

Hiwot Bayissa

Developing Cloud Based Web Application for Wireless Sensors

Case: Contactless Bed Sensor for Health Monitoring

Helsinki Metropolia University of Applied Sciences

Master of Engineering

Information Technology

Master's Thesis

Date May 22, 2017

| | |
|--|--|
| Author(s) Title | Hiwot Bayissa Developing Cloud Based Web Application for Wireless Sensors Case: Contactless Bed Sensor for Health Monitoring |
| Number of Pages Date | 40 pages |
| Degree | Master of Engineering |
| Degree Programme | Information Technology |
| Instructor(s) | Erja Nikunen, Principal Lecturer |
| <p>Integrating internet of things with cloud computing has become a key solution for properly managing huge amounts of data generated from IoT devices. Health monitoring sensors are one of the entities of IoT devices that bring improvement on the quality of health care service.</p> <p>The focus of this thesis was to develop a prototype application for a sensor that monitors physiological data of patients or elderly. The data generated from this sensor needs to be stored and visualized for caregivers to keep track of the wellbeing of patients.</p> <p>A cloud based platform was used for storing the data in a cloud database as well as deploying the application displaying the data. The result was tested by connecting the sensor with the application server and both the storing and visualizing of the sensor data were successful.</p> | |
| Keywords | Sensor, Cloud Computing, Visualization |

Contents

Abbreviations and Terms

List of Tables

List of Figures

List of Listing

List of Tables

Abbreviations and Terms

List of tables

List of Figures

List of Listings

| | |
|--|----|
| 1 Introduction | 1 |
| 1.1 Scope and Objective | 1 |
| 1.2 Research Method | 2 |
| 1.3 Solution | 2 |
| 1.4 Outline | 3 |
| 2 Background Information and Literature Review | 4 |
| 2.1 Internet of Things | 4 |
| 2.1.1 Internet of Things Architecture | 5 |
| 2.2 Wireless Sensor Network | 6 |
| 2.3 Cloud Computing | 6 |
| 2.3.1 Characteristics of Cloud Computing | 6 |
| 2.3.2 Cloud Computing Service Models | 8 |
| 2.3.3 Cloud Computing Deployment Models | 10 |
| 2.3.5 Cloud Computing Data Storage | 12 |
| 2.4 Google Cloud Platform | 12 |
| 2.4.1 Components of Google Cloud | 12 |
| 2.4 Data Visualization | 14 |
| 2.4.1 Data Visualization Process | 14 |
| 2.4.2 Data Visualization Techniques | 15 |

| | |
|--|----|
| 2.4.3 Visualization for Health | 16 |
| 2.5 Related Work | 16 |
| 3 Research Methods and Materials | 18 |
| 3.1 Application Overview | 18 |
| 3.2 Materials and Tools | 18 |
| 3.2.1 Google App Engine (GAE) | 18 |
| 3.2.3 HTML5, CSS3, jQuery and Flot chart | 19 |
| 4 Research Implementation | 20 |
| 4.1 Requirements for Application | 20 |
| 4.2 Working Environment Setup | 20 |
| 4.2.1 Installing Python 2.7 | 20 |
| 4.2.2 Installing Google Cloud SDK | 21 |
| 4.2.3 Installing App Engine SDK for Python | 21 |
| 4.2.4 Installing Third Party Libraries | 22 |
| 4.2.5 Installing MySQL | 22 |
| 4.3 Design Architecture | 23 |
| 4.4 Implementation of Backend Application | 24 |
| 4.4.1 Sensor Configuration | 24 |
| 4.4.2 Posting and Handling Data | 27 |
| 4.4.3 Database Structure and Storing Data | 30 |
| 4.5 Implementation of Client Web Application | 33 |
| 4.5.1 Transmission of Data to Client | 33 |
| 4.5.2 Creating Chart | 35 |
| 5 Results and Evaluation | 37 |
| 6 Conclusion | 38 |
| References | 39 |

Abbreviations and Terms

| | |
|------|--|
| AJAX | Asynchronous JavaScript and XML |
| BSN | Bed Sensor Network |
| CSS | Cascading Style Sheet |
| GAE | Google App Engine |
| GUI | Graphical User Interface |
| HTTP | Hyper Text Transfer Protocol |
| IaaS | Infrastructure as a Service |
| IoT | Internet of Things |
| JSON | JavaScript Object Notation |
| NIST | National Institute of Standards and Technology |
| PaaS | Platform as a Service |
| RFID | Radio Frequency Identification |
| SaaS | Software as a Service |
| VM | Virtual Machines |
| WSN | Wireless Sensor Networks |

List of tables

| | |
|--|----|
| Table 1. Cloud computing service model summary | 9 |
| Table 2. Google Cloud Components | 13 |
| Table 3. Description of Visualizing Process | 15 |

List of Figures

| | |
|--|----|
| Figure 1 . Cloud computing essential characteristics | 7 |
| Figure 2. Different types of cloud deployment | 12 |
| Figure 3. Google App Engine Launcher | 23 |
| Figure. 4. Proposed Architecture | 24 |
| Figure 5. Configuration and Calibration using configuration WEB page | 26 |
| Figure 6. Tables of the database used | 31 |
| Figure 7. Sample JSON Data | 35 |
| Figure 8. Combined 24 hour chart | 36 |

List of Listings

| | |
|--|----|
| Listing 1. Command for running Google cloud SDK | 22 |
| Listing 2. Sample XML data that is generated by muRata sensor node | 27 |
| Listing 3. URI router snippet | 29 |
| Listing 4. Processing measurement data for storage | 30 |
| Listing 5. Code snippet that stores parsed data to database | 32 |
| Listing 6. Flask API Application | 34 |

1 Introduction

Rapid changes of technologies take place faster than ever before. Everything has become connected to everything via internet with a variety and large quantity of applications being developed alongside. Internet of Things (IoT) is the term used to describe these statements. It is the network of multiple objects and things connected together to the internet for automatic sensing, information processing, understanding, reasoning, decision and response to the virtual and physical world (Vermesan & Friess, 2013).

Entities of IoT such as sensors, actuators, mobiles, and computers generate a massive amount of real time data. Effectively storing and handling the huge volume of data locally is quite costly and almost impossible. A better solution is the integration of IoT with Cloud Computing. Cloud computing is a virtualized infrastructure that can be used to utilize resources efficiently. Such Services, provided by cloud computing include, networks, storages, and servers. Therefore, integrating IoT with cloud computing has great advantage in terms of having scalable resource usage, automatic management of resources, on-demand resource usage and so much more.

Among different IOT applications, eHealth has a great impact on improving today's health care service. Monitoring chronic diseases, lifesaving in emergency situations and ability to provide round the clock health care to patients in hospitals and elderlies in their home or care centers are just some benefits of eHealth.

A contactless bed sensor manufactured by muRata Electronics is used as a case study in the thesis. This device is targeted to be used in hospitals, care centers for the elderly, assisted living and homes. The sensor monitors the well being of patients by continuously measuring heart rate, respiration rate, relative stroke volume etc. The measured data is sent to cloud through wireless sensor network (WSN) as an XML raw data. The stored data is then accessed using HTTP REST APIs. Therefore, an application that captures and places the data to a permanent data storage is required. Moreover, the data collected and stored also need to be visualized for further analysis.

1.1 Scope and Objective

The main focus of the thesis was to develop a prototype application utilizing the information gathered by the contactless bed sensor. The application has two major

components, the backend and frontend. The backend application receives and stores the measured data from the sensor to a database. The fact that a large number of sensors sends information to the backend each second leads to the generation of a huge amount of data. For this reason, a cloud based database is indispensable to collect and store the data. The front-end application is responsible for making an http request to get the data as JSON (JavaScript Object Notation) dumps and generate visualization in terms of the collected measurement data.

To achieve this goal, a backend application was implemented to process and store the data posts made by sensor nodes. The contactless bed sensor has a configuration method which enables it to be connected to the web server that runs the data receiving application. With the configuration capability, a group of sensor nodes can be connected to the cloud running the backend being identified by the network ID they are connected from. The record stored in the database is identified by the network ID and node ID attributes sent as part of the sensed data. The client is then able to make queries to get data specific to some patient or person being monitored and present it to the practitioners in a way that makes the result easily understandable.

1.2 Research Method

An application was designed to receive the data generated from sensors. The backend is written with the Python scripting language and the front-end is designed with JavaScript, HTML5 and CSS3. As a deployment environment, Google Cloud Platform is used as Platform as a Service (PaaS) for deploying the application on a cloud infrastructure. The data is stored in a relational database.

1.3 Solution

The integration of IoT devices with the cloud computing is the main contribution of this thesis. The massive amounts of real time data generated by sensors are saved in a cloud database with unlimited storage capability. The implementation of the user interface enables the data to be displayed in a meaningful way for end users. Existing techniques are used to insure availability, scalability, and security of both the application and the data stored.

