Experiment No. : 06

Date : 23/11/2024

EXPERIMENT-06

INTERFACE BLUETOOTH TO SEND AND RECEIVE DATA TO/FROM PHONE

OBJECTIVE - Send and Receive data to/from phone via Bluetooth with Raspberry Pi programming interface.

RESOURCE REQUIRED - Raspberry Pi, Mobile Phone with Bluetooth, Relay module, Jumper wires.

STEPS AND PROCEDURE -

1. Download 'Pi3 Bluetooth Manager' from your phone playstore.
2. Connect Raspberry pi 4 board with display, mouse, keyboard, Adapter only
3. To change the connected Raspberry Pi board Bluetooth name, open terminal window and type bluetoothctl, press Enter, type system-alias 'new name', press Enter.
4. Turn on the Bluetooth on your phone
5. Turn on Raspberry Pi Bluetooth.
6. Click on Bluetooth icon and select make discoverable and Bluetooth icon starts blinking Blue and Green.
7. Click on icon and select 'add device'
8. Open bluetooth settings on phone

Experiment No. 06. Date :

9. Raspberry Pi start searching for your phone device bluetooth and click on your searched phone bluetooth and pair it.

10. Click on pair in your phone.

11. Click OK.

12. Pop up window says pairing successfully and click OK. Now you should be able to connect to Raspberry Pi.

13. In case if previous step dint work, type sudo /etc/init.d/Bluetooth restart in terminal window and press enter.

14. Open Bluetooth settings in your phone and then connect to the Raspberry Pi Bluetooth device.

15. Open Pi3 Bluetooth manager application in your phone.

16. To know the MAC addres of the device, open terminal window and type bluetoothctl, press Enter and type scan on then press Enter. MAC address of the device will be seen as controller and the same will be seen in your phone Pi 3 Bluetooth Manager application too.

17. Press control z on your keyboard to exit out of the bluetooth shell.

18. Now connect ethernet cable to Raspberry Pi board.

19. To install bluetooth package, open terminal window type sudo apt-get install

Experiment No. : 06 Date :

python-bluetooth, press Enter, say Yes and then type sudo apt-get install python-rpi.gpio, press Enter.

20. Open Thonny window and write Code, save it in Home.

21. To run and check the output, open terminal window and type python filename.py, press Enter.

22. Connect to MAC address from phone application.

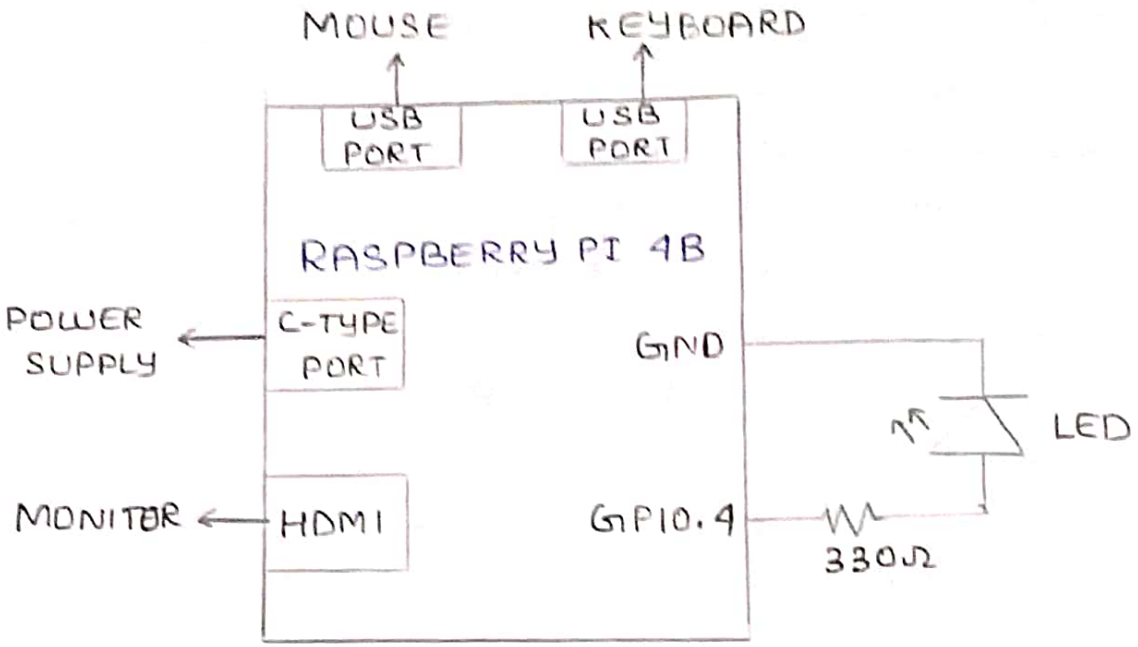
23. Once you connect to MACaddress It will start showing screen in your phone as shown below.

