

PRACTICAL 9

AIM: Raspberry Pi: Remote Access Setup and LED Control via Python Programming.

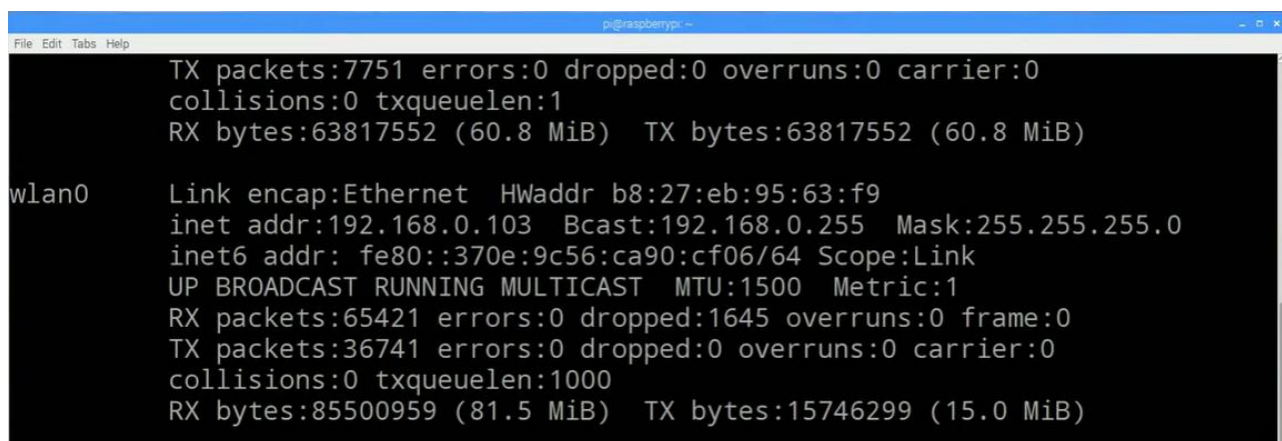
PREREQUISITE: Basics of programming, microcontrollers and basic electronics.

OUTCOME: Access the Raspberry Pi remotely using SSH and VNC, and demonstrate control of its GPIO to blink an LED using Python programming.

PROCEDURE:

A) For Remote Access:

Step 1: Connect the Raspberry Pi to a Wi-Fi network and note its IP address.



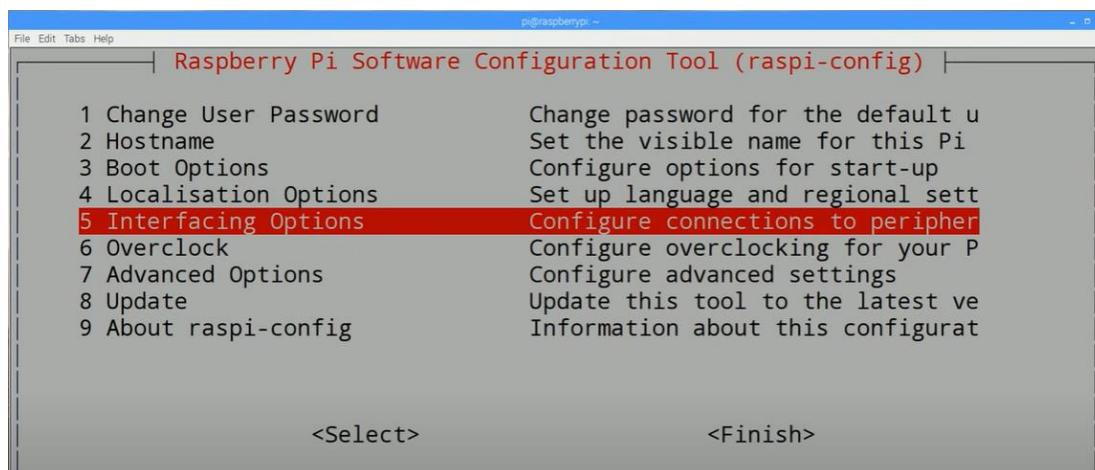
```
pi@raspberrypi ~  
File Edit Tabs Help  
TX packets:7751 errors:0 dropped:0 overruns:0 carrier:0  
collisions:0 txqueuelen:1  
RX bytes:63817552 (60.8 MiB) TX bytes:63817552 (60.8 MiB)  
  
wlan0 Link encap:Ethernet HWaddr b8:27:eb:95:63:f9  
inet addr:192.168.0.103 Bcast:192.168.0.255 Mask:255.255.255.0  
inet6 addr: fe80::370e:9c56:ca90:cf06/64 Scope:Link  
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
RX packets:65421 errors:0 dropped:1645 overruns:0 frame:0  
TX packets:36741 errors:0 dropped:0 overruns:0 carrier:0  
collisions:0 txqueuelen:1000  
RX bytes:85500959 (81.5 MiB) TX bytes:15746299 (15.0 MiB)
```

