

How to use UDP/IP with Arduino ESP32

1. Introduction

- Create a UDP server using Python and Arduino ESP32 UDP client. Client will send the data to the server; the server will convert it to upper case and respond it to the client.

2.1 Python server

```
import socket
print("Listening...")
# bind all IP
HOST = '0.0.0.0'
# Listen on Port
PORT = 44444
#Size of receive buffer
BUFFER_SIZE = 1024
# Create a TCP/IP socket
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
# Bind the socket to the host and port
s.bind((HOST, PORT))
while True:
    # Receive BUFFER_SIZE bytes data
    # data is a list with 2 elements
    # first is data
    #second is client address
    data = s.recvfrom(BUFFER_SIZE)
    if data:
        #print received data
        print('Client to Server: ' , data)
        # Convert to upper case and send back to Client
        s.sendto(data[0].upper(), data[1])
# Close connection
s.close()
```

2.2 Arduino ESP32 UDP client

```
#include <WiFi.h>
#include <WiFiUdp.h>

/* WiFi network name and password */
const char * ssid = "ola12345";
const char * pwd = "olaola123";

// IP address to send UDP data to.
// it can be ip address of the server or
// a network broadcast address
// here is broadcast address
const char * udpAddress = "192.168.137.1"; //TIVE DE PÔR O Wireless LAN
adapter Local Area Connection* 10
const int udpPort = 44444;

//create UDP instance
WiFiUDP udp;

void setup(){
  Serial.begin(115200);

  //Connect to the WiFi network
  WiFi.begin(ssid, pwd);
  Serial.println("");

  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  //This initializes udp and transfer buffer
  udp.begin(udpPort);
}

void loop(){
  //data will be sent to server
  uint8_t buffer[50] = "hello world";
  //send hello world to server
```

```

udp.beginPacket(udpAddress, udpPort);
udp.write(buffer, 11);
udp.endPacket();
memset(buffer, 0, 50);
//processing incoming packet, must be called before reading the buffer
udp.parsePacket();
//receive response from server, it will be HELLO WORLD
if(udp.read(buffer, 50) > 0){
    Serial.print("Server to client: ");
    Serial.println((char *)buffer);
}
//Wait for 1 second
delay(1000);
}

```

3. Result

The screenshot displays two windows from an IDE. The top window is a terminal titled 'powerShell' showing the output of a Python script 'udpsrvr.py'. The script is running on a Windows machine (PS C:\Users\35193\desktop> py udpsrvr.py). The output shows the server listening on port 4444 and receiving five 'hello world' messages from a client at IP 192.168.137.68. The bottom window is a serial monitor titled 'COM6' showing the same messages being received by the Arduino. The messages are: '15:43:14.693 ->', '15:43:15.222 ->', '15:43:17.203 -> Connected to olal2345', '15:43:17.203 -> IP address: 192.168.137.68', and five subsequent '15:43:18.222 -> Server to client: HELLO WORLD' messages.

```

PS C:\Users\35193\desktop> py udpsrvr.py
listening...
Client to Server: (b'hello world', ('192.168.137.68', 4444))
Client to Server: (b'hello world', ('192.168.137.68', 4444))
Client to Server: (b'hello world', ('192.168.137.68', 4444))
Client to Server: (b'hello world', ('192.168.137.68', 4444))
Client to Server: (b'hello world', ('192.168.137.68', 4444))
Client to Server: (b'hello world', ('192.168.137.68', 4444))

COM6
15:43:14.693 ->
15:43:15.222 -> .....
15:43:17.203 -> Connected to olal2345
15:43:17.203 -> IP address: 192.168.137.68
15:43:18.222 -> Server to client: HELLO WORLD
15:43:19.200 -> Server to client: HELLO WORLD
15:43:20.242 -> Server to client: HELLO WORLD
15:43:21.236 -> Server to client: HELLO WORLD
15:43:22.216 -> Server to client: HELLO WORLD

```

Figure 1 - UDP/IP with Arduino ESP32