TRANSFORMING MOBILE DEVICES INTO SMART IOT GATEWAYS

M. A.V. Nanayakkara

(158228P)

Degree of Master of Science in Computer Science

Department of Computer Science and Engineering

University of Moratuwa
Sri Lanka

April 2019

TRANSFORMING MOBILE DEVICES INTO SMART IOT GATEWAYS

M. A.V. Nanayakkara

(158228P)

Thesis submitted in partial fulfillment of the requirements for the degree
Master of Science in Computer Science

Department of Computer Science and Engineering

University of Moratuwa
Sri Lanka

April 2019

DECLARATION

I declare that this is my own work and this dissertation does not incorporate without acknowledgment any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.
Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: …………………….   Date: ……………….
Name: M. A. V. Nanayakkara

The supervisor/s should certify the dissertation with the following declaration.

The above candidate has carried out research for the Masters Dissertation under my supervision.

Signature of the supervisor: ………………………….   Date: ……………….
Name: Dr. Indika Perera

ACKNOWLEDGMENTS

First and foremost, I am grateful to my supervisor Dr. Indika Perera for the guidance and support provided to complete the research successfully. His continuous guidance and support contributed a lot in completing this research.

There is no doubt that the elite academic staff of University of Moratuwa successfully managed to improve our character and competencies which refined our professionalism and made us achievers in our career professions.

My sincere appreciation goes to all the faculty members and academic/non-academic staff members of University of Moratuwa who have assisted me in various ways.

The course materials & other relevant facilities provided by the University of Moratuwa also helped me throughout the journey of MSc Programme.

Last but not the least, special gratitude goes to my family for giving me the unconditional support & strength to make this journey a great success.

ABSTRACT

Internet of Things (IOT) which is considered as the ‘Next Industrial Revolution’ is a heavily discussed topic across the globe which is all about inter-connection of *things* for the betterment of the human beings. Apart from the end nodes which are either sensors or actuators, a reliable and robust connection is the most significant component of any IOT deployment. Therefore, different concepts of IOT gateways have been introduced by many IOT vendors to the industry to cater various types of connectivity options. However, still there are major challenges and weaknesses prevailing in the industry, often in the connectivity component unanswered which have limited the full potential of IOT deployments.

This research is focused on prototyping a smarter IOT gateway using an Android based mobile device which can overcome some of these limitations identified. nRF51822 Bluetooth based sensor tag which gives five different sensor readings and a cloud instance of Splunk IOT Data Analytics platform were used along with the Prototype gateway to demonstrate an actual IOT deployment in this research.

The prototype successfully managed to push processed and filtered sensor data into the Splunk cloud which significantly reduced the data load which saved cloud storage. A python-based application was used to process the raw sensor data to meaningful information. In addition to that this prototype added value to the IOT gateways, by having features such as real-time alerting leveraging on Google Cloud Messaging services.

However, using Android based mobile devices as IOT gateways created new set of limitations where few recommendations such as developing hardware-addons to accept other wireless IOT connectivity technologies and customizing the Android OS etc. were proposed towards the end of this research.

Keywords: IOT Gateways, Data Analytics and Visualizations, Data Translation, Real-time alerting, Bluetooth Sensor Tags

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

ALS	Ambient Light Sensor
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
BLE	Bluetooth Low Energy
CCM	Central control, Context detection & Management
COAP	Constrained Application Protocol
DA	Device Abstraction
GATT	Generic Attribute Profile
GCM	Google Cloud Messaging
HTTP	Hyper Text Transfer Protocol
IaaS	Infrastructure as a Service
IIOT	Industrial Internet of Things
IOT	Internet of Things
IR	Infra-Red
IT	Information Technology
JMS	Java Message Service
LAN	Local Area Network
LoRa	Long Range
MQTT	MQ Telemetry Transport
NFC	Near Field Communication
PaaS	Platform as a Service
PS	Proximity Sensor
REST	Representational State Transfer
RFID	Radio-Frequency IDentification
SaaS	Software as a Service
TCP	Transmission Control Protocol
UUID	Universally Unique Identifier
Wi-Fi	Wireless Fidelity

XMPP	Extensible Messaging and Presence Protocol

LIST OF FIGURES

LIST OF TABLES

Chapter 1

1. INTRODUCTION

1.1 Background

IOT (Internet of Things) refers to *The Networking of Things* in a given environment in the most effective manner. These *Things* include huge arrays of sensors, actuators and smart devices which are not just limited to electronic devices but also the general objects people use every day. Thus, effective communication between the internet and the *Things* has become a huge challenge.

Mostly these sensors are used either to collect environmental data such as atmospheric temperature, humidity and weather data or to obtain data regarding human body. John Walker once stated in his site that, "*The shift in the* ***Internet of Things and connected devices*** *is affecting every major industry whether it's industrial, commercial or residential, these connected devices represent a new channel through which data flows in both directions providing invaluable data to all sectors of business.".* Further he emphasized that, "*This data is crucial for any business as they can now analyze data in real-time and make decisions that will change the operations to be not only more profitable but create more streamlined, and efficient workflows.".*

The massive chunks of gathered data are being sent to a cloud hosted platform to be analyzed and to obtain advanced analytics which will be used in critical decision making. However, in most situations, the data gathered is not manipulated or filtered properly which is unnecessarily consuming network resources. Therefore, *IOT Middleware* platforms play a great role standing in between the end-devices and the internet, utilizing the network resources. Mohan [15], states that, "*Middleware for IOT acts as a bond joining the mixed domains of applications communicating over heterogeneous interfaces.*". Further middleware helps to mitigate the challenges that come across when integrating the end-devices with the software platforms as they are developed by different vendors using different protocols. Blouin [16], in her blog discussed on a new creative mode of IOT middleware where she has stated that, "*The mobile device is really the clear choice for our interactions with sensors and devices. Smart phones in general have become a large part of our daily life. We communicate with them, do work on them, use them to navigate and pass free time by playing games on them. By using them as our gateway to the Internet of Things we can add more context into all of those activities.*".

Therefore, this research would be focused on developing an IOT gateway in a mobile phone to collect data from sensors and push them to relevant IOT analytics platforms whilst providing important alerts to the user from the gateway itself in real-time.

1.2 Motivation

Many researches have been done in the past to find out the challenges in existing IOT Gateways and then how to overcome those. However, it is believed that at least some of these challenges listed below can be overcome by using an intelligent IOT gateway.

- *Interoperability:* A gateway which can accept data via different communication technologies such as Wi-Fi, NFC, BLE, 3G and 4G with different data protocols such as MQTT, COAP, TCP and HTTP and send them to different cloud platforms can help to overcome this challenge to a certain extent.
- *Software-complexity:* A gateway which can process extensive software applications locally in-order-to process and filter the data coming from different sensor nodes with different software systems will help to manage these nodes more productively.
- *Data volumes:* A gateway which can handle large volumes of data locally rather than just being a data pipe, will put less effort on raw data processing in IOT platforms which will improve the data handling capacity of them.
- *Security and privacy:* A gateway which can receive and send encrypted sensor data will improve the security of the data, as the data becomes more vulnerable in the communication in between the platform and the gateway.
- *Fault tolerance:* A gateway which can function properly under different environmental conditions, sometimes in harsh environments will ensure smooth data transmission from the end nodes to the cloud
- *Power Supply:* A gateway with optimized power consumption techniques which consumes minimal power to receive, process and transmit IOT data can be a big advantage most of the remote IOT deployments with poor infrastructure conditions.
- *Cost*: A quality gateway with a reasonable price point can help in more IOT deployments since the major portion of the cost of a typical IOT deployment is due to the hardware involved.

Very limited studies have been carried out in using a mobile phone as an IOT gateway even though it carries most of the key features of a traditional IOT gateway such as basic routing capabilities, accepting multiple data formats via multiple communication protocols, ability to connect with multiple end devices/ sensors, higher data processing, efficient power consumption and advanced programmable functionalities etc.

John Walker once discussed on leveraging on a mobile phone as an IOT Gateway. He stated that, "*Smartphone will continue to play an important role in IOT particular where there is a need to collect data around human interaction and sensor information from IOT. Smartphones that are equipped with Wifi, Bluetooth and NFC will continue to communicate and control applications and many different use cases for IOT. The mobile devices will eventually be the primary user interface for applications as they provide real-time data to be manipulated on site, which will increase efficiency and consumer relationships.*" [11].

Therefore, it is very encouraging to utilize a smart phone as an IOT Gateway to obtain raw data from end sensors/ actuators, process the data into information, build intelligence into that and push real-time alerts and notifications.

1.3 Objectives of the research

Main objective of the research is to prototype a mobile phone-based application which acts as an IOT gateway mitigating some of the challenges faced by traditional IOT gateways.

