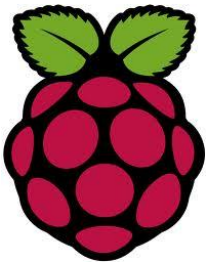
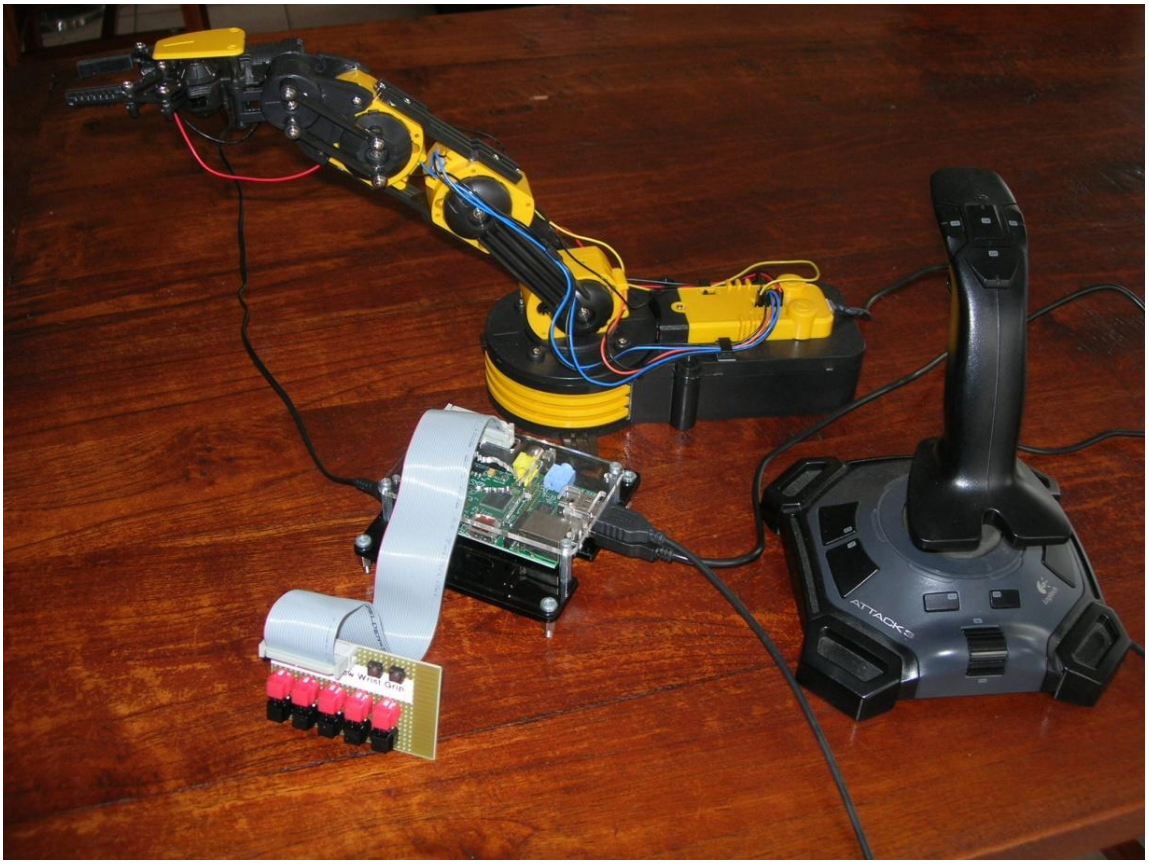


# Raspberry PI Robot Arm

## Constructors Manual




Bob Rathbone Computer Consultancy

www.bobrathbone.com



17th of April 2022

# Contents

# Figures



# Tables

# Introduction

This project has been designed to help students get started with a Robot on the Raspberry PI. The principal hardware required to build the Robot Arm consists of the following components:

- A Raspberry PI computer
- A Ex-Maplin Robotic Arm or modern replacement (See
- Optional Joystick (Logitech or similar)
- Optional twelve button interface (Self build)

## Raspberry PI computer

The **Raspberry Pi** is a credit-card-sized single-board computer developed in the United Kingdom by the Raspberry Pi Foundation with the intention of promoting the teaching of basic computer science in schools.

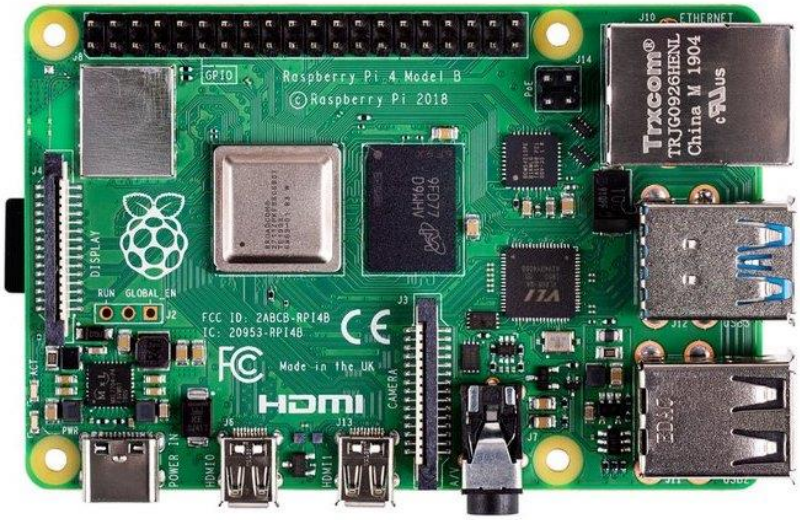



Figure 1 Raspberry PI model 4B Computer

Any model of Raspberry Pi can be used for this project as it only uses the first 26 pins on the GPIO header. More information on the Raspberry PI computer may be found here:
http://en.wikipedia.org/wiki/Raspberry_Pi

If you are new to the Raspberry PI, try the following beginners guide.
http://elinux.org/RPi_Beginners

Note: This project originally used the Maplin Robotic Arm with USB interface. Sadly, Maplin ceased high street trading in 2018 and at the last check no longer supply this product. It is however possible to purchase the same Robotic arm with its own button interface. The button interface can be replaced with an optional USB interface made by Rapid.

