

A Project Report
On
GPIO Linux Device Driver Development

BY
Utkarsh Pareek (2020H1400223H)
Aruna (2020H1400220H)

**SUBMITTED IN FULLFILLMENT OF THE REQUIREMENTS OF
Device Drivers (EEE G547)**



**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI (RAJASTHAN)
HYDERABAD CAMPUS**

(December 2021)

CONTENTS

1. Abstract.....	3
2. Setting up Raspberry pi 3b plus.....	4
3. Hardware Required.....	12
4. Accessing the GPIO in Linux Kernel.....	12
5. Testing the device driver.....	12
6. Circuit Connection.....	13
7. VNC viewer.....	13
8. Terminal.....	14
Conclusion.....	15
References.....	16

Abstract

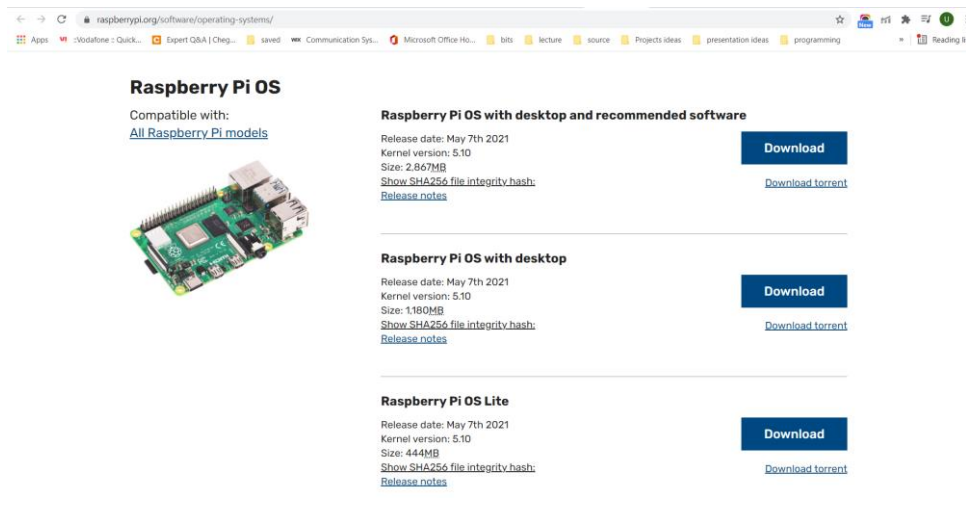
GPIO pins or General-purpose Input Output pins are digital signal pins of the raspberry pi board which can be used as input pins, output pins or interrupt pins. The GPIO pins can be configured to act as desired using the GPIO hardware registers. In this project we have designed a GPIO device driver by controlling these hardware registers. Although the raspberry pi kernel has a GPIO device class, we will make our own device class and object to learn more about the actual hardware access. We have written a driver to toggle an LED connected to a GPIO pin. The period with which the LED toggles is modified using the user-mode application via the “sysfs” interface. The system uses the software timers available in the linux kernel to toggle the LED.

Setting up Raspberry pi:

Components Required:

1. **Raspberry Pi**
2. **Micro SD card**
3. **Micro SD card adapter**
4. **Smartphone**

a) Downloading of Raspberry Pi OS with desktop:



The screenshot shows the Raspberry Pi OS download page. It features a Raspberry Pi image and three download options:

- Raspberry Pi OS**
Compatible with: [All Raspberry Pi models](#)
Release date: May 7th 2021
Kernel version: 5.10
Size: 2.867MB
[Show SHA256 file integrity hash](#)
[Release notes](#)
[Download](#) [Download torrent](#)
- Raspberry Pi OS with desktop and recommended software**
Release date: May 7th 2021
Kernel version: 5.10
Size: 1.180MB
[Show SHA256 file integrity hash](#)
[Release notes](#)
[Download](#) [Download torrent](#)
- Raspberry Pi OS Lite**
Release date: May 7th 2021
Kernel version: 5.10
Size: 444MB
[Show SHA256 file integrity hash](#)
[Release notes](#)
[Download](#) [Download torrent](#)

We will download below version:

