Assignment No. B-8

Roll No:

Aim :

Develop a network based application by setting IP address on BeagleBoard/ ARM Cortex A5.

Software Required :

- Linux Operating System
- GCC Compiler.

Hardware Required :

- Beaglebone Black/ ARM Cortex Processor
- Interfacing cables

Theory :

Sockets

Sockets are endpoints of a bidirectional communication channel . Sockets may communicate within a process , between two different processes on different machines. Sockets may be implemented over a number of different channel types: Unix domain sockets, TCP, UDP, and so on. The socket library provides specific classes for handling the common transports as well as a generic interface for handling the rest. To create a socket, you must use the socket.socket() function available in socket module, which has the general syntax:
s = socket.socket (socket_family, socket_type, protocol=0)
Here is the description of the parameters:

- **socket_family:** This is either AF_UNIX or AF_INET, as explained earlier.
- **socket_type:** This is either SOCK_STREAM or SOCK_DGRAM.
- **protocol:** This is usually left out, defaulting to 0.

Once you have socket object, then you can use required functions to create your client or server program. Following is the list of functions required:

Table 1: Socket Terms

Term	**Description**
Domain	The family of protocols that will be used as the transport mechanism. These values are constants such as AF_INET, PF_INET, PF_UNIX, PF_X25, and so on.
Type	The type of communications between the two endpoints, typically SOCK_STREAM for connection-oriented protocols and SOCK_DGRAM for connectionless protocols.
Protocol	Typically zero, this may be used to identify a variant of a protocol within a domain and type
Port	Each server listens for clients calling on one or more ports. A port may be a Fixnum port number, a string containing a port number, or the name of a service.

Table 2: **Socket Server Methods :**

Method	**Description**
s.bind()	This method binds address (hostname, port number pair) to socket.
s.listen()	This method sets up and start TCP listener.
s.accept()	This passively accept TCP client connection, waiting until connection arrives (blocking).

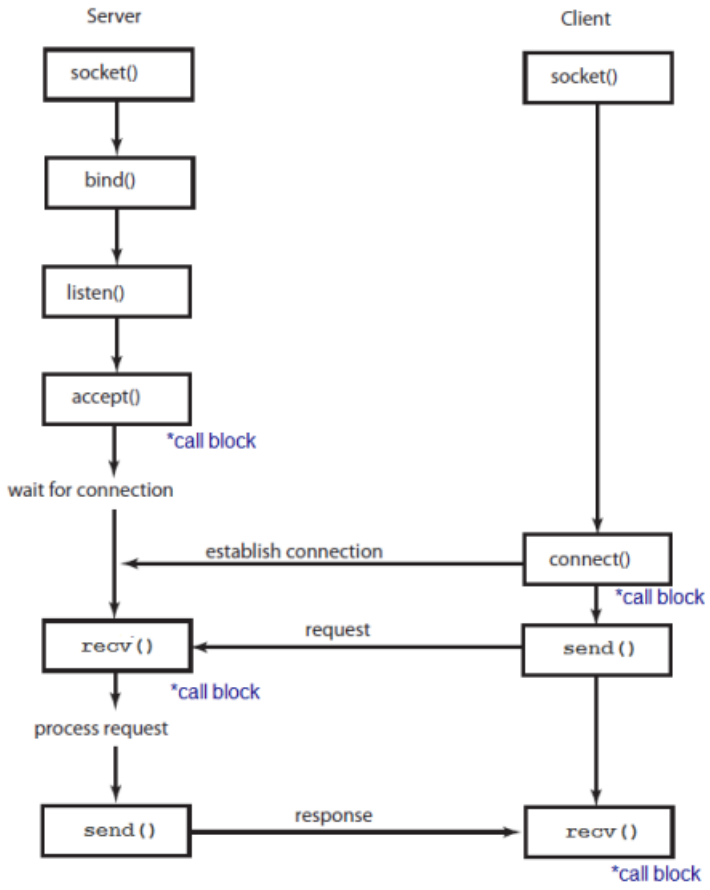
Table 3: **Client Server Methods :**

Method	**Description**
s.connect()	This method actively initiates TCP server connection.