The optional Rapid USB interface is available from.
https://www.rapidonline.com/Catalogue/Product?Id=06-9350

Note: The Rapid USB interface has not been tested with the software described here and there is a small risk that software may require further modification. See technical support on page .

# Features

The Robot Arm interface provides five separate ways of driving a Maplin Robot Arm. These are:

1. A USB Joystick (Logitech or similar)
2. A twelve-push button interface
3. A keyboard interface
4. Using an input file containing the required robot commands
5. A command line interface using required robot commands

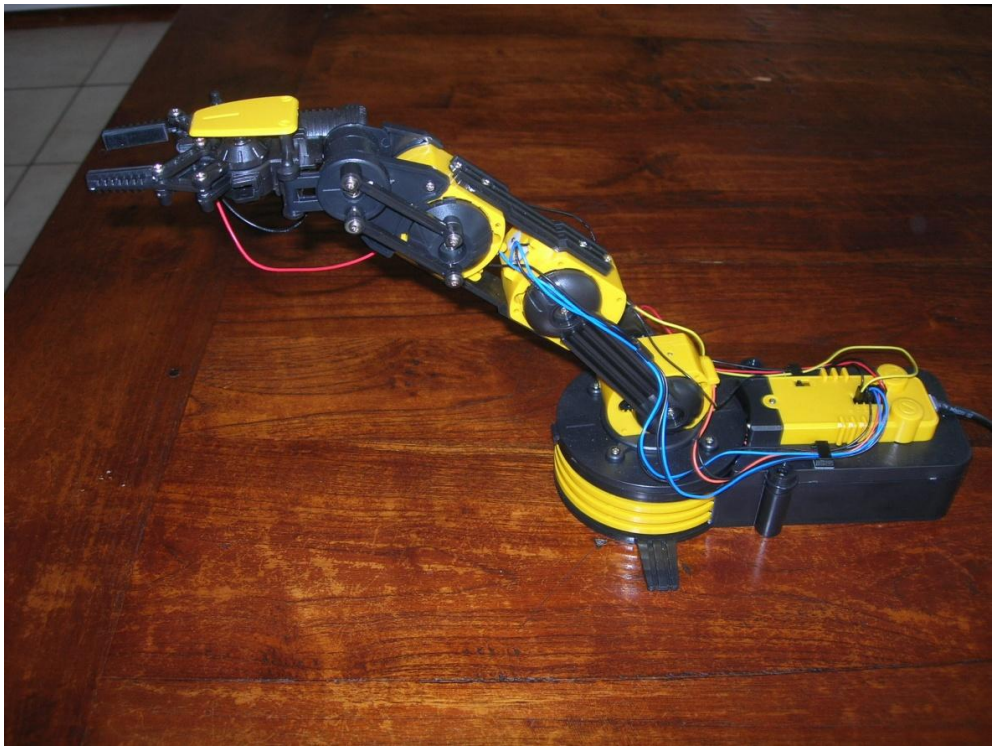


Figure 2 Maplin Robotic Arm with USB interface

Note: The above figure shows the original; Maplin Robotic Arm with USB interface. Sadly, Maplin ceased high street trading in 2018 and at the last check no longer supply this product in their on-line store. Fortunately, an equivalent product is available from Rapid at www.rapidonline.com

## Limitations

The Maplin Robot Arm does not have any positional feedback that the program can make use of. The program has no idea of the current position of the robot arm or gripper. This means that you must observe the Robot Arm position whilst moving it around to position it to say pick up an object. This means that the robot arm can never be programmed to carry out a repetitive task with any accuracy. This is purely a fun project with very limited application.

# Parts list

The following table shows the parts list for the Raspberry PI Robot interface.

Table 1 Parts List

| Quantity | Item | Description | Supplier |
|---|---|---|---|
| **1** | Raspberry PI | Raspberry PI credit card computer | Pi-Hut |
| **1** | Logitech ATK Joystick | Two axis joystick with at least 8 buttons | Any computer store |
| **1** | Maplin Robot Arm | Ex-Maplin Robotic Arm with USB interface as used with this project | No longer available See next two items |
| **1** | Rapid Robotic Arm | Rapid Robotic Arm with wired control Part number 06-9349 | Rapid on-line |
| **1** | Rapid USB interface | Rapid USB Interface Kit for Robotic Arm p/n 06-9349 | Rapid on-line |
| **1** | 16 Gigabyte SD card | For the Raspberry Pi OS Bullseye operating system | Any computer or photographic store |
| **1** | Prototype board | Button interface board | Tandy or Farnell Element 14 |
| **10** | Push to make PCB mount button | Button interface board push buttons (Robot movement buttons) | Tandy or Farnell Element 14 |
| **2** | Push to make miniature PCB mount button | Button interface board push buttons (light on/off) | Tandy or Farnell Element 14 |
| **1** | 26-way PCB mount male connector | To take the ribbon cable connection to the button interface board | Tandy or Farnell Element 14 |
| **1** | 26-way ribbon cable | Ribbon cable connection to the button interface board | Tandy or Farnell Element 14 |
| **n/a** | Wiring | Thin wire for PCB wiring prototype board | Any electronics shop |

Figure 3 Rapid Robotic Arm with wired control

As the Maplin Robotic arm is no longer available, the **Rapid Robotic Arm with wired control** can be used instead. This comes with its own control unit and does not come with a USB interface.

For this project it is necessary to purchase the Rapid USB Interface Kit for the Robotic Arm as shown in Figure 4 on page 7.

Once the USB interface has been fitted the original control unit supplied with the Robot arm can no longer used and it is necessary to use either a keyboard, joystick or button interface as shown later in this project.