1.4 Outline

The structure of the remaining thesis is as follow. The second chapter outlines the overview of the research in internet of things and cloud computing. The third chapter focuses on the materials and methods used to design and implement the application for collecting and analyzing the data. The fourth chapter describes the design and implementation of the application. The fifth chapter explains the result and evaluation of the application. Finally, the last chapter provides the conclusion and future work to be considered.

2 Background Information and Literature Review

This section describes the technological areas related to the project. These areas are IoT, Wireless Sensor Network and Cloud Computing. Previous studies conducted to develop applications of wireless sensor networks by integrating with cloud computing are also described in the coming sections.

2.1 Internet of Things

IoT has become a widely spread technology that touches everyday life. Everything is becoming connected to everything with the internet which provides information for building variety of services that changes our way of living. Buildings, vehicles, goods, appliances, plants are some of the things that are networked on the internet. By the year 2020, it is estimated that over 26 Billion devices excluding Smartphone, PCs and tablets will be connected to the internet. The estimation shows 30 times increase of the devices that will be connected compared to the year 2009 where there were around 0.9 billion devices (Gartner, 2013).

Many studies have been made to bring IoT to the current level of technology but the root concept of the term IOT was first put in general form by Professor Kevin Ashton while he was working in Auto-ID lab in Massachusetts Institute of Technology in 1999. He and his group were conducting research in the field of radio frequency identification (RFID) by connecting objects to the network which opened the door for connecting things to the internet. (Journal, 2002)

According to Gartner, the definition of IoT is “The Internet of Things is the network of physical objects that contain embedded technology to communicate and sense or interact with their internal states or the external environment.” (Gartner, 2013) .It is predicted that objects which can be recognized, addressed, located, recognized and controlled by means of internet technology such as WSN (wireless sensor network), RFID, wide-area network and more, will create a huge information network. Therefore, the objective of IoT is to make everything in our environment be connected with each other at anytime and anyplace. Specifically in healthcare, collecting, recording and analyzing patients’ data by means of IoT technologies means improving the quality of healthcare services by minimizing errors that could occur by human data collection.

2.1.1 Internet of Things Architecture

IoT architecture is fundamental for having standardization on implementing a system. IoT is composed of four layers namely; Application Layer, Service and Application support Layer, Network Layer and Device Layer.

Application Layer contains variety of applications that industry sectors use IoT system for enhancing their services. The service domains that application layer contains are, Smart Cities, Smart Home, Smart retail, Smart Office, Smart Agriculture, Smart Energy and Fuel, Smart Transportation and Smart Military. (Kavis, 2014)

Service and Application support Layer provides service management for connecting and communicating IoT devices which implement similar service type. It has got a link to the database for storing the information it collects from the Network layer. In addition, it has a decision unit where it gives automatic decision by processing the information and performing ubiquitous computation. (Fortino, 2014)

Network Layer is a layer that is responsible for the transmission of information from sensor devices to the system that process information. It has control function which provides security for accessing and transporting resources. The technology for transmission varies depending on the sensor device. Examples of such technologies are Infrared, Bluetooth, Wifi, Zigbee and so on. (Fortino, 2014)

Device Layer is physical object that contains sensor devices. It is responsible for identifying and collecting object specific information from the sensor devices. The information varies depending on the sensor device. Some examples of the information can be Humidity, Temperature, Motion, Location, Vibration etc. The Network layer then receives this information for securely transmitting to the system that process the information.

2.2 Wireless Sensor Network

Wireless network technologies (WSN) have a great impact on the growth of today's IoT. It is now being used in several commercial, governmental, industrial and environmental areas. WSN is defined as Network of devices or nodes that sense and collect information from the surrounding and communicate via the wireless links to other controllers or monitors to be utilized locally or with other network through gateway. (Verdone, Dardari, and Mazzini, 2008)

In recent years, the application area of IoT has become immense and diverse and it is spreading in all areas of individuals' everyday life, society and enterprises. As a result, wireless sensor networks in healthcare brought a greater impact on improving the quality of service on giving care for patients. For example, real time patient monitoring with WSN enable early detection of health status before deterioration (Chipara et al., 2009), quality of life among elderly people improves with the help of smart environment (Wood et al., 2008), and provide more field of studies on human behavior and diseases (Lorincz et al., 2004). Despite this fact, resources for data storage is one challenge that applications of healthcare face for storing information collected from WSNs (Fortino and Pathan, 2014).

2.3 Cloud Computing

Cloud Computing is a model that runs applications and services through the internet with distributed resources (Antonopoulos & Gillam 2010). These resources are limitless in a way that users can utilize them depending on the need. In addition, the software that runs on the physical systems is hidden from the users which will result for companies to focus on what really matters and avoid extra load of hiring employees for maintaining and troubleshooting the systems.

2.3.1 Characteristics of Cloud Computing

The U.S. National Institute of Standards and Technology (NIST) have codified five essential characteristics of cloud computing which make compute infrastructures of IT to be called as Cloud Computing. These characteristics are: On-demand self-service, broad software network access, resource pooling, rapid elasticity and measured service. (Brown, Swenson, 2016).

Figure 1 represents the five cloud computing essential characteristics pictorially. Each characteristics definitions and examples are also described below.

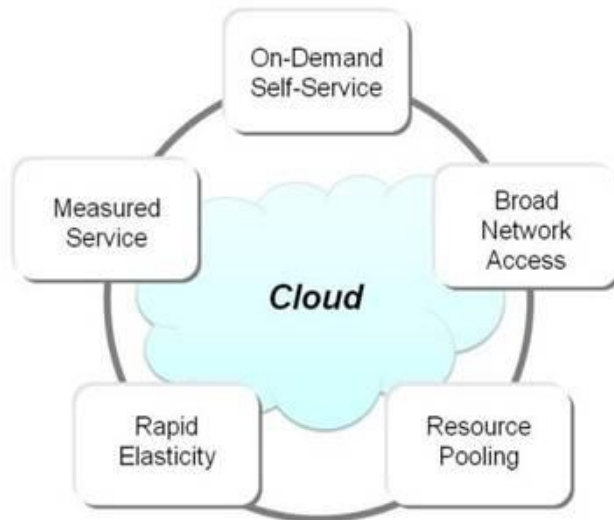


Figure 1 . Cloud computing essential characteristics

On-demand self-service: enables consumers to provision the computing infrastructure such as storage, server time, network, power and software, in accordance with the demand. The user can scale in and out the required infrastructure without interaction of human with the vendors of the cloud computing services.

Broad network access: Capabilities of cloud computing available on top of the network can be accessed by variety of devices such as Laptops, Workstations, Smartphone, Thin clients etc.

Resource pooling: IT resources such as servers, file storage, software and network bandwidth are pooled by vendors of cloud computing infrastructure in order to serve multiple customers of cloud services.

Rapid elasticity: cloud computing consumers are able to purchase facilities of cloud from vendors in order to scale in and scale out the resources they use without limit at any time.

Measured service: Cloud service providers control and monitor resources that are utilized by measuring with automatic cloud system. As a result of this, consumers are billed for only the resources they used from the cloud service provider.

2.3.2 Cloud Computing Service Models

Cloud computing has got three main service models known as Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS). These service models are replacing the traditionally existed on-premises data centers. As a result, the efforts needed by consumers in order to maintain and manage IT systems are reduced. (Sosinsky, 2011, p.65)

IaaS provides infrastructure resources that contain hardware and software components. Some of these components are; Cloud Software, Computer Hardware, Network and Internet Connectivity, Platform Virtualization, Utility Computing, Service Level Agreements (Sosinsky, 2011, p.66-68). It is the abstraction of physical IT infrastructure such as servers, networking, and disk storage are hidden from the user and made available as a set of services by means of web based management consoles or coding. In IaaS, developers and administrators are still needed to design and code the whole applications and also maintain and manage the entire IT systems respectively. IaaS only abstracts the physical data centers and infrastructures of the computer systems for its consumers (Kavis, 2014, p. 13).

Some of the main advantages of IaaS are; it reduces the requirement to have data centers, it eliminates the need for purchasing and assembling hardware, it reduces the cost incurred for managing data centers and infrastructures. Moreover, it enables end-users not only to concentrate more on developing and managing applications than managing data centers, but also use the resources as their own demand and pay only for the measured resources they use (Kavis, 2014, p. 15)

PaaS provides a platform where consumers can maintain and develop their own applications on a cloud based infrastructure. The utilities such as networks, storages, operating systems, servers, middleware for programming runtime environment, development tools etc. are provided by PaaS service providers. Consumers are not responsible in maintaining the infrastructure as well as deploying the platform. They only use a browser based utility on the platform and develop their applications and pay for only the services they use (Antonopoulos & Gillam 2010).