Table 4: **General Socket Methods :**

Method	**Description**
s.recv()	This method receives TCP message
s.send()	This method transmits TCP message
s.recvfrom()	This method receives UDP message
s.sendto()	This method transmits UDP message
s.close()	This method closes socket
s.gethostname()	Returns the hostname.

Basic Client-Server Working



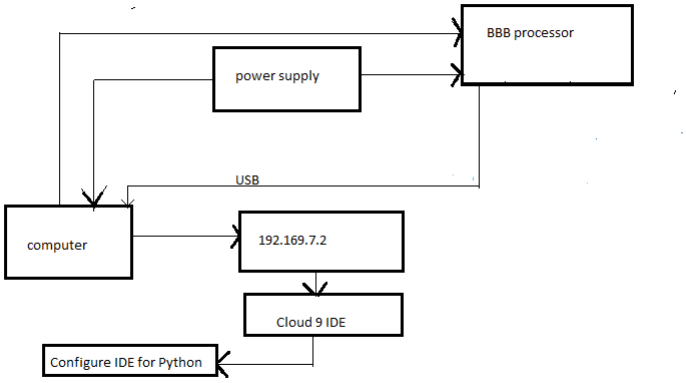
By default, the IP address of the Beaglebone Black is set dynamically using the DHCP server on our network. That is very useful; however, if we wish to map the Beaglebone to an external port (virtual server) on your network, or you wish to force your Beaglebone to have a specific identifiable location in your home you need to change our IP address to be static. The static address of BEAGLE BLACK BONE and Linux system should be same so it is useful to communicate between them.

Algorithm

- connect usb cable to BeagleBone Black and PC.

- Go to network interfaces of BeagleBone Black and change IP address of eth0.
- Connect BeagleBone Black to PC by LAN cable
- Type IP adress of BeagleBone Black in browser
- Select Cloud9E.
- Write Server code in Cloud9E
- Write Client code in gedit in PC
- Run Server
- Run Client
- Exit

Interfacing Diagram



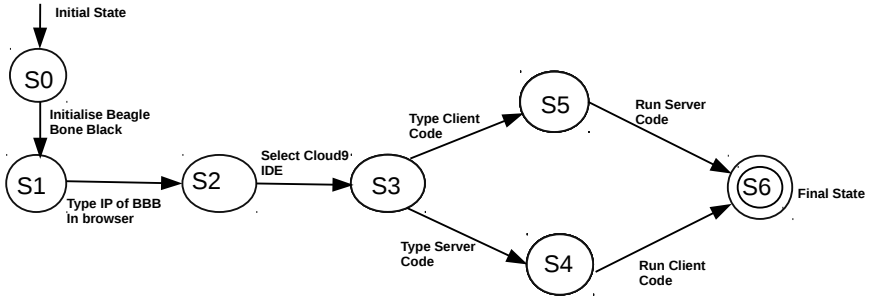
Mathematical Model

Let S be a set such that

S={s, e, i, o, f, DD, NDD, success, failure}

s= initial state

e = end state



i= input of the system.

o= output of the system.

f= functions

DD-Deterministic Data it helps identifying the load store functions or assignment functions.

NDD- It is Non deterministic data of the system S to be solved.

Success-Client-Server communicate successfully

Failure-Desired outcome not generated or forced exit due to system error.