Figure 4 Rapid USB Interface Kit for Robotic Arm

The Rapid USB Interface Kit for Robotic Arm 06-9349 comes as a  with PC test software on a CD for PC and instruction booklet.

This replaces the original control unit supplied with the Robot arm. This can then no longer be used.

# Construction

## Robot Arm

The Robot Arm comes in kit form with full instructions how to build it. If using the Rapid Robot Arm with its own controller, first build and test the unit out-of-the-box before attempting to fit the USB Interface adapter.

## Button Interface Wiring

Wire 3.3v on pin 1 to one side of all of the switches. Wire the other side to the GPIO pin shown in the following table.

**Table 2 Button Interface Wiring**

| GPIO pin | Switch | Description |
|---|---|---|
| **1** | All switches | Common +3.3 volt |
| **7** | Base clockwise | |
| **8** | Base anti-clockwise | |
| **11** | Shoulder Up | |
| **10** | Shoulder Down | |
| **13** | Elbow Up | |
| **12** | Elbow Down | |
| **18** | Wrist Up | |
| **16** | Wrist Down | |
| **21** | Gripper Open | |
| **19** | Gripper Close | |
| **22** | Light On | |
| **23** | Light | |

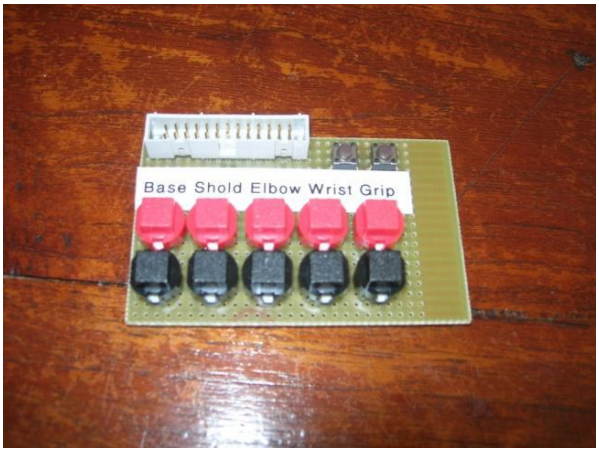



Figure 5 The twelve-button interface board

The picture on the left shows the completed twelve button interface board. The larger red buttons are the Up or Open (grip) functions. The larger black buttons are the Down or Close (gripper) functions. The two smaller buttons on the top left are the LED light on and off respectively.

Figure 6 The underside of the twelve-button interface

The actual wiring as shown by this view of the underside of the board is relatively simple. It consists of wiring one side of all of the switches to the 3.3v on pin 1. Use a volt meter to test between pin 6 (GND) of the GPIO header and the switches voltage supply is 3.3 volts and not 5 volts. The other side of each switch is wired to the appropriate GPIO pin shown in Table 1 above.

**Caution: Do not wire the switches to the +5 volt supply on pin 2 by mistake or you will irreparably damage the raspberry PI. Check for the correct voltage before continuing to wire the switches.**

## GPIO Hardware Notes

The following shows the pin outs for the GPIO pins for both 26-pin and 40-pin Raspberry PIs. For more information see: http://elinux.org/RPi_Low-level_peripherals.

### GPIO Numbers

**Raspberry Pi B**
**Rev 1 P1 GPIO Header**

| | Pin No. | | |
|---|---|---|---|
| 3.3V | 1 | 2 | 5V |
| GPIO0 | 3 | 4 | 5V |
| GPIO1 | 5 | 6 | GND |
| GPIO4 | 7 | 8 | GPIO14 |
| GND | 9 | 10 | GPIO15 |
| GPIO17 | 11 | 12 | GPIO18 |
| GPIO21 | 13 | 14 | GND |
| GPIO22 | 15 | 16 | GPIO23 |
| 3.3V | 17 | 18 | GPIO24 |
| GPIO10 | 19 | 20 | GND |
| GPIO9 | 21 | 22 | GPIO25 |
| GPIO11 | 23 | 24 | GPIO8 |
| GND | 25 | 26 | GPIO7 |

**Raspberry Pi A/B**
**Rev 2 P1 GPIO Header**

| | Pin No. | | |
|---|---|---|---|
| 3.3V | 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 |
| 3.3V | 17 | 18 | GPIO24 |
| GPIO10 | 19 | 20 | GND |
| GPIO9 | 21 | 22 | GPIO25 |
| GPIO11 | 23 | 24 | GPIO8 |
| GND | 25 | 26 | GPIO7 |

**Raspberry Pi B+**
**B+ J8 GPIO Header**

| | Pin No. | | |
|---|---|---|---|
| 3.3V | 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 |
| 3.3V | 17 | 18 | GPIO24 |
| GPIO10 | 19 | 20 | GND |
| GPIO9 | 21 | 22 | GPIO25 |
| GPIO11 | 23 | 24 | GPIO8 |
| GND | 25 | 26 | GPIO7 |
| DNC | 27 | 28 | DNC |
| GPIO5 | 29 | 30 | GND |
| GPIO6 | 31 | 32 | GPIO12 |
| GPIO13 | 33 | 34 | GND |
| GPIO19 | 35 | 36 | GPIO16 |
| GPIO26 | 37 | 38 | GPIO20 |
| GND | 39 | 40 | GPIO21 |

**Key**

| | |
|---|---|
| Power + | UART |
| GND | SPI |
| I²C | GPIO |

# Software installation

## Conventions used in this tutorial

Installation of the radio program requires you to enter lines at the command line prompt. This requires you to log into the Raspberry PI as user '**pi**'. The default password is **raspberry**.