Some of the advantages of PaaS cloud service model are; high-level infrastructures accessibility, scalability of services, reduced cost of administration (Antonopoulos & Gillam 2010).

SaaS provides sheer application of cloud service. It is software deployed by host cloud providers and can be accessed through the internet. The cloud service provider is responsible in handling the entire infrastructure, deployment, application logic and everything related with the process on delivering the services to its consumers. Consumers are only responsible in managing their users and configuring some specific application parameters. The applications running on the cloud infrastructure can be accessed either by using Application Programming Interface or by using thin client or web browser (Kavis, 2014, p. 42).

Main advantages of SaaS are on demand availability over the internet, reduced costs of maintenance and monitoring, automatic upgrade and update of software, on demand scalability, support for multiple users on through multi-tenancy model and so on (Antonopoulos & Gillam 2010, p.72). Table 1 provides a summary of the service model.

Table 1. Service model summary

| Description | Service Models |
|---|------------------------------|
| End user application delivered as a service rather than on-premises software | Software as a Service (SaaS) |
| Application platform for middleware as a service on which developers can build and deploy custom Applications | Platform as a Service (PaaS) |
| Compute, Storage or other IT infrastructure as a service, rather than dedicated capability | Infrastructure as a Service |

Different persons use cloud service models based on the need. For example IaaS is primarily used by IT service providers, PaaS is primarily used by developers and SaaS

is mainly used by end users.

2.3.3 Cloud Computing Deployment Models

Cloud deployment model is classified into four groups based on the type of cloud environment. These are Private Clouds, Public Clouds, Hybrid Clouds and Community Clouds.

Private Clouds: According to NIST definition, private cloud is exclusive use of cloud infrastructure by a single organization which may consists of multiple consumers or business units (Brown and Swenson, 2016).

The organization can host private cloud either on-premises or off-premises (hosted cloud). For on-premises implemented private cloud, the consumers manage the data center and they can obtain any kind of hardware configuration. For hosted private cloud, consumers are dependent on the service provider due to the fact that the cloud infrastructure is provided from a vendor, but all the resources are private and are not shared with any other consumers. Private cloud has a single tenant deployment nature and because of this, it has got some advantages such security, privacy, data ownership. Despite these, it loses some of the core cloud computing advantages such as on-demand scalability, rapid elasticity, resource pooling and pay as you go pricing which can be considered as disadvantages.

The NIST definition of public cloud is that, open usage of the cloud is supplied for general public users. The cloud infrastructure is provided in the premises of service provider and it may be owned by different entities such as governmental organizations, academic institutions or business entities (Brown and Swenson, 2016). Public cloud is a multi-tenant environment where different clients share the same computational resource.

Some of the main benefits of using public cloud are, utility pricing where clients pay for only the resources they use. This allows the user to scale up and down and consume what is needed. The other benefit is elasticity where the client handles the compute resources to increase and decrease during pick time and vice versa. Moreover, core competency is another benefit where end user outsources its data center to other companies for management and spends less time to managing the infrastructure.

Leveraging public cloud involves some risk such as control of the infrastructure where end users must depend on the vendors to get the best performance. In addition, regulatory issues such as data privacy and security are another challenge of using public clouds.

A hybrid cloud is defined as a cloud model that combines two or more cloud models such as private, public or community in order to enable portability of data and application (Brown and Swenson, 2016). As it is mentioned in the previous paragraphs, use of hybrid cloud ease the problem that one can encounter while using private cloud or public cloud. By using public cloud more in order to get the benefits of cloud services and by leveraging private cloud for sensitive operations that need privacy and security; clients can get the advantage of hybrid cloud.

Community cloud means the sharing cloud infrastructure between several organizations with similar interests and concerns. Community cloud can be hosted internally in the community or externally by third party. The management of the infrastructure can also be carried out by own or externally. Figure 2 summarizes the types of deployment models in cloud infrastructure.

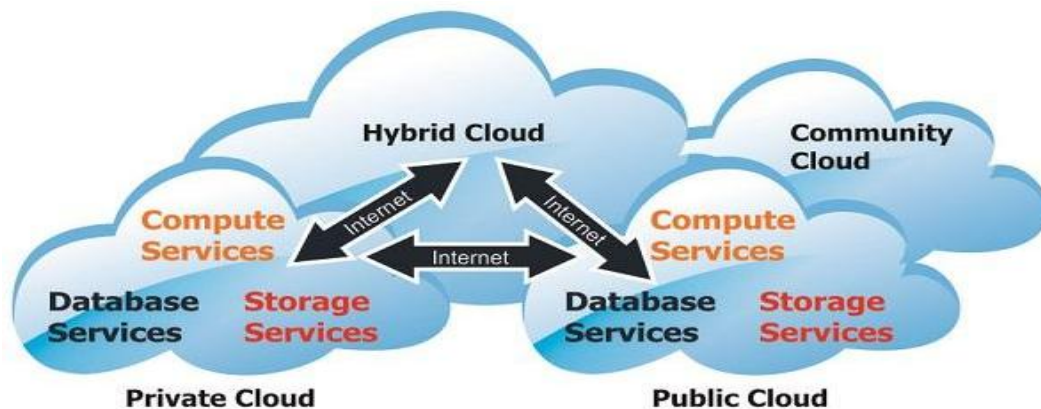


Figure 2. Different types of cloud deployment (What is Cloud Computing and What are Its Advantages and Disadvantage?, 2012)

The deployment models in cloud infrastructure are Private clouds, Public cloud, Community cloud and Hybrid Cloud. Figure 2 also describes the services each models

provides.

2.3.5 Cloud Computing Data Storage

One of the most important services that cloud computing provides is data storage. Despite the fact that it has challenges in creating secure and reliable data storage facilities, it is providing great services for diversity of business firms as well as industry. With the help of networking, storing data in a local server and PCs attached storage has moved to distributed storage in the cloud. Distributed storage use similar hardware technology for storing information as directly attached storages of local computers and servers use to store information. (Shvartsman, 2014)

2.4 Google Cloud Platform








Google cloud platform is a cloud infrastructure that provides cloud based services. The platform offers infrastructure for building web sites and applications with high speed networking, rapid accessibility, security and scalability. Moreover, google cloud platform provides pay as you go model that enables users to pay for only the resources they use.

2.4.1 Components of Google Cloud

Google cloud platform can be divided into three. Compute, Storage and Service. Each of these platforms has components for storing and analyzing data. These components are Google Cloud Storage, Google Cloud SQL, Google BigQuery, Google App Engine, Google Compute Engine and Google Endpoints. (Google Cloud Platform, 2017)

Table 2 shows the wide range of computing services that Google cloud platform provides. The Google cloud platform enables customers to have a flexible and fully utilized application platform.

Table 2. Google Cloud Platform Components

| Google Cloud Platform | Platform components | Definition and Benefit |
|-----------------------|--|--|
| Compute |  App Engine | Platform for developing and hosting application. (Paas) Benefits: Security, scalability, reliability, cost Saving |
| |  Compute Engine | IaaS that runs services such as Gmail, youtube. Enables users to create VM. Benefits: Scalable, low cost, fast and efficient networking, load balancing |
| Storage |  Cloud Storage | RESTful online storage for storing and retrieving unstructured data. Benefits: cost effective, availability, security, consistency |
| |  Cloud SQL | Fully managed relational database. Benefits: Availability, security, fully managed |
| |  Cloud Datastore | NoSQL document database. Benefits: automatic scaling, high performance, flexible storage |
| Services |  Cloud BigQuery | Data warehouse for storing and querying data. Benefits: Reliability, scalability, maintenance. |
| |  Cloud Endpoints | Framework consisting tools, libraries and capabilities for deploying and managing API's. |

The wide variety of options offered by the Google cloud platform enables users to concentrate on the actual job to be performed instead of spending time, money and energy for organizing an operational development environment. Users can choose from the components of the Google cloud platform described in Table 2 depending on the service they require and pay only for the services they use. For example, the fully

managed App Engine service allows users to write a code, test locally and deploy without worrying about the operation to keep the application running and available all the time.

2.4 Data Visualization

It is known that a sensor generates data with a numerical value which makes it difficult to retrieve and get understanding of the data. For this reason, visualizing the data is essential in order to make the task of getting a full understanding of it simple. Visualization makes understanding and interacting with data much easier by analyzing, extracting and presenting the data graphically.

Data visualization is a technique to represent, describe and bring understanding of large amount of information in an easy and effective way. In general data visualization enables users to understand the importance of data by putting it in pictorial or graphical format. (Fry, 2007)

2.4.1 Data Visualization Process

Although there are different approaches for information visualization, the general process to visualize data remains the same. Ben Fry (2007) describes seven steps for effective data visualization. These are, Acquire, Parse, Filter, Mine, Represent, Refine and Interact. Table 3 describes each steps of visualizing process.