24. Start typing in your phone and click on SEND TO PI, the same will be seen in PI terminal window.

25. You can also send message through Voice by touching MIC icon and speak in your phone App and same thing will be seen in PI terminal window.

AIM : Rig up LED circuit and control LED through Bluetooth with Raspberry Pi programming Interface.
[Send and Receive data to/from phone].

```
import bluetooth
# Importing the GPIO library to use GPIO
pins of Raspberry Pi
```



CIRCUIT DIAGRAM

Experiment No. : 08 Date :

```
import RPi.GPIO as GPIO.
led_pin = 4
GPIO.setmode (GPIO.BCM) # BCM Numbering
GPIO.setup (led_pin, GPIO.out) # Declaring
the pin 40 as output pin
host = " "
port = 1
server = bluetooth.BluetoothSocket(bluetooth.RFCOMM)
print ("Bluetooth socket created")
try:
    server.bind ((host, port))
    print ("Bluetooth binding completed")
except:
    print ("Bluetooth binding failed")
server.listen(1)
client, address = server.accept()
print ("connected to", address)
print ("client :", client)
try:
    while True:
        data = client.recv(1024)
        print (data)
        data = data.decode ("utf-8")
        if data == "1":
            GPIO.output (led_pin, True)
            send_data = "Light_on"
        elif data == "0":
            GPIO.output (led_pin, False)
            send_data = "Light_off"
```

Experiment No. : 06 Date :

```
else:
    send_data = "type 1 or 0"
    client.send(send_data)
except:
    GPIO.cleanup()
    client.close()
    server.close()
```

RESULT Successfully implimented sending & recieving messages to/from phone via bluetooth with Raspberry pi programming Interface.

Experiment No. : 07 Date : 23/11/2024

EXPERIMENT-06

USE UDP FOR CLIENT OPERATIONS.

OBJECTIVE - Send and Recive message to/from phone using UDP for client and server with Raspberry Pi - Programming Interface.

STEPS AND PROCEDURE -

1. Write UDP server and client codes seperate in thonny and save them in Home.
2. Open two new terminals windows for server and client seperately (if you are working in local Host i.e., in same system).
3. Type python serverfilname.py in one terminal and press Enter. Server will start for up and Running.
4. Type python clientfilname.py in another terminal and press Enter. You will start getting the message sent by server in client terminal window.
5. If you want to work in two different systems, one as server and another as client, type server code in server system and client code in client system.
6. Type ifconfig in terminal window to know the IP address to Ethernet (to be connected) from both systems. If you want to send message from server to client, type server IP address

Experiment No. : 07 Date :

In both programs and follow steps 3 in server system and 4 in client system.

7. We can switch the role of both systems by IP address.

UDP SERVER PROGRAM USING PYTHON:

```
import socket
localIP = "127.0.0.1"
localPort = 20001
bufferSize = 1024
msgFromServer = "Hello UDP client"
bytesToSend = str.encode(msgFromServer)
UDPServerSocket = socket.socket(family =
 socket.AF_INET, type = socket.SOCK_DRGM)
UDPServerSocket.bind((localIP, localPort))
print("UDP server up and running")
while (True):
    bytesAddressPair = UDPServerSocket.recvfrom
                                    (bufferSize)

    message = bytesAddressPair(0)
    address = bytesAddressPair[1]
    clientMsg = "Message from client : {}".
                                format(message)
    clientIP = "Client IP Address : {}". format
                                    (address)

    print(clientMsg)
    print(clientIP)
UDPServerSocket.sendto(bytesToSend, address)
```

