**Techno India NJR Institute of Technology**

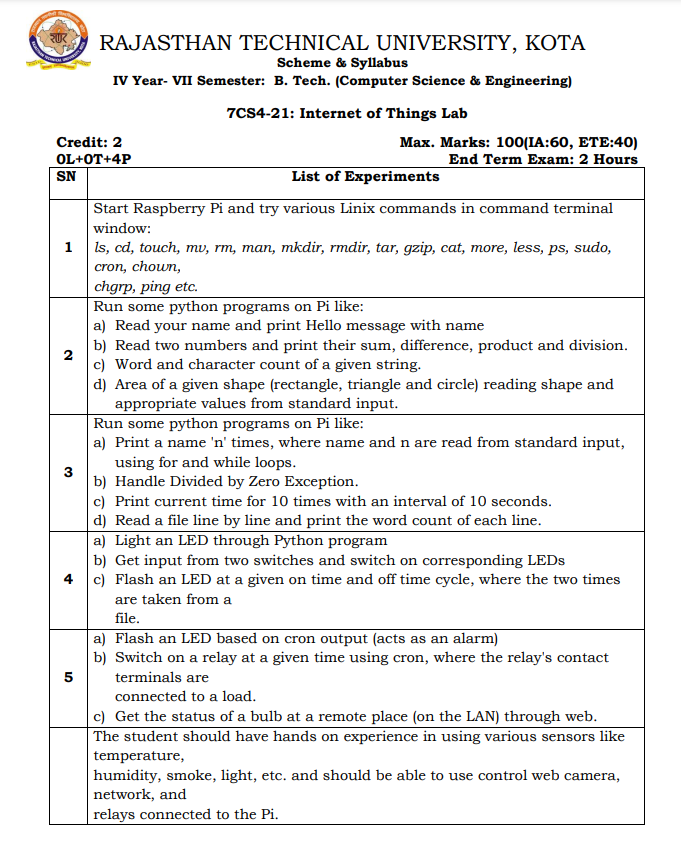


**LabManual**

**Internet of Things Lab(7CS4-21)**

Yogendra Singh Solanki

**DepartmentofECE**



**CourseOverview:**This course introduces the basics of Raspberry Pi and IoT, Python programming overview, Sensors and actuators and their automation with Raspberry Pi.This course is ideal for those who are interested in exploring the possibilities of Internet of Things using Raspberry Pi. The course assumes basic knowledge of computer hardware and software. However, knowledge of Operating system Installation and networking will help you in getting up to speed.The GPIO pins on Raspberry Pi are a great way to interface with physical devices like resistors and sensors. The implementation of IOT within the device will change the operating possibilities. If you are a technology enthusiast and want to learn the cutting-edge technology then this is the right course for you.

|  |  |  |
| --- | --- | --- |
| **CONo** | **CognitiveLevel** | **CourseOutcome(LAB)** |
| 1 | Comprehension | Student will learn to use various Linux OS related commands and shell scripting. |
| 2 | Application | Student will be able to write basic python code for Raspberry pi and illustrate how to run the python code on RPI. |
| 3 | Application | Student will be interface sensor and actuators with RPI GPIO and control them using python code. |
| 4 | Application | Student will Apply of IOT in automation of Commercial and Real-World examples |
| 5 | Application | Student will be able to Design a simple IOT systemcomprising sensors, edge devices and wireless network connections |

**Course Outcome Mapping with Program Outcome (LAB):**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CourseOutcome** | **ProgramOutcome(LAB)** | | | | | | | | | | | | | | |
| CONo. | **DomainSpecific** | | | | | **DomainIndependent** | | | | | | | **PSO** | | |
|  | **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **PO10** | **PO11** | **PO12** | **PSO1** | **PSO2** | **PSO3** |
| **CO 1** | 2 | 3 | 2 | 2 |  |  |  |  |  |  |  | 3 | 2 | 2 | 1 |
| **CO 2** | 2 | 2 | 2 | 2 |  |  |  |  | 2 |  | 2 | 2 | 2 | 2 | 1 |
| **CO 3** | 2 | 2 |  |  |  |  |  |  |  |  |  | 2 | 1 | 1 | 1 |
| **CO 4** | 2 | 3 |  | 2 |  |  |  |  |  |  |  |  | 1 |  | 1 |
| **CO 5** | 2 | 1 | 2 |  |  |  |  |  |  |  |  | 3 | 2 |  | 1 |
| **1:Slight(Low),2:Moderate(Medium),3:Substantial(high)** | | | | | | | | | | | | | | | |

**Experiment: - 1**

**Aim: -** Start Raspberry Pi and try various Linux commands in command terminal window: ls, cd, touch, mv, rm, man, mkdir, rmdir, tar, gzip, cat, more, less, ps, sudo, cron, chown, chgrp, ping etc.

**Apparatus Required: -** Raspberry Pi Board, SD Card, Card Reader, HDMI Cable, Micro USB Cable, LCD monitor, Keyboard, Mouse, Laptop or desktop PC.

**Exercise 1 - Preparing the OS**

The Raspberry Pi is nothing more than a CPU, GPU, and some peripheral devices. To bring the RaspberryPi into a usable state, it requires an Operating System - software that schedules and executes essentialroutines and user programs. The Raspberry Pi is configured to boot from an SD card - that is to say, itreads a bootable image from an SD card. Essentially, it reads in CPU instructions from the SD card thatfurther instruct it to load up the rest of the Operating System software.To put an Operating System on an SD card, you typically download a compiled image and flash it to theSD card.

**The procedure (for a computer running Windows) is as follows:**

1. Download the Raspbian image from http://downloads.raspberrypi.org/raspbian\_latest

2. Extract the .zip file

3. Install Win32DiskImager

4. Run Win32DiskImager as Administrator (from right-click menu)

5. Insert the SD card into your SD card reader, if you have not done so

6. In Win32DiskImager, “Browse” to the extracted .img file

7. Select the SD card reader as the destination device - make sure you don’t select your hard drive!

For Linux, Mac OSX, and Unix-variants, the procedure is slightly different:

1. Download the Raspbian image from http://downloads.raspberrypi.org/raspbian\_latest

2. Extract the .zip file (unzip -x file.zip)

3. Insert the SD card if you have not done so

4. use dmesg | tail to determine the path to your SD card (should be /dev/sdX )

5. use sudo dd bs=4M if=/path/to/rpi.img of=/dev/(SD Card) to flash the image - this will take

a while.

Finally, plug your Raspberry Pi into a power supply (micro-B USB cable), keyboard, and monitor, and youshould see the Raspberry Pi logo and some scrolling text on the screen.

