
Exploring Computer Science with the Raspberry Pi

Derek C. Schuurman

Professor of Computer Science, Calvin University
2023-2024



Creative Commons Attribution-NonCommercial-ShareAlike License

Contents

1	Introduction	4
1.1	The Raspberry Pi	4
1.2	Initial Setup of the Raspberry Pi	5
1.2.1	Getting Started with the Command Line	5
1.3	Text Editors	7
1.3.1	Nano	8
1.3.2	Emacs	8
1.3.3	Vim	9
1.4	Configuring the Raspberry Pi OS	10
1.5	Connecting Remotely to the Raspberry Pi	10
1.5.1	Connecting using a USB-to-TTL serial cable	11
1.5.2	Connecting over Ethernet or Wi-Fi	12
1.6	Remote Editing with Vscode	15
1.7	Proper Shutdown	16
2	Programming Languages and the Raspberry Pi	17
2.1	Running an Assembly Language Program	17
2.2	Higher Level Programming Languages	18
2.2.1	The Python Programming Language	19
2.3	Compiling and Running a C/C++ Program	22
2.3.1	Using a C Debugger	23
2.3.2	Library Documentation	25
2.3.3	Other Tools for C and C++	25
2.4	Compiling and Running Java Programs	26
2.5	Other Programming Languages	27
2.6	Comparing Runtime Efficiency of Different Programming Languages	27
2.6.1	Measuring execution time	28
2.6.2	Execution Speedup using Parallel Execution	31
3	The Raspberry Pi Operating System	35
3.1	Introduction	35
3.2	Process Management	35
3.2.1	Tools for Managing Processes	35
3.2.2	Example Program to fork a new process	36
3.2.3	Multithreaded programming	37
3.3	Memory Management	38

3.4	Managing I/O	38
3.5	Controlling Input and Output	38
3.6	Other Support Functions	39
3.6.1	Logfiles	39
3.6.2	Updating the Operating System	39
3.6.3	Securing your Raspberry Pi	40
3.6.4	Setting up a Print Server	42
3.7	Compiling the Linux Kernel	42
4	Networking	45
4.1	Networking Utilities	45
4.1.1	ping	45
4.1.2	traceroute	45
4.1.3	mtr	46
4.1.4	dig	46
4.1.5	wget	47
4.1.6	curl	47
4.1.7	telnet	48
4.1.8	Drill Exercises	48
4.2	The Web	48
4.2.1	Lighttpd	49
4.2.2	Nginx	49
4.3	Java Network Programming	50
5	Databases	54
5.1	SQL Databases and the Raspberry Pi	54
5.1.1	Using SQLite	54
5.1.2	Using MySQL	55
5.1.3	Backing Up MySQL Data	60
5.1.4	Using PostgreSQL	62
5.1.5	Creating Tables	63
6	Embedded System and the Internet of Things	67
6.1	The Raspberry Pi Hardware	67
6.2	Controlling GPIO outputs	67
6.2.1	Using I2C	72
6.2.2	Using a Camera	72
6.3	Message Queuing Telemetry Transport Protocol (MQTT) for IoT	74
6.3.1	Sending MQTT messages from the command line	74

6.3.2	Controlling an LED using Python and MQTT	74
6.3.3	Using MQTT to control Zigbee Devices	76
7	Exploring Artificial Intelligence on the Raspberry Pi	81
7.1	SciKit Learn	81
7.1.1	Using SciKit Learn with Linear Discriminate Analysis (LDA)	82
7.1.2	Using SciKit Learn with Principal Component Analysis (PCA)	83
7.1.3	Using SciKit Learn with Support Vector Machines (SVM)	84
7.1.4	Using SciKit Learn for SVM Image Classification	87
7.2	TensorFlow Lite	91
8	Technical Tools for Computer Scientists	92
8.1	Document Preparation	92
8.1.1	LaTeX	92
8.1.2	pandoc	92
8.1.3	PDF Utilities	93
8.2	File utilities	93
8.2.1	Diff	93
8.2.2	Grep	94
8.2.3	Hexdump	94
8.2.4	Readelf	94
8.3	Software Version Control Systems	95
8.3.1	Using Git and GitHub	95
8.3.2	Mercurial Version Control	97
8.4	Mathematical Tools	100
8.5	SageMath	100
8.5.1	Octave	100
8.5.2	gnuplot	102
8.6	Circuit Simulation with NGSpice	103
8.6.1	Defining a circuit file for simulation	103
8.6.2	Example Circuit Simulation	105
8.7	Ham Radio Applications for the Raspberry Pi	106
9	Troubleshooting Tips for the Raspberry Pi	107
Closing Note		110

1 Introduction

The respected computer scientists Edsger Dijkstra once remarked that “computer science is no more about computers than astronomy is about telescopes.” Even so, having access to a good educational computer is certainly an asset in the study of computer science. As it turns out, the Raspberry Pi is a wonderful little computer for experimenting with many basic concepts in computer science. The body of knowledge in computer science includes a wide range of topics such as programming languages, data structures and algorithms, digital logic, computer organization, operating systems, networking and the web, the Internet of Things (IoT), and artificial intelligence. The chapters of this book are organized around many key topics which are explored with practical examples and exercises which can all be performed on a recent model of the Raspberry Pi.

Computer science includes many abstract and theoretical concepts, including topics from discrete math, algebra, and calculus. While such theoretical foundations are essential for researchers, there are many concepts in computer science that are quite accessible to the keen student or hobbyist. This book is written for such an audience, containing many practical examples for those with only a modest mathematical background. Furthermore, this book also serves as a gentle introduction to the configuration and usage of Linux.

1.1 The Raspberry Pi

The Raspberry Pi is a nifty, new, little computer about the size of a deck of cards and capable of running a full Linux desktop operating system while consuming only modest power. It includes USB ports for connecting a keyboard and mouse along with a variety of other peripherals, an ethernet adapter, and HDMI monitor connections. The Raspberry Pi was originally constructed for hobbyists, but has found fruitful applications in a variety of areas including education, [home automation](#), [industrial applications](#), and as an appropriate technology for use in [schools in the majority world](#). It is manufactured to comply with RoHS (Restriction of Hazardous Substances) directives and relies on a single microSD card for its storage. It runs a variant of Linux called the Raspberry Pi OS, and supports a wide variety of open source software, including a plethora of educational programs. Moreover, it can be purchased at a modest price.

This guide was written primarily with engineering and computer science students in mind, but it will be of interest to others who have a keen interest in learning more about programming and technical computing.

This document is provided *as-is* for the purpose of providing an introduction to various concepts in computer science using the Raspberry Pi. The author specifically disclaims all warranties, express or implied including, but not limited to, implied warranties of merchantability and fitness

for a particular purpose.

1.2 Initial Setup of the Raspberry Pi

Insert an SD card with the Raspberry Pi OS in the Raspberry Pi and apply power. When you first boot the Raspberry Pi it will be running the PiXeL desktop environment (a modified version of the LXDE desktop environment) with a friendly menu-driven interface. Upon the first boot, a dialogue box will appear guiding you through an initial setup. Follow the prompts to configure the country, language, time zone, and keyboard. Change the password as prompted (the default password is “raspberry” which should be changed). Configure your Wi-Fi settings and select the option to “Update Software” (note that this can take a very long time when you first setup the Raspberry Pi).

1.2.1 Getting Started with the Command Line

There is also a command-line based version of the operating system called **Raspberry Pi OS Lite**. This version of the OS boots directly to a command line. The Raspberry Pi OS Lite consumes less power than the regular Raspberry Pi OS running a desktop environment and can be used on older models of the Raspberry Pi with less RAM.

When running the Desktop OS, the easiest way to access the command line is through the *Terminal* app, but it can also be accessed using a *virtual console* by pressing CTRL+ALT+F1 which will enter a full screen terminal. If a login prompt appears, you can login using the default username `pi` and default password `raspberry`. One can return to the desktop from a virtual console by pressing CTRL+ALT+F7. Additional virtual consoles can be independently accessed using CTRL+ALT along with the keys F2 through F6.

1.2.1.1 The Shell Once you enter the command line, you will be running in a Linux *shell*. In simple terms, a shell is a command interpreter which provides a rich set of commands which can be used to execute programs and interface with the operating system. The Raspberry Pi OS uses the **Bash** (Bourne Again Shell) by default, a shell based on an older shell called the *Bourne Shell*. BASH is popular among users of Linux and features automatic command line completion using the tab key and can be used to create programs called *shell scripts*.

There are variety of different Linux shells that can be used. As indicated, the default Linux shell is the **Bash** shell, but other shells are also available. Each shell has its own features and options. For example, to switch the default shell from **Bash** to the **Z shell (zsh)**, type the following

```
sudo apt install zsh -y  
chsh -s /bin/zsh
```

After issuing these command, logout and then back in and you should now be running with **zsh**. Various configuration options can be set inside a file named `.zshrc` located within your home folder.

1.2.1.2 Shell commands

Command	Description
<code>cd directory</code>	Changes the current working directory to <i>directory</i>
<code>pwd</code>	Displays the name of the current working directory
<code>mkdir <directory></code>	Create a new directory called <i><directory></i>
<code>rmdir <directory></code>	Removes the directory called <i><directory></i>
<code>ls</code>	Display a list of files in the current directory
<code>cp <f1> <f2></code>	Copy a file from the source <i><f1></i> to the destination <i><f2></i>
<code>rm <filename></code>	Remove a file <i><filename></i>
<code>mv <f1> <f2></code>	Move a file from <i><f1></i> to <i><f2></i>
<code>ftp <host></code>	Transfer files to and from <i><host></i>
<code>diff <f1> <f2></code>	Display the difference between file <i><f1></i> and file <i><f2></i>
<code>cat <filename></code>	Displays the contents of <i><filename></i>
<code>more <filename></code>	Displays contents of <i><filename></i> pausing when screen is filled
<code>head <filename></code>	Displays contents at the beginning of <i><filename></i>
<code>tail <filename></code>	Displays contents at the end of <i><filename></i>
<code>wc <filename></code>	Displays the number of newlines, words, and bytes in <i><filename></i>

In addition to the commands described above there are numerous other commands some of which are summarized in the following table:

Command	Description
<code>clear</code>	Clears the terminal display.
<code>date</code>	Display the current date and time.
<code>df</code>	Report the amount of free disk space available
<code>du</code>	Report the amount of disk usage

Command	Description
<code>find path conditions</code>	Utility for finding files
<code>locate search-string</code>	Utility for locating files
<code>free</code>	Report information about memory usage
<code>hostname</code>	Display the name of the current host system
<code>talk user</code>	Talk to user on your machine or on another host
<code>uptime</code>	Display the current time stats
<code>who</code>	Display the users currently logged-on the system
<code>whoami</code>	Display your User ID

These commands only represent a portion of the user commands available in a Linux shell. An on-line manual referred to as the **man** (manual) pages provides help on the many commands and programs that can be called from the shell. The syntax for invoking the **man** utility is as follows:

```
man command-name
```

The information regarding the specified `command-name` will then be displayed on the screen. To search for a keyword in the man pages, type:

```
man -K keyword
```

To find out more about the **man** utility simply type

```
man man
```

from the command prompt. There are several basic system commands which are commonly used. Many of the commands listed below have additional options which may be invoked. Consult the **man** pages for more information.

1.3 Text Editors

Text editors are used to edit system configuration files as well as program source files. There are several simple text editors available with Linux distributions, including with the Raspberry Pi. These include a simple text editor named **nano** and some classic text editors like **Emacs** and **vim**. These editors can be used within a simple command line environment.

To get started, a very brief summary of the basic commands for these text editors are summarized below. In particular, the **Emacs** and **vim** editors possess more powerful and advanced features which

the reader may wish to learn once the basics have been mastered. For more information, consult the [man](#) pages or the web.

1.3.1 Nano

The [nano](#) editor is a very basic command line editor. It is quite limited but also very easy to use. It is also included my default in all Raspberry Pi software distributions. To edit a file using [nano](#), type:

```
nano filename.txt
```

where [filename.txt](#) is the name of the file you want to edit. If the file does not exist, it will be created. Other editors are available in the command line environment, like [emacs](#) and [vi](#), which are simple but far more powerful alternatives to [nano](#).

1.3.2 Emacs

One editor you may wish to consider using is Emacs, although Emacs is much more than just an editor. When running under X Windows, X-Emacs provides a toolbar menu of the various commands. However, when running in a text terminal the user must use various control keys to issue the editor commands.

The list of basic commands for Emacs is summarized in the table below:

Command	Description
Ctrl-x Ctrl-s	Save
Ctrl-x Ctrl-c	Exit
Ctrl-h	Help
Ctrl-h t	Emacs tutorial
Del	Delete preceding character
Ctrl-g	Cancel Command
Ctrl-x	Search (in the forward direction)
Ctrl-x	Move down one page
Ctrl-x u	Undo last change
Esc <	Move to the beginning of the file
Esc >	Move to the end of the file

Use the help command (`ctrl-h`) to list additional commands or for more information.

1.3.3 Vim

Another popular editor is the **vim** editor. To edit a file using vim, type:

```
vi filename
```

Vim has two basic modes of operation: *command mode* and *input mode*. Text may only be entered while in input mode. The user must hit the escape key to exit input mode and enter command mode before any editing commands are entered. The list of basic commands is summarized below.

Command	Description
<code>↑↓→←</code>	Move up, down, right, left (if terminal emulation is configured right)
<code>0</code>	Move to the start of the current line
<code>\$</code>	Move to the end of the current line
escape key	Ends input mode and returns to command mode.
<code>a</code>	Appends text after current position and enters input mode
<code>A</code>	Appends text at the end of the current line and enter input mode
<code>i</code>	Inserts text at current position and enters input mode
<code>o</code>	Inserts a line below the current line and enters input mode
<code>rc</code>	Replaces the current character with c
<code>R</code>	Replaces the current text with the text you type after R
<code>x</code>	Deletes current character
<code>dw</code>	Deletes a word
<code>dd</code>	Deletes current line of text
<code>:q</code>	Quits and exits from vi (only if no changes were made)
<code>:q!</code>	Quit and exits without saving
<code>:w</code>	Saves the file you are editing
<code>:wq</code>	Saves the file you are editing and quits
<code>/pattern</code>	Searches the file for text matching pattern

1.4 Configuring the Raspberry Pi OS

The Raspberry Pi OS comes with a utility called `raspi-config` that can be used to configure a wide variety of settings and services. To run this utility, type:

```
sudo raspi-config
```

This will launch the Raspberry Pi configuration utility within the terminal with a main menu like the one shown below.

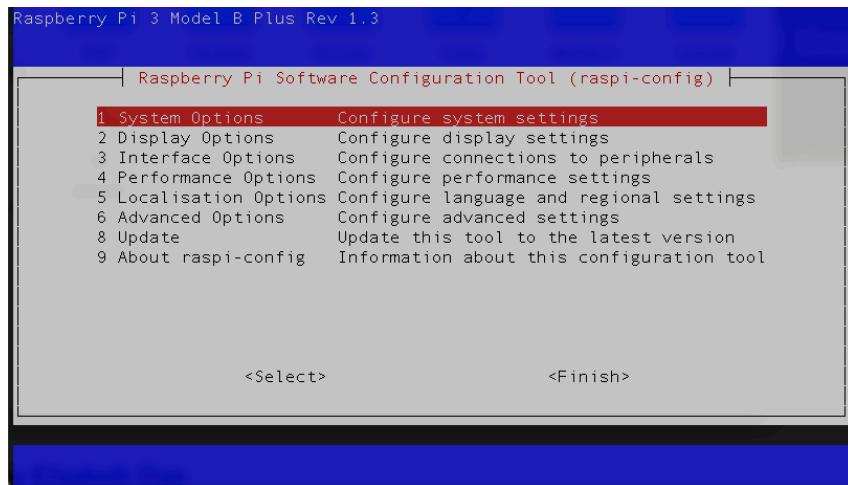


Figure 1: Raspberry Pi configuration utility

The Raspberry Pi configuration utility can be used to enable the camera interface as well as serial, I2C, and SPI communications. It also includes an option to boot directly to the command line rather than the desktop.

1.5 Connecting Remotely to the Raspberry Pi

It is possible to run the Raspberry Pi *headless*, without a screen, keyboard, or mouse. This is often the case when using the Raspberry Pi in an embedded or IoT (Internet of Things) configuration. The following subsections describe how to connect to your Raspberry Pi *remotely* using one of the following options:

- using a USB-to-TTL serial cable
- using SSH over an Ethernet or Wi-Fi connection

Some models of the Raspberry Pi can be connected using a USB cable. This works by enabling *USB Gadget Mode*, which allows a USB port to present itself as a variety of different types of devices. This

is known to work with the Raspberry Pi Zero and not with most other models. The two approaches described below should work with all models of the Raspberry Pi.

1.5.1 Connecting using a USB-to-TTL serial cable

A USB-to-TTL serial console cable can allow you to log into your Raspberry Pi from another computer using a serial link. A serial port console must be first enabled on the Raspberry Pi using [raspi-config](#). To enable the serial console, type:

```
sudo raspi-config
```

The console service can then be enabled in the [raspi-config](#) menu under *Interface Options → serial*.

Next, a suitable USB-to-TTL serial console cable must be connected to the proper pins on the GPIO port. With the power off, start connecting the cable wires to the GPIO pins on the Raspberry Pi as follows:

- black lead to GND on the GPIO port
- white lead to TXD on the GPIO port
- green lead to RXD on the GPIO port
- Leave the red lead disconnected!

A diagram showing the serial cable connections to the GPIO port is shown below.

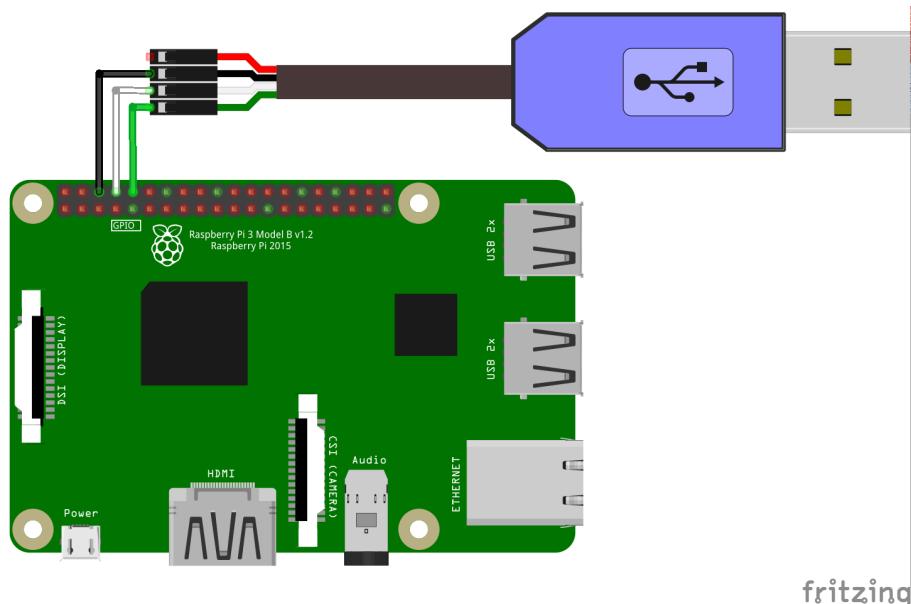


Figure 2: USB-to-TTL serial cable GPIO wiring details

Note: Ensure that the Raspberry Pi is powered **off** before adding or removing the USB serial cable pins from the GPIO port. In general, always power down before connecting or disconnecting circuits to the GPIO port pins!

Next, connect the USB-to-TTL serial console cable to a USB port on a Linux workstation. Type the following command in a terminal window on the workstation:

```
screen /dev/ttyUSB0 115200
```

where `/dev/ttyUSB0` is the serial port device and 115200 is the bit rate (or baud rate) of the serial connection. The `screen` program is a serial communications program that can use the USB port to communicate with the Pi. Power up the Pi and, after a few moments, hit enter a few times. It may take a few moments for the serial connection to “sync,” and you may initially see “gibberish” on the screen. If the connection has trouble syncing, try re-inserting the USB cable or hitting return a few more times. A login prompt should eventually appear. Log into your Raspberry Pi and confirm that the serial connection is working.

Note: Here the serial port device is assumed to be `/dev/ttyUSB0`, however the USB device name may be different on your system. While Linux includes all necessary drivers to use the USB-TTL serial cable, OSX or Windows may require installing extra drivers. Moreover, OSX and Windows will require using a different serial communication program.

1.5.2 Connecting over Ethernet or Wi-Fi

The previous two approaches allow remote connections to the Raspberry Pi, but the remote distance is limited to the length of the serial or USB cables being used. It is also possible to use a network connection which enables secure, high speed remote connections over much greater distances. However, you need to know the IP address of the Raspberry Pi in order to connect.

We will be using SSH (secure shell) over a network connection. Before you can connect to your Raspberry Pi using SSH, you need to enable the **secure shell server** in `raspi-config`. The SSH service can be enabled in the `raspi-config` menu under *Interface Options* → *SSH*. Once the server is running, users can log in remotely using the **SSH** protocol.

Before we can connect, we need to know the IP address of the Raspberry Pi. The `ifconfig` program allows you to query the status of your WiFi or Ethernet network as follows:

```
ifconfig wlan0
```

If the Wi-Fi network is running, an IP address should be assigned to the `wlan0` adapter. For example, the first two lines should look like this: :: small

```
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
      inet 10.1.2.3 netmask 255.255.192.0 broadcast 153.106.63.255
```

::: The decimal dotted number beside the `inet` label is your IP address. Make a note of the IP address (in the example above, the IP address is `10.1.2.3`). Note that if you were to use Ethernet, the IP address assigned can be queried by typing:

```
ifconfig eth0
```

Now that we have determined the Wi-Fi or Ethernet IP address of your Raspberry Pi we can connect to a shell from a remote computer using SSH. The SSH client is normally included with Linux and OSX operating systems, but for Windows you will need to install an SSH client, such as *Putty*.

To make an ssh connection, open a terminal window on a remote Raspberry Pi or another Linux computer and type:

```
ssh pi@10.1.2.3
```

where `pi` is the username and `10.1.2.3` is the IP address of the Raspberry Pi. If everything is working, you should be able to log in successfully using the username and password you configured. If your remote computer is a Windows machine, SSH operation can be realized using the [Putty utility](#).

