

# ESP32 Communication Protocols

## 1. ESP32 – 1 as Server (Access-Point) and ESP32 – 2 as Client

- The ESP32 Server creates its own wireless network (ESP32 Soft-Access-Point). So, other WiFi devices can connect to that network.
- What you need:
  - 2 ESP32 with micro USB cable
- These source code were tested using PlatformIO in Visual Studio Code with .ini configuration as below :

```
platform = espressif32
board = esp32doit-devkit-v1
framework = arduino
monitor_speed = 115200
monitor_rts = 0
monitor_dtr = 0
```

- This examples use “ESPAsyncWebServer” library which you can download instantly from PlatformIO Library menu in VSCode
- This code can be accessed at :

<https://github.com/bimanjayaaaji/raspberry-pi--esp32-comm-TCP.git>

### a) ESP32 - Server Source Code

```
// Import required libraries
#include <Arduino.h>
#include "WiFi.h"
#include "ESPAsyncWebServer.h"

// Set your access point network credentials
const char* ssid = "ESP32-Access-Point";
const char* password = "12345678";

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);

void setup(){
    // Serial port for debugging purposes
    Serial.begin(115200);
    Serial.println();
    // Setting the ESP as an access point
    Serial.print("Setting AP (Access Point)...\n");
    // Remove the password parameter, if you want the AP (Access Point) to be open
    WiFi.softAP(ssid, password);

    IPAddress IP = WiFi.softAPIP();
    Serial.print("AP IP address: ");
    Serial.println(IP);
```

```

server.on("/var1", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", "1");
});
server.on("/var2", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", "2");
});
server.on("/var3", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", "3");
});
// Start server
server.begin();
}

void loop(){
}

```

## b) ESP32 – Client Source Code

```

#include <Arduino.h>
#include <WiFi.h>
#include <HTTPClient.h>

const char* ssid = "ESP32-Access-Point";
const char* password = "12345678";

//Your IP address or domain name with URL path
const char* serverNameVar1 = "http://192.168.4.1/var1";
const char* serverNameVar2 = "http://192.168.4.1/var2";
const char* serverNameVar3 = "http://192.168.4.1/var3";

String var1;
String var2;
String var3;

unsigned long previousMillis = 0;
const long interval = 500;
unsigned long start;
unsigned long end;

String httpGETRequest(const char* serverName) {
    WiFiClient client;
    HTTPClient http;
    // Your Domain name with URL path or IP address with path
    http.begin(client, serverName);
    // Send HTTP POST request
    int httpResponseCode = http.GET();
    String payload = "--";
    if (httpResponseCode>0) {
        Serial.print("HTTP Response code: ");
    }
}

```

```

        Serial.println(httpResponseCode);
        payload = http.getString();
    }
    else {
        Serial.print("Error code: ");
        Serial.println(httpResponseCode);
    }
    // Free resources
    http.end();

    return payload;
}

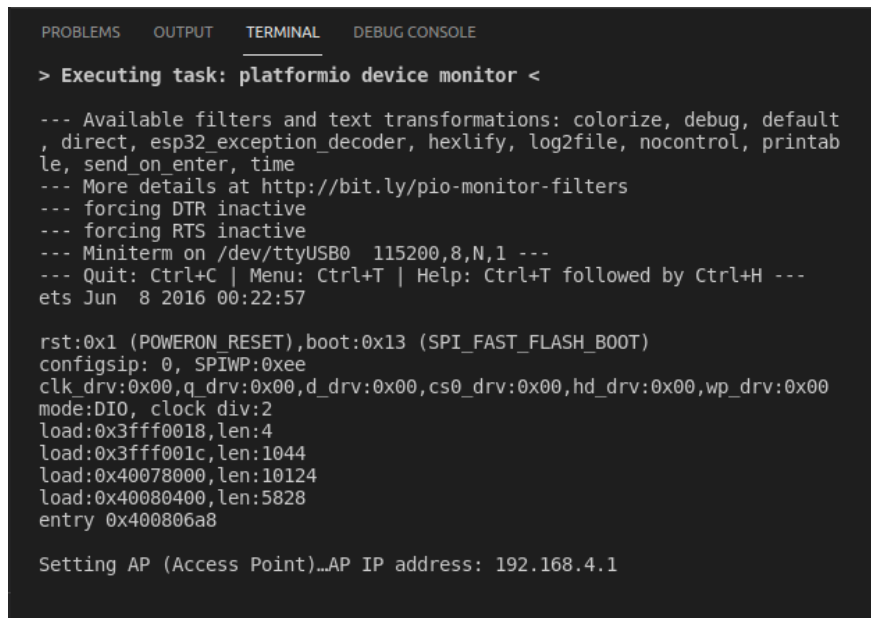
void setup() {
    Serial.begin(115200);
    WiFi.begin(ssid, password);
    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.print("Connected to WiFi network with IP Address: ");
    Serial.println(WiFi.localIP());
}

void loop() {
    unsigned long currentMillis = millis();
    if(currentMillis - previousMillis >= interval) {
        // Check WiFi connection status
        if(WiFi.status()== WL_CONNECTED ){
            start = millis();
            var1 = httpGETRequest(serverNameVar1);
            var2 = httpGETRequest(serverNameVar2);
            var3 = httpGETRequest(serverNameVar3);
            Serial.println(var1);
            Serial.println(var2);
            Serial.println(var3);
            Serial.println(millis()-start);

            // save the last HTTP GET Request
            previousMillis = currentMillis;
        }
        else {
            Serial.println("WiFi Disconnected");
        }
    }
}

