TCP connection with Python.

BeagleBone Black as server, PC as client.

Author: Moises Reyes Pardo.

November 22, 2017

Contents

1 Main goal **2**

2 Equipment used **2**

2.1 BeagleBone Black as TCP Server 2

2.2 PC as TCP Client 2

3 Programming a TCP connection with Python. **3**

3.1 General diagram of the system. 3

3.2 Python software on BeagleBone Black 3

3.3 Python software on PC 6

4 Testing the whole system. **7**

1 Main goal

The main goal of this small project is adquiere experience with Python programming language, besides to create the basis for a future project that already exist in my mind.

This project create a communication channel over a pre existent LAN network, interconnecting different devices according to our desires. This way of communication hast to be enough robust and scalable so, in the future, could be access from anywhere.

2 Equipment used

2.1 BeagleBone Black as TCP Server

Table 1: BeagleBone Black - Relevant specifications.

BeagleBone model:	BeagleBone Black Rev C
O.S. installed:	bone-debian-9.2-iot-armhf-2017-10-10-4gb.img
Kernel release [uname -r]:	4.4.91-ti-r133
O.S. running [uname -o]:	GNU/Linux
Python version:	Python 3.5.3

2.2 PC as TCP Client

Table 2: PC - Relevant specifications.

O.S. running:	Windows 10 Home
Python version:	Python 3.6.3

3 Programming a TCP connection with Python.

3.1 General diagram of the system.

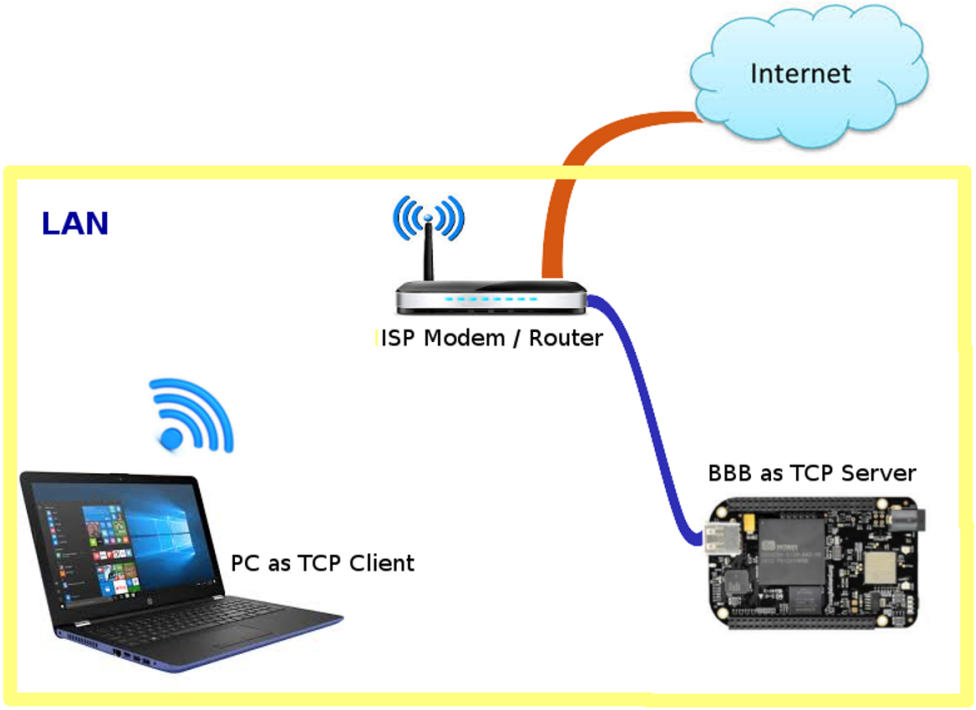

Figure 1: General diagram of the system.

3.2 Python software on BeagleBone Black

The "Figure 2" shows a flow diagram to explain the logic behiind the code used to programm the software `"Py_TCP_Server_v2"`, which implement a TCP server on BeagleBone Black hardware.