Table 3. Description of visualizing process

| Description | Visualization Process |
|---------------------------------------|-----------------------|
| Obtaining the data | Acquire |
| Structuring and categorizing the data | Parse |
| Avoiding unnecessary data | Filter |
| Applying methods | Mine |
| Choosing visualization models | Represent |
| Improving the visualization | Refine |

| | |
|---------------------------------------|----------|
| Manipulating and controlling features | Interact |
|---------------------------------------|----------|

Depending on the type of data to be visualized, some of the steps shown in Table 3 may or may not be followed for visualization process. After the data is acquired from either a large data system or simple text file, it will be parsed to be tagged according to its use. Next to this process, filtering and mining of the data process takes place. The represent step determines the form of the data such as trees, lists etc. The refine step evolves more details of presenting the data. The final step is the interact step which lets the user for controlling the way the data is displayed.

2.4.2 Data Visualization Techniques

Data is visualized to give a brief and effective understanding of data. Identifying a visualization technique according to the data type and the purpose of the visualization is important to have a clear understanding about the data. There are several types of charts and graphs to represent data. In this paper five types of charting methods are discussed.

Bar chart

It is a chart used to display relative sizes of different data groups. It displays a comparison of different data with bars of different height plotted in a graph horizontally or vertically.

Line Chart

Graphically displays data that changes continuously over time. Series of points connected with a line show the continuous change of the data over a time sequence. Line chart is useful for displaying trends of data changing with a time series interval.

Pie Chart

It displays data in a circular graph which is partitioned in to sectors to represent the amount of data existence. It is useful to display a comparison between static data.

Scatter Plot

Represent data graphically by mapping set of data points with a plot in two axes. Each point determines the position of the data value in a horizontal and vertical axis. It is used to identify relationship between one data variable and the other how one affects the other.

Tree map

It is a visualization method that represents hierarchically structured data by splitting up the size of the data variable with rectangles. Tree map is useful to make comparison of nodes and sub nodes by looking the data presentation in one picture.

2.4.3 Visualization for Health

Different approaches have been made recently to visualize patients' health information. Displaying patient's health history in a time series visualization technique facilitates monitoring of health status with a given period. Several systems have been implemented in the area of manipulating effective health data. Review of such systems can be found in (Taowei, 2008).

2.5 Related Work

In recent years, wireless sensor networks have been widely used for enhancing ubiquitous health systems. Since the use of wireless sensors for monitoring patients, demand to have a high quality and cost effective data management, integrating wireless sensor network with cloud computing is a solution that interests researchers to conduct researches in the area. In this section, related works with this thesis in regard to the design and implementation of wireless sensors applications in cloud infrastructure are presented.

A similar work by (Koch 2010) was performed on developing a framework for collecting patients' data. The main focus was to eliminate the manual work performed by healthcare professionals for collecting information. A system was proposed for collection of patients' information automatically by connecting medical equipments with wireless sensor networks. Cloud computing is used for storing, processing and analyzing the data.

A Master thesis (Song and Xu, 2013) describes a web based application of a self-care system for Diabetic patients. The main objective of the authors was to develop an application which can be accessed in any platform as well as store vast amount of data in a cost effective way in cloud infrastructure.

3 Research Methods and Materials

This chapter briefly describes the methodology and materials used for receiving, storing and visualizing data from the sensor. It includes the research context where sensor configuration, receiving data, storing data and visualizing are described in details and also the materials and tools used.

3.1 Application Overview

The aim of this project was to develop a web application that visualizes data collected from IoT devices. The case study was done with a contactless bed sensor. This sensor provides vital patient information such as heart rate, respiration rate, heart rate variability which associates with stress level, relative stroke volume which is the amount of blood the heart pumps, bed status (weather the bed is empty, occupied by patient or patient is moving), data log of timestamp, signal strength etc. (Co, 2017).

3.2 Materials and Tools

To implement the prototype application, different platforms and technologies were used. In the following subsections, the technologies used are briefly described.

3.2.1 Google App Engine (GAE)

Among different PaaS cloud computing infrastructures, GAE was chosen for building the IoT Web application for the reason that it is free to start working with the cloud infrastructure until usage of resources exceeds free level. It is a platform used to build and run scalable web applications with built-in services. Only uploading the application is needed and maintaining and provisioning of servers is abstracted.

3.2.2 Python

Python programming language was used for writing the project application code. The language has easy syntax for scripting applications and high-level data structure. Among several cloud platforms, GAE uses Python as one of the programming language for developing web application.

3.2.3 HTML5, CSS3, jQuery and Flot chart

HTML5 was chosen for developing the client side application of the project. It was the latest version of Hypertext Markup Language which is flexible in developing webpage that is capable of loading pages quicker, accessible from any device and easy to add video, audio and images compared to the previous versions.

Cascading Style Sheets (CSS3) was also used for styling the view of the webpage. By combining HTML5 and CSS3 for this project, a responsive web design technique was used to build the web page to be accessed with any device.

For manipulating and interacting with the different HTML DOM elements, a JavaScript library called jQuery was used. Other than handling the DOM elements, a charting library called Flot.js was used.

4 Research Implementation

This chapter describes the implementation of the prototype application developed for collecting and visualizing data from the contactless bed sensor which is used as project case study. The application requirement, design architecture and implementation of the backend and front-end will be discussed briefly in the following subsections.

4.1 Requirements for the application

The purpose of the project was to develop a prototype application which handles big data generated from the contactless health monitoring sensors. Each sensor node is connected with cloud server interface API which contains interfaces for sending the measured sensor data to the cloud (Co, 2017). Therefore, the main goal was to receive these data from the cloud and store permanently in data storage for data visualization. As a result, the following functional areas are identified as the basic requirements to be fulfilled by the prototype application:

- Receive sensor data
- Connect the sensor nodes to the application
- Parse the data
- Store the data in cloud storage
- Visualize all the measured data in a chart

4.2 Working Environment Setup

The environment set up that was used while developing the prototype application needed Python 2.7, Google Cloud SDK, Google App Engine SDK and MySQL server. The backend was written with Python script, the database was developed on MySQL. The Google App Engine was used as local server and later to deploy the application to Google cloud server.

4.2.1 Installing Python 2.7

One of the languages supported by the Google Cloud Platform is Python. Therefore, Python 2.7 was installed on the local development environment. The latest version of Mac OS X, Sierra, comes with Python 2.7 out of the box, so there was no a need for fresh installation. But in order to make python portable with the database, MySQLdb

was installed. MySQLdb is an interface that provides an API for python database.

4.2.2 Installing Google Cloud SDK

Google Cloud SDK provides a set of tools for managing products of the Google Cloud Platform such as Google Compute Engine, Google Cloud Storage, Google BigQuery and other products and services. For the purpose of this project, Google App Engine and Google Cloud SQL are chosen from the compute and the storage products of the cloud platform.

Google cloud SDK is installed on the computer by running a single line of command on the terminal. After the installation is completed, the cloud SDK is authorized so that the tools can access the compute platform. The command that performs these tasks are shown in Listing 1.

```
curl https://sdk.cloud.google.com | bash
gcloud auth login
```

Listing 1. Command for running Google cloud SDK

By running the above command in a terminal of Macintosh computer, Google clod SDK can be installed easily.

4.2.3 Installing App Engine SDK for Python

Google App Engine SDK provides tools for developing, deploying and managing App Engine applications. While in the development phase, it provides a local development server where the application can be run and tested before actual deployment. The installation is done by downloading the Google App Engine SDK. For this project the SDK to set up the Python environment was installed. Figure 3 shows the Google App Engine Launcher.

| Name | Path | Admin Port | Port |
|--------------------|--|------------|-------|
| ● guestbook | /Users/hiwotbayissa/guestbook | 8001 | 8080 |
| ● sensor-applic... | /Users/hiwotbayissa/Desktop/SensorApp/data/push | 8003 | 8088 |
| ● sensorlistner | /Users/hiwotbayissa/Desktop/sensorListnere/data/push | 8004 | 11080 |
| ● sensor-applic... | /Users/hiwotbayissa/Desktop/AppUpload/data/push | 8005 | 8080 |
| ● testsensor-12 | /Users/hiwotbayissa/Desktop/finalapp/data/push | 8004 | 8080 |
| ● testsensor-12... | /Users/hiwotbayissa/Downloads/finalapp/data/push | 8006 | 12080 |
| ● sensor-applic... | /Users/hiwotbayissa/Documents/SensorProject/tested... | 8007 | 13080 |
| ● application | /Users/hiwotbayissa/Desktop/hw/data/push | 8008 | 14080 |
| ● testsensor-12... | /Users/hiwotbayissa/Desktop/hw 2/data/push | 8011 | 15082 |
| ● testsensor-12... | /Users/hiwotbayissa/Documents/SensorProject/finalap... | 8012 | 16082 |
| | | | |
| | | | |

Figure 3. Google App Engine Launcher

As shown in Figure 3 above, after installing the SDK, the App Engine for managing local applications and later deployment of an application to the cloud becomes available. Therefore, installing the Cloud SDK automatically sets a local development server as well as the tooling for deploying and managing the application in App Engine.

