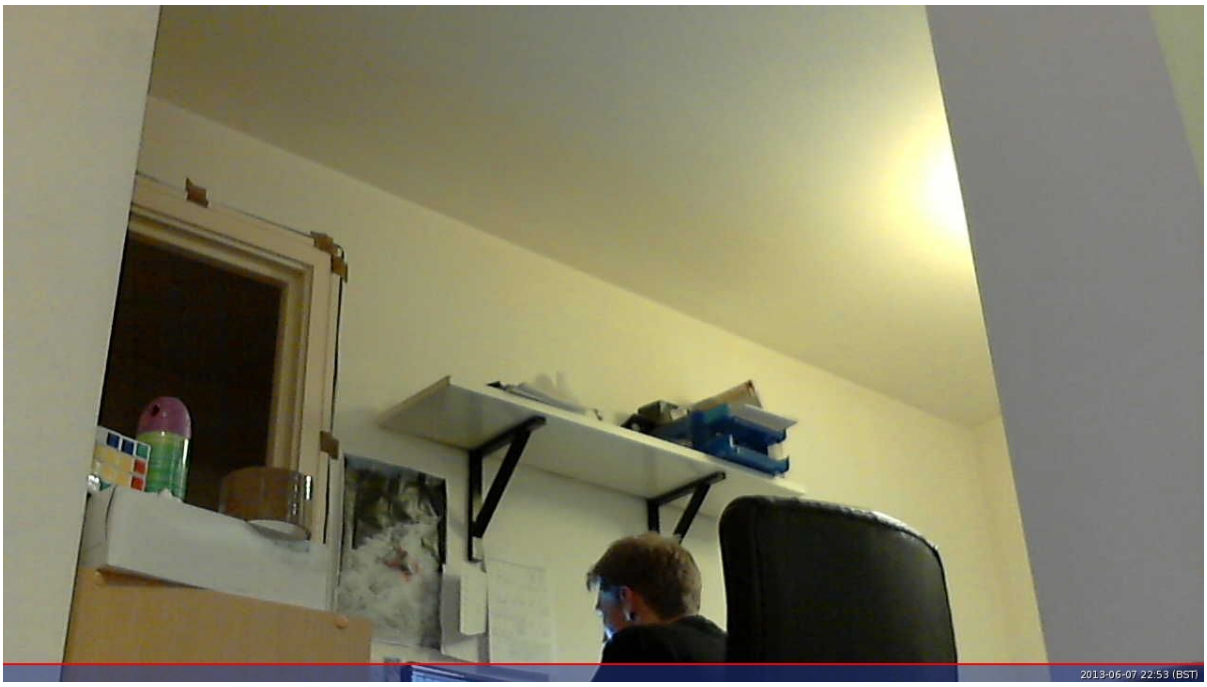
The small default resolution used, and the presence of a banner showing the timestamp.

The webcam used in this example has a resolution of `1280 x 720` so to specify the resolution I want the image to be taken at, use the `-r` flag:

```
fswebcam -r 1280x720 image2.jpg
```

This command will show the following information:

```
--- Opening /dev/video0...
Trying source module v4l2...
/dev/video0 opened.
No input was specified, using the first.
--- Capturing frame...
Corrupt JPEG data: 1 extraneous bytes before marker 0xd5
Captured frame in 0.00 seconds.
--- Processing captured image...
Writing JPEG image to 'image2.jpg'.
```



Picture now taken at the full resolution of the webcam, with the banner present.

Removing the Banner

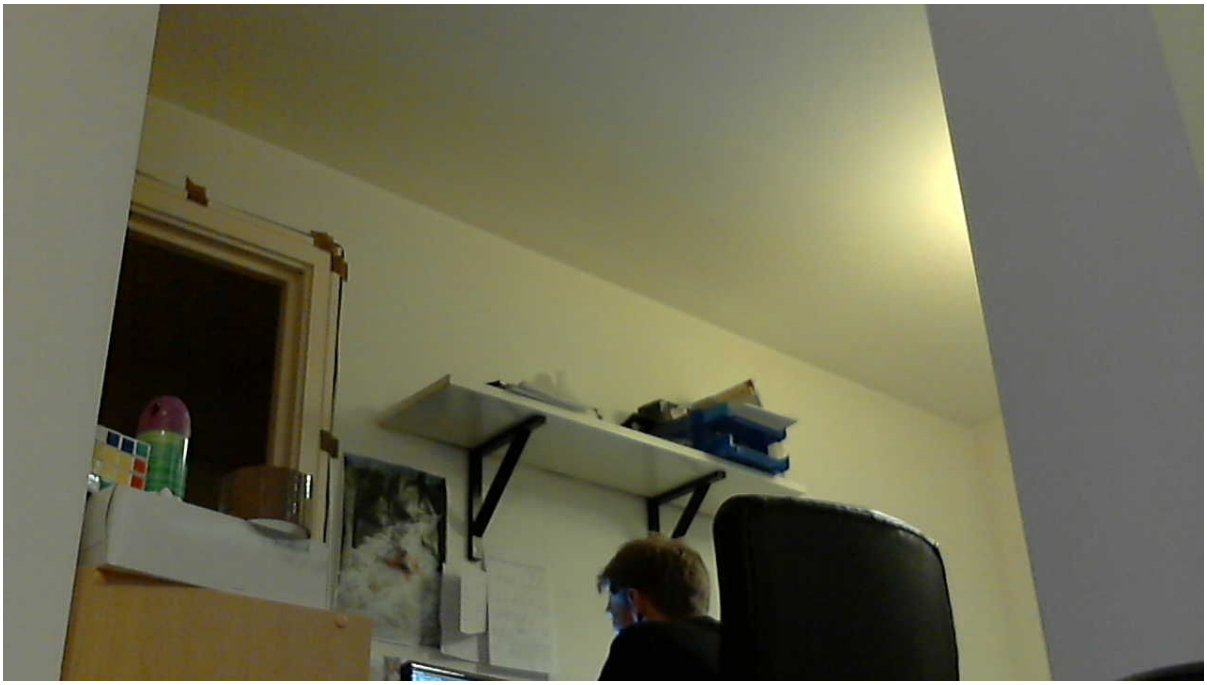
Now add the `--no-banner` flag:

```
fswebcam -r 1280x720 --no-banner image3.jpg
```