Step 2: Write command: `sudo raspi-config`

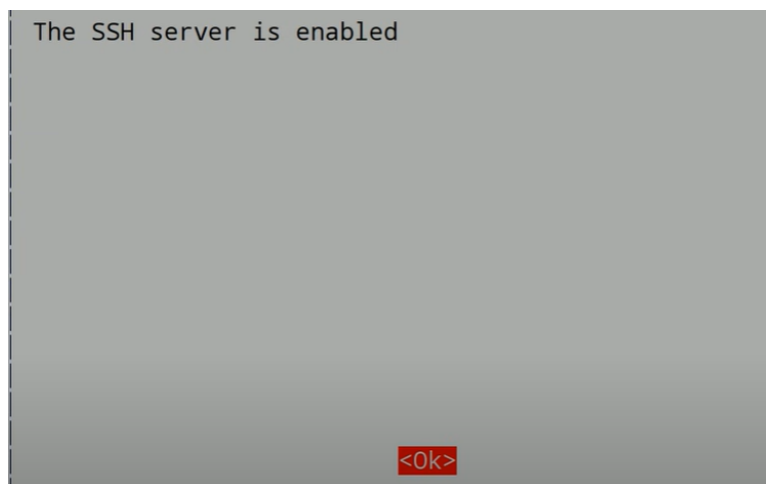
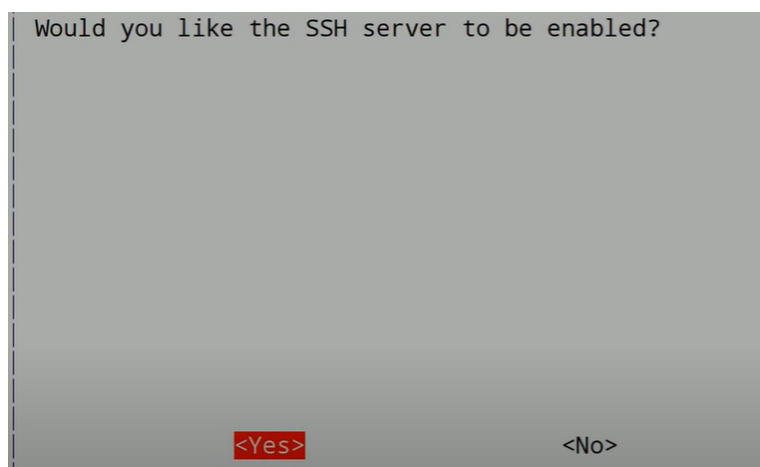
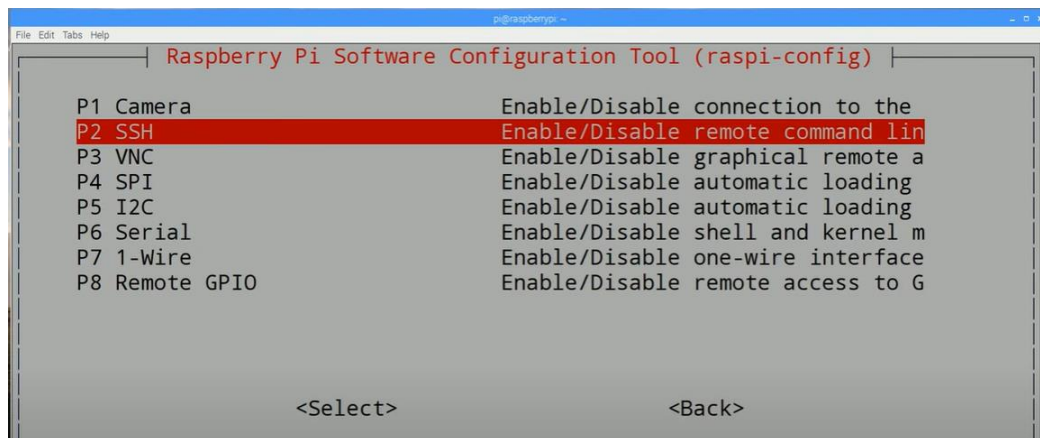


