CHRIST
(DEEMED TO BE UNIVERSITY)
BENGALURU · INDIA

Christ University, Faculty of Engineering.
Department of Computer Science and Engineering.

Data Logger for Servers in IoT

By

M Shankar Ganapathy
1660478
IV Semester,2018

Performed at

Centre for Development of Advanced Computing

No 1, Old Madras Road, adjacent to Baiyappanahalli metro station, Bengaluru, Karnataka 560038, India.

Internship Period: 1/04/18 – 31/05/18

CONTENTS

Acknowledgement:

I would like to express my heartfelt gratitude to Mr. Bindhumadhava B.S., Senior Director, (RTS & IOT) Mr Hari Babu P, Joint Director, (RTS & IOT), CDAC, Bangalore for giving me a chance to explore and dive into this booming field in Technology used in abundance even in the years to come.

I would also like to express my gratitude towards my teacher Professor Balamurugan from Christ University for their kind cooperation and encouragement in taking new challenges in this upcoming field of Information Technology.
I would also like to mention the extra effort that Mr Aman Kale, RTS & IoT Group, in helping me understand this new field and debug some of the unique errors that I faced in my code.

I extend my gratitude to my fellow trainees for helping me in not only getting used to the environment but also for sharing their knowledge and helping me out whenever possible.

Abstract:

The explosively growing demand of Internet of Things (IoT) has rendered broad scale advancements in the fields across sensors, radio access, network, and hardware/software platforms for mass-market applications. In spite of the recent advancements, limited coverage and battery for persistent connections of IoT devices still remain a critical impediment to practical service applications. IoT Devices have a lot of implementations in a smart House/ room depending on their need.
Here my Internship deals with the introduction of IoT Devices to office buildings or closed rooms where one would have to rely on air-conditioning for the necessary ventilation.

Preface:

As a part of B.Tech Curriculum, we are needed to carry out projects in various fields pertaining to the subject and I have carried out this project in the field IOT.
The motive behind this project is to gain an insight into this booming field hence develop the necessary knowledge required to work in this field in future.

This project deals with the creation of a client-server model which can handle the data sent from the IoT Device in the form of TCP packets and in-turn give an analysed and presentable view of what that means to the user.

Chapter 1. Introduction

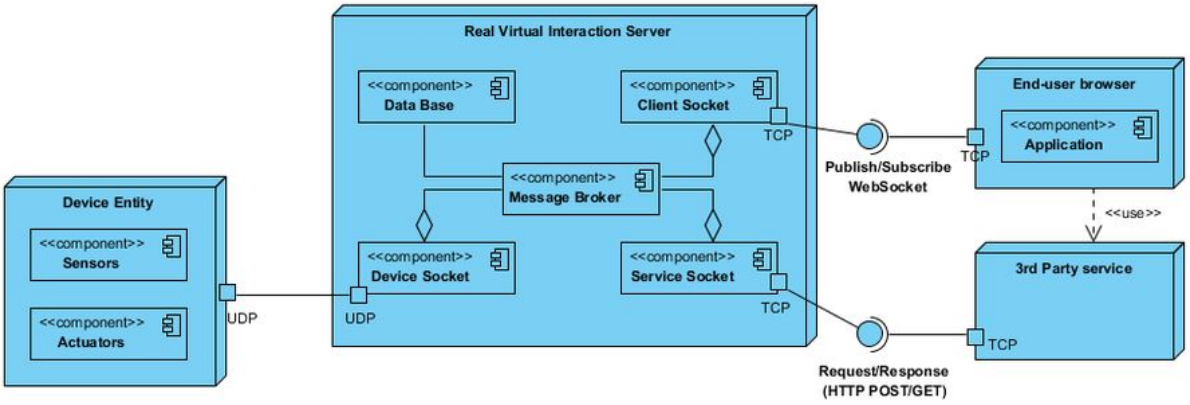
1.1 Background & Objectives

IOT is an ecosystem of connected physical objects that are accessible through the internet. There is an embedded technology in the objects helps them to interact with internal states or the external environment.
In offices or closed rooms in buildings, there is a need to continuously monitor the temperature, humidity or other factors that may play a role in how well people or employees work efficiently. Here the primary focus is on the occupants of a room, may they be people in homes, shops etc or employees at offices.
The data collected from the IoT Devices that have sensors would help in processing that data to make changes to the situation of that room at that moment.
Now IoT devices have a lot of sensors depending on use and hence generate huge amounts of data, which intern has to be handled by a server capable of storing and working on that data which may be used later.