Experiment No. : 07 Date :

UDP client using Python

```
import socket
msgFromClient = "Hello UDPServer"
bufferSize = 1024
bytesToSend = str.encode(msgFromClient)
serverAddrPort = ("127.0.0.1", 20001)
UDPClientSocket = socket.socket(family = socket.
AF_INET, type = socket.SOCK_DGRAM)
UDPClientSocket.sendto(bytesToSend, serverAddrPort)
msgFromServer = UDPClientSocket.recvfrom
                                (bufferSize)
msg = "Message from sensor {}".format
                        (msgFromServer[0])
print(msg)
```

RESULT -

Implemented a Raspberry Pi program to send and Recieve message to/from using UDP for client and Server.

Experiment No. : 07 Date :

ASSIGNMENT -

Send and Recieve message from two different systems using UDP for client and server with Raspberry pi - Programming Interface .

UDP Server using python .

```
import socket
localIP = "172.1.13.68"
localPort = 20001
bufferSize = 1024
msgFromServer = "Hello UDP Client"
bytesToSend = str.encode(msgFromServer)
UDPServerSocket = socket.socket(family = socket.AF
INET, type = Socket.SOCK DGRAM)
UDPServerSocket.bind((localIP, localPort))
print("UDP server up and Running")
while (True):
    bytesAddressPair = UDPServerSocket.recvfrom
                                    (bufferSize)

    message = bytesAddressPair[0]
    address = bytesAddressPair[1]
    clientMsg = "Message from Client : {}".
                                    format(message)

    clientIP = "Client IP Address : {}". format
                                    (address)
    print(clientMsg)
    print(clientIP)
    UDPServer.Socket.sendto(bytesTosend, address)
```

Experiment No. : 07 Date :

UDP client using Python.

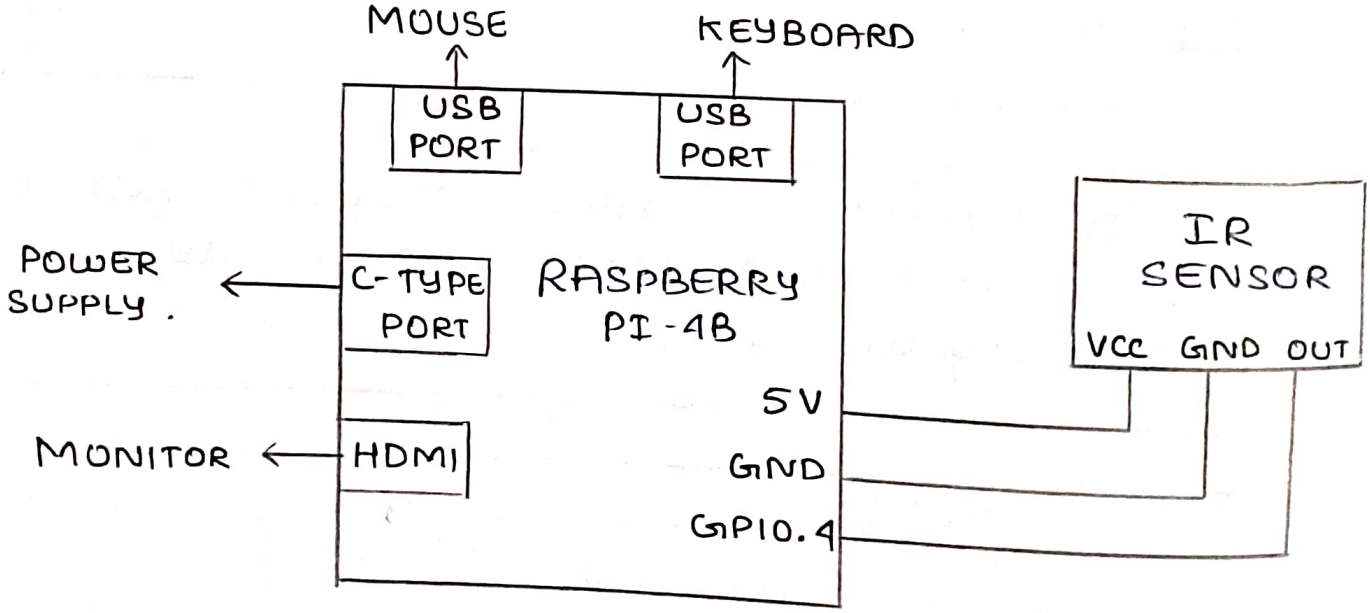
```
import socket
msgFromClient = "Hello UDP Server"
bufferSize = 1024
bytesToSend = str.encode(msgFromClient)
serverAddrsPort = ("172.1.13.68", 20001)
UDPClientSocket = socket.socket(family=socket.
        AF_INET, type = socket.SOCK_DGRAM)
UDPClientSocket.sendto(bytesToSend, server
                                AddrsPort)
msgFromServer = UDPClientSocket.recv from
                                (bufferSize)
msg = "Message from Server {}".format(msg
                                FromServer[0])
print(msg)
```