which shows the following information:

```
--- Opening /dev/video0...
Trying source module v4l2...
/dev/video0 opened.
No input was specified, using the first.
--- Capturing frame...
Corrupt JPEG data: 2 extraneous bytes before marker 0xd6
Captured frame in 0.00 seconds.
--- Processing captured image...
```

```
Disabling banner.
Writing JPEG image to 'image3.jpg'.
```



Now the picture is taken at full resolution with no banner.

Automating Image Capture

You can write a Bash script which takes a picture with the webcam. The script below saves the images in the `/home/pi/webcam` directory, so create the `webcam` subdirectory first with:

```
mkdir webcam
```

To create a script, open up your editor of choice and write the following example code:

```
#!/bin/bash

DATE=$(date +"%Y-%m-%d_%H%M")

fswebcam -r 1280x720 --no-banner /home/pi/webcam/$DATE.jpg
```

This script will take a picture and name the file with a timestamp. Say we saved it as `webcam.sh`, we would first make the file executable:

```
chmod +x webcam.sh
```

Then run with:

```
./webcam.sh
```

Which would run the commands in the file and give the usual output:

```
--- Opening /dev/video0...
Trying source module v4l2...
/dev/video0 opened.
No input was specified, using the first.
--- Capturing frame...
Corrupt JPEG data: 2 extraneous bytes before marker 0xd6
Captured frame in 0.00 seconds.
--- Processing captured image...
Disabling banner.
Writing JPEG image to '/home/pi/webcam/2013-06-07_2338.jpg'.
```

Time-Lapse Captures

You can use `cron` to schedule taking a picture at a given interval, such as every minute to capture a time-lapse.

First open the cron table for editing:

```
crontab -e
```

This will either ask which editor you would like to use, or open in your default editor. Once you have the file open in an editor, add the following line to schedule taking a picture every minute (referring to the Bash script from above):

```
* * * * * /home/pi/webcam.sh 2>&1
```

Save and exit and you should see the message:

```
crontab: installing new crontab
```

Ensure your script does not save each picture taken with the same filename. This will overwrite the picture each time.

Useful Utilities

Edit this on GitHub

There are several useful command line

tvservice

`tvservice` is a command line application used to get and set information about the display, targeted mainly at HDMI video and audio.

Typing `tvservice` by itself will display a list of available command line options.

-p, --preferred

Power on the HDMI output with preferred settings.

-o, --off

Powers off the display output.

NOTE

> Powering off the output using this command will also destroy any framebuffers/dispmanx layers associated with the display. These are NOT re-established with a subsequent power on, so will result in a blank screen.

A better option is to use the vcgencmd display_power option, as this will retain any framebuffers, so when the power is turned back on the display will be the returned to the previous power on state.

-e, --explicit="Group Mode Drive"

Power on the HDMI with the specified settings

Group can be one of `CEA`, `DMT`, `CEA_3D_SBS`, `CEA_3D_TB`, `CEA_3D_FP`, `CEA_3D_FS`.
Mode is one of the modes returned from the `-m, --modes` option.
Drive can be one of `HDMI`, `DVI`.

-t, --ntsc

Use 59.94Hz (NTSC frequency) rather than 60Hz for HDMI mode.

-c, --sdtvon="Mode Aspect [P]"

Power on the SDTV (composite output) with the specified mode, `PAL` or `NTSC`, and the specified aspect, `4:3`, `14:9`, `16:9`. The optional `P` parameter can be used to specify progressive mode.

-m, --modes=Group

where Group is `CEA` or `DMT`.

Shows a list of display modes available in the specified group.

-M, --monitor

Monitors for any HDMI events, for example unplugging or attaching.