You can also use SSH to transfer files between your computer and the Raspberry Pi using a tool related to SSH called *secure copy* (SCP). To us SCP, type:

```
scp filename pi@10.1.2.3:
```

This will transfer a local file named `filename` to the home folder of the `pi` user on the Raspberry Pi. Note that you will be prompted for the password of the user `pi`.

1.5.2.1 What's My IP Address? Before you can use ssh to remotely log into your Raspberry Pi, you need to know your Wi-Fi or Ethernet IP address. However, to know your IP address, you normally need to log into your Raspberry Pi. This may seem like a circular situation, but there are ways to determine your IP address.

The obvious way to discover your IP address is to plug in a monitor and keyboard and check the IP address using the `ifconfig` command as described above. This is fine when a monitor is available but may not be practical if you are running “headless,” i.e. without a screen or keyboard.

The Raspberry Pi runs a multicast DNS service by default using the Avahi service, so on certain local networks you might be able to find your Raspberry Pi by typing:

```
ping hostname.local
```

where hostname is the hostname that you configured earlier for your Raspberry Pi. If the ping command finds your Raspberry Pi it will display statistics about the ping packets exchanged along with the IP address of your Raspberry Pi. Note that this service will only work if you are on the same local network as your Raspberry Pi (note this may not work on all networks).

Another clever option is to program the Raspberry Pi to use sound to “speak” the IP address through the audio jack when it boots up. To do this, insert a speaker into the audio jack and carefully adjust the audio levels appropriately. The master audio level output can be set as follows (use the up/down arrow keys to set the volume and press Esc to exit):

```
alsamixer
```

Adjust and exit `alsamixer` to mid-volume and then check the audio output by running this test program (if using headphones, be sure to distance them from your ear till you adjust the audio levels):

```
speaker-test -t wav --nloops 3
```

After setting a comfortable audio level, install a text-to-speech program as follows:

```
sudo apt install festival
```

We can use festival to “speak” the IP address. Give it a try by typing the following:

```
hostname -I | festival --tts
```

Next, create a systemd startup file to run this instruction at startup after the network is connected and the audio service is running. You can edit this startup file from the command line using a simple text editor called nano as follows:

```
sudo nano /etc/systemd/system/say-ip-address.service
```

Enter the following text into this file:

```
[Unit]
Description=Speak IP address at boot time
After=NetworkManager-wait-online.service sound.target
[Service]
User=username
ExecStartPre=sleep 15
ExecStart=/bin/sh -c "hostname -I|awk '{print \"\$1\"}'|festival --
  tts"
[Install]
WantedBy=multi-user.target
```

Replace `username` with the username you created at setup and save the file by typing `ctrl-x` and select “yes” to save. Note that it was found that the `sleep 15` command was needed for additional

time to ensure the network is fully up and running with an IP address assigned, so this may require some further experimentation. Once everything is entered, enable the systemd file as follows:

```
sudo systemctl enable say-ip-address.service
```

Next, set the default audio output by editing a sound configuration file as follows:

```
sudo nano /etc/asound.conf
```

Enter the following text, then type ctrl-x and save the file:

```
defaults.pcm.card 1  
defaultsctl.card 1
```

If you have a pair of headphones, test the service by plugging them into the audio jack and typing the following (again use caution in case the volume is higher than expected):

```
sudo systemctl start say-ip-address.service
```

You should hear the IP address being spoken. This service should now run each time the Raspberry Pi is booted and a network is present. Reboot the Raspberry Pi to test that it is working as expected.

1.6 Remote Editing with Vscode

Using text editors like [nano](#) on the Raspberry Pi can be cumbersome. Another option is to make use of a more elaborate editor on a desktop or laptop and *remotely* edit files on the Raspberry Pi. One such program for doing this is [Visual Studio Code](#), or simply [vscode](#). Vscode is a general-purpose editor that provides a variety of useful features and plugins for editing programs.

While vscode can be installed locally on the Raspberry Pi, it can also be used for remote editing on desktop or laptop (running Windows, OSX, and Linux). After launching vscode, perform the following steps:

- type [`ctrl-shift-x`](#) to bring up the vscode extensions
- type “Remote SSH” in the search box
- select and install the *Remote-SSH extension*

After the installation is complete, you should see a new green “connect” icon appear in the bottom left corner of vscode. Click the green connect icon and select “Remote SSH: Connect current window to host”. Next, enter the SSH connection details, for example, [`pi@10.1.2.3`](#) where `pi` is your user-name and `10.1.2.3` is IP address of your Raspberry Pi. Finally, you will be prompted to enter your Raspberry Pi password and then you will need to wait briefly as vscode initializes and establishes a connection.

Once the setup and initialization is complete, click on the link to “open folder” (or select File→Open Folder) and your home folder on the remote Raspberry Pi should appear! Select the folder you wish to work within and click “OK.” This will become your current “working folder” on the Raspberry Pi. A list of files should appear on the left where you can select and open source files for editing. Vscode provides a quick and convenient way to ssh into the Raspberry Pi by typing `crtl+`` which opens a new remote terminal window to the Pi. In the terminal window you can compile and run the programs you edit.

Note how much more delightful it is to use an advanced editor over SSH rather than editing code locally with the `nano` editor.

1.7 Proper Shutdown

A final note regarding proper shutdown of the Raspberry Pi. One should never just remove power from the Raspberry Pi since this can lead to corruption of the SD card. In order to perform a proper shutdown of the Raspberry Pi, type the following command:

```
sudo halt
```

Once Linux has shutdown and the green activity LED has stopped flashing, you may unplug the power supply. Alternately, you may reboot the Raspberry Pi from the command line by typing:

```
sudo reboot
```

2 Programming Languages and the Raspberry Pi

The first large-scale electronic digital computer, called the ENIAC, was completed in 1946. It weighed 30 tons, occupied a 30x50 foot space, and was powered by thousands of vacuum tubes. The ENIAC was programmed by physically wiring interconnections between various components.

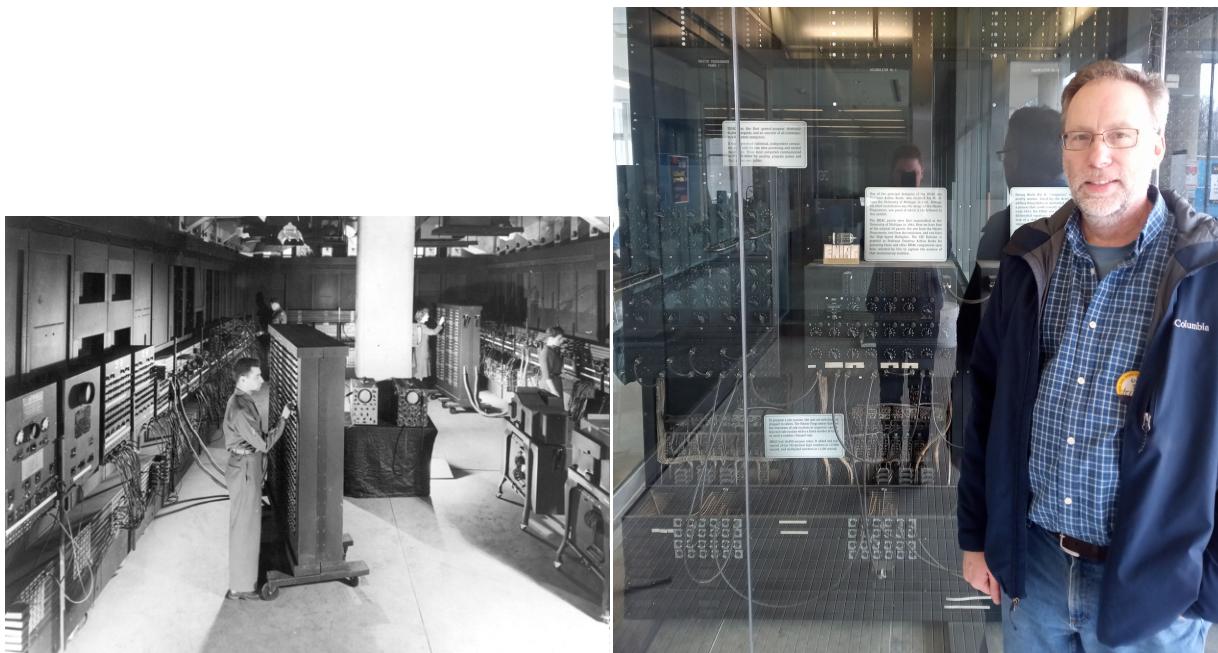


Figure 3: The ENIAC in a US Army Photo, 1946 (public domain) and a photo of the author posing next to a section of the original ENIAC on display at the University of Michigan Computer Science Department

Later, innovations in computer design led to the *Von Neuman architecture*, which enabled programs to be stored in memory. Computers could then be programmed without altering the physical wiring of a computer by using native machine instructions.

2.1 Running an Assembly Language Program

Assembly language is a *machine-specific* programming language with a one-to-one correspondence between its statements and the computer's native machine language. Programs written in Assembly Language are translated into the native *Machine Language* using an *Assembler*. Assembly language programming is difficult since it takes longer to write, is more difficult to code, debug, and maintain. Assembly Language is sometimes used in computer science education when learning about computer

organization and architecture.. Furthermore, assembly language can be useful to bypasses restrictions and limitations of higher level languages and for high-speed or time-critical applications that require direct communication with hardware.

The Raspberry Pi uses an ARM processor which can be programmed at a low-level using ARM assembly language. An example of ARM assembly is given below and can be entered using a plain text editor such as [nano](#) as follows:

```
nano program1.s
```

Using nano (or another editor of your choosing), enter the following ARM assembly code program:

```
.data
message: .ascii  "Hello World!\nAssembled and run on a Raspberry Pi.\n"
.text
.global _start
_start:
    mov r0, #0
    ldr r1, =message
    mov r2, #49
    mov r7, #4
    svc 0
    mov    r0, #0
    mov    r7, #1
    svc 0
```

Next, assemble, link, and run the program by typing the following sequence of commands:

```
as -o program1.o program1.s
ld -o program1 program1.o
./program1
```

If everything was entered correctly, your program should run and you should see a message displayed. Note how many primitive instructions are required to display a simple message in assembly language.

2.2 Higher Level Programming Languages

Grace Hopper is a famous pioneer in computer science who is often referred to as the person “who taught computers to talk.” Her pioneering work led to the development of the *compiler*, a program that translates higher level instructions into the primitive machine code that computers can execute directly. Her work contributed to the development of the COBOL programming language and blazed a trail for the development of future programming languages. For this reason, Grace Hopper has sometimes been referred to as “Amazing Grace.”



Figure 4: Grace Hopper, U.S. Navy, 1984 (public domain)

The Raspberry Pi repositories include support for COBOL, as well as a rich set of more modern programming languages such as Python, C, C++, and Java. It has support for more niche and legacy programming languages.

Generally, the paradigms for programming languages fall into three general categories:

- procedural programming languages
- object oriented programming languages
- functional programming languages

2.2.1 The Python Programming Language

Python was invented in the early 1990's by Guido van Rossum. Python can be written using either a procedural or object oriented paradigm. It is an open source project that is widely available, uses simple syntax and includes rich libraries and offers a variety of programming tools. Python source code is run on a *virtual machine* which translates code into the specific machine-code executed by the processor. Because Python is interpreted by a virtual machine, it is *platform independent*.

The Raspberry Pi should have Python installed by default and can run Python programs directly from the command line. To enter a program, you first need a plain text editor. If you are using a desk-

top environment, there is a friendly, graphical, integrated development environment (IDE) for Python suitable for beginners called [Thonny](#). To install [Thonny](#), type:

```
sudo apt install thonny
```

For more advanced users in a graphical environment, the [vscode](#) program provides an excellent editor for coding in a variety of different languages, including Python. As described earlier, [vscode](#) can also be run to edit files remotely.

If you are using the command line, you can use any of the command line editors described in the preceding sections, including [vi](#), [emacs](#), or [nano](#). For example, to edit a Python source file called [hello.py](#), type the following:

```
nano hello.py
```

Next, enter the following code into the source file:

```
name = input('What is your name? ')
print('Hi', name, 'welcome to the Raspberry Pi!')
print('Good Bye')
```

Next, save and exit [nano](#) and run the file by typing:

```
python3 hello.py
```

The program should run as expected.

2.2.1.1 Plotting in Python **Matplotlib** is a nifty Python plotting library which produces publication quality figures including plots, histograms, power spectra, bar charts, error charts, scatter plots, and more. Of course, using Matplotlib presupposes a graphical desktop environment in order to display the plots. To install [matplotlib](#), type:

```
sudo apt install python3-matplotlib
```

Once the library is installed, it can then be imported and used in a Python program. For example, the following code takes two lists and plots them in an x-y scatter chart shown in the figure below.

```
import matplotlib.pyplot as plt

xdata = [1, 2, 3, 4, 5, 6, 7, 8]
ydata = [1, 4, 9, 16, 25, 36, 49, 64]

plt.plot(xdata, ydata)
plt.xlabel('some numbers')
plt.ylabel('some squares')
plt.show()
```

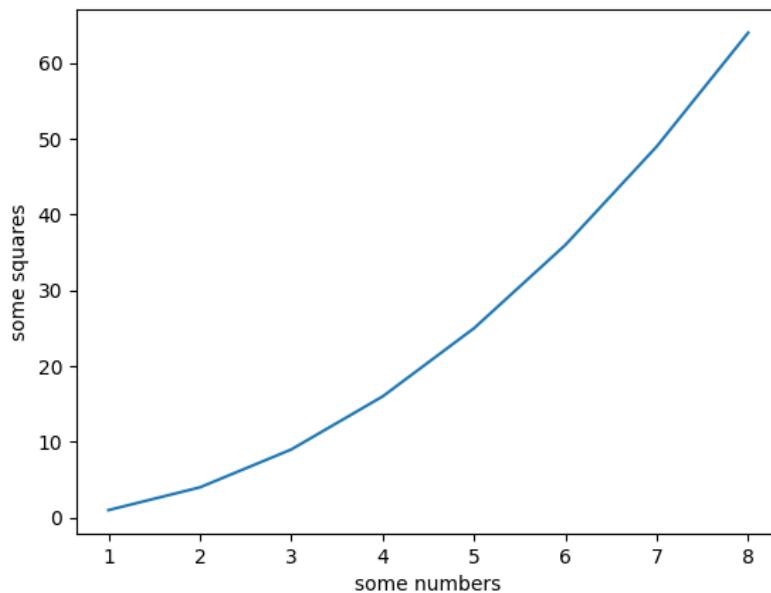


Figure 5: Example of a simple matplotlib plot

Further examples of matplotlib in action can be found on the [matplotlib documentation pages](#).

2.2.1.2 Setting up a Python Virtual Environment For most situation, one can install Python globally with all libraries and dependencies. However, there are times when tinkering with Python that you want to isolate all the settings from your system-wide settings. This may be the case if you are developing libraries or experimenting with bleeding-edge Python libraries and releases. In this case, you can install Python in a *virtual environment* that is isolated from your system-wide libraries and settings. In essence, a Python *virtual environment* gives you a “sandbox” in which you can adjust settings, libraries, and Python versions without impacting your any other settings. According to the [Python documentation](#), “Each virtual environment has its own Python binary (allowing creation of environments with various Python versions) and can have its own independent set of installed Python packages...”

To install the tools for a Python virtual environment, type:

```
pip3 install virtualenv
```

To create a new environment, issue the following command:

```
python -m venv NAME
```

where `NAME` is a name for the new virtual environment. To “activate” this new virtual environment,

type:

```
source NAME/bin/activate
```

Once a virtual environment is activated, you can install any packages or libraries you wish, and these will be isolated within the virtual environment. To leave a virtual environment, type:

```
deactivate
```

2.3 Compiling and Running a C/C++ Program

Linux has a variety of tools to support software development in both C and C++. In fact, the Linux operating system itself is written in C. The C programming language is a procedural language, and C++ builds on C to provide support for object oriented programming. The compilers which we will use under Linux are the GNU C Compiler (`gcc`) and the GNU C++ compiler (`g++`). To ensure the GNU C/C++ compiler tools are installed, type:

```
sudo apt install gcc g++ gdb build-essential
```

For example, to enter a simple C program named `hello.c` using the `nano` editor, type the following:

```
nano hello.py
```

Using the editor, enter the following code into the source file:

```
/* A Raspberry Pi C program */
#include <stdio.h>

int main(void)
{
    printf("Hello world.\n");
    printf("Compiled and run on a Raspberry Pi.\n");
    return 0;
}
```

Save and exit the editor. To compile your source code type the following at the prompt in a terminal window. For example, to compile the `hello.c` program above, you can enter:

```
gcc -Wall -o hello hello.c
```

The `gcc` compiler has numerous other command line options which you may use. For more information consult the man pages. Note that the C++ compiler can be invoked by using `g++` in place of `gcc`.

To run the compiled program in the current working directory, do not forget to specify a . / in front of the program name to specify the path as the current directory. For example, to run the `hello.c` program after compiling it as ini, type:

```
.\hello
```

If no output filename was provided to the compiler the default output filename will be `a.out`.

2.3.1 Using a C Debugger

Debugging is frequently part of the process of programming. There are several techniques to debug code. Sometimes debugging may be accomplished by sprinkling `printf` statements throughout your code to display the state of your program and variables as it executes. Another way to debug code is to use a special debugging tool that allows you to see what is going on inside your program while it executes. The `gcc` compiler has a powerful debugger called `gdb` (the GNU debugger). Most integrated development environments (IDEs) provide a friendly interface for using the debugger. In this tutorial we will look at using the debugger from the command line.

To demonstrate `gdb` in action, create the sample program as shown below to perform simple calculations based on user input.

```
/* Sample C program
An Introductory Guide to Using Linux with the Raspberry Pi
This program multiplies 3 numbers provided by the
user and prints the result to the display */
#include <stdio.h>

int main(void)
{
    float x,y,z;

    printf("\nWhat is the first number? ");
    scanf("%f",&x);
    printf("\nWhat is the second number? ");
    scanf("%f",&y);
    printf("\nWhat is the third number? ");
    scanf("%f",&z);
    printf("\n%f * %f * %f is equal to %f\n",x,y,z,(x*y*z));
    return 0;
}
```

Practice using the debugger with the code you wrote in the previous step. Load the source file and then re-compile it using the following command:

```
gcc -Wall -g multiply.c -o multiply
```

Note that the extra `-g` option tells the gcc compiler to include information in the output file for use by the debugger. To start the GNU debugger, type:

```
gdb multiply
```

At the prompt in the debugging window you can now enter various commands. For example, to set a breakpoint for the start of the `main()` function by type:

```
break main
```

A breakpoint stops program execution allowing you trace your program step-by-step. To begin running your program type:

```
run
```

The program will halt at the first line in your program which will be high-lighted in the source file. To display the contents of the `x`, `y` and `z` variables type:

```
display x  
display y  
display z
```

Note that before these variables are initialized they contain “garbage” values. This will show why it is always a good idea to initialize variables to a known value. You may begin single-stepping through each line of your code by typing “`s`” (for “step”) in the debugging window.

When your program ends, you may quit the debugger by typing:

```
quit
```

The debugger includes several additional commands that are useful when debugging code. Some of the `gdb` commands which can be entered after the (`gdb`) prompt are listed in the table below:

gdb Command	Description
<code>run</code>	Start running a program
<code>break function</code>	Set a breakpoint at the start of a function
<code>break line number</code>	Set a breakpoint at a line number
<code>info break</code>	list all breakpoints
<code>disable breakpoint</code>	disables a breakpoint
<code>enable breakpoint</code>	enables a breakpoint
<code>continue</code>	resume running the program

gdb Command	Description
<code>list</code>	list the next source lines
<code>next</code>	execute the next statement
<code>step</code>	same as next, but step into a function
<code>print variable</code>	print the value of a variable
<code>display variable</code>	display the value of a variable after each step
<code>set variable=value</code>	assign a new value to a variable
<code>help</code>	display a list of gdb commands
<code>help command</code>	display help on a specific gdb command
<code>quit</code>	quit gdb

The debugger can also be used to examine core dumps when segmentation faults occur. For more information on using the debugger, type `help` at the debug prompt or type `man gdb` for more information.

2.3.1.1 Strace Another debugging tool is the `strace` program which can be used to trace systems calls made during the execution of a compiled program. For example, to trace the execution of the `hello` program in the previous section, type:

```
strace ./hello
```

2.3.2 Library Documentation

The `man` pages include documentation on most of the functions found in the various C Libraries. To display information on a C function type the following:

```
man -S3 function_name
```

where `function_name` is the name of the function you want documentation for. Documentation also includes the names of the `#include` header files that must be included for a given function.

2.3.3 Other Tools for C and C++

The `make` utility can be used to automatically determine which pieces of a large program need to be recompiled. Type `man make` for more information.

Another program to assist C programmers is [splint](#), a tool for statically checking C programs for security vulnerabilities and coding mistakes. To use [splint](#), it should be first installed as follows:

```
sudo apt install splint
```

Once it is successfully installed, it can be invoked as follows:

```
splint source.c
```

where `source.c` is the name of your C source file.

2.4 Compiling and Running Java Programs

It is also possible to develop and run Java programs using Linux on the Raspberry Pi. There are two different packages which can be installed: one provides the Java Runtime Environment (JRE) and another provides the Java Development Kit (JDK). The JRE just allows you to run Java programs, but the JDK enables one to compile and run Java programs.

To install the OpenJDK Java development kit, type the following:

```
sudo apt install default-jdk
```

Once this is installed you can compile a Java program. For example, enter the following simple Java program named `hello.java` using a plain text editor:

```
/* A Java program */
class Main {
    public static void main(String args[]) {
        System.out.println("Hello world.\n");
    }
}
```

Compile the program by typing the following at the prompt:

```
javac hello.java
```

where `myprogram.java` is the name of a Java source file. To run a Java program, type the following at the prompt:

```
java Main
```

where `Main` is the name of the class where the program begins.