RESULT

Implimented a python program using UDP for client and server between two different systems.

LDR SENSOR PINOUTS -

CIRCUIT DIAGRAM



Experiment No. : 08

Date : 07/12/2024.

EXPERIMENT- 08

OBJECTIVE - Performing basic SQL operation with Raspberry Pi - Programming Interface.

AIM -

1. Learn SQL Operations
2. Learn about Databases.
3. Database Access tranfer to multiple users
4. Rimote Sensor data transfer to a database from raspberry Pi.

STEPS TO BE FOLLOWED -

(1) Turn ON Raspberry-pi with Ethernet connection.

(2) Type the following code and save it as a .py file on Home of Raspberry-pi.

EXAMPLE CODE FOR IR SENSOR

< Connect output of IR sensor to GPIO4 >

```
import time
import datetime
import csv
import MySQLdb
import RPi.GPIO as GPIO

GPIO.setwarnings(False)
```

Experiment No. : 08 Date :

```
GPIO.cleanup()
pin = 4

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(pin, GPIO.IN)
dB = MySQLdB.connect(host = "localhost", user =
     "lab", passwd = "pimylifeup", db = "er")
cur = db.cursor() # Creates cursor to pass on
                    demands to MySQL/MariaDB
while True: # collects data indefinitely.
    degres = GPIO.input(pin) # Read from sensor.
    timnow = datetime.datetime.utcnow()
    print (degres)
    #excecutes SQL command in MySQL/MariaDB
    to insert data.
    cur.execute("INSERT INTO Sensorstats(data-time,
    temperature) VALUES('%s','%s');",(timnow,degres));
    db.commit() # commits data to table
    time.sleep(1) # collects data after 1second again
```

(3) Now open a new terminal and run the following commands

```
>> sudo apt update.
>> sudo apt upgrade.
>> sudo apt install mariadb-server mariadb-
                                    client

>> sudo apt-get install python3-mysqldb.
```

Experiment No. : 08

Date :

⇒ For securing database operations with password.

>> sudo mysql_secure_installation

Set password accordingly by saying Yes [y] to all the prompts as shown in the fig.

Now login to SQL and go on to create your database and tables as follows.

```
>> sudo mysql -u root -p
>> CREATE DATABASE lr;
>> CREATE USER 'examplesuser'@'localhost'
   IDENTIFIED BY 'pimylifeup';
>> USE lr;
>> CREATE TABLE SensorStats (date_time VARCHAR
   (50), temperature (FLOAT);
>> GRANT ALL PRIVILEGES ON lr.* TO 'exampl
   user'@'localhost';
>> FLUSH PRIVILEGES;
>> quit
```

⇒ Now open a new terminal and run the python code that you had saved in the beginning.

You should be able to log the sensor data now. To view what data has been logged, stop the python program and do the following:

```
>> sudo mysql -u root -p      (OR)
>> sudo mysql -u examplesuser -p
```

Experiment No. 08.

Date :

⇒ Enter the password accordingly as you just set in the previous step and login.
You should be able to enter Maria DB shell.
Inside the shell type the following -

```
>> USE ir; # Database name.
>> SELECT * FROM Sensorstats; # Table name.
```

⇒ You should be able to see previously logged sensor data along with timestamp as shown in the image.

RESULTS :

Basic SQL operations (to store IR sensor data) were performed with Raspberry Pi-Programming Interface.