



GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN



Georg-August University Göttingen
Faculty of Mathematics and Computer Science
Telematics Group
Sensor Lab

Final Project: Summer 2014

**Food Warehouse
Environment Monitoring
with Wireless Sensors
Network using Smartphone**

Lab Advisor: Dr. Omar Alfandi
Lab Assistant: Arne Bochem

Submitted by:
Hari Raghavendar Rao Bandari

Roll No: 11334055

Enrolled for:
M.Sc. Applied Computer Science

Abstract:

The aim of this project is to develop a mobile application, using Android Plat-form, to display in a mobile device the values taken by a Wireless Sensor Network (WSN). So Objective of this project is to monitor the “Food warehouse environments” like temperature, humidity pressure and visible Light with wireless Sensor Networks using smartphone devices.

Acknowledgments:

First of all, I would like to thank Dr. Omar Alfandi for his supervision in this project. Secondly, Arne was really helpful throughout this course, helped during the labs and also responded to our emails very fast.

Contents

Abstract	i
Acknowledgements	ii
1. Introduction	
1 1.1 Food warehouse Environment and mobile Industry	1
1 1.2 Sensor Network and TinyOS	2
1 1.3 Web Service.....	2
11.4 Project Contribution.....	3
2. Architecture	
2.1 N-Tier Architecture.....	4
2.2 Integrated Development Environment (IDE).....	5
2.2.1 Eclipse IDE.....	5
2.3 Software Development Kit (SDK).....	5
2.4 Mobile Architecture.....	6
2.5 Mobile Development.....	6
2.5.1 Android Development.....	6
2.5.1.1 Activity Life Cycle.....	6
2.5.1.2 Screen Sizes in Android	7
2.5.1.3 Different Platform Version.....	7
2.6 Web Services.....	8
2.7 Tiny OS Architecture.....	8
3. Requirements Analysis	9
4. Sensor Network & TinyOS	13
5. Database Design	16
6. Web Services	17.
7. Android Application	19
8. Conclusion	26
8.1 Summary.....	26
8.2 Future work.....	27
Bibliography	

Chapter 1:

Introduction

1.1 Food Warehouse Environment:

Food Warehouses are intended for the storage and physical protection of food products. In the context of Food storage, 'Food' primarily refers to all types of eating related Items. It may also include raw Milk and equipment required for the packaging and handling it and Meat safety control although, in an ideal situation, such items should be stored separately in warehouse. So here actual control will take place certain items should be placed in certain temperature, humidity and light. It is very important to store food products under certain temperature for example if Meat products strictly should store in cold places like huge Refrigerators if the temperature is low then Meat Products will spoil and as well as Milk products even other eating products like (Chesses, Butter, Yogurt and Eggs).

As our technology getting changed with the modern living and usage of smart phones has increased.

Usage of Smartphones:-

The use of smartphone devices over the past years seems to follow a growing trend. In (2014) 4.55 billion People worldwide are using smartphones. The total number of mobile users browsing Internet will be more than that of the desktop. The number of new subscriptions to Android, iOS systems and Windows systems, which at the moment lead the smartphone market share. Recently Windows Systems announced open source for smartphones and tablets.

Mobile phones are expected to dominate overall device shipments, with 1.9 billion mobile phones shipped in 2014, a five percent increase from 2013. Ultra mobiles, which include tablets, hybrids and clamshells, will take over as the main driver of growth in the devices market from 2014, with a growth rate of 54 percent.

In the OS market, Android continues to be the OS of choice across all devices. Gartner estimates that Android will reach 1.1 billion users in 2014, a 26 percent increase from 2013. "There is no doubt that there is a volume versus value equation, with Android

users also purchasing lower-cost devices compared to Apple users. Android holds the largest number of installed-base devices, with 1.9 billion in use in 2014, compared with 682 million iOS/Mac OS installed-base devices," said Annette Zimmerman, principal analyst at Gartner.

1.2 Sensor Network and TinyOS

A wireless sensor network (WSN) consists of spatially distributed autonomous sensors that monitor physical or environmental conditions, such as temperature, sound, pressure and while they cooperatively pass their data wirelessly through the network to the base station. The development of wireless sensor networks was motivated by military applications such as battlefield surveillance. Today such networks are used in many industrial and consumer applications, such as in the industrial process of monitoring and control, machine health monitoring.

The sensor nodes used in this application use TinyOS as their operating system. TinyOS is a free open source, BSD-licensed, component-based operating system, designed for low-power wireless devices. TinyOS is written in nesC (network embedded systems C) programming language. There is a worldwide community from academia and industry that uses, develops and supports TinyOS and its associated tools. This operating system is less common in the embedded world of sensing and control. In the area of the embedded systems, applications are usually bound to specific hardware. This is preferred due to the very limited hardware resources and the degree of specialization of the applications.

TinyOS was designed specifically for WSNs. It introduces a structured event-driven execution model and a component-based software design that enhances robustness, and minimizes power consumption to the minimum. The components that can be used by this system use well-defined interfaces to connect with each other. These interfaces resemble schematic wires that "glue" hardware components together.

1.3 Web Services

A Web service is a method of communication between two electronic devices over a network. It is a software function provided at a network address over the web with the service always on as in the concept of utility computing. For Example "JSON". Since, in our case, the Android device had to be able to connect to the sensor network, without any

proximity limitations to the area where the sensor network was installed, a web service had to be used. The primary concept of web services is that a client sends a request over HTTP to an address in which the web service is hosted and “listens to”. When the web service receives it, it sends a response back to the client with the requested data.

1.4 Project Contribution

The main object of this project was to find a way to assemble these three parts, the sensor network, the android application and the web service, in such manner that will let users use a sensor network without any prior knowledge in food warehouse environment monitoring fields. In order to achieve goal, it was important to make sure that final user would not have to worry about technical issues since that user could be virtually anyone. Any user just wants to keep track of the temperature, humidity, pressure and light levels of certain area in warehouse. It will be easy to keep track of the environment changes.