2.5 Other Programming Languages

The Linux platform provides a plethora of open source developments tools and programming languages. The options range from fashionable programming languages to more niche languages as well as many historical programming languages. The following table lists some additional programming languages that are available as well as the name of the corresponding package in the `apt` repository.

Programming Language	Package Name(s)	Description
Fortran	<code>gfortran</code>	GNU Fortran project implementation of FORTRAN 2018
PHP	<code>php</code>	scripting language for web development
Nodejs	<code>nodejs,npm</code>	JavaScript runtime environment
Perl	<code>perl</code>	classic scripting language
C/C++	<code>gcc,g++</code>	GNU C/C++ compiler
Python	<code>python3</code>	Python version 3
Ruby	<code>ruby</code>	high-level, interpreted programming language
Java	<code>default-jdk</code>	OpenJDK Java compiler and virtual machine
Gambas	<code>gambas3</code>	object-oriented version of BASIC programming language
Mono	<code>mono-complete</code>	open-source .NET compatible software framework
Scheme	<code>mit-scheme</code>	dialect of the Lisp family of programming languages
Prolog	<code>swi-prolog</code>	classic logic programming language
Tcl	<code>tcl,tk</code>	Tool Command Language (pronounced “tickle”)

2.6 Comparing Runtime Efficiency of Different Programming Languages

One advantage of C is that it is *compiled* to machine code, a process by which the code is converted to native machine language (in the case of the Raspberry Pi, the native code for the ARM processor). In contrast, Python runs using an *interpreter* which translates the Python code, step-by-step, into the local machine code. As such, Python has additional overhead at run-time.

The difference in execution time can be demonstrated by running the same algorithm implemented in three different programming languages: C, Python, and Java.

2.6.1 Measuring execution time

The runtime of a program can then be determined using the special `time` utility. For example, the following command:

```
time sleep 3
```

returns the time taken for the command to execute. In this example, it will return something like the following:

```
sleep 3  0.00s user 0.00s system 0% cpu 3.002 total
```

This shows the `user` time (the number of CPU seconds spent in user mode), the `system` time (the number of CPU seconds spent in kernel mode), and the `cpu` time (the elapsed “wall clock” time).

The `time` utility can resolve runtimes on the order of a millisecond or so. Hence, comparing execution times requires running programs that require a non-trivial execution time, ie. much greater than a millisecond. For our runtime comparison, we will implement a classic numerical integration algorithm. The approach to numerical integration we will use is sometimes referred to as a *Riemann summation*. In this approach, the area under a function $f(x)$ over the interval $[a, b]$ can be approximated by summing a series of n rectangles. The amount of computation time (and the accuracy of the numerical integration) is directly related to the number of steps, n . Thus, an integral can be approximated by the following:

$$\int_b^a f(x) dx \approx \Delta x [f(a) + f(a + \Delta x) + f(a + 2\Delta x) + \dots + f(a + (n - 1)\Delta x)] = \sum_{i=0}^{n-1} f(x_i) \Delta x$$

where $\Delta x = \frac{b-a}{n}$. A plot illustrating this expression for $f(x) = \sqrt{x}$, $a = 1$, and $b = 5$ with $n = 10$ steps is shown below.

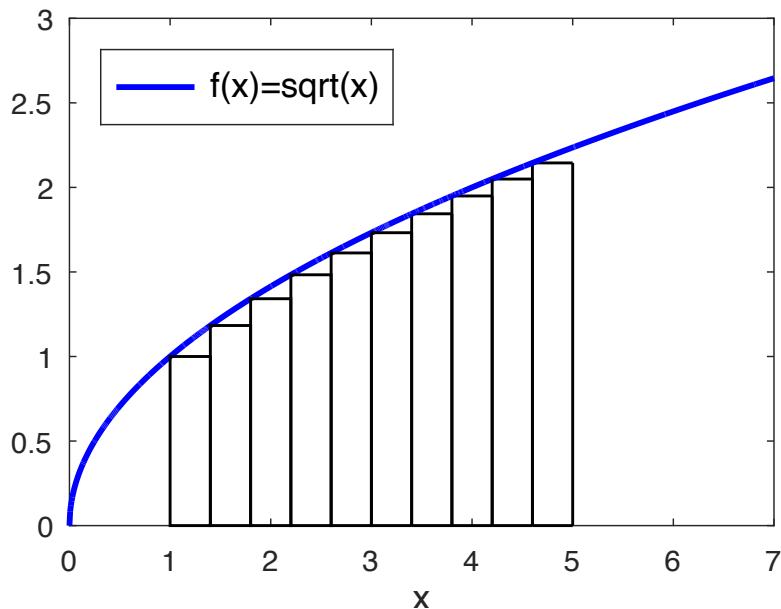


Figure 6: Illustration of integration using a Riemann sum of rectangles

The Riemann summation for the integral $\int_1^5 \sqrt{x} dx$ using $n = 1000000$ can be computed in Python as follows:

```
import math

NUM_STEPS = 1000000 # number of integration steps
a = 1 # lower limit of integration
b = 5 # upper limit of integration
DELTA_X = (b-a)/NUM_STEPS
sum = 0.0
x = a
for step in range(NUM_STEPS):
    y = math.sqrt(x)
    sum += y
    x += DELTA_X
integral = sum * DELTA_X
print(integral)
```

Thus, the runtime of the Python program can be determined as follows:

```
time python3 integral.py
```

where `integral.py` is the name of the Python program. The same program can be implemented in C as follows:

```
#include <math.h>
```

```
#include <stdio.h>
#define NUM_STEPS 1000000 // number of integration steps
#define a 1 // lower limit of integration
#define b 5 // upper limit of integration
#define DELTA_X (double)(b-a)/NUM_STEPS

int main()
{
    double sum = 0.0;
    double y, integral;
    double x = a;

    for (int step = 0; step<NUM_STEPS; step++) {
        y = sqrt(x);
        sum += y;
        x += DELTA_X;
    }
    integral = sum * DELTA_X;
    printf("%f\n", integral);
}
```

This program can be compiled and the runtime determined as follows:

```
gcc -Wall integral.c -lm
time ./a.out
```

The runtime performance of a C program can sometimes be further tweaked by making suggestions to the optimizer in the compiler. This can be done from the command line as follows:

```
gcc -Wall -Ofast ctest.c -lm -o ctest
```

The `-Ofast` command line parameter instructs the `gcc` compiler to optimize for speed. Running the program again with this compiler option should generally provide some slight improvements in speed.

Finally, the same program can be implemented in the Java programming language as follows:

```
import java.lang.Math;

class Integrate {
    public static void main(String args[])
    {
        final int NUM_STEPS = 1000000; // number of integration steps
        final int a = 1; // lower limit of integration
        final int b = 5; // upper limit of integration
        final double DELTA_X = (double)(b-a)/NUM_STEPS;

        double sum = 0.0;
        double y, integral;
        double x = a;
```

```

    for (int step = 0; step<NUM_STEPS; step++) {
        y = Math.sqrt(x);
        sum += y;
        x += DELTA_X;
    }
    integral = sum * DELTA_X;
    System.out.println(integral);
}
}

```

This program can be compiled and the runtime determined as follows:

```

javac integrate.java
time java Integrate

```

The algorithm used in this numerical integration is the same for all three of these programs. However, the C program runs many times faster than Python due to the fact that it is compiled and run using native ARM instructions. Java provide “just in time compilation” and so adds overhead but runs efficiently once the compilation is complete. As one can observe, runtime efficiency can sometimes vary by orders of magnitude for different programming languages. However, efficiency is just one consideration when selecting a programming language. Other considerations include maintainability, cost, interoperability, availability of tools, and community support.

2.6.2 Execution Speedup using Parallel Execution

One potential way to increase the speed of execution is to perform parallel computations using the multiprocessing features of the Linux operating system. Modern processors typically include multiple *cores*, which are independent processing units that can be scheduled by the operating system. The Raspberry Pi 3, 4, and 5 include four ARM cores that we can use to speedup the execution of programs by looking for opportunities to parallelize execution. In general, the *speedup*, S , that can be achieved from running a task in parallel is given by **Amdahl's law** as follows:

$$S = \frac{1}{(1-p) + \frac{p}{N}}$$

where p represents the proportion of execution time that can be parallelized and N represents the number of processors. Hence, with a four core processor ($N = 4$) and a program which can be fully parallelized ($s = 1$), the theoretical *speedup* should be four times. In reality, the theoretical speedup is never achieved due to the overhead of managing parallel processes.

As it turns out, numerical integration is an example of an *embarrassingly parallel* algorithm — one that is trivial to parallelize because the area can be broken down into separate chunks that are computed independently. For example, an integration over the range $[a, b]$ can be subdivided into four separate

summations as follows:

$$\sum_{i=0}^{n-1} f(x_i)\Delta x = \sum_{i=0}^x f(x_i)\Delta x + \sum_{i=x+1}^y f(x_i)\Delta x + \sum_{i=y+1}^z f(x_i)\Delta x + \sum_{i=z+1}^{n-1} f(x_i)\Delta x$$

such that $0 < x < y < z < n - 1$. Each of the above summations can run independently as parallel processes and the final answer can be obtained by summing the results of each.

2.6.2.1 Parallel Execution in Python Next, we will implement a program that computes these summations in parallel using each of the four cores on a Raspberry Pi. This can be implemented in Python using the [multiprocessing](#) library as follows:

```
from multiprocessing import Pool
import math

PROCESSES = 4 # number of processes
NUM_STEPS = 1000000 # total number of integration steps
a = 1 # lower limit of integration
b = 5 # upper limit of integration

def integrate_sqrt(a,b,n):
    ''' integrate sqrt(x) over[a,b] in n steps
    '''
    sum = 0.0
    x = a
    delta_x = (b-a)/n
    for step in range(n):
        y = math.sqrt(x)
        sum += y
        x += delta_x
    return sum * delta_x

if __name__ == '__main__':
    # start a pool of worker processes split over [a,b]
    with Pool(processes=PROCESSES) as pool:
        subrange = (b-a)/PROCESSES
        integrals=pool.starmap(integrate_sqrt,
                               zip([a+x*subrange for x in range(PROCESSES)],
                                    [a+x*subrange for x in range(1,PROCESSES+1)],
                                    [int(NUM_STEPS/PROCESSES)]*PROCESSES))
    print(sum(integrals))
```

Using the above code running on four processing cores available on our Raspberry Pi, we can time the execution as follows:

```
time python3 parallel_integral.py
```

We should observe a *speedup* of roughly four times over the prior Python implementation. Note that

the speedup will not be exactly four times faster due to the associated overhead in creating and managing four new processes.

2.6.2.2 Parallel Execution in C There are a variety of libraries that can be used to perform multiprocessing in Linux using the C programming language. In this example we will use the [OpenMP](#) library and the code is shown below:

```
#include <stdio.h>
#include <math.h>
#include <omp.h>

#define NUM_STEPS 10000000 // number of integration steps
#define a 1 // lower limit of integration
#define b 5 // upper limit of integration
#define DELTA_X (double)(b-a)/NUM_STEPS

int main()
{
    int step;
    double total_integral = 0.0;
    double sum;
    double y, integral;
    double x;

    printf("Number of cores: %d\n", omp_get_num_procs());
    #pragma omp parallel private(sum,y,integral,x,thread_id)
    {
        sum = 0.0;

        // parallelize this chunk of code
        #pragma omp for
        for (step = 0; step<NUM_STEPS; step++) {
            x = (double)a + (double)(b-a)*(double)(step)/NUM_STEPS;
            y = sqrt(x);
            sum += y;
        }
        integral = sum * DELTA_X;

        #pragma omp critical
        {
            total_integral += integral;
        }
    }
    printf("%f\n", total_integral);
    return 0;
}
```

This program can be compiled and the runtime determined as follows:

```
gcc -Wall -fopenmp integral.c -lm  
time ./a.out
```

Once again, we can observe that there is a speedup over the sequential C code that was given earlier.

2.6.2.3 Parallel Execution on a Graphics Processing Unit (GPU) Additional speedup can also be realized by utilizing a GPU (Graphical Processing Unit). The Raspberry Pi's Broadcom ARM processor includes an on-chip graphics processing unit (GPU). There are standard libraries for taking advantage of GPUs, such as [OpenCL](#) and [CUDA](#), but, as of the time of writing, programming support for the Raspberry Pi GPU is limited. Search the [Raspberry Pi forums](#) for more up-to-date information about programming with the GPU.

3 The Raspberry Pi Operating System

An operating system is software to control the hardware of a computer. The Raspberry Pi runs the Raspberry Pi OS, a variant of the free *Linux* operating system originally written by Linus Torvalds when he was a graduate student at the University of Helsinki in Finland. Linux has grown over time with support from developers around the world and continues to grow increasingly popular. It is distributed under the GNU general public license (GPL) and is freely available.

3.1 Introduction

In simple terms, an Operating System (or **OS**) makes computing power available to users by controlling the hardware resources. In general, an OS is responsible for the following functions:

- Process Management: process creation, scheduling, switching, and synchronization
- Memory Management: allocation, swapping, page management
- Input/Output (I/O) Management
- Other Support functions: accounting, monitoring, updates

3.2 Process Management

A process is a program in execution which is managed by the OS. Modern operating systems like Linux are *multitasking* which allows for the interleaved execution of two or more processes on a single processor.

The execution of different processes is controlled by the *Dispatcher*. The Dispatcher is a program that switches execution from one process to another. The main job of the Dispatcher is to ensure that the processor time is allocated appropriately to all the processes that are active.

A process is normally represented by a *Process Control Block* (PCB), a data structure within the OS that contains information such as the process ID, its state, and other information required by the OS.

Linx is not only a multitasking OS, it is also a *multiprocessor* OS with support for systems with multiple processors or cores.

3.2.1 Tools for Managing Processes

What is a process? The following table lists some terminal commands related to processes:

Command	Description
<code>ps</code>	Display a list of processes
<code>kill <pid></code>	Terminate the process with process ID <pid>
<code>htop</code>	Observe all processes running on the system

3.2.2 Example Program to fork a new process

A process which spawns another process is called a *parent* and the new process is called a *child*. In Linux, new processes may be started using one of several C library functions as summarized in the table below.

System call	Description
<code>system()</code>	executes a shell command string and waits for it to complete
<code>exec()</code>	replaces the current process with another
<code>fork()</code>	duplicates the current process

A sample C programming demonstrating creating a child process using a call to `fork` is shown below:

```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

int main()
{
    pid_t pid;      /* Process ID */
    pid = fork();   /* Call to fork a child */

    switch (pid) { /* check for parent or child process*/
    case -1:
        printf("Unable to create process");
        exit(1);
        break;
    case 0:
        printf("Child Process started...\n");
        break;
    default:
        printf("This is the Parent Process...\n");
        break;
    }
}
```

```
    }
    return 0;
}
```

3.2.3 Multithreaded programming

Multithreading refers to the ability of an OS to support multiple *threads* of execution within a single process. A *thread* is like a process that is independently scheduled but it shares the address space with other threads in a process.

To run a thread in Java, implement a class that extends the `Thread` class as illustrated in the following example code.

```
public class Main
{
    public static void main (String[] args) {
        HelloThread t1 = new HelloThread ("Bob");
        HelloThread t2 = new HelloThread ("Larry");
        t1.start ();
        t2.start ();
    }
}

class HelloThread extends Thread
{
    private String name;

    public HelloThread (String aName) {
        name = aName;
    }

    public void run () {
        for (int i = 0; i < 10; i++) {
            try
            {
                System.out.println ("Hello " + name);
                sleep (1000); // Wait one second
            }
            catch (InterruptedException exception)
            { }
        }
    }
}
```

3.3 Memory Management

Memory management is the task of dynamically subdividing memory to accommodate multiple processes.

3.4 Managing I/O

3.5 Controlling Input and Output

The Raspberry Pi includes a variety of inputs and outputs that are controlled by the operating system. These include a camera input, USB inputs, HDMI, and a bank of pins named the GPIO (General Purpose I/O) pins.

3.5.0.1 Controlling the GPIO pins includes a variety of GPIO (General Purpose I/O) pins that can be used for sensing or controlling the outside world. It also has a dedicated camera port.

You can view a diagram of the GPIO port and each the pins from the command line by typing the following command:

```
pinout
```

Before wiring the GPIO ports, remember to turn off power to the Raspberry Pi. It is always prudent to check your wiring to ensure you are accessing the correct pin and that there are no exposed leads touching each other as this could cause short circuits. It is important to make sure you are properly interfacing with external sensors and actuators to ensure you comply with the electrical limits of the GPIO ports.

You can test your GPIO connections using the `raspi-gpio` command line utility. To read all the GPIO pins, type the following:

```
raspi-gpio readall
```

This should display the state of all the GPIO pins. Note that there are several different pin numbering schemes that can be used with the Raspberry Pi, which can lead to some confusion. Note that we will be using the BCM numbering scheme. BCM represents the Broadcom SOC channel and reflects the numbering scheme used by the Broadcom ARM processor and the `raspi-gpio` utility.

3.6 Other Support Functions

3.6.1 Logfiles

A variety of log files are kept in the folder `/var/log` which keep track of the state of the system. Use the `tail` command to read the latest log entries as follows:

```
tail syslog
```

Viewing logfile is useful for troubleshooting.

3.6.2 Updating the Operating System

Part of keeping an operating system secure is to ensure that it is kept up to date with the latest updates and patches. Un-patched security issues can become attack vectors for malicious hackers. To view some current security vulnerabilities, visit the [NIST National Vulnerability Database](#). This is even more critical for a system that is connected to a network such as an IoT device. In fact, IoT devices must remain up-to-date after they are deployed in the field for the entire life-time of the product! To update the Raspberry Pi from the command line, use the following commands:

```
sudo apt update  
sudo apt full-upgrade
```

After performing an update, you may want to clear space by removing any packages that are obsolete or are no longer required:

```
sudo apt autoremove
```

Furthermore, you can clean any files left behind during the update process by using the following command:

```
sudo apt clean
```

3.6.2.1 Enabling Automatic Updates Rather than typing this command each day, one approach is to use a program to automate updates. There are several ways to accomplish this in Linux. One approach is to use a special package for automatic updates named `unattended-upgrades`. To install this package, type:

```
sudo apt install unattended-upgrades
```

Next, edit the configuration file as follows:

```
sudo nano /etc/apt/apt.conf.d/50unattended-upgrades
```

This will open a configuration file to setup unattended updates. Remove the double slashes (//) in front of the following lines in the configuration file to enable various automatic updates:

```
"origin=Debian,codename=${distro_codename}-updates";
"origin=Debian,codename=${distro_codename},label=Debian";
"origin=Debian,codename=${distro_codename},label=Debian-Security";
"origin=Debian,codename=${distro_codename}-security,label=Debian-Security"
;
```

To save the file, type ctrl-x and select “yes” to save and exit. Finally, ensure automatic updates are enabled by running:

```
sudo dpkg-reconfigure unattended-upgrades
```

At this point, automatic updates should be enabled to run daily.

3.6.3 Securing your Raspberry Pi

The Raspberry Pi is like any other networked computer — with all the wonderful possibilities as well as the threats. While physical security of a computing device is one consideration, once it is connected to a network a whole new set of vulnerabilities arise. For this reason, it is prudent to follow basic procedure for *system hardening* to reduce the vulnerability of your Raspberry Pi.

The first step is to disable all unnecessary services that are running. Each service expands the possible **attack surface** and exposes new vulnerabilities. This is particularly true of programs that open networking ports that are accessible by other computers. To obtain a list of all the services running on a Raspberry Pi, type:

```
systemctl --type=service --state=running
```

If any of these services are not required, it can be stopped and disabled from starting up on subsequent reboots as follows:

```
systemctl disable --now service-name
```

where `service-name` is the name of the service you wish to stop.

In addition, the Raspberry Pi OS comes with a whole set of tools and utilities to help secure your device through intrusion detection and prevention.

3.6.3.1 Virus Scanning There is an open source virus scanning program called **ClamAV** that can be used to scan your Raspberry Pi for viruses. To install `clamav`, type:

```
sudo apt-get install clamav
```

Before you can start scanning for viruses, an updated ClamAV virus signature database must be installed. A tool called `freshclam` is used to download and update this database. First, the `freshclam` configuration file must be setup properly by editing:

```
sudo nano /etc/clamav/freshclam.conf
```

Once the configuration file is in place, `frashclam` can be run by typing:

```
sudo freshclam
```

Once the virus signature database is update, the whole hard drive can be scanned as follows:

```
clamscan -r -i --bell /
```

3.6.3.2 Scanning for Root-kits Another utility provides the ability to scan for known *root-kits*. A *root-kit* is a program that enables an unauthorized user to gain control of a computer system. To install the `chkrootkit`, type:

```
sudo apt-get install chkrootkit
```

Next, we can scan for root-kits as follows:

```
sudo chkrootkit
```

To display only warnings and suspected infected files, type: `bash sudo chkrootkit -q`

3.6.3.3 fail2ban If you are connecting to your Raspberry Pi using SSH, you are susceptible to both *brute force attacks* and *dictionary attacks*. A *brute force attack* connects the the SSH port and tries cracking a password using different combination of characters. A *dictionary attack* connects to an SSH port and attempts to log in using a list of common passwords.

The **fail2ban** utility is an intrusion prevention system that monitors the SSH port (as well as other ports) for repeated unsuccessful login attempts. If a particular client exceeds a certain threshold of failed login attempts, its IP address is blocked.

To install **fail2ban**, type:

```
sudo apt install fail2ban
```

To protect the Raspberry Pi from SSH attacks, edit the configuration file located at `/etc/fail2ban/jail.d/defaults-debian.conf`. This file should contain at least the following settings for SSH:

```
[sshd]
```

```
enabled = true
```

The configuration file can be used to fine-tune others settings. For example, you can configure the maximum allowable login retries and the time that an IP address is banned as follows:

```
[sshd]
enabled = true
filter = sshd
maxretry = 5
bantime = 600
```

3.6.4 Setting up a Print Server

Printing in Linux is typically done using CUPS (Common Unix Printing System). CUPS enables you to use your Raspberry Pi as a print server which can allow you to share a printing device with others on your local network. To begin, install CUPS on the Raspberry Pi, type:

```
sudo apt install cups
```

Next, add the default user, `pi`, to the group of users that can access the printer:

```
sudo usermod -a -G lpadmin pi
```

A variety of CUPS settings can be modified in the file `/etc/cupsd/cupsd.conf` to enable operation of a CUPS server (and to adjust access control). The server includes a web interface to view print jobs and administer the printers. Once running, the web interface can be accessed by pointing a browser to the IP address of the Raspberry Pi followed by :631. For example, if the Raspberry Pi is located at IP address 192.168.1.1, you should point your browser to 192.168.1.1:631. Consult the online [CUPS documentation](#) for more information.