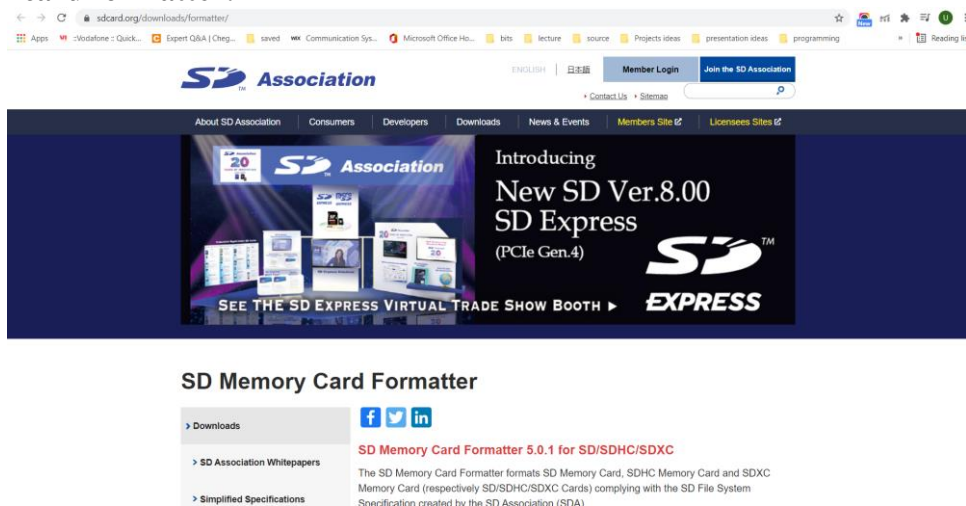
Raspberry Pi OS with desktop

Release date: May 7th 2021
Kernel version: 5.10
Size: 1,180MB
[Show SHA256 file integrity hash](#)
[Release notes](#)

[Download](#)

[Download torrent](#)

b) Download SD card formatter:



The screenshot shows the SD Association website. It features a banner for the new SD Ver.8.00 SD Express (PCIe Gen.4) and a section for the SD Memory Card Formatter.

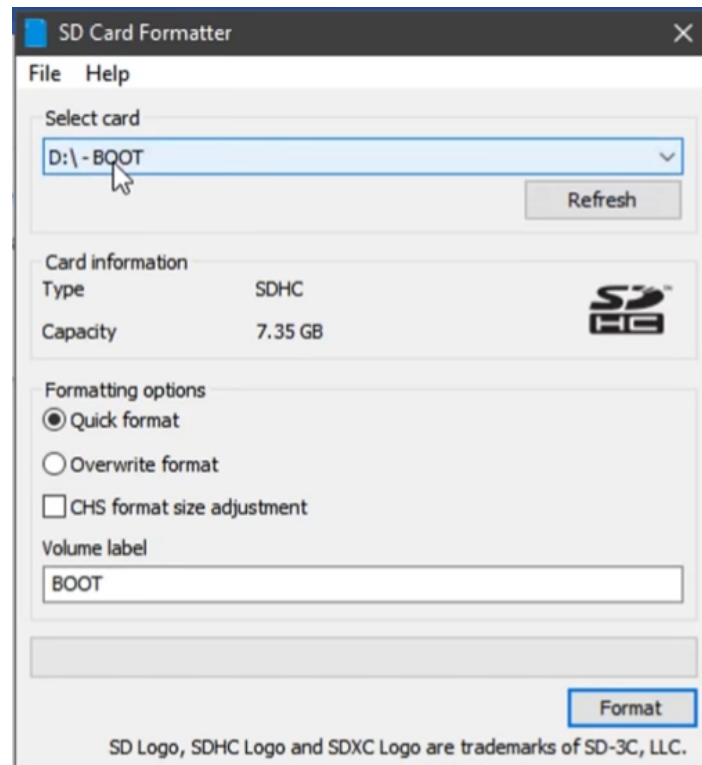
SD Memory Card Formatter

[Downloads](#) [SD Association Whitepapers](#) [Simplified Specifications](#)

[SD Memory Card Formatter 5.0.1 for SD/SDHC/SDXC](#)

The SD Memory Card Formatter formats SD Memory Card, SDHC Memory Card and SDXC Memory Card (respectively SD/SDHC/SDXC Cards) complying with the SD File System Specification created by the SD Association (SDA).

Now, install and load the SD memory card formatter in laptop and run it.



Now we will require following things:

1. Etcher
2. Advance Ip Scanner
3. Putty
4. VNC Viewer

Etcher: to write OS in SD card

Advance IP scanner: to scan IP of raspberry pi

Putty: for doing SSH (secure shell)

VNC viewer: To see raspberry pi desktop

A) Download and install etcher:



We will select portable version of windows

B) Download and install putty:



Download PuTTY

PuTTY is an SSH and telnet client, developed originally by Simon Tatham for the Windows platform. PuTTY is open source software that is available with source code and is developed and supported by a group of volunteers.

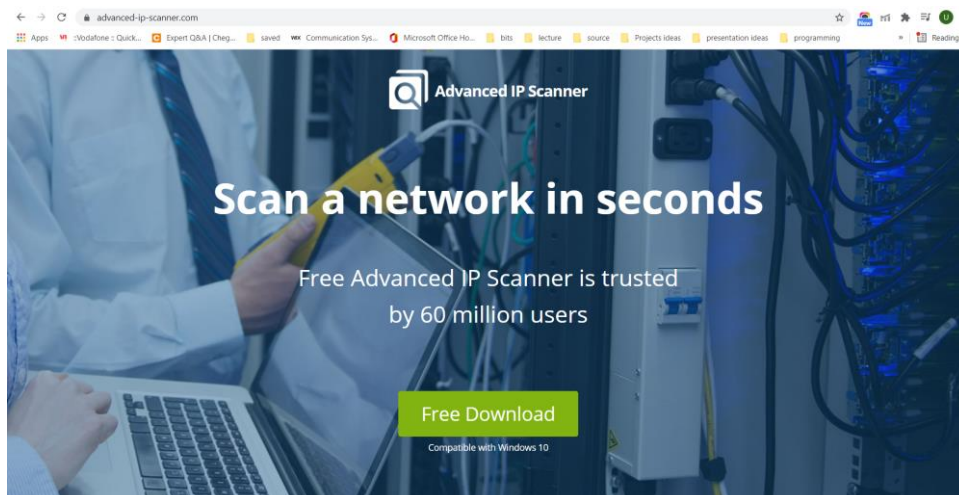
You can download PuTTY [here](#).