An Android based smartphone and nRF51822 Bluetooth Low-Energy module connected with multiple sensor nodes to shall be used to prototype the solution in the proposed research. The mobile phone will act as the IOT gateway which shall collect raw data from the BLE module via Bluetooth connectivity and process them for other applications.

The challenges that are expected to overcome are,

- *Data Volumes:* The proposed prototyped application will be processing and filtering the raw data collected leveraging on the processing power of smart phones, without just pushing them to the cloud which will make the cloud deployments easy and cost-effective.

- *Power Supply:* Smart phones have optimally utilized the power consumption in processing and transmitting data which would made them an ideal IOT gateway in remote deployments where continuous power source is scarce.
- *Cost:* The cost of mobile phones is continuously decreasing due to mass production which leads to economies of scale whereas the traditional IOT gateways remain to be expensive as still they don’t benefit on economies of scale.

1.4 Thesis Organization

The chapter framework of the research is as follows.

Chapter 1: Introduction

An introduction to the research is provided in this chapter along with the motivation to carry-out the research. The objective of the research highlights the necessity and importance as to why it is an interesting research. Further it briefly defines the scope of the prototype development.

Chapter 2: Literature Review

Chapter 2 begins with an introduction to Internet of Things (IOT) which is followed by the in-depth literature reviews on each key component of IOT Concept such as Sensors, Gateways and Platforms. The existing literature found provides distinctive definitions on IOT gateways and cloud platforms which are analyzed and correctly taken according to the requirement of the research.

Chapter 3: Research Methodology

An overview to the chapter is given at the beginning followed by the solution architecture which is thoroughly explained. Then the methodology of collecting raw data from sensors are explained in detail and the methodology of data filtering, analyzing in the IOT Gateway is described as well.

Chapter 4: Research Implementation

This chapter mainly explains the key hardware components used to develop the solution and how those components were put together to achieve the expected end-result. This is followed by the related software components developed and their functionality.

Chapter 5: Research Evaluation

This chapter starts with an introduction, compares the prototype functionalities against a traditional IOT gateway functionalities and discusses how the challenges identified in the research objective can be overcome.

Chapter 6: Discussion

A comprehensive discussion on the new findings and challenges that were faced during the research implementation and possible improvements that can be done to overcome the challenges.

Chapter 7: Conclusion

Finally, this chapter concludes the research by briefing on the challenges identified and how the prototype helped to overcome them. Further this briefly recommends on how to proceed forward from the prototype development.

Chapter 2

2. LITERATURE REVIEW

2.1 Introduction

Smart phones are embedded with immense array of sensors, from accelerometer which detects acceleration, tilt and vibration to determine the movement and orientation, gyroscope, light sensors, proximity sensors, finger print sensor, infrared sensor and much more [8]. These sensors can output responses to a particular input from the physical environment. Different sensors can monitor different parameters and most commonly measured parameters are temperature, humidity, pressure etc. Output can be varying and huge. And, data can be in structured or unstructured forms. When sensor devices provide high load of information analyzing them is the next big challenge for businesses [9]. Smart phones can exclusively play a major role in IOT where collecting and communication of information can done with it. Companies looking forward to develop mobile applications specifically on IOT and data communications [9]. Smart phones are enabled with technologies such as Wi-Fi, Blue-Tooth, NFC which help to work in different areas of IOT. For example, as stated by Hausenblas [10], there are some areas that smart phones are already been used.

- In the *Personal IOT* we see an increasing number of offerings around fitness and health. Then, there are a number of gadgets aiming to address everyday's tasks and problems, like NFC rings that can unlock your phone.
- Applications in the *Group IOT* area benefit from smart phones in a number of ways: for example, in the context of the connected car, to control or check the system status. The same is true for smart homes.
- For the *Community IOT,* smart phones play a vital role as well: for example, citizens can contribute to a smart city via crowd sourcing applications [10].

It is witnessed that mobile devices play important role in communicating and analyzing IOT data which help the industries to move to a new step in business world. As stated by Walker [11], "*As a part of the IOT ecosystem, mobile devices play an important part in streamlining the collection, management and use of the data generated. The early adopters are seeing that by taking advantage of already existing technology like the smartphone is* *expediting their IOT initiatives and helping to bring their solutions to market faster.”*. Moreover, author goes on to say that “*The mobile devices will eventually be the primary user interface for applications as they provide real-time data to be manipulated on site, which will increase efficiency and consumer relationships”* [11].

There are three key components of a typical IOT deployment as shown in the figure 2.1 below.

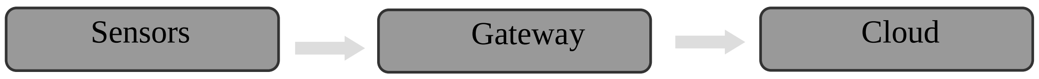


Figure 2.1: Components of IOT Deployment

Section 2.2 will give a basic introduction to IOT, its definition, architecture and IOT communication technologies. Section 2.3 and 2.4 will cover up the currently existing sensors and the gateways. And in the sub section 2.4.1 will discuss about the prior works done on developing gateways. Section 2.5 will discuss the cloud platforms that are working with IOT.

2.2 Internet of Things (IOT)

Gartner [1], defines Internet of Things (IOT) as “the network of physical objects that contain embedded technology to communicate and sense or interact with their internal states or the external environment”.

PrismTech [2], in their paper highlights the idea that “IOT conceptually refers to the general things, especially everyday objects that are readable, recognizable, locatable, addressable, and/or controllable via the Internet; whether via RFID, wireless LAN, wide-area network, or other means”. Further it emphasizes on the idea that “everyday objects mean not only the electronic devices but also the things that are not electronic at all such as food, clothing, shelter, materials, parts and etc”.

Bandyopadhyay et al. [3], expressed that “Internet of Things (IOT) is a combined part of future Internet and ubiquitous computing. It demands interactions with the heterogeneous raw sensors, aggregators, actuators and diverse domain of context aware applications, preserving the security and privacy”.

Furthermore, they define the term IOT. As mention in the paper,” It *comprises two definite components Internet and things. Internet is a global network infrastructure with self-configuring, scalable, dynamic expansion capabilities based on standard and* *interoperable communication protocols whereas "things" are physical objects/devices or virtual- objects/devices/information having identities, physical attributes, and virtual personalities and use intelligent interfaces. "Things" are heterogeneous in nature and seamlessly integrated into the information network*". Then comes the messaging technologies in to the play. Sensor devices need a way to communicate with the other parties which are participating in the event. For this purpose, different messaging technologies and protocols are used.

AMQP, MQTT, JMS, REST and XMPP were all designed to run on networks that use TCP/IP as the underlying transport. AMQP, MQTT and JMS support brokered publish-and-subscribe message exchanges between device nodes [2]. This heterogeneous network connects a range of devices together. It encompasses devices which connect to the Internet using various types of wireless, mobile and LAN technologies such as Wi-Fi, ZigBee, Bluetooth, and 3G or 4G technologies among other evolving communication technologies [12].

IOT should have a special architecture to work with different devices and communicate among them. IOT architecture consist of five layers which makes the IOT layered architecture [4]. Figure 2.2 shows the generic five layered architecture of IOT which are discussed in the following.

- Edge Technology layer

This is a hardware layer that consists of embedded systems, RFID tags, sensor networks and all the other sensors in different forms. This hardware layer can perform several functions, such as collecting information from a system or an environment, processing information and supporting communication [4].

- Access Gateway layer

This layer is concerned with data handling and is responsible for publishing and subscribing the data [4].

- Middleware layer

This layer has some critical functionalities, such as aggregating and filtering the received data from the hardware devices, performing information discovery and providing access control to the devices [4].

- Application layer

This layer is responsible for delivering various application services. These services are provided through the middleware layer to different applications and users in IOT-based systems.

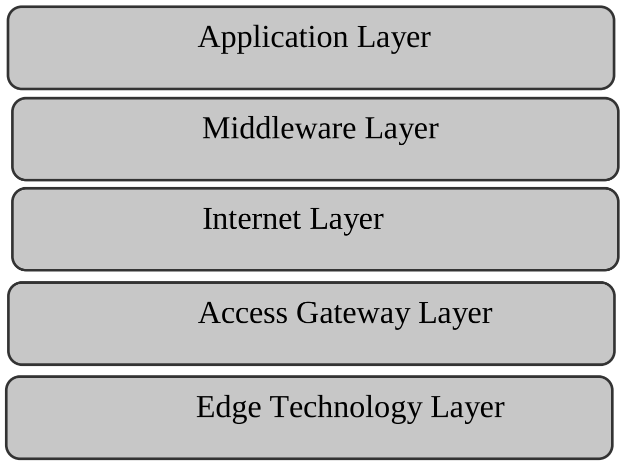


Figure 2.2: IOT Layered Architecture
(Source: http://fmt.cs.utwente.nl/files/sprojects/235.pdf)

2.3 Sensors

There are different types of sensors available in different fields such as agriculture, retail and health care etc. These sensors can measure various parameters, temperature, humidity, proximity, Pressure, water quality etc.