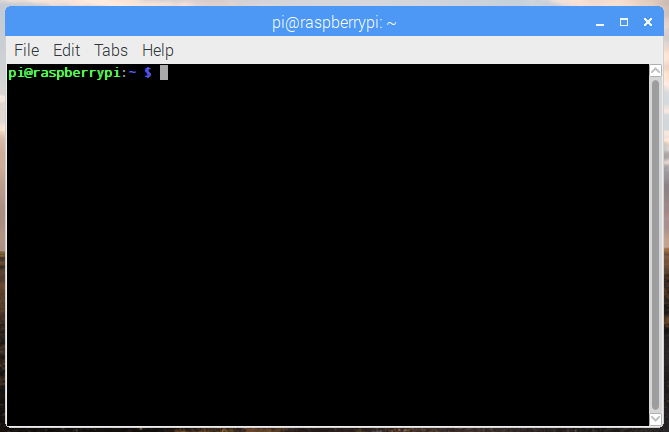
**Exercise 2 - Shell Basics& Linux Commands**

**Terminal**

The terminal (or 'command-line') on a computer allows a user a great deal of control over their system. Users of Windows may already have come across Command Prompt or Powershell, while mac OS users may be familiar with Terminal. All of these tools allow a user to directly manipulate their system through the use of commands. These commands can be chained together and/or combined together into complex [scripts](https://www.raspberrypi.com/documentation/computers/using_linux.html#shell-scripts) that can potentially complete tasks more efficiently than much larger traditional software packages.

**Opening a Terminal window**

On the Raspberry Pi OS, the default terminal application is called LXTerminal. This is known as a 'terminal emulator', this means that it emulates the old style video terminals — from before Windowing systems were developed — inside a graphical environment. The application can be found on the Raspberry Pi desktop, and when started will look something like this:



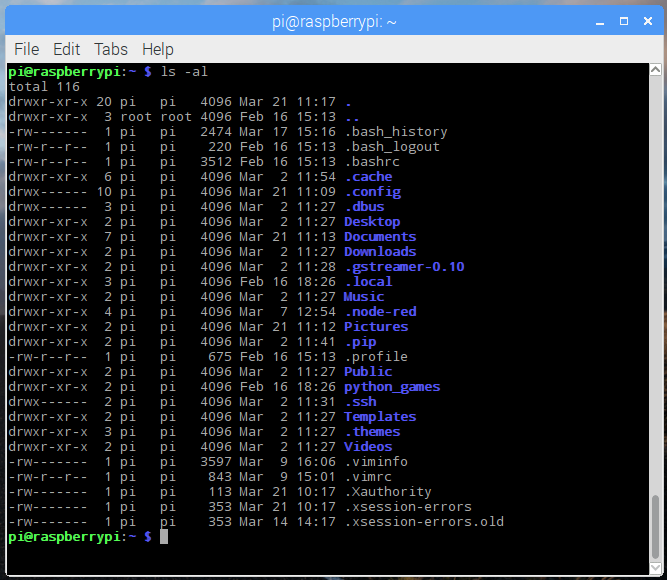
In the terminal window you should be able to see the following prompt:

pi@raspberrypi ~ $

This shows your username and the hostname of the Pi. Here the username is pi and the hostname is raspberrypi.

**Navigating and browsing your Pi**

One of the key aspects of using a terminal is being able to navigate your file system. Go ahead and type ls -la into the Terminal window, and then hit the RETURN key. You should see something similar to:



The ls command lists the contents of the directory that you are currently in (your present working directory). The -la component of the command is what’s known as a 'flag'. Flags modify the command that’s being run. In this case the l displays the contents of the directory in a list, showing data such as their sizes and when they were last edited, and the a displays all files, including those beginning with a ., known as 'dotfiles'. Dotfiles usually act as configuration files for software and as they are written in text, they can be modified by simply editing them.

In order to navigate to other directories the change directory command, cd, can be used. You can specify the directory that you want to go to by either the 'absolute' or the 'relative' path. So if you wanted to navigate to the python\_games directory, you could either do cd /home/pi/python\_games or just cd python\_games (if you are currently in /home/pi). There are some special cases that may be useful: ~ acts as an alias for your home directory, so ~/python\_games is the same as /home/pi/python\_games; . and .. are aliases for the current directory and the parent directory respectively, e.g. if you were in /home/pi/python\_games, cd .. would take you to /home/pi.

### History and auto-complete

Rather than type every command, the terminal allows you to scroll through previous commands that you’ve run by pressing the up or down keys on your keyboard. If you are writing the name of a file or directory as part of a command then pressing tab will attempt to auto-complete the name of what you are typing. For example, if you have a file in a directory called aLongFileName then pressing tab after typing a will allow you to choose from all file and directory names beginning with a in the current directory, allowing you to choose aLongFileName.

### The sudo command

Some commands that make permanent changes to the state of your system require you to have root privileges to run. The command sudo temporarily gives your account (if you’re not already logged in as root) the ability to run these commands, provided your user name is in a list of users ('sudoers'). When you append sudo to the start of a command and press enter, the command following sudo will be run using root privileges. Be very careful: commands requiring root privileges can irreparably damage your system! Note that on some systems you will be prompted to enter your password when you run a command with sudo.

Further information on sudo and the root user can be found on the [linux root page](https://www.raspberrypi.com/documentation/computers/using_linux.html#root-and-sudo).

### Installing software using apt

You can use the [apt](https://www.raspberrypi.com/documentation/computers/os.html#using-apt) command to install software in Raspberry Pi OS. This is the 'package manager' that is included with any Debian-based Linux distributions, including Raspberry Pi OS. It allows you to install and manage new software packages on your Raspberry Pi.

In order to install a new package, you would type sudo apt install <package-name>, where <package-name> is the package that you want to install.

Running sudo apt update will update a list of software packages that are available on your system. If a new version of a package is available, then sudo apt full-upgrade will update any old packages to the new version.

Finally, sudo apt remove <package-name> removes or uninstalls a package from your system.

### Other useful commands

There are a few other commands that you may find useful, these are listed below:

* cp makes a copy of a file and places it at the specified location (essentially doing a 'copy-paste'), for example - cp file\_a /home/other\_user/ would copy the file file\_a from your home directory to that of the user other\_user (assuming you have permission to copy it there). Note that if the target is a folder, the filename will remain the same, but if the target is a filename, it will give the file the new name.
* mv moves a file and places it at the specified location (so where cp performs a 'copy-paste', mv performs a 'cut-paste'). The usage is similar to cp, so mv file\_a /home/other\_user/ would move the file file\_a from your home directory to that of the specified user. mv is also used to rename a file, i.e. move it to a new location, e.g. mv hello.txt story.txt.
* rm removes the specified file (or directory when used with -r). Warning: Files deleted in this way are generally not restorable.
* mkdir: This makes a new directory, e.g. mkdirnew\_dir would create the directory new\_dir in the present working directory.
* cat lists the contents of files, e.g. cat some\_file will display the contents of some\_file.