Now we will download for 64-bit x86:

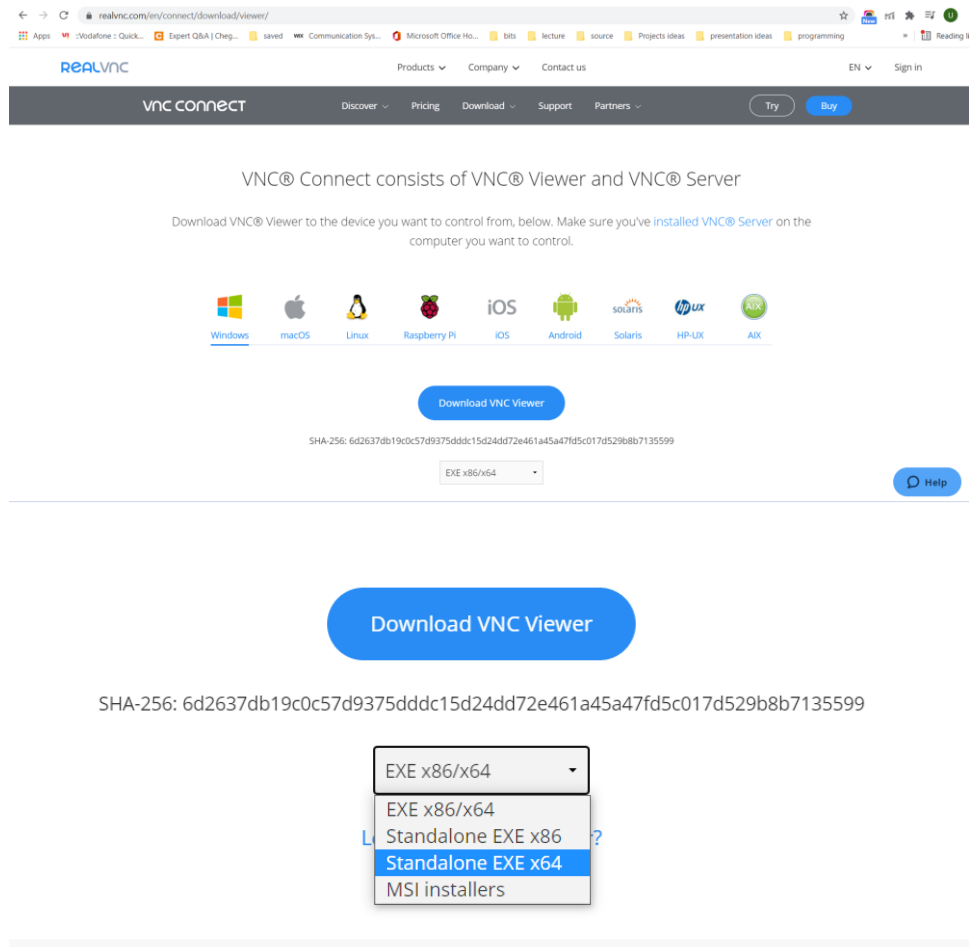
putty.exe (the SSH and Telnet client itself)

64-bit x86:	putty.exe	(or by FTP)	(signature)
64-bit Arm:	putty.exe	(or by FTP)	(signature)
32-bit x86:	putty.exe	(or by FTP)	(signature)

C) Download and install advanced IP scanner:



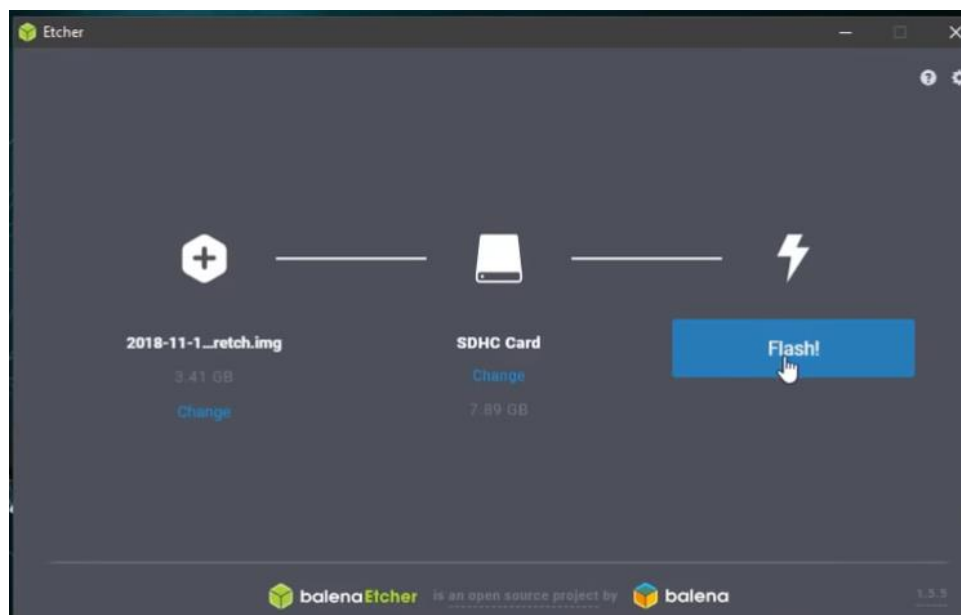
C) Download and install VNC viewer:

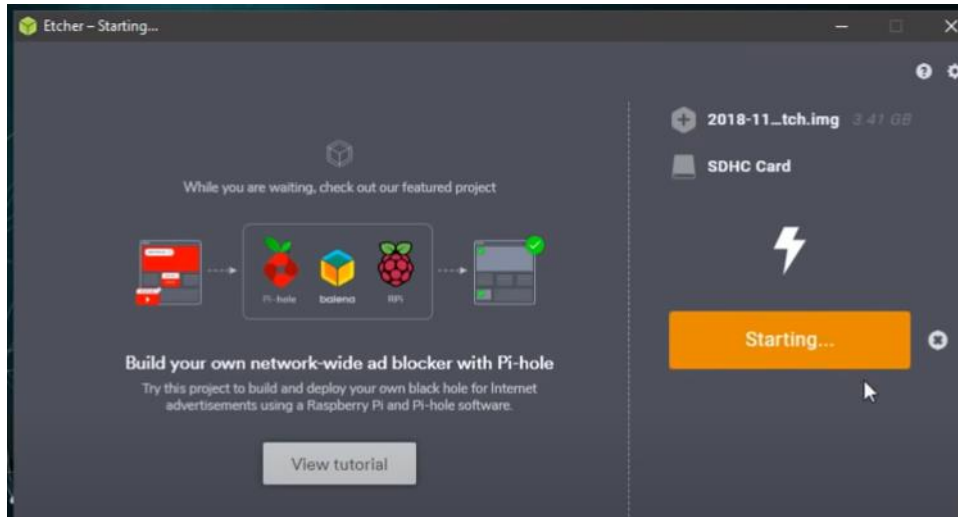


We are selecting standalone EXE x64.

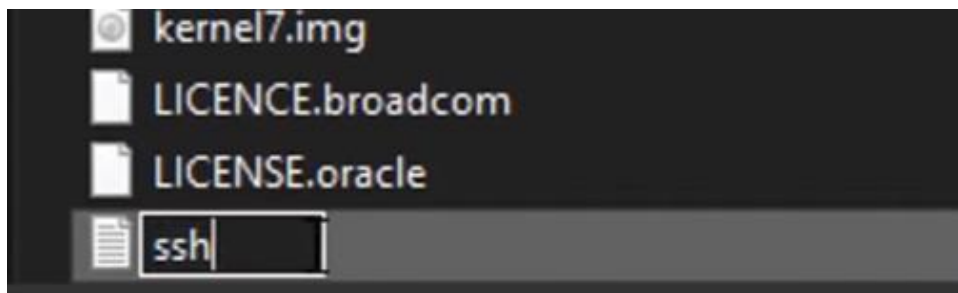
TO BURN OS INTO MEMORY CARD

Step 1: Open Etcher -> Open downloaded Raspberry Pi OS zip file -> Click on Flash



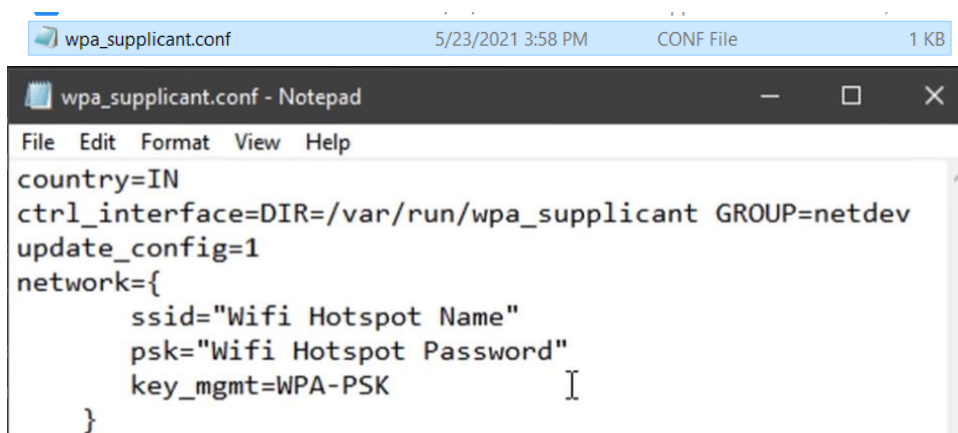


Step 2: Open This PC -> Open BOOT -> Right click -> new -> Select text doc -> Rename it to SSH and remove extension



Step 3: Turn on mobile hotspot

Step 4: Copy wpa_supplicant.conf file to SD card and change WIFI username and Password into it -> copy this file into BOOT