3.7 Compiling the Linux Kernel

Using the `gcc` tool chain, it is also possible to customize and recompile the Linux kernel itself! Before beginning, install the `git` tool along with other build dependencies:

```
sudo apt install git bc bison flex libssl-dev make libncurses5-dev
```

Next, use `git` to download the kernel source code as follows:

```
git clone --depth=1 https://github.com/raspberrypi/linux
```

Note that omitting the `\depth` parameter will download the entire repository history and will take a very long time! The next step is to prepare the default configuration of the kernel. For the Raspberry

Pi 4 this is done as follows:

```
cd linux
KERNEL=kernel7l
make bcm2711_defconfig
```

In the commands above, `kernel7l` and `bcm2711_defconfig` refer to the default kernel filename and build target respectively for the Raspberry Pi 4. If you are using a different model of the Raspberry Pi, you will need modify these commands with a different a different kernel filename and build target as summarized below:

Model	Default kernel filename	Config build target
Raspberry Pi Zero	kernel.img	bcmrpi_defconfig
Raspberry Pi 2, 3, 3+	kernel7.img	bcm2709_defconfig
Raspberry Pi 4	kernel7l.img	bcm2711_defconfig

If you are not sure which model you are using, it should be printed on the Raspberry Pi circuit board, otherwise you can query your device model as follows:

```
cat /proc/device-tree/model
```

After running the appropriate make command, a `.config` file will be generated with various settings for your particular Raspberry Pi model. These kernel settings can be customized according to your wishes and requirements. To make changes to these kernel setting we can use the `makemenu` utility as follows:

```
make menuconfig
```

This utility essentially provides a friendly interface to modify numerous configuration settings stored in the `.config` file. These settings will later be used to guide the compilation process to produce a customized kernel image.

For example, one simple customization that might be made is to give the new kernel a custom name to distinguish it from other kernels. You can change the kernel name under *General setup -> Local Version*. Once all the customizations are set, save the configuration. After making changes to the configuration, a 32-bit kernel can then be compiled by typing the following:

```
make -j4 zImage modules dtbs
```

Note that the `-j4` flag will allow the compilation workload to be spread across all four cores in the ARM processor thus speeding up the compilation. Even so, the compilation process will take a very

long time on a Raspberry Pi since the Linux kernel is a large program with millions of lines of code. When compilation ends successfully, you may install the new kernel as follows:

```
sudo make modules_install  
sudo cp arch/arm/boot/dts/*.dtb /boot/  
sudo cp arch/arm/boot/dts/overlays/*.dtb* /boot/overlays/  
sudo cp arch/arm/boot/dts/overlays/README /boot/overlays/  
sudo cp arch/arm/boot/zImage /boot/$KERNEL.img
```

Note that it is possible use a kernel with a different filename by setting a kernel parameter in [/boot/config.txt](#). After these steps are successfully completed, reboot the Raspberry Pi to make the new kernel active. Connect using SSH and verify that the new kernel is running using the following command:

```
uname -a
```

The new name you configured for the kernel should now be reported along with the Linux kernel version. You will need to reboot each time you change your kernel configuration for the new kernel to become active.

To reinstall a default kernel from the Raspberry Pi repository, type:

```
sudo apt --reinstall install raspberrypi-kernel
```

Note that to speed up kernel compilation it is possible to cross-compile a modified ARM kernel on a desktop workstation and then transfer the new kernel to the Raspberry Pi.

4 Networking

4.1 Networking Utilities

The Linux shell supports a variety of tools and utilities for networking. What follows are some helpful practical tools related to networking.

4.1.1 ping

Ping (Packet Inter-Netwrok Gopher) is helpful utility for testing end-to-end connectivity and transport delays in a TCP/IP network. Ping works by sending an Internet Control Message Protocol (ICMP) packet with an “echo request” to the specified machine. When a machine receives an echo request, it normally replies with an echo reply packet (note that network interfaces can also be configured to ignore ping requests). To `ping` a remote host, simply type:

```
ping hostname
```

where `hostname` represents the host name (or a “dotted” IP address) of the host you wish to ping. By default, the “pings” will continue each second, and the results along with the delay time will be displayed for each ping.

4.1.2 traceroute

Traceroute uses similar ICMP packets that are used by `ping` but with the time-to-live (TTL) packet field set so that it can get replies from each of the hops along the way to a destination machine. To install `traceroute`, type:

```
sudo apt install traceroute
```

To run `traceroute`, simply type the following:

```
traceroute hostname
```

where `hostname` is the host name to which you want to trace the route. Typically, a list of hosts and routers that packets take on their way to `hostname` should be listed along with their corresponding delays. `traceroute` can be handy for identifying the number of hops and sources of congestion in a route between two machines on the internet.

4.1.3 mtr

Closely related to [traceroute](#) is [mtr](#), which is another network diagnostic tool. [mtr](#) continually sends packets along the route to a destination and keeps track of the statistics of the response times at each hop in the route. A summary diagnostic is displayed showing the average, best, and worst times along with the percentage of packet losses detected. To install [traceroute](#), type:

```
sudo apt install mtr
```

To use [mtr](#), type:

```
mtr hostname
```

where [hostname](#) indicates the destination and route that you want to analyze.

4.1.4 dig

Not so long ago, when the Internet was comprised of a limited number of hosts, there was a file called [/etc/hosts](#) contained information about every hosts on the network. This file was maintained and distributed across the Internet. This approach became impractical as the number of hosts grew large. A hierarchical, domain-based, naming scheme called DNS (the Domain Name System) was invented and is defined in RFC 1034 and 1035. The DNS system uses UDP packets (Request/Reply) and a resolver is used to map names to an IP address by sending a request to a local DNS server. The [/etc/hosts](#) file is still found on some hosts, but is often used to store local addresses such as the address of the local machine.

The Internet is divided into various top level domains which are approved by [ICANN](#). There are about a dozen root servers in the world which know the addresses of the top-level domain servers. These top-level domains are further subdivided into subdomains, which can be further partitioned, and so on. These domains on the Internet can be represented by a tree with the leaves of the tree representing hosts or groups of hosts.

Every domain has a set of resource records associated with it, and DNS servers use these resource records when replying to queries. When a host has a name query, it passes the query to one of the local name servers. If the name falls under the jurisdiction of the local name server, it returns the authoritative resource records. If the requested name cannot be satisfied locally and is part of a remote domain, then a recursive query can be made to a remote name server.

Dig is a flexible tool for interrogating DNS name servers. It performs DNS lookups and displays the answers to queries returned from your name server(s). The [dig](#) utility is often used to troubleshoot DNS problems. To install the [dig](#) utility (alongside other DNS utilities), type:

```
sudo apt-get install dnsutils
```

Once installed, a basic `dig` hostname query is made as follows:

```
dig @server hostname
```

where `server` is the DNS server, and `hostname` is the name to query. The `server` parameter is optional; if it is not given the default name server will be used. To perform a *reverse look-up* (i.e. determine the hostname given an IP address), do the following:

```
dig -x 1 2.3.4
```

where `1.2.3.4` is the IP address of the host to query. Type `man dig` from the command line for a detailed description of how it can be used.

4.1.5 wget

Wget is a handy utility for downloading web files from the command line. The basic syntax for `wget` is as follows:

```
wget URL
```

where `URL` is the *uniform resource locator* of the file you wish to download. For example, to download this guide, use:

```
wget https://sites.calvin.edu/derek/tutorials/rpi-guide.pdf
```

`wget` includes a rich set of command line options; for more information, type:

```
wget --help
```

4.1.6 curl

Curl is a tool to transfer data from or to a server. It supports many different protocols, including HHTP, HTTPS, FTP, and FTPS. The basic syntax for `wget` is as follows:

```
wget URL
```

where `URL` is the *uniform resource locator*, including its protocol, of the file you wish to download. By default, `curl` writes the received data to the screen but can be instructed to save the data to a local file using the `-o` option. For example, to download this guide to a local file named `guide.pdf`, use:

```
curl https://sites.calvin.edu/derek/tutorials/rpi-guide.pdf -o guide.pdf
```

`curl` includes a rich set of command line options so that it can operate without any user interaction. For more information, type:

```
curl --help
```

4.1.7 telnet

Telnet is a client for connecting to remote systems using the Transmission Control Protocol (TCP). For example, to use telnet to connect to a web server, type:

```
telnet hostname 80
```

This will connect to port 80 on server `hostname`. If it is running a web server you can type `GET` and after hitting enter, you should see HTML code sent in reply.

Note: `telnet` can connect to any port, but note that all text is sent *in the clear*, ie. insecurely and without encryption. Anyone observing the traffic on the network will be able to read any text that is sent.

4.1.8 Drill Exercises

1. What is the IP address of your local name server?
2. What is the IP address of `www.calvin.edu`?
3. What is a DNS MX record? What is the MX record for `calvin.edu`?
4. Do DNS requests use the TCP or UDP transport protocol? Why does DNS use the transport protocol that it does?
5. Use `wget` to download this book.

4.2 The Web

It is possible to configure the Raspberry Pi to run a *web server*. A web server is a program that accepts requests using **HTTP** (Hypertext transfer Protocol) or **HTTPS** (secure HTTP). Typically, web servers are listed for incoming HTTP requests on port 80 and on port 443 for HTTPS. Various open source web servers are available, including the Apache web server, Nginx, and Lighttpd. In the following sections we will explore both Lighttpd and Nginx.

4.2.1 Lighttpd

Lighttpd is a simple, lightweight web server which can be installed from the command line as follows:

```
sudo apt update  
sudo apt -y install lighttpd
```

It is recommended that you consult the online manuals for Lighttpd to ensure the server is configured and setup securely.

Test the webserver by creating a simple web file as follows:

```
sudo nano /var/www/html/index.html
```

Using an editor of your choice, enter a simple HTML file as follows:

```
<html>  
<head></head>  
<body>  
    Hello world!  
</body>  
</html>
```

Point a browser to <http://a.b.c.d> where **a.b.c.d** is the local IP address of your Raspberry Pi. The message, “hello world” should appear in a web page.

You can see the status of the server by using the following command:

```
sudo service lighttpd status
```

To disable the lighttpd web service from starting up on the next reboot boot, type:

```
sudo systemctl disable lighttpd
```

4.2.2 Nginx

Nginx is a full-featured open source web server which can be installed from the command line as follows:

```
sudo apt update  
sudo apt -y install nginx
```

If the web server needs to support PHP, type the following:

```
apt install php php-fpm php-cli
```

Test the webserver by creating a simple web file in the root web folder as follows:

```
sudo nano /var/www/html/index.html
```

Using an editor of your choice, enter a simple HTML file as follows:

```
<html>
<head></head>
<body>
    Hello world!
</body>
</html>
```

Point a browser to <http://a.b.c.d> where **a.b.c.d** is the local IP address of your Raspberry Pi. The message, “hello world” should appear in a web page.

You can view the status of the server by using the following command:

```
sudo service nginx status
```

To disable the nginx web service, type:

```
sudo systemctl disable nginx
```

4.3 Java Network Programming

The Berkeley *sockets* interface was originally developed at the University of California at Berkeley as a tool to for network programming. A **Socket** is a handle to a communications link over a network with another application. A socket *connection* includes a local **IP address** and **port number** and a remote **IP address** and remote **port number**.

Java is one of the first languages designed with networking in mind. Java applications can conveniently send and receive data across the Internet. The `java.net` package provides the classes for implementing networking applications. Several of the key classes for Java networking are summarized in the table below.

Class	Description
ServerSocket	implements server sockets
Socket	implements client sockets
InetAddress	represents an IP Address
DatagramPacket	represents a UDP datagram packet

Class	Description
DatagramSocket	socket for sending and receiving UDP datagram packets
MulticastSocket	used for sending and receiving IP multicast packets
URL	represents a Uniform Resource Locator, a pointer to a “resource” on the World Wide Web

4.3.0.1 ServerSocket Class The `ServerSocket` class is used by servers to listen for client connections. The `ServerSocket` class specifies a *port* number to listen on for connections. This constructor method blocks until a connection is made and then returns a socket for communicating with the client. Once the communication is complete, a `close` method can be called to close the socket connection.

4.3.0.2 Sample Java Network Code What follows is a sample java program for establishing a Server Socket on port 7777, waiting for a connection, and performing a simple exchange of text messages with a remote client. Use a text editor to enter the program and save it as `Main.java`.

```
import java.net.*;
import java.util.*;
import java.io.*;

class Server {
    public static void main(String args[]) throws IOException {
        System.out.println("Creating server socket on port 7777 and
                           waiting for connection:");
        ServerSocket server = new ServerSocket(7777);
        Socket socket = server.accept();

        // Read text from the socket connection and display it
        Scanner in = new Scanner(socket.getInputStream());
        String msg = in.nextLine();
        System.out.println("Received message: "+msg);

        // Send a simple message back to the client
        PrintWriter out = new PrintWriter(socket.getOutputStream());
        out.println("Hello from the server side!");
        out.flush(); // empty the buffer

        // Close socket and server
        socket.close();
        server.close();
    }
}
```

To compile this program to java *bytecode* and then run it, type:

```
javac Server.java  
java Server
```

This program will create a server socket and wait for a client connection. You can test the program by connecting to the server socket using the [telnet](#) utility, a program that can connect to a remote server and send and receive text characters. To install telnet on your Raspberry Pi, type:

```
sudo apt install telnet
```

You can connect to the Java server program remotely using the IP address of the Raspberry Pi or you can connect locally using the *local loopback interface* which uses a reserved IP address of 127.0.0.1 and a special hostname of [localhost](#). Open another terminal session and use a [telnet](#) connection to the local loopback interface on port 7777 as follows:

```
telnet 127.0.0.1 7777
```

Type a short message and you should see it displayed by the server, and then the client should display the server message: [Hello from the server side!](#).

Rather than using [telnet](#), you can also create your own Java program for making a client connection to the server. What follows is a sample java program for communicating with our simple server on the local loopback interface:

```
import java.net.*;  
import java.util.*;  
import java.io.*;  
  
class Client {  
    public static void main(String args[]) throws IOException {  
        // Create a socket connection to a server port 7777  
        Socket socket = new Socket("localhost", 7777);  
  
        // Send some data to the server  
        PrintWriter out = new PrintWriter(socket.getOutputStream());  
        out.println("Hello from the client!");  
        out.flush();  
  
        // Read reply from server and display it  
        Scanner in = new Scanner(socket.getInputStream());  
        String msg = in.nextLine();  
        System.out.println("Received message: "+msg);  
  
        // Close socket connection  
        socket.close();  
    }  
}
```

Note that if you want to run the client program remotely, replace the `localhost` hostname with the hostname or IP address of the remote server. To compile the client program, type:

```
javac Server.java
```

Next, start the server in one terminal window. It will create a server socket and wait for a connection. On another terminal window, run the client java program as follows:

```
java Server
```

Observe both terminal sessions and note the messages that are displayed as the client connects to the server, exchanges a pair of messages, and then exits.

For more information, see the [OpenJDK Java documentation](#).

5 Databases

5.1 SQL Databases and the Raspberry Pi

A database is a structured collection of logically related data. One common type of database is the relational database, a term that was originally defined and coined by Edgar Codd in 1970. In a relational database the data is stored in 2-dimensional tables of rows (called tuples or records) and columns (called attributes or fields). A table or relation is defined as a collection of records or tuples that have the same fields or attributes.

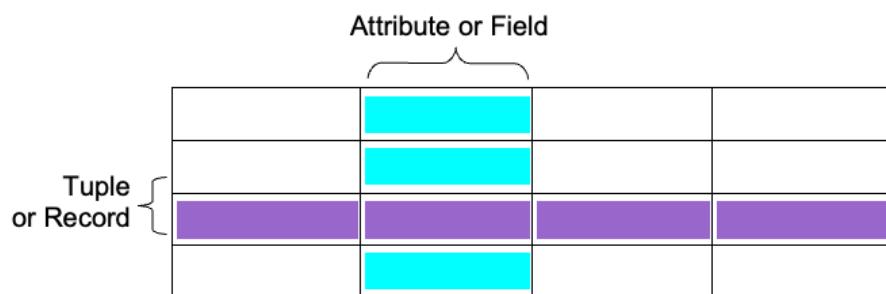


Figure 7: Data is stored in 2-dimensional tables of rows

Structured Query Language or SQL is a standard language used to interact with relational database systems and is maintained as an ISO standard. SQL provides a convenient level of abstraction to interact with a database to define, manage, and query data. SQL is a declarative programming language as opposed to an imperative programming language such as the C programming language in which all the computations are explicitly stated step-by-step. A declarative language is one that defines what the program should accomplish, rather than describing *how* to go about accomplishing it. An SQL statement is processed by the database system which determines the best way to return the requested data. Although there are small differences in SQL syntax between relational database systems, the syntax is largely the same across many systems.

There are numerous commercial and open source SQL database systems available. Some examples of open source projects include SQLite, MySQL, and PostgreSQL, all of which are included in the standard Raspberry Pi software repositories.

5.1.1 Using SQLite

SQLite is a simple, a friendly, lightweight database program which supports most SQL commands. SQLite is also convenient since it does not run as a server and stores data in a single file which can be placed anywhere.

SQLite can be installed as follows:

```
sudo apt install sqlite3
```

To setup an example sqlite database, type the following:

```
sqlite3 temperature.db
```

Once SQLite launches, create a table to store temperature data by typing the following SQL command:

```
CREATE TABLE TemperatureData  
(datetime TEXT NOT NULL, temperature double NOT NULL);
```

Make sure you include a semicolon at the end of the command to indicate that your SQL command is complete. Note that SQLite does not have a storage class for dates and/or times so we will store the date and time as a text field. Next, check that the table was successfully created by typing:

```
sqlite> .tables
```

Type the following sqlite3 command to show the structure of the table you just created:

```
pragma table_info('TemperatureData');
```

Once the table is defined you can insert, delete, edit, and query data in the table. For example, to insert data into the table, type the following:

```
INSERT INTO TemperatureData VALUES (datetime('now','localtime'), 23.0);
```

To show the table data, use a query like the following:

```
SELECT * FROM TemperatureData;
```

Finally, to quit SQLite, type the following at the prompt:

```
sqlite> .quit
```

5.1.2 Using MySQL

Although there are some graphical user interfaces available for MySQL, this tutorial will focus on the command line interface for MySQL. MySQL can be installed as follows:

```
sudo apt install mariadb-server
```

Run the following command and follow the prompts to setup and secure the MySQL server:

```
sudo mysql_secure_installation
```

To access your MySQL server with root access, type:

```
sudo mysql -u root -p
```

Once other usernames and passwords have been established, one can access MySQL using them as follows:

```
mysql -u username -p
```

Where `username` is the username, and after pressing enter a prompt will appear for a password. In order to use a database or create new ones, you must have sufficient privileges assigned to your username.