States: { S0 ,S1 , S2 , S3 , S4 , S5 }

S0: initial State (Power supply)

S1: BeagleBone initialisation

S2: BeagleBone Black connected to Network

S3: Connected to Cloud9E

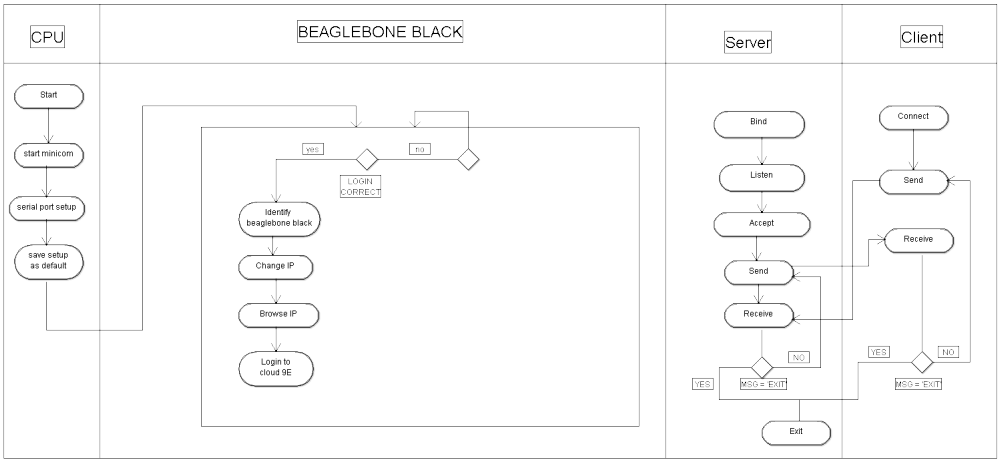
S4: Server code

S5: Client Code

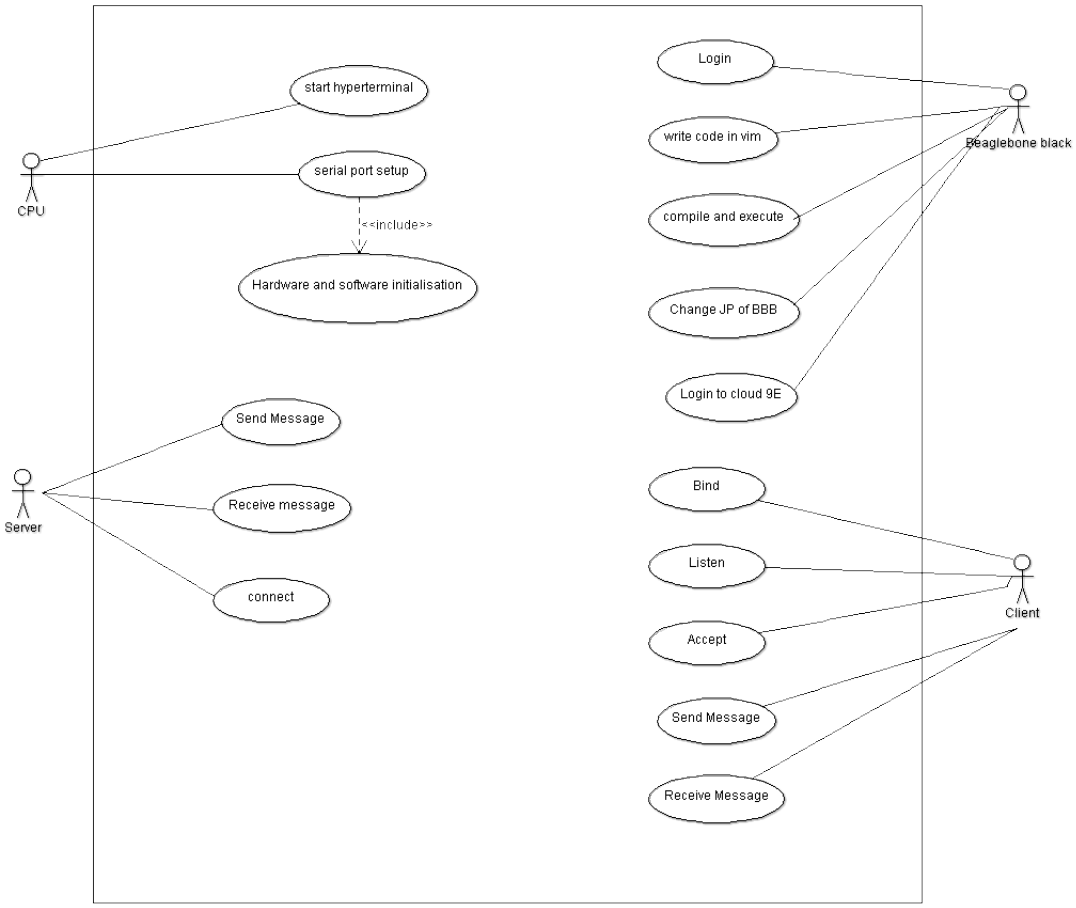
S6 : Server and Client Communication (Final State)