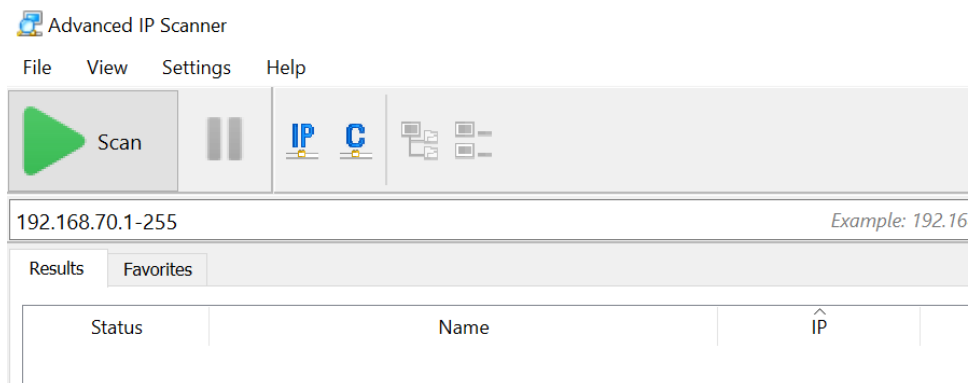


Step 5: Eject Boot

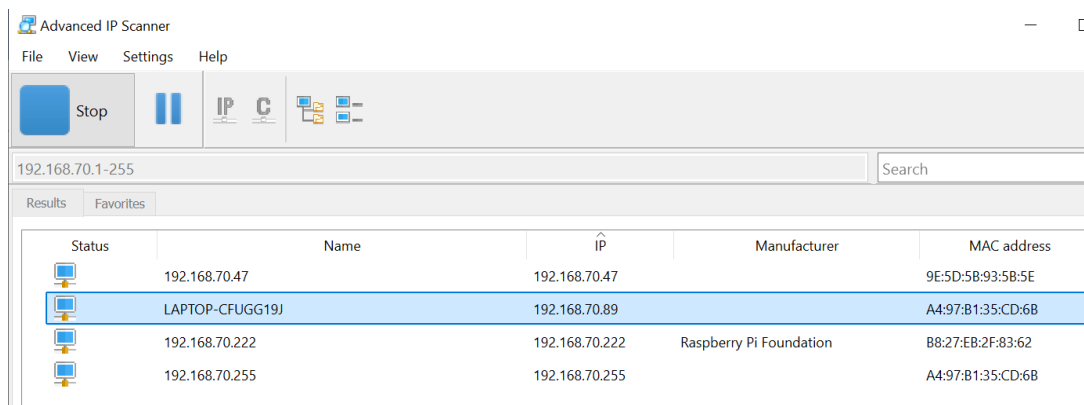
- Now, plug in memory card to raspberry pi and power on raspberry pi with USB cable using laptop's USB port.
- Wait up to 20seconds for it to Boot up.
- Connect laptop to hotspot to bring laptop and raspberry pi on same network.

Run Advanced IP scanner in portable version and follow the steps:

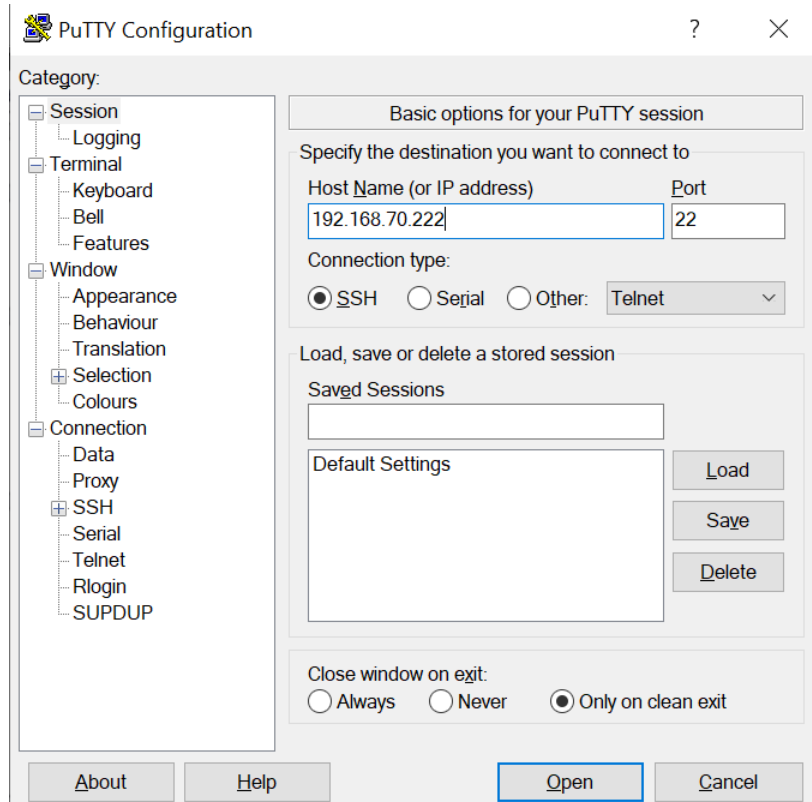
1. To get the base address, open command prompt-> Type “ipconfig” -> note the IP address and paste it in Advanced IP scanner and replace the last bit by 1-254