**Note:** Don't carry out any of the following commands just yet. They are just examples.

```
Raspberrypi login: pi
Password: raspberry
Last login: Mon Apr 11 09:43:53 2022 from fe80::44d4:59a1:ce1:e5e1%wlan0
pi@raspberrypi:~$
```

The prompt line is displayed ending with a **$** sign. The **pi@raspberrypi:~** string means user 'pi' on host machine called 'raspberrypi'. The ~ character means the user 'pi' home directory **/home/pi**. In this tutorial if you are required to do something as user **pi** then only the $ sign will be shown followed by the command as shown in the example below:

```
$ cd pirobot
$ sudo ./robotd start
```

Some commands produce output which does not need to be shown. In such a case a ':' is used to indicate that some output has been omitted.

```
$ Linux raspberrypi 5.15.32-v7l+ #1538 SMP Thu Mar 31 19:39:41 BST 2022
armv7l
: {Output ommitted}
Last login: Mon Apr 11 09:43:53 2022 from fe80::44d4:59a1:ce1:e5e1%wlan0
```

END OF EXAMPLE COMMANDS.

## Installing the OS on an SD card

See https://www.raspberrypi.com/documentation/computers/getting-started.html

The Raspberry Pi usually boots from a SD card. This software uses the 32 bit desktop **Raspberry Pi OS**. Use a 16GByte SD card. First download and install **Raspberry Pi OS imager** onto a **PC** or **Mac** from: https://www.raspberrypi.com/software.

Run the imager software and select Raspberry Pi OS Desktop (Bullseye) using the **Operating System** button. Then select your SD card using the **Storage** button.

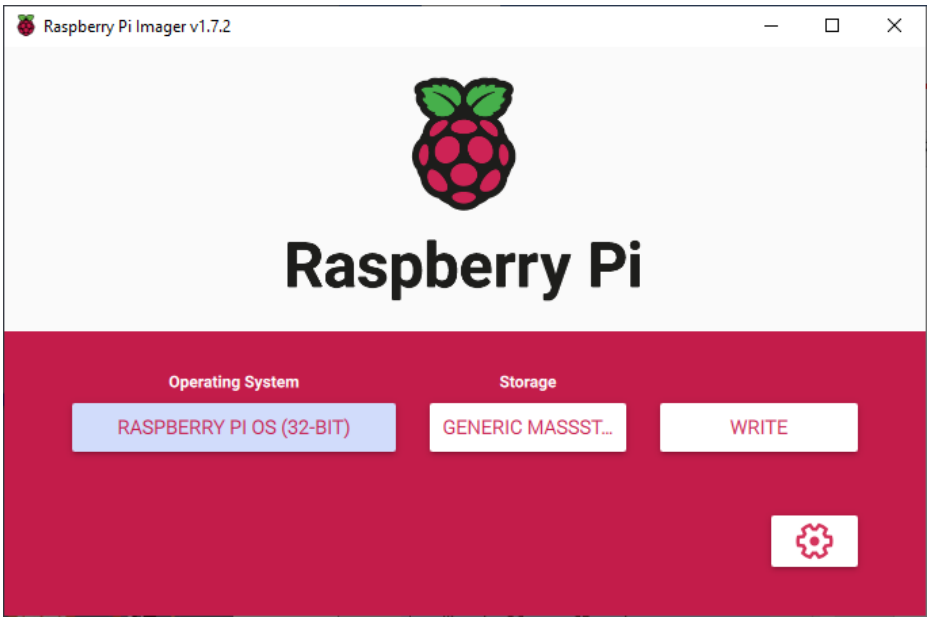



Figure 7 Raspberry Pi imager software

Click on the settings (Options) icon in the bottom right of the above imager dialogue. Set the hostname and enable SSH.

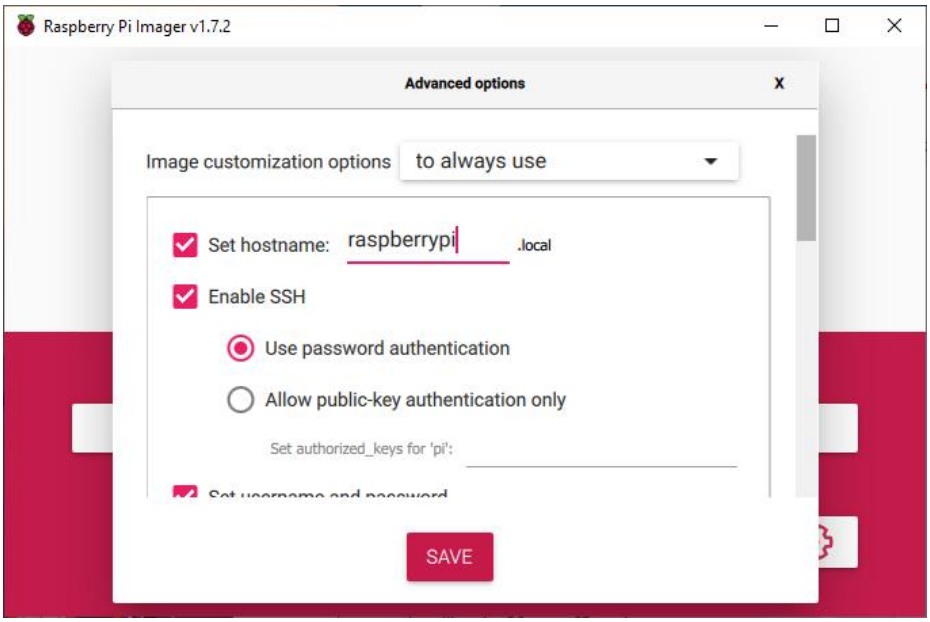



Figure 8 Imager advanced settings (1)

Scroll down to set the user name to pi with a password of your choice. If using a Wi-Fi network set the SSID of your router and password. Finally press “Save”.