To list all the databases on the MySQL server, type:

```
SHOW DATABASES;
```

SQL statements include one or more SQL keywords that are often written in uppercase as a matter of style and which end with a semi-colon. To create new database in SQL from the command line, use the CREATE DATABASE statement as follows:

```
CREATE DATABASE school;
```

After executing this statement, MySQL should return a message indicating whether the command was successful or not. The SHOW DATABASES statement will show all the databases on the database server. After creating a new database, the SHOW DATABASES statement can be used to verify that the new database has been created. To show all the databases on the server, type the following:

```
SHOW DATABASES;
```

MySQL should return with a list of the current databases. The USE command is issued to select and use a specific database. For example, to use the database we just created, type:

```
USE school;
```

A database is a collection of tables. Once the database selected, you can query and access the tables in the database. To create a table within the school database, you use the CREATE TABLE statement. For example, to create a table of students names type the following:

```
CREATE TABLE students (
    studentNumber int NOT NULL,
    lastName varchar(50) NOT NULL,
    firstName varchar(50) NOT NULL,
    PRIMARY KEY (studentNumber)
);
```

The table name is specified after CREATE TABLE statement and then the columns names are given

followed by data type, size, NOT NULL or not. A field which is specified as NOT NULL must contain a value. It is also possible to specify the primary key of the table. The primary key is used to uniquely identify each row in the table. Since student numbers are supposed to be unique, the primary key in this example is set to studentNumber. If the table has more than one primary key, you can separate them by a comma. In order to view details about a table, including information about fields and data types, use the DESCRIBE statement. For example, type:

```
DESCRIBE students;
```

This will return information about the table we just created and information about the fields that comprise it. The output from MySQL should resemble the following:

Field	Type	Null	Key	Default	Extra
studentNumber	int(10)	NO	PRI	NULL	
lastName	varchar(50)	NO		NULL	
firstName	varchar(50)	NO		NULL	

To modify the structure or type of data in existing tables, MySQL provides an ALTER command. Although the SQL keywords themselves are not case sensitive, the database and tables may be case sensitive depending on the underlying operating system being used. The data type of each of MySQL fields or attributes must also be specified. MySQL supports a variety of numeric, character and binary data types. Some of the common data types supported in MySQL are summarized in the following table:

Data Type	Description
INT	A signed integer (4 bytes)
FLOAT	A floating-point number (4 bytes)
DOUBLE	Double precision floating-point number (8 bytes)
DATE	Date in the format YYYY-MM-DD
DATETIME	Date and time
CHAR(M)	A fixed-length string with a length of M characters (between 1 to 255 characters)
VARCHAR(M)	A variable-length string with a length of up to M characters (between 1 to 255)
TEXT	A text field with a maximum length of 65535 characters
BLOB	A binary large object (used to store binary data such as images)

In order to show all the tables in a database, you can use the SHOW TABLES statements as follows:

```
SHOW TABLES;
```

To add records to the students database, use the INSERT statement. For example, to add “John Calvin” with a student ID number of 12345 into the students table, do the following:

```
INSERT INTO students  
(studentNumber, firstName, lastName) VALUES  
(12345, "John", "Calvin");
```

5.1.2.1 MySQL Queries You can also ask MySQL to search for data by submitting a query asking for all the records or rows that match a specific criteria. The SQL SELECT statement is used to perform queries on an SQL table. For example, to list all the students in the table, type:

```
SELECT * FROM students;
```

The * indicates that all columns should be returned. The SELECT query returns the requested data as text in a tabular format like follows:

studentNumber	lastName	firstName
12345	Calvin	John

To display specific columns, replace the * with a comma-separated list of columns that you would like to see displayed. For example, to display just the lastName and firstName columns, type:

```
SELECT lastName, firstName FROM students;
```

To list the students in alphabetical order using the ORDER BY clause as follows:

```
SELECT * FROM students ORDER BY lastName, firstName;
```

The order can be explicitly set to be ascending or descending by placing the ASC or DESC keywords at the end of the query. To prevent the SELECT statement from returning duplicate values in the results, the DISTINCT keyword can be used. For example, to list all the distinct first names in the students table, type:

```
SELECT DISTINCT firstName FROM students;
```

It is also possible to restrict the search to meet specific criteria using the WHERE keyword. For example, to list all the students who have the first name of “John”, type:

```
SELECT * FROM students WHERE firstName = 'John';
```

The conditions to restrict search results can be further combined with boolean operators such as AND and OR operators to express search based on different conditions. In addition to WHERE, there are other keywords such as LIKE and BETWEEN which can be used to restrict the results of a search. For example, to list all the students whose first name starts with a “J”, type:

```
SELECT * FROM students WHERE firstName LIKE 'J%';
```

where % is a wildcard which matches any character or sequence of characters. The BETWEEN keyword will restrict a search to values that fall in a range between some minimum and maximum value. For example, to return all the last names that lie alphabetically between “Calvin” and “Luther”, type:

```
SELECT * FROM students WHERE lastName  
BETWEEN 'Calvin' AND 'Luther';
```

One can also query the number of rows that match a certain condition using the COUNT keyword:

```
SELECT COUNT(*) FROM students WHERE firstName LIKE 'J%';
```

To delete data from a table, use the DELETE statement. For example, to delete all the students with the last name of “Calvin”, do the following:

```
DELETE FROM students WHERE lastName = 'Calvin';
```

To make another table called courses that stores a course code and student number for each course a student is enrolled in, use the CREATE statement again as follows:

```
CREATE TABLE courses (  
    studentNumber int NOT NULL,  
    courseCode varchar(7) NOT NULL );
```

In this table the studentNumber will not necessarily be unique since it will appear once for each course in which a student is enrolled. The courseCode will also not necessarily be unique since it will be repeated for each student in the course. Note that the students names do not need to be stored again; they can be retrieved if required by looking up the studentNumber in the students table. To add some records to the courses database, use the INSERT statement once again:

```
INSERT INTO courses (studentNumber,courseCode) VALUES (12345, """CSC101A""");
```

To change or modify data in a table, use the UPDATE keyword as follows:

```
UPDATE students SET studentNumber = 123  
WHERE firstName = 'John' AND lastName = 'Calvin';
```

This statement will modify the students table and replace the studentNumber for the student with the name John Calvin. It is possible to modify multiple field values with an UPDATE statement using a comma-separated list of assignments. The WHERE clause in this case uses a boolean AND operator to

make a more complex condition. You can also ask MySQL to search data from multiple tables by using a JOIN operation. The JOIN keyword relates two or more tables, typically by using values that are common between them. The students and courses database have a common value of studentNumber that can be used to join them. The ON keyword can be used to specify a condition with which to join tables. For example, to list all the first names and last names of students enrolled in CSC101A, you can use a join operation based on the condition of matching a studentNumber in a query as follows:

```
SELECT firstName, lastName  
FROM students  
JOIN courses  
ON students.studentNumber = courses.studentNumber  
WHERE courseCode = 'CSC101A';
```

To close a database, type the following:

```
CLOSE DATABASE school;
```

It is also possible to delete a table and all its contents using the DROP command. This command should be used with care since it permanently deletes your table and cannot be undone.

```
DROP TABLE students;
```

Finally, you can quit MySQL at any time by typing the QUIT command.

```
quit
```

5.1.3 Backing Up MySQL Data

You can use the `mysqldump` utility to create a simple backup of your database to a file using the following syntax:

```
mysqldump -u username -p password databasename > backup.sql
```

where `username` and `password` are your MySQL username and password and `databasename` is the name of the database you want to backup. The resultant file called `backup.sql` will contain all the SQL statements needed to create the table and populate the table in a new database server. If you examine the file `backup.sql` in a text editor you will observe the necessary SQL commands to create the database, its tables, and all the data contents within the tables. The data can be restored by typing the following:

```
mysql -u username -p password databasename < backup.sql
```

5.1.3.1 Using PHP and MySQL Once you are familiar with MySQL syntax in the command-line environment, you can begin to write PHP code which can connect to a MySQL database and query it using SQL statements. PHP includes several functions to connect to a MySQL server and perform various queries. Some of the many PHP MySQL functions are shown in the following table:

Function	Description
mysql_connect	Opens a connection to a MySQL database server
mysql_select_db	Opens a database on the MySQL server
mysql_query	Performs a MySQL query on the currently selected database
mysql_close	Closes a MySQL database connection
mysql_fetch_array	Returns an associative array with the next row from a MySQL query
mysql_error	Returns the text of the error message from a previous MySQL operation
mysql_create_db	Create a MySQL database
mysql_drop_db	Drop a MySQL database
mysql_real_escape_string	makes data safer before sending query to MySQL

The SELECT syntax used on the MySQL command line is the same syntax used to query data using the PHP `mysql_query` function (the query is passed as a string argument). However, before data can be queried, the proper PHP functions must be called in order to connect to the MySQL server and to select the appropriate database. For example, the following PHP code connects to the school database and retrieves a list of students and displays it as a list within a webpage:

```
<ul>
<?php
    $db = mysql_connect("localhost", "username", "password");
    mysql_select_db("school");
    $results = mysql_query("SELECT * FROM students");
    while ($row = mysql_fetch_array($results)) { ?>
        <li> <?= $row["firstName"]." ".$row["lastName"] ?> </li>
    ?>
}
mysql_close($db);
?>
</ul>
```

5.1.3.2 Using MySQL with Java MySQL can also be used with the Java Programming language. The JDBC (Java Database Connectivity) API is provides DBMS connectivity to a wide range of SQL

databases including MySQL as well as access to other tabular data sources such as spreadsheets. The JDBC API includes several classes all found in the `java.sql` package.

5.1.4 Using PostgreSQL

PostgreSQL can be installed as follows:

```
sudo apt install postgresql
```

This package includes a command line client utility called `psql` which can be used to interact with the PostgreSQL server. After the installation you can verify that the `postgresql` service is active as follows:

```
sudo service postgresql status
```

It is recommended that you consult the online manuals for PostgreSQL to ensure the server is setup securely.

When you first run PostgreSQL a default administrator named “postgres” is created. This username can then be used with `psql` to connect to the PosgreSQL service as follows:

```
sudo -u postgres psql
```

Since the new “postgres” user has no password, your first action should be to set the password as follows:

```
\password postgres
```

You can now create a new database as follows:

```
CREATE DATABASE school;
```

This command creates a new database name `school`. Within this database we can define tables, such as tables to store students and courses.

Rather than using the administrator account, we can create a new PostgreSQL user for this database as follows:

```
create user pi with encrypted password 'raspberry';
grant all privileges on database school to pi;
```

These commands create a new user name `pi` with password `raspberry` and privileges to use the new `school` database. We can then type `\q` to quit `psql` and connect to the `school` database using this new username and password as follows:

```
psql school pi
```

An interactive prompt should appear with full access to the school database. To access a different database, you must initiate a new connection as follows:

```
psql database username
```

where `dbname` and `username` are the database name and username you wish to use.

5.1.5 Creating Tables

To create a table within the `school` database, you use the CREATE TABLE statement. For example, to create a table of student names, type the following:

```
CREATE TABLE students (
    studentnumber int NOT NULL,
    lastname varchar(50) NOT NULL,
    firstname varchar(50) NOT NULL
);
```

The table name is specified after CREATE TABLE statement and then the columns names are given followed by data type, size, NOT NULL or not. A field which is specified as NOT NULL must contain a value. Identifiers such as column names are converted to lowercase in PostgreSQL unless they are enclosed in double quotes. In order to view details about a table, including information about fields and data types, use the DESCRIBE statement. For example, type:

```
\d students;
```

This will return information about the table we just created and information about the fields that comprise it. The output from PostgreSQL should resemble the following:

Column	Type	Collation	Nullable
studentnumber	integer		not null
lastname	character varying(50)		not null
firstname	character varying(50)		not null

To modify the structure or type of data in existing tables, PostgreSQL provides an ALTER command. The data type of each of PostgreSQL fields or attributes must also be specified. PostgreSQL supports a variety of numeric, character and binary data types. Some of the common data types supported in PostgreSQL are summarized in the following table:

Data Type	Description
integer	A signed integer (4 bytes)
double precision	Double precision floating-point number (8 bytes)

Data Type	Description
date	Date in the format YYYY-MM-DD
timestamp	Date and time
char [(n)]	A fixed-length string with a length of n characters
varchar [(n)]	A variable-length string with a length of up to n characters
text	variable-length character string

In order to show all the tables in a database, you can use the SHOW TABLES statements as follows:

```
\dt
```

To add records to the students database, use the INSERT statement. For example, to add “John Calvin” with a student ID number of 12345 into the students table, do the following:

```
INSERT INTO students
(studentnumber,firstname,lastname) VALUES
(12345, 'John', 'Calvin');
```

5.1.5.1 PostgreSQL Queries You can also ask PostgreSQL to search for data by submitting a query asking for all the records or rows that match a specific criteria. The SQL SELECT statement is used to perform queries on an SQL table. For example, to list all the students in the table, type: SELECT * FROM students; The * indicates that all columns should be returned. The SELECT query returns the requested data as text in a tabular format like follows:

studentNumber	lastName	firstName
12345	Calvin	John

To display specific columns, replace the * with a comma-separated list of columns that you would like to see displayed. For example, to display just the lastName and firstName columns, type:

```
SELECT lastname, firstname FROM students;
```

To list the students in alphabetical order using the ORDER BY clause as follows:

```
SELECT * FROM students ORDER BY lastname, firstname;
```

The order can be explicitly set to be ascending or descending by placing the ASC or DESC keywords at the end of the query. To prevent the SELECT statement from returning duplicate values in the results,

the DISTINCT keyword can be used. For example, to list all the distinct first names in the students table, type:

```
SELECT DISTINCT firstname FROM students;
```

It is also possible to restrict the search to meet specific criteria using the WHERE keyword. For example, to list all the students who have the first name of “John”, type:

```
SELECT * FROM students WHERE firstname = 'John';
```

The conditions to restrict search results can be further combined with boolean operators such as AND and OR operators to express search based on different conditions. In addition to WHERE, there are other keywords such as LIKE and BETWEEN which can be used to restrict the results of a search. For example, to list all the students whose first name starts with a “J”, type:

```
SELECT * FROM students WHERE firstname LIKE 'J%';
```

where % is a wildcard which matches any character or sequence of characters. The BETWEEN keyword will restrict a search to values that fall in a range between some minimum and maximum value. For example, to return all the last names that lie alphabetically between “Calvin” and “Luther”, type:

```
SELECT * FROM students WHERE lastname  
BETWEEN 'Calvin' AND 'Luther';
```

One can also query the number of rows that match a certain condition using the COUNT keyword:

```
SELECT COUNT(*) FROM students WHERE firstname LIKE 'J%';
```

To delete data from a table, use the DELETE statement. For example, to delete all the students with the last name of “Calvin”, do the following:

```
DELETE FROM students WHERE lastname = 'Calvin';
```

To make another table called courses that stores a course code and student number for each course a student is enrolled in, use the CREATE statement again as follows:

```
CREATE TABLE courses (  
    studentnumber int NOT NULL,  
    coursecode varchar(7) NOT NULL );
```

In this table the studentnumber field will not necessarily be unique since it will appear once for each course in which a student is enrolled. The coursecode will also not necessarily be unique since it will be repeated for each student in the course. Note that the student names do not need to be stored again; they can be retrieved if required by looking up the studentNumber in the students table. To add some records to the courses database, use the INSERT statement once again:

```
INSERT INTO courses (studentnumber,courseCode)
VALUES (12345, 'CSC101A');
```

To change or modify data in a table, use the UPDATE keyword as follows:

```
UPDATE students SET lastname = 'knox'
WHERE studentnumber = 12345;
```

This statement will modify the students table and replace the studentNumber for the student with the name John Calvin. It is possible to modify multiple field values with an UPDATE statement using a comma-separated list of assignments. The WHERE clause in this case uses a boolean AND operator to make a more complex condition. You can also ask PostgreSQL to search data from multiple tables by using a JOIN operation. The JOIN keyword relates two or more tables, typically by using values that are common between them. The students and courses database have a common value of studentNumber that can be used to join them. The ON keyword can be used to specify a condition with which to join tables. For example, to list all the first names and last names of students enrolled in CSC101A, you can use a join operation based on the condition of matching a studentNumber in a query as follows:

```
SELECT firstname, lastname
FROM students
JOIN courses
ON students.studentnumber = courses.studentnumber
WHERE courseCode = 'CS101A';
```

It is also possible to delete a table and all its contents using the DROP command. This command should be used with care since it permanently deletes your table and cannot be undone.

```
DROP TABLE students;
```

To close a database connection and quit `psql`, type the following:

```
\q
```

6 Embedded System and the Internet of Things

6.1 The Raspberry Pi Hardware

The Raspberry Pi is a single-board computer (SBC) built around the ARM processor (an acronym for Advanced RISC Machines). The ARM processor is uses a **RISC** (Reduced Instruction Set Computer) architecture, an approach to processor design that emerged in the early 1980's as the result of research performed at the University of California in Berkeley. The RISC architecture simplified the instruction set of a processor so that the instructions could run faster and consume less power and space on the chip.

The Raspberry Pi combines an ARM processor with memory and various inputs and outputs, including USB ports, an Ethernet port, WiFi, HDMI video output, and a general purpose I/O (GPIO) port. Secondary storage is accomplished using a micro-SSD card. The result is a small package that fits in your hand and can run a modern Linux operating system.

A table summarizing the hardware specifications of some recent Raspberry Pi models is summarized in the table below.

Model	Processor	CPU Clock	RAM
Raspberry Pi 5	QUAD Core BCM2712	2.4 GHz	4GB or 8GB
Raspberry Pi 4	QUAD Core BCM2711	1.5 GHz	1GB, 2GB, or 4GB
Raspberry Pi 3B+	QUAD Core BCM2837B0	1.4 GHz	1 GB
Raspberry Pi Zero 2W	QUAD Core BCM2710A1	1 GHz	512MB

While many of the examples provided in the following sections will work on other Linux platforms, the focus of this book will be assume Linux on the Raspberry Pi.

6.2 Controlling GPIO outputs

The Raspberry Pi is a single board computer that can be used as an *embedded system*. An embedded system is one that combines computer hardware and software with various sensors and actuators within another device to perform a specific function. Examples of embedded systems include robots, automobiles, factory automation, and home appliances.

The Raspberry Pi includes a GPIO port that can serve as a means of interfacing inputs and outputs with the real world. Controlling the GPIO port is relatively simple. As an example, suppose we want to control BCM 16 as an output. This can be configured as follows:

```
raspi-gpio set 16 op
```

Next, if we want to set the output of BCM 16 high, we issue the following command:

```
raspi-gpio set 16 dh
```

Likewise, the command to set BCM 16 low is:

```
raspi-gpio set 16 dl
```

If BCM 16 should be connected through a suitable current-limiting resistor to an LED, as indicated in the schematic diagram below.

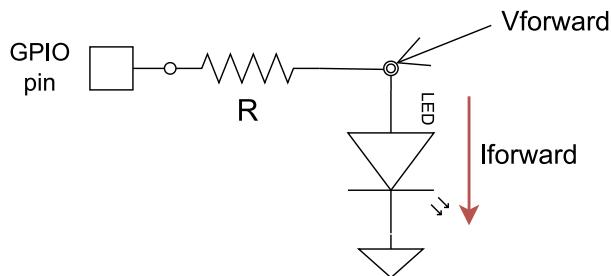


Figure 8: LED current limiting resistor circuit

Selecting a suitable resistance, R , will depend on the forward current required to light up the LED, as well as the forward voltage of the LED. The required resistance, R , to achieve the appropriate LED current is determined by solving the following equation (based on Ohm's Law):

$$R = (V_{GPIO} - V_{forward}) / I_{forward}$$

where:

R = the value of series resistor required

V_{GPIO} = the voltage of the GPIO port when turned on (approximately 3.3V)

$V_{forward}$ = the “forward voltage” of the LED when it is on (approximately 1.8V)

$I_{forward}$ = the forward current required to illuminate the LED (roughly 1.5mA)

According to the datasheet for a particular LED, $V_{forward} \approx 1.8V$ and $I_{forward} \approx 1.5mA$. Using these values, we get a resistance of roughly 1000 ohms or 1 kohm. A suitable wiring diagram connecting the

LED to the Raspberry Pi GPIO port pin is shown below (note the resistor color codes displayed in the diagram reflect a different resistance).

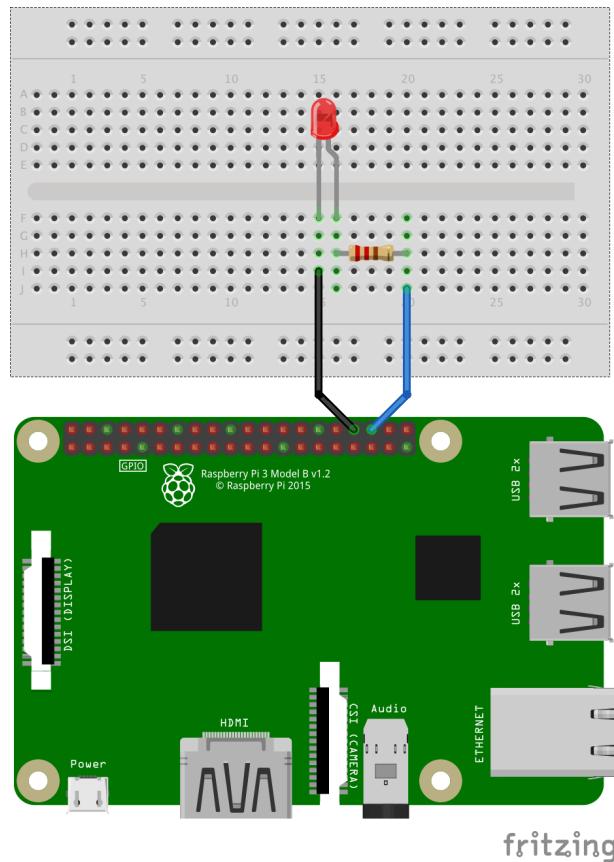


Figure 9: BCM 16 connected via resistor to an LED

Caution should always be observed when connecting GPIO pins to the outside world. Ensure that the power remains **off** while you are performing wiring and always check your wiring before applying power! For some general guidelines to help ensure you do not damage your Raspberry Pi , refer to the section on [Troubleshooting Tips](#).

6.2.0.1 Controlling GPIO inputs We can also set BCM 12 as an input as follows:

```
raspi-gpio set 12 ip
```

We can also turn on a weak internal “pull up” resistor using the “up” parameter. This ensures that when the pin is unconnected the input value “floats” high.

```
raspi-gpio set 12 pu
```

We can read the input state using the gpio command on BCM 12 as follows:

```
raspi-gpio get 12
```

We can repeatedly call this command to monitor any changes to the input and observe the state of the input pin. To learn more about the gpio utility, type:

```
raspi-gpio help
```

6.2.0.2 Controlling GPIO pins using C The following is a sample program that will flash the LED on and off with a delay of 1 second. This coded uses a GPIO library called [pigpio](#). Edit the program using a standard text editor and name the source file [blink.c](#).

```
#include <stdio.h>
#include <pigpio.h>
#include <unistd.h>

#define LED 16
#define DELAY 1

int main (int argc, char *argv[])
{
    if (gpioInitialise()<0) return 1;

    gpioSetMode(LED, PI_OUTPUT);
    while (1)
    {
        gpioWrite(LED, PI_ON);
        printf("LED ON\n");
        sleep(DELAY);
        gpioWrite(LED, PI_OFF);
        printf("LED OFF\n");
        sleep(DELAY);
    }
    gpioTerminate();
    return 0;
}
```

Next, compile the program from the command line as follows:

```
gcc -Wall blink.c -o blink -lpigpio
```

Finally, run the program in the terminal by typing:

```
sudo ./blink
```

Note that this code requires ‘sudo’ privileges. Once started, the LED should begin blinking. Type ctrl-C to exit.

6.2.0.3 Controlling GPIO pins using Python The following is an equivalent Python source file that uses the `RPi.GPIO` library to flash the LED:

```
import time
import RPi.GPIO as GPIO

GPIO16 = 16
DELAY = 1.0

GPIO.setmode(GPIO.BCM)
GPIO.setup(GPIO16, GPIO.OUT)

for count in range(20):
    GPIO.output(GPIO16, True)
    print('LED: on')
    time.sleep(DELAY)
    print('LED: off')
    GPIO.output(GPIO16, False)
    time.sleep(DELAY)

print("Done!")
GPIO.cleanup()
```

Save the file as `blink.py` and then run it as follows:

```
python3 blink.py
```

The GPIO pins can be used to control actuators and read sensors connected to the real world. One example is the [Go-Pi-Go robot](#), a simple educational robot platform built around the Raspberry Pi.