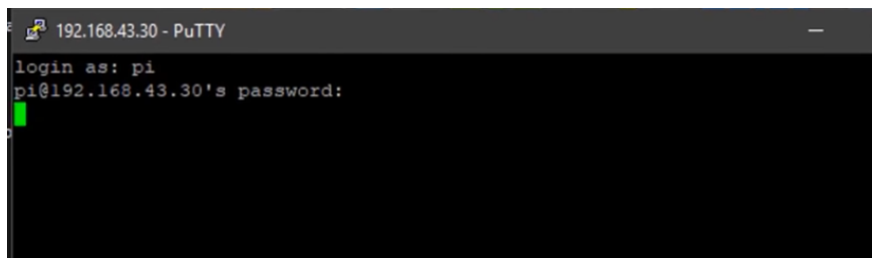
2. It will scan all the IP address mapping from 1-254 -> click on scan to get the raspberry pi IP address



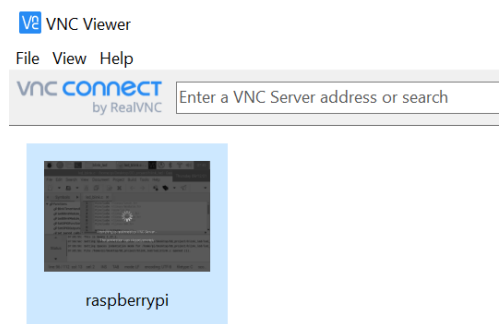
3. Copy and IP address and paste it in putty (set port to 22 and select SSH)

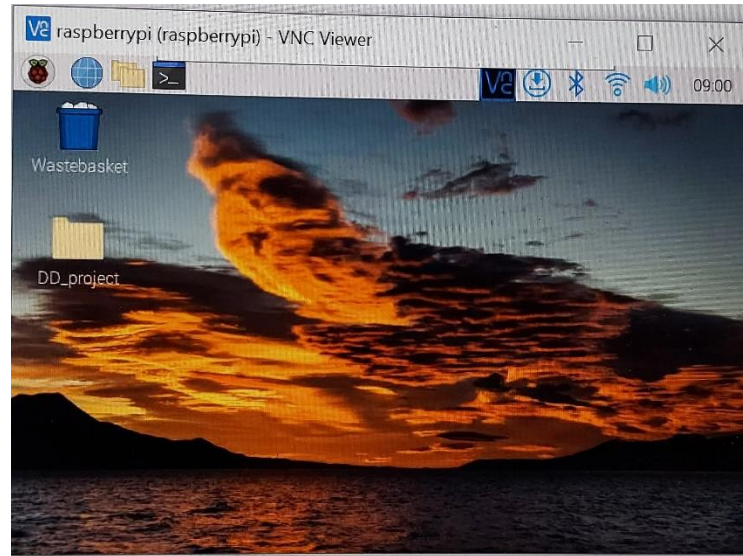


4. Open and give default username: pi and default password: raspberry



5. Now, write `sudo raspi-config` and goto interfacing options -> Enable SSH and VNC viewer
6. Run VNC viewer -> paste IP address of raspberry pi -> click enter to get raspberry pi desktop -> reboot