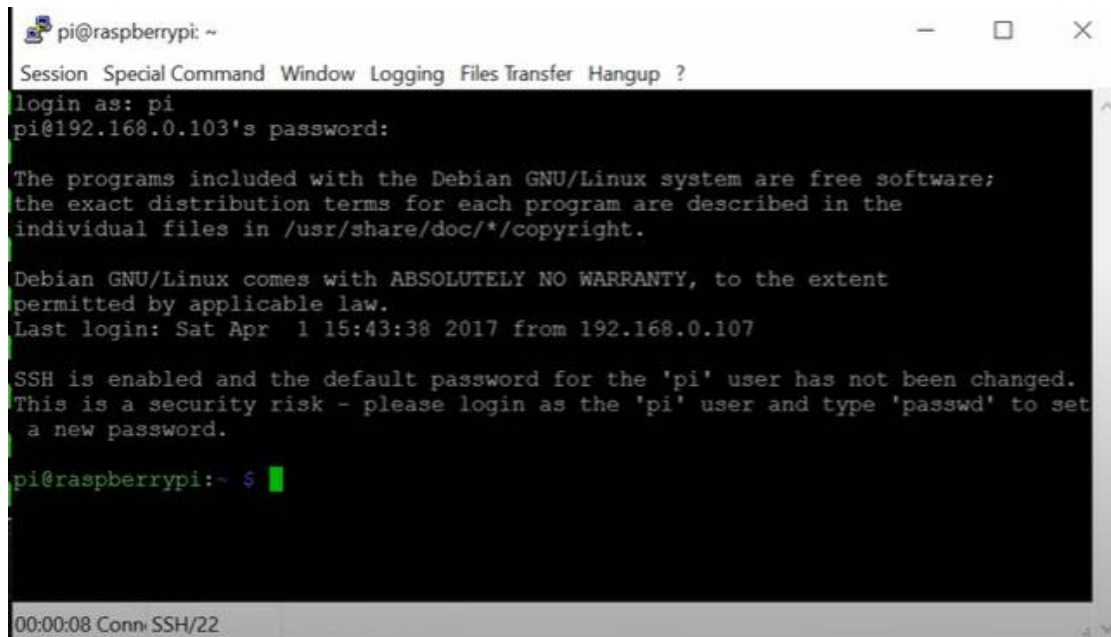
```
pi@raspberrypi:~ $  
pi@raspberrypi:~ $ sudo raspi-config
```

Step 3: Click on Interfacing Options.



```
pi@raspberrypi ~  
File Edit Tabs Help  
Raspberry Pi Software Configuration Tool (raspi-config)  
  
1 Change User Password      Change password for the default u  
2 Hostname                  Set the visible name for this Pi  
3 Boot Options              Configure options for start-up  
4 Localisation Options      Set up language and regional sett  
5 Interfacing Options        Configure connections to peripher  
6 Overclock                 Configure overclocking for your P  
7 Advanced Options          Configure advanced settings  
8 Update                    Update this tool to the latest ve  
9 About raspi-config         Information about this configurat  
  
<Select>                    <Finish>
```