```

### c) Screenshots and Descriptions



```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

> Executing task: platformio device monitor <

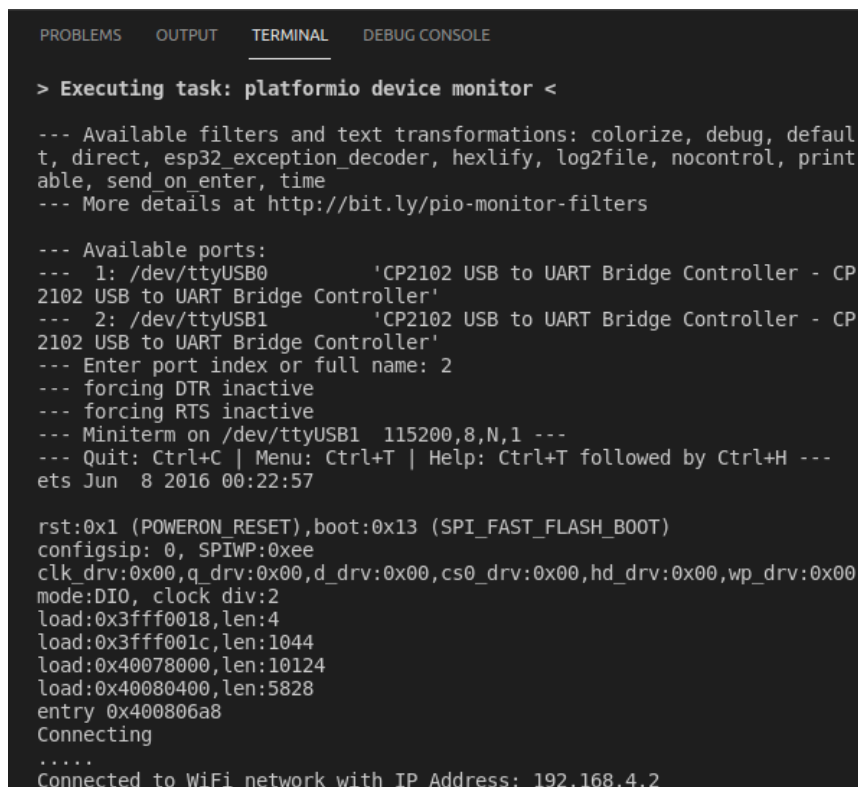
--- Available filters and text transformations: colorize, debug, default,
direct, esp32_exception_decoder, hexlify, log2file, nocontrol, printable,
send_on_enter, time
--- More details at http://bit.ly/pio-monitor-filters
--- forcing DTR inactive
--- forcing RTS inactive
--- Miniterm on /dev/ttyUSB0 115200,8,N,1 ---
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0018,len:4
load:0x3fff001c,len:1044
load:0x40078000,len:10124
load:0x40080400,len:5828
entry 0x400806a8

Setting AP (Access Point)...AP IP address: 192.168.4.1
```