UML Diagrams :

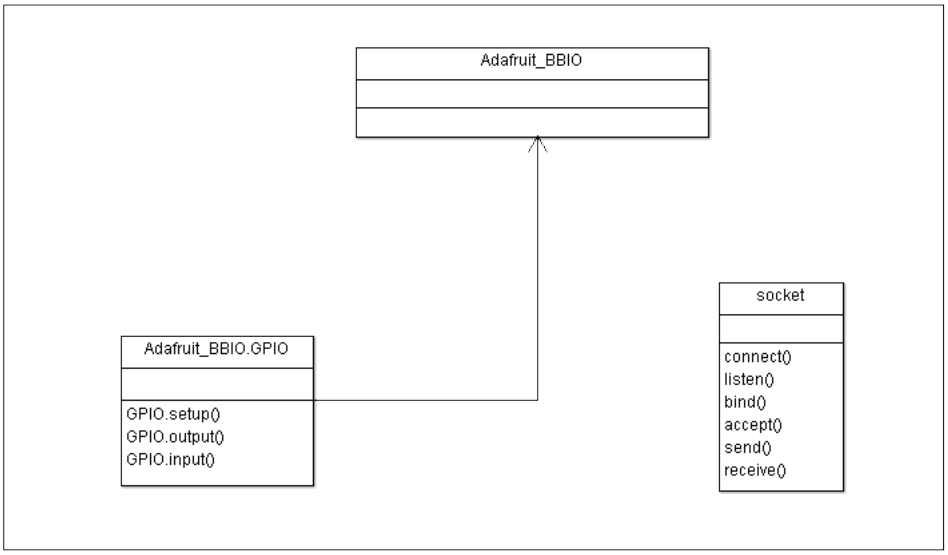
Activity Diagram



Use-case Diagram



Class Diagram



CONCLUSION :

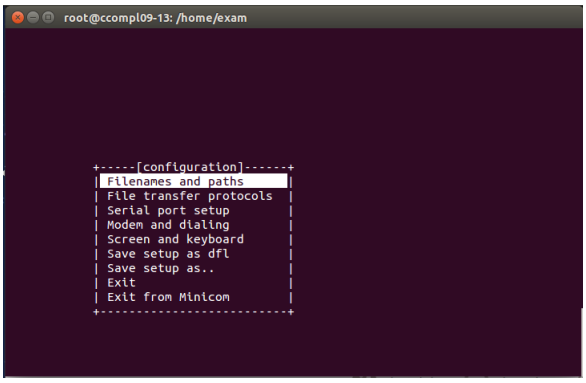
Hence, we develop network application like chat application using static IP address of BEAGLE BLACK BONE.

