IOE Server Flow-Chart

We first define a Class named Device, which has 4 attributes, “name”, “IP_address”, ”read_value”, and “target_value”

```
##define a Device object
class Device(object):
        name = None
        IP_address = None
        read_value = None
        target_value = None
```

We declare a list to store all the device objects to be created and create 2 initial devices and add them to the “Devices” list by calling “Devices.append(var_name)”

```
##create a list to store all the devices
Devices  = []
##set-up first two devices and add to the list
mydevice = Device()
mydevice.name = "Thermostat-Main Room"
mydevice.IP_address = '177.68.25.17'
mydevice.read_value = 19
mydevice.target_value = 23
Devices.append(mydevice)


mydevice = Device()
mydevice.name = "Thermostat-Living Room"
mydevice.IP_address = '177.68.25.18'
mydevice.read_value = 18
mydevice.target_value = 22
Devices.append(mydevice)
```

Then, we create a socket object by calling “socket.socket(socket.AF_INET, socket.SOCK_STREAM)” from the Socket library, and we bind the “sock” object to our local IP address and assign a desire port to it. Then set the “sock” to listen state, by calling sock.listen(1).

```
# Create a TCP/IP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# Bind the socket to the port
server_address = ('172.17.176.100', 10000)
print >>sys.stderr, 'starting up on %s port %s' % server_address
sock.bind(server_address)
# Listen for incoming connections
sock.listen(1)
```

↓

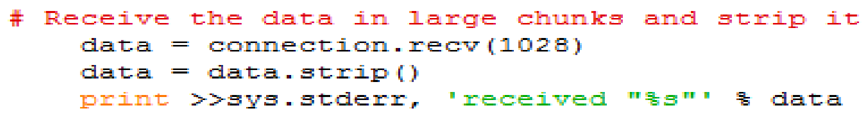
Enter While loop to wait for a connection, and when a connection arrives, the connection object and the client’s address are stored by calling “sock.accept()” method.

```
while True:
    # Wait for a connection
    print >>sys.stderr, 'waiting for a connection'
    connection, client_address = sock.accept()
```

↓

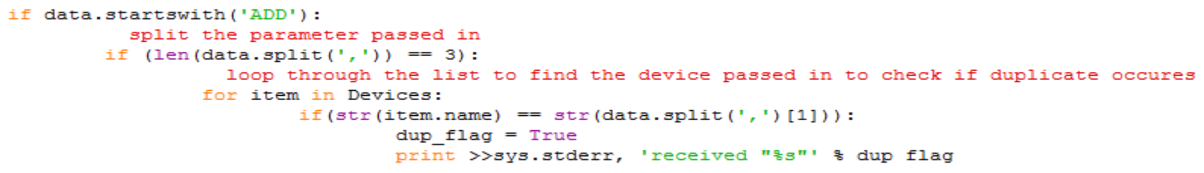
IOE Server Flow-Chart

the data received from the client is received in the amount of bytes provided in the parameters (In our case, we used 1028 bytes), the data is then stripped to remove any spaces leading or ending the data by calling data.strip()



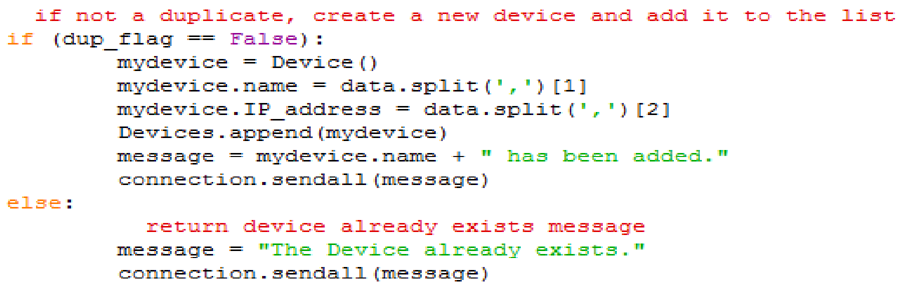
```
# Receive the data in large chunks and strip it
    data = connection.recv(1028)
    data = data.strip()
    print >>sys.stderr, 'received "%s"' % data
```

If the data startswith "ADD" then first loop through the "Devices" list to check if the parameters passed in is a duplicate or not.



```
if data.startswith('ADD'):
          split the parameter passed in
        if (len(data.split(',')) == 3):
                  loop through the list to find the device passed in to check if duplicate occures
                for item in Devices:
                        if(str(item.name) == str(data.split(',')[1])):
                                dup_flag = True
                                print >>sys.stderr, 'received "%s"' % dup flag
```

If not duplicate then create a new Device object "mydevice" and assign attributes to the parameters then append to the list. Finally, **send a sucssesful message back to the client using connection.sendall(message).**



```
  if not a duplicate, create a new device and add it to the list
if (dup_flag == False):
        mydevice = Device()
        mydevice.name = data.split(',')[1]
        mydevice.IP_address = data.split(',')[2]
        Devices.append(mydevice)
        message = mydevice.name + " has been added."
        connection.sendall(message)
else:
          return device already exists message
        message = "The Device already exists."
        connection.sendall(message)
```

IOE Server Flow-Chart

If the data startswith “Remove” then;

loop through the “Devices” list to check if the device to be removed is in the list or not. If it is then call “.remove(name)” to remove the device from the list. If it not in the list, send feedback stating device is not in the list, using connection.sendall(message)

```
elif data.startswith ("REMOVE"):
    if (len(data.split(',')) ==2):
              loop through the list to find the device to be removed
            for item in Devices:
                if(str(item.name) == str(data.split(',')[1])):
                         if device is found, then remove the device from the list
                        Devices.remove(item)
                        message = data.split(',')[1] + " has been removed from the list of devices."
                        connection.sendall(message)
                        exist_flag = True
            if (exist_flag == False):
                      return device not in list message
                    message = "Device could not be found in the list"
                    connection.sendall(message)
```

If the data startswith “READ” then;

loop through the “Devices” list to check if the device to be read and send the message back to the client using “sendall” or if the device is not in the list then send back error feedback.

```
if (str(item.name) == str(data.split(',')[1])):
    message = str(item.name) + "'s read-value is " +  str(item.read_value) + ", the target-value is " + str(item.target_value)
    connection.sendall(message)
    exist_flag = True
```

If the data startswith “WRITE” then;

loop through the “Devices” list to check if the device to be write to and assign the attribute to the new value and send the message back to the client using “sendall” or if the device is not in the list then send back error feedback.

```
        item.target_value = data.split(',')[2]
        message = item.name + "'s new target_value is " +  item.target_value
        connection.sendall(message)
        exist_flag = True
 if (exist_flag == False):
          return device not found message
        message = "Device could not be found in the list"
        connection.sendall(message)
```

IOE Server Flow-Chart

If the data startswith “QUIT” then;

Quit is handled onn the client side, so the server just respondes with a feedback message to the client using “sendall(message)”

```
elif data.startswith ("QUIT"):
            send back connection closed message
        message = "Connection ended on request."
        connection.sendall(message)
```

Similar logic for data starts with “CONNECT”

If the data doesn’t start with the proper commands then;

Send back feedback message using “sendall”

```
else:
            send command unknown message
        message = "Please make sure you type the right command."
        connection.sendall(message)
```

If at anytime, an error occurs, then close the connection on the server side.

```
    finally:
# if any event to cause the while loop to stop then clean up the connection
        connection.close()
```