Assignment 10

Name : Samyak Shah

Class : TE Comp

Roll : 9085

Title:

Write a server application to be deployed on Raspberry-Pi /Beagle board. Write client applications to get services from the server application.

Aim/Objectives:

- To understand the working principle of client server with Raspberry Pi

Software:

- Raspbian OS (IDLE)

Theory:

- Sockets are the endpoints of a bidirectional communications channel.
- Sockets may communicate within a process, between processes on the same machine, or between processes on different continents.
- 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.

Socket Module

- To create a socket, you must use the socket.socket() function in socket module, which has the general syntax:

  s = socket.socket (socket_family, socket_type, protocol=0)

  socket_family: This is either AF_UNIX or AF_INET.

  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.

Server Socket Methods

- 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).
- s.connect() This method actively initiates TCP server connection.

General Socket Methods

- 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
- socket.gethostname() Returns the hostname.

Safety precautions:

- Raspberry-Pi provides 3.3V and 5V VCC pins □ Raspberry-Pi operates on 3.3V.
- Various sensors and actuators operate on different voltages.
- Read datasheet of a given sensor or an actuator and then use appropriate VCC pin to connect a sensor or an actuator.
- Ensure that signal voltage coming to the Raspberry-Pi from any sensor or actuator does not exceed 3.3V.
- If signal/data coming to Raspberry-Pi is greater than 3.3V then use voltage level shifter module to decrease the incoming voltage.
- The Raspberry-Pi is a costly device, hence you should show the circuit connections to your instructor before starting your experiment.

Procedure:

- Write the program as per the algorithm given.
- Save the program
- Run code using Run module.

Observation:

- Observe the output on console.

Code:

Client code :

```
import socket s=socket.socket()
s.connect(("localhost",1234))
data=raw_input("Enter the string ")
s.send(data) print " upper case is : ",
s.recv(1024) s.close()
```

Client temp:

```
import socket s=socket.socket()
s.connect(("localhost",1236))
#data=raw_input("Enter the string
") #s.send(data) print " Temp is : ",
s.recv(1024) s.close()
```

Server code:

```
import socket s=socket.socket()
s.bind(("localhost",1234))
s.listen(5) while True:
c,addr=s.accept()     print 'Got
Connection from ', addr
data=c.recv(10)
    c.send(data.upper())
    c.close()
```

Server temp:

```
import socket import
Adafruit_DHT
s=socket.socket()
s.bind(("localhost",1236))
s.listen(5)
while True:     print "waitting for client
commection"

    c,addr=s.accept()
hum,temp=Adafruit_DHT.read_retry(11,4)     print
'Got Connection from ', addr
```

```
#data=c.recv(10)
c.send(str(temp))
c.close()
```