-s, --status

Shows the current settings for the display mode, including mode, resolution, and frequency.

-a, --audio

Shows the current settings for the audio mode, including channels, sample rate and sample size.

-d, --dumpid=filename

Save the current EDID to the specified filename. You can then use `edidparser <filename>` to display the data in a human readable form.

-j, --json

When used in combination with the `--modes` options, displays the mode information in JSON format.

-n, --name

Extracts the display name from the EDID data and shows it.

-l, --list

Lists all attached displays and their display ID.

-v, --device=display

Specifies the ID of the device to use; see the output of `--list` for available IDs.

vcgencmd

The `vcgencmd` tool is used to output information from the VideoCore GPU on the Raspberry Pi. You can find source code for the `vcgencmd` utility on Github.

To get a list of all commands which `vcgencmd` supports, use `vcgencmd commands`. Some useful commands and their required parameters are listed below.

vcos

The `vcos` command has two useful sub-commands:

- `version` displays the build date and version of the firmware on the VideoCore
- `log status` displays the error log status of the various VideoCore firmware areas

version

Displays the build date and version of the VideoCore firmware.

get_camera

Displays the enabled and detected state of the Raspberry Pi camera: `1` means yes, `0` means no. Whilst all firmware except cutdown versions support the camera, this support needs to be enabled by using raspi-config.

get_throttled

Returns the throttled state of the system. This is a bit pattern - a bit being set indicates the following meanings:

Bit	Hex value	Meaning
0	0x1	Under-voltage detected
1	0x2	Arm frequency capped

Bit	Hex value	Meaning
2	0x4	Currently throttled
3	0x8	Soft temperature limit active
16	0x10000	Under-voltage has occurred
17	0x20000	Arm frequency capping has occurred
18	0x40000	Throttling has occurred
19	0x80000	Soft temperature limit has occurred

measure_temp

Returns the temperature of the SoC as measured by its internal temperature sensor; on Raspberry Pi 4, `measure_temp pmic` returns the temperature of the PMIC.

measure_clock [clock]

This returns the current frequency of the specified clock. The options are:

clock	Description
arm	ARM core(s)
core	GPU core
h264	H.264 block
isp	Image Sensor Pipeline
v3d	3D block
uart	UART
pwm	PWM block (analogue audio output)
emmc	SD card interface
pixel	Pixel valves
vec	Analogue video encoder
hdmi	HDMI
dpi	Display Parallel Interface

e.g. `vcgencmd measure_clock arm`

measure_volts [block]

Displays the current voltages used by the specific block.

block	Description
core	VC4 core voltage
sdram_c	SDRAM Core Voltage
sdram_i	SDRAM I/O voltage
sdram_p	SDRAM Phy Voltage

otp_dump

Displays the content of the OTP (one-time programmable) memory inside the SoC. These are 32 bit values, indexed from 8 to 64. See the OTP bits page for more details.

get_config [configuration item|int|str]

Display value of the configuration setting specified: alternatively, specify either `int` (integer) or `str` (string) to see all configuration items of the given type. For example:

```
vcgencmd get_config total_mem
```

returns the total memory on the device in megabytes.

get_mem type

Reports on the amount of memory addressable by the ARM and the GPU. To show the amount of ARM-addressable memory use `vcgencmd get_mem arm`; to show the amount of GPU-addressable memory use `vcgencmd get_mem gpu`. Note that on devices with more than 1GB of memory the `arm` parameter will always return 1GB minus the `gpu` memory value, since the GPU firmware is only aware of the first 1GB of memory. To get an accurate report of the total memory on the device, see the `total_mem` configuration item - see `get_config` section above.

codec_enabled [type]

Reports whether the specified CODEC type is enabled. Possible options for type are AGIF, FLAC, H263, H264, MJPA, MJPB, MJPG, **MPG2**, MPG4, MVC0, PCM, THRA, VORB, VP6, VP8, **WMV9**, **WVC1**. Those highlighted currently require a paid for licence (see the this config.txt section for more info), except on the Raspberry Pi 4 and 400, where these

hardware codecs are disabled in preference to software decoding, which requires no licence. Note that because the H.265 HW block on the Raspberry Pi 4 and 400 is not part of the VideoCore GPU, its status is not accessed via this command.

get_lcd_info

Displays the resolution and colour depth of any attached display.

mem_oom

Displays statistics on any OOM (out of memory) events occurring in the VideoCore memory space.

mem_reloc_stats

Displays statistics from the relocatable memory allocator on the VideoCore.

read_ring_osc

Returns the current speed voltage and temperature of the ring oscillator.

hdmi_timings

Displays the current HDMI settings timings. See Video Config for details of the values returned.

dispmanx_list

Dump a list of all dispmanx items currently being displayed.

display_power [0 | 1 | -1] [display]

Show current display power state, or set the display power state. `vcgencmd display_power 0` will turn off power to the current display. `vcgencmd display_power 1` will turn on power to the display. If no parameter is set, this will display the current power state. The final parameter is an optional display ID, as returned by `tvservice -l` or from the table below, which allows a specific display to be turned on or off.