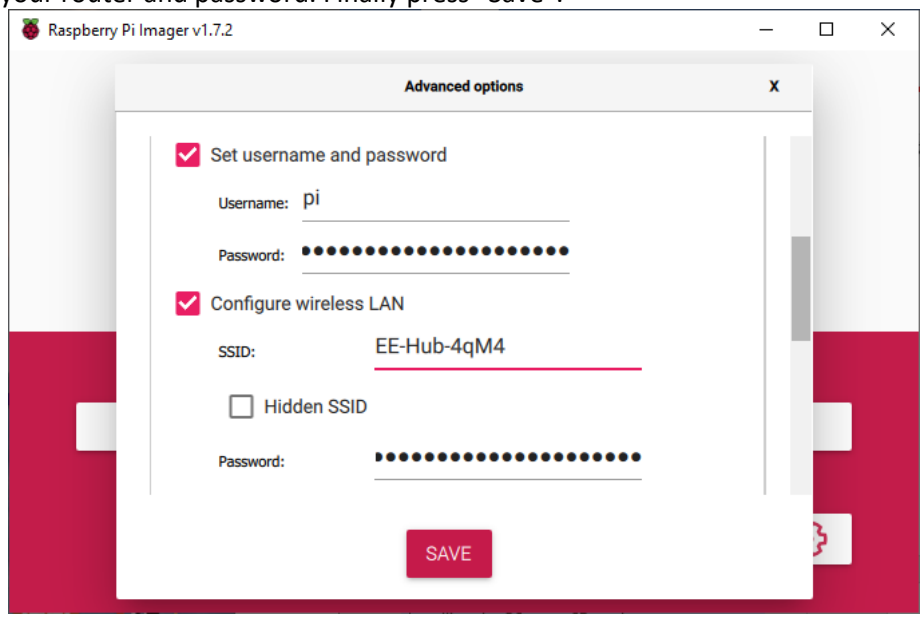



Figure 9 Imager advanced options (2)

Now press “Write”.


Once finished remove the SD card from the PC. Put the SD card into Raspberry Pi and switch it on.

## Logging into the Raspberry Pi for the first time

There are two choices to access the Raspberry Pi:

1. Connect a keyboard, mouse and HDMI screen and access the RPi via the desktop interface
2. Use either **Putty** or **Bitvise** software installed on the PC to log into the Raspberry Pi:

### Bitvise

Bitvise can be downloaded from https://www.bitvise.com. Once installed and opened the following screen will be displayed:

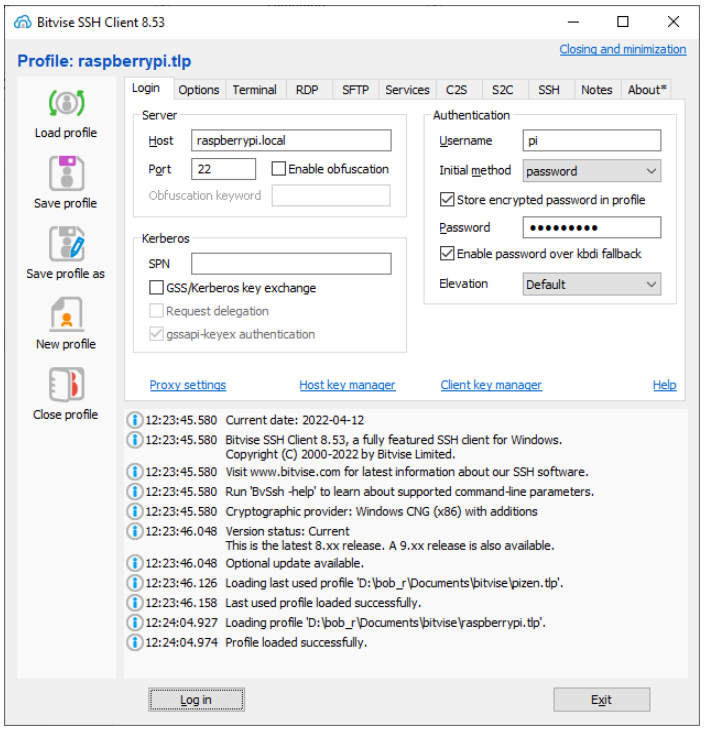



Figure 10 Bitvise SSH client

### Raspberry Pi OS Desktop

Clicking on “Log in” will, if all is correct, open a terminal widow.
The alternative is to access the RPi using the Raspberry Pi OS desktop.

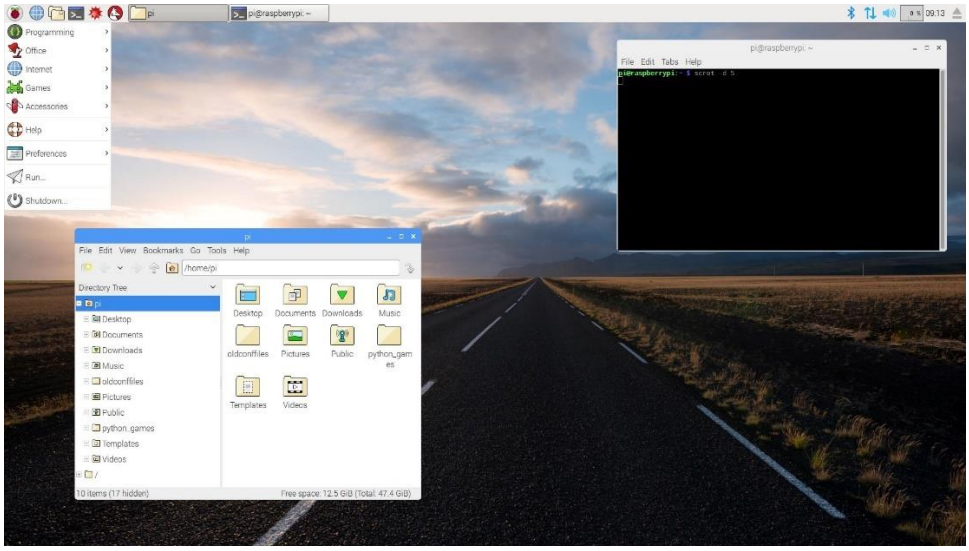


Figure 11 Raspberry Pi OS Desktop

## Installing the Robot Arm software

Install necessary to install some packages

```
$ sudo apt install python3-pygame python3-usb
```

