



WEEK 9

SOCKET SERVER DAN CLIENT



Arranged By:

Rajendra Rakha Arya Prabaswara
(1941720080/20)

PROGRAM STUDI D-IV TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG



PRACTICUM 1

1. Create the server.py using nano or anotherone, just copy paste it from jbbsheet

```
GNU nano 2.9.3 server.py

import socket
from threading import Thread

# Multithreaded Python server
class ClientThread(Thread):

    def __init__(self, ip, port):
        Thread.__init__(self)
        self.ip = ip
        self.port = port
        print("Incoming connection from " + ip + ":" + str(port))

    def run(self):
        while True:
            try:
                data = conn.recv(2048)
                if len(data) == 0:
                    break

                print("length: " + str(len(data)))
                print("Server received data:", data)
                # MESSAGE = input("Input response:")
                MESSAGE = "OK"
                conn.send(MESSAGE.encode("utf8")) # echo
            except Exception as e:
                print(e)
                break

TCP_IP = "0.0.0.0"
TCP_PORT = 1886
BUFFER_SIZE = 20

tcpServer = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
tcpServer.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
tcpServer.bind((TCP_IP, TCP_PORT))
threads = []

while True:
    tcpServer.listen(4)
    print("Server started on " + TCP_IP + " port " + str(TCP_PORT))
    (conn, (ip, port)) = tcpServer.accept()
    newthread = ClientThread(ip, port)
    newthread.start()
    threads.append(newthread)

for t in threads:
    t.join()

Get Help  Write Out  Where Is  Cut Text  Justify  Cur Pos  Undo  Mark Text
Exit      Read File  Replace  Uncut Text  To Linter  Go To Line  Redo  Copy Text
```

2. Change the file permission of server.py to make it excutable using chmod +x

```
-rwxr-xr-x 1 root root 1267 May 17 03:43 server.py
```

3. Execute server.py

```
Server started on 0.0.0.0 port 1886
```



4. Test Connection by sending message

```
Server started on 0.0.0.0 port 1886
Incoming connection from 192.168.3.25:10593
Server started on 0.0.0.0 port 1886
length: 1
Server received data: b'h'
length: 1
Server received data: b'a'
length: 1
Server received data: b'l'
length: 1
Server received data: b'l'
length: 1
Server received data: b'o'
length: 1
Server received data: b'a'
length: 1
Server received data: b'b'
length: 1
Server received data: b'd'
length: 1
Server received data: b'u'
length: 1
Server received data: b'l'
```

```
Telnet 192.168.3.247
OKaOKlOKlOKoOKaOKbOKdOKuOKlOK_
```

After successfully running socket sever, then it is necessary to create a socket client that runs on the controller or ESP8266 Amica or Lolita that you have. Create the following code To be able to communicate with the server socket, ESP8266 already has a wifi module ready to use. Change the code below to suit your needs.

5. Change the Platformio.in setting

```
practicum1 > platformio.ini
1 ; PlatformIO Project Configuration File
2 ;
3 ; Build options: build flags, source filter
4 ; Upload options: custom upload port, speed and extra flags
5 ; Library options: dependencies, extra library storages
6 ; Advanced options: extra scripting
7 ;
8 ; Please visit documentation for the other options and examples
9 ; https://docs.platformio.org/page/projectconf.html
10
11 [env:esp12e]
12 platform = espressif8266
13 board = esp12e
14 framework = arduino
15 upload_speed = 115200
16 monitor_speed = 115200
```

6. Copy & Paste From Example Code



7. Build Code And Upload to NodeMCU

```
10
11 void connect_wifi()
12 {
13     Serial.printf("Connecting to %s ", ssid);
14     WiFi.begin(ssid, password);
15     while (WiFi.status() != WL_CONNECTED)
16     {
17         delay(500);
18         Serial.print(".");
19     }
20     Serial.println(" connected");
21 }
```

U
U
U

TERMINAL PROBLEMS 5 OUTPUT DEBUG CONSOLE PlatformIO: Mc

[Sending a request]
[Response:]
OK

[Disconnected]

[Connecting to 192.168.3.247 ... connected]
[Sending a request]
[Response:]