Note that for the 7" Raspberry Pi Touch Display this simply turns the backlight on and off. The touch functionality continues to operate as normal.

`vcgencmd display_power 0 7` will turn off power to display ID 7, which is HDMI 1 on a Raspberry Pi 4.

Display	ID

Display	ID
Main LCD	0
Secondary LCD	1
HDMI 0	2
Composite	3
HDMI 1	7

To determine if a specific display ID is on or off, use -1 as the first parameter.

`vcgencmd display_power -1 7` will return 0 if display ID 7 is off, 1 if display ID 7 is on, or -1 if display ID 7 is in an unknown state, for example undetected.

vcdbg

`vcdbg` is an application to help with debugging the VideoCore GPU from Linux running on the ARM. It needs to be run as root. This application is mostly of use to Raspberry Pi engineers, although there are some commands that general users may find useful.

`sudo vcdbg help` will give a list of available commands.

NOTE

> Only options of use to end users have been listed.

version

Shows various items of version information from the VideoCore.

log

Dumps logs from the specified subsystem. Possible options are:

log	Description
msg	Prints out the message log
assert	Prints out the assertion log
ex	Prints out the exception log
info	Prints out information from the logging headers

log	Description
level	Sets the VCOS logging level for the specified category, n|e|w|i|t
list	List the VCOS logging levels

e.g. To print out the current contents of the message log:

```
vcdbg log msg
```

malloc

List all memory allocations current in the VideoCore heap.

pools

List the current status of the pool allocator

reloc

Without any further parameters, lists the current status of the relocatable allocator. Use `sudo vcdbg reloc small` to list small allocations as well.

Use the subcommand `sudo vcdbg reloc stats` to list statistics for the relocatable allocator.

hist

Commands related to task history.

Use `sudo vcdbg hist gnuplot` to dump task history in gnuplot format to task.gpt and task.dat

Python

Edit this on GitHub

Python is a powerful programming language that's easy to use easy to read and write and, with Raspberry Pi, lets you connect your project to the real world. Python syntax is clean, with an emphasis on readability, and uses standard English keywords.

Thonny

The easiest introduction to Python is through Thonny, a Python 3 development environment. You can open Thonny from the desktop or applications menu.

Thonny gives you a REPL (Read-Evaluate-Print-Loop), which is a prompt you can enter Python commands into. Because it's a REPL, you even get the output of commands printed to the screen without using `print`. In the Thonny application, this is called the Shell window.

You can use variables if you need to but you can even use it like a calculator. For example:

```
>>> 1 + 2
3
>>> name = "Sarah"
>>> "Hello " + name
'Hello Sarah'
```

Thonny also has syntax highlighting built in and some support for autocompletion. You can look back on the history of the commands you've entered in the REPL with `Alt + P` (previous) and `Alt + N` (next).

Basic Python usage

Hello world in Python:

```
print("Hello world")
```

Simple as that!

Indentation

Some languages use curly braces `{` and `}` to wrap around lines of code which belong together, and leave it to the writer to indent these lines to appear visually nested. However, Python does not use curly braces but instead requires indentation for nesting. For example a `for` loop in Python:

```
for i in range(10):
    print("Hello")
```

The indentation is necessary here. A second line indented would be a part of the loop, and a second line not indented would be outside of the loop. For example:

```
for i in range(2):
    print("A")
    print("B")
```

would print:

```
A
B
A
B
```

whereas the following:

```
for i in range(2):
    print("A")
print("B")
```

would print:

```
A
A
B
```

Variables

To save a value to a variable, assign it like so:

```
name = "Bob"
age = 15
```

Note that data types were not specified with these variables, as types are inferred, and can be changed later.

```
age = 15
age += 1  # increment age by 1
print(age)
```

This time I used comments beside the increment command.

Comments

Comments are ignored in the program but there for you to leave notes, and are denoted by the hash # symbol. Multi-line comments use triple quotes like so:

```
"""
This is a very simple Python program that prints "Hello".
That's all it does.
"""

print("Hello")
```

Lists

Python also has lists (called arrays in some languages) which are collections of data of any type:

```
numbers = [1, 2, 3]
```

Lists are denoted by the use of square brackets `[]` and each item is separated by a comma.

Iteration

Some data types are iterable, which means you can loop over the values they contain. For example a list:

```
numbers = [1, 2, 3]

for number in numbers:
    print(number)
```

This takes each item in the list `numbers` and prints out the item:

```
1
2
3
```

Note I used the word `number` to denote each item. This is merely the word I chose for this - it's recommended you choose descriptive words for variables - using plurals for lists, and singular for each item makes sense. It makes it easier to understand when reading.