The Robot Arm software is available from GitHub

```
$ cd
$ git clone https://github.com/bobrathbone/pirobot
```

Change to the pirobot directory and make all the Python file executable.

```
$ cd pirobot
$ chmod +x *.py
```

Connect the Robotic Arm to any USB port and switch it on.
Now run the test program

```
$ sudo ./robot.py
```

The following is displayed

```
pygame 1.9.6
Hello from the pygame community. https://www.pygame.org/contribute.html
Key Command
--- -------
a   base-anti-clockwise
z   base-clockwise
d   shoulder-up
c   shoulder-down
f   elbow-up
v   elbow-down
g   wrist-up
b   wrist-down
h   grip-open
n   grip-close
j   light-on
m   light-off
k   stop
x   Exit program
```

Execute some commands, for example j, m, d and c via the keyboard.

```
execute light-on
execute light-off
execute shoulder-up
execute shoulder-down
```

# Operation

Connect the Robotic Arm to any USB port on the Raspberry Pi and switch it on.
Connect the joystick, if you have one, to the second USB port.
If you have built the twelve-button keypad then connect it to the 26 or 40-pin GPIO header.

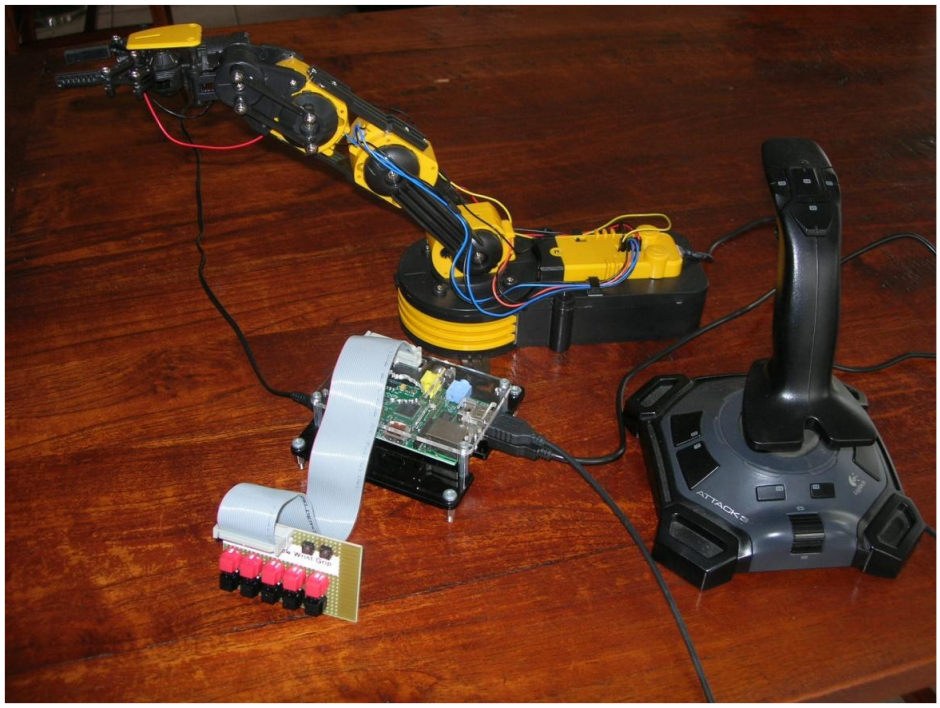


Figure 12 Connecting up the Robotic arm

Log in to the Raspberry as user pi and issue the following commands:

```
$ cd pirobot
$ sudo ./robotd.py
```

This will display the following message:

```
Usage: ./robotd.py start|stop|restart|status|version|nodaemon<command>
Commands: keyboard - Use keyboard
execute <file> - Execute commands in <file>
```

## Using the keyboard

Issue the following command:

```
$ sudo ./robotd.py keyboard
```

This will display the following:

```
Key Command
--- -------
a base-anti-clockwise
c shoulder-down
b wrist-down
d shoulder-up
g wrist-up
f elbow-up
h grip-open
k stop
j light-on
m light-off
n grip-close
v elbow-down
z base-clockwise
x Exit program
Enter command:
```

Now press the appropriate keys on the keyboard to operate the robot arm. Press key ‘x’ to exit the program.

If this doesn’t work go to the troubleshooting section on page TO BE DONE to determine the exact problem. There is no point attempting to use the joystick or twelve button keypad until the above commands are working.

## Using the Joystick

To use the joystick first start the robot daemon. A daemon is a special type of program which runs in the background and is not connected to a terminal.

```
$ sudo ./robotd.py start
```

If it started correctly, you can check its status with the following command:

```
$ sudo./robotd.py status
robotd running pid 2437
```

It will display the process ID (pid) of the running **robotd** daemon. The **pid** displayed will be different each time the program is run.

You can now operate the joystick. Left and right of the joystick will turn the base clockwise and anti-clockwise. The buttons will move the shoulder, wrist and the gripper. Two more buttons will operate the LED light.

See Table TO BE DONE on page 13 for the complete set off commands. Joystick buttons are labelled 0 onwards. Bob Rathbone |Raspberry PI Robotic Arm Running the robotic arm program

Table 3 Joystick commands

| Commands | Joystick movement | Joystick button |
|---|---|---|
| **grip-close** | | 0 |
| **grip-open** | | 1 |
| **wrist-up** | | 9 |
| **wrist-down** | | 10 |
| **elbow-up** | | 6 |
| **elbow-down** | | 5 |
| **shoulder-up** | back | |
| **shoulder-down** | forward | |
| **base-clockwise** | left | 4 |
| **base-anti-clockwise** | right | 3 |
| **light-off** | n/a | 7 |
| **light-on** | n/a | 8 |

## Using the twelve-button keypad

This is optional. If you did not build the keypad then you can use operate the robotic arm using the joystick only (and of course using the keypad). The following table and illustration show the robotic arm to button mapping.