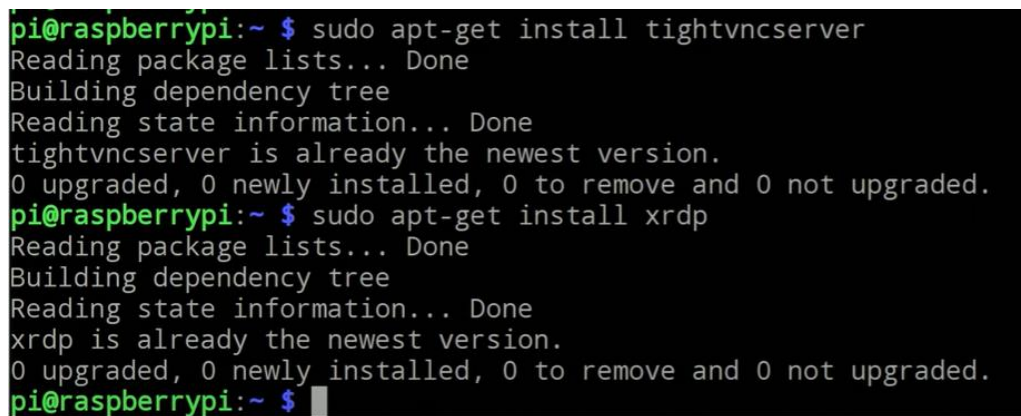
Step 4: Click on P2 SSH**Step 5:** Use the SSH client to remotely access the Raspberry Pi by entering its IP address.



```
pi@raspberrypi: ~  
Session Special Command Window Logging Files Transfer Hangup ?  
login as: pi  
pi@192.168.0.103's password:  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Sat Apr 1 15:43:38 2017 from 192.168.0.107  
  
SSH is enabled and the default password for the 'pi' user has not been changed.  
This is a security risk - please login as the 'pi' user and type 'passwd' to set  
a new password.  
  
pi@raspberrypi:~ $
```

00:00:08 Conn: SSH/22

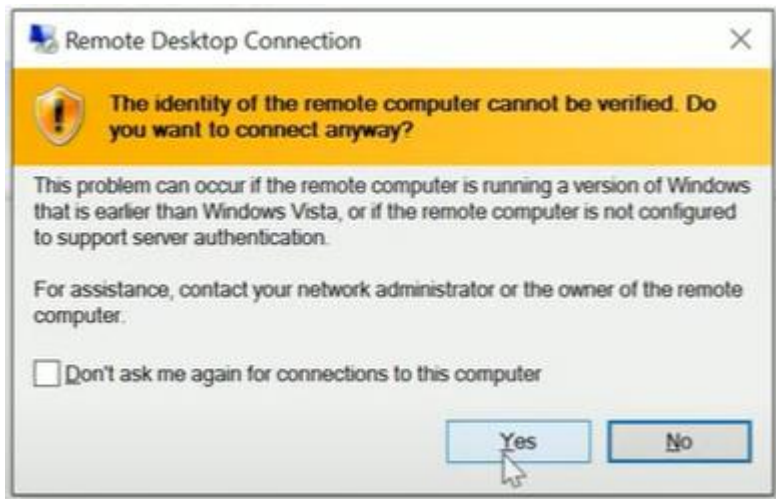
Step 6: Install an SSH client (e.g., PuTTY) and a VNC viewer (e.g., tightvnc Server).