Other commands you may find useful can be found in the [commands page](https://www.raspberrypi.com/documentation/computers/using_linux.html#linux-commands).

### Finding out about a command

To find out more information about a particular command then you can run the man followed by the command you want to know more about (e.g. man ls). The man-page (or manual page) for that command will be displayed, including information about the flags for that program and what effect they have. Some man-pages will give example usage.

## The Linux File System

It is important to have a basic understanding of the fundamentals of the Linux file system: where your files are kept, where software is installed, where the danger zones are, and so on. For more information, please refer to the Linux [Filesystem Hierarchy Standard](https://wiki.debian.org/FilesystemHierarchyStandard).

### Home

When you log into a Pi and open a terminal window, or you boot to the command line instead of the graphical user interface, you start in your home folder; this is located at /home/pi, assuming your username is pi.

This is where the user’s own files are kept. The contents of the user’s desktop is in a directory here called Desktop, along with other files and folders.

To navigate to your home folder on the command line, simply type cd and press Enter. This is the equivalent of typing cd /home/pi, where pi is your username. You can also use the tilde key (~), for example cd ~, which can be used to relatively link back to your home folder. For instance, cd ~/Desktop/ is the same as cd /home/pi/Desktop.

Navigate to /home/ and run ls, and you’ll see the home folders of each of the users on the system.

Note that if logged in as the root user, typing cd or cd ~ will take you to the [root](https://www.raspberrypi.com/documentation/computers/using_linux.html#root-and-sudo) user’s home directory; unlike normal users, this is located at /root/ not /home/root/.

## Linux Commands

Here are some fundamental and common Linux commands with example usage:

### Filesystem

#### ls

The ls command lists the content of the current directory (or one that is specified). It can be used with the -l flag to display additional information (permissions, owner, group, size, date and timestamp of last edit) about each file and directory in a list format. The -a flag allows you to view files beginning with . (i.e. dotfiles).

#### cd

Using cd changes the current directory to the one specified. You can use relative (i.e. cd directoryA) or absolute (i.e. cd /home/pi/directoryA) paths.

#### pwd

The pwd command displays the name of the present working directory: on a Raspberry Pi, entering pwd will output something like /home/pi.

#### mkdir

You can use mkdir to create a new directory, e.g. mkdirnewDir would create the directory newDir in the present working directory.

#### rmdir

To remove empty directories, use rmdir. So, for example, rmdiroldDir will remove the directory oldDir only if it is empty.

#### rm

The command rm removes the specified file (or recursively from a directory when used with -r). Be careful with this command: files deleted in this way are mostly gone for good!

#### cp

Using cp makes a copy of a file and places it at the specified location (this is similar to copying and pasting). For example, cp ~/fileA /home/otherUser/ would copy the file fileA from your home directory to that of the user otherUser (assuming you have permission to copy it there). This command can either take FILE FILE (cp fileAfileB), FILE DIR (cp fileA /directoryB/) or -r DIR DIR (which recursively copies the contents of directories) as arguments.

#### mv

The mv command moves a file and places it at the specified location (so where cp performs a 'copy-paste', mv performs a 'cut-paste'). The usage is similar to cp. So mv ~/fileA /home/otherUser/ would move the file fileA from your home directory to that of the user otherUser. This command can either take FILE FILE (mv fileAfileB), FILE DIR (mv fileA /directoryB/) or DIR DIR (mv /directoryB /directoryC) as arguments. This command is also useful as a method to rename files and directories after they’ve been created.

#### touch

The command touch sets the last modified time-stamp of the specified file(s) or creates it if it does not already exist.

#### cat

You can use cat to list the contents of file(s), e.g. cat thisFile will display the contents of thisFile. Can be used to list the contents of multiple files, i.e. cat \*.txt will list the contents of all .txt files in the current directory.

#### head

The head command displays the beginning of a file. Can be used with -n to specify the number of lines to show (by default ten), or with -c to specify the number of bytes.

#### tail

The opposite of head, tail displays the end of a file. The starting point in the file can be specified either through -b for 512 byte blocks, -c for bytes, or -n for number of lines.

#### chmod

You would normally use chmod to change the permissions for a file. The chmod command can use symbols u (user that owns the file), g (the files group) , and o (other users) and the permissions r (read), w (write), and x (execute). Using chmodu+x filename will add execute permission for the owner of the file.

#### chown

The chown command changes the user and/or group that owns a file. It normally needs to be run as root using sudo e.g. sudochownpi:root filename will change the owner to pi and the group to root.

#### ssh

ssh denotes the secure shell. Connect to another computer using an encrypted network connection. For more details see [SSH (secure shell)](https://www.raspberrypi.com/documentation/computers/remote-access.html#ssh)

#### scp

The scp command copies a file from one computer to another using ssh. For more details see [SCP (secure copy)](https://www.raspberrypi.com/documentation/computers/remote-access.html#using-secure-copy)

#### sudo

The sudo command enables you to run a command as a superuser, or another user. Use sudo -s for a superuser shell. For more details see [Root user / sudo](https://www.raspberrypi.com/documentation/computers/using_linux.html#root-and-sudo)

#### dd

The dd command copies a file converting the file as specified. It is often used to copy an entire disk to a single file or back again. So, for example, dd if=/dev/sdd of=backup.img will create a backup image from an SD card or USB disk drive at /dev/sdd. Make sure to use the correct drive when copying an image to the SD card as it can overwrite the entire disk.

#### df

Use df to display the disk space available and used on the mounted filesystems. Use df -h to see the output in a human-readable format using M for MBs rather than showing number of bytes.

#### unzip

The unzip command extracts the files from a compressed zip file.

#### tar

Use tar to store or extract files from a tape archive file. It can also reduce the space required by compressing the file similar to a zip file.

To create a compressed file, use tar -cvzf filename.tar.gz directory/ To extract the contents of a file, use tar -xvzf filename.tar.gz

#### pipes

A pipe allows the output from one command to be used as the input for another command. The pipe symbol is a vertical line |. For example, to only show the first ten entries of the ls command it can be piped through the head command ls | head

#### tree

Use the tree command to show a directory and all subdirectories and files indented as a tree structure.

#### &

Run a command in the background with &, freeing up the shell for future commands.

#### wget

Download a file from the web directly to the computer with wget. So wget https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf will download the Raspberry Pi 4 datasheet and save it as raspberry-pi-4-datasheet.pdf.

#### curl

Use curl to download or upload a file to/from a server. By default, it will output the file contents of the file to the screen.

#### man

Show the manual page for a file with man. To find out more, run man man to view the manual page of the man command.

### Search

#### grep

Use grep to search inside files for certain search patterns. For example, grep "search" \*.txt will look in all the files in the current directory ending with .txt for the string search.

The grep command supports regular expressions which allows special letter combinations to be included in the search.