Other data types are iterable, for example the string:

```
dog_name = "BINGO"

for char in dog_name:
    print(char)
```

This loops over each character and prints them out:

```
B
I
N
G
O
```

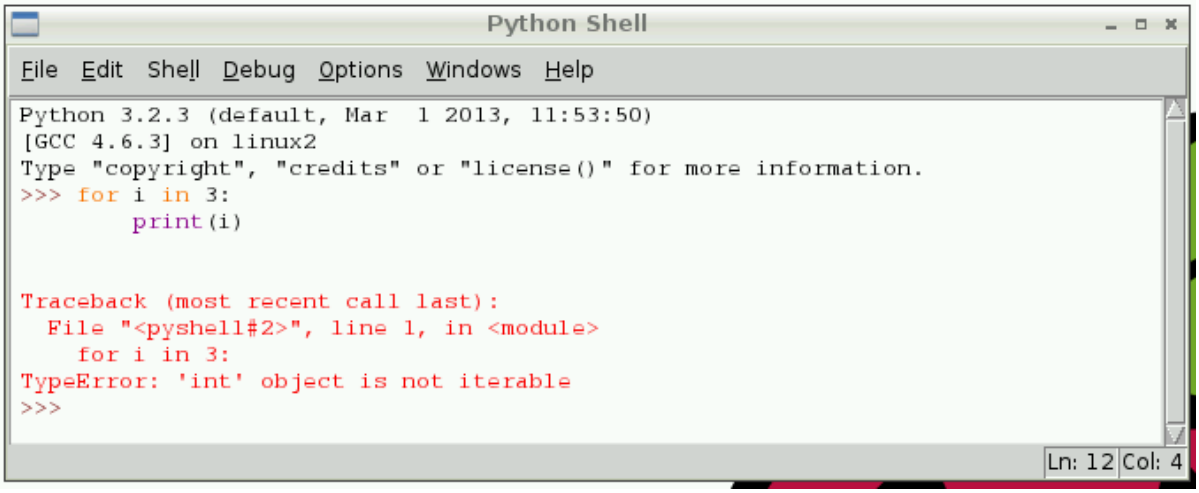
Range

The integer data type is not iterable and trying to iterate over it will produce an error. For example:

```
for i in 3:
    print(i)
```

will produce:

```
TypeError: 'int' object is not iterable
```



However you can make an iterable object using the `range` function:

```
for i in range(3):
    print(i)
```

`range(5)` contains the numbers `0`, `1`, `2`, `3` and `4` (five numbers in total). To get the numbers `1` to `5` (inclusive) use `range(1, 6)`.

Length

You can use functions like `len` to find the length of a string or a list:

```
name = "Jamie"
print(len(name))  # 5

names = ["Bob", "Jane", "James", "Alice"]
print(len(names))  # 4
```

If statements

You can use `if` statements for control flow:

```
name = "Joe"

if len(name) > 3:
    print("Nice name,")
    print(name)
else:
    print("That's a short name,")
    print(name)
```

Python files in Thonny

To create a Python file in Thonny, click `File > New` and you'll be given a window. This is an empty file, not a Python prompt. You write a Python file in this window, save it, then run it and you'll see the output in the other window.

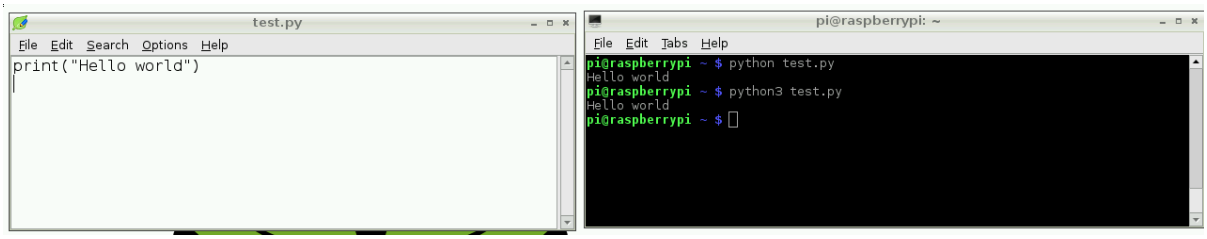
For example, in the new window, type:

```
n = 0

for i in range(1, 101):
    n += i

print("The sum of the numbers 1 to 100 is:")
print(n)
```

Then save this file (`File > Save` or `Ctrl + S`) and run (`Run > Run Module` or hit `F5`) and you'll see the output in your original Python window.

Using the Command Line

You can write a Python file in a standard editor, and run it as a Python script from the command line. Just navigate to the directory the file is saved in (use `cd` and `ls` for guidance) and run with `python3`, e.g. `python3 hello.py`.



Other Ways of Using Python

The standard built-in Python shell is accessed by typing `python3` in the terminal.

This shell is a prompt ready for Python commands to be entered. You can use this in the same way as Thonny, but it does not have syntax highlighting or autocompletion. You can look back on the history of the commands you've entered in the REPL by using the `Up/Down` keys. Use `Ctrl + D` to exit.