Picture 1. Serial Monitor from ESP32 – Server

Picture 1 shows the serial monitor from ESP32 – Server. As you can see, it prints the IP-Address of the ESP32 as an Access Point.



```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

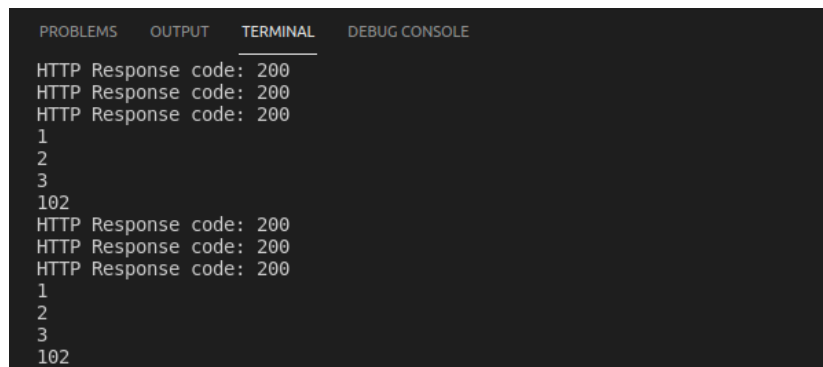
> Executing task: platformio device monitor <

--- Available filters and text transformations: colorize, debug, default,
direct, esp32_exception_decoder, hexlify, log2file, nocontrol, printable,
send_on_enter, time
--- More details at http://bit.ly/pio-monitor-filters

--- Available ports:
--- 1: /dev/ttyUSB0 'CP2102 USB to UART Bridge Controller - CP
2102 USB to UART Bridge Controller'
--- 2: /dev/ttyUSB1 'CP2102 USB to UART Bridge Controller - CP
2102 USB to UART Bridge Controller'
--- Enter port index or full name: 2
--- forcing DTR inactive
--- forcing RTS inactive
--- Miniterm on /dev/ttyUSB1 115200,8,N,1 ---
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0018,len:4
load:0x3fff001c,len:1044
load:0x40078000,len:10124
load:0x40080400,len:5828
entry 0x400806a8
Connecting
.....
Connected to WiFi network with IP Address: 192.168.4.2
```

Picture 2. Serial Monitor from ESP32 – Client



```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
HTTP Response code: 200
HTTP Response code: 200
HTTP Response code: 200
1
2
3
102
HTTP Response code: 200
HTTP Response code: 200
HTTP Response code: 200
1
2
3
102
```

Picture 3. Serial Monitor from ESP32 – Client – 2

Picture 2 and 3 show the serial monitor from ESP32 – Client. Picture 2 shows an attempt to connect to the server. “Available ports:” option appears because we connect 2 ESP32 simultaneously. As we can see, the client is connected to the server.

Picture 3 shows the output of our `httpGETRequest()` function. The ‘Response code’ shows the value we set in the server code, then we print the variable value we get for each `var1`, `var2`, and `var3`. And the last line is the calculated elapsed time from each loop, which is around 102ms.

Reference :

<https://randomnerdtutorials.com/esp32-client-server-wi-fi/>

## 2. TCP Communication Using Raspberry Pi as Server and ESP32 as Client

- ESP32 source code were tested using Arduino IDE with configuration as below :

```
Board = ESP32 Dev Module
Upload Speed = 921600
CPU Frequency = 240MHz (WiFi/BT)
Flash Frequency = 80MHz
Flash Mode = QIO
Flash Size = 4MB (32Mb)
Partition Scheme = Default 4MB with spiffs (1.2MB APP/1.5MB SPIFSS)
Core Debug Level = None
PSRAM = Disabled
```

- This code can be accessed at :

<https://github.com/bimanjayaaaji/raspberry-pi--esp32-comm-TCP.git>

### a) Raspberry Pi **Server** Source Code

```
import socket
from time import time,time_ns

s = socket.socket()

s.bind(('0.0.0.0', 8090 ))

s.listen(0)
```

```

def main():
    while True:
        client, addr = s.accept()

        while True:
            start = time_ns()
            content = client.recv(32)

            if len(content) == 0:
                break
            else:
                print(content.decode(), type(content))
                client.send("Hello from PC".encode())
                #print(time_ns()-start)

        print("Closing connection")
        client.close()

if __name__ == '__main__':
    main()