#### awk

awk is a programming language useful for searching and manipulating text files.

#### find

The find command searches a directory and subdirectories for files matching certain patterns.

#### whereis

Use whereis to find the location of a command. It looks through standard program locations until it finds the requested command.

### Networking

#### ping

The ping utility is usually used to check if communication can be made with another host. It can be used with default settings by just specifying a hostname (e.g. ping raspberrypi.com) or an IP address (e.g. ping 8.8.8.8). It can specify the number of packets to send with the -c flag.

#### nmap

nmap is a network exploration and scanning tool. It can return port and OS information about a host or a range of hosts. Running just nmap will display the options available as well as example usage.

#### hostname

The hostname command displays the current hostname of the system. A privileged (super) user can set the hostname to a new one by supplying it as an argument (e.g. hostname new-host).

#### ifconfig

Use ifconfig to display the network configuration details for the interfaces on the current system when run without any arguments (i.e. ifconfig). By supplying the command with the name of an interface (e.g. eth0 or lo) you can then alter the configuration: check the manual page for more details.

## Linux Users

User management in Raspberry Pi OS is done on the command line. The default user is pi, and the password is raspberry. You can add users and change each user’s password.

### Changing your Password

Once you’re logged in as the pi user, it is highly advisable to use the passwd command to change the default password to improve your Pi’s security.

Enter passwd on the command line and press Enter. You’ll be prompted to enter your current password to authenticate, and then asked for a new password. Press Enter on completion and you’ll be asked to confirm it. Note that no characters will be displayed while entering your password. Once you’ve correctly confirmed your password, you’ll be shown a success message (passwd: password updated successfully), and the new password will apply immediately.

If your user has sudo permissions, you can change another user’s password with passwd followed by the user’s username. For example, sudo passwd bob will allow you to set the user bob's password, and then some additional optional values for the user such as their name. Just press Enter to skip each of these options.

#### Remove a User’s Password

You can remove the password for the user bob with sudo passwd bob -d. Without a password the user will not be able to login to a Terminal.

|  |  |
| --- | --- |
| **NOTE** | This is useful for users that need to exist for system reasons, but you don’t want it to be possible to login to the account for security reasons. |

### Creating a New User

You can create additional users on your Raspberry Pi OS installation with the adduser command.

Enter sudoadduser bob and you’ll be prompted for a password for the new user bob. Leave this blank if you don’t want a password.

#### Your Home Folder

When you create a new user, they will have a home folder in /home/. The pi user’s home folder is at /home/pi/.

##### The skel Command

Upon creating a new user, the contents of /etc/skel/ will be copied to the new user’s home folder. You can add or modify dot-files such as the .bashrc in /etc/skel/ to your requirements, and this version will be applied to new users.

### Deleting a User

You can delete a user on your system with the command userdel. Apply the -r flag to remove their home folder too:

sudouserdel -r bob

## Root and Sudo

*Edit this*[*on GitHub*](https://github.com/raspberrypi/documentation/blob/develop/documentation/asciidoc/computers/using_linux/linux-root-user.adoc)

The Linux operating system is a multi-user operating system which allows multiple users to log in and use the computer. To protect the computer (and the privacy of other users), the users' abilities are restricted.

Most users are allowed to run most programs, and to save and edit files stored in their own home folder. Normal users are not normally allowed to edit files in other users' folders or any of the system files. There’s a special user in Linux known as the superuser, which is usually given the username root. The superuser has unrestricted access to the computer and can do almost anything.

### The sudo Command

You won’t normally log into the computer as root, but you can use the sudo command to provide access as the superuser. If you log into your Raspberry Pi as the pi user, then you’re logging in as a normal user. You can run commands as the root user by using the sudo command before the program you want to run.

For example, if you want to install additional software on Raspberry Pi OS then you normally use the apt tool. To update the list of available software, you need to prefix the apt command with sudo:

sudo apt update

You can also run a superuser shell by using sudosu. When running commands as a superuser there’s nothing to protect against mistakes that could damage the system. It’s recommended that you only run commands as the superuser when required, and to exit a superuser shell when it’s no longer needed.

### The Sudo’ers List

The default pi user on Raspberry Pi OS is a member of the sudo group. This gives the ability to run commands as root when preceded by sudo, and to switch to the root user with sudosu.

To add a new user to the sudo group, use the adduser command:

sudoadduser bob sudo

Note that the user bob will be prompted to enter their password when they run sudo. This differs from the behaviour of the pi user, since pi is not prompted for their password. If you wish to remove the password prompt from the new user, create a custom sudoers file and place it in the /etc/sudoers.d directory.

1. Create the file using sudovisudo /etc/sudoers.d/010\_bob-nopasswd.
2. Insert the following contents on a single line: bob ALL=(ALL) NOPASSWD: ALL
3. Save the file and exit.

Once you have exited the editor, the file will be checked for any syntax errors. If no errors were detected, the file will be saved and you will be returned to the shell prompt. If errors were detected, you will be asked 'what now?' Press the 'enter' key on your keyboard: this will bring up a list of options. You will probably want to use 'e' for '(e)ditsudoers file again', so you can edit the file and fix the problem.

|  |  |
| --- | --- |
| **NOTE** | Choosing option 'Q' will save the file with any syntax errors still in place, which makes it impossible for any user to use the sudo command. |
| **NOTE** | It is standard practice on Linux to have the user prompted for their password when they run sudo, since it makes the system slightly more secure. |

## The `.bashrc ` File

In your home folder you will find a hidden file called .bashrc which contains some user configuration options. You can edit this file to suit your needs. Changes made in this file will be actioned the next time a terminal is opened, since that is when the .bashrc file is read.

If you want your changes to take place in your current terminal, you can use either source ~/.bashrc or exec bash. These actually do slightly different things: the former simply re-executes the .bashrc file, which may result in undesirable changes to things like the path, the latter replaces the current shell with a new bash shell, which resets the shell back to the state at login, throwing away any shell variables you may have set. Choose whichever is most appropriate.

Some useful adaptions are provided for you; some of these are commented out with a by default. To enable them, remove the and they will be active next time you boot your Pi or start a new terminal.

For example, some ls aliases:

alias ls='ls --color=auto'

#aliasdir='dir --color=auto'

#aliasvdir='vdir --color=auto'

alias grep='grep --color=auto'

alias fgrep='fgrep --color=auto'

alias egrep='egrep --color=auto'

Aliases like these are provided to help users of other systems like Microsoft Windows (dir is the ls of DOS/Windows). Others are to add colour to the output of commands like ls and grep by default.

More variations of ls are also provided:

# some more ls aliases

#aliasll='ls -l'

#alias la='ls -A'

#alias l='ls -CF'

Ubuntu users may be familiar with these as they are provided by default on that distribution. Uncomment these lines to have access to these aliases in future.