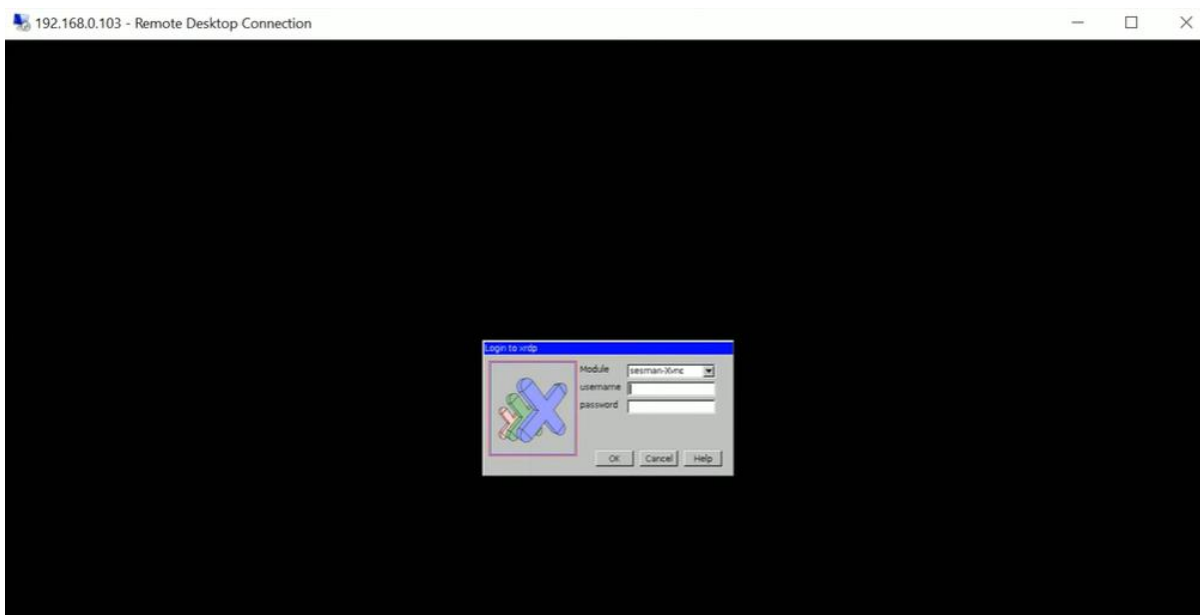
```
pi@raspberrypi:~ $ sudo apt-get install tightvncserver  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
tightvncserver is already the newest version.  
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.  
pi@raspberrypi:~ $ sudo apt-get install xrdp  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
xrdp is already the newest version.  
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.  
pi@raspberrypi:~ $
```

Step 7: Open the VNC viewer, input the IP address of the Raspberry Pi, and log in to access the desktop remotely.





Step 8: Raspberry Pi remote access successful



B) Blink an LED using Python programming.

Code:

```
import RPi.GPIO as GPIO
import time

# Set up GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.OUT)
```

Blink LED

try:

while True:

GPIO.output(17, GPIO.HIGH) # LED on

time.sleep(1)

GPIO.output(17, GPIO.LOW) # LED off

time.sleep(1)

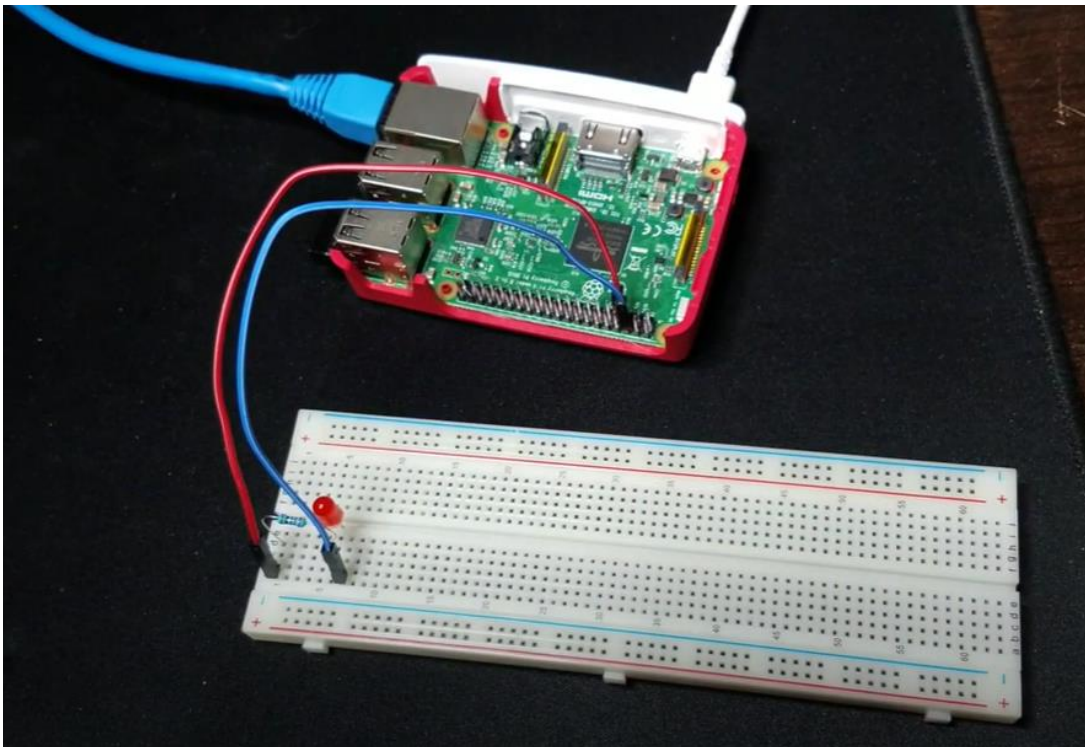
except KeyboardInterrupt:

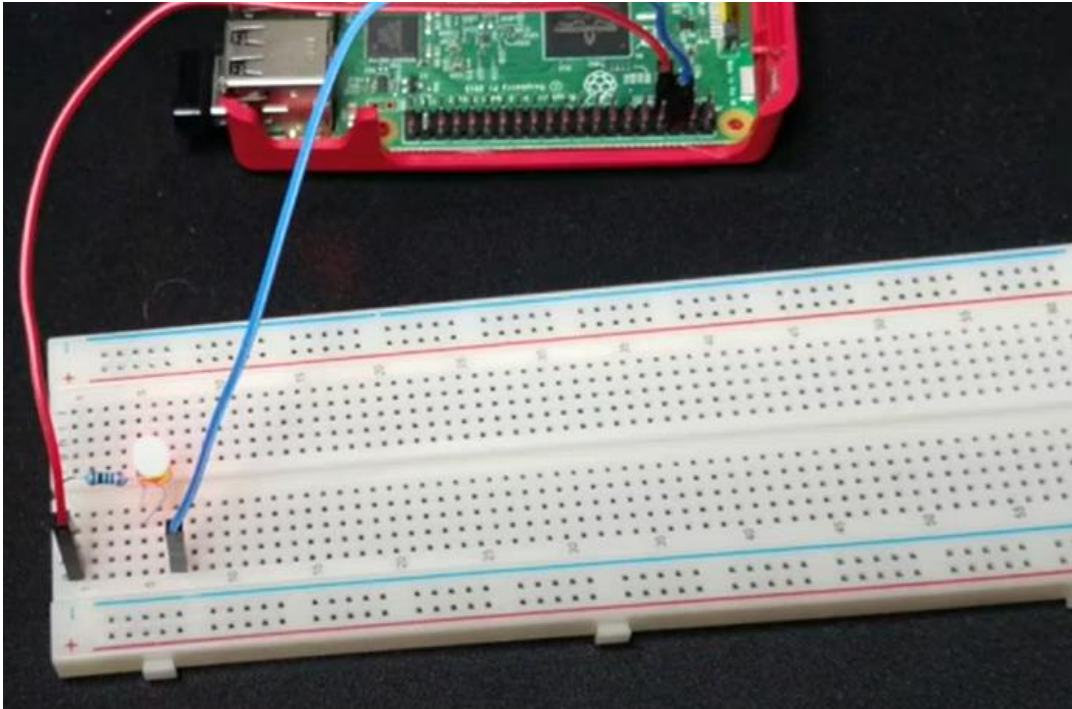
pass

finally:

GPIO.cleanup()

OUTPUT:





Observation

During the experiment, we successfully accessed the Raspberry Pi remotely through SSH and VNC, allowing us to manage it without a physical display. We learned to control GPIO pins using Python programming to blink an LED, gaining hands-on experience with basic circuit connections and Raspberry Pi's GPIO setup. This experiment reinforced our understanding of remote management of IoT devices and the flexibility of Python in hardware control.

Conclusion

In conclusion, remote access to Raspberry Pi through SSH and VNC is highly effective for managing and programming the device without a monitor. This experiment demonstrated how Raspberry Pi GPIO pins could be controlled using Python to interact with hardware components like LEDs. Such setups can be expanded to control more complex hardware, showcasing the Raspberry Pi's suitability for various IoT and automation projects.