```

## b) ESP32 Client Source Code

```

#include <WiFi.h>

const char* ssid = "ABA_ROBOTICS2";
const char* password = "aba1234567";
const uint16_t port = 8090;
const char* host = "192.168.0.138";
unsigned long start;
WiFiClient client;

void setup() {
    Serial.begin(115200);
    Serial.println("starting");
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500); Serial.println("...");
    }

    Serial.print("WiFi connected with IP: ");
    Serial.println(WiFi.localIP());
    client.connect(host, port);

    if (!client.connect(host, port)) {
        Serial.println("Connection to host failed");
        delay(1000);
        return;
    }
}

void loop() {
    start = millis();

    client.print("Hello from ESP32!");

    //while (!client.available()){
    //Serial.println("not receiving data");
    //}

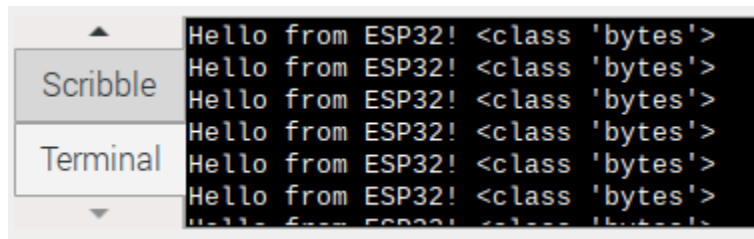
    Serial.print("Receiving : ");
    Serial.println(String(client.read()));
    Serial.print("elapsed :");
    Serial.println(millis()-start);

    //Serial.println("Disconnecting...");
    //client.stop();
}

```

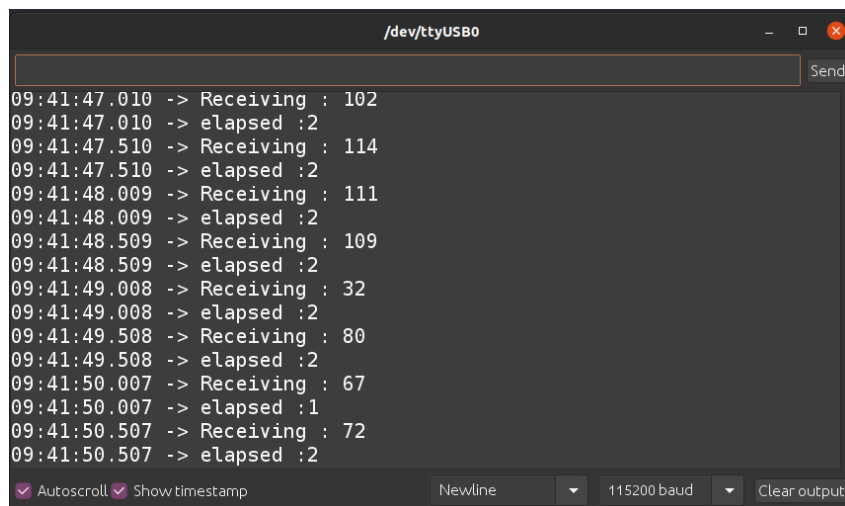
```
    delay(500);
}
```

### c) Screenshots and Description



Picture 4. Terminal Printing from Raspberry-Pi

Picture 4 shows the terminal printing from Raspberry-Pi script. First, it prints string value “Hello from ESP32!” sent by ESP32 itself, then the type of the message received by Raspi before its decoded for debugging purpose.



Picture 5. Serial Monitor from ESP32

Picture 5 shows the serial monitor printed from ESP32. In each loop, first it prints the value received from Raspberry-Pi, then it prints the delta time to send message to Raspi and receive a message from Raspi in millisecond, which is around 2ms.

Reference :

<https://www.dfrobot.com/blog-1003.html>

<http://www.iotsharing.com/2017/05/tcp-udp-ip-with-esp32.html>

## 3. MQTT Communication Using Raspberry Pi and 2 ESP32 Devices

- These ESP32 source code were tested using PlatformIO in Visual Studio Code with .ini configuration as below :

```
platform = espressif32
board = esp32doit-devkit-v1
framework = arduino
monitor_speed = 115200
```

```
monitor_rts = 0
monitor_dtr = 0
lib_deps = knolleary/PubSubClient@2.8
```

- This code can be accessed at :