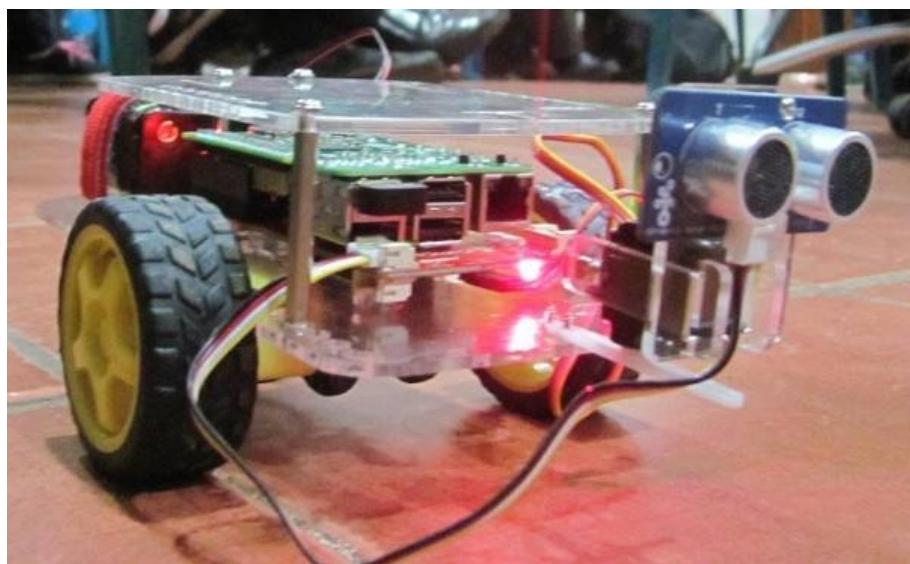


Figure 10: Go-Pi-Go Robot in action - controlled by a Raspberry Pi

6.2.1 Using I2C

Certain GPIO pins also be used to perform I2C communications. I2C (Inter-Integrated Circuit) is a synchronous serial bus used to connect peripheral chips to processors and microcontrollers. I2C requires only two bidirectional wires: the serial data line (SDA) and serial clock line (SCL).

A set of I2C command line tools can be installed as follows:

```
sudo apt-get install i2c-tools
```

In order for these to work, the I2C kernel module should be enabled using the `raspi-config` utility. Next, ensure your username (eg. pi) is in the group permissions for talking to I2C devices as follows:

```
sudo usermod -a -G i2c pi
```

You can then scan the I2C bus for devices as follows:

```
i2cdetect 1
```

If an I2C device is present and wired properly, you should see a device reported along with its I2C address. To query a specific device, issue the following command:

```
i2cget -y 1 address 0 b
```

where `address` is the I2C address of the device you are interested in querying. If everything is wired correctly and functioning, you should see value returned from this query.

6.2.2 Using a Camera

The Raspberry Pi includes a dedicated camera port on the circuit board located between the Ethernet port and the HDMI port. This port can be used to install the standard Raspberry Pi Camera or the NoIR Camera which can be used in low light applications. Remove power before installing the camera cable. Be sure to consult the camera manual to ensure it is inserted properly and is snuggly inserted with the right cable orientation.

Once it is installed, you can test your camera by entering the following command:

```
raspistill -o output.jpg
```

If the camera has problems, ensure that the camera feature has been enabled in `raspi-config`.

To write code that uses the camera, consider using OpenCV, an open source computer vision and machine learning software library. To install OpenCV for Python, type:

```
sudo apt install python3-opencv
```

The following is a sample Python program that uses OpenCV to capture an image and store it to a file.

```
import sys
import cv2

# Initialize camera
print("Initializing camera...")
cap = cv2.VideoCapture(0)
if not cap.isOpened():
    print('Cannot open camera...')
    sys.exit(1)

# Capture a frame
ret, frame = cap.read()
if not ret:
    print('Frame capture failed...')
    sys.exit(1)

# Save frame
cv2.imwrite("image.jpg", frame)
cap.release()
```

One application of computer vision is the sensing of [AprilTags](#), a type of visual fiducial developed at the University of Michigan. AprilTags are similar to QR codes but can be used for localization by computing a 3D position relative to the AprilTag. A simple python program using OpenCV along with the AprilTag library is shown below.

```
import cv2
from apriltag import apriltag

# Initialize camera
cap = cv2.VideoCapture(0)

# Capture one frame
ret, frame = cap.read()
cap.release()

# Detect AprilTag
detector = apriltag('tagStandard41h12')
detections = detector.detect(frame)
print(detections)
```

In the chapter on artificial intelligence, we provide additional examples of computer vision with image classification and recognition.

6.3 Message Queuing Telemetry Transport Protocol (MQTT) for IoT

A Raspberry Pi equipped with sensors and actuators can be controlled and monitored using machine-to-machine (M2M) communications. M2M provides an important enabling technology for applications like **IoT** (the Internet of Things). One type of M2M is a protocol called *Message Queuing Telemetry Transport Protocol* or MQTT. MQTT is a bandwidth-efficient, lightweight protocol allowing clients to publish and subscribe data to a special “broker” server.

6.3.1 Sending MQTT messages from the command line

The [Eclipse Mosquitto](#) project provides an open source MQTT message broker and tools. To install the mosquitto client tools, enter the following:

```
sudo apt install mosquitto-clients
```

These client tools will allow you to publish and subscribe messages to a broker. For this guide, we will make use of a public MQTT broker. Two options are mqtt.eclipse.org or test.mosquitto.org. These public MQTT brokers are open and do not require usernames or passwords (which comes with some security implications).

To send an MQTT message to a broker using the command line, type:

```
mosquitto_pub -h test.mosquitto.org -t raspberry/test
```

This command uses the MQTT protocol to *subscribe* with the broker server test.mosquitto.org to the topic [raspberry/test](#).

Next, switch to another terminal on your local Raspberry Pi (or use another Raspberry Pi) and *publish* a message to the same topic and broker server by entering the following:

```
mosquitto_pub -t raspberry/test -m "Hello World" -h test.mosquitto.org
```

Note that after you enter the above command you should see the “Hello World” message appear where you subscribed to the broker. You have now successfully transferred a message using MQTT. Note that the message was transferred without needing to know the IP address of the publisher or subscriber since messages are nicely handled through a broker server.

6.3.2 Controlling an LED using Python and MQTT

Python programs can be written to publish and subscribe to MQTT messages. To begin, install the Python MQTT library on your Raspberry Pi using the command line as follows:

```
sudo pip3 install paho-mqtt
```

Next, consider the GPIO controlled LED as shown in Figure 9. MQTT can be used to control this LED for a remote Raspberry Pi using the following code:

```
import RPi.GPIO as GPIO
import paho.mqtt.client as mqtt

# Constants
TOPIC = 'raspberry/LED'
PORT = 1883
QOS = 0
KEEPALIVE = 60
LED = 16

# Set hostname for MQTT broker
BROKER = 'test.mosquitto.org'

# Setup GPIO mode
GPIO.setmode(GPIO.BCM)

# Configure GPIO for LED output
GPIO.setup(LED, GPIO.OUT)

# Callback when a connection has been established with the MQTT broker
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print(f'Connected to {BROKER} successful.')
    else:
        print(f'Connection to {BROKER} failed. Return code={rc}')

# Callback when client receives a message from the broker to toggle LED state
def on_message(client, data, msg):
    print(f'MQTT message received -> topic:{msg.topic}, message:{msg.payload}')
    if msg.topic == TOPIC:
        if GPIO.input(LED) == 1:
            GPIO.output(LED, 0)
            print("LED off")
        else:
            GPIO.output(LED, 1)
            print("LED on")

# Setup MQTT client and callbacks
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

# Connect to MQTT broker and subscribe to the topic
```

```
client.connect(BROKER, PORT, KEEPALIVE)
client.subscribe(TOPIC, qos=QOS)

try:
    client.loop_forever()
except KeyboardInterrupt:
    client.disconnect()
    GPIO.cleanup()
    print('Done')
```

This program subscribes to a broker and waits for certain MQTT messages to remotely control the LED. For example, the following command issued from a remote computer will toggle the state of the LED on the Raspberry Pi:

```
mosquitto_pub -t raspberry/LED -m "test" -h test.mosquitto.org
```

Each time an MQTT message is received with the topic `raspberry/LED`, the state of the LED will be toggled on or off.

Note there are security implications since this code uses a public MQTT broker and these MQTT messages are sent in the clear. Anyone could potentially publish messages to the public broker with the same topic and therefore control the LED. There are ways to encrypt and authenticate MQTT messages that are beyond the scope of this guide.

For more information about MQTT, see the [MQTT webpage](#) and the [Mosquitto project webpage](#).

6.3.3 Using MQTT to control Zigbee Devices

It is also possible to use MQTT on the Raspberry Pi to control a network of smart devices and sensors connected over a [Zigbee](#) wireless network. **Zigbee** is built on the IEEE 802.15.4 specification to enable a low-cost, low-power wireless network of smart devices for home automation. [Zigbee2MQTT](#) is a program that provides a wireless *bridge* between Zigbee and MQTT and thus allows you to control Zigbee devices through MQTT messages.

The first step is to setup a local MQTT broker using [mosquitto](#). This can be installed alongside the [mosquitto](#) client utilities on the command-line as follows:

```
sudo apt install -y mosquitto mosquitto-clients
```

We can configure [mosquitto](#) to act as a local MQTT broker and listen *only* on the local loopback interface by adding the following lines in `/etc/mosquitto/conf.d/local.conf`:

```
listener 1883 127.0.0.1
allow_anonymous true
```

Next, enable the `mosquitto` broker service as follows:

```
sudo systemctl enable mosquitto.service
```

Ensure the `mosquitto` service is now running by typing:

```
sudo service mosquitto status
```

Note that any service started on the Raspberry Pi that opens a port exposes you to potential security issues.

Next, there are several dependencies for **Zigbee2MQTT** that need to be installed from the command-line as follows:

```
sudo apt-get install -y npm git make g++ gcc
```

Unfortunately, the Raspberry Pi repos may have an older version of the **nodejs** package, and Zigbee2MQTT requires a recent version of **nodejs**. You can add the repository and install a recent version of **nodejs** as follows:

```
curl -fsSL https://deb.nodesource.com/setup_lts.x | sudo -E bash -  
sudo apt install nodejs
```

Once the dependencies are installed, Zigbee2MQTT can be installed from github by typing the following commands:

```
sudo mkdir /opt/zigbee2mqtt  
sudo chown -R ${USER}: /opt/zigbee2mqtt  
git clone --depth 1 https://github.com/Koenkk/zigbee2mqtt.git /opt/  
    zigbee2mqtt  
cd /opt/zigbee2mqtt  
npm ci
```

Note that the `npm ci` may produce some warnings which can be ignored. Zigbee2MQTT requires a **YAML** configuration file which may be edited by typing:

```
sudo nano /opt/zigbee2mqtt/data/configuration.yaml
```

Edit the configuration file so that it includes the following settings:

```
homeassistant: false  
permit_join: true  
  
# MQTT settings  
mqtt:  
  base_topic: zigbee2mqtt  
  server: 'mqtt://127.0.0.1'
```

```
# Location of Zigbee USB adapter
serial:
  port: /dev/ttyACM0

# use a custom network key
advanced:
  network_key: GENERATE

# Start web frontend
frontend:
  port: 8081

# Enable over-the-air (OTA) updates for devices
ota:
  update_check_interval: 1440
  disable_automatic_update_check: false
```

Next, insert a **Zigbee adapter** into a USB port on the Raspberry Pi. The configuration above assumes the Zigbee USB adapter appears as `/dev/ttyACM0`. You can use the `dmesg` command to find the device file associated with your particular Zigbee USB adapter and then update the configuration file accordingly. Rather than hard-coding a unique network key, the `network_key` setting used above generates a new random key when Zigbee2MQTT is first run.

Security Notes

It's recommended to disable `permit_join` after all the Zigbee devices have been paired with your Zigbee adapter to prevent further devices from attempting to join and possibly exposing the network key.

Note that the `frontend` setting provides a web frontend at a specified port for viewing the Zigbee network. While this can be useful for setup and debugging, you may wish to disable it later.

It is recommended that over-the-air (OTA) updates be enabled for all devices to keep them up-to-date.

Once the setup and configuration are complete, ensure the Zigbee USB adapter is inserted in the Raspberry Pi and start Zigbee2MQTT as follows:

```
cd /opt/zigbee2mqtt
npm start
```

This will build and launch `zigbee2mqtt` from the command-line. Once it builds and launches successfully, you can exit the program by hitting `ctrl-c`. To launch automatically on boot under Linux, [setup Zigbee2MQTT to run using systemctl](#). For more detailed information about installing Zigbee2MQTT, refer to the official [Zigbee2MQTT installation instructions](#).

Next, we need to establish a network of Zigbee devices by pairing each new device with the Zigbee

hub on the Raspberry Pi. Zigbee2MQTT supports a plethora of Zigbee devices and a [friendly device webpage](#) includes notes on compatibility, pairing, and details on what values are exposed.

6.3.3.1 Pairing Zigbee devices Pairing can be easily accomplished using the web frontend to Zigbee2MQTT. The web frontend can be found by pointing a web browser to the IP address of the Raspberry Pi and the port number specified in the `configuration.yaml` file (port 8081 in the example file above). In the web frontend, click the `Devices` tab and then the button labelled `Permit join (All)`. Once this button is clicked a countdown will proceed during which time new devices can be paired to the Zigbee network (typically the countdown lasts for 255 seconds).

Typically a new device is paired by performing a factory reset of the device. The way to perform a factory reset varies by device type and manufacturer. For example, some bulbs can be factory reset by toggling the power a certain number of times and other devices can be factory reset using a reset button in a small pinhole. A few moments after resetting a device, the web page should report the pairing of the device. Clicking on the devices tab on the web page should display a list of paired devices along with each manufacturer, model, and IEEE address. The web frontend provides many nifty features like displaying a network map and the ability to perform updates on connected devices.

In addition to the IEEE address each Zigbee device may be configured with a “friendly name.” By default, the “friendly name” is initialized to the IEEE address, but it is recommended that you assign a more meaningful “friendly name”. For example, a bulb could be named “bulb1” or “porch light”. This allows devices to be controlled and referenced using a *name* rather than relying on a cumbersome IEEE address. Keep a list of the “friendly names” since these will be needed later to control the devices.

6.3.3.2 Controlling Zigbee devices over MQTT Once devices have been paired, they can be controlled simply by sending specially crafted MQTT messages to the local broker. These messages must be published to the topic `zigbee2mqtt/FRIENDLY_NAME/set` where `FRIENDLY_NAME` is the friendly name for a device. In the case of a bulb or smartplug, sending a message of “ON” or “OFF” to the appropriate topic for the device will turn the device on or off.

MQTT messages can be sent from the command line on the Raspberry Pi using tools included with the `mosquitto-clients` package. For example, to turn on a light bulb with the friendly name of “bulb1” using the mosquitto client tool, type:

```
mosquitto_pub -h 127.0.0.1 -t zigbee2mqtt/bulb1/set -m "ON"
```

where `127.0.0.1` is the local loopback address to connect to the local MQTT broker and `zigbee2mqtt/bulb1/set` is the MQTT topic to control the settings for the device with the friendly name `bulb1`.

By subscribing the MQTT topic for a sensor you can receive updates from a sensor. Consult the online Zigbee2MQTT documentation for a [complete list of MQTT topics and messages](#). For an example of using a Raspberry Pi with MQTT, Zigbee, and Python in a smart home application for controlling bulbs and monitoring sensors, visit the [pi-home project](#).

7 Exploring Artificial Intelligence on the Raspberry Pi

Despite its modest hardware capabilities, the Raspberry Pi can be used to run and develop software using artificial intelligence (AI). Several powerful libraries are available for the Raspberry Pi, including **scikit-learn** and **TensorFlow** Lite. Python also includes a number of powerful supporting libraries, such as **Matplotlib** and **plotly** for plotting, **NumPy** providing fast operations on arrays, and **Pandas** for data analysis and manipulation. The following sections provides examples of some of these libraries in action.

7.1 SciKit Learn

SciKit Learn is an open source machine learning library that works with Python. SciKit Learn provides many different features, including regression and clustering algorithms, Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and Support Vector Machines (SVMs).

All forms of machine learning require a dataset which is used for training. SciKit includes a selection of **toy datasets** that can be used to experiment with the machine learning libraries. The following sections demonstrate PCA and SVMs using the classic **iris flower dataset** which is part of the collection of toy datasets. This iris dataset was originally compiled by biologist Ronald Fisher in a well-known 1936 paper. This dataset comprises samples of three species of the Iris flower (*Iris setosa*, *Iris virginica*, and *Iris versicolor*) and measurements of the length and the width of both the *sepals*¹ and the *petals*. The **SciKit Learn** library includes this iris dataset for testing and demonstration purposes. The following Python code uses **Matplotlib** to plot the sepal width versus the sepal length for each of the three iris classes in the dataset.

```
from sklearn import datasets
import matplotlib.pyplot as plt

# Load the classic iris dataset
iris = datasets.load_iris()

# Plot sepal lengths vs. widths for irises in dataset
fig, ax = plt.subplots()
scatter = ax.scatter(iris.data[:, 0], iris.data[:, 1], c=iris.target)
ax.set_title("Plot of Iris features: sepal lengths and widths")
ax.set_xlabel(iris.feature_names[0], ylabel=iris.feature_names[1])
fig = ax.legend(scatter.legend_elements()[0], iris.target_names, loc="best",
                 title="Classes of Irises")
plt.show()
```

Running this code results in the following plot of the iris dataset:

¹A *sepal* is a green leaf-like structure at the base of a flower.

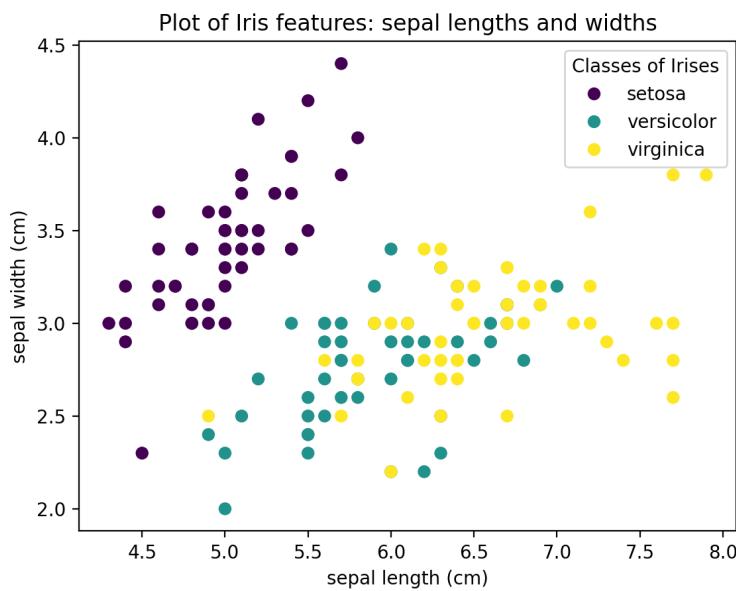


Figure 11: Iris flower dataset plotted with sepal length and width features

7.1.1 Using SciKit Learn with Linear Discriminate Analysis (LDA)

Linear Discriminate Analysis (LDA) is a technique that finds a linear combination of features to best separate the classes in a dataset. A program that performs an LDA transformation on the iris dataset is shown in the program below.

```
from sklearn import datasets
import matplotlib.pyplot as plt
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

# Load the classic iris dataset
iris = datasets.load_iris()

# transform iris dataset using LDA
lda = LinearDiscriminantAnalysis(n_components=2)
X = lda.fit_transform(iris.data, iris.target)

# Show plot with title and legend
fig, ax = plt.subplots()
scatter = ax.scatter(X[:, 0], X[:, 1], c=iris.target)
ax.set_title("Plot of LDA features of IRIS dataset")
fig = ax.legend(scatter.legend_elements()[0],
                iris.target_names, loc="best", title="Classes of irises")
plt.show()
```

The plot that is produced by this code is shown below.

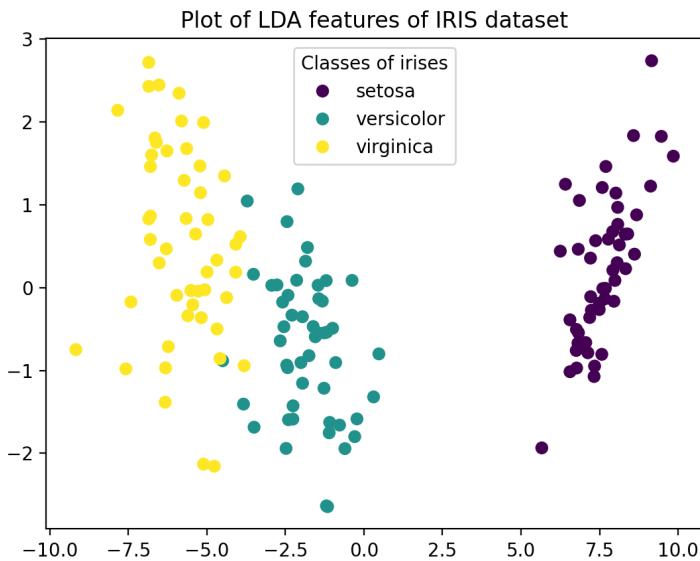


Figure 12: Iris flower dataset transformed with LDA

Comparing this plot to the one shown in Figure 11 *clearly* shows that the LDA transformation separates the different classes of irises much better than just using sepal length and width as features.

