Experiment 6

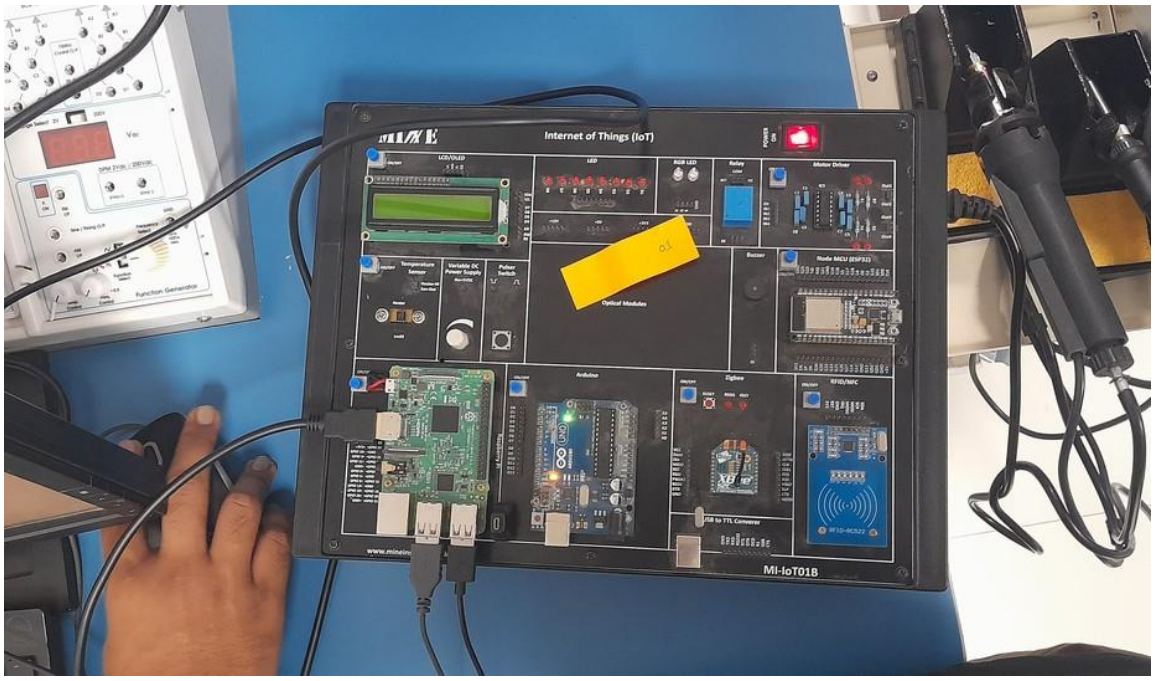
(By Kumar Anmol)

Aim:- To create client server communication with raspberry pi over Wi-Fi using UDP protocol.

Equipment Used:-

1. HDMI cables.
2. Two Raspberry Pie 4 Board
3. Two Monitor
4. Key board and mouse

Steps:- An HDMI cable is used to connect the monitor to the Raspberry Pi 4 board, while USB cables are used to connect the mouse and keyboard to the Raspberry Pi 4 board. The same setup is required for the other monitor and Raspberry Pi 4 board. One of these will function as the server, and the other will function as the client. The connections are illustrated below:-



Results:- On connecting the server and client systems using comman Wifi of "IITI" and then when we wrote "server client comm group 1" on server system then we got same output in client system as shown in below drive links:-

Drive link (Client) Drive link

(Server)

Conclusion:- We learnt about how to develop client server communication with raspberry pi over Wi-Fi using UDP protocol, how connections are made and also the key thing about putting correct host and port addresses to develop connection between systems.

Code:- Code was written in “Thonny” software, code is given below for server and client:-

Server code:-

```
import socket
 host = '10.205.2.238'
 port = 8000
 storedValue = "server client comm group 1" def
 setupServer():
   s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
   print("Socket created.")
   try:
      s.bind((host, port))
   except socket.error as msg:
      print(msg)
   print("Socket bind comlete.")
   return s
 def setupConnection():
   s.listen(1) # Allows one connection at a time. conn,
   address = s.accept()
   print("Connected to: " + address[0] + ":" + str(address[1]))
   return conn
 def GET():
   reply = storedValue
   return reply
 def REPEAT(dataMessage):
   reply = dataMessage[1]
   return reply
 def dataTransfer(conn):
   # A big loop that sends/receives data until told not to. while
   True:
      # Receive the data
```

```
        data = conn.recv(1024) # receive the data
        data = data.decode('utf-8')
        # Split the data such that you separate the command #
        from the rest of the data.
        dataMessage = data.split(' ', 1)
        command = dataMessage[0] if
        command == 'GET':
            reply = GET()
        elif command == 'REPEAT': reply
            = REPEAT(dataMessage)
        elif command == 'EXIT':
            print("Our client has left us :(")
            break
        elif command == 'KILL':
            print("Our server is shutting down.")
            s.close()
            break
        else:
            reply = 'Unknown Command'
        # Send the reply back to the client
        conn.sendall(str.encode(reply))
        print("Data has been sent!")
    conn.close()

s = setupServer()
 while True:
    try:
        conn = setupConnection()
        dataTransfer(conn)
    except:
        break
```

Client code:-

```
import socket
 from time import sleep
 host = '10.205.2.238'
 port = 8000
```

```
def setupSocket():
  s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) s.connect((host,
  port))
  return s
def sendReceive(s, message):
  s.send(str.encode(message)) reply
  = s.recv(1024)
  print("We have received a reply")
  print("Send closing message.")
  s.send(str.encode("EXIT"))
  s.close()
  reply = reply.decode('utf-8')
  return reply
def transmit(message): s
  = setupSocket()
  response = sendReceive(s, "GET")
  return response
response = transmit("GET")
print(response)
```