1.2 Main Contribution

This project deals with a combination of hardware and software side. I work on the Software side which deals with the setting up of the server and necessary features required by the client like storing data to database and intimation through mail on stored data. While the Hardware side deals with the actual IoT device which transmits the data collected from sensors in JSON format.

1.2.1 Workflow of the Project



The project follows a similar workflow where data is received from a device entity with the help of its sensors and actuators. This data is then passed on to the Server where it would be stored in the database with the help of sockets.

Data is sent to and fro from the client to the server and at times to a 3rd party service depending upon need like a service that analyses and predicts some valuable information. Depending on whether or not the server gets valid data a general email and an error email is sent to the client to inform of the recent developments may it be an error or not.

Chapter 2.

2.1 Project Specifications

The project involved the use of varied languages ranging from Java to JSP to JavaScript and XML in different environments. Below I have mentioned the specification of the project.

2.1.1 Prerequisites

This project deals with the integration of many different types of files in an Eclipse IDE.

a. Understanding of Client-Server Model
b. Understanding of Servlets.
c. Implementation of TCP & HTTP protocols in java
d. Basic knowledge of MySQL.

2.1.2 Responsibilities

My responsibility in this project is to create a multi threaded server capable of handling multiple requests while it processes and stores the data that it receives from multiple, similar IoT devices.

Once the data received is processed some calculation have to be made on it depending on the device ID and the respective user has to be informed by mail about the recent developments or problems they would face if any.

All data may it be useful or constraint violated data would be stored in a database which when required is accessed by the server.

2.2. Learning Outcomes

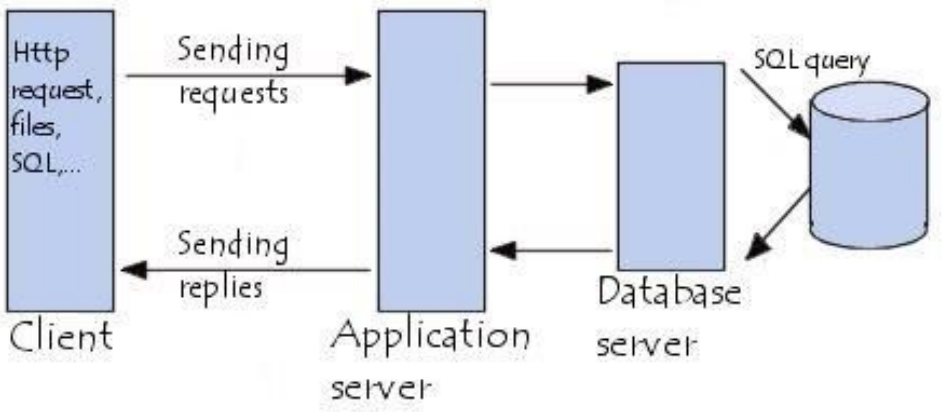
Client-Server Model

The Client-Server architecture used here is a 3-Tier Client-Server Architecture where the client (through the browser) requests the resources from a server.

There is an intermediate application server which again requests resources from a data server.

The application server is the one that deals with the actual Client using an interface, while the data server is the one that processes and stores the required data in the database for later usage.

The application server is multithreaded to be able to access multiple client-requests and works with the help of servlets because of the usage of the Dynamic web page. There is also an XML file that deals with the mapping of the servlet with their respective HTML and JavaScript.



Implementation of TCP/IP Sockets

Most Web applications use sockets to implement network communication protocols to transfer required data to and from the server. TCP is a connection-oriented protocol; on the other hand, a connection must be established before communications between the pair of sockets start. TCP is a reliable protocol it is guaranteed that the packets you send will be received in the order in which they were sent. This can be implemented using the "Socket" class which has parameters IP address and Port number. There should be two sockets to establish a connection a client and server socket along with an input and output stream for the transfer of data to and from the server.