### The .bash\_aliases File

.bashrc also contains a reference to a .bash\_aliases file, which does not exist by default. You can add it to provide a handy way of keeping all your aliases in a separate file.

if [ -f ~/.bash\_aliases ]; then

. ~/.bash\_aliases

fi

The if statement here checks the file exists before including it.

Then you just create the file .bash\_aliases and add more aliases like so:

alias gs='git status'

You can add other things directly to this file, or to another and include that file like the .bash\_aliases example above.

## Shell Scripts

Commands can be combined together in a file which can then be executed. As an example, copy the following into your favourite text editor:

#!/usr/bin/bash

while :

do

echo Raspberry Pi!

done

Save this with the name fun-script.

Before you can run it you must first make it executable; this can be done by using the change mode command chmod. Each file and directory has its own set of permissions that dictate what a user can and can’t do to it. In this case, by running the command chmod +x fun-script, the file fun-script will now be executable.

You can then run it by typing ./fun-script (assuming that it’s in your current directory).

This script infinitely loops and prints Raspberry Pi!; to stop it, press Ctrl + C. This kills any command that’s currently being run in the terminal.

## Scheduling Tasks with Cron

Cron is a tool for configuring scheduled tasks on Unix systems. It is used to schedule commands or scripts to run periodically and at fixed intervals. Tasks range from backing up the user’s home folders every day at midnight, to logging CPU information every hour.

The command crontab (cron table) is used to edit the list of scheduled tasks in operation, and is done on a per-user basis; each user (including root) has their own crontab.

### Editing the crontab File

Run crontab with the -e flag to edit the cron table:

crontab -e

|  |  |
| --- | --- |
|  |  |

### Adding a Scheduled Task

The layout for a cron entry is made up of six components: minute, hour, day of month, month of year, day of week, and the command to be executed.

# m h dommondow command

# \* \* \* \* \* command to execute

# ┬ ┬ ┬ ┬ ┬

# │ │ │ │ │

# │ │ │ │ │

# │ │ │ │ └───── day of week (0 - 7) (0 to 6 are Sunday to Saturday, or use names; 7 is Sunday, the same as 0)

# │ │ │ └────────── month (1 - 12)

# │ │ └─────────────── day of month (1 - 31)

# │ └──────────────────── hour (0 - 23)

# └───────────────────────── min (0 - 59)

For example:

0 0 \* \* \* /home/pi/backup.sh

This cron entry would run the backup.sh script every day at midnight.

### Viewing Scheduled Tasks

View your currently saved scheduled tasks with:

crontab -l

### Erase scheduled tasks

Delete all currently scheduled tasks:

crontab -r

### Running a Task on Reboot

To run a command every time the Raspberry Pi starts up, write @reboot instead of the time and date. For example:

@reboot python /home/pi/myscript.py

This will run your Python script every time the Raspberry Pi reboots. If you want your command to be run in the background while the Raspberry Pi continues starting up, add a space and & at the end of the line, like this:

@reboot python /home/pi/myscript.py &

## The systemd Daemon

In order to have a command or program run when the Pi boots, you can add it as a service. Once this is done, you can start/stop enable/disable from the linux prompt.

### Creating a Service

On your Pi, create a .service file for your service, for example: myscript.service

[Unit]

Description=My service

After=network.target

[Service]

ExecStart=/usr/bin/python3 -u main.py

WorkingDirectory=/home/pi/myscript

StandardOutput=inherit

StandardError=inherit

Restart=always

User=pi

[Install]

WantedBy=multi-user.target

So in this instance, the service would run Python 3 from our working directory /home/pi/myscript which contains our python program to run main.py. But you are not limited to Python programs: simply change the ExecStart line to be the command to start any program or script that you want running from booting.

Copy this file into /etc/systemd/system as root, for example:

sudo cp myscript.service /etc/systemd/system/myscript.service

Once this has been copied, you have to inform systemd that a new service has been added. This is done with the following command:

sudosystemctl daemon-reload

Now you can attempt to start the service using the following command:

sudosystemctl start myscript.service

Stop it using following command:

sudosystemctl stop myscript.service

When you are happy that this starts and stops your app, you can have it start automatically on reboot by using this command:

sudosystemctl enable myscript.service

The systemctl command can also be used to restart the service or disable it from boot up.

**Experiment: - 2**

**Aim: -**Run some python programs on Pi like: a) Read your name and print Hello message with name b) Read two numbers and print their sum, difference, product and division. c) Word and character count of a given string. d) Area of a given shape (rectangle, triangle and circle) reading shape and appropriate values from standard input.

**Apparatus Required: -** Raspberry Pi Board, SD Card, Card Reader, HDMI Cable, Micro USB Cable, LCD monitor, Keyboard, Mouse, Laptop or desktop PC.

**Exercise 1 –** Write a python code toread your name and print Hello message with name

The program will essentially prompt the user for information, and then when the user hits *enter*, will take whatever they entered and be able to use it going forward in it's execution.

The code for accepting user input is actually not that much more complex than the code for printing. In Python, we can get user input like this:

name = input("Enter your name: ")

print("Hello", name + "!")

The code above simply prompts the user for information, and the prints out what they entered in.

**Exercise 2–** Write a python code toread two numbers and print their sum, difference, product and division.

In this program, user is asked to input two numbers and the operator (+ for addition, – for subtraction, \* for multiplication and / for division). Based on the input, program computes the result and displays it as output.

num1 =int(input("Enter First Number: "))

num2 =int(input("Enter Second Number: "))

print("Enter which operation would you like to perform?")

ch=input("Enter any of these char for specific operation +,-,\*,/: ")

result =0

ifch=='+':

result = num1 + num2

elifch=='-':

result = num1 - num2

elifch=='\*':

result = num1 \* num2

elifch=='/':

result = num1 / num2

else:

print("Input character is not recognized!")

print(num1,ch, num2,":", result)

**Exercise 3–** Write a python code toprintword and character count of a given string.

# initializing string

test\_string**=**"Hello world"

# printing original string

**print**("The original string is : "**+**test\_string)

# using split()

# to count words in string

res **=**len(test\_string.split())

# printing result

**print**("The number of words in string are : "**+**str(res))

print("The number of characters in string are : ", len(test\_string))

**Exercise 4–** Write a python code tocalculate area of a given shape (rectangle, triangle and circle) reading shape and appropriate values from standard input.

**def**calculate\_area(name):\

  # converting all characters

  # into lower cases

  name **=**name.lower()

  # check for the conditions

**if**name **==**"rectangle":

    l **=**int(input("Enter rectangle's length: "))

    b **=**int(input("Enter rectangle's breadth: "))

    # calculate area of rectangle

    rect\_area**=**l **\***b

**print**(f"The area of rectangle **is**

          {rect\_area}.")