<https://github.com/bimanjayaaji/raspberry-pi--esp32-comm-TCP.git>

#### a) ESP32 – 1 Source Code

```
#include <Arduino.h>
#include <WiFi.h>
#include <PubSubClient.h>

const char* ssid = "ABA_ROBOTICS2";
const char* password = "aba1234567";
const char* mqtt_server = "192.168.0.138";

WiFiClient espClient;
PubSubClient client(espClient);
char msg[50];
int value = 0;
const char* device_name = "ESP32";

void setup_wifi() {
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);

    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println("");
    Serial.println("WiFi connected");
    Serial.print("IP address : ");
    Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* message, unsigned int length) {
    Serial.print("Message: ");
    String messageTemp;

    for (int i = 0; i < length; i++) {
        Serial.print((char)message[i]);
        messageTemp += (char)message[i];
    }
}
```



```

        Serial.println();

        if (String(topic) == "my_topic") {
            client.publish("my_topic2", "esp32_1");
        }
    }

    void reconnect() {
        while (!client.connected()) {
            Serial.print("Attempting MQTT connection...");
            if (client.connect(device_name)) {
                Serial.println("connected");
                client.subscribe("my_topic");
            }
            else {
                Serial.print("failed, rc=");
                Serial.print(client.state());
                Serial.println(" try again in 1 second");
                delay(1000);
            }
        }
    }

    void setup() {
        Serial.begin(115200);
        setup_wifi();
        client.setServer(mqtt_server, 1883);
        client.setCallback(callback);
    }

    void loop() {
        if (!client.connected()) {
            reconnect();
        }
        client.loop();
    }

```

## b) ESP32 – 2 Source Code

```

#include <Arduino.h>
#include <WiFi.h>
#include <PubSubClient.h>

const char* ssid = "ABA_ROBOTICS2";
const char* password = "aba1234567";
const char* mqtt_server = "192.168.0.138";

WiFiClient espClient;
PubSubClient client(espClient);
char msg[50];
int value = 0;

```

```

const char* device_name = "ESP32_2";

void setup_wifi() {
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);

    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println("");
    Serial.println("WiFi connected");
    Serial.print("IP address : ");
    Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* message, unsigned int length) {
    Serial.print("Message: ");
    String messageTemp;

    for (int i = 0; i < length; i++) {
        Serial.print((char)message[i]);
        messageTemp += (char)message[i];
    }
    Serial.println();

    if (String(topic) == "my_topic") {
        client.publish("my_topic2", "esp32-2");
    }
}

void reconnect() {
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        if (client.connect(device_name)) {
            Serial.println("connected");
            client.subscribe("my_topic");
        }
        else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" try again in 1 second");
            delay(1000);
        }
    }
}

```

```

void setup() {
    Serial.begin(115200);
    setup_wifi();
    client.setServer(mqtt_server, 1883);
    client.setCallback(callback);
}

void loop() {
    if (!client.connected()) {
        reconnect();
    }
    client.loop();
}

```

### c) Raspberry-Pi Source Code

```

import paho.mqtt.client as mqtt
from time import time, sleep, time_ns

mqtt_address = '192.168.0.138'

def on_connect(client,userdata,flags,rc):
    print('Connected with result code '+str(rc))
    client.subscribe('my_topic2')

def on_message(client,userdata,msg):
    global start
    print('Message received from '+ msg.topic + ' : ' + str(msg.payload))
    print(time()-start)

def on_publish(client,userdata,mid):
    print('data published')

def main():
    global start, mqtt_client

    mqtt_client.loop_start()
    start = time()
    timeout = time() + 1
    while time() < timeout : pass
    mqtt_client.publish('my_topic','r')

if __name__ == '__main__':
    mqtt_client = mqtt.Client()
    mqtt_client.on_connect = on_connect
    mqtt_client.on_message = on_message
    mqtt_client.on_publish = on_publish
    mqtt_client.connect(mqtt_address, 1883)

    while True:
        main()

```

#### d) Screenshots and Descriptions

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
> Executing task: platformio device monitor <

--- Available filters and text transformations: colorize, debug, default
, direct, esp32_exception_decoder, hexlify, log2file, nocontrol, printab
le, send_on_enter, time
--- More details at http://bit.ly/pio-monitor-filters