2.2.3. Database and Mail Connectivity

In this project there is a need to save all the data received from the IoT Devices hence I have used MySQL to store this data into five tables

- Data table: This table stores acceptable data values for processing.
- Error table:This table stores the data that cannot be used because of constraint violation.
- FromMail table: to store the username,password, port and host through which the mails would be sent.
- ToMail table:This table stores the recipient's Name and email address.
- ThreshHoldData table:This table stores the threshold and critical values for the respective IoT devices.

To access these tables from the database I have used "mysql-connector.jar" corresponding to the JDK I am using.

To send and receive mails we have to import some special classes which are available from "activation.jar" and "mail.jar"files. Depending on the host i have configured the authentication process to send email to users from one standard email address. Later to send an email I have retrieved the recipient's information from the database and used it whenever necessary.

To send emails four classes have to be used

- Properties:This deals with the authentication of the sender's email address with the password.
- Session: This takes care of opening a dialog/session for the email to be sent.

- MimeMessage: which takes care of the To,From, BCC, CC, Subject and Message in the email.
- Transport: This takes care of the sending of an email.

Chapter 3. Code

Config.java

```
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;
import javax.servlet.annotation.WebListener;

@WebListener     // annotation to register a class as a listener for ServletContextListener's events
public class Config implements ServletContextListener
  { public void contextInitialized(ServletContextEvent arg0) {  // Overloads class that  tells the compiler
                                                                //   what to do at the servlet initialization
     System.out.println("In Listener");
     TCPServer tcp=new TCPServer();
     tcp.tcpReadWrite();                  // accessing the tcpReadWrite method to START server
     }

    public void contextDestroyed(ServletContextEvent arg0)   // Overloads class that  tells the compiler
    {                                                        //  what to do before servlet is Destroyed
      System.out.println("In contextDestroyed");
      TCPServer.stopServer();                    // accessing method to STOP server
      }

} // class Block
```

TCPServer.java

```
public class TCPServer {

    private static ServerSocket welcomeSocket = null;
    private static boolean connection=true;
    protected ExecutorService threadPool = Executors.newFixedThreadPool(100);//created a fixed thread pool of size 100 for
                                                                              //handling data from different WiFi nodes

    public  void tcpReadWrite()
    {
        new Thread()
        {
            @Override
                public void run() {
                try {
                        welcomeSocket = new ServerSocket(10000);   // Creating a socket with port 10000
                    } catch (IOException e1) {
                        e1.printStackTrace();
                    }
                    System.out.println("Started");
                     Socket connectionSocket = null;
                    while (connection) {
                        try {
                        connectionSocket = welcomeSocket.accept();   // accepting the socket Connection
                        } catch (IOException e) {
                            e.printStackTrace();
                        }
                    threadPool.execute(new ServerThread(connectionSocket)); // executing the threads in thread pool
                }
                    threadPool.shutdown();   // shutting down threadpool after completion
            }
        }.start();
    }
```

```
    public static void stopServer()         // method to STOP Server before servlet is destroyed
    {
        if(welcomeSocket!=null)
        {
            connection=false;
            try {
                welcomeSocket.close();      // closing the socket used by the server.
            } catch (Exception e) {
                e.printStackTrace();}
          }
       }
     }

//Class which handles packets from WiFi node
class ServerThread extends Thread
{   Socket s=null;
    byte[] bytedata=new byte[100];
    String data="";
    int temp,hum,id;

    public ServerThread(Socket s) {
        super();
        this.s = s;
        }

    public void run()
    { try
        {  InputStream in=s.getInputStream(); // connecting to the socket's input stream
                  int i=0;
                  while(i>-1){              // Loop used to convert data from input stram into
                      i=in.read();          //  valuable readable data.
                      if(i!=-1)
                          data=data+(char)i;
                  }


            JSONObject obj=new JSONObject(data); // class called to access the data sent in .JSON format
                  try {
                      temp=obj.getInt("Temp");
                      hum=obj.getInt("HUM");
                      id= obj.getInt("id");}
                  catch (JSONException e)
                    { e.printStackTrace(); }
                                          // Displaying the input .JSON Data in console
                  System.out.println(" DATA--->"+data +"\n IP "+(s.getInetAddress()).getHostAddress());

                  try {                       // input data is sent to be stored in DB
                      SqlSaveDatabase.thresholdCheck=true;
                      SqlSaveDatabase.dataStore(temp,hum,id);
                  } catch (Exception e)
                      { e.printStackTrace(); }

        } catch (Exception e1)
           { e1.printStackTrace(); }

         finally {
                if (s != null) {
                    try
                    { s.close(); }
                    catch (IOException e)
                     { // log error just in case
                        }
                     }// if block
                } //finally block
            } // run method block

    final protected  char[] hexArray = "0123456789ABCDEF".toCharArray();

} // ServerThread class block
```