4.2.4 Installing Third Party Libraries

The App Engine SDK for Python runtime environment provides several libraries for local development environment. Meanwhile, there are some platform dependent libraries that should be installed locally in order to use them on the local development server. For this project, *xmltodict* which parses the raw data, Flask that allows an application to run as a built in web server and *MySQLdb* that connects Python to MySQL database are the third-party libraries installed and used in the local development environment.

4.2.5 Installing MySQL

Google Cloud SQL provides a relational database that can be used with App Engine Application. In the project a local MySQL client server was installed to store

measurement data from sensor node and be able to connect with Cloud SQL for the final deployment. In addition to MySQL server MySQLdb was installed on the development environment. MySQLdb is a thread-compatible interface to MySQL database server that provides the Python database API.

4.3 Design Architecture

The project implemented has two distinct sections that interact with each other in addition to the sensor nodes that transmit the data. The backend application runs on Google Cloud, processes and stores measurement data. The client application which visualizes measurement data retrieves data in JSON format from the storage. Figure 4 below, depicts the high level design architecture used as a baseline while developing the prototype application.

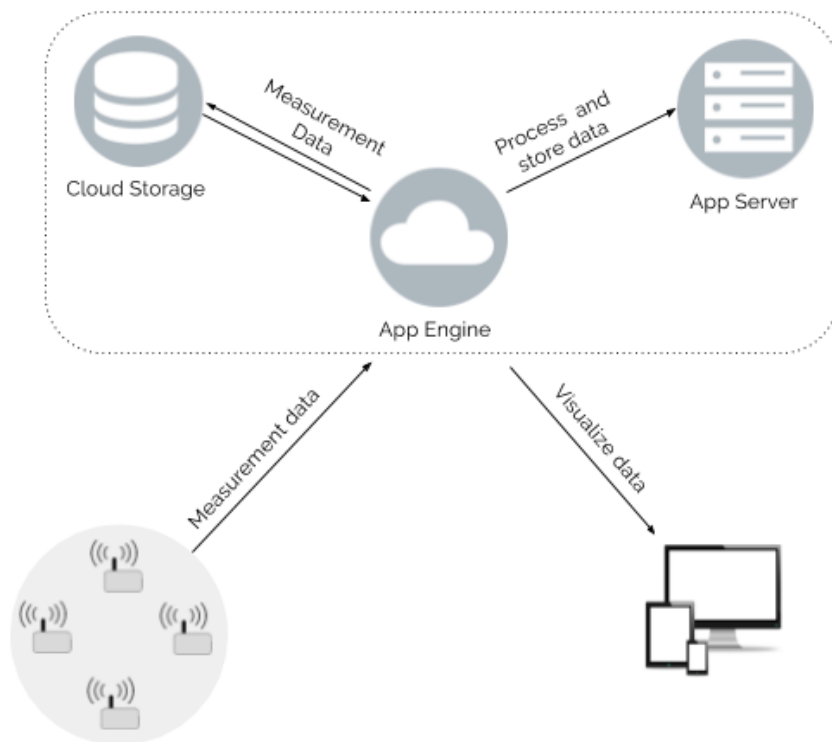


Figure. 4. Proposed Architecture

As shown in Figure 4 the sensor nodes are connected to the cloud application following the wireless connection configuration they come up with. An application that is deployed in GAE cloud infrastructure is responsible for receiving post data from the

different sensors and store it in a relational database hosted on the cloud. The client side of the application is then provided with API's to retrieve and process the data in way that is more relevant to the users and practitioners.

4.4 Implementation of the backend application

In this section of the paper, all the steps and processes followed to implement the backend and front-end are discussed.

4.4.1 Sensor Configuration

The muRata BSN comes with configuration steps that need to be followed to join them to the network they are meant to be used from. Each sensor has a configuration mode where it creates its own network that is identified by a network Id. Through this network, the configuration is done using web page where user can choose between local and cloud mode.

- Attach the sensor to a bed or a chair and check device orientation.
- Remove the BSN from the attachment plate
- Plug in power and wait for 5 seconds
- Attach the BSN to attachment plate
- Connect laptop to the configuration network: <http://192.168.253.1>
- Start BSN configuration

Figure 5 below is a flowchart that shows the steps for setting up BSN to configuration mode.

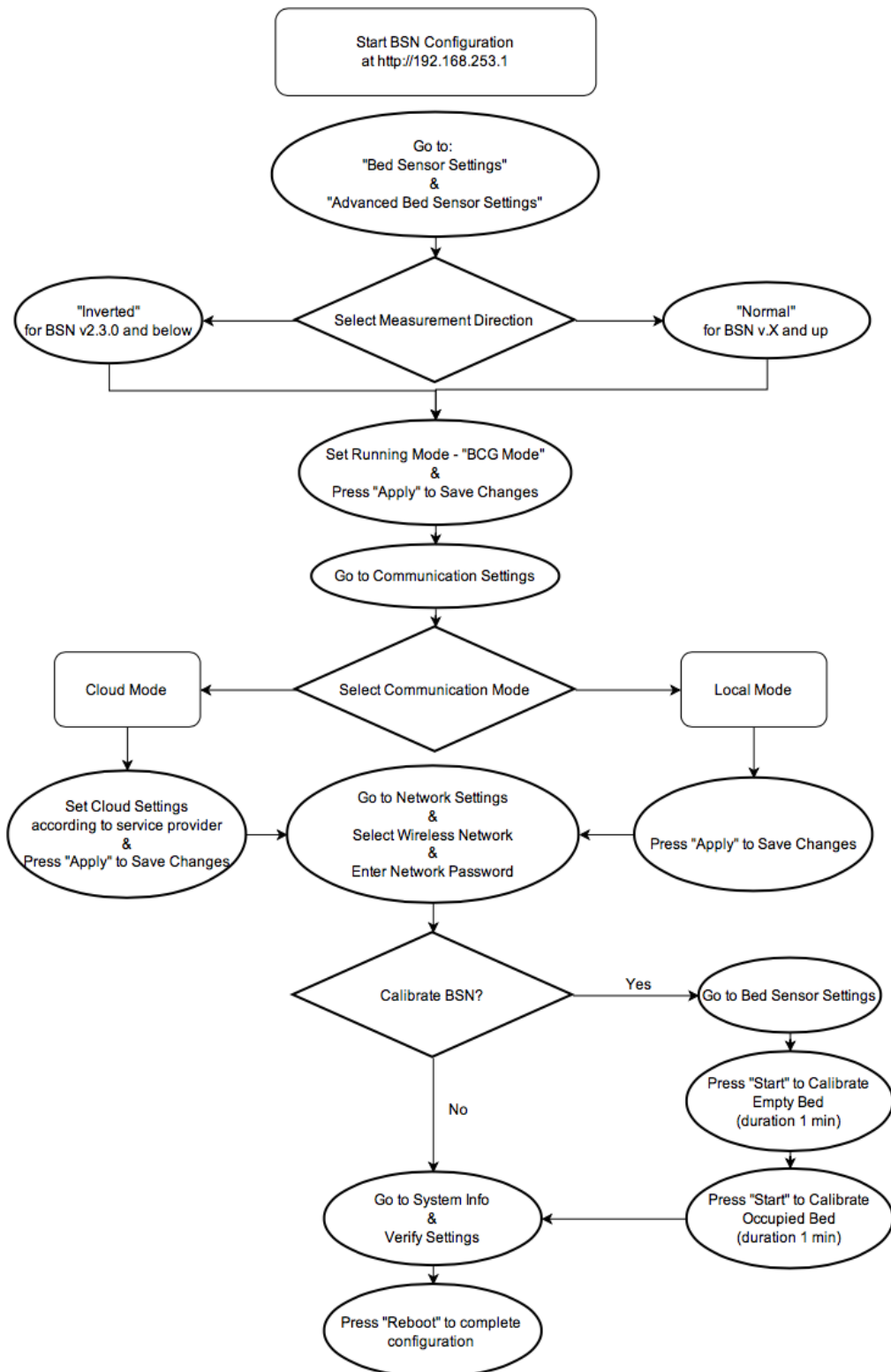


Figure 5. Configuration and Calibration using configuration WEB page (Co, 2017)