--- Available ports:
--- 1: /dev/ttyUSB0 'CP2102 USB to UART Bridge Controller - CP2
102 USB to UART Bridge Controller'
--- 2: /dev/ttyUSB1 'CP2102 USB to UART Bridge Controller - CP2
102 USB to UART Bridge Controller'
--- Enter port index or full name: 1
--- forcing DTR inactive
--- forcing RTS inactive
--- Miniterm on /dev/ttyUSB0 115200,8,N,1 ---
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0018,len:4
load:0x3fff001c,len:1044
load:0x40078000,len:10124
load:0x40080400,len:5828
entry 0x400806a8

Connecting to ABA_ROBOTICS2
.....
WiFi connected
```

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
> Executing task: platformio device monitor <

--- Available filters and text transformations: colorize, debug, default
, direct, esp32_exception_decoder, hexlify, log2file, nocontrol, printab
le, send_on_enter, time
--- More details at http://bit.ly/pio-monitor-filters

--- Available ports:
--- 1: /dev/ttyUSB0 'CP2102 USB to UART Bridge Controller - CP2
102 USB to UART Bridge Controller'
--- 2: /dev/ttyUSB1 'CP2102 USB to UART Bridge Controller - CP2
102 USB to UART Bridge Controller'
--- Enter port index or full name: 2
--- forcing DTR inactive
--- forcing RTS inactive
--- Miniterm on /dev/ttyUSB1 115200,8,N,1 ---
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0018,len:4
load:0x3fff001c,len:1044
load:0x40078000,len:10124
load:0x40080400,len:5828
entry 0x400806a8

Connecting to ABA_ROBOTICS2
.....
WiFi connected
```

Picture 6. Serial Monitor from ESP32 1 (Left) and 2 (Right) Connection Attempt

```
IP address : 192.168.0.151
Attempting MQTT connection...connected
Message: r
Message: r
Message: r
Message: r
Message: r
Message: r
Message: r
Message: r
Message: r
Message: r
Message: r
Message: r
Message: r
Message: r
Message: r
Message: r
```

```
IP address : 192.168.0.150
Attempting MQTT connection...connected
Message: r
Message: r
Message: r
Message: r
Message: r
Message: r
Message: r
Message: r
Message: r
Message: r
Message: r
Message: r
Message: r
Message: r
Message: r
Message: r
```

Picture 7. Serial Monitor from ESP32 1 (Left) and 2 (Right) Received Message

Status	data published Message received from my_topic2 : b'esp32_1' 0.09965276718139648
Compiler	Message received from my_topic2 : b'esp32-2' 0.11531782150268555
Messages	data published Message received from my_topic2 : b'esp32_1' 0.12959504127502441
Scribble	Message received from my_topic2 : b'esp32-2' 0.1454164981842041
Terminal	

Picture 8. Terminal from Raspberry-Pi

Picture 6 shows the attempt from both ESP32 to connect to the network. “Available ports:” option appears because we connect 2 ESP32 simultaneously. As we can see, both ESP32 are connected to the network.

Picture 7 shows the single character message ‘r’ received from Raspberry-Pi, as the result of both STM32 subscribe to topic “my\_topic”.

Picture 8 shows the terminal from Raspberry-Pi. First, it prints string “data published” as a note that it has published data to “my\_topic”. Then, it prints the message received from

“my\_topic2” which ‘contains’ 2 message sent from ESP32-1 and ESP32-2 simultaneously, and the time elapsed from loop start time to receiving first message from ESP32-1 and ESP32-2. After some analysis, we get the minimum time needed is 66ms and maximum is around 130ms.

Reference :

<https://github.com/eclipse/paho.mqtt.python>

#### 4. TCP Communication Between 3 ESP32 Devices

- 3 ESP32 communicates each other by PING-ing its next ESP then wait for other’s PING and so on.
- What you need:
  - 3 ESP32 with micro USB cable
- These source code were tested using PlatformIO in Visual Studio Code with .ini configuration as below :

```
platform = espressif32
board = esp32doit-devkit-v1
framework = arduino
monitor_speed = 115200
monitor_rts = 0
monitor_dtr = 0
```

- This examples use “WiFi.h” library
- This code can be accessed at :