**elif**name **==**"square":

    s **=**int(input("Enter square's side length: "))

    # calculate area of square

    sqt\_area**=**s **\***s

    print(f"The area of square **is**

          {sqt\_area}.")

**elif**name **==**"triangle":

    h **=**int(input("Enter triangle's height length: "))

    b **=**int(input("Enter triangle's breadth length: "))

    # calculate area of triangle

    tri\_area**=**0.5**\***b **\***h

    print(f"The area of triangle **is**

          {tri\_area}.")

**elif**name **==**"circle":

    r **=**int(input("Enter circle's radius length: "))

    pi **=**3.14

    # calculate area of circle

    circ\_area**=**pi **\***r **\***r

    print(f"The area of triangle **is**

          {circ\_area}.")

**elif**name **==**'parallelogram':

    b **=**int(input("Enter parallelogram's base length: "))

    h **=**int(input("Enter parallelogram's height length: "))

    # calculate area of parallelogram

    para\_area**=**b **\***h

**print**(f"The area of parallelogram **is**

          {para\_area}.")

**else**:

**print**("Sorry! This shape is not available")

# driver code

**if**\_\_name\_\_ **==**"\_\_main\_\_":

**print**("Calculate Shape Area")

  shape\_name**=**input("Enter the name of shape whose area you want to find: ")

  # function calling

  calculate\_area(shape\_name)

**Experiment: - 3**

**Aim: -**Run some python programs on Pi like: a) Print a name ‘n’ time, where name and n are read from standard input, using for and while loops. b) Handle Divided by Zero Exception. c) Print current time for 10 times with an interval of 10 seconds. d) Read a file line by line and print the word count of each line.

**Apparatus Required: -** Raspberry Pi Board, SD Card, Card Reader, HDMI Cable, Micro USB Cable, LCD monitor, Keyboard, Mouse, Laptop or desktop PC.

**Exercise 1–** Write a python code toPrint a name ‘n’ time, where name and n are read from standard input, using for and while loops.

//To print a given Name in n Times  
a = input("Enter your name: ")  
n = int(input("Enter the number you want to print that times: "))  
i = 1  
 while i <= n:  
 print(a)  
i+=1

**Exercise 2–** Write a python code toHandle Divided by Zero Exception.

n=int(input("Enter the value of n:"))

d=int(input("Enter the value of d:"))

c=int(input("Enter the value of c:"))

try:

q=n/(d-c)

print("Quotient:",q)

except ZeroDivisionError:

print("Division by Zero!")

**Exercise 3–** Write a python code toPrint current time for 10 times with an interval of 10 seconds.

import time

from datetime import datetime

#For 10 times

for x in range(10):

# Get current time

now = datetime.now()

# Make a string of it

current\_time = now.strftime("%H:%M:%S")

# Print it

print(current\_time)

# Wait for 10 seconds

time.sleep(10)

**Exercise 4–** Write a python code to Read a file line by line and print the word count of each line.

# Python program to count

# number of words in each line of a text file separately

# Input filename, open it in

# read mode. display its cntents

# count and display number of words

# www.EasyCodeBook.com

file\_name = input("Enter file name:")

file1 = open(file\_name, "r")

word\_count = 0

i = 0

str1 = ""

print("Contents of file " + file\_name + " are:")

# display and count number of words in each line of text file

for line in file1:

i+=1

print(line, end='')

words\_in\_line = len(line.split())

str1 = str1 + "Words in Line No: " + str(i) + " are : " + str(words\_in\_line)+"\n"

word\_count+=words\_in\_line

print('\n\n ' + str1)

print('\n\nTotal Number of words in this file are = ' + str(word\_count))

file1.close()

**Experiment: - 4**

**Aim: -**a) Light an LED through Python program

b) Get input from two switches and switch on corresponding LEDs

c) Flash an LED at a given on time and off time cycle, where the two times

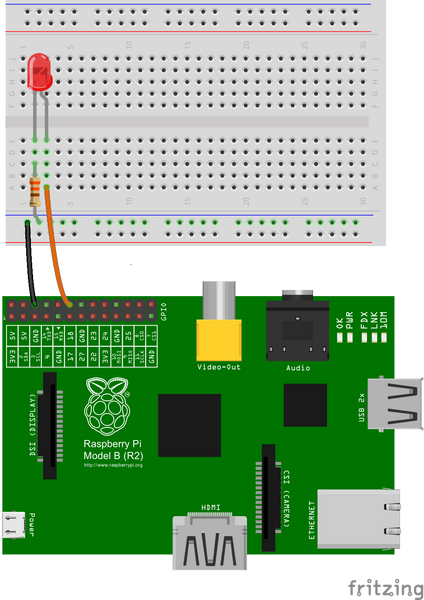
are taken from afile.

**Apparatus Required: -** Raspberry Pi Board, SD Card, HDMI Cable, Micro USB Cable, LCD monitor, Keyboard, Mouse, LED, Push Buttons, Bread Board.

**Exercise 1 –** Light an LED through Python program

The circuit consists of a power supply (the Raspberry Pi), an LED that lights when the power is applied, and a resistor to limit the current that can flow through the circuit.

You will be using one of the ‘ground’ (GND) pins to act like the ‘negative’ or 0 volt ends of a battery. The ‘positive’ end of the battery will be provided by a GPIO pin. Here we will be using pin 18. When they are ‘taken high’, which means it outputs 3.3 volts, the LED will light. Now take a look at the circuit diagram below.



You should turn your Raspberry Pi off for the next bit, just in case you accidentally short something out.

* Use one of the jumper wires to connect a ground pin to the rail, marked with blue, on the breadboard. The female end goes on the Raspberry Pi's pin, and the male end goes into a hole on the breadboard.
* Then connect the resistor from the same row on the breadboard to a column on the breadboard, as shown above.
* Next, push the LEDs legs into the breadboard, with the long leg (with the kink) on the right.
* Lastly, complete the circuit by connecting pin 18 to the right hand leg of the LED. This is shown here with the orange wire.

Turn on your Raspberry Pi and open the terminal window.