Finoit Technologies [13], on their site discusses that industries and organizations have been using various kind of sensors for a long time but the invent of the Internet of things has taken role of sensors and evolutions of sensors to a completely different level. IOT platforms function and deliver valorous kind of intelligence and data using a variety of sensors. They serve to collect data, pushing it and sharing it with a whole network of connected devices. All this collected data makes it possible for devices to autonomously function, and the whole ecosystem is becoming “smarter” every day.

Smart phones can be used as IOT gateways as they are embedded with huge array of sensors. Figure 2.3 shows some of the sensors available in smart phones.

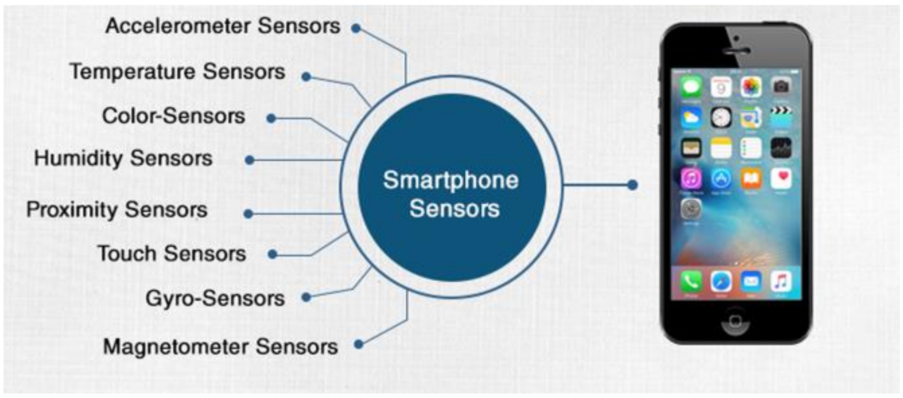


Figure 2.3: Smartphone Sensors
(Source: http://www.vensi.com/smartphones-with-sensor-technology-enhances-our-daily-operations/)

2.4 Gateways

The IOT changes the way that we understand the world, using sensors to continuously monitor the environment around us, providing more information about traffic, weather, health, fleet management, vehicle control, allowing the Information Systems to provide value-added information for every single person [5].

The adoption of middleware helps to avoid some common problems in IOT development, such as:

- Hides the heterogeneity of hardware components, operation systems and communication protocols
- Interconnects parts running in distributed locations
- Provides uniformly high level of standard interfaces for developers and application integrators, making these applications easy to build, reuse and inter- operate
- Provides a set of common services to perform various general-purpose functions, avoiding repeated efforts of the developing team [5].

Moreover, author explains the functional behavior of a Middleware. Gateway is a layer or set of software sub-layers interposed between levels of operational and communicative application. The middleware has several features, the primary being to hide details from different technologies, protocols, network environments, data replication, and parallelism. Another feature of middleware is to exempt the programmer from issues that are not directly linked to final application, because middleware masks the heterogeneity of computer architectures, operating systems, programming languages, and network technologies [5].

The basic goal of the gateway is to settle the heterogeneity between different end-point networks and the Internet, strengthen the management of the endpoint networks and bridge the traditional Internet with endpoint networks [14].

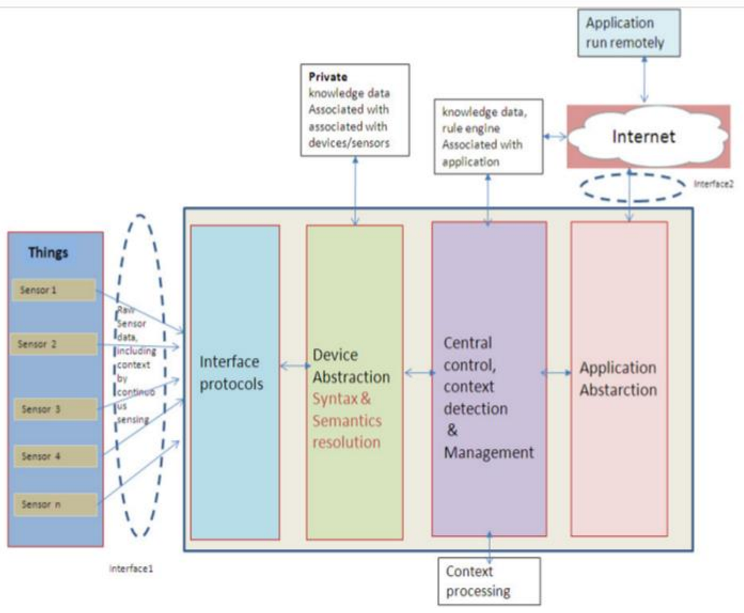

Application run remotely
Private knowledge data Associated with associated with devices/sensors
knowledge data, rule engine Associated with application
Internet
Interface2
Things
Sensor 1
Sensor 2
Sensor 3
Sensor 4
Sensor n
Raw Sensor data, including context by continuous sensing
Interface protocols
Device Abstraction Syntax & Semantics resolution
Central control, context detection & Management
Application Abstarction
Interface1
Context processing

Figure 2.4: Functional Components of IOT Middleware
(Source: http://fmt.cs.utwente.nl/files/sprojects/235.pdf)

2.4.1 *Interface protocols*

This component oversees providing technical interoperability. Interoperability in the context of Interface protocols means: the ability of two systems to interoperate by using the same communication protocols. This wrapper can be placed either on the device side or the middleware side. If we want to have direct interaction with devices, we should place the wrapper in the middleware side [4].

2.4.2 *Device Abstraction (DA)*

This component is responsible for providing an abstract format to facilitate the interaction of the application components with devices. This abstraction provides syntactic and semantic interoperability,

Syntactic interoperability is associated with data formats. Semantic interoperability is usually associated with the meaning of the content of which is understandable for human [4].

2.4.3 *Central control, Context detection & Management (CCM)*

Context characterizes the situation of an entity, which can be a place, a person or an object that is relevant to the user, applications and their interactions. A middleware for

IOT-based systems must be context-aware to work in smart environments. Smart environments refer to a physical world that is richly and invisibly interwoven with sensors, actuators, displays, and computational elements, embedded seamlessly in the everyday objects of our lives, and connected through a continuous network [4].

2.4.4 Application Abstraction

This functional component provides an interface for both high-level applications and end users to interact with devices. For instance, this interface can be a RESTful interface or can be implemented with some query-based language [4].

2.4.5 Data Security

Considering the data security in IOT middleware, five major security violation categories are identified such as breach of confidentiality" when "an unauthorized person reads and gets access to the data" [27], "breach of integrity" when "the attacker reads and modifies the data" [27], "breach of availability" when "the attacker destroys and deletes the data" [27], "denial of service" when "the attacker attacks the limited resources and "*theft of services"* when "the resources are used by an unauthorized person"[27].

However, smartphones today are equipped with advanced data encryption technologies considering the data being stored, networking technologies where only the preferred nodes can be connected such as Bluetooth device paring features and lastly remote wiping features in case of a data breach [28].

2.5 Real-world IOT Systems

2.5.1 Libelium

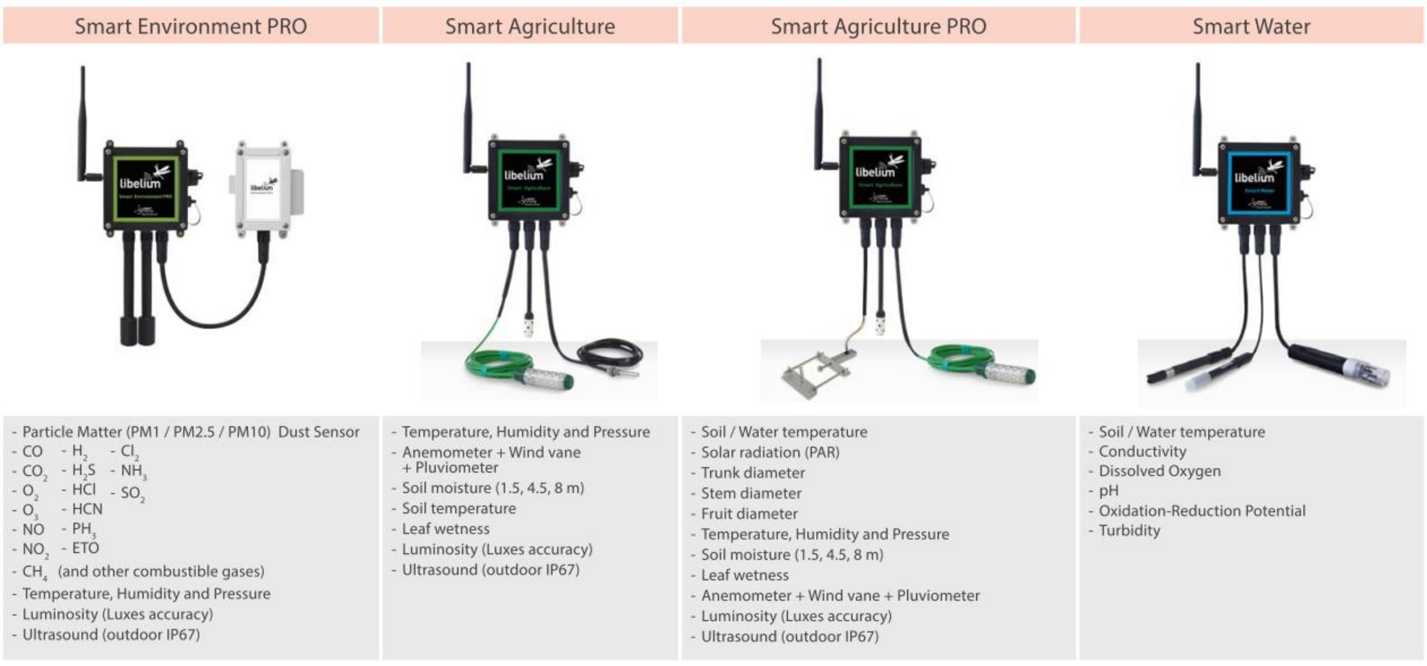
Libelium is one of the top global IOT hardware solution providers who owns broadest range of IOT sensors which is more than 160 covering wide range of different use cases and applications. Further, Libelium designs and manufactures fully integrated products for wireless sensor networks that enable companies to design and implement reliable Internet of Things (IoT), M2M on any Wireless Sensor Network, including the following Smart solutions [25].