IPython

IPython is an interactive Python shell with syntax highlighting, autocompletion, pretty printing, built-in documentation, and more. IPython is not installed by default. Install with:

```
sudo pip3 install ipython
```

Then run with `ipython` from the command line. It works like the standard `python3`, but has more features. Try typing `len?` and hitting `Enter`. You're shown information including the docstring for the `len` function:

```
Type:       builtin_function_or_method
String Form:<built-in function len>
Namespace:  Python builtin
```

```
Docstring:
len(object) -> integer

Return the number of items of a sequence or mapping.
```

Try the following dictionary comprehension:

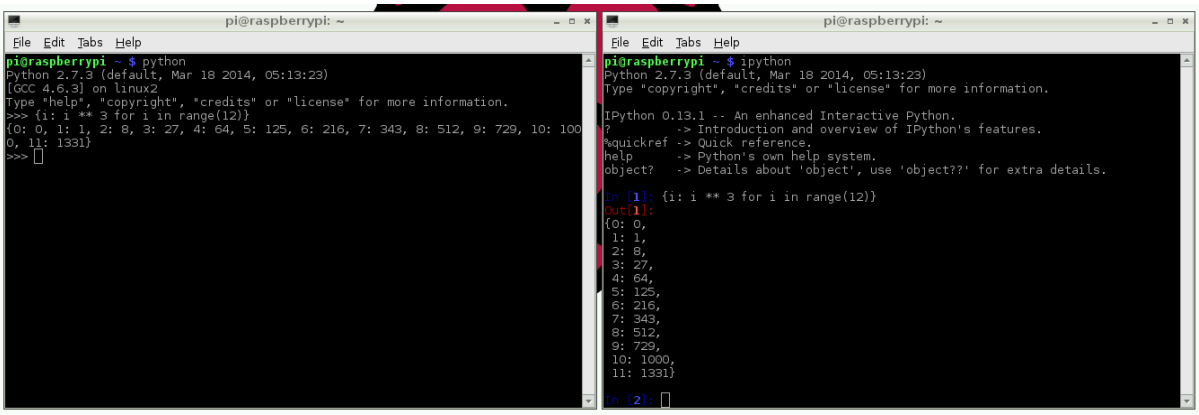
```
{i: i ** 3 for i in range(12)}
```

This will pretty print the following:

```
{0: 0,
 1: 1,
 2: 8,
 3: 27,
 4: 64,
 5: 125,
 6: 216,
 7: 343,
 8: 512,
 9: 729,
 10: 1000,
 11: 1331}
```

In the standard Python shell, this would have printed on one line:

```
{0: 0, 1: 1, 2: 8, 3: 27, 4: 64, 5: 125, 6: 216, 7: 343, 8: 512, 9: 729, 10: 10
00, 11: 1331}
```



You can look back on the history of the commands you've entered in the REPL by using the `Up/Down` keys like in `python`. The history also persists to the next session, so you can exit `ipython` and return (or switch between v2/3) and the history remains. Use `Ctrl + D` to exit.

Installing Python Libraries

apt

Some Python packages can be found in the Raspberry Pi OS archives, and can be installed using apt, for example:

```
sudo apt update
sudo apt install python-picamera
```

This is a preferable method of installing, as it means that the modules you install can be kept up to date easily with the usual `sudo apt update` and `sudo apt full-upgrade` commands.

pip

Not all Python packages are available in the Raspberry Pi OS archives, and those that are can sometimes be out of date. If you can't find a suitable version in the Raspberry Pi OS archives, you can install packages from the Python Package Index (known as PyPI).

To do so, install pip:

```
sudo apt install python3-pip
```

Then install Python packages (e.g. `simplejson`) with `pip3`:

```
sudo pip3 install simplejson
```

piwheels

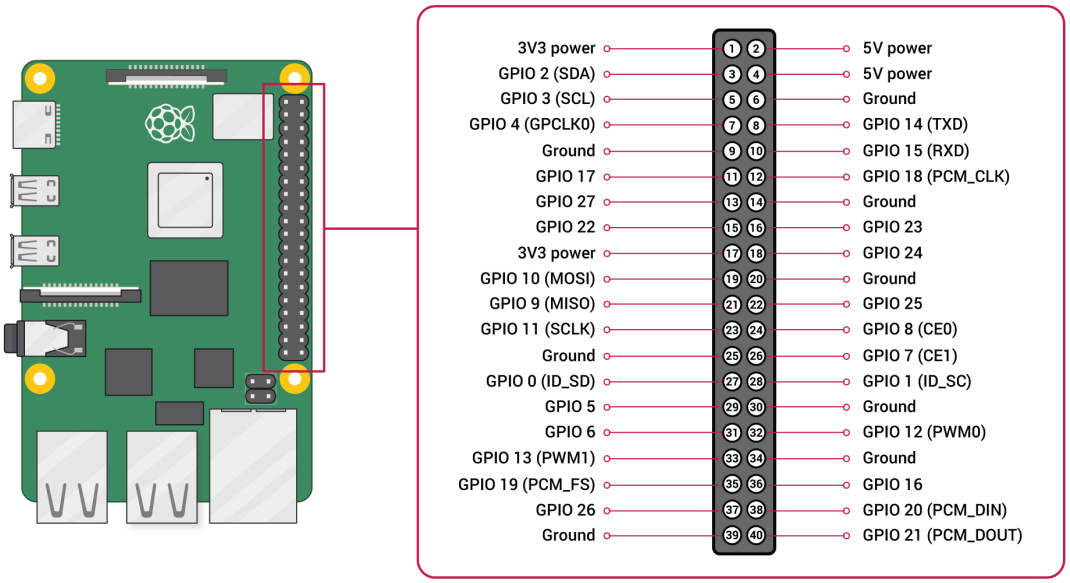
The official Python Package Index (PyPI) hosts files uploaded by package maintainers. Some packages require compilation (compiling C/C++ or similar code) in order to install them, which can be a time-consuming task, particlarly on the single-core Raspberry Pi 1 or Raspberry Pi Zero.