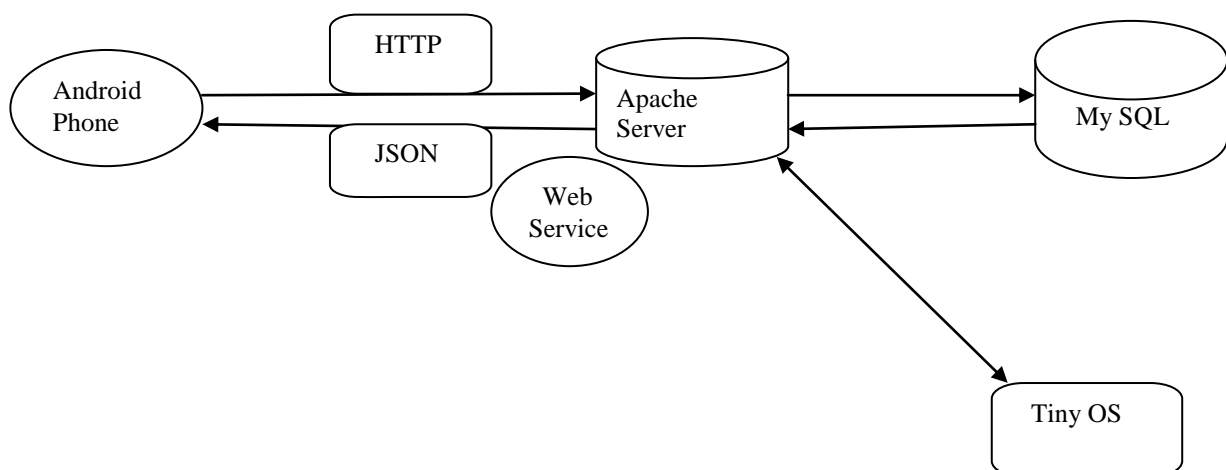
Chapter 2

2.1 N-Tier Architecture

There is the Android application, the web service, the sensor network and the database. The Android application behaves as a client. The user interacts with the client in order to use the sensor network. The client interacts with the web service to monitor the sensor network data. He should just be aware of the important data and the state of the system.

The web service on the other hand acts as the connecting link between the client and the database. It can be imagined as the middleman of the system. It exchanges messages with the client, the database and the sensor network in order to bind these components together. The web service should provide all the necessary information to the client (and indirectly to the user), while at the same time it ensures the security and the well-being of the system.

The sensing node send environment information to the Basestation in certain intervals and that data will be stored in the database. As soon as data is stored in database using PHP scripts we gather the last updated environment data and it will convert to JSON format and all time the PHP Page will be refreshed so that we get updated information. On other hand Android Application will interact with the PHP page through “HTTP POST” connection and acquire the JSON format which is recently updated environment data.



Finally, a database is used to store the received measurements of the sensor network. Apart from the measurements, the database keeps track of the environment data day to day with the time and date.

In order to implement the expected functionality, we had to create a 4-Tier system. The client in our case is a mobile application, run on an Android device, the TinyOS application clearly runs on the sensor nodes, while the web service and the database are hosted on the same machine.

2.2 Integrated Development Environments (IDE)

An Integrated Development Environment is a PC application that provides to the developer all the tools necessary for a successful software development. Usually it consists of:

- A source code editor
- A build automation tool
- A debugger

2.2.1 Eclipse IDE

Eclipse is also a well-known cross platform multi-language software development environment written mostly in Java. Additional functionality is provided by various modules that can be installed on top of it. One of these modules is the ADT plugin. This tool is designed to provide an environment suitable for the development of Android applications. ADT extends the capabilities of Eclipse allowing easiest creation of the application's UI and offering a variety of other equally important tools.

2.3 Software Development Kit (SDK)

While developing an application the programmer might need a set of tools and libraries in order to be able to use a certain resource of the system, or just to implement a function easier using code that already exists. This set of tools is called SDK. Usually SDK is just an application programming interface (API) which is a set of files that are used in order to get access to already implemented code. For example, when developing an android app a set of methods may have to be called or implemented (interface) in order for the application to run on the specific platform.

SDK may also include a set of tools that are used from the IDE to produce a more appropriate coding experience which is suitable for the corresponding platform. It usually, also includes

documentation files that provide information about specific functions and sample code for the developer to decrease its learning curve.

2.4 Mobile Architecture:

In the past computers needed to be disconnected from their internal network if they needed to be taken or moved anywhere. Mobile architecture allows maintaining this connection whilst during transit. Each day the number of mobile devices is increasing, mobile architecture is the pieces of technology needed to create a rich, connected user experience. Currently there is a lack of uniform interoperability plans and implementation. There is a lack of common industry view on architectural framework. This increases costs and slows down 3rd party mobile development. An open approach is required across all industries to achieve same end results and services.

2.5 Mobile Development:

Mobile application development is the process by which application software is developed for low-power handheld devices, such as personal digital assistants, enterprise digital assistants or mobile phones. These applications can be pre-installed on phones during manufacturing, downloaded by customers from various mobile software distribution platforms, or delivered as web applications using server-side or client-side processing (e.g. JavaScript) to provide an "application-like" experience within a Web browser. Application software developers also have to consider a lengthy array of screen sizes, hardware specifications and configurations because of intense competition in mobile software and changes within each of the platforms. Mobile app development has been steadily growing, both in terms of revenues and jobs created. A 2013 analyst report estimates there are 529,000 direct App Economy jobs within the EU 28 members, 60% of which are mobile app developers.

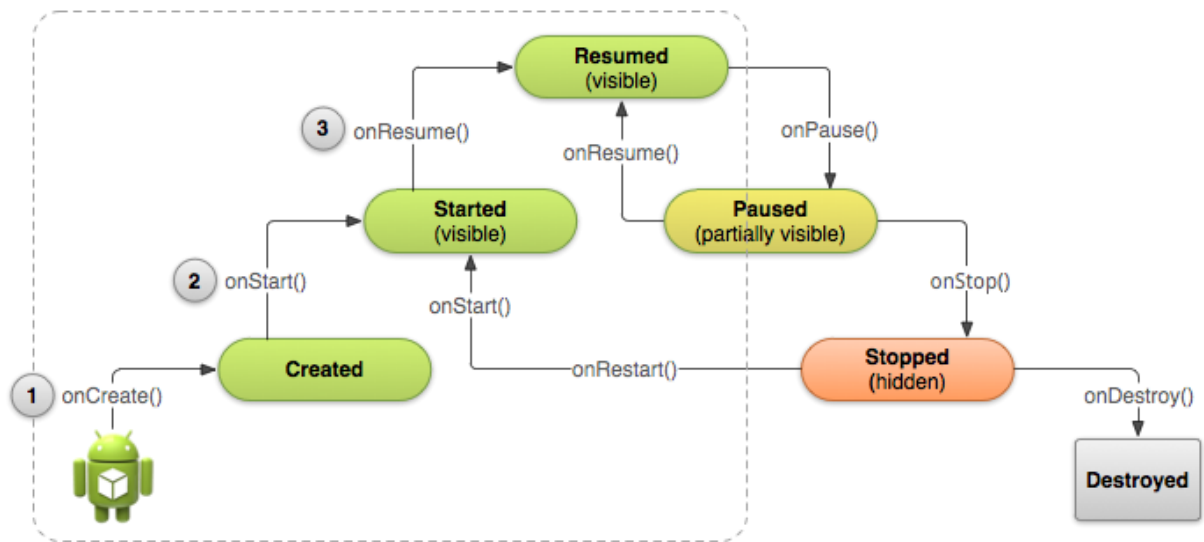
2.5.1 Android Development

Android uses Activities to show the UI of the applications. It is a single focused screen the user can interact with. The programmer has in his disposal some standard functions to manipulate the behavior of this screen during the various stages of the application. These various stages of an application are known as the Activity's Lifecycle. Activities are managed using an activity stack. When a new activity is started, it is placed on the top of the stack, which contains other

running activities. An activity remains below another newly added Activity until the later one exits. At that time the older Activity returns to the foreground.