<https://github.com/bimanjayaaji/raspberry-pi--esp32-comm-TCP.git>

a) ESP32 – A

```
/*
   Source Code ESP32 TCP-Communication
   ESP - A (Inisiator)
*/

#include <Arduino.h>
#include <WiFi.h>

const char* ssid = "ABA_ROBOTICS2";
const char* password = "aba1234567";

const char* host_target = "192.168.0.150";
const int port_target = 8080;

WiFiServer server(8088);
WiFiClient client;
String local_IP;
IPAddress sender_IP;
```

```

uint16_t sender_port;

unsigned long loop_start;
unsigned long start;
uint8_t data[30];

void connect2wifi(){
    Serial.print("Connecting to ");
    Serial.println(ssid);

    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected with IP address: ");
    local_IP = WiFi.localIP().toString();
    Serial.println(local_IP);
}

void connect2default(){
    Serial.print("Connecting to : "); Serial.println(host_target);
    while (!client.connect(host_target,port_target)) {
        delay(500);
    }
    Serial.print("Connected to : "); Serial.println(host_target);
}

void disconnect(){
    client.stop();
    Serial.println("Disconnected");
}

void initiator(){
    loop_start = millis();
    client.print("ping from " + local_IP);
    Serial.print("PING sent to"); Serial.println(host_target);
}

void get_ping(){
    Serial.print("Waiting to get msg ... ");
    int x = 0;
    while (x == 0){
        WiFiClient client = server.available();

        if (client) {
            Serial.println("Msg incoming");
            while (client.connected()) {
                if (client.available()) {
                    int len = client.read(data, 30);

```

```

        client.print("response from " + local_IP);
        Serial.print("Msg : "); Serial.println((char *)data);
        Serial.print("Response sent to :");
        Serial.println(client.remoteIP());
        x = 1;
        break;
    }
}

}

}

}

void get_response(){
    Serial.println("Waiting for response...");
    int x = 0;
    while (x == 0){
        if (client.available()) {
            int len = client.read(data, 30);
            Serial.print("CYCLE-TIME : "); Serial.println(millis()-loop_start);
            Serial.print("Res : "); Serial.println((char *)data);
            x = 1;
        }
    }
    disconnect();
    delay(10);
}

void delaying(){
    Serial.println("Delaying 10secs");
    unsigned long start = millis();
    unsigned long timeout = start + 10000;
    while (millis() < timeout);
}

void setup(){
    Serial.begin(115200);
    connect2wifi();
    server.begin();

    Serial.println("");
    connect2default();
    Serial.println("");
    initiator();
}

void loop(){
    Serial.println("");
    get_response();

    Serial.println("");
    get_ping();

```

```

        Serial.println("");
        disconnect();

        Serial.println("");
        delaying();

        Serial.println("");
        connect2default();

        Serial.println("");
        initiator();
    }

```

b) ESP32 – B

```

/*
    Source Code ESP32 TCP-Communication
    ESP - B
*/

#include <Arduino.h>
#include <WiFi.h>

const char* ssid = "ABA_ROBOTICS2";
const char* password = "aba1234567";

const char* host_target = "192.168.0.111";
const int port_target = 8000;

WiFiServer server(8080);
WiFiClient client;
String local_IP;
IPAddress sender_IP;
uint16_t sender_port;

unsigned long loop_start;
unsigned long start;
uint8_t data[30];

void connect2wifi(){
    Serial.print("Connecting to ");
    Serial.println(ssid);

    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected with IP address: ");

```



```

        Serial.println(WiFi.localIP());
        local_IP = WiFi.localIP().toString();
    }

void connect2default(){
    Serial.print("Connecting to : "); Serial.println(host_target);
    while (!client.connect(host_target,port_target)) {
        delay(500);
    }
    Serial.print("Connected to : "); Serial.println(host_target);
}

void disconnect(){
    client.stop();
    Serial.println("Disconnected");
}

void initiator(){
    loop_start = millis();
    client.print("ping from " + local_IP);
    Serial.print("PING sent to"); Serial.println(host_target);
}

void get_ping(){
    Serial.print("Waiting to get msg ... ");
    int x = 0;
    while (x == 0){
        WiFiClient client = server.available();

        if (client) {
            Serial.println("Msg incoming");
            while (client.connected()) {
                if (client.available()) {
                    int len = client.read(data, 30);
                    client.print("response from " + local_IP);
                    Serial.print("Msg : "); Serial.println((char *)data);
                    Serial.print("Response sent to :");
                    Serial.println(client.remoteIP());
                    x = 1;
                    break;
                }
            }
        }
    }
}

void get_response(){
    Serial.println("Waiting for response...");
    int x = 0;
    while (x == 0){
        if (client.available()) {
            int len = client.read(data, 30);