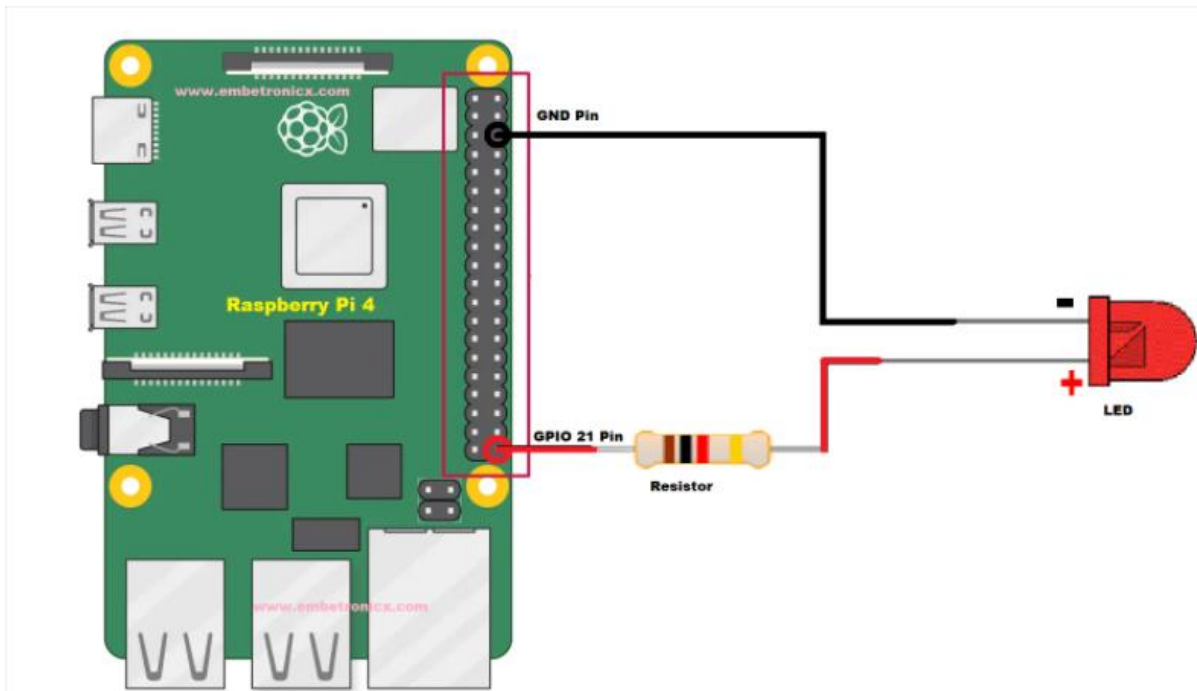




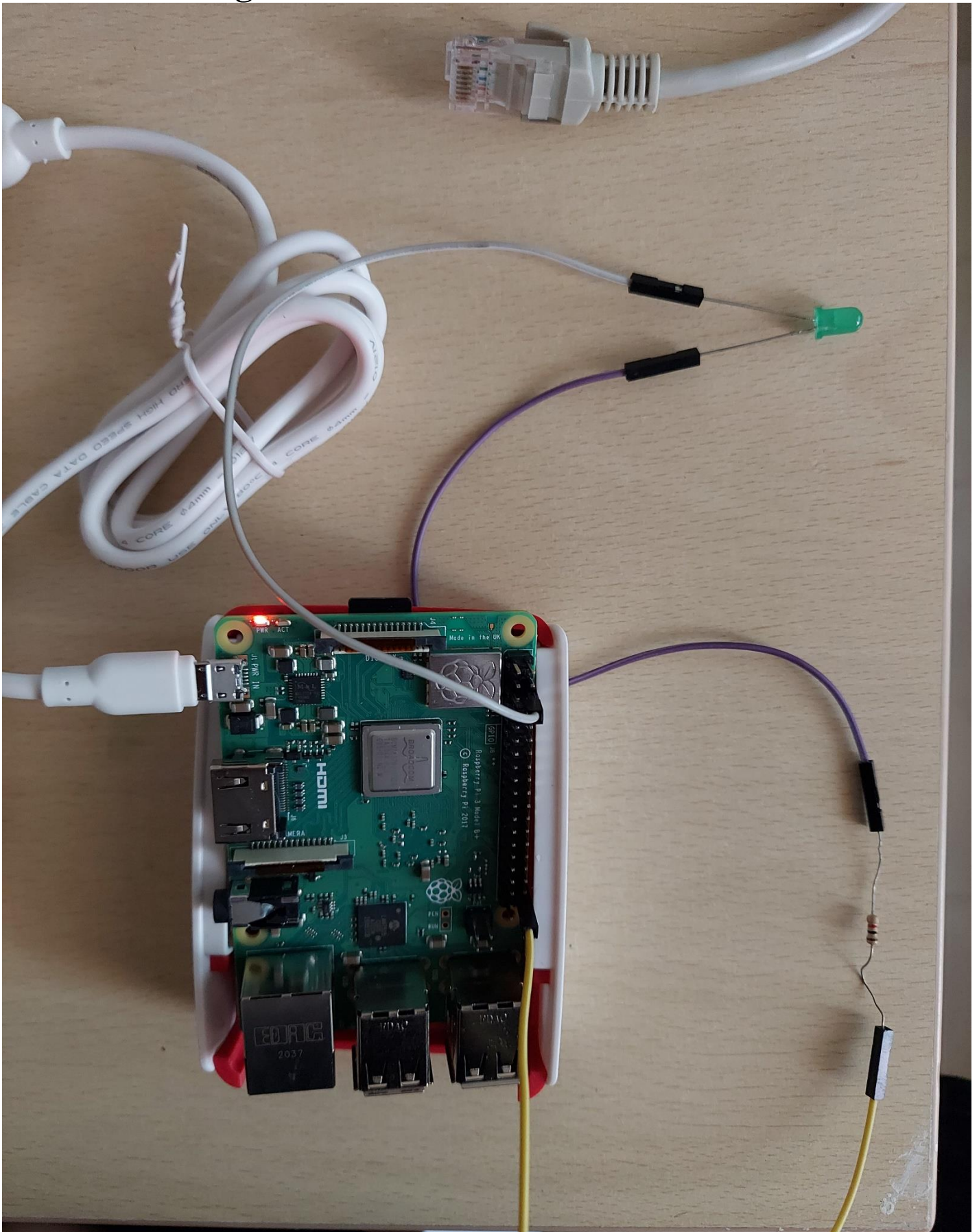
Hardware Required:

- Raspberry pi
- LED
- Resistor (1k ohm)
- Jumper wires

Circuit Connection:



Hardware Image:



Setup Raspberry pi:

- Using “sudo apt install raspberrypi-kernel-headers” install the kernel header.