2.5.1.1 Activity Lifecycle

- Resumed
- Paused
- Stopped



2.5.1.2 Screen Sizes in Android:

Special directories are provided for the necessary resources, which have to be properly scaled to the various density buckets. The appropriate scale for each density bucket is:

- ❖ xhdpi: 2.0
- ❖ hdpi: 1.5
- ❖ mdpi: 1.0 (baseline)
- ❖ ldpi: 0.75

2.5.1.3 Different Platform Versions

Android is constantly under development. This results in a broad range of changes on the various methods and functionalities from one version to the other. While from one perspective this is a positive feature, as it means that bugs and security issues are resolved, the changes made from one Android version to the other, may require specific code for a functionality to run properly, that depends on the version of Android currently 4.4 KitKat.

2.6 Web Services:

A Web service is a method of communication between two electronic devices over a network. It is a software function provided at a network address over the web with the service always on as in the concept of utility computing.

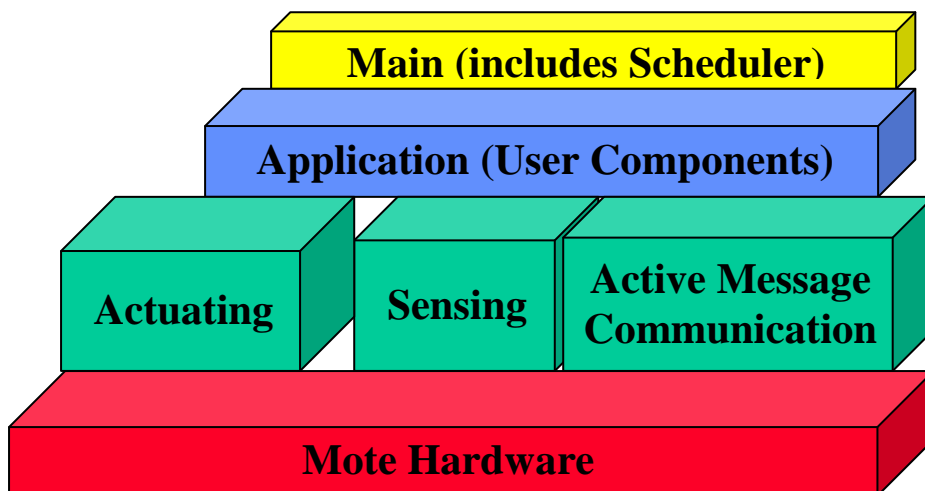
JSON-RPC is a remote procedure call protocol encoded in JSON. It is a very simple protocol (and very similar to XML-RPC), defining only a handful of data types and commands. JSON-RPC allows for notifications (data sent to the server that does not require a response) and for multiple calls to be sent to the server which may be answered out of order.

Example:

```
[{"Temperature": "28.62", "Pressure": "9872", "Humidity": "35.54", "VisibleLight": "1039"}, {"Temperature": "28.62", "Pressure": "9872", "Humidity": "35.54", "VisibleLight": "9"}]
```

2.7 TinyOS Architecture:

TinyOS is a free and open source software component-based operating system and platform targeting wireless sensor networks (WSNs). TinyOS is an embedded operating system written in the nesC programming language as a set of cooperating tasks and processes. It is intended to be incorporated into smart dust. TinyOS started as collaboration between the University of California, Berkeley in co-operation with Intel Research and Crossbow Technology, and has since grown to be an international consortium, the TinyOS Alliance.



Chapter 3

3. Requirements Analysis:

TinyOS is a free open source operating system. It is written in nesC programming language, which is a dialect of C optimized for the low memory usage of sensor devices. TinyOS programs form components. A component can use and can be used by other components. Interfaces are used to enable the interconnection of the components. Certain interfaces are provided by TinyOS. These modules usually support basic operations such as packet communications, storage, timers and so forth. Additionally, the motes can interact with computers to print messages on the terminal or to send messages to the serial port by using the TinyOS API and implementing the appropriate methods.

IRIS Mote

The IRIS is a 2.4 GHz Mote used for enabling low-power wireless sensor networks. IRIS provides users with high-level functional integration designed to optimize the addition of wireless mesh networking technology to a wide variety of custom sensing applications providing up to three times improved ratio range and twice the program memory over previous generations of MICA Motes (“MEMSIC: Wireless Modules, IRIS, MicaZ, TelosB, Imote2, Cricket”)

Features Include:

- 2.4 GHz IEEE 802.15.4, tiny wireless measurement system
- Designed specifically for deeply embedded sensor networks
- 250 kbps, High data rate radio
- Wireless communications with every node as Router capability
- Expansion connector for Light, Temperature, RH, Barometric Pressure, Acceleration/Seismic, Acoustic, Magnetic etc. ("IRIS Wireless Measurement System")



Figure 2: IRIS Sensor (“Dignan, Larry. *IRIS Sensor*”.)

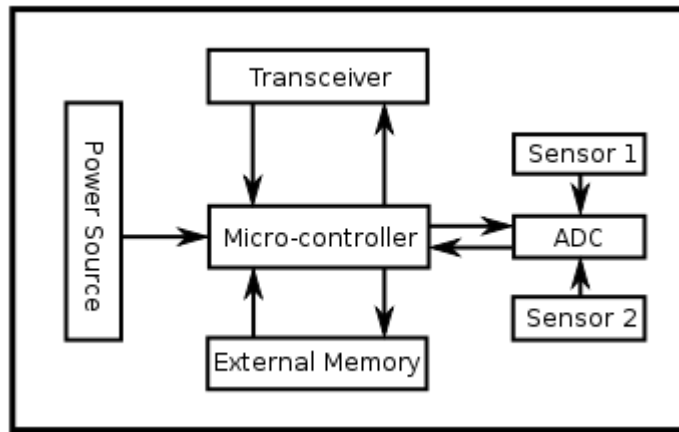


Figure 3: IRIS Sensor Mote Architecture (“Wikipedia: [Sensornode.svg](#).”)