Course Outcomes :

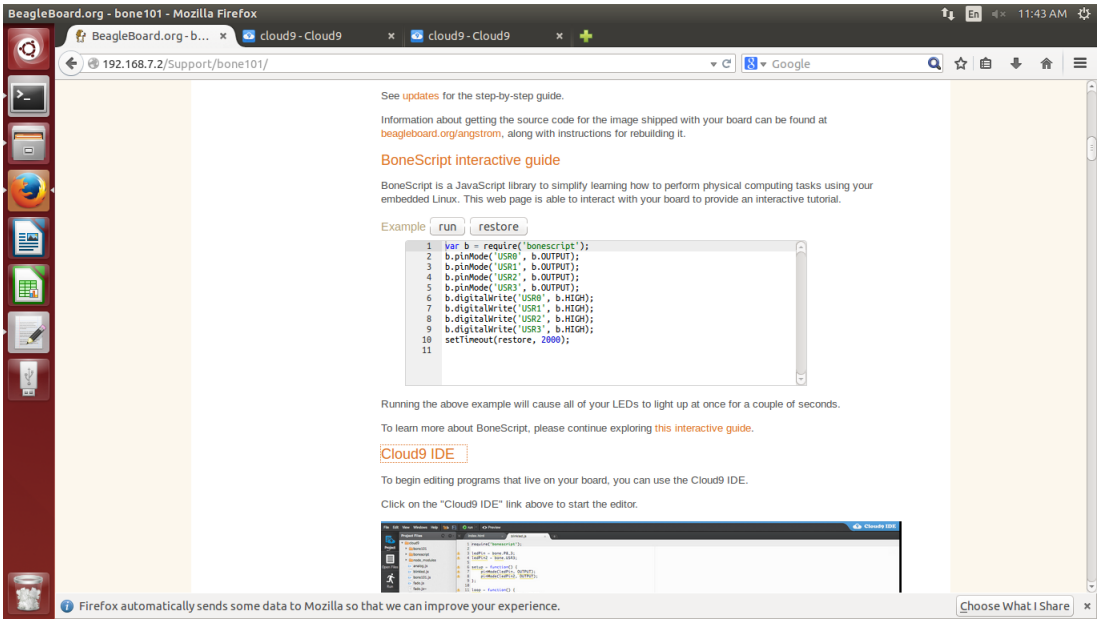
Course Outcomes	Tick [√]
Ability to perform multi-core, Concurrent and Distributed Programming	
Ability to perform Embedded Operating Systems Programming	
Ability to write Software Engineering Document	
Ability to perform Concurrent and Distributed Programming	

Output (Screenshots)