7.1.2 Using SciKit Learn with Principal Component Analysis (PCA)

Next, we will use **Principal Component Analysis** (PCA) to transform the iris sepal lengths and widths to another set of features based on eigenvectors computed from the dataset. These eigenvectors provide an orthonormal axis that captures the statistically most significant directions in the data. To perform a PCA transformation on `iris.data`, one can use SciKit Learn library and the transformed data can be plotted on a new set of axes defined by the first two eigenvectors as follows:

```
from sklearn import datasets
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

# Load the classic iris dataset
iris = datasets.load_iris()

# legends, ax = plt.subplots()
fig, ax = plt.subplots()

# transform sepal length into eigenspace using PCA
```

```
pca = PCA(n_components=2).fit_transform(iris.data)

# Show plot with titles and axis names
scatter = ax.scatter(pca[:, 0], pca[:, 1], c=iris.target)
ax.set_title("Plot of first two PCA dimensions")
ax.set_xlabel("First Eigenvector")
ax.set_ylabel("Second Eigenvector")
fig = ax.legend(scatter.legend_elements()[0], iris.target_names,
                 loc="lower right", title="Classes of irises")
plt.show()
```

This produces a plot using the eigenvectors as the axes as shown below:

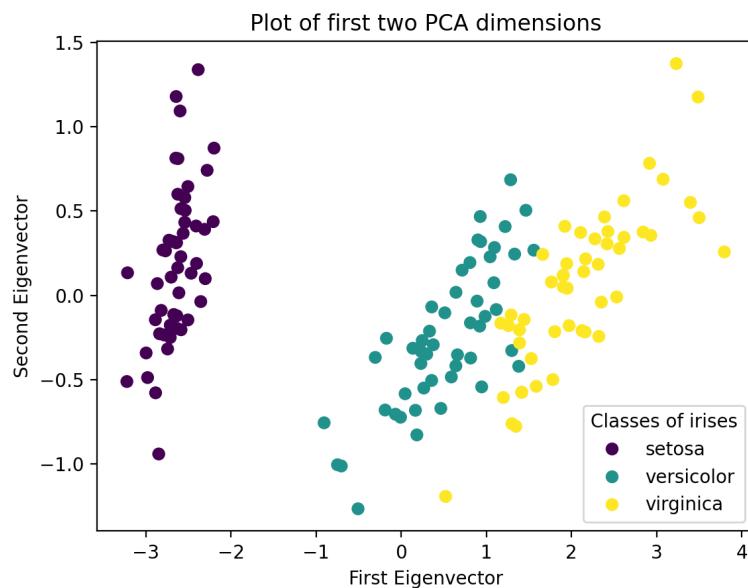


Figure 13: Iris flower dataset plotted using Eigenvectors

Note that the plot of the iris data using the Eigenvector basis functions also appears to separate the classes more clearly. This is due to the fact that PCA produces eigenvector basis functions that capture the statistically most significant features in the data.

7.1.3 Using SciKit Learn with Support Vector Machines (SVM)

SciKit Learn can also be used with a machine learning approach called Support Vector Machines (SVMs). The goal of an SVM is to determine a *hyperplane* which defines the boundary between different data points in order to classify them appropriately. **SciKit Learn** includes a library for machine learning using an SVM.

The following example uses the iris flower dataset and uses SVM to computer a hyperplane between the different iris classes. This example is derived from the [SciKit Learn SVM examples](#) found on the SciKit documentation pages.

```
import matplotlib.pyplot as plt
from sklearn import datasets, svm
from sklearn.inspection import DecisionBoundaryDisplay

# import classic iris dataset
iris = datasets.load_iris()

# Take the first two features from the iris dataset (sepal width and
# length)
X = iris.data[:, :2]
y = iris.target

# create an SVM to fit the data
clf = svm.SVC(kernel="linear", C=1.0)
clf.fit(X, y)

# Plot the data and support vectors
fig, ax = plt.subplots()
X0, X1 = X[:, 0], X[:, 1]
disp = DecisionBoundaryDisplay.from_estimator(clf, X,
                                              response_method="predict", cmap=plt.cm.coolwarm, alpha=0.8,
                                              ax=ax, xlabel=iris.feature_names[0], ylabel=iris.feature_names[1])
ax.scatter(X0, X1, c=y, cmap=plt.cm.coolwarm, s=20, edgecolors="k")
ax.set_title('Support Vector Machine with Linear Kernel')
plt.show()
```

Running the above program displays the following plot, showing the sepal lengths and widths of the iris dataset along with the hyperplane boundaries which can be used for classification. Note that SVM algorith was not able to compute a perfect linear decision boundary that perfectly sepearates the classes. Consequently, when used to identify irises this will likely lead to occassional classification errors.

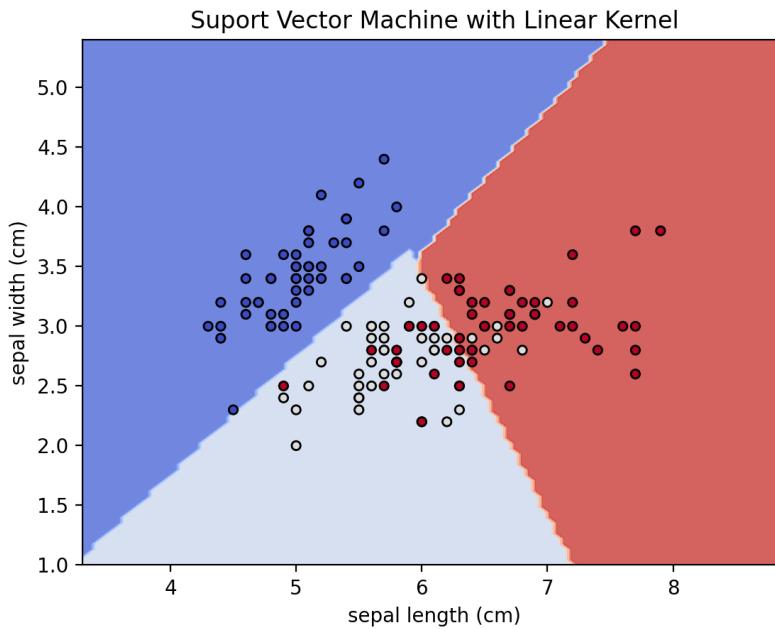


Figure 14: Iris dataset with three classes plotted with SVM boundaries

Finally, we can project the iris dataset features into eigenspace using PCA and then determine the support vectors for the new eigenspace as follows:

```
import matplotlib.pyplot as plt
from sklearn import datasets, svm
from sklearn.inspection import DecisionBoundaryDisplay
from sklearn.decomposition import PCA

# import classic iris dataset
iris = datasets.load_iris()

# transform the iris dataset features into eigenspace using PCA
pca = PCA(n_components=2).fit_transform(iris.data)

# create an SVM to fit the 2D eigenspace
clf = svm.SVC(kernel="linear", C=1.0)
clf.fit(pca, iris.target)

# Plot the PCA features and support vectors
fig, ax = plt.subplots()
disp = DecisionBoundaryDisplay.from_estimator(clf, pca,
                                              response_method="predict", cmap=plt.cm.coolwarm, alpha=0.8, ax=ax)
ax.scatter(pca[:, 0], pca[:, 1], c=iris.target,
           cmap=plt.cm.coolwarm, s=20, edgecolors="k")
ax.set_title('Suport Vector Machine for PCA features')
```

```
ax.set_xlabel('Eigenvector #1')
ax.set_ylabel('Eigenvector #2')
plt.show()
```

The resulting plot showing the eigenvector features and the support vector boundaries is shown below. Note that the boundaries between iris classes are more sharply separated using Eigenspace features as compared to using just the sepal length and width as shown in Figure 14. A similar approach could be taken by performing an LDA transformation to better separate the classes prior to computing the SVM.

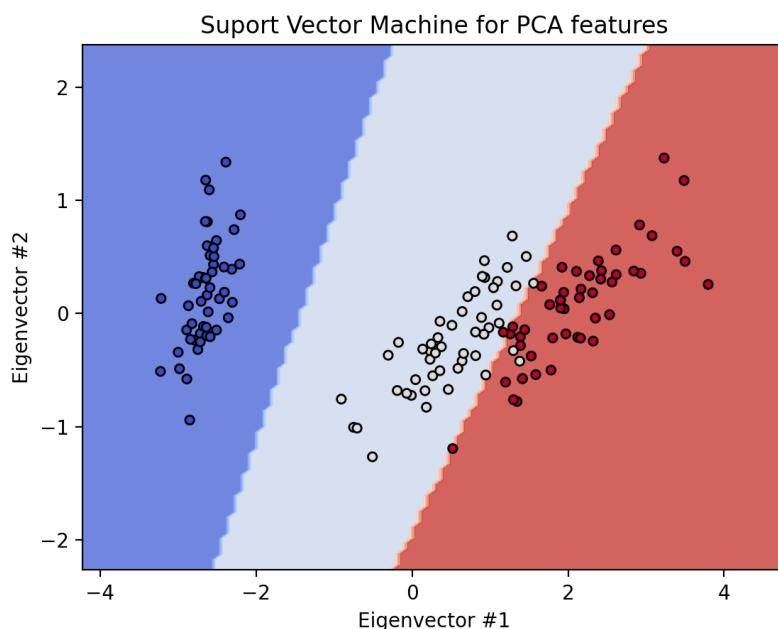


Figure 15: Support Vector Machine boundaries of features in Eigenspace

7.1.4 Using SciKit Learn for SVM Image Classification

A support vector machine can also be used to perform image classification. The following example is inspired by the [SciKit example on recognizing hand-written digits](#) and uses the hand-written digits dataset from the [UC Irvine machine learning repository](#). This dataset has 1797 image samples comprised of an 8×8 array of pixels with 10 classes where each class refers to one digit (0-9).

A short Python program to load the hand-written digits dataset and display them is shown below:

```
import matplotlib.pyplot as plt
from sklearn import datasets, metrics, svm
from sklearn.model_selection import train_test_split
```

```
# load hand-written digits dataset
digits = datasets.load_digits()

# display a sample of ten hand-written digits in the dataset
fig, axes = plt.subplots(nrows=2, ncols=5)
for ax, digit in zip(axes.flatten(), digits.images):
    ax.set_axis_off()
    ax.imshow(digit, cmap='gray')
fig.tight_layout()
plt.show()
```

The plot showing ten hand-written digits in the dataset is shown below.



Figure 16: Ten 8x8 hand-written digits from the UC Irvine digits dataset

We can split the hand-written digits dataset into two sets: a training set and a set for testing. SciKit has a function for taking the larger digits dataset and splitting it in half into training and test sets as follows:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5)
```

where `X` is an array of features and `y` is a vector of labels that classify the data.

Hence we can proceed to train an SVM using half the dataset for training and then use the remaining half of the dataset for testing. The accuracy of the SVM can be determined by comparing the digits predicted by the SVM with the actual labels for the hand-written digits and the *recognition rate* can be reported. The *recognition rate* is the total number of correctly identified digit images divided by the total number of test images. The following code defines an SVM classifier based on half of the dataset and then determines the recognition rate using the other half of the dataset as test images.

```
import matplotlib.pyplot as plt
from sklearn import datasets, metrics, svm
from sklearn.model_selection import train_test_split

# Read digits and reshape digits as a vector of images
digits = datasets.load_digits()
```

```
n_samples = len(digits.images)
data = digits.images.reshape((n_samples, -1))

# Create a support vector machine classifier
svm = svm.SVC()

# Split the dataset: half for training and half for testing
X_train, X_test, y_train, y_test = train_test_split(
    data, digits.target, test_size=0.5, shuffle=False)

# Train on hand-written digits in the training set
svm.fit(X_train, y_train)

# Predict the value of the digit using the testing set
predicted = svm.predict(X_test)

# compute the recognition rate
matches = 0
for x in range(len(y_test)):
    if y_test[x] == predicted[x]:
        matches += 1
recognition_rate = (matches/len(y_test)) * 100;
print(f'Recognition rate: {recognition_rate:.2f}%')
```

The output of this code shows the following:

```
Recognition rate: 96.11%
```

This indicates a respectable recognition rate using an SVM for handwritten digit classification.

For more details, we can plot a *confusion matrix* to visualize the accuracy of a machine learning algorithm. The *confusion matrix* is organized into rows and columns: each row represents each of the classes in a dataset and each column represents the predictions that were made. *Ideally*, the actual classes and the predictions will perfectly align such that the diagonal of the *confusion matrix* is populated with 100's and with 0's everywhere else. The matrix diagonal corresponds to true recognition rates whereas all other cells represent the occurrences of false classifications. SciKit includes a nifty library that can create a beautifully formatted *confusion matrix* as follows:

```
import matplotlib.pyplot as plt
from sklearn import datasets, metrics, svm
from sklearn.model_selection import train_test_split

# Read digits and reshape digits as a vector of images
digits = datasets.load_digits()
n_samples = len(digits.images)
data = digits.images.reshape((n_samples, -1))

# Create a support vector machine classifier
svm = svm.SVC()
```

```
# Split the dataset: half for training and half for testing
X_train, X_test, y_train, y_test = train_test_split(
    data, digits.target, test_size=0.5, shuffle=False)

# Train on hand-written digits in the training set
svm.fit(X_train, y_train)

# Predict the value of the digit using the testing set
predicted = svm.predict(X_test)

# display the resulting Confusion Matrix
disp = metrics.ConfusionMatrixDisplay.from_predictions(y_test, predicted)
disp.figure_.suptitle("Confusion Matrix")
plt.show()
```

The resulting confusion matrix is shown below.

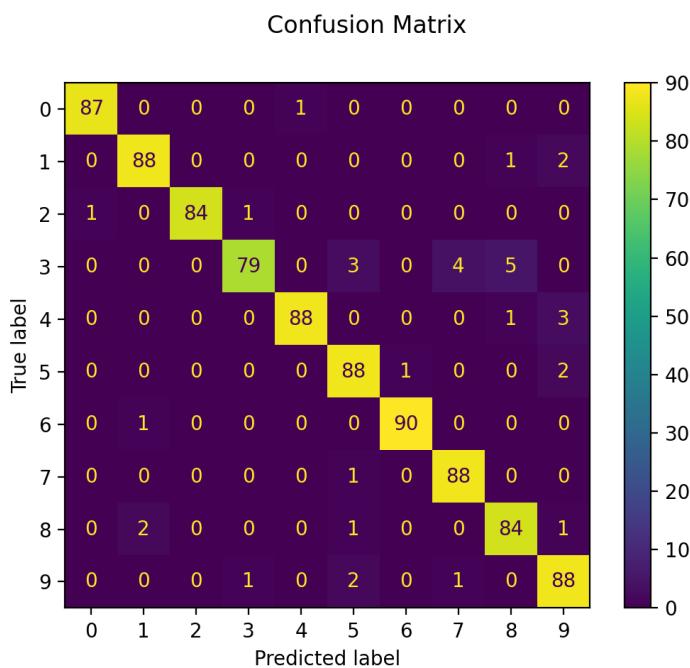


Figure 17: Confusion Matrix for SVM classification of hand-written digits dataset

More information about SciKit Learn with many nifty examples can be found on the [SciKit Learn web-pages](#).

7.2 TensorFlow Lite

TensorFlow provides an open source library that can be used to develop and train machine learning models. TensorFlow was developed by the folks at Google and released under an open source license. It includes libraries that can be used with Python, C++, or Java. While the standard TensorFlow package has large footprint, a “lite” version is also available that is suitable for running on the Raspberry Pi. To install a “lite” version of TensorFlow for Python on the Raspberry Pi, type:

```
python3 -m pip install tflite-runtime
```

We can now use TensorFlow Lite to perform tasks like image recognition. While it is possible to collect your own image set and train your own models, there are a variety of pre-trained models available. One such model is the [MobileNet v1 model](#) which is a good demonstration of a model that is trained to recognize 1,000 different objects. To run a demo using this model, follow the instructions on the [TensorFlow Lite Python image classification demo page](#).

For more information, consult the [TensorFlow Lite documentation](#).

8 Technical Tools for Computer Scientists

The Raspberry Pi provides a variety of tools for computer scientists, ranging from document preparation to software version control to a variety of mathematical and simulation tools.

8.1 Document Preparation

8.1.1 LaTeX

LaTeX is a document preparation system for high-quality typesetting and for creating “beautiful documents.” It is often used for making mathematical, technical or scientific documents, but it can be used for almost any type of document. There are numerous resources available on the web describing how to edit a LaTeX document file. To install LaTeX and its supporting files and utilities, type the following:

```
sudo apt-get install texlive
```

You can edit your LaTeX document using any plain text editor and saving the file with a `.tex` file extension. Once you have prepared your LaTeX file, you can create a DVI (Device Independent file format) file. To do this, run the following from the command line:

```
latex file.tex
```

If your document contains errors, LaTeX prints a message, but if LaTeX was successful it should generate a DVI (Device Independent) file along with some other files. The DVI file can then be converted to a PDF file as follows:

```
dvipdf file.dvi
```

The PDF file is a platform independent file format that can then be shared widely.

8.1.2 pandoc

Pandoc is the “swiss army knife” of document conversion—a utility that can convert documents between a multitude of different formats. As such, `pandoc` is a convenient utility for the creation of PDF or HTML files from markdown (in fact, this book is written in markdown using pandoc!). To convert a markdown file to a PDF, type:

```
pandoc -o output.pdf input.md
```

Likewise, to convert from markdown to HTML, type:

```
pandoc -s -o output.html input.md
```

where the `-s` parameter indicates that the output should produce a standalone document (ie. a complete HTML document with `<head>` and `<body>` sections).

For more details, consult the [pandoc documentation](#).

8.1.3 PDF Utilities

The PDF Toolkit (`pdftk`) is a utility for PDF manipulations in Linux. To install `pdftk`, type:

```
sudo apt install pdftk
```

Once installed, `pdftk` can be used to perform many different PDF operations from the command line. For example, to join two files, `in1.pdf` and `in2.pdf` into a new PDF named `output.pdf`, type:

```
pdftk in1.pdf in2.pdf cat output.pdf
```

To remove page 13 from `in.pdf` and create a new file called `out.pdf`:

```
pdftk in.pdf cat 1-12 14-end output out.pdf
```

To find out more features in `pdftk`, type:

```
pdftk --help
```

8.2 File utilities

In addition to text editors, Linux includes a wide variety of nifty command line tools for working with files, including source code files. Here are descriptions of a few of those utilities.

8.2.1 Diff

A utility for comparing text files (including source code files) line-by-line is the `diff` utility. To compare two files, type:

```
diff -color file1.py file2.py
```

This will show all the differences (the “diff”) between the two files, `file1.py` and `file2.py`.

8.2.2 Grep

Grep is a classic command to search files for the occurrence of a string of characters that matches a specified pattern. For example, to search a file named `file.txt` for the word “raspberry”, type:

```
grep raspberry file.txt
```

This will return all line numbers that include the string “raspberry”. `grep` can also be used with *regular expressions* to perform more complex search. For example:

```
grep "^[A-Z]" file.txt
```

will search for all lines that begin with a capital letter. To learn more, consult online documents about [regular expressions](#).

8.2.3 Hexdump

Hexdump is a file viewing utility that displays a file’s contents in hexadecimal, decimal, octal, and ascii format. This can be particularly helpful when viewing binary files. To use hexdump, type:

```
hexdump file.dat
```

where `file.dat` is a binary file that you want to view. For example, to view the binary file with the `ls` program, type:

```
hexdump /usr/bin/ls
```

In order to display the output one screen at a time, the output from `hexdump` can be piped through the `more` utility as follows:

```
hexdump /usr/bin/ls | more
```

8.2.4 Readelf

Readelf is a program to display information about executable files in Linux in ELF format (Executable and Linkable File Format). For example, to view information about the `ls` command, type:

```
readelf -hA /usr/bin/ls
```

8.3 Software Version Control Systems

Source code management systems are used to manage the revision history of a software project. A source code management system has databases that stores the changes that are made to files in a project. It can be used as a tool to allow different programmers to collaborate on a software project by allowing them to blend and merge the various changes that are made.

8.3.1 Using Git and GitHub

Git is a free and open source distributed version control system. Git enables you track changes to your code and push and pull changes to remote repositories. Git is one of the most popular systems in use today. To install git, type:

```
sudo apt install git
```

After installing git, you will need to setup your local `git` `username` and `email` as follows:

```
git config --global user.name "user_name"  
git config --global user.email "email_id"
```

It is possible to setup your own private `git` server (on a Raspberry Pi or other computer). This can be accomplished by setting up an SSH server and creating a `git` user and uploading public SSH keys to allow remote users to access the server. With proper SSH keys installed, remote users can create and push changes to a repository located on the server.

Another option is to use a publicly available cloud-based `git` server, such as [GitHub](#), [GitLab](#), or [BitBucket](#). Many services include a basic free tier with limited storage or free options for students and educators.

8.3.1.1 Example git project

To initialize a new `git` project, type:

```
git init new-project
```

where `new-project` is the name of the project you wish to create. A new folder will with the project name will be created. To enter the new project folder, type:

```
cd new-project
```

At this point, you can create and edit your project files. One project file that is recommended to be created is one called `README.md`, a markdown file that contains information about the project. If we create a new file called `README.md`, we can add it to the repository as follows:

```
git add README.md
```

Likewise, we can add other files that belong to the project using the `git add` command. Once all the files are added, we can *commit* it to our repository. Use the command:

```
git commit -m "some_message"
```

The project is now ready to be uploaded to a remote repository in the cloud. For example, if you are using [github](#), goto [GitHub.com](#) and create a new repository. Get the repository **URL** and return to your local terminal and type:

```
git remote add origin REMOTE-URL
```

where `user_name` is your user name on your remote server and `REMOTE-URL` is the project URL on the server. To verify that you have set the remote **URL** correctly, type:

```
git remote -v
```

Once the remote **URL** is properly set, you can “push” all the local changes to the remote repository by typing:

```
git push -u origin main
```

Each time you reach a point where you want to check in the changes to your project, you can now *commit* and *push* the project and it will be saved in the remote `git` repository.

Alternately, you can *clone* an existing `git` project from the cloud using:

```
git clone REMOTE-URL
```

where `REMOTE-URL` is the project URL on the remote server.

8.3.1.2 Git Integration with VScode VScode extensions for GitHub, GitLab, and various other popular git servers exist. Install the appropriate extension, and you should be able to push, pull, and commit changes to the remote git server from within the VScode editor.

8.3.1.3 Download an existing git project The “clone” command downloads an existing `git` repository to your local computer. To clone an existing `git` project, you will need the project **URL**. Using the **URL**, you can clone a remote project as follows:

```
git clone https://github.com/gittower/git-crash-course.git
```

For a public repo: \$ cd folder/to/clone-into/ \$ git clone <https://github.com/gittower/git-crash-course.git>

8.3.2 Mercurial Version Control

Mercurial is friendly and relatively easy to use for those who are new to software repositories. To install with Mercurial on the Raspberry Pi, type:

```
sudo apt install mercurial
```

Once installed, you can test your installation of Mercurial by typing:

```
hg
```

If this returns a list of `hg` commands that are available, then the program is installed. To get a complete list of commands, type:

```
hg help
```

Note that mercurial is invoked using the `hg` command, which also happens to be the chemical symbol for mercury on the periodic chart.

Step 2: Setup a mercurial configuration file

Before you start using mercurial, you should setup a local configuration file to tell it who you are. In local home folder, create a file called `.hgrc` using your favorite text editor with the following contents:

```
[ui]
username=Firstname Lastname username@youraddress.com
merge=internal:merge
```

In the `username` line, substitute your own name and email address. You are now ready to make your own software repository!

8.3.2.1 Example of creating a Mercurial software repository A repository is a database that contains the files in a project. But we must first indicate the files that we want in the repository and those that we are not interested in saving. Typically, we are interested in saving all the source code files. However, we are not usually interested in saving all the intermediate object files and binary files since these can always be recreated from the source files.

Within your project folder, create a new text file called `.hgignore` which will contain information about the files that should be *excluded* from the repository. A sample `.hgignore` file is show below:

```
syntax: glob
\*.class
\*.o
```

This file indicates that *.**class** files (for Java) and *.**o** object files (for C/C++) should be excluded from the repository. There is no point storing files that can be easily recreated from the source files.

To initialize the repository, issue the following command from within the project folder:

```
hg init
```

This starts a new repository for a project and creates a new **.hg** folder in the project folder. You can now start to add and edit files within the project folder. As you do, you can check the status of these files with respect to the repository as follows:

```
hg status
```

This shows the current status of the repository. Initially, all the files in the project folder will be listed with a ? in front of them indicating that they are *unmanaged*. To add them to your mercurial repository, type the following:

```
hg add
```

This adds all the files in the current folder except the ones indicated by the **.hgignore** file. Running the status command again will show that files are now part of the repository by placing an A in front of each of the filenames. To remove a file from the repository, use the following command:

```
hg remove filename
```

where **filename** is the name of the file you want to remove from the repository

Once you are done editing your changes, you can decide to *commit* the current changes to the repository. This can be accomplished by typing:

```
hg commit -m "message"
```

where you can replace the **message** string with a meaningful summary of the changes that were made. Now run the status command again:

```
hg status
```

This time, no file status are displayed because all the files have been committed to the repository. We could now delete a file as follows:

```
rm filename
```

If this is a file that was part of the repository, it can now be easily restored as follows:

```
hg revert filename
```

You can also edit `filename` and make changes. To see the differences between a file and its copy in the repository, issue the following command:

```
hg diff
```

Subsequent updates can be committed to the repository as time goes on. You can get a complete history of past commits that were performed by typing:

```
hg log
```

This command prints a brief summary of each change that was committed including the date, time and name of the user who performed the commit along with any message that was saved with the commit.

8.3.2.2 Example of using a remote repository If you setup a mercurial repository on a remote server, you can copy and update your repository over the network using SSH. For example, to get a copy of a project from the remote server copied over to your local machine, type:

```
hg clone ssh://username@server.domain/project
```

where `username` is your user name on the server, and the `project` is the folder name of the project you want to copy. You will be prompted for your password on the server. The step of entering a password can be eliminated if you setup SSH keys.

To bring in changes from a remote repository to a local repository, use the `pull` command as follows:

```
hg pull ssh://username@server.domain/path/project_folder
```

Note that this updates the repository database, but not does not (by default) touch the files in the working project directory. Instead, use the `update` command to do this as follows:

```
hg update
```

To *push* your changes to a remote repository, use the `push` command:

```
hg push ssh://username@server.domain/path/project_folder
```

Note that the `push` command updates the remote repository database, but does not update the actual working files in the project directory since other people may be working on them while the update is in progress. To update the working files in the remote project directory, issue an update on the remote server after completing the push operation.

This simple introduction should help you get started with small personal and team projects. Once you are more comfortable with the basics, check out the [Mercurial webpage](#) to learn more.

8.4 Mathematical Tools

8.5 SageMath

SageMath is a free open-source mathematics software system with many mathematical capabilities including algebra, graph theory, numerical analysis, number theory, calculus, and statistics. Recent apt repositories include a package for SageMath, but due to the computational demands of sage it is recommended you use a more recent model of the Raspberry Pi with as much memory as possible.

To install sage from the apt repositories, type the following:

```
sudo apt install sagemath
```

The install process can take a long time, so feel free to walk away from your Raspberry Pi while it is installing. Once the install is complete, you can launch the program as follows:

```
sage
```

Once sage is loaded, give it a try. For example, to add two numbers type the following at the sage prompt:

```
sage: 2 + 3
```

To plot a graph of $y=x^2$ in sage, type: sage sage: plot(x^2) The following shows a prompt to determine the symbolic integration of $\int \sin(x)dx$ along with the output from sage:

```
sage: integral(sin(x),x)
-cos(x)
```

For more information about using other powerful features found in SageMath, visit the [Sage Documentation pages](#).

8.5.1 Octave

GNU Octave is an open source mathematics program that has some similarities to Matlab. It has both a GUI mode as well as a command line interface. To install Octave, type:

```
sudo apt install octave
```

To start octave in command line mode, type:

```
octave-cli
```

Octave is particularly good at performing linear algebra operations. For example, to define a matrix like the following:

$$x = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 1 & 4 \\ 5 & 6 & 7 \end{bmatrix}$$

simply type:

```
x = [1 2 3; 2 1 4; 5 6 7]
```

To determine x^T , the transpose of x , type:

```
x'  
ans =  
1 2 5  
2 1 6  
3 4 7
```

To find the eigenvalues for the matrix, simply type:

```
eig(x)  
ans =  
11.5453  
-0.7892  
-1.7561
```

To find the determinant for the matrix, type:

```
det(x)  
ans = 16
```

To determine the inverse matrix, type:

```
inv(x)  
ans =  
-1.0625 0.2500 0.3125  
0.3750 -0.5000 0.1250  
0.4375 0.2500 -0.1875
```

Matrix multiplication can easily perform using the `*` operator as follows. For example, multiplying a matrix by its inverse should produce the identity matrix:

```
x * inv(x)  
ans =  
1.0000 -0.0000 0  
0 1.0000 0.0000  
-0.0000 0.0000 1.0000
```

Octave can also generate large row vectors of values. For instance, the command:

```
x = linspace(a ,b)
```

generates a row vector x of 100 points linearly spaced between and including a and b . For example, to create a vector with 100 time steps for $0 \leq t \leq 6\pi$, type:

```
t = linspace(0, 6*pi)
```

In graphical environments, [octave](#) can also be used to produce plots of vectors. To plot a sinusoid for the range of time steps t , type:

```
plot(t, sin(t))
```

A new plot window will appear like the one shown below.

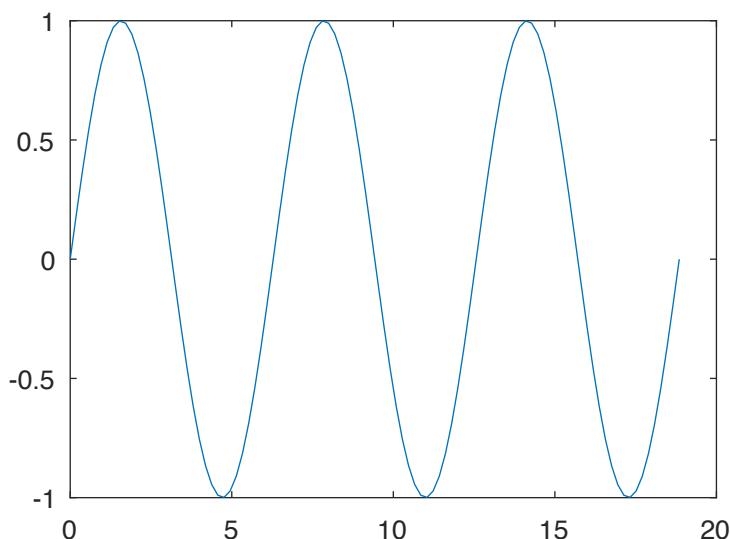


Figure 18: An octave plot of $\sin(t)$ for $0 \leq t \leq 6\pi$

For more information about [octave](#), visit the [Octave Wiki](#).

8.5.2 gnuplot

Gnuplot is a free portable command-line driven graphing utility that works on many platforms. To install [gnuplot](#), type:

```
sudo apt install gnuplot
```

To start [gnuplot](#), simply type: [gnuplot](#). You can then command [gnuplot](#) to plot various mathematical functions. While plots can be saved to image files, they can also be displayed in a graphical

window (which presupposes a graphical desktop environment). For example, to display a plot of a sine and cosine waveform, type:

```
plot [-10:10] sin(x),cos(x)
```

A $\sin(x)$ and $\cos(x)$ plot will appear for $-10 \leq x \leq 10$. For more information and demos of the features of [gnuplot](#), visit the [official gnuplot documentation page](#).

8.6 Circuit Simulation with NGSpice

NGSpice is a general-purpose, open-source circuit simulation program which is descended from SPICE, a program that began development in 1973 at the University of California at Berkeley. The name SPICE is short for “Simulation Program with Integrated Circuit Emphasis”. SPICE is an industry-standard tool capable of linear AC analysis and non-linear transient analysis of electronic circuit by solving device model equations for each of the circuit elements based on Kirchhoff’s current and voltage laws at each node.

SPICE was designed for integrated circuit simulation, but has also been used to simulate board-level circuit design. It has also been used for the [simulation of power electronics circuits](#).

To install NGSpice on the Raspberry Pi, type:

```
sudo apt install ngspice
```

8.6.1 Defining a circuit file for simulation

To begin using NGSpice, you must first define a *circuit file*, a text file that describes a *netlist* of the electronic components and the nodes to which they are connected. A *circuit file* may also contain various “dot commands” which are used to adjust various settings as well as specify the type of simulation to be performed.

In general, simple circuit elements like resistors, capacitors, and inductors are defined as using the following syntax:

Element	Definition
Resistor	R[name] [node 1] [node 2] [resistance value (Ohms)]
Capacitor	C[name] [node 1] [node 2] [capacitance value (Farads)]
Inductor	L[name] [node 1] [node 2] [inductance value (Henries)]

Note each element must have a unique name and [node 1] and [node 2] represent the node names or numbers that the element is connected to. For example, a series RC circuit, with corresponding element values 1kohm and $10\mu\text{F}$, could be represented as follows:

```
R1 1 2 1.0k
C1 2 0 10uF
```

The special node numbered “0” indicates a ground connection. NGSpice includes many other circuit elements as well, including semiconductor models for JFET, MOS, and bipolar transistors and diodes.

Circuit files can be entered using your favorite text editor. For example, consider the an RC circuit with nodes labelled 1 and 2 fed by a sinusoidal voltage source in the diagram shown below:

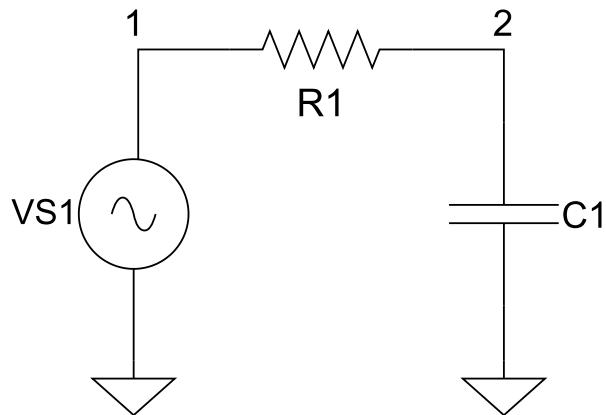


Figure 19: An RC circuit fed by a sinusoidal voltage source

Below is the circuit file corresponding to the schematic diagram shown above. This circuit file defines values to the circuit elements and initiates a transient analysis capturing the voltage at nodes 1 and 2.

```
TRANSIENT RESPONSE IN AN AC CIRCUIT WITH RC ELEMENTS
* Sinusiodal Source
VS1 1 0 SIN(0 10V 60Hz 0)

* Circuit elements
R1 1 2 1.0k
C1 2 0 10uF

.control
TRAN 100us 60ms
PRINT V(1) V(2) > output.txt
.endc
.END
```

Note the format of the circuit file includes a title on the first line and comments are included by putting an asterisk at the beginning of the line. This circuit file defines a sinusoidal voltage source (VS1) connected to two series connected circuit elements (R1 and L1). This circuit file also includes a few “dot” commands—special commands that begin with a dot and are used to adjust settings and define the type of analysis to be performed. In particular, the `.control` and `.endc` lines surround the analysis and output to be captured in the simulation. In this case, a transient analysis is performed over a time scale of 60ms with a suggested time step of $100\mu\text{s}$. The voltages of nodes 1 and 2 at each time step will be sent to a file named `output.txt`. All circuit files must conclude with a special dot command, `.END`.

8.6.2 Example Circuit Simulation

To simulate this circuit, invoke the simulator as follows:

```
ngspice -b example.cir
```

where `example.cir` is the name of the circuit file to be simulated and `-b` indicates that the simulation is to be run in *batch mode* (rather than in an interactive mode). A summary of the simulation will be printed and, as indicated above, the voltages at nodes 1 and 2 at various timesteps will be printed to `output.txt`. This output file is arranged into rows of timestamps and voltages that can be imported into a spreadsheet or a plotting package to plot the transient response.

If you are running in a graphical environment, the simulation output can also be directly plotted using `gnuplot` (described in a [previous section](#)). To add a control command to plot the transient output, add the following command in place of the `PRINT` command in the circuit file above:

```
gnuplot plot.data V(1) V(2)
```

Run the simulation again as follows:

```
ngspice -b example.cir
```

A graphical window should appear like the one below showing a plot of the voltages at nodes 1 and 2.

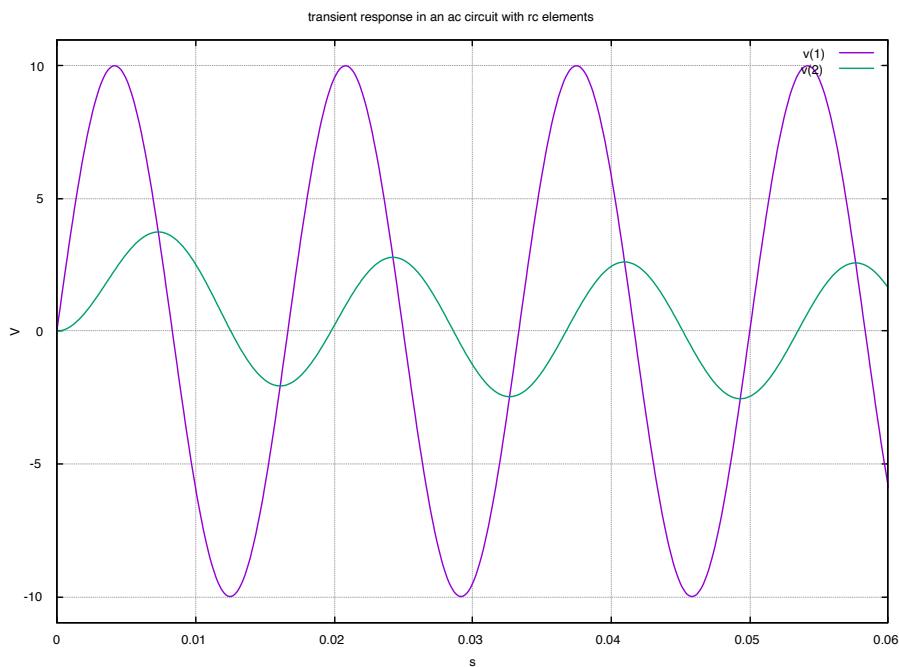


Figure 20: NGSpice voltage transient plot

For more information and to view the official [ngspice](#) manual, consult the [ngspice home page](#).

8.7 Ham Radio Applications for the Raspberry Pi

The Raspberry Pi repositories include a variety of Ham Radio programs, some of which are Additional programs included in the Raspberry Pi repositories are summarized in the table below. Many of these programs can be installed using the `apt` utility.

Program	Description
xlog	a simple logging program
trustedqsl	digitally sign logbooks for submission to Logbook of the World (LoTW)
fldigi	digital communications program for modes such as PSK31, RTTY, Olivia, and CW
flrig	a program that provides software radio control
gpredict	Satellite tracking program

For licensed amateur radio operators, WSJT can be used to exchange packets on various amateur

bands using digital modes such as FT4 and FT8. A version of WSJT for the Raspberry pi can be downloaded from the [WSJT project website](#).

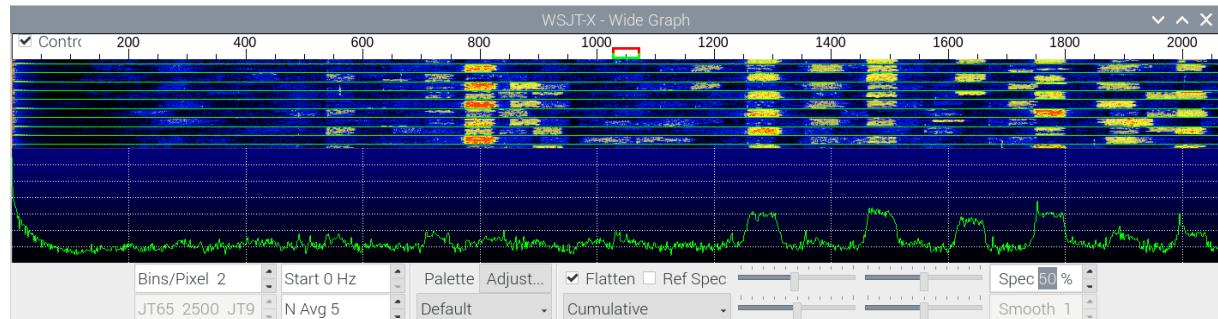


Figure 21: FT8 Spectrum “waterfall” window on WSJT program

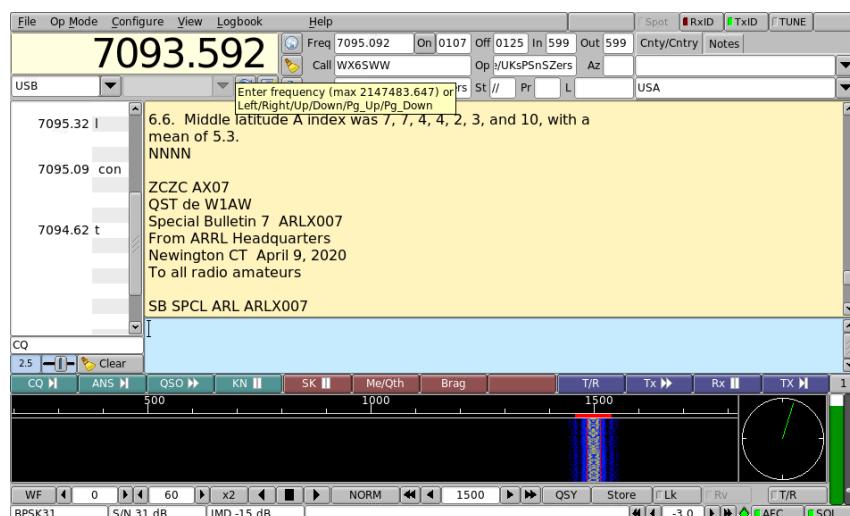


Figure 22: fldigi program window showing PSK31 transmission from station W1AW

9 Troubleshooting Tips for the Raspberry Pi

Here are some general guidelines to ensure you do not damage your Raspberry Pi or other electronic components:

- Do not remove the power from the pi until after you first perform a software shutdown of the Linux operating system. Removing the power before performing a proper shutdown could corrupt your SD card.
- Do not run your Raspberry Pi circuit board outside the case on a metal surface! The metal can short out your board and destroy it. It's best to run your Raspberry Pi in its case.

- Do not drop metal wire parts or wire clippings onto your Raspberry Pi. These could short out the circuit board causing damage.
- The GPIO pins on the Raspberry Pi use 3.3V logic. Do not connect them to a circuit powered at higher voltages (like a 5V logic circuit) which can destroy the board.
- Carefully check the GPIO pin numbers before connecting them. There are 40 pins on the header, and it is easy to make mistakes. Double check your wiring before powering up your circuit!
- Do not perform any wiring while the Raspberry Pi is powered on!
- Don't use any other power supply except the one supplied with the kit. Regular phone chargers may not have enough current to meet peak power demands.
- If you get a new SD card, make sure it has a class 10 speed rating or better.

Also, if you suspect your SD card is damaged, the Raspberry Pi 3 model B+ and later versions support booting from a USB Flash drive! You can download the Raspberry Pi OS to a regular flash drive and boot from that.

Be aware that ESD (Electrostatic Discharge) can damage electronic components. Ideally, one should typically use an antistatic mat or wriststrap when handling circuit boards and components. To reduce the risk of static damage, avoid working in carpeted areas. Also, when handling electronic components, pick them up using a non-conducting edge and try to avoid touching any pins or other connectors.

The Raspberry Pi includes a green activity LED that can aid in troubleshooting. Normally this LED flashes to indicate it is reading from the SD card. However, if an error is encountered during booting, the LED will flash an error code. The “blink codes” for the Raspberry Pi 4 comprise a series of long and short flashes as summarized below. Note that the blink codes for the Raspberry Pi 4 are different than for prior versions of the Raspberry Pi.

Long flashes	Short Flashes	Status
0	3	Generic failure to boot
0	4	start*.elf not found
0	7	Kernel image not found
0	8	SDRAM failure
0	9	Insufficient SDRAM
0	10	In HALT state
2	1	Partition not FAT
2	2	Failed to read from partition

Long flashes	Short Flashes	Status
2	3	Extended partition not FAT
2	4	File signature/hash mismatch
4	4	Unsupported board type
4	5	Fatal firmware error
4	6	Power failure type A
4	7	Power failure type B

Closing Note

This document is provided *as-is* for the purpose of providing an introduction to using the Linux command line on the Raspberry Pi. It is by no means complete, and may not reflect the latest software changes and updates. The author specifically disclaims all warranties, express or implied including, but not limited to, implied warranties of merchantability and fitness for a particular purpose. This document is provided under the terms of the *Creative Commons Attribution-NonCommercial-ShareAlike License*.

