IoT exercises – Week 6

This week, we will start using the Net Client/Server and Socket to send/receive data. You will create the client and server for communication. NodeMCU/ESP8266 allows you to create many clients but only one server for both TCP/UDP communication. Particularly, the web socket will be more explored with multiple functions provided by the bespoke module with an emphasis on the TCP communication. The net module will be used in this practical and throughout the rest.

The official documentation is provided here again for your reference. Please remember, when you are building your own IoT project in the future, always refer to the bespoke built-in modules and read their documentation first, which will be helpful!!!

https://nodemcu.readthedocs.io/en/master/

The details of net module can be found in

https://nodemcu.readthedocs.io/en/master/modules/net/

Exercise 1:

In this exercise, you will need to create a client and communicate with the server.

- Get your NodeMCU/ESP8266 online first in the Station mode.
- Get a TCP socket tester app on your mobile device for test. (I chose the Simple TCP Socket Tester, but you can find any alternative application.)
- Now create a client in your NodeMCU/ESP8266, and see the received message and try sending out some messages from your mobile tester.

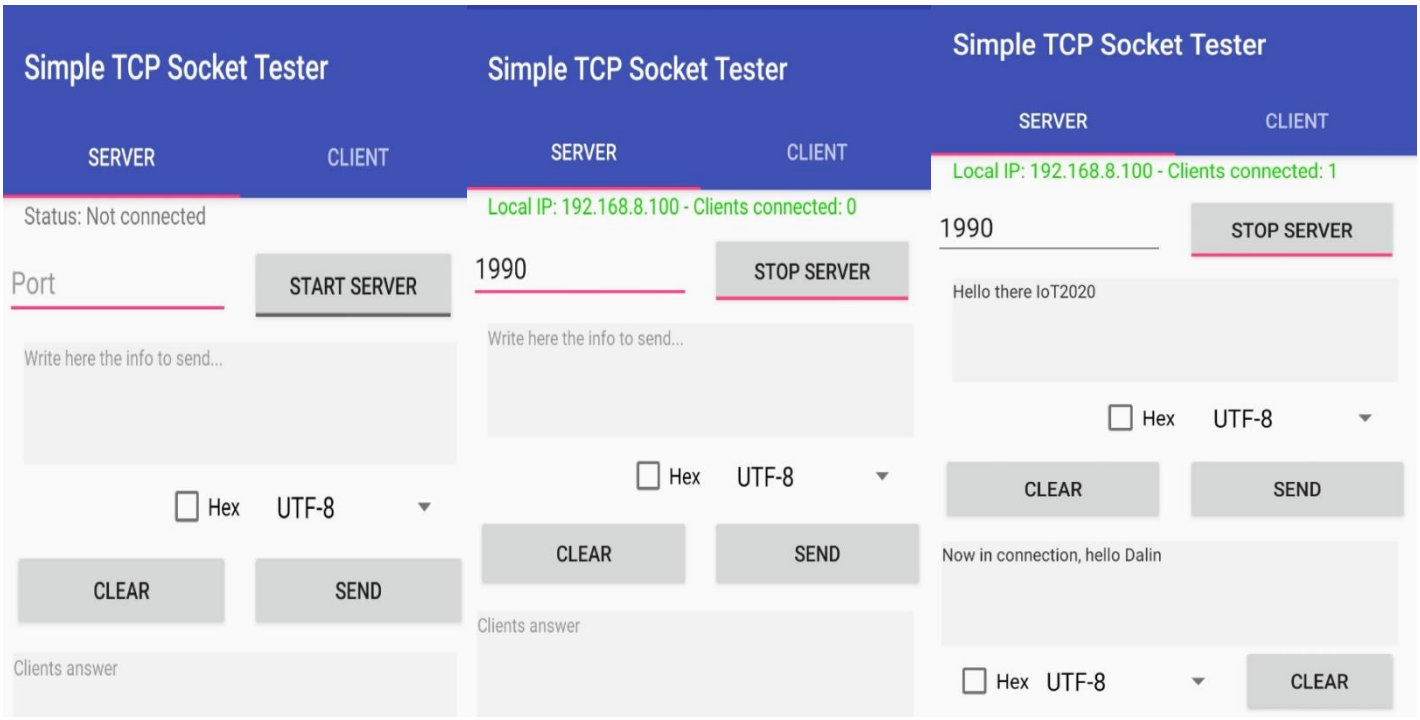
==========

```
--make sure you are online first - Dalin

cl = net.createConnection(net.TCP, 0)

--create a TCP based not encryped client

cl:on("connection",function(conn, s)
```

```
conn:send("Now in connection, hello Dalin\n")

end)

cl:on("disconnection",function(conn, s)

print(“Now we are disconnected\n”)

end)

cl:on("sent",function(conn, s)

print(“Message has been sent out\n”)

end)

cl:on("receive", function(conn, s)

print(“What we receive from the server\n” .. s .. “\n”)

end)

--on(event, function())

--event can be "connection", "reconnection", "disconnection", "receive" or "sent"

--function(net.socket[, para]) is the callback function.

--The first parameter of callback is the socket.

--If event is "receive", the second parameter is the received data as string.

--If event is "disconnection" or "reconnection", the second parameter is error code.

cl:connect(1990,”192.168.8.100”)

--the local IP of your test server
```

==========

- For testing, first you have to connect your mobile device to the same AP with NodeMCU/ESP8266.
- Then set the tester with a Port No. when it is a server
- Connect your NodeMCU/ESP8266 client to the IP address shown to you with the right Port.



Exercise 2:

In this exercise, you will need to create a server and listen to the client. The functions will be examined with the tester as well.