8. Observe in Server

```
Server received data: b'Hai from ESP8266'
Incoming connection from 192.168.3.233:57016
Server started on 0.0.0.0 port 1886
length: 16
Server received data: b'Hai from ESP8266'
Incoming connection from 192.168.3.233:64527
Server started on 0.0.0.0 port 1886
length: 16
Server received data: b'Hai from ESP8266'
Incoming connection from 192.168.3.233:50361
Server started on 0.0.0.0 port 1886
length: 16
Server received data: b'Hai from ESP8266'
Incoming connection from 192.168.3.233:60007
Server started on 0.0.0.0 port 1886
length: 16
```



PRACTICUM 2

1. Create Server.py

```
server.py E:\...NOT platformio.ini .\ U server.py E:\...practicum-2 1, U PIO Home main.cpp platf
practicum-2 > server.py ClientThread > run
1 import socket
2 from threading import Thread
3
4
5 # Multithreaded Python server
6 class ClientThread(Thread):
7
8     def __init__(self, ip, port):
9         Thread.__init__(self)
10        self.ip = ip
11        self.port = port
12        print("Incoming connection from " + ip + ":" + str(port))
13    def run(self):
14        while True:
15            try:
16                MESSAGE = input("Input response:")
17                conn.send(MESSAGE.encode("utf8")) # echo
18            except Exception as e:
19                print(e)
20                break
21            sleep(0.25)
22
23
24 TCP_IP = "0.0.0.0"
25 TCP_PORT = 1886
26 BUFFER_SIZE = 20
27
28 tcpServer = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
29 tcpServer.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
30 tcpServer.bind((TCP_IP, TCP_PORT))
31 threads = []
```

2. Change Server.py in server

```
GNU nano 2.9.3 server2.py Modified
import socket
from threading import Thread

# Multithreaded Python server
class ClientThread(Thread):

    def __init__(self, ip, port):
        Thread.__init__(self)
        self.ip = ip
        self.port = port
        print("Incoming connection from " + ip + ":" + str(port))
    def run(self):
        while True:
            try:
                MESSAGE = input("Input response:")
                conn.send(MESSAGE.encode("utf8")) # echo
            except Exception as e:
                print(e)
                break
            sleep(0.25)

TCP_IP = "0.0.0.0"
TCP_PORT = 1886
BUFFER_SIZE = 20

tcpServer = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
tcpServer.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
tcpServer.bind((TCP_IP, TCP_PORT))
threads = []

while True:
    tcpServer.listen(4)
    print("Server started on " + TCP_IP + " port " + str(TCP_PORT))
    (conn, (ip, port)) = tcpServer.accept()
    newthread = ClientThread(ip, port)
    newthread.start()
    threads.append(newthread)

for t in threads:
    t.join()

Save modified buffer? (Answering "No" will DISCARD changes.)
Yes
No Cancel
```



3. Run The Server.py

```
Server started on 0.0.0.0 port 1886
Incoming connection from 192.168.3.233:59950
Server started on 0.0.0.0 port 1886
Input response:
```

4. Change The Code Like The Example Code

5. Success

The screenshot shows an IDE with a C++ code file named `main.cpp` in the `src` directory. The code implements a server that listens on port 1886. When a client connects, it prints a response and reads a string from the client. If the string is "led-on", it sets a pin to HIGH, delays for 3000ms, and sets it to LOW. Otherwise, it connects the server again and delays for 250ms. The terminal output shows the server starting and receiving multiple "Response:" messages.

```
practicum-2 > src > main.cpp > loop0
40 connect_wifi();
41 connect_server();
42 }
43
44 void loop()
45 {
46   if (client.connected())
47   {
48     Serial.print("[Response:]");
49     String line = client.readStringUntil('\n');
50     Serial.println(line);
51     if (line.equalsIgnoreCase("led-on"))
52     {
53       pinMode(D0, HIGH);
54       delay(3000);
55       pinMode(D0, LOW);
56     }
57   }else{
58     connect_server();
59   }
60   delay(250);
61 }
```

PlatformIO: Monitor (practicum-2) - Task

```
[Response:]
[Response:]
[Response:]
[Response:]
[Response:]
[Response:]
```