Once the sensor node is properly powered on and put in configuration mode, there are two ways that can be followed to complete the setup. In cloud communication mode, WiFi network and the application server URL are configured in order to make connection with the sensor node. With this predefined web server, the sensor starts sending its data as an *HTTP POST* to the application server. In local communication mode the muRata Bed Sensor Node connects to predefined WiFi network. Therefore by choosing between the local and cloud mode configuration, the sensor can be forced to work in local network or be able to send data to a cloud server. In both cases, the message contents from the sensors are expressed *WSN OpenA-PI/XML format [WOA]*. (Co, 2017) Listing 2 is a sample XML document taken from the sensor.

```

<Data version="1.7" xmlns="urn:wsn-openapi:sidf">
  <Network id="test_network">
    <Node id="98">
      <Sensor id="0">
        <Measurement quantity="BioSignal" time="2014-03- 07T13:18:04+00:00">
          <Component id="heart rate" unit="bpm" />
          <Component id="respiration rate" unit="rpm"/>
          <Component id="relative stroke volume" unit="µl"/>
          <Component id="heart rate variability" unit="ms"/>
          <Component id="measured signal strength"/>
          <Component id="status"/>
          <Component id="beat-to-beat time" unit="ms"/>
          <Component id="beat-to-beat time -1" unit="ms"/>
          <Component id="beat-to-beat time -2" unit="ms"/>
          <Values tick="sec">
            10,83,12,39,41,14280,2,911,0,0
            20,83,12,39,41,14280,2,911,0,0
          </Values>
        </Measurement>
      </Sensor>
    </Node>
  </Network>
</Data>

```

Listing 2. Sample XML data that is generated by muRata sensor node

The XML data in Listing 2 shows the structure and content of the data that is sent out from the sensors. Below is a list of the data type and related attributes that are sent out to a local or cloud storage:

- Network ID - Sensor network identifier
- Node ID - Sensor node ID
- Report Interval - Number of samples in one message (5...90)
- Timestamp Reset Multiplier - Time sync / timestamp reset parameter (1...3600)
- Measurement Components
 - ◆ Heart Rate (HR)
 - ◆ Respiration Rate (RR)
 - ◆ Stroke Volume (SV)
 - ◆ Heart Rate Variability (HRV)
 - ◆ Signal Strength (SS)
 - ◆ Status
 - ◆ Beat-to-beat time (B2B)
 - ◆ Beat-to-beat time1 (B2B)
 - ◆ Beat-to-beat time2 (B2B)

The time is expressed in YYYY-mm-ddTHH:MM:SS+00:00. Time interval can also be set in the configuration mode and the time in the data content is reset according to this interval. For the sample data in *listing 1*, the interval is set to 10 seconds. Therefore, the value element shows the measured data within 10 seconds from the previous measurement. Each record in *listing 1* is a set of data representing the Timestamp, HR, RR, SV, HRV, Signal Strength (SS), Status, B2B, B2B, B2B.

4.4.2 Posting and Handling Data

After properly configuring the nodes to send data over the local network or the cloud, the next phase was developing the backend application that handles data storage and retrieval request. Technologies that were used to develop the backend application were MySQL and Python script.

To be able to handle post requests made when sensors send data to the backend, a handler was implemented and put in a path as per the specification of the node manufacturer, muRata. To direct the requests to the proper handler, webapp2 was used.

Webapp2 is a simple web application framework that is provided by App Engine. It is meant to allow the development of a simple web application for the Python 2.7 runtime. Webapp2 provide features that make web application development easier by improving support for URI routing, session management and localization. In this application, the webapp2 was used to handle URI routing. Listing 3 below is the code for the application to handle a requested URI.

```
application = webapp.WSGIApplication([
    ('/report', mainh.MainHandler),
    ('/data/push/', post.PostDataHandler),
    ('/ajax/getData', ajax.GetReportData),
], debug=True)

def main():
    util.run_wsgi_app(application)

if __name__ == '__main__':
    main()
```

Listing 3. URI router snippet

In Listing 3, the snippet that was used to route client URI requests to proper handler is shown. The handler that saves posted data is *post.PostDataHandler*. The handler basically does two tasks; it first gets the post that come as request body. Then it parses the data that come in xml structure and put in proper format to be saved on the database.

The snippet in Listing 4 shows the way the xml data sent by the sensors was processed to get the different measurement values to be stored in the database.

```

def post(self):
    # Get the post from the url and upload
    uploaded_file = self.request.body_file
    doc1 = uploaded_file.read()
    doc = xmltodict.parse(doc1)
    json_data=json.dumps(doc, sort_keys=True, indent=2)

    # Get sensor id
    sensorID=doc['Data']['Network']['Node']['Sensor']['@id']

    # Get network id
    networkID=doc['Data']['Network']['@id']

    # Get node id
    nodeID=doc['Data']['Network']['Node']['@id']

    # Get time of measurement
    timestamp= doc['Data']['Network']['Node']['Sensor']['Measurement']['@time']

    # Format date string
    time = datetime.datetime.strptime(timestamp, "%Y-%m-%dT%H:%M:%S")

    # Get values measured
    values= doc['Data']['Network']['Node']['Sensor']['Measurement']['Values']['#text']

    # Split contents of Values xml node based on new line delimiter
    records = values.split("\n")

    for record in records:
        # Each line is a record to be stored in DB. Use comma delimiter
        # to get the values of of Components.
        x=record.split(",")
        MT= time + datetime.timedelta(0,int(x[0]))
        HR=int(x[1])
        RR=int(x[2])
        RSV=int(x[3])
        HRV=int(x[4])
        MSS=int(x[5])
        Status=int(x[6])
        B2B=int(x[7])
        B2B1=int(x[8])
        B2B2=int(x[9])

```

Listing 4. Processing measurement data for storage

A single post comes with information about the sensor id, network id, node id, measurement timestamp and one or more sets of measurements. As shown in Listing 4, based on the interval with which the sensors are set to send out measurement data, iterating through the amount of measurement data is necessary. After getting the right

values from the xml structure, the next task is to permanently store the values in a database. *Section 4.3.3* briefly describes the database structure and the backend part that handles the saving of data to the database.

4.4.3 Database Structure and Storing Data

One can choose from a couple available storage options that are managed by Google Cloud Platform. For this application, MySQL database was used and Google Cloud provides Google Cloud SQL which is a fully-managed database service meant to make the setup, maintenance, management and administration of a relational MySQL database easy.

After properly setting the billing information for the project and enabling the Cloud SQL Administration API, Cloud SQL instance can be created with proper name and root user. The developed application can then be configured to connect to the Google Cloud SQL instance. For the sake of development, a local environment with MySQL server was used. Figure 7 below shows the two tables that were used for the prototype application.

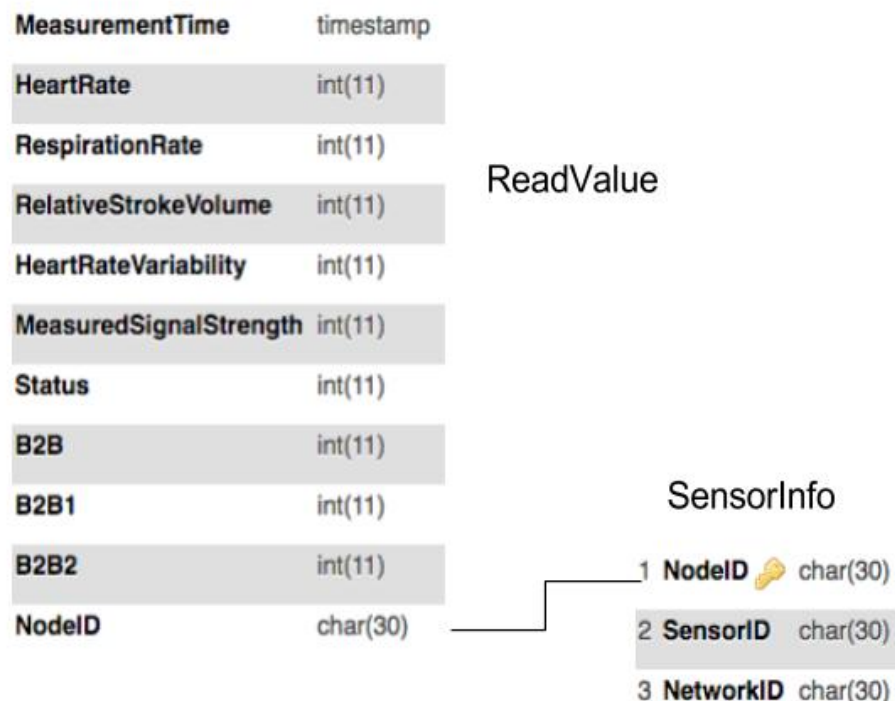


Figure 7. Tables of the database used

The table named *ReadValue* is used to store the actual measurement values that are sent by the sensor. The attributes defined in the table are MeasurementTime, HeartRate, RespirationRate, RelativeStrokeTime, HeartRateVariability, SignalStength, status, B2B, B2B1, B2B2 and NodeId. The NodeId is a foreign key that is used to relate the measurements with specific node.

The table named *SensorInfo* is used to store the identification information related to a sensor. The attributes it has are NodeID, SensorID and NetworkID. The NodeId is the unique identifier of a sensor which is used as a foreign key in *ReadValue* table.

After the XML measurement data is parsed, the values to be stored are structured based on the database design described above. The snippet in Listing 5 below is part of the post handler code responsible for connecting and storing values to the database.

```

# Connect to DB
db = MySQLdb.connect(host='localhost', port=3607, user='root', passwd='', db='sensor_data')
cursor = db.cursor()

# Create table if it is not created yet
cursor.execute('''CREATE TABLE IF NOT EXISTS sensor_data.SensorInfo(
    node_id char(30) primary key NOT NULL,
    sensor_id text NOT NULL,
    network_id char(30) NOT NULL)
''')

# Insert node, network and sensor IDs to DB
cursor.executemany("INSERT IGNORE INTO SensorInfo (NodeID, NetworkID, SensorID) \
VALUES(%s, %s, %s)", [(nodeID,networkID,sensorID)])

# Insert measurement values to DB
cursor.executemany("INSERT INTO ReadValue \
(MeasurementTime, HeartRate, RespirationRate, RelativeStrokeVolume, \
HeartRateVariability, MeasuredSignalStrength, \
Status, B2B, B2B1, B2B2, NodeID) \
VALUES( %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)", \
[(MT, HR, RR, RSV, HRV, MSS, Status, B2B, B2B1, B2B2, nodeID)])

cursor.execute('''CREATE TABLE IF NOT EXISTS sensor_data.ReadValue (
    MeasurementTime INT,
    HeartRate INT,
    RespirationRate INT,
    RelativeStrokeVolume INT,
    HeartRateVariability INT,
    MeasuredSignalStrength INT,
    Status INT,
    B2B INT,
    B2B1 INT,
    B2B2 INT,
    NodeID char(30),
    FOREIGN KEY (NodeID) REFERENCES SensorInfo(node_id)
''')

```

Listing 5. Code snippet that stores parsed data to database

From a single post, information that is needed to be stored both on the *ReadValue* and *SensorInfo* is parsed and stored.

4.5 Implementation of Client Web Application

Once the data is processed and stored on a database, it needs to be retrieved, processed and be displayed in a way the intended user could understand. For that purpose a simple Graphical User Interface (GUI) was designed. A Graphical User Interface is a human-computer interface which allows users to interact with an application hiding the command carried out behind the human understand interface.

The user interface for the application was implemented using HTML5, jQuery, flot.JS and CSS3. It enables users to select and visualize the data graphically for specific sensor user and time interval.

4.5.1 Transmission of Data to Client

The client application receives the sensor data from the database as a JSON (Javascript Object Notation) format. JSON is a file format that stores information in a human readable format. A flask application was developed in order to fetch the MySQL data into JSON data. The steps that were followed in order to get the MySQL data to JSON format is discussed here.

1. A connection to the database was established.
2. All the data from a ReadValue table were fetched by MySQL query.
3. Each fetched data were assigned to an array of object list and dumped to a json
4. Request handler was created to provide the JSON data as an API using Flask which is a framework that provides libraries, tools and technologies for building web application.

Listing 6 below is the code for excuting the JSON data from the stored MySQL data.

```

#instance created from Flask class
app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

#database connection established
con=MySQLdb.connect(host="localhost", user="root", passwd="h
cur=con.cursor()

#fetch the data from the table
cur.execute("SELECT * FROM ReadValue")
rows = cur.fetchall()

# data listed in array
objects_list = []

# assign name for each row
for row in rows:
    d = collections.OrderedDict()
    d['HeartRate'] = row[0]
    d['RespirationRate'] = row[1]
    d['RelativeStrokVolume'] = row[2]
    d['HeartRateVariability'] = row[3]
    d['MeasuredSignalVolume'] = row[4]
    d['Status'] = row[5]
    d['B'] = row[6]
    d['B1'] = row[7]
    d['B2'] = row[8]
    d['MeasurementTime'] = row[9]
    d['SensorID'] = row[10]
    objects_list.append(d)

# dump to json
j = json.dumps(objects_list, sort_keys=True, indent=2)

@app.route('/data', methods=['GET'])
def get_tasks():
    return j
if __name__ == '__main__':
    app.run(debug=True)

```

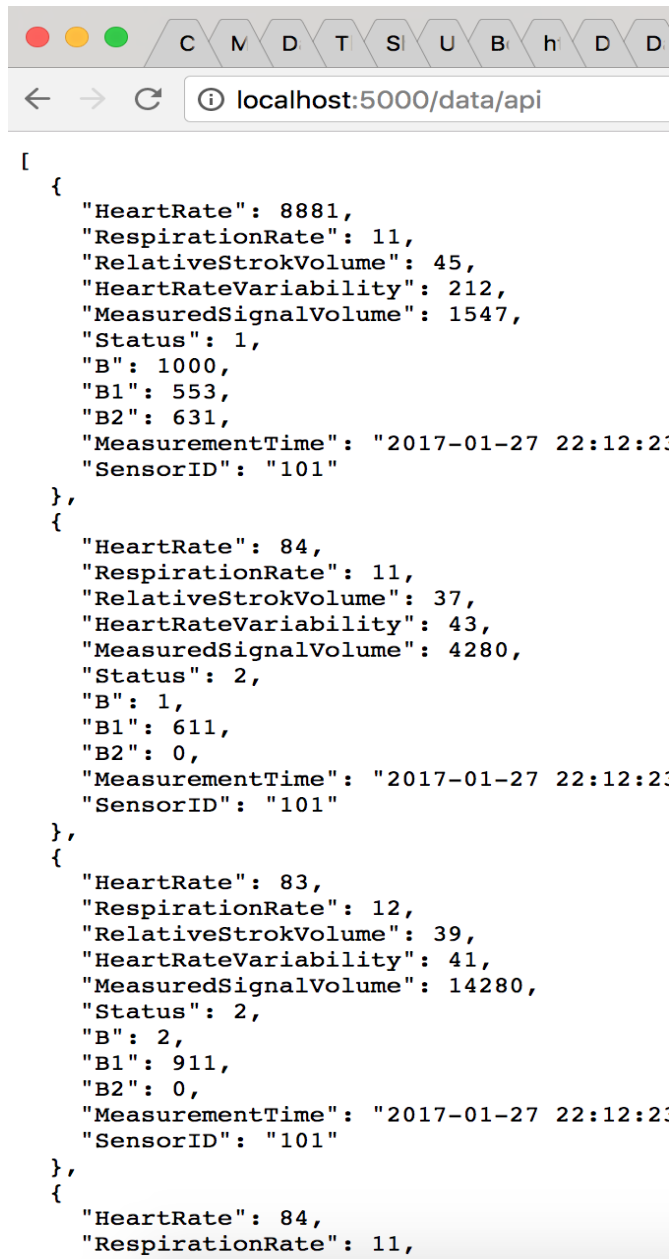
Listing 6. Flask API Application

As it is shown in Listing 6, the JSON file is generated and become accessible for the client application through an HTTP request.

4.5.2 Creating Chart

A line chart was used for plotting the stored sensor data in a time series graph. The JavaScript libraries used for creating the chart were Flot.js, Moment.js and jQuery.

When the HTML page gets loaded for the first time, Flot.js maps the data that comes as a JSON in a graph. As it was described in the previous section, the json file can be accessed by sending an HTTP GET request to the provided URL. Figure 8 is a sample data retrieved by sending request to the Flask application



```
[
  {
    "HeartRate": 8881,
    "RespirationRate": 11,
    "RelativeStrokVolume": 45,
    "HeartRateVariability": 212,
    "MeasuredSignalVolume": 1547,
    "Status": 1,
    "B": 1000,
    "B1": 553,
    "B2": 631,
    "MeasurementTime": "2017-01-27 22:12:20",
    "SensorID": "101"
  },
  {
    "HeartRate": 84,
    "RespirationRate": 11,
    "RelativeStrokVolume": 37,
    "HeartRateVariability": 43,
    "MeasuredSignalVolume": 4280,
    "Status": 2,
    "B": 1,
    "B1": 611,
    "B2": 0,
    "MeasurementTime": "2017-01-27 22:12:20",
    "SensorID": "101"
  },
  {
    "HeartRate": 83,
    "RespirationRate": 12,
    "RelativeStrokVolume": 39,
    "HeartRateVariability": 41,
    "MeasuredSignalVolume": 14280,
    "Status": 2,
    "B": 2,
    "B1": 911,
    "B2": 0,
    "MeasurementTime": "2017-01-27 22:12:20",
    "SensorID": "101"
  },
  {
    "HeartRate": 84,
    "RespirationRate": 11,
```

Figure 8. Sample JSON Data

An AJAX call was used to perform the process of getting the data. After the JSON data was accessed with the requested url, mapping of the data to a time series chart was performed. In the HTML5 file, user input were set to send POST data which filters the kind of data to be displayed. Based on the selection of the user input the JSON data were filtered to display the data on the chart.

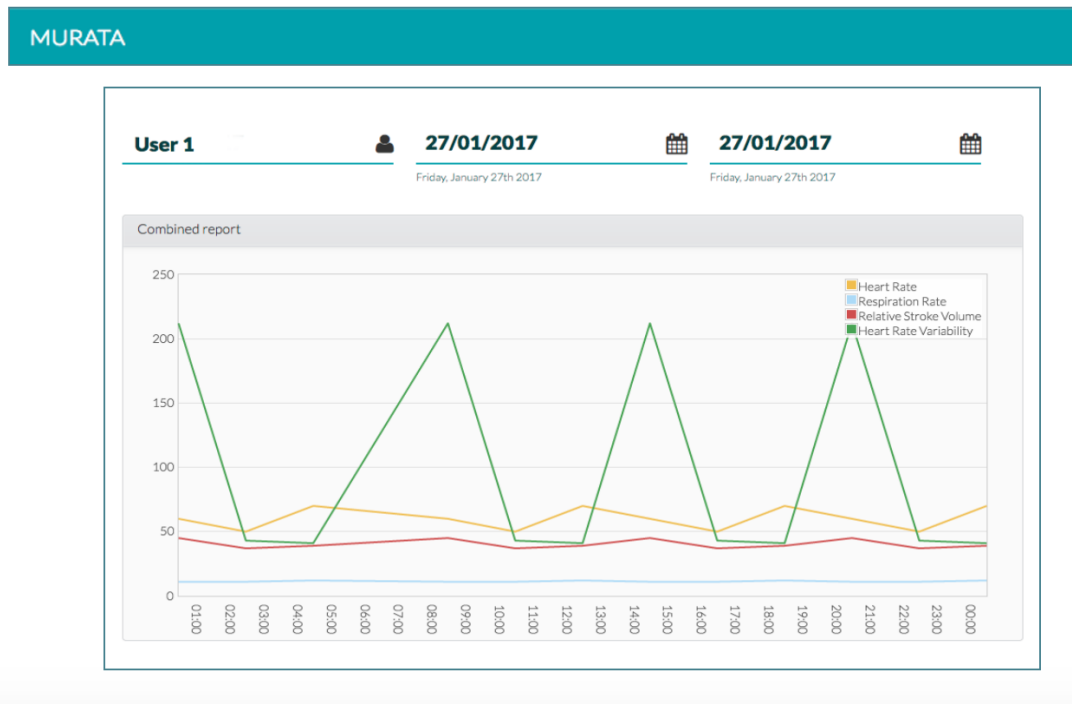


Figure 8. Combined 24 hour chart

As shown in Figure 8, basic usage of the data received from the sensor can be achieved by displaying the data collected in a time chart. Each data is represented by different colors and the value of the corresponding color is also labeled in the chart.

The data from the backend was retrieved by a query which selects the contents of sensor data such as measurement time, node id, heart rate, respiration rate and relative stroke volume. After the selection is made, the data is serialized to JSON (JavaScript Object Notation) so that the chart could be generated to display a combined report.

5 Results and Evaluation

As it was mentioned in Section 1.2 the goal of the project was to develop an application that collects stores and visualizes information from BSN sensor used for health monitoring. In this section, the result and the evaluation on how the goal was completed are described.

After the BSN configuration and backend implementation were completed, it was tested by connecting the sensor to the application web server. The actual measurements that were generated by the sensors were received and stored in the database being identified by a unique node id. Even if the clinical ways of representation were not followed, the data were displayed on a composite chart diagram. The chart was a time based chart to represent the data as time series. As shown in Figure 8, user selection and then the representation of measurement data of the selected user is shown in a single page. From usability point of view, it is a straightforward process to select and see the resulting representation.

6 Conclusion

The main focus of this project was to build a prototype web application that collects and visualizes data from health monitoring sensor in a cloud computing infrastructure. The application developed was able to store the sensor data in MySQL database and visualize this data in a line chart for end users.

In general, three steps were carried out for implementing the backend and the client side of the application. The first step was configuring the sensor in order to send the information to the intended application. Next to this, developing the backend application to handle data storage and retrieving requests was performed. Technologies that were used in developing the backend application were MySQL and Python scripting language. The last phase was designing a GUI that processes, retrieves and displays the stored data. For the implementation of the GUI, web technologies such as HTML5, jQuery, Flot.JS and CSS3 were used.

For the future, additional features could be added to the application in order to increase the overall functionality. Such features could be user administration where the authentication and access control of users could be implemented with an in depth architectural design. Since the data processed is medical data, encryption of data is also another feature that needs to be added to attain data security. Moreover, different kinds of charts can be added to the client application to allow better analysis by the end users.

References

[Antonopoulos, N. & Gillam, L., 2010. Cloud Computing: Principles, Systems and Applications, Springer Science & Business Media.](#)

Fry, B., 2008. Visualizing Data exploring and explaining data with the processing environment, Sebastopol (Ca): O'Reilly.

Brown, E. and Swenson, G. (2016) Final version of NIST cloud computing definition published. Available at: <https://www.nist.gov/news-events/news/2011/10/final-version-nist-cloud-computing-definition-published> (Accessed: 14 October 2016).

Chipara, Octav; Lu, Chenyang; Bailey, Thomas C.; and Roman, Gruia-Catalin, "Reliable Patient Monitoring: A Clinical Study in a Step-down Hospital Unit" Report Number: wucse-2009-82 (2009). All Computer Science and Engineering Research. http://openscholarship.wustl.edu/cse_research/33

Co, M.M. (2017) Murata manufacturing Co., Ltd. Available at: http://www.murata.com/en-us/products/sensor/accel/sca10h_11h/sca11h (Accessed: 4 February 2017).

Taowei D.W, Plaisant C., A. J. Quinn, R. Stanchak, S. Murphy, and B.(2008) Shneiderman. Aligning temporal data by sentinel events: Discovering patterns in electronic health records. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems

[Dupont, S. et al., 2016. Bringing Dynamics to IoT Services with Cloud and Semantic Technologies - An Innovative Approach for Enhancing IoT based Services. In Proceedings of the International Conference on Internet of Things and Big Data. Available at: http://dx.doi.org/10.5220/0005933001850190.](#)

Fortino, G. and Pathan, M. (2014) 'Integration of cloud computing and body sensor networks', Future Generation Computer Systems, 35, pp. 57–61. doi: 10.1016/j.future.2014.02.001.

Fry, B, 2007. Visualizing Data: Exploring and Explaining Data with the Processing Environment. 1. O'Reilly Media.

Gartner (2013) Gartner says the Internet of things installed base will grow to 26 Billion units by 2020. Available at: <http://www.gartner.com/newsroom/id/2636073> (Accessed: 3 June 2016).

Google App Engine Documentation 2017 App Engine tion[ONLINE] Google Cloud Platform. Google App Engine Documentation , App Engine Documentation , Google Cloud Platform. Available at: <https://cloud.google.com/appengine/docs/> . [Accessed 9 March 2017].

Journal, R. (2002) RFID (radio frequency identification) technology news & features. Available at: <http://www.rfidjournal.com> (Accessed: 23 September 2016).

Kavis, M.J. (2014) Architecting the cloud: Design decisions for cloud computing service models (SaaS, PaaS, and IaaS). United States: John Wiley & Sons.

[Kuor-Hsin Chang & Kuor-Hsin, C., 2014. Bluetooth: a viable solution for IoT? \[Industry Perspectives\]. IEEE Wireless Communications, 21\(6\), pp.6–7.](#)

Lorincz, K., Malan, D.J., Fulford-Jones, T.R.F., Nawoj, A., Clavel, A., Shnayder, V., Mainland, G., Welsh, M. and Moulton, S. (2004) 'Sensor networks for emergency response: Challenges and opportunities', IEEE Pervasive Computing, 3(4), pp. 16–23. doi: 10.1109/mprv.2004.18.

Verdone, R., Dardari, D. and Mazzini, G. (2008) Wireless sensor and actuator networks: Technologies, analysis and design. Amsterdam: St Louis, Missouri, U.S.A.: Academic Pr.

Shvartsman, A.A., 2014. Distributed storage, San Rafael: Morgan & Claypool.

Song, Q. and Xu, J. (2013) Study and Implementation of Patient Data Collection and Presentation for an eHealth Application. Available at: <http://www.diva-portal.org/smash/record.jsf?pid=diva2:651456>.

Wood, A., Stankovic, J., Virone, G., Selavo, L., He, Z., Cao, Q., Doan, T., Wu, Y., Fang, L. and Stoleru, R. (2008) 'Context-aware wireless sensor networks for assisted living and residential monitoring', IEEE Network, 22(4), pp. 26–33. doi: 10.1109/mnet.2008.4579768