Accessing the GPIO in Linux Kernel:

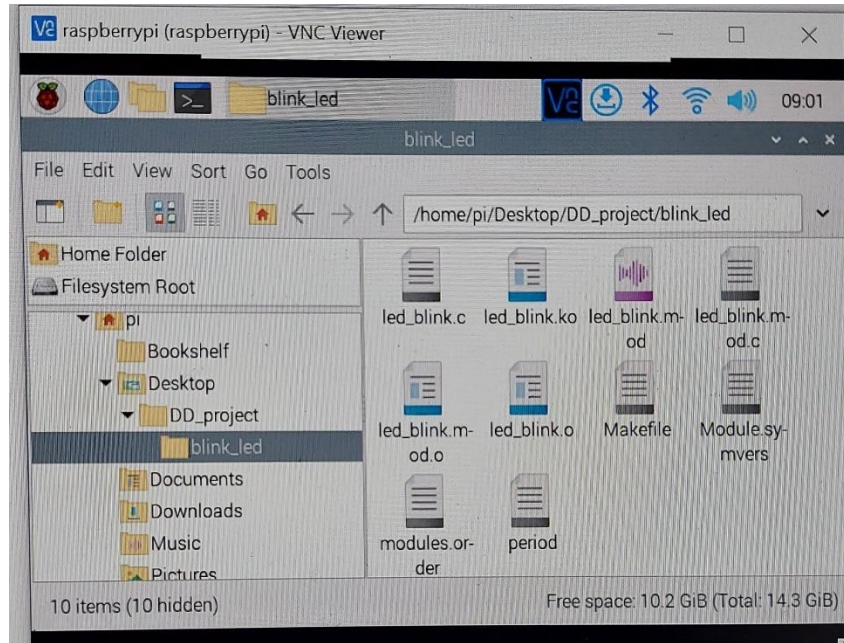
Following steps are used to access the GPIO.

1. Use the GPFSELx register to configure pin number ‘x’ as input or output. In our case, pin number 21 is configured as output.
2. Use the GPFSETx/GPFCLR registers to set or clear the chosen pin.
3. Set the GPIO's direction and initialize the timer in the init() function.
4. A sysfs user mode application is initialized by registering a device class, registering a device object within the class and creating a device file called “period” within the device object.
5. The file will receive a string value from the terminal and convert it into integer and use it to toggle the LED.
6. Finally, all the devices, objects and files are destroyed in the exit() function.

Testing the Device Driver:

1. Build the driver using Makefile using the command “make”.
2. Load the driver using “sudo insmod led_blink.ko”
3. Use sudo su and enter the password if required to get the root permission.
4. Use echo period_value > /sys/class/LedBlink/LedBlink/period to start toggling the led at mentioned period.

VNC viewer:



Terminal:

```

pi@raspberrypi: ~/Desktop/DD_project/blink_led
File Edit Tabs Help

pi@raspberrypi:~/Desktop/DD_project/blink_led $ make
make -C /lib/modules/5.10.63-v7+/build M=/home/pi/Desktop/DD_project/blink_led
modules
make[1]: Entering directory '/usr/src/linux-headers-5.10.63-v7+'
make[1]: Leaving directory '/usr/src/linux-headers-5.10.63-v7+'
pi@raspberrypi:~/Desktop/DD_project/blink_led $ sudo insmod led_blink.ko
insmod: ERROR: could not insert module led_blink.ko: File exists
pi@raspberrypi:~/Desktop/DD_project/blink_led $ sudo rmmod led_blink.ko
pi@raspberrypi:~/Desktop/DD_project/blink_led $ sudo insmod led_blink.ko
pi@raspberrypi:~/Desktop/DD_project/blink_led $ cat period
pi@raspberrypi:~/Desktop/DD_project/blink_led $ echo 2000 > period
bash: period: Permission denied
pi@raspberrypi:~/Desktop/DD_project/blink_led $ sudo su
root@raspberrypi:/home/pi/Desktop/DD_project/blink_led# echo 2000 > period
root@raspberrypi:/home/pi/Desktop/DD_project/blink_led# echo 2000 > /sys/class/L
edBlink/LedBlink/period
root@raspberrypi:/home/pi/Desktop/DD_project/blink_led# echo 20 > /sys/class/L
edBlink/LedBlink/period
root@raspberrypi:/home/pi/Desktop/DD_project/blink_led# echo 500 > /sys/class/L
edBlink/LedBlink/period
root@raspberrypi:/home/pi/Desktop/DD_project/blink_led# echo 1000 > /sys/class/L
edBlink/LedBlink/period
root@raspberrypi:/home/pi/Desktop/DD_project/blink_led#

```

Conclusion

We have implemented gpio linux device driver to toggle the LED connected to GPIO pin 21 of Raspberry pi 3b plus.

Github Links:

https://github.com/h20201400220/G547/tree/main/GPIO_device_driver_LED_blink

<https://github.com/h20201400223/G547/tree/main/GPIO%20linux%20device%20driver%20development>

REFERENCES

- [1] <https://embetronicx.com/tutorials/linux/device-drivers/gpio-driver-basic-using-raspberry-pi/>
- [2] <https://sysprogs.com/VisualKernel/tutorials/raspberry/leddriver/>