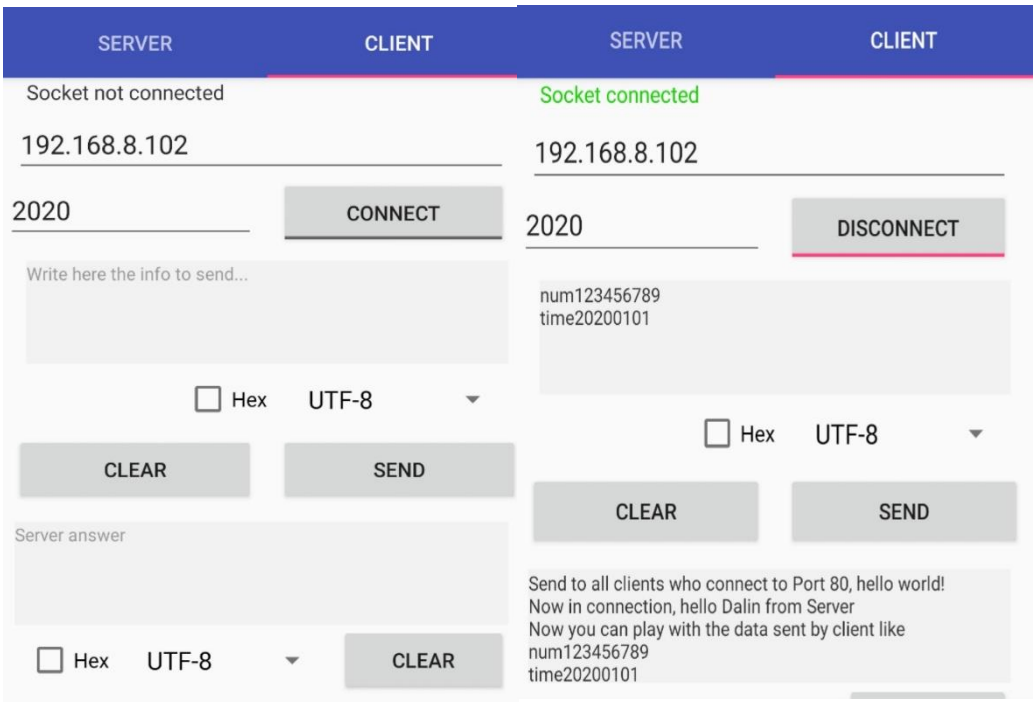
==========

```
srv = net.createServer(net.TCP,30)
--TCP, 30s for an inactive client to be disconnected
--try srv = net.createServer(net.UDP,10)
srv:listen(2020, function(conn)
      conn:send("Send to all clients who connect to Port 80, hello world! \n")
      conn:on("receive", function(conn, s)
            print(s)
            conn:send(s)
       end)
       conn:on("connection",function(conn, s)
            conn:send("Now in connection, hello Dalin from Server\n")
```

```
        end)
        conn:on("disconnection", function(conn, s)
            print("Now we are disconnected\n")
        end)
        conn:on("sent",function(conn, s)
            print("Message has been sent out from the Server\n")
        end)
        conn:on("receive", function(conn, s)
            print("What we receive from the Client\n" .. s .. "\n")
        end)
end
)
--srv:close()
```

==========

Your server is now listening to the event on the Port and able to send and receive messages from its clients. The data within the messages can be used for control variable to your IoT systems.



Exercise 3:

- You can try more clients to communicate with the same server at the same time (remember that the NodeMCU/ESP8266 can create multiple clients at the same time) to see how the client/server pairs work.
- Use the NodeMCU/ESP8266 as a server and your TCP socket tester as a controller. Send ON/OFF to turn on/off the LED light in your platform
- Please refer to the net module for more details. The socket based callback function and communication are IMPORTANT for you to work with IoT applications.

https://nodemcu.readthedocs.io/en/master/modules/net/

Exercise 4:

Now you have implemented both client and server using the net module. Recall the live demo in the first lecture. A mobile device and the NodeMCU/ESP8266 + DHT11 temperature and humidity meter are connected to the same AP. Within the LAN, you can browse the temperature and humidity with Internet browser by visiting the IP address of NodeMCU/ESP8266.

In this exercise, you will use the NodeMCU/ESP8266 in its server mode and listen to the refresh request from the client (your browser).

- The following codes might be needed. And the variable buf will be sent to the client as an argument containing temperature and humidity information.
- Include the code in your server. And identify the limitation for improvement.

==========

```
local status, temp, humi , temp_dec, humi_dec = dht.read11(pinDHT)

local buf = ""

if status == dht.OK then

print("DHT Temperature:"..temp.." Humidity:"..humi..”\n”)

end

buf=buf.."<html>"

buf=buf.."<head> <title>DHT11 LAN update</title> <meta http-equiv=\"refresh\" content=\"3\"> </head>"

buf=buf.."<body><p>Temperature: "..temp.."."..temp_dec.."C</p>"
```

```
buf=buf.."<p>Humidity: "..humi.."."..humi_dec.."%RH</p>"

buf=buf.."<p>Did you spot any limitation here ? And can you improve it ?".."</p></body></html>"
```

==========

Now you have implemented the LAN based DHT11 update server using the net TCP Server listening.

(Optional) Exercise 5:

Recall the enduser module you have used as a more user-friendly interface to get your device online by communication through your own mobile device.

In this exercise, the function of the enduser that you used will be implemented by the net module and functions. (Similar to the Exercise 4, but require different content to be sent from the server to the client/your browser.)

https://nodemcu.readthedocs.io/en/master/modules/enduser-setup/