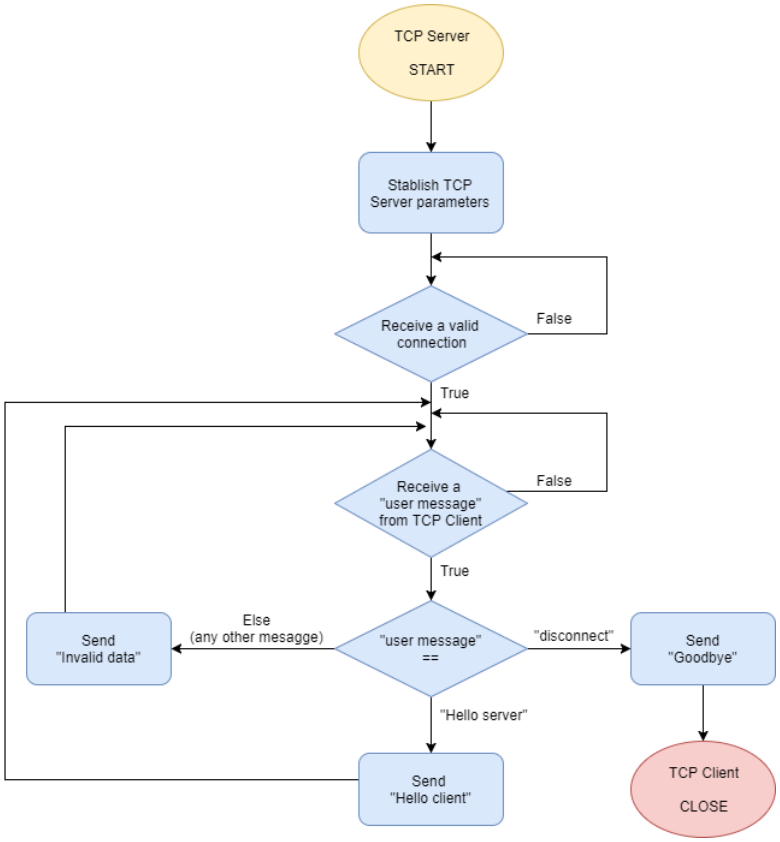

Figure 2: Flow diagram, "Py_TCP_Server_v2" software.

Now the complet source code of the software "Py_TCP_Server_v2" is given.

```
import socket
import netifaces as ni
from netifaces import AF_INET, AF_INET6, AF_LINK
import threading

class ClientThread(threading.Thread):
    def __init__(self, client, details):
        self.client = client
        self.details = details
        threading.Thread.__init__ ( self )
```

```
    def run(self):
        print('Received connection:', self.details[0])

        while True:

            data = self.client.recv(1024)
            client_details = self.details[0]
            print("[*] Received '", data, client_details,
                  "' from the client")
            strhello = "Hello server"
            strdisconnect = "disconnect"

            if data == strhello.encode('utf-8'):
                server_reply = 'Hello client'
                self.client.sendall(
                    server_reply.encode('utf-8'))
                print("Processing done.\n[*] Reply sent")
            elif data == strdisconnect.encode('utf-8'):
                server_reply = 'Goodbye'
                self.client.sendall(
                    server_reply.encode('utf-8'))
                self.client.close()
                print('Closed connection:',
                      self.details[0])
                break
            else:
                server_reply = 'Invalid data'
                self.client.sendall(
                    server_reply.encode('utf-8'))
                print("Processing done Invalid data.\n[*] "
                      "Reply sent")

#Get the IP address of LAN eth0 interface
ip = ni.ifaddresses('eth0')[AF_INET][0]['addr']
```

```
print (ip)
port = 1234
address = (ip,port)
server.bind(address)
server.listen(5)

while True:
    print("[*] Started listening on ", ip, ":", port)
    client, details = server.accept()
    ClientThread(client, details).start()
```

3.3 Python software on PC

The "Figure 3" shows a flow diagram to explain the logic behind the code used to programm the software `"Py_TCP_Client_v2"`, which implement a TCP client on a PC.

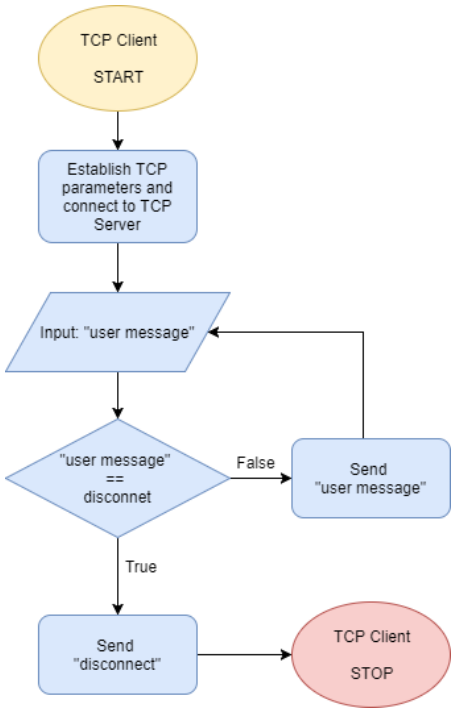

Figure 3: Flow diagram, `"Py_TCP_Client_v2"` software.

Now the complet source code of the software "Py_TCP_Client_v2" is given.

```
import socket
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# ((Ip,port)) of TCP server
client.connect(('192.168.2.110',1234))
while True:

    data = input('Type your message!: ')
    print("Message typed :",data)



    if data == "disconnect":
        client.sendall(data.encode('utf-8'))
        print("Message:'", data, "' was send.")
        print(client.recv(1024))
        client.close()
        break
    else:
        client.sendall(data.encode('utf-8'))
        print("Message: ", data, "was send.")
        print(client.recv(1024))
```

4 Testing the whole system.

In order to test the system and show you how it works in a real environment, a video was recorded. The video "Demo_TCP_connection_with_Python" shows in the same screen the BeagleBone Black prompt (ssh connection) running the TCP server and the PC running the TCP client software.

The "Figure 4" shows a captured image from that video, which expose a established connection between the BeagleBone Black as TCP server (IP 192.168.2.110:1234) a the PC as TCP client (IP 192.168.2.104:1234).

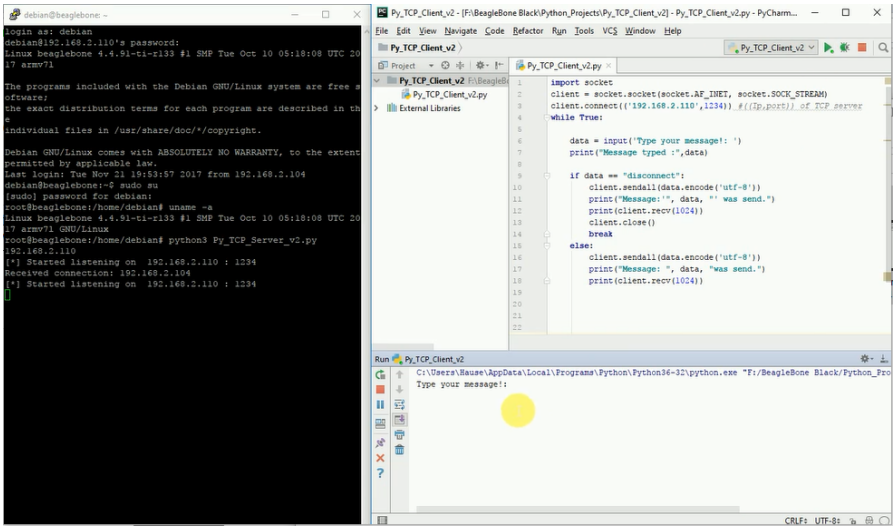

Figure 4: Capture 1 from video "Demo_TCP_connection_with_Python".

The "Figure 5" shows a second captured image from the video, which expose a fully interaction between the TCP server and the client, from the very beginnen until the command to close the connection is sended.

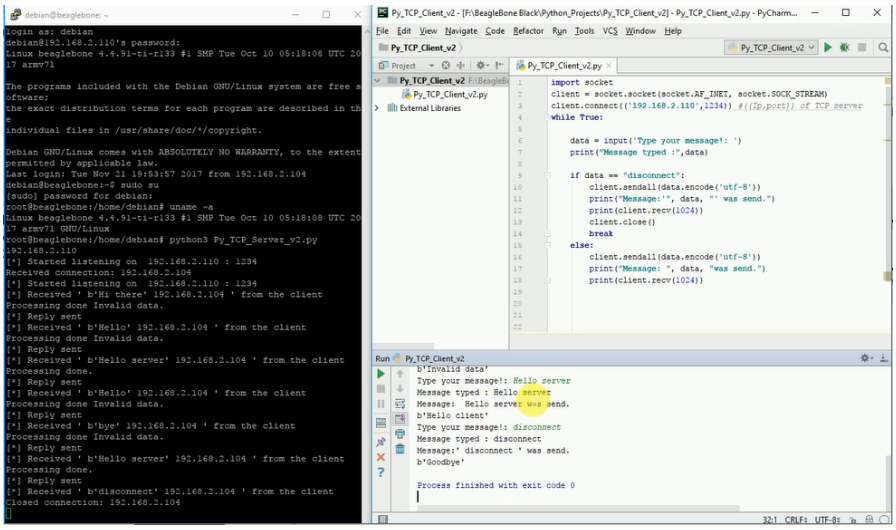

Figure 5: Capture 2 from video "Demo_TCP_connection_with_Python".