SqlSaveData.java

```
public class SqlSaveDatabase {
static int port;
static String host, mname, mpass;
static boolean thresholdCheck = false;
static String nameuser,toemail;

public static void dataStore(int temp, int hum, int id) throws Exception
{  Class.forName("com.mysql.jdbc.Driver");
   String url = "jdbc:mysql://localhost:3306/aliens"; //aliens => database name on localhost
   String uname = "root";                             // Database username
   String pass = "cdac123";                           // Database password
   Connection con = DriverManager.getConnection(url, uname, pass); // connecting to Database
   Statement st = con.createStatement();
   String query = "Select * from thresholddata";
   ResultSet res = st.executeQuery(query);   // executing sql statement and storing the result
   if (thresholdCheck == true)               // Checking with Threshold & Critical values
    { while (res.next())
       {                                       // Retreiving data from database
        int tempthresh = res.getInt("Temp Threshold");
        int tempcrit = res.getInt("Temp Critical");
        int humthresh = res.getInt("Hum Threshold");
        int humcrit = res.getInt("Hum Critical");
        if (id == res.getInt("Device Id"))
         { if (temp >= tempthresh && temp <= tempcrit && hum >= humthresh && hum <= humcrit) {
            String qry = "insert into data(ID,Temp,hum) values("+ id + "," + temp + "," + hum + ");" ;
            st.execute(qry);
            System.out.println("Entry to Database Done\n");

            // Retreiving data for sending mail
            String qry1 = "Select * from frommail"; // frommail => table in DB
            ResultSet rs = st.executeQuery(qry1);
            System.out.println("Got from mail address from DB");
```

```
        // Storing data in local varibles
          while (rs.next()) {
            port = rs.getInt("port");
            host = rs.getString("host");
            mname = rs.getString("username");
            mpass = rs.getString("password"); }

        // Sending General Email
            String qry2 = "Select * from tomail";
            ResultSet rs3 = st.executeQuery(qry2);

        // Storing in Local Variable
            while (rs3.next()) {
             System.out.println("\n Displaying Recievers details \n");
             nameuser = rs3.getString("name");
             toemail = rs3.getString("email");
             GenEmailReply.sendEmail(id, temp, tempthresh,tempcrit, hum, humthresh, humcrit);
            }
       }

     else // Data is past Threshold and Critical values
      { System.out.println("Error:  Recieved Data not above Threshold\n");
         String qry = "insert into errordata(ID,Temp,hum) values("+ id + "," + temp + "," + hum + ");";
         st.execute(qry);
         String qry1 = "Select * from frommail";
         ResultSet rs = st.executeQuery(qry1);
         System.out.println("Got from mail address from DB");

         // Storing data in local varibles
         while (rs.next()) {
             port = rs.getInt("port");
             host = rs.getString("host");
             mname = rs.getString("username");
             mpass = rs.getString("password"); }



    // code for "To" address retrieval from DB
        String qry2 = "Select * from tomail";
        ResultSet rs1 = st.executeQuery(qry2);

    // Storing in Local Variable
     while (rs1.next()) {
          System.out.println("\n Displaying Recievers details \n");
          nameuser = rs1.getString("name");
          toemail = rs1.getString("email");
          ErrorEmailReply.sendEmail(id, temp, tempthresh,tempcrit, hum, humthresh, humcrit);
                }
            }// if Block
        } // if Block  (same device id)
        else
        { System.out.println("\n Device Threshold & Critical Value not in Database"); }
      }
    } st.close();
     con.close();
  }
}
```