- *Smart City Solutions:* Noise level monitoring, waste management, smart lighting and smart roads etc.
- *Smart Parking Solutions:* Comprehensive parking management solutions.
- *Smart Environment Solutions:* Monitoring environmental parameters such as temperature, humidity, atmospheric pressure and types of gases for city pollution

measurement, emissions from farms and hatcheries, control of chemical and industrial processes, forest fires, etc.

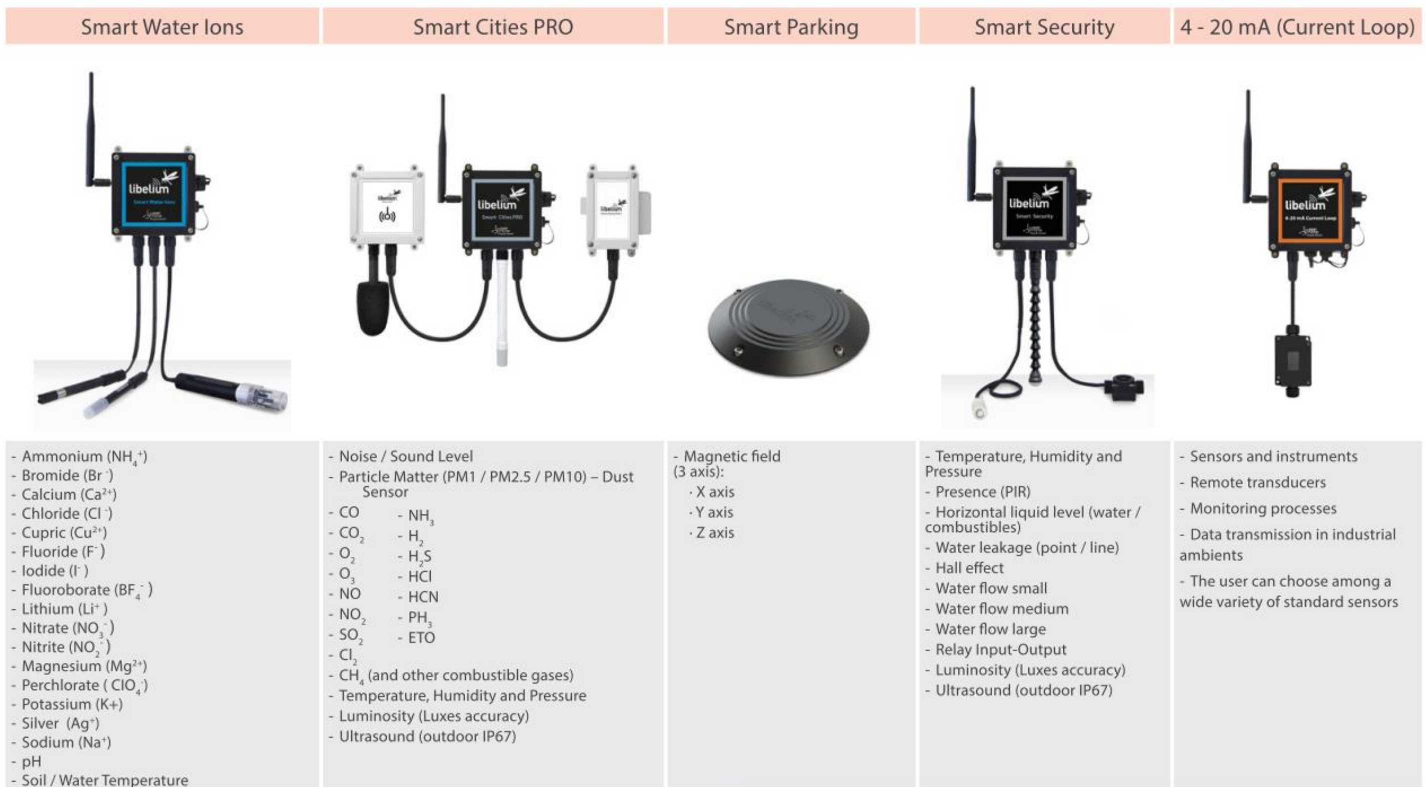
- *Smart Agriculture Solutions:* Vineyards monitoring for enhancing wine quality, selective irrigation on golf courses and conditions control in greenhouses etc.
- *Smart Water:* Water leakage monitoring, water pollution levels in seas, chemical leakage detection in rivers, river flooding etc.

are few of the smart solutions developed by Libelium.



Smart Environment PRO	Smart Agriculture	Smart Agriculture PRO	Smart Water
- Particle Matter (PM1 / PM2.5 / PM10) Dust Sensor - CO - H_2 - Cl_2 - CO_2 - H_2S - NH_3 - O_2 - HCl - SO_2 - O_3 - HCN - NO - PH_3 - NO_2 - ETO - CH_4 (and other combustible gases) - Temperature, Humidity and Pressure - Luminosity (Luxes accuracy) - Ultrasound (outdoor IP67)	- Temperature, Humidity and Pressure - Anemometer + Wind vane + Pluviometer - Soil moisture (1.5, 4.5, 8 m) - Soil temperature - Leaf wetness - Luminosity (Luxes accuracy) - Ultrasound (outdoor IP67)	- Soil / Water temperature - Solar radiation (PAR) - Trunk diameter - Stem diameter - Fruit diameter - Temperature, Humidity and Pressure - Soil moisture (1.5, 4.5, 8 m) - Leaf wetness - Anemometer + Wind vane + Pluviometer - Luminosity (Luxes accuracy) - Ultrasound (outdoor IP67)	- Soil / Water temperature - Conductivity - Dissolved Oxygen - pH - Oxidation-Reduction Potential - Turbidity

Figure 2.5: Libelium IOT Products
(Source: http://libelium.com/)



Smart Water Ions	Smart Cities PRO	Smart Parking	Smart Security	4 - 20 mA (Current Loop)
- Ammonium (NH_4^+) - Bromide (Br^-) - Calcium (Ca^{2+}) - Chloride (Cl^-) - Cupric (Cu^{2+}) - Fluoride (F^-) - Iodide (I^-) - Fluoroborate (BF_4^-) - Lithium (Li^+) - Nitrate (NO_3^-) - Nitrite (NO_2^-) - Magnesium (Mg^{2+}) - Perchlorate (ClO_4^-) - Potassium (K+) - Silver (Ag^+) - Sodium (Na^+) - pH - Soil / Water Temperature	- Noise / Sound Level - Particle Matter (PM1 / PM2.5 / PM10) – Dust Sensor - CO - NH_3 - CO_2 - H_2 - O_2 - H_2S - O_3 - HCl - NO - HCN - NO_2 - PH_3 - SO_2 - ETO - Cl_2 - CH_4 (and other combustible gases) - Temperature, Humidity and Pressure - Luminosity (Luxes accuracy) - Ultrasound (outdoor IP67)	- Magnetic field (3 axis): · X axis · Y axis · Z axis	- Temperature, Humidity and Pressure - Presence (PIR) - Horizontal liquid level (water / combustibles) - Water leakage (point / line) - Hall effect - Water flow small - Water flow medium - Water flow large - Relay Input-Output - Luminosity (Luxes accuracy) - Ultrasound (outdoor IP67)	- Sensors and instruments - Remote transducers - Monitoring processes - Data transmission in industrial ambients - The user can choose among a wide variety of standard sensors

Figure 2.6: Libelium IOT Products
(Source: http://libelium.com/)

2.5.2 Libelium Sensor Nodes

Libelium follows a modular architecture where sensors can be plugged-in or out from the sensor controller based on the application or the use case. These sensor controllers are called ‘Plug and Sense’ which is a box shaped unit as shown in Figure 2.7, which can connect with a maximum of 6 sensors at a given time [25].