piwheels is a service providing pre-compiled packages (called *Python wheels*) ready for use on the Raspberry Pi. Raspberry Pi OS is pre-configured to use piwheels for pip. Read more about the piwheels project at www.piwheels.org.

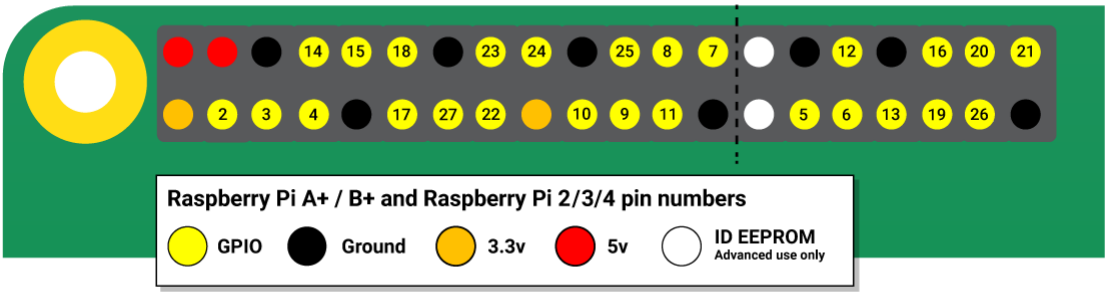
GPIO and the 40-pin Header

Edit this on GitHub

A powerful feature of the Raspberry Pi is the row of GPIO (general-purpose input/output) pins along the top edge of the board. A 40-pin GPIO header is found on all current Raspberry Pi boards (unpopulated on Raspberry Pi Zero, Raspberry Pi Zero W and Raspberry Pi Zero 2 W). Prior to the Raspberry Pi 1 Model B+ (2014), boards comprised a shorter 26-pin header.



Any of the GPIO pins can be designated (in software) as an input or output pin and used for a wide range of purposes.



NOTE

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 (see below).

Voltages

Two 5V pins and two 3.3V pins are present on the board, as well as a number of ground pins (0V), which are unconfigurable. The remaining pins are all general purpose 3.3V pins, meaning outputs are set to 3.3V and inputs are 3.3V-tolerant.

Outputs

A GPIO pin designated as an output pin can be set to high (3.3V) or low (0V).

Inputs

A GPIO pin designated as an input pin can be read as high (3.3V) or low (0V). This is made easier with the use of internal pull-up or pull-down resistors. Pins GPIO2 and GPIO3 have fixed pull-up resistors, but for other pins this can be configured in software.

More

As well as simple input and output devices, the GPIO pins can be used with a variety of alternative functions, some are available on all pins, others on specific pins.

- PWM (pulse-width modulation)
    - Software PWM available on all pins
    - Hardware PWM available on GPIO12, GPIO13, GPIO18, GPIO19
- SPI
    - SPI0: MOSI (GPIO10); MISO (GPIO9); SCLK (GPIO11); CE0 (GPIO8), CE1 (GPIO7)
    - SPI1: MOSI (GPIO20); MISO (GPIO19); SCLK (GPIO21); CE0 (GPIO18); CE1 (GPIO17); CE2 (GPIO16)
- I2C
    - Data: (GPIO2); Clock (GPIO3)