Create a new text file “LED.py” by typing the following:

nano LED.py

Type in the following code:

import RPi.GPIO as GPIO  
import time  
GPIO.setmode(GPIO.BCM)  
GPIO.setwarnings(False)  
GPIO.setup(18,GPIO.OUT)  
print "LED on"  
GPIO.output(18,GPIO.HIGH)  
time.sleep(1)  
print "LED off"  
GPIO.output(18,GPIO.LOW)

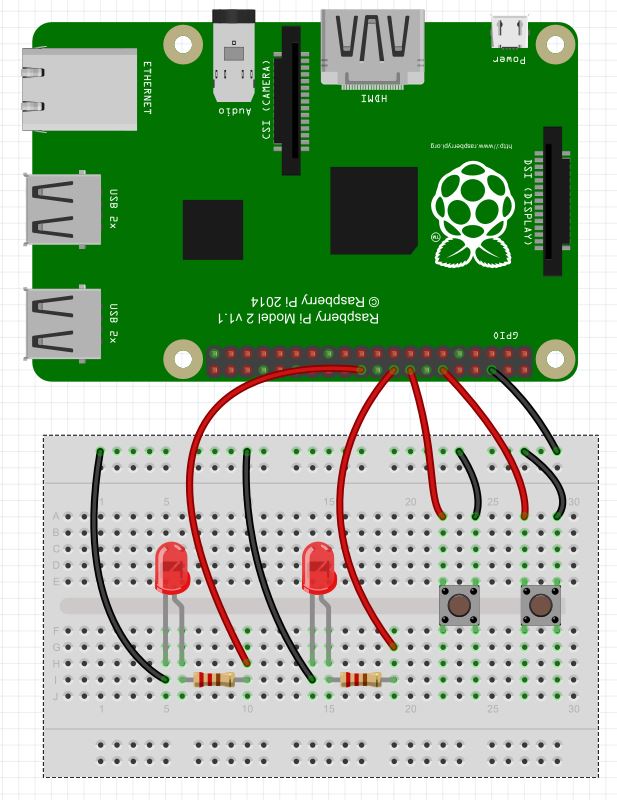
Once you have typed all the code and checked it, save and exit the text editor with “Ctrl + x” then “y” then “enter”.

**To run this code type:**

sudo python LED.py

You will see the LED turn on for a second and then turn off.

**Exercise 2 –** Get input from two switches and switch on corresponding LEDs



fromtimeimportsleep     # Import sleep Library

importRPi.GPIOasGPIO    # Import GPIO Library

GPIO.setmode(GPIO.BOARD)   # Use Physical Pin Numbering Scheme

button1=16                 # Button 1 is connected to physical pin 16

button2=12                 # Button 2 is connected to physical pin 12

LED1=22                    # LED 1 is connected to physical pin 22

LED2=18                    # LED 2 is connected to physical pin 18

GPIO.setup(button1,GPIO.IN,pull\_up\_down=GPIO.PUD\_UP)# Make button1 an input, Activate Pull UP Resistor

GPIO.setup(button2,GPIO.IN,pull\_up\_down=GPIO.PUD\_UP)# Make button 2 an input, Activate Pull Up Resistor

GPIO.setup(LED1,GPIO.OUT,)# Make LED 1 an Output

GPIO.setup(LED2,GPIO.OUT)  # Make LED 2 an Output

BS1=False                  # Set Flag BS1 to indicate LED is initially off

BS2=False                  # Set Flag BS2 to indicate LED is initially off

while(1):                  # Create an infinite Loop

        ifGPIO.input(button1)==0:            # Look for button 1 press

                print"Button 1 Was Pressed:"

                ifBS1==False:                # If the LED is off

                        GPIO.output(LED1,True)# turn it on

                        BS1=True              # Set Flag to show LED1 is now On

                        sleep(.5)             # Delay

                else:                         # If the LED is on

                        GPIO.output(LED1,False)# Turn LED off

                        BS1=False               # Set Flag to show LED1 is now Off

                        sleep(.5)

        ifGPIO.input(button2)==0: #Repeat above for LED 2 and button 2

                print"Button 2 Was Pressed:"

                ifBS2==False:

                        GPIO.output(LED2,True)

                        BS2=True

                        sleep(.5)

                else:

                        GPIO.output(LED2,False)

                        BS2=False

                        sleep(.5)

**Experiment: - 5**

**Aim: -**a) Flash an LED based on cron output (acts as an alarm)

b) Switch on a relay at a given time using cron, where the relay's contactterminals are

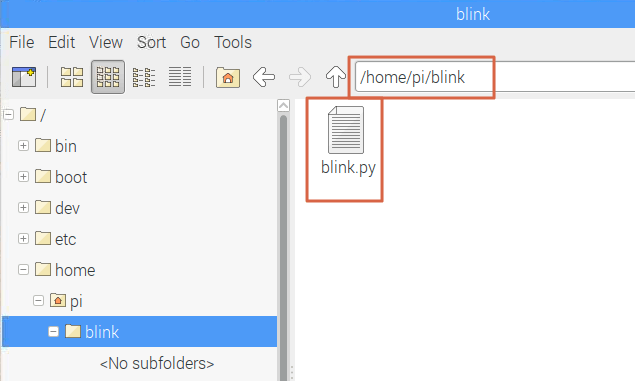
connected to a load.

c) Get the status of a bulb at a remote place (on the LAN) through web.

**Apparatus Required: -** Raspberry Pi Board, SD Card, HDMI Cable, Micro USB Cable, LCD monitor, Keyboard, Mouse, LED, Bread Board.

**Step 1:**

First, create a ‘**blink**’ folder and then create an empty ‘**blink.py**’ file. It should be inside the path /home/pi/blink. (The /home/pi/ is the default user files path).

****

**Python code: *blink.py***

import RPi.GPIO as GPIO

import time

LedPin = 11    # pin11

def setup():

  GPIO.setmode(GPIO.BOARD)       # Numbers GPIOs by physical location

  GPIO.setup(LedPin, GPIO.OUT)   # Set LedPin's mode is output

  GPIO.output(LedPin, GPIO.HIGH) # Turn ON led

def blink():

  while True:

    GPIO.output(LedPin, GPIO.HIGH)  # led on

    time.sleep(1)

    GPIO.output(LedPin, GPIO.LOW) # led off

    time.sleep(1)

def destroy():

  GPIO.output(LedPin, GPIO.LOW)   # led off

  GPIO.cleanup()                  # Release resource

if \_\_name\_\_ == '\_\_main\_\_':     # Program start from here

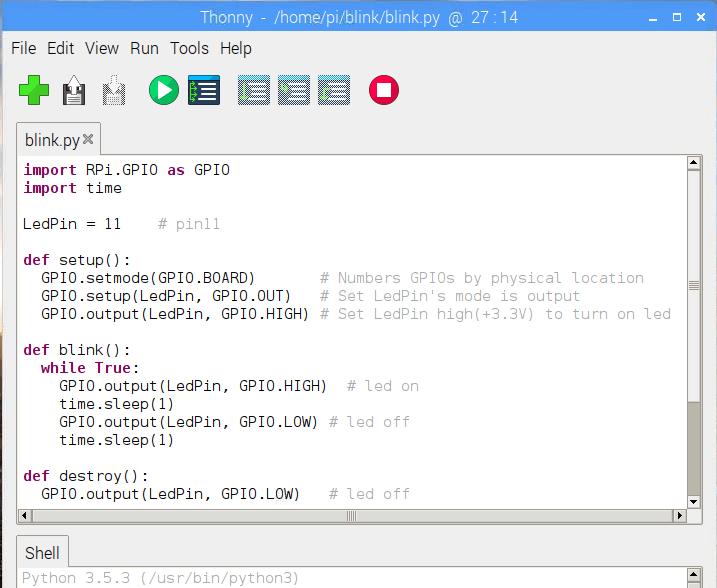
  setup()

  try:

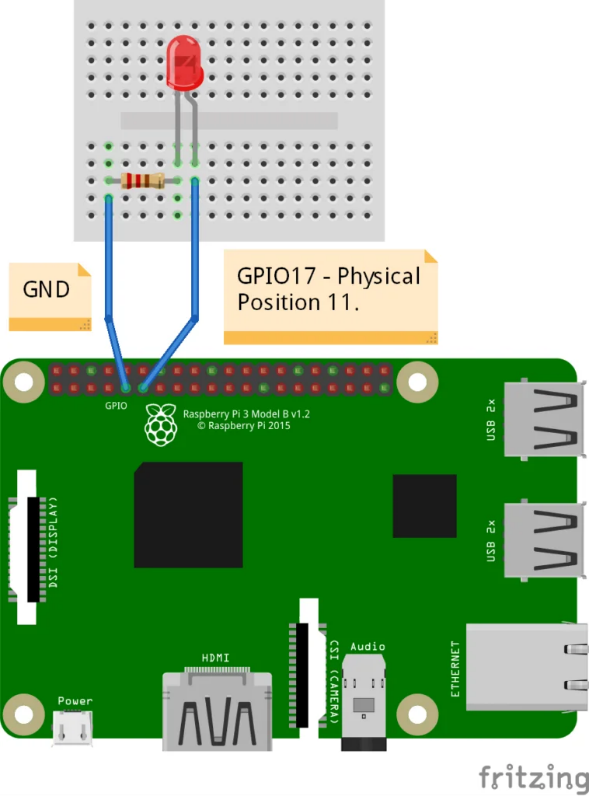
    blink()

  except KeyboardInterrupt:  # When 'Ctrl+C' is pressed, the child program destroy() will be  executed.

    destroy()



**Circuit Diagram:**



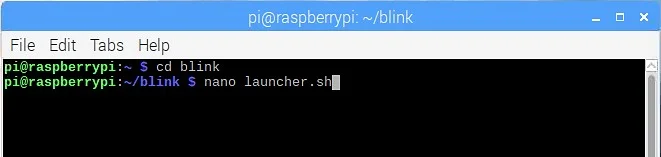
**Step 2:**

Open the terminal and navigate to the ‘blink’ folder by using the following command

cd blink

 Create a launcher script by typing the command in the terminal.

nano launcher.sh



Now the editor will get launched. Copy and paste the following code to the editor. This shell script will navigate to the blink code.

#!/bin/sh

# launcher.sh

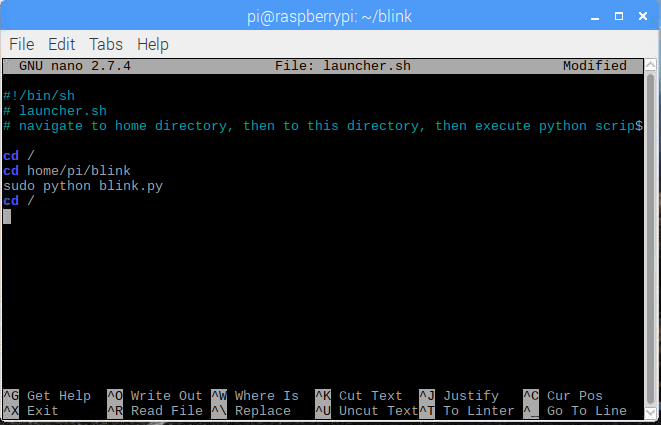
# navigate to home directory, then to this directory, then execute python script, then back home

cd /

cd home/pi/blink

sudo python blink.py

cd /

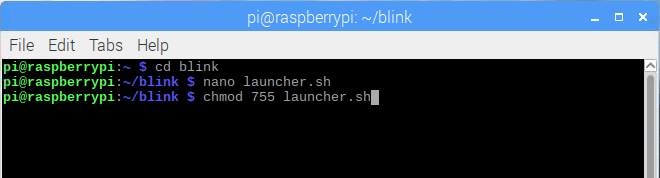


Now press Ctrl – X and the press Y to Save. It will ask for filename leave it as ‘launcher.sh’ itself and press enter.

**Step 3:**

We want to make this launcher.sh script executable. Type the following command for it.

chmod 755 launcher.sh



You can test the launcher script by typing the following command if you want to test.

sh launcher.sh

**Step 4:**

Create a log directory for capturing logs incase of any errors. Navigate back to home directory and create a log directory there.

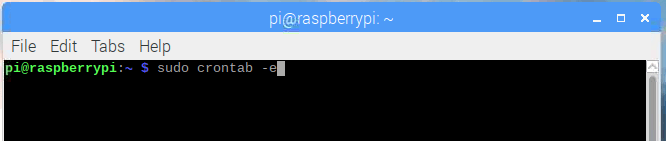
cd

mkdir logs

**Step 5:**

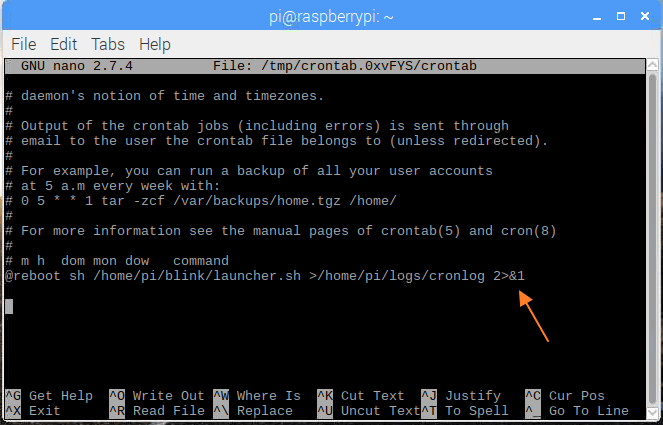
We need to schedule the launcher script to run during startup itself. Open the *crontab* by using the command.

sudo crontab –e



Once the crontab window opens add this line at the end of the file.

@rebootsh /home/pi/bbt/launcher.sh >/home/pi/logs/cronlog 2>&1



This will run the launcher script during startup. Now reboot the Pi and check the python code runs automatically. Connect the LED according to the circuit and you will see the LED blinking (blink.py runs) automatically when the Pi is powered up.You can modify the file path and file name to run a python program during startup or the line below lets you run your script at 6 AM every day, essentially giving you an alarm at 6 AM.

0 6 \* \* \* cd /home/pi/Desktop/folder\_name&& $(which python3) file\_name.py >> ~/cron.log 2>1&1