MTS400 Sensor Motes

MTS400 offer five basic environmental sensing parameters and an optional GPS module (MTS420). These sensor boards utilize the latest generation of IC-based surface mount sensors. These energy-efficient digital devices in turn provide extended battery-life and performance wherever low maintenance field deployed sensor nodes are required. These versatile sensor boards are intended for a wide variety of applications ranging from a simple wireless weather station to a full mesh network of environmental monitoring nodes. Applicable industries include Agricultural, Industrial, Forestry, HVAC and more. They can be applied in Agricultural monitoring, Art preservation, Environmental monitoring, Sensor location mapping (“*MTS-MDA Series Users Manual*”).

Features and Benefits:

- Compatible with IRIS/MICAz/MICA2 Processor/ Radio Boards
- Onboard Temperature & Humidity, Barometric Pressure and Ambient Light Sensors
- Dual-Axis Accelerometer
- 64K EEPROM (CC versions), 2K EEPROM (CB versions) for User Configuration Data
- Optional GPS Module (MTS420 Only) (“*MTS-MDA Series Users Manual*”).



Figure 4: MTS400 (“MTS400CC Sensor Board. Digital image”)

Humidity and Temperature Sensor

The Sensirion® (<http://www.sensirion.com/>) SHT11 is a single-chip humidity and temperature multi sensor module comprising a calibrated digital output. The chip has an internal 14-bit analog-to-digital converter and serial interface. SHT11s are individually calibrated. This sensor’s power is enabled through a programmable switch. The control interface signals are also enabled through a programmable switch. An analog-to-digital converter in the sensor does the conversion from humidity and temperature to digital units (“*MTS-MDA Series Users Manual*”).

2-Axis Accelerometer

The accelerometer is a MEMS surface micro-machined 2-axis, ± 2 g device. It features very low current draw ($< 1\text{mA}$). The sensor can be used for tilt detection, movement, vibration, and/or seismic measurement. The sensor outputs are connected to ADC channels on the Mote’s ADC1 and ADC2 channels (“*MTS-MDA Series Users Manual*”).

Light Sensor

The TLS2550 is a digital light sensor with a two-wire, SMBus serial interface. It is manufactured by TAOS, Inc (<http://www.taosinc.com>). It combines two photodiodes and a compounding analog-to-digital converter on a single CMOS integrated circuit to provide light measurements over an effective 12-bit dynamic range. This sensor’s power is enabled through a programmable switch. The control interface signals are also enabled through a programmable switch. An analog-to-digital converter in the sensor does the conversion from light to digital units (“*MTS-MDA Series Users Manual*”).

Turning Sensors On and Off

Power for all of the sensors on the MTS400/420 sensor board is controlled through an analog power switch at location U7. It can be programmed enable and disable power to individual sensors. The default condition for the sensors is off. This design helps minimize power draw by the sensor board (*“MTS-MDA Series Users Manual”*).

MIB520 USB Gateway

The MIB520CB provides USB connectivity to the IRIS and MICA family of Motes for communication and in-system programming. Any IRIS/MICAz/MICA2 node can function as a base station when mated to the MIB520CB USB interface board. In addition to data transfer, the MIB520CB also provides a USB programming interface. The MIB520CB offers two separate ports: one dedicated to in-system Mote programming and a second for data communication over USB. The MIB520CB has an on-board processor that programs Mote Processor Radio Boards. USB Bus power eliminates the need for an external power source.

Its features and benefits include Base Station for Wireless sensor Networks, USB Port Programming for IRIS/MICAz/MICA2 Hardware platforms, Supports JTAG code debugging, USB Bus power. It can be used for applications like USB Interface, Test bed Deployments and In-System Programming (*“MEMSIC, Inc - Gateways | MIB520”*).

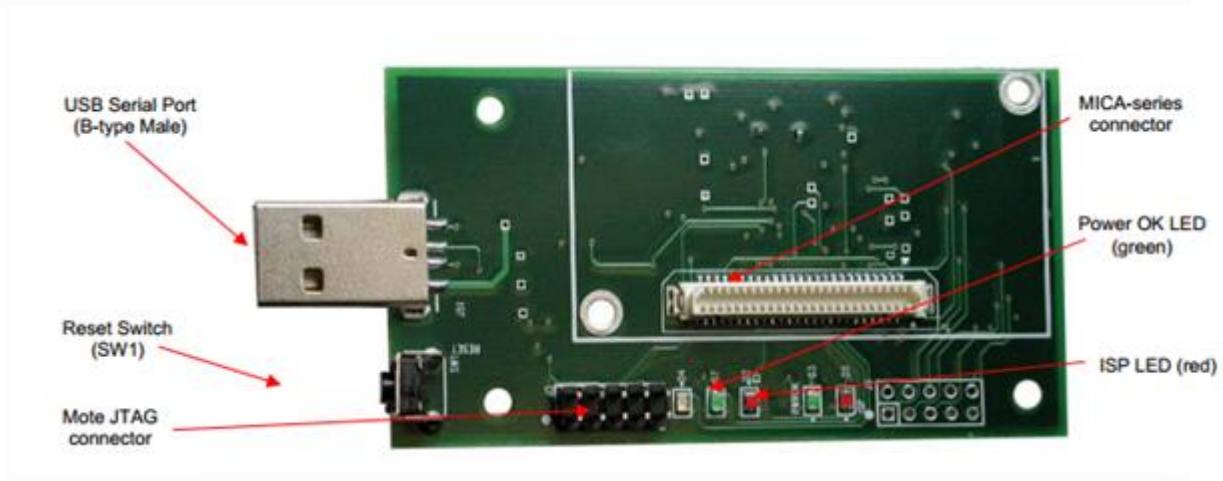


Figure 5: MIB520 USB Gateway (*“MIB520 Board. Digital image”*)

Chapter 4

4. Sensor Network and TinyOS:

TinyOS is a free open source operating system. It is written in nesC programming language, which is a dialect of C optimized for the low memory usage of sensor devices. TinyOS programs form components. A component can use and can be used by other components. Interfaces are used to enable the interconnection of the components.

Internal Programming

By complementation of the sensor node programs the network will be able to transmit the collected data to the available nodes transmit finally to the database. Here the main functionality for data sinking and transmitting the information to the database is implemented.

Base Station Software

Base station software is complex program compared to the individual mote program. Once the packets are received by sink motes which are attached by the USB interface, the node forwards the information java program which helps to store in the database. After receiving the packet it is discarded in order to avoid redundancy.

MySQL Data Base

When the data received by the base station is structured in proper way it is sent to the MYSQL database which can be located on any remote computer which can be accessed via internet. All the recorded data consists of the timestamp with the event and its type etc.