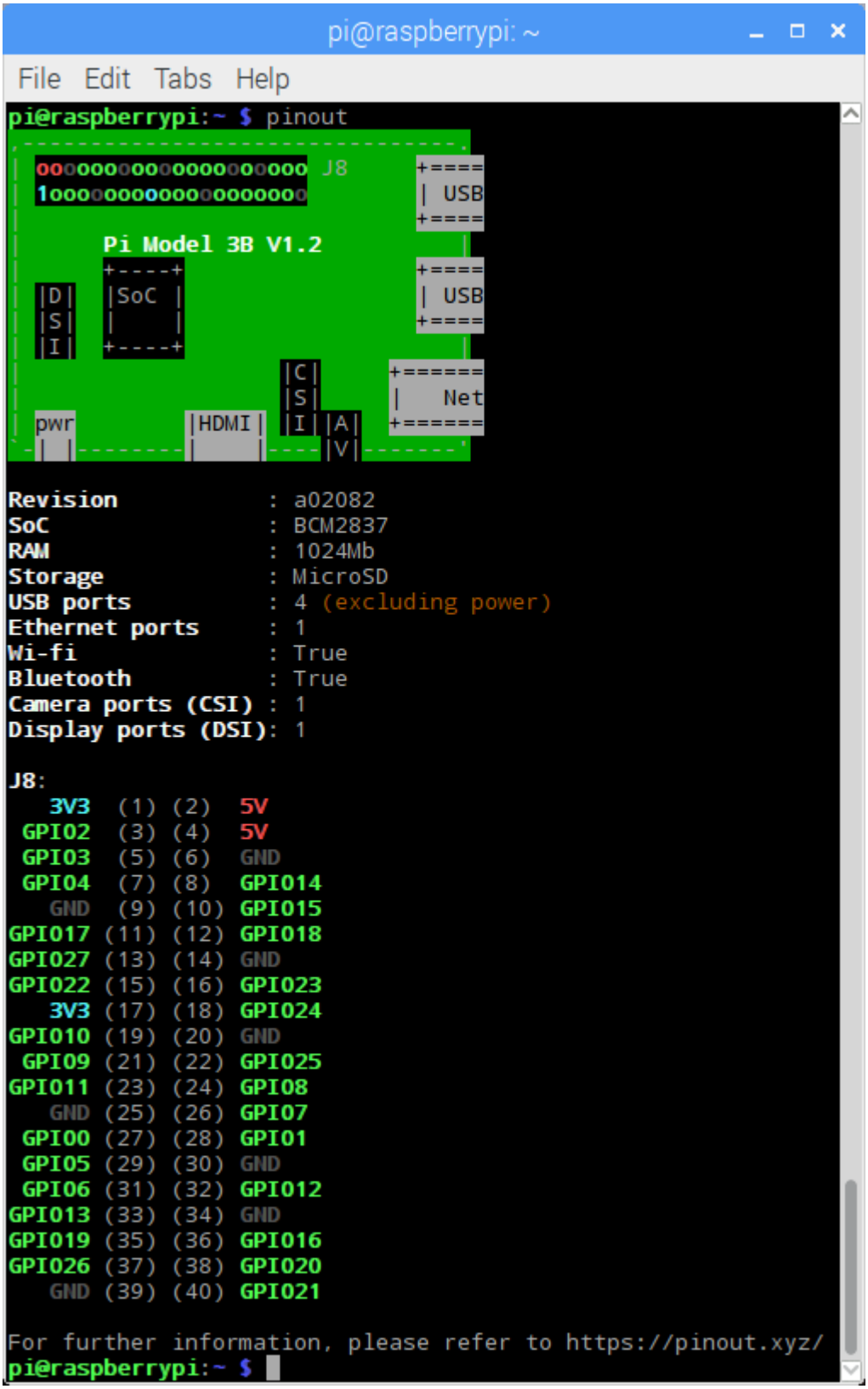
    - EEPROM Data: (GPIO0); EEPROM Clock (GPIO1)

- Serial

    - TX (GPIO14); RX (GPIO15)

GPIO pinout

A handy reference can be accessed on the Raspberry Pi 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 desktop image, but not on Raspberry Pi OS Lite.



For more details on the advanced capabilities of the GPIO pins see gadgetoid's interactive pinout diagram.

WARNING

While connecting up simple components to the GPIO pins is perfectly safe, it's important to be careful how you wire things up. 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.

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>
```

GPIO in Python

Using the GPIO Zero library makes it easy to get started with controlling GPIO devices with Python. The library is comprehensively documented at gpiozero.readthedocs.io.

LED

To control an LED connected to GPIO17, you can use this code:

```
from gpiozero import LED
from time import sleep

led = LED(17)

while True:
    led.on()
    sleep(1)
    led.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()`, and `blink()`.

Button

To read the state of a button connected to GPIO2, you can use this code:

```
from gpiozero import Button
from time import sleep

button = Button(2)

while True:
    if button.is_pressed:
        print("Pressed")
    else:
        print("Released")
    sleep(1)
```

Button functionality includes the properties `is_pressed` and `is_held`; callbacks `when_pressed`, `when_released`, and `when_held`; and methods `wait_for_press()` and `wait_for_release`.

Button + LED

To connect the LED and button together, you can use this code:

```
from gpiozero import LED, Button

led = LED(17)
button = Button(2)

while True:
    if button.is_pressed:
        led.on()
    else:
        led.off()
```

Alternatively:

```
from gpiozero import LED, Button

led = LED(17)
button = Button(2)

while True:
    button.wait_for_press()
    led.on()
    button.wait_for_release()
    led.off()
```

or:

```
from gpiozero import LED, Button

led = LED(17)
button = Button(2)
```

```
button.when_pressed = led.on
button.when_released = led.off
```