ErrorEmailReply.java

```
public class ErrorEmailReply
 {  static int i=1;
    public static void sendEmail(int id, int temp, int tempthresh, int tempcrit, int hum, int humthresh, int humcrit) throws Exception
    { Properties prop= new Properties();
       prop.put("mail.smtp.auth", "true");
       prop.put("mail.smtp.starttls.enable", "true");
       prop.put("mail.smtp.host",SqlSaveDatabase.host);
       prop.put("mail.smtp.port",SqlSaveDatabase.port);

       // Password Authentication is done here
       Session session = Session.getInstance(prop, new javax.mail.Authenticator() {
            protected PasswordAuthentication getPasswordAuthentication()
             {return new PasswordAuthentication(SqlSaveDatabase.mname, SqlSaveDatabase.mpass); }
        });

     MimeMessage message = new MimeMessage(session);

      // Here Below the data to be sent in email is executed
        message.setFrom(new InternetAddress(SqlSaveDatabase.mname));
        message.addRecipients(Message.RecipientType.TO,SqlSaveDatabase.toemail );
        message.setSubject("ALERT from IOT Device");
        message.setText("Dear "+SqlSaveDatabase.nameuser
                +",\n The data is past Threshold Limit "
                +"\n Device ID: "+id
                +"\n Current Temperature: "+temp
                +"\n Current Humidity: "+hum
                +"\n Allowed Temp Threshold Range: "+tempthresh+" -- "+tempcrit
                +"\n Allowed Hum  Threshold Range: "+humthresh+" -- "+humcrit);
        Transport.send(message);
       System.out.println("Done, Sent Error Email to Recepient "+ i++ +".\n");
    }
}// class
```

GenEmailReply.java

```
public class GenEmailReply
{  // Email to confirm Data saved to DB
    static int i=1;
    public static void sendEmail(int id, int temp, int tempthresh, int tempcrit, int hum, int humthresh, int humcrit) throws Exception
    { Properties prop= new Properties();
       prop.put("mail.smtp.auth", "true");
       prop.put("mail.smtp.starttls.enable", "true");
       prop.put("mail.smtp.host",SqlSaveDatabase.host);
       prop.put("mail.smtp.port",SqlSaveDatabase.port);

    // Password Authentication is done here
       Session session = Session.getInstance(prop, new javax.mail.Authenticator() {
            protected PasswordAuthentication getPasswordAuthentication()
            { return new PasswordAuthentication(SqlSaveDatabase.mname, SqlSaveDatabase.mpass); }
        });

      MimeMessage message = new MimeMessage(session);

    // Here Below the data to be sent in email is executed
        message.setFrom(new InternetAddress(SqlSaveDatabase.mname));
        message.addRecipients(Message.RecipientType.TO,SqlSaveDatabase.toemail );
        message.setSubject("ALERT from IOT Device");
        message.setText("Dear "+SqlSaveDatabase.nameuser
                +",\n The data is Successfully Stored in DB "
                +"\n Device ID: "+id
                +"\n Current Temperature: "+temp
                +"\n Current Humidity: "+hum
                +"\n Allowed Temp Threshold Range: "+tempthresh+" -- "+tempcrit
                +"\n Allowed Hum  Threshold Range: "+humthresh+" -- "+humcrit);
        Transport.send(message);
        System.out.println("Done, Sent General Email to Recepient "+ i++ +".\n");
    }
 } // class
```

Chapter 3. Conclusion

I am glad to have been given a chance to do this project in the server side of IoT as it was something that I could not completely understand earlier. Hence this project has given me an understanding of what the world of IoT devices can and how it make life easy for everyone. This is the first project where I learnt the meaning of implementing after reading up on the theory of a new technology, so this has been a great experience and an insight as to what it would mean to work in the field of IoT.