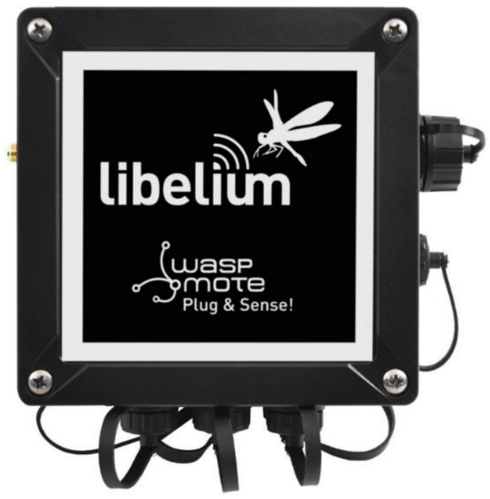


Figure 2.7: Libelium Sensor Node
(Source: http://libelium.com/)

The unit is designed to be robust with IP65 certified enclosures where 6 solid, water-tight sockets as shown in figure 2.8 are used to connect the sensors. The controller circuit along with the communication module which can be one of many protocols such as GPRS, 3G, 4G, Wi-Fi, LoRaWAN, SigFox and Low Power Wi-Fi etc., is fitted inside this robust enclosure to protect from harsh outdoor conditions.

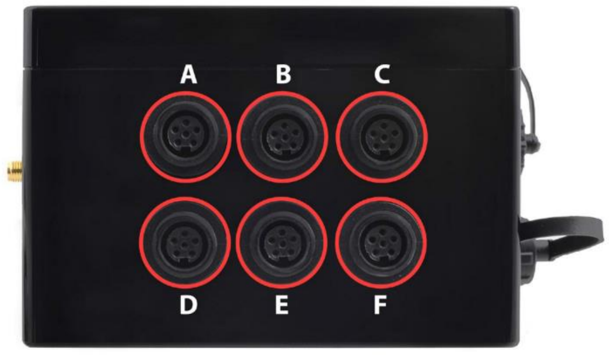


Figure 2.8: Libelium Sensor Sockets
(Source: http://libelium.com/)

2.5.3 *Cloud Integrations*

Libelium have developed plug-ins with most of the IOT cloud solution providers such that these frames can be easily decoded at the cloud end and make meaningful information [25].

Figure 2.9: Libelium Cloud Integrations
(Source: http://libelium.com/)

However, most of these plug-ins are developed in the Libelium Gateway which is called the "Meshlium", creating a bottleneck that all the sensor nodes need be connected to the Meshlium to have easy filtering of raw data. If not, the decoding has to be done at the cloud end which adds more complexities and increase costs.

2.5.4 *Meshlium Gateway*

Meshlium is a full-fledged gateway which is designed to accept data from Libelium sensor nodes via multiple connectivity protocols and push them to different IOT clouds and remote databases [25].

It is equipped with a robust IP65 certified enclosure which can withstand out door conditions, full-featured local database and many connectivity options as shown in the Figure 2.10 below which has made Meshlium one of the best IOT gateways in the market.

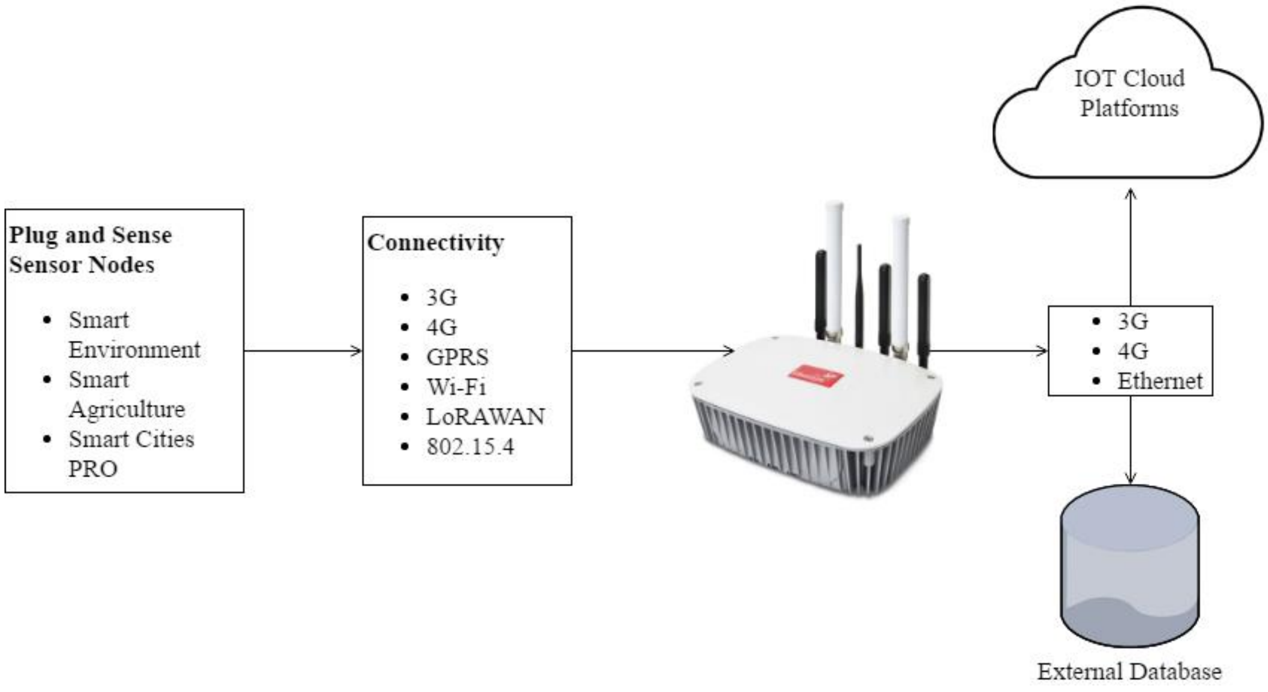


Figure 2.10: Meshlium Gateway
(Source: http://libelium.com/)

However, there are few limitations of Meshlium which are identified such as,

- *Cost:* Due to the features and functionalities of the Meshlium, the unit cost is high which has limited its potential to be deployed in large quantities. If a standard IOT deployment is considered where, simply the sensor data needs to be sent to a pre-decided cloud supported by Meshlium, it can be configured easily without using most of the features and functionalities of Meshlium. However still the client has to bear the cost of the complete unit without even using most of it.
- *Proprietary Protocols:* As discussed in section 3.5.13, Libelium use their own data frame protocols and system architectures which has stopped 3rd party sensor nodes from connecting with Meshlium and using it as a gateway. This has limited the potential of a full-fledged IOT deployment, as there can be requirements to use specific sensors which Libelium does not have, still along with the Libelium sensor nodes which has made the client to use two different connectivity mechanisms to push the sensor data possibly to the same IOT cloud platform.
- *Sending Alerts or Notifications:* Even though, Meshlium filters data in the gateway itself, it does not analyze and make decisions based on the filtered data which has introduced a different set of limitations which will be overcome from the proposed prototype that are discussed in later sections.

2.5.5 *Advantech*

Advantech is one of the top Industrial grade IOT solution providers in the industry who provides end to end solutions starting from the sensor to the IOT analytics platform. However, they do not have a broad sensor range as Libelium, but still equipped with commonly used set of sensors in IOT applications such as,

- Flood and water level monitoring
- Smart car parks; vehicle counting, air quality
- Smart irrigation systems monitoring soil moisture, environmental conditions, leaks
- Mechanical condition monitoring/preventative maintenance
- Energy measurements and audits on a per system or machine basis
- Data center environmental monitoring
- Tank and lift stations
- Condition monitoring and optimization in industrial environments
- Traffic monitoring of over-height vehicles for tunnels and bridges

2.5.6 *Advantech Sensor Nodes*

The sensor node which is called the "Wzzard" comes with an in-built sensor as well as a communication module where it uses ultra-low power 802.15.4e and Bluetooth technologies to connect with the Advantech gateway. Each sensor node comes with one sensor which can measure one parameter among pressure, flow rate, electric current, voltage, acceleration and temperature values [26].

Further the rugged, IP66-rated and fiber reinforced polyester PBT enclosure has made Wzzard sensor to be used in specific applications with challenging conditions. Unlike Libelium, Advantech uses standard data protocols such as MQTT and JSON. This has opened up many integration possibilities for Advantech product line.