Table 4 Keypad commands

| Command | Button | Colour |
|---|---|---|
| **base-clockwise** | Base | Red |
| **base-anti-clockwise** | Base | Black |
| **shoulder-up** | Shold | Red |
| **shoulder-down** | Shold | Black |
| **elbow-up** | Elbow | Red |
| **elbow-down** | Elbow | Black |
| **wrist-up** | Wrist | Red |
| **wrist-down** | Wrist | Black |
| **grip-open** | Grip | Red |
| **grip-close** | Grip | Black |
| **light-off** | Top left | Chrome |
| **light-on** | Top right | Chrome |


Figure 13 Push-button keypad

## Displaying the software version

Issue the following command:

```
$ sudo ./robotd.py version
Version 1.0
```

## Stopping the robot program

```
$ sudo ./robotd.py stop
```

## Using a command input file

It is possible to execute commands from an input file using the execute option. To use a commands file, use the execute command followed by the name of the commands file.

```
$ sudo./robotd.py execute <commands file name>
```

For example to execute the commands in a file called commands.txt, enter the following.

```
$ sudo ./robotd.py execute commands.txt
```

Table 5 lists all commands and the maximum travel time and half movement travel time. For example, if you wish to rotate the base clockwise from its centre position to complete left the command in the commands text file would be:

```
base-clockwise 10
```

**Table 5 File input commands and maximum travel time in seconds**

| Command | Maximum time in seconds | ½ travel time |
|---|---|---|
| **base-clockwise** | 20 | 10 |
| **base-anti-clockwise** | 20 | 10 |
| **shoulder-up** | 14 | 7 |
| **shoulder-down** | 14 | 7 |
| **elbow-up** | 13 | 6.5 |
| **elbow-down** | 13 | 6.5 |
| **wrist-up** | 9 | 4.5 |
| **wrist-down** | 9 | 4.5 |
| **grip-open** | 3 | 1.5 |
| **grip-close** | 3 | 1.5 |
| **light-off** | n/a | n/a |
| **light-on** | n/a | n/a |
| **wait** | any | n/a |

## Debugging using the nodaemon mode

Normally **robotd.py** runs in the background so apart from looking at the log file it is not possible to see what is happening, for example, in the event of a program crash. However, it is possible to run the program in the diagnostic **nodaemon** mode as shown below.

```
$ sudo ./robotd.py nodaemon
Radio running pid 1375
```

If the program crashes the crash dump will be seen on screen allowing diagnosis.

## Starting the the Robot Daemon automatically at boot time

Copy the **robotd.service** startup script to the **/usr/lib/systemd/**system directory

```
$ cd pirobot
$ sudo cp robotd.service /usr/lib/systemd/system/.
```

Enable the Robot service.

```
$ sudo systemctl enable robotd.service
Created symlink /etc/systemd/system/multi-user.target.wants/robotd.service →
/lib/systemd/system/robotd.service.
```

## Logging

All logging is written to the **/var/log/robot/robot.log** file.

The log level is **INFO**, **WARNING**, **ERROR** and **DEBUG**. The log level is by default set to **INFO** in the **/var/lib/robotd/loglevel** file. If greater logging detail is required, set this to DEBUG.

```
$ sudo echo DEBUG > /var/lib/robotd/loglevel
```

Typical log output.

```
2022-04-15 10:09:38,273 INFO Robot daemon running pid 3349
2022-04-15 10:09:38,597 INFO Waiting for joystick count = 0
2022-04-15 10:09:41,716 INFO Waiting for joystick count = 0
2022-04-15 10:09:44,834 INFO No Joystick found!
2022-04-15 10:09:45,001 DEBUG Listening for commands
2022-04-15 10:10:29,360 DEBUG Switch event 7
2022-04-15 10:10:29,443 DEBUG sendCommand [0, 2, 0]
2022-04-15 10:10:29,722 DEBUG sendCommand [0, 0, 0]
2022-04-15 10:10:34,656 DEBUG Switch event 12
2022-04-15 10:10:34,721 DEBUG sendCommand [32, 0, 0]
2022-04-15 10:10:34,901 DEBUG sendCommand [0, 0, 0]
```

# Trouble shooting problems

## Boot problems

### Newly created desktop version of Raspberry Pi OS will not boot

This is a very recent problem. After creating a new SD card using Raspberry Pi OS (32 bit) dated 1st January 2022 or later the Raspberry Pi will not boot.

This appears to have started in March 2022. It even seems to be trying to reboot several times. This has been experienced on the Raspberry Pi 4B. Other models may also be affected. This appears to be a problem with new DRM VC4 V3D screen driver (**vc4-kms-v3d**).

Insert the SD card into a PC. The PC will display a boot partition as shown in the example below:

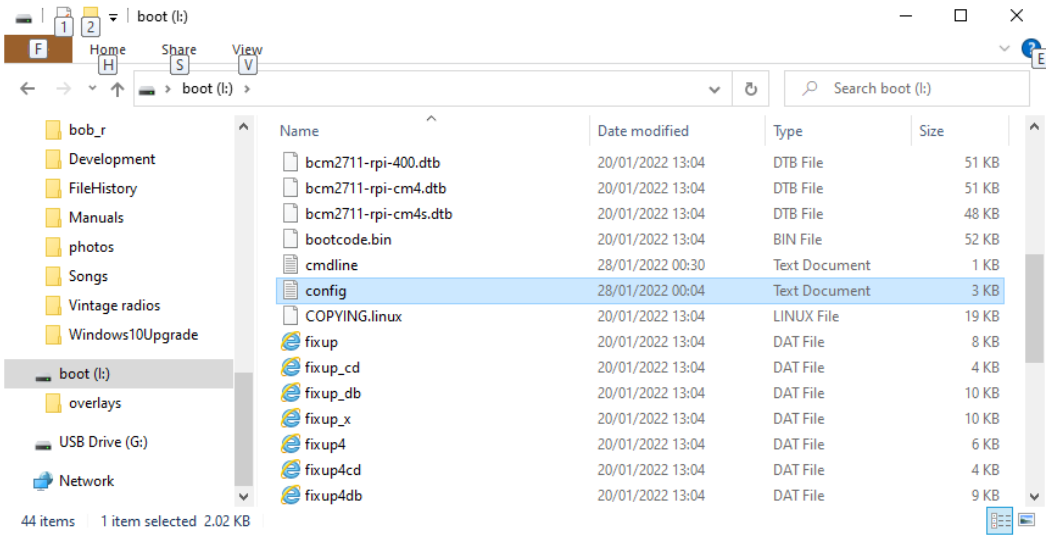



Figure 14 SD card boot patition

In the above example this is on drive I: but will most likely be different on your PC. Find the config file (config.txt) on the boot drive and open it with **Notepad** or any other text editor. Find the entry **dtoverlay=vc4-kms-v3d**.

```
# Enable DRM VC4 V3D driver
#dtoverlay=vc4-kms-v3d
max_framebuffers=2
```

Either disable the driver by putting a **#** character at the beginning of the line or use the older **vc4-fkms-v3d dtoverlay as shown below**

Using the older **vc4-fkms-v3d** dtoverlay:

```
dtoverlay=vc4-fkms-v3d
```

Save the file and try booting off the modified SD card. All being well it should reboot OK. If not look for other causes as shown in the rest of this section.

## The Raspberry Pi will not boot from a previously good SD card

This is always worrying if this happens but doesn't always mean that the situation is irrecoverable. It often indicates a SD card corruption problem. Connect the HDMI output of the Raspberry Pi to the HDMI input of a TV set and attach a USB keyboard. Reboot the Pi. If you see the following:

```
[....] An automatic file system check (fsck) of the root filesystem failed.
A manual fsck must be performed, then the system restarted. The fsck should
be performed in maintenance mode with the root filesystem mounted in read-on
[FAIL] ... failed!
[warn] The root filesystem is currently mounted in read-only mode. A
maintenance shell will now be started. After performing system maintenance,
please CONTROL-D to terminate the maintenance shell and restart the system.
... (warning).
sulogin: root account is locked, starting shell
root@raspberrypi:~#
```

You are asked to press enter which brings up the dollar $ prompt

```
Cannot access console, press enter to continue
$
```

Enter a root password that you can remember.

```
$ sudo passwd root
Enter new UNIX password:
Retype new UNIX password:
Password successfully changed
$ sudo reboot
```

When reboot has finished you will be asked to enter the root password you just entered which will take you to the # prompt. Then run the following commands:

```
# umount /
# fsck -y /dev/mmcblk0p2
```

This will, with luck, correct the file system. When **fsck** has finished reboot the system. Once rebooted use **vi** or **nano** to modify **/etc/default/rcS**. Add the following line to **/etc/default/rcS**.

```
# automatically repair filesystems with inconsistencies during boot
FSCKFIX=yes
```

Finally reboot the Raspberry Pi.

```
$ sudo reboot
```

## Bad SD card

If experiencing boot problems, a bad SD card cannot be ruled out. If all of the above efforts fail to resolve the problem try installing the **Raspberry Pi OS** on a different SD card.

### Robot programs say Daemon already running

When running the robot programs the following is seen:

```
$ sudo ./robotd.py nodaemon
pidfile /var/run/robotd.pid already exist. Daemon already running?
```

This is because the /var/run/robotd.pid file still exists due to program interruption or a crash. To overcome this simply issue a “stop” command.

```
$ sudo ./robotd.py stop
```

# Technical support

Technical support is on a voluntary basis by e-mail only at bob@bobrathbone.com. If there are any problems with this email address then also CC r.h.rathbone@gmail.com. Before asking for support, please first consult the troubleshooting section on page Trouble shooting problems on page 21**Error! Bookmark not defined.**. I will always respond to e-mails requesting help and will never ignore them. I only ask that you do the same (i.e. Did my suggestions help or not?). Be sure to provide the following information:

- A clear description of the fault.
- What you have already done to locate the problem?
- Is anything displayed on the LCD or Graphics screen?
- Did you run the test programs and what was the result?
- Did you vary from the procedure in the manual or add any other software?

Please note that support for general Raspberry PI problems is not provided. Only issues relating to the Radio software will be investigated.

For general Raspberry PI support see the following site:
http://www.raspberrypi.org/forums/

# Glossary

**CLI** Command Line Interface

**GPIO** General Purpose IO (On the Raspberry PI)

**RPi** Raspberry Pi

**SSH** Secure Shell. Either server or client

**USB** Universal Serial Bus

# Appendix A - Licences

The software and documentation for this project is released under the GNU General Public Licence.

The GNU General Public License (GNU GPL or GPL) is the most widely used free software license, which guarantees end users (individuals, organizations, companies) the freedoms to use, study, share (copy), and modify the software. Software that ensures that these rights are retained is called free software. The license was originally written by Richard Stallman of the Free Software Foundation (FSF) for the GNU project.

The GPL grants the recipients of a computer program the rights of the Free Software Definition and uses *copyleft* to ensure the freedoms are preserved whenever the work is distributed, even when the work is changed or added to. The GPL is a *copyleft* license, which means that derived works can only be distributed under the same license terms. This is in distinction to permissive free software licenses, of which the BSD licenses are the standard examples. GPL was the first *copyleft* license for general use. This means that you may modify and distribute the software and documentation subject to the conditions of the licences.

See http://www.gnu.org/licenses for further information on the GNU General Public License.

The licences for the source and documentation for this project are:

| | |
|---|---|
| GNU General Public License. | See http://www.gnu.org/licenses/gpl.html |
| GNU AFFERO General Public License. | See http://www.gnu.org/licenses/agpl.html |
| GNU Free Documentation License. | See http://www.gnu.org/licenses/fdl.html |