```

```

        Serial.print("CYCLE-TIME : "); Serial.println(millis()-loop_start);
        Serial.print("Res : "); Serial.println((char *)data);
        x = 1;
    }
}
disconnect();
delay(10);
}

```

```

void delaying(){
    Serial.println("Delaying 10secs");
    unsigned long start = millis();
    unsigned long timeout = start + 10000;
    while (millis() < timeout);
}

```

```

void setup(){
    Serial.begin(115200);
    connect2wifi();
    server.begin();
}

```

```

void loop(){
    Serial.println("");
    get_ping();

    Serial.println("");
    disconnect();

    Serial.println("");
    delaying();

    Serial.println("");
    connect2default();

    Serial.println("");
    initiator();

    Serial.println("");
    get_response();
}

```

c) ESP32 – C

```

/*
    Source Code ESP32 TCP-Communication
    ESP - C
*/

```

```

#include <Arduino.h>
#include <WiFi.h>

```

```

const char* ssid = "ABA_ROBOTICS2";
const char* password = "aba1234567";

const char* host_target = "192.168.0.151";
const int port_target = 8088;

WiFiServer server(8000);
WiFiClient client;
String local_IP;
IPAddress sender_IP;
uint16_t sender_port;

unsigned long loop_start;
unsigned long start;
uint8_t data[30];

void connect2wifi(){
    Serial.print("Connecting to ");
    Serial.println(ssid);

    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected with IP address: ");
    Serial.println(WiFi.localIP());
    local_IP = WiFi.localIP().toString();
}

void connect2default(){
    Serial.print("Connecting to : "); Serial.println(host_target);
    while (!client.connect(host_target,port_target)) {
        delay(500);
    }
    Serial.print("Connected to : "); Serial.println(host_target);
}

void disconnect(){
    client.stop();
    Serial.println("Disconnected");
}

void initiator(){
    loop_start = millis();
    client.print("ping from " + local_IP);
    Serial.print("PING sent to"); Serial.println(host_target);
}

```

```

void get_ping(){
    Serial.print("Waiting to get msg ... ");
    int x = 0;
    while (x == 0){
        WiFiClient client = server.available();

        if (client) {
            Serial.println("Msg incoming");
            while (client.connected()) {
                if (client.available()) {
                    int len = client.read(data, 30);
                    client.print("response from " + local_IP);
                    Serial.print("Msg : "); Serial.println((char *)data);
                    Serial.print("Response sent to :");
                    Serial.println(client.remoteIP());
                    x = 1;
                    break;
                }
            }
        }
    }
}

void get_response(){
    Serial.println("Waiting for response...");
    int x = 0;
    while (x == 0){
        if (client.available()) {
            int len = client.read(data, 30);
            Serial.print("CYCLE-TIME : "); Serial.println(millis()-loop_start);
            Serial.print("Res : "); Serial.println((char *)data);
            x = 1;
        }
    }
    disconnect();
    delay(10);
}

void delaying(){
    Serial.println("Delaying 10secs");
    unsigned long start = millis();
    unsigned long timeout = start + 10000;
    while (millis() < timeout);
}

void setup(){
    Serial.begin(115200);
    connect2wifi();
    server.begin();
}

```

```

void loop(){
    Serial.println("");
    get_ping();

    Serial.println("");
    disconnect();

    Serial.println("");
    delaying();

    Serial.println("");
    connect2default();

    Serial.println("");
    initiator();

    Serial.println("");
    get_response();
}

```

#### d) Screenshots and Descriptions



Picture 9. Terminal from ESP-A, ESP – B, and ESP – C (top to bottom)

Picture 9 shows every serial printing from each ESP32, ESP-A to ESP-C from top to bottom. The cycle starts from ESP-A sending message to ESP-B, then it waits for ESP-B to reply the message. After that, ESP-A will print the time required to send and receive the message, then, ESP-A will go to “hibernate” mode waiting for message from ESP-C. Next phase, after ESP-B reply the message from ESP-A, it will sleep for 10 seconds, then the cycle is the same but ESP-B will send a message to ESP-C.

After several trial, the modulus time needed to send and receive message is about 150 ms – 200 ms. In some case, the time can jump to 20ms.