User Interface

After the data is successfully sent to the MYSQL database with the given user credentials the user can login to view the network data. If they are on same computer no internet is required but on remote then internet is required for the communication. We intend to develop android application by which we can visualize the environment information

Design

- MTS420 is used for the measurement of the details for the added advantage of GPS module which can be easily used by the user at any remote place for different kind of reports.
- The time period for the sending the values from the sensor nodes are 1 min.
- The mts400Tester.java communicates with the BaseStation via serial port while receiving the data from the connected sensors. The packets are processed and values

along with the respective time stamp. It converts all the unstructured data to the structured form

- The mts400Tester.java saves the file in the database server (MySQL).
- Later DataMsg.java extracts the data in to the local machine.
- Next Using PHP language will retrieve the data from the MYSQL and it will convert into JSON format.

Deployment

Step 1: Set up Base Station:-

Take the node keep on board go to the path where Base station files are present in the directory.

Then run the following commands

Ex: CD opt/tinyOS2.x/apps/project/Basestation

SENSORBOARD=mts400 make iris reinstall.1 mib520,/dev/ttyUSB0

Step 2: Set up the sensing node:

Now remove the Base station installed node from the board and keep the sensing node on it. Now we need to set the path where mts400 directory is present on the computer. Now we have to install the programs on sensing node by assigning node number (I'e: 4) respectively. Once program install remove node from board place the batteries in it and on the sensing node.

Ex: CD opt/tinyOS2.x/apps/project

SENSORBOARD=mts400 make iris reinstall.4 mib520,/dev/ttyUSB0

Step 3: Now Set up the Base station Again

Now we need to set up Base Station node on board then we have run below following commands in the same mts400 directory.

javac Mts400Tester.java

java Mts400Tester -comm serial@/dev/ttyUSB1:iris

Now see below diagram its show the data which is received from the sensing node to Base station and at same time it is storing the data into “MYSQL”

```
Terminal - h.bandari@ws3: ~/mts400
File Edit View Terminal Go Help

Node Address: 4
Food warehouse_Intersema temperature: 271
Food warehouse_Intersema pressure: 9862
Food warehouse_Sensirion temperature: 27.06
Food warehouse_Sensirion humidity: 45.91
Food warehouse_Taos visible light: 247
Food warehouse_Taos infrared light: 0
Successfully climate values inserted

The Food warehouse climate results are

Node Address: 4
Food warehouse_Intersema temperature: 271
Food warehouse_Intersema pressure: 9862
Food warehouse_Sensirion temperature: 27.05
Food warehouse_Sensirion humidity: 45.87
Food warehouse_Taos visible light: 247
Food warehouse_Taos infrared light: 0
Successfully climate values inserted
```

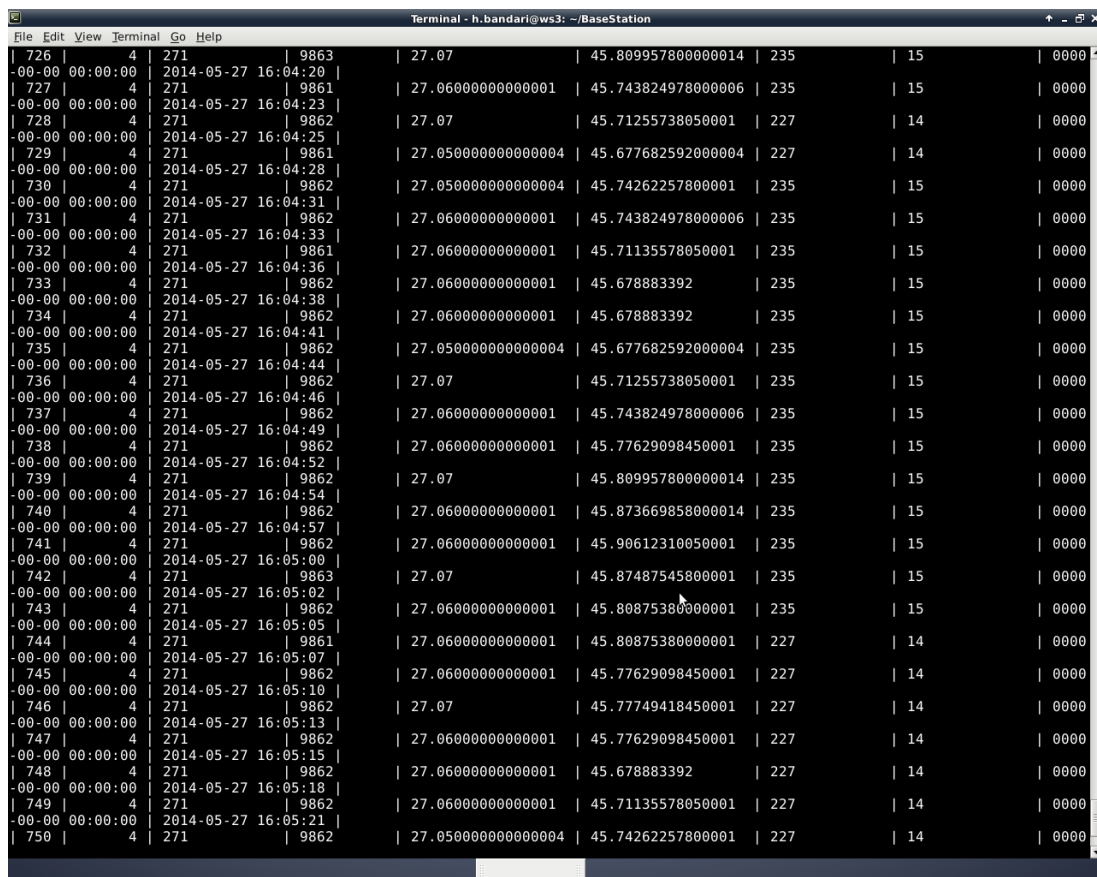
Chapter 5

Database Design

Creating MySQL table in the data base to store received data from the sensing node:

```
CREATE TABLE climate(  
  cid int(11) primary key auto_increment,  
  node_id int(11) not null,  
  temperature_i varchar(100) not null,  
  pressure_i varchar(100) not null,  
  temperature_s varchar(100) not null,  
  humidity_s varchar(100) not null,  
  visible_light varchar(100) not null,  
  infrared_light varchar(100) not null,  
  created_at timestamp default '0000-00-00 00:00:00',  
  updated_at timestamp DEFAULT now() ON UPDATE now());
```

Above query will create a “Climate” table in the Database. Below screenshot shows the results which are stored in Database.



726	4	271	9863	27.07	45.809957800000014	235	15	0000
00-00 00:00:00	2014-05-27 16:04:20							
727	4	271	9861	27.060000000000001	45.743824978000006	235	15	0000
00-00 00:00:00	2014-05-27 16:04:23							
728	4	271	9862	27.07	45.71255738050001	227	14	0000
00-00 00:00:00	2014-05-27 16:04:25							
729	4	271	9861	27.050000000000004	45.677682592000004	227	14	0000
00-00 00:00:00	2014-05-27 16:04:28							
730	4	271	9862	27.050000000000004	45.74262257800001	235	15	0000
00-00 00:00:00	2014-05-27 16:04:31							
731	4	271	9862	27.060000000000001	45.743824978000006	235	15	0000
00-00 00:00:00	2014-05-27 16:04:33							
732	4	271	9861	27.060000000000001	45.71135578050001	235	15	0000
00-00 00:00:00	2014-05-27 16:04:36							
733	4	271	9862	27.060000000000001	45.678883392	235	15	0000
00-00 00:00:00	2014-05-27 16:04:38							
734	4	271	9862	27.060000000000001	45.678883392	235	15	0000
00-00 00:00:00	2014-05-27 16:04:41							
735	4	271	9862	27.050000000000004	45.677682592000004	235	15	0000
00-00 00:00:00	2014-05-27 16:04:44							
736	4	271	9862	27.07	45.71255738050001	235	15	0000
00-00 00:00:00	2014-05-27 16:04:46							
737	4	271	9862	27.060000000000001	45.743824978000006	235	15	0000
00-00 00:00:00	2014-05-27 16:04:49							
738	4	271	9862	27.060000000000001	45.77629098450001	235	15	0000
00-00 00:00:00	2014-05-27 16:04:52							
739	4	271	9862	27.07	45.809957800000014	235	15	0000
00-00 00:00:00	2014-05-27 16:04:54							
740	4	271	9862	27.060000000000001	45.873669858000014	235	15	0000
00-00 00:00:00	2014-05-27 16:04:57							
741	4	271	9862	27.060000000000001	45.90612310050001	235	15	0000
00-00 00:00:00	2014-05-27 16:05:00							
742	4	271	9863	27.07	45.87487545800001	235	15	0000
00-00 00:00:00	2014-05-27 16:05:02							
743	4	271	9862	27.060000000000001	45.80875380000001	235	15	0000
00-00 00:00:00	2014-05-27 16:05:05							
744	4	271	9861	27.060000000000001	45.80875380000001	227	14	0000
00-00 00:00:00	2014-05-27 16:05:07							
745	4	271	9862	27.060000000000001	45.77629098450001	227	14	0000
00-00 00:00:00	2014-05-27 16:05:10							
746	4	271	9862	27.07	45.77749418450001	227	14	0000
00-00 00:00:00	2014-05-27 16:05:13							
747	4	271	9862	27.060000000000001	45.77629098450001	227	14	0000
00-00 00:00:00	2014-05-27 16:05:15							
748	4	271	9862	27.060000000000001	45.678883392	227	14	0000
00-00 00:00:00	2014-05-27 16:05:18							
749	4	271	9862	27.060000000000001	45.71135578050001	227	14	0000
00-00 00:00:00	2014-05-27 16:05:21							
750	4	271	9862	27.050000000000004	45.74262257800001	227	14	0000

Chapter 6

Web Server:

PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language. As of January 2013, PHP was installed on more than 240 million websites (39% of those sampled) and 2.1 million web servers. Originally created by Rasmus Lerdorf in 1994, the reference implementation of PHP is now produced by The PHP Group.[6] While PHP originally stood for Personal Home Page, it now stands for PHP: Hypertext Preprocessor, a recursive backronym.

Now to retrieve the data from the database I use PHP language to connect to the database and it will retrieve the data and at same time it will convert into JSON format also.

```
<?php
    header('Refresh: 10');
    $objConnect
mysql_connect("192.168.22.50","slsummer14_g3","tjXyVpopLsETvkNIiqdtomv2QC");

if (!$objConnect)
{
    die('Could not connect: ' . mysql_error());
}
    $objDB = mysql_select_db("slsummer14_g3");
    $strSQL = "SELECT Node_id,ROUND(temperature_s,2) AS Temperature,presure_i
AS Pressure,ROUND(humidity_s,2) AS Humidity,visible_light AS VisibleLight FROM
climate ORDER BY updated DESC LIMIT 1";
    $objQuery = mysql_query($strSQL);
    $intNumField = mysql_num_fields($objQuery);
    $resultArray = array();
    while($objResult = mysql_fetch_array($objQuery))
    {
        $arrCol = array();
        for($i=0;$i<$intNumField;$i++)
        {
```

```

        $arrCol[mysql_field_name($objQuery,$i)] = $objResult[$i];
    }
    array_push($resultArray,$arrCol);
}

```

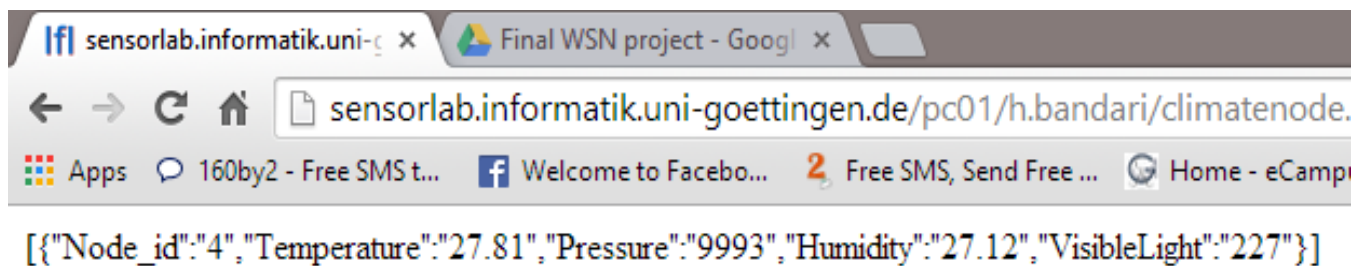
```
mysql_close($objConnect);
```

```
echo json_encode($resultArray);
```

?>

Output:

```
[{"node_id":"4","Temperature":"27.81","Pressure":"9993","Humidity":"27.12","VisibleLight":
"227"}]
```



Chapter 7

Android Application:

7.1 Introduction

In this chapter, we describe the implementation of the Android application. This app is what the end user utilizes to interact with our system. This application should provide easy-to-use functionality, as the end-user could be virtually anyone. Additionally, a certain level of abstraction had to be implemented, because users without any prior knowledge of sensor networks, databases or web services should be able to operate our system without any problems.

7.2 Mobile Limitations:

Since the targeted medium in our occasion is a mobile device, we should take into account several limitations that are bound together with these devices.

- ❖ Firstly, their processing power is significantly lower when compared to conventional computers. Therefore, it is preferable to execute complex calculations on the server side, and just let the client consume these results.
- ❖ Additionally, battery consumption is undoubtedly a major factor.
- ❖ Storage is another important issue on these devices.
- ❖ Furthermore, since our target devices are mobile phones with limited screen size it is significant to take into account that limited screen real estate.

7.3 Code

MainActivity.java

```
package com.climate;
import java.io.IOException;
import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.params.BasicHttpParams;
import org.apache.http.params.HttpConnectionParams;
```

```
import org.apache.http.params.HttpParams;
import org.apache.http.util.EntityUtils;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
```

```
import android.os.Bundle;
import android.os.StrictMode;
import android.app.Activity;
//import android.content.Intent;
import android.view.Menu;
//import android.view.MenuItem;
import android.widget.TextView;
```

```
public class MainActivity extends Activity {
```

```
    @Override
```

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
        StrictMode.ThreadPolicy policy = new StrictMode.
            ThreadPolicy.Builder().permitAll().build();
        StrictMode.setThreadPolicy(policy);
```

```
        TextView temp = (TextView) findViewById(R.id.temp);
        TextView press = (TextView) findViewById(R.id.press);
        TextView humi = (TextView) findViewById(R.id.humi);
        TextView visi = (TextView) findViewById(R.id.visi);
```

```
        JSONObject json = null;
        String str = "";
        HttpResponse response;
        //Create the HTTP request
        HttpParams httpParameters = new BasicHttpParams();
        //Setup timeouts
        HttpConnectionParams.setConnectionTimeout(httpParameters, 8000);
```



```

        HttpURLConnectionParams.setSoTimeout(httpParameters, 8000);
        HttpClient myClient = new DefaultHttpClient(httpParameters);
        HttpPost myConnection = new
HttpPost("http://192.168.1.7/android_connect/climatenode.php");

        try {
            response = myClient.execute(myConnection);
            str = EntityUtils.toString(response.getEntity(), "UTF-8");

            } catch (ClientProtocolException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        try{
            JSONArray jArray = new JSONArray(str);
            json = jArray.getJSONObject(0);

            temp.setText(json.getString("Temperature"));
            press.setText(json.getString("Pressure"));
            humi.setText(json.getString("Humidity"));
            visi.setText(json.getString("VisibleLight"));

            } catch ( JSONException e) {
                e.printStackTrace();
            }
        }

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
            // Inflate the menu; this adds items to the action bar if it is present.
            getMenuInflater().inflate(R.menu.main, menu);
            return true;
        }

```

}

Activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:background="#E8E8E8"
    tools:context=".MainActivity">
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:text="@string/info"
    android:textSize="18sp"
    android:textStyle="bold" />
```

```
<TextView
    android:id="@+id/temp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="150dp"
    android:layout_marginTop="50dp" />
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="50dp"
    android:text="@string/temp"
    android:textStyle="bold" />
```

```
<TextView
    android:id="@+id/press"
    android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content"
android:layout_below="@+id/temp"
android:layout_marginLeft="150dp"
android:layout_marginTop="50dp" />
```

<TextView

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@+id/temp"
android:layout_marginTop="50dp"
android:text="@string/press"
android:textStyle="bold" />
```

<TextView

```
android:id="@+id/humi"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@+id/press"
android:layout_marginLeft="150dp"
android:layout_marginTop="50dp" />
```

<TextView

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@+id/press"
android:layout_marginTop="50dp"
android:text="@string/humi"
android:textStyle="bold" />
```

<TextView

```
android:id="@+id/visi"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@+id/humi"
android:layout_marginLeft="150dp"
android:layout_marginTop="50dp" />
```

<TextView

```
android:layout_width="wrap_content"
```

```

        android:layout_height="wrap_content"
        android:layout_below="@+id/humi"
        android:layout_marginTop="50dp"
        android:text="@string/visi"
        android:textStyle="bold" />
</RelativeLayout>

```

Main.xml

```

<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="com.example.weather.MainActivity" >

    <item
        android:id="@+id/action_settings"
        android:orderInCategory="100"
        android:title="@string/action_settings"
        app:showAsAction="never"/>

</menu>

```

Strings.xml

```

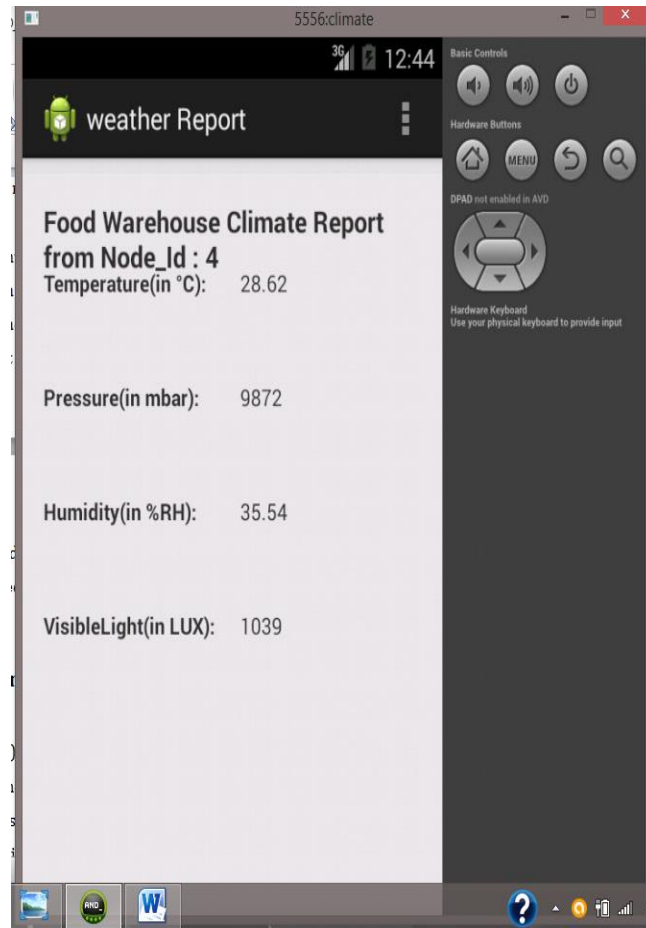
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">weather Report</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
    <string name="info">Food Warehouse Climate Report from Node_Id : 4</string>
    <string name="temp">Temperature(in °C):</string>
    <string name="press">Pressure(in mbar):</string>
    <string name="humi">Humidity(in %RH):</string>
    <string name="visi">VisibleLight(in LUX):</string>

</resources>

```

Output:



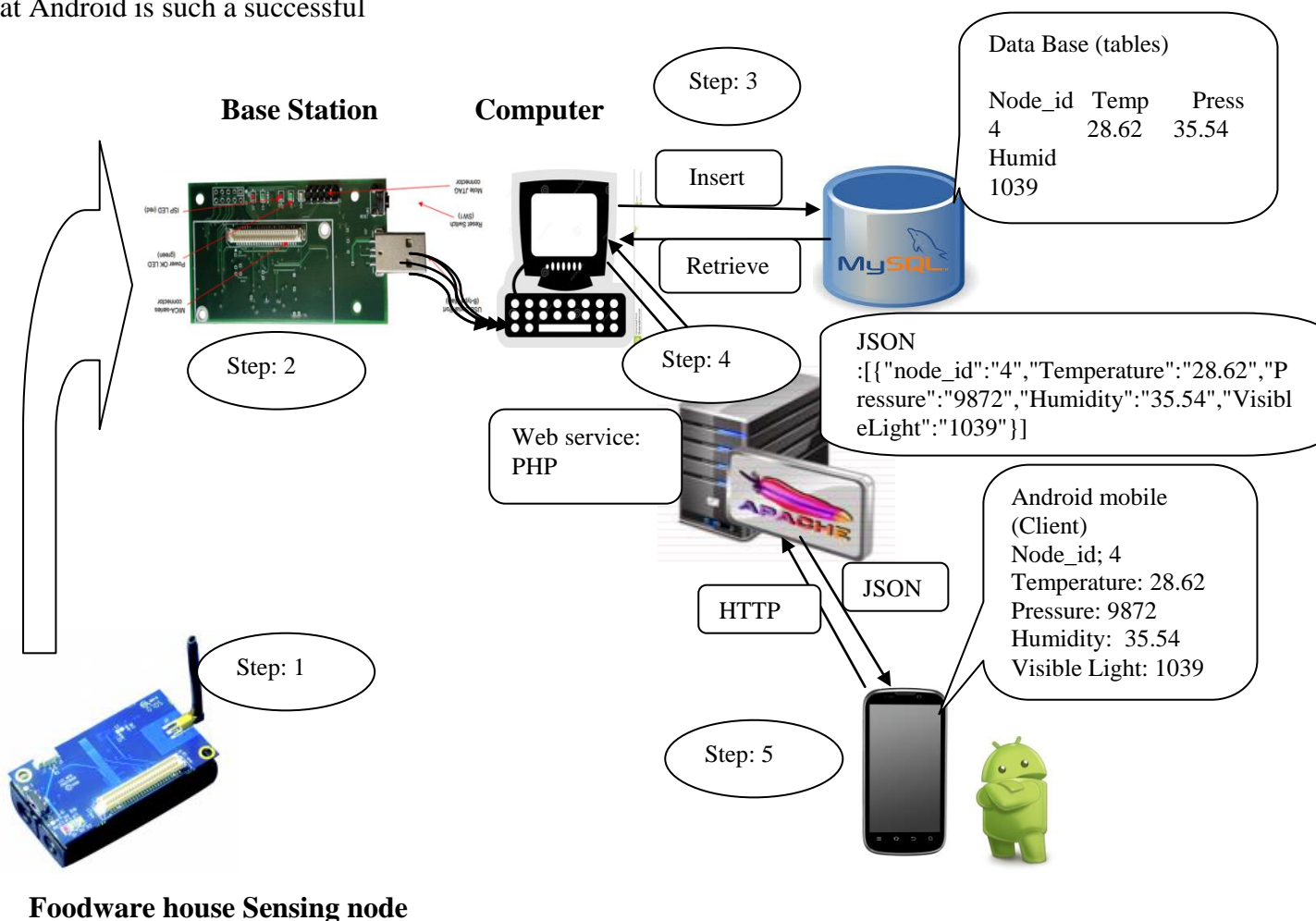
Chapter 8

Conclusion

8.1 Summary

In this project implementation, we created a complete system that enables the manipulation of a sensor network using a mobile application. The user of this system is able to execute operations such as simple sensor measurement retrievals, without having to deal with software that requires advanced programming skills to run. Additionally, while several applications have been developed in the past, enabling the direct manipulation of a sensor network, neither of them was free from certain limitations..

These limitations, certainly do not apply to our system. Firstly, the client program used by the end user to enable him interact with our system is an Android application. This application was developed while keeping in mind that it had to run flawlessly both on older and newer versions of Android. Additionally, since Android is a very popular operating system with millions of activations per day, it is rather self-explanatory that the target audience is particularly big in number, so there should not be any real limitations there. It is worth mentioning that, the fact that Android is such a successful



Perhaps the most important advantage of this system is the mobility factor. The users can be virtually anywhere and still be able to use our system, provided that there is an internet connection. The fact that this system is using a web service, as an intermediate level between the Android application and the actual sensor network can verify the flexibility that comes with our system. The web service also guarantees the security of our system, since the client cannot not operate directly on the sensor network. That way several activities performed by potential hackers with intentions to cause problems to our system, are limited. Additionally, since this is a RESTful application, there is literally no limitations concerning the operating system the server uses. The web service is hosted on conventional HTTP servers, such as Tomcat which run both on Windows and Linux operating systems.

8.2 Future Work

In Foodware house we have many types of Items which we need to preserve certain items under certain temperature, humidity and light. So when ever the season changes then user will get the notification about the item with the expiry date and time of it.

Bibliography

1. Philip Levis and David Gay. TinyOS Programming. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

2. TinyOS. Event-based operating environment/framework designed for use with embedded networked sensors, to support concurrency intense operations needed by. Only available online: <http://www.tinyos.net/>

3. David E. Culler. Tinyos: Operating system design for wireless sensor networks. Only available online: <http://www.sensorsmag.com/networking-communications/tinyos-operating-system-design-wireless-sensor-networks-918>.

4. Private Company deals with food warehouse temperature monitoring
<http://www.tempgenius.com/TempGenius%20Humidity%20&%20Temperature%20Sensor.pdf>

5. Cooltrax also deals with warehouse monitoring
<http://cooltrax.com/cold-chain/for-warehouse/>

6. Elpro Company
<http://www.elpro.com/en/industries/food/application/warehouse-central/ic/Application/ia/show/>

7. Android Application development-- <http://developer.android.com/index.html>

8. Research Paper
http://www.researchgate.net/publication/260071804_Ubiquitous_monitoring_solution_for_Wireless_Sensor_Networks_with_push_notifications_and_end-to-end_connectivity.