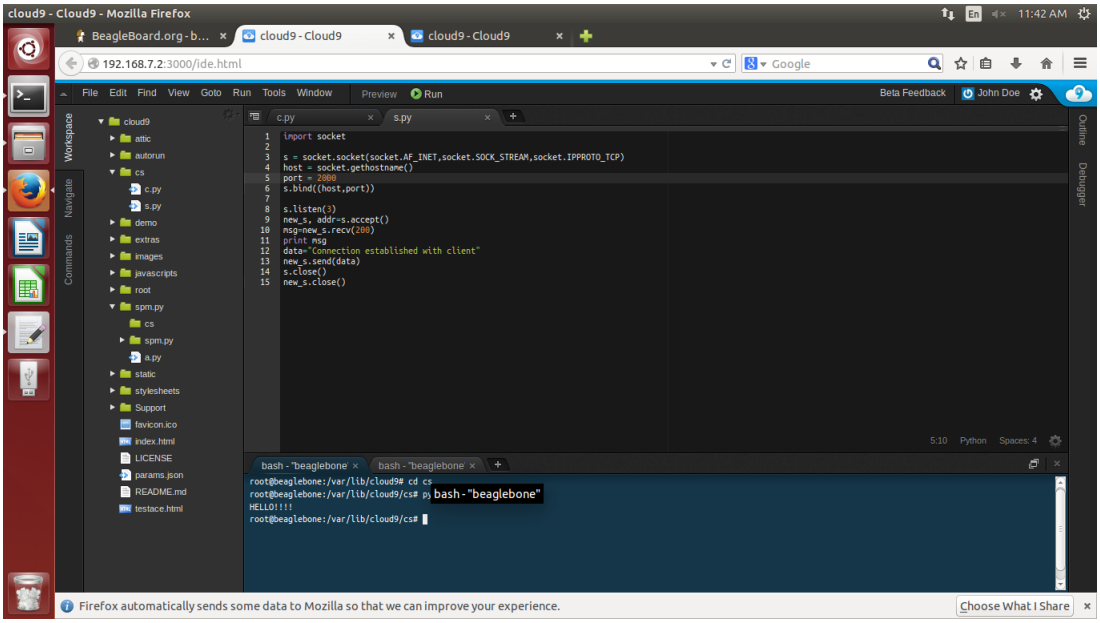
Minicom Terminal



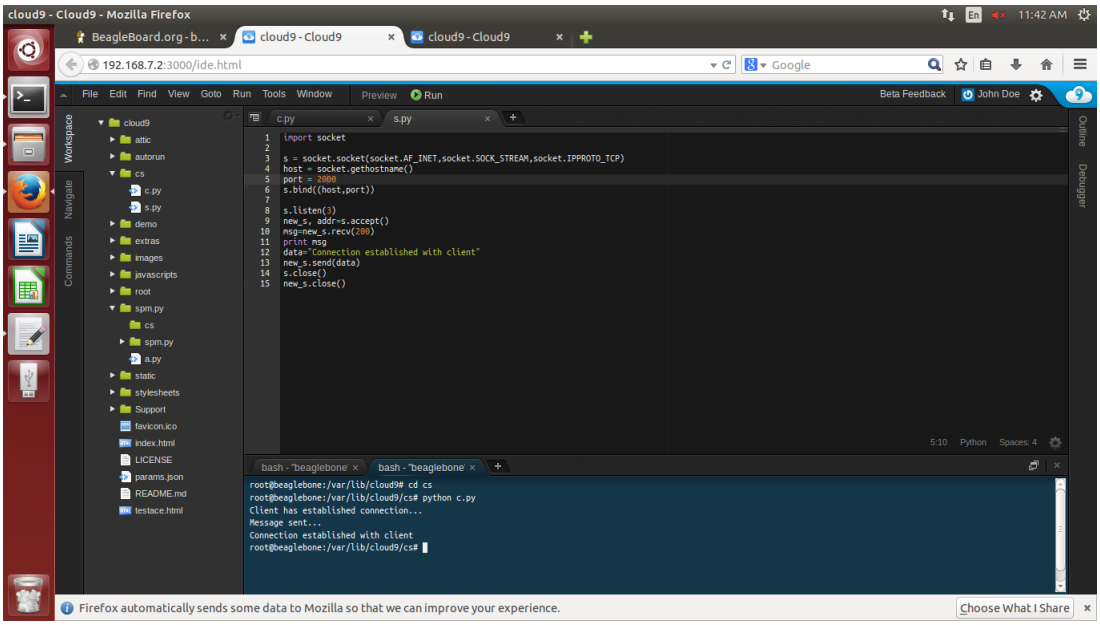
Cloud9E



Server Example



Client Example



Server Code :

```
import socket
s = socket.socket(socket.AF_INET,socket.SOCK_STREAM,socket.IPPROTO_TCP)
host = socket.gethostname()
port = 12348
s.bind((host,port))
s.listen(5)
while (True) :
    new_s, addr=s.accept()
    print 'Got connection from proxy....sending data...'
    user = new_s.recv(200)
    print 'User : ',user
    new_s.send("CONNECTION WAS SUCCESSFULL....")
s.close()
new_s.close()
```

Client Code :

```
# CLIENT CODE

import socket
s = socket.socket(socket.AF_INET,socket.SOCK_STREAM,socket.IPPROTO_TCP)
host = socket.gethostname()
port = 12347
print "CONNECTING ......"
s.connect((host, port))
me = raw_input('ROHAN : ')
s.sendall(me)
server = s.recv(200)
print 'SERVER :',server
s.close()
```