Figure 2.11: Advantech Sensors
http://advantech-bb.com/

2.5.7 SmartSwarm Gateways

Below diagram showcases the possible integrations that can be done via the Advantech IOT gateway which is called as the "SmartSwarm" to the third party IOT platforms. Further the in-built Node-RED user applications environment has enabled the gateway to accept data from web feeds, databases and process them along with the data coming in from the Wzzard sensors. Also, the SmartSwarm gateway can be configured to push emails, SMS and Tweets etc [26].

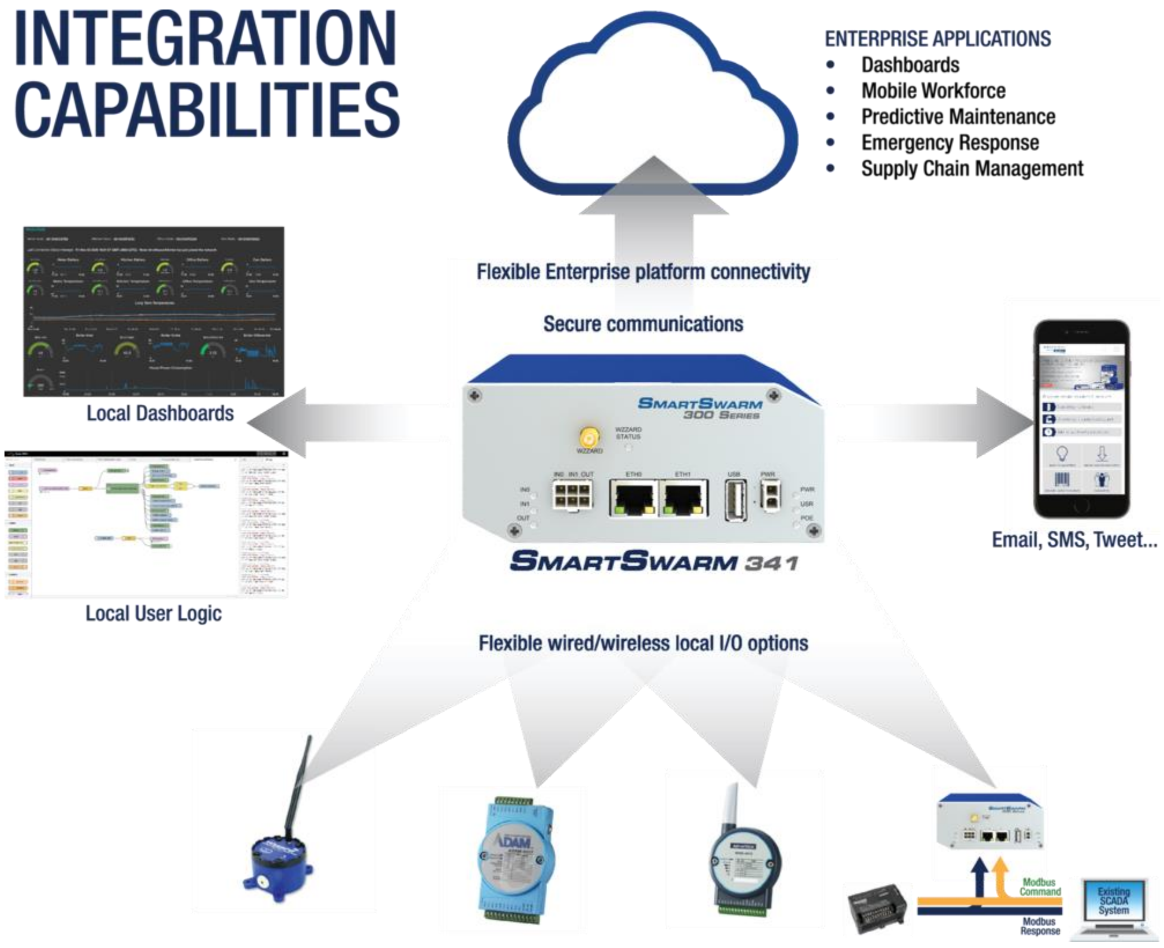


Figure 2.12: SmartSwarm Gateway
http://advantech-bb.com/

However, there are few key limitations of SmartSwarm gateway such as,

- *Cost:* Due to number of features it provides, the cost of a unit becomes a bottleneck in large scale IOT deployments. Similar to Meshlium, this gateway provides lots of features which might never be used sometimes.
- *Proprietary Network protocols:* Even though low-power Wi-Fi which is 802.15.4 protocol is a standard IOT network protocol, the SmartSwarm gateway connects

only with Wzzards sensor nodes which has limited the potential of a full-fledged multi-vendor IOT deployment.

2.6 Cloud Platforms

KGNuggets [7], explains about cloud platforms and their behavior. As mentioned in the site, the advent of cloud computing has acted as a catalyst for the development and deployment of scalable Internet-of-Things business models and applications. In this section first, I will give a brief introduction to the main cloud concepts. Next, IOT and cloud computing integration.

Cloud computing is the delivery of computing services – servers, storage, databases, networking, software, analytics and more – over the Internet ("the cloud"). Companies offering these computing services are called cloud providers and typically charge for cloud computing services based on usage [6].

2.6.1 Benefits of cloud computing

Cloud computing is a big shift from the traditional way businesses think about IT resources[6]. Why cloud computing is so popular? There are heap of sites and research papers answer the aforementioned question. According to the KGNuggets [7], here are six common reasons why organizations are turning to cloud computing services.

- *Cost*

Cloud computing eliminates the capital expense of buying hardware and software and setting up and running on-site data centers – the racks of servers, the round-the-clock electricity for power and cooling, the IT experts for managing the infrastructure.

- *Speed*

Most cloud computing services are provided as self service and on demand, so even vast amounts of computing resources can be provisioned in minutes, typically with just a few mouse clicks, giving businesses a lot of flexibility and taking the pressure off capacity planning.

- *Global scale*

The benefits of cloud computing services include the ability to scale elastically. In cloud speak, that means delivering the right amount of IT resources – for example, more or less: computing power, storage, bandwidth – exactly when it's needed.

- *Productivity*

On-site data centers typically require a lot of "racking and stacking" – hardware setup, software patching and other time-consuming IT management chores. Pushing hard work to the cloud makes IT team focus mainly on business goals.

- *Performance*

The biggest cloud computing services run on a worldwide network of secure data centers, which are regularly upgraded to the latest generation of fast and efficient computing hardware. This offers several benefits over a single corporate data center, including reduced network latency for applications and greater economies of scale.

- *Reliability*

Cloud computing makes data backup, disaster recovery and business continuity easier and less expensive, because data can be mirrored at multiple redundant sites on the cloud provider's network.

2.6.2 Types of cloud services: IaaS, PaaS, SaaS

Most cloud computing services fall into three broad categories: infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS). These are sometimes called the cloud computing stack, because they build on top of one another. KGNuggets [7] on their site explains the three main cloud concepts.

- *Infrastructure as a service (IaaS)*

The most basic category of cloud computing services. With IaaS, service consumer rent IT infrastructure – servers and virtual machines (VMs), storage, networks, operating systems – from a cloud provider on a pay-as-you-go basis.

- *Platform as a service (PaaS)*

Platform as a service (PaaS) refers to cloud computing services that supply an on-demand environment for developing, testing, delivering and managing software applications. PaaS is designed to make it easier for developers to quickly create web or mobile apps, without worrying about setting up or managing the underlying infrastructure of servers, storage, network and databases needed for development.

- *Software as a service (SaaS)*

Software as a service (SaaS) is a method for delivering software applications over the Internet, on demand and typically on a subscription basis. With SaaS, cloud providers host and manage the software application and underlying infrastructure, and handle *any maintenance, such as software upgrades and security patching. Users connect to the application over the Internet, usually with a web browser on their phone, tablet or PC.*

2.6.3 Types of cloud deployments: public, private, hybrid

Public cloud, Private cloud, Hybrid cloud are three different ways to deploy cloud computing resources [6].

- *Public cloud*

Public clouds are owned and operated by a third-party cloud service provider, which delivers computing resources such as servers and storage over the Internet. Microsoft Azure is an example of a public cloud. You access these services and manage your account using a web browser.

- *Private cloud*

A private cloud refers to cloud computing resources used exclusively by a single business or organization. A private cloud can be physically located on the company's on-site data center. Some companies also pay third-party service providers to host their private cloud. A private cloud is one in which the services and infrastructure are maintained on a private network.

- *Hybrid cloud*

Hybrid clouds combine public and private clouds, bound together by technology that allows data and applications to be shared between them. Hybrid cloud gives businesses greater flexibility and more deployment options.

2.6.4 IOT / Cloud Convergence

According to the KGNuggets [7], IOT and cloud is an excellent blend. As they say, Internet-of-Things can benefit from the scalability, performance and pay-as-you-go nature of cloud computing infrastructures. Moreover, they provide some examples where IOT and cloud have been used. As stated in the site, *IOT applications produce large volumes of data and comprise multiple computational components (e.g, data processing and analytics algorithms), their integration with cloud computing infrastructures could provide them with opportunities for cost-effective on-demand scaling* [7].

- *A Small Medium Enterprise (SME) developing an energy management IOT product, targeting smart homes and smart buildings. By streaming the data of the product* *(e.g., sensors and WSN data) into the cloud it can accommodate its growth needs in a scalable and cost effective fashion. As the SMEs acquires more customers and performs more deployments of its product, it is able to collect and manage growing volumes of data in a scalable way, thus taking advantage of a "pay-as-you-grow" model.*

- *A smart city can benefit from the cloud-based deployment of its IOT systems and applications. A city is likely to deploy many IOT applications, such as applications for smart energy management, smart water management, smart transport management, urban mobility of the citizens and more. These applications comprise multiple sensors and devices, along with computational components.*

All these applications consist of large volume of data and need the support for expansion.

Chapter 3

3. RESEARCH METHODOLOGY

3.1 Introduction

The objective of this research project is to use a mobile phone as an IOT gateway considering its Pros over Cons with compared to the traditional IOT Gateways. Based on the literature reviews carried out, there has been many occasions where many researchers have attempted to use Mobile phones as IOT gateways mainly due to their data protection and encryption capabilities which is not a strong factor of traditional IOT Gateways.

However not much research has been carried out on using the mobile phone as an IOT Gateway with data filtering and real-time notification sending functionalities which is not evident in most of the traditional IOT Gateways. Therefore, this research project is based on that research gap identified where a system will be developed to emphasize on aforesaid functionalities.

Therefore, this chapter discusses the proposed solution architecture of the system which needs to be implemented and the comprehensive study carried out on each key component.

3.2 Solution Architecture

In every generic IOT deployment, there are few key components that can be easily identified such as the IOT sensors, IOT sensor controller, IOT gateway and the IOT/ data analytic platform. Below diagram showcases those components and how they are linked with each other.

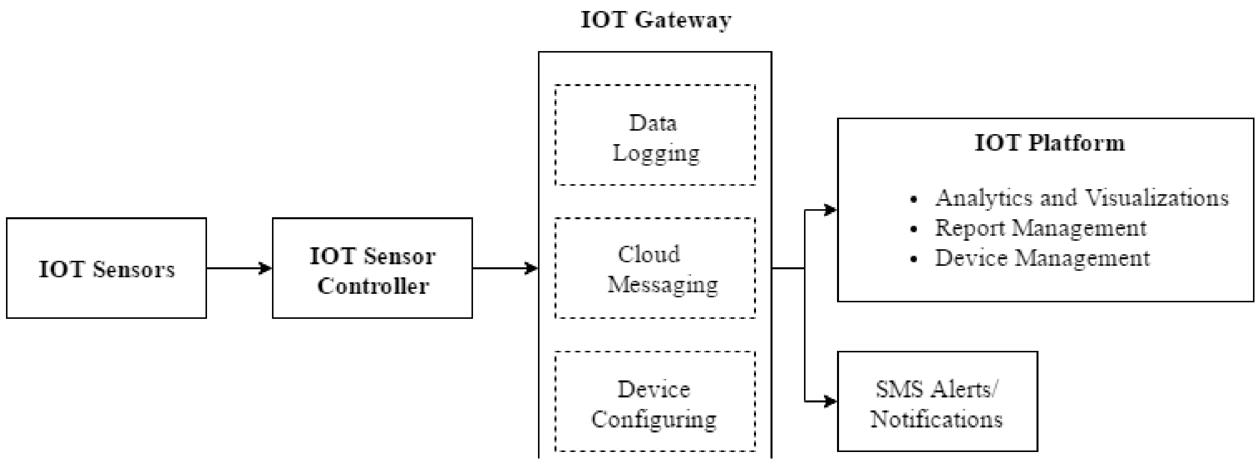


Figure 3.1: Solution Architecture

It can be noticed that in addition to the key components, the alerts/ notification service is also added to the system architecture, considering this research where real-time alerts/ notification service plays a major role.

3.3 Real-world IOT Solution deployments

Even though the standard IOT deployment structure is as explained above, the practical deployments could largely vary, mostly from IOT sensors to the IOT gateways. Below shown are some of the key solution setups which are widely being implemented across IOT projects.

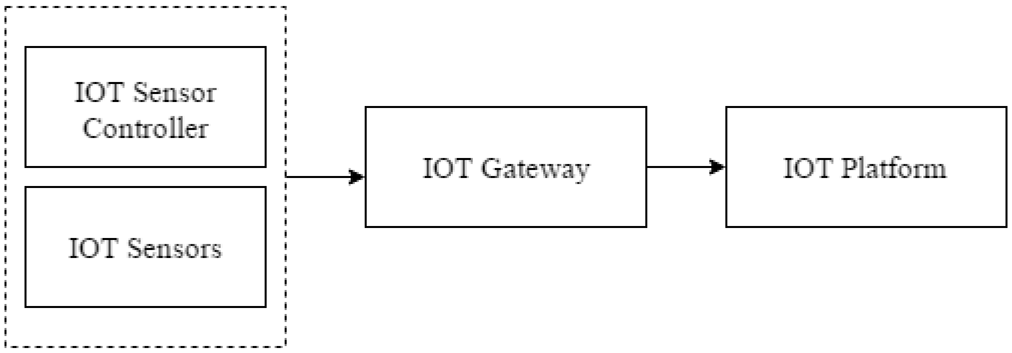


Figure 3.2: Deployment Option 1

Above deployment where the sensor controller is embedded with the sensors can be seen mainly in industrial IOT products where few key sensor readings are needed and the overall set up needs to be ruggedized and capable of withstanding extreme conditions. The controller is getting connected to the IOT gateway via traditional IOT connectivity protocols such as LoRa and Low Power Wi-Fi which will be pushed to the cloud via the Gateway using Wi-Fi and Cellular networks.

Advantech B+B Products can be an ideal example for such a deployment considering their different sensors and multiple IOT gateways.

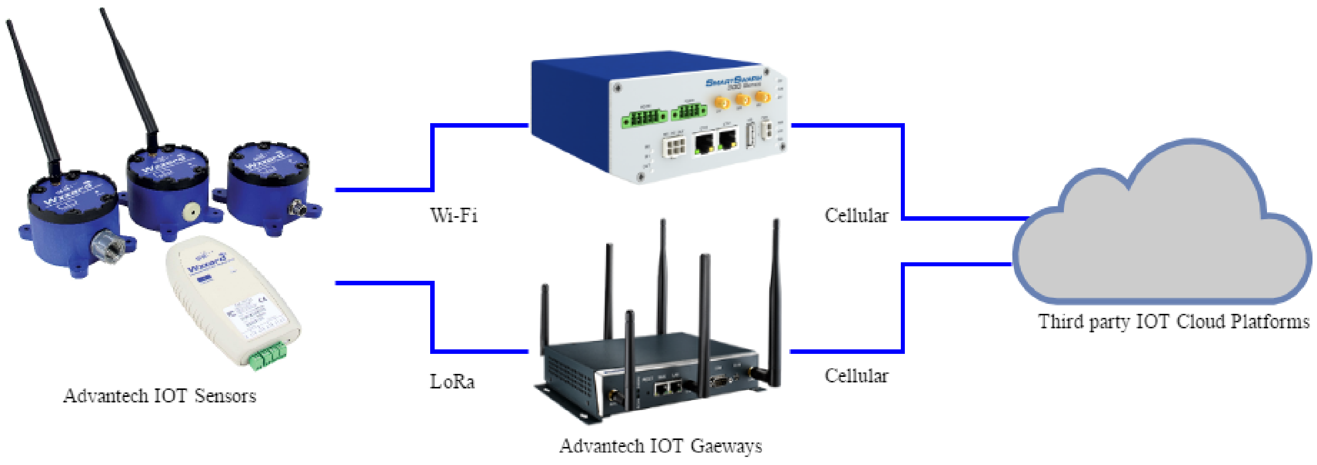


Figure 3.3: Advantech IOT Products
(Source: http://advantech-bb.com/)

Below deployment is where the IOT gateway is also embedded inside the IOT Controller to minimize the points of dependencies. The sensors can be connected to the sensor controller based on the required output.

These types of deployments are commonly seen in smart city projects where the sensors can vary depending on the expected result, however the sensor controller is the same to reduce the costs and to maintain the flexibility.

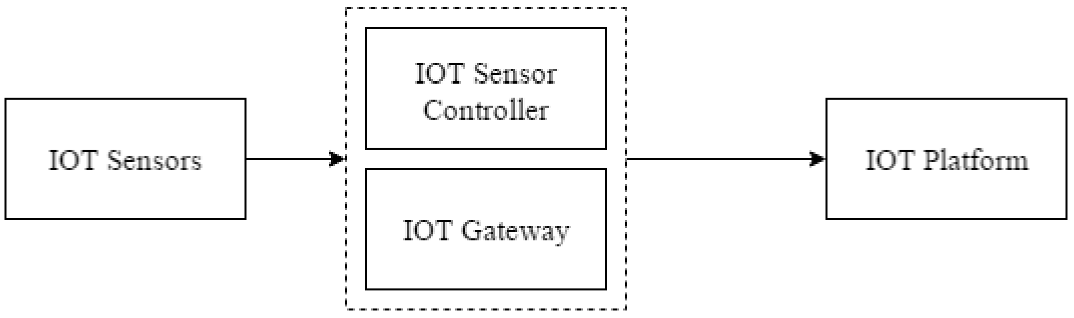


Figure 3.4: Development Option 2

Libelium products can be identified as an ideal example for such product series. They have over more than 160 sensors which can work with nearly 10 sensor controllers with multiple IOT protocols.

3.4 Prototype Selection

Considering above deployments, this research will be based on a prototype developed according to the architecture in Figure 3.2 where the sensors are embedded to the sensor controller.

A sensor tag which is built around a Nordic Semiconductor is used in this study which contains multiple sensors as explained below.

3.4.1 nRF51822 Sensor Tag

The **nRF51822** is a powerful, highly flexible multiprotocol SoC ideally suited for *Bluetooth*® low energy and 2.4GHz ultra low-power wireless applications. The **nRF51822** is built around a 32-bit ARM® Cortex™ M0 CPU with 256kB/128kB flash + 32kB/16kB RAM for improved application performance.

The embedded 2.4GHz transceiver supports both Bluetooth low energy and the Nordic Gazell 2.4 GHz protocol stack which is on air compatible with the nRF24L series products from Nordic Semiconductor.

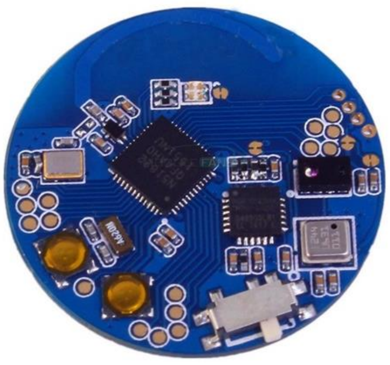


Figure 3.5: nRF51822 Sensor Tag (Source: https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF51822)

3.4.2 3-axis accelerometer MPU6050

The MPU-6050 is a serious little piece of motion processing tech! By combining a MEMS 3-axis gyroscope and a 3-axis accelerometer on the same silicon die together with an onboard Digital Motion Processor™ (DMP™) capable of processing complex 9-axis MotionFusion algorithms, the MPU-6050 does away with the cross-axis alignment problems that can creep up on discrete parts.

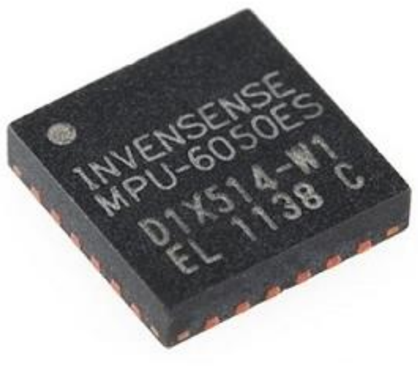


Figure 3.6: 3-axis accelerometer MPU6050

3.4.3 *Bosch temperature and pressure sensor BMP180*

The BMP180 is a new digital barometric pressure sensor of Bosch Sensortec, with a very high performance, which enables applications in advanced mobile devices, such as smartphones, tablet PCs and sports devices.

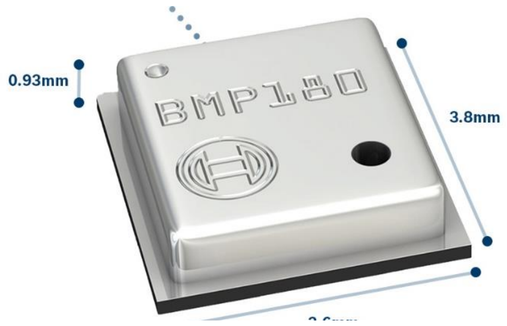


Figure 3.7: Bosch temperature and pressure sensor BMP180 (Source: https://www.bosch-sensortec.com/bst/products/all_products/bmp180)

Key specifications of this sensor would be,

- Ability to sense the barometric pressure and altitude
- Availability of multiple IO interfaces
- Smaller size
- Very low power requirement
- Higher reliability and performance

3.4.4 *Ambient light sensor/PIR/Motion detection AP3216*

The AP3216C/D is an integrated ALS & PS module that includes a digital ambient light sensor [ALS], a proximity sensor [PS], and an IR LED in a single package. 3It is used in mobile phones, to measure the ambient light and adjust the screen brightness, or to detect when you hold your phone to the ear and disable the touchscreen.

Figure 3.8: AP3216 Sensor (Source: http://www.dyna-image.com/english/product/optical-sensor-detail.php?cpid=3&dpid=17#doc)

Key specifications of this sensor would be,

- Ability to measure the ambient light levels (ALS), Proximity sensor (PS) and IR based motion detection (IR)
- Smaller size
- Ability select different modes such as ALS, PS+IR, ALS+PS+IR, PD, ALS once, SW Reset, PS+IR once and ALS+PS+IR once

- Wider operating temperature range (-30°C to +80°C)

3.5 Prototype Development

It is decided to connect the above sensor tag with an Android Mobile device via BLE (Blue-tooth Low Energy) to capture the data logs which is called the GATT profile log for further data processing.

After that the processed data will be sent to the cloud where Splunk will be used as the IOT cloud for further data analytics and visualizations. In parallel, critical alerts and notifications shall be triggering from the android gateway itself for more proactive and faster reactions.

Below diagram explains the technical setup of the prototype to be developed.

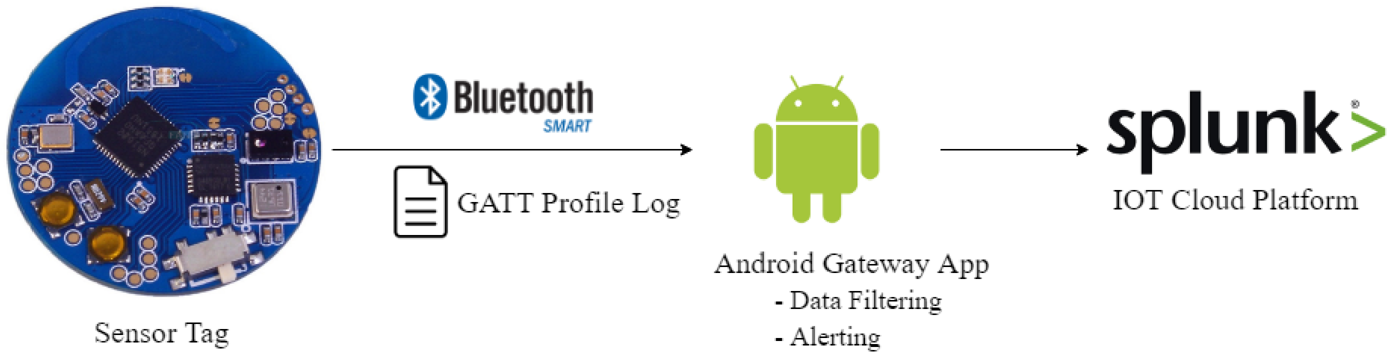


Figure 3.9: Prototype Key Components

3.5.1 *Blue-tooth Low Energy*

Android OS Version 4.3 (API level 18) comes up with built-in platform support for Bluetooth Low Energy (BLE) and provides APIs that apps can use to discover devices, query for services, and transmit information.

Common use cases include the following:

- Transferring small amounts of data between nearby devices.
- Interacting with proximity sensors like Google Beacons to give users a customized experience based on their current location.

In contrast to Classic Bluetooth, Bluetooth Low Energy (BLE) is designed to provide significantly lower power consumption. This allows Android apps to communicate with BLE devices that have stricter power requirements, such as proximity sensors, heart rate monitors, and fitness devices [17].

3.5.2 *Generic Attribute Profile (GATT)*

The GATT profile is a general specification for sending and receiving short pieces of data known as "attributes" over a BLE link. All current Low Energy application profiles are based on GATT [17].

The Bluetooth SIG defines many profiles for Low Energy devices. A profile is a specification for how a device works in a particular application. Note that a device can implement more than one profile. For example, a device could contain a heart rate monitor and a battery level detector [17].

3.5.3 *Services and Characteristics*

GATT transactions in BLE are based on high-level, nested objects called Profiles, Services and Characteristics [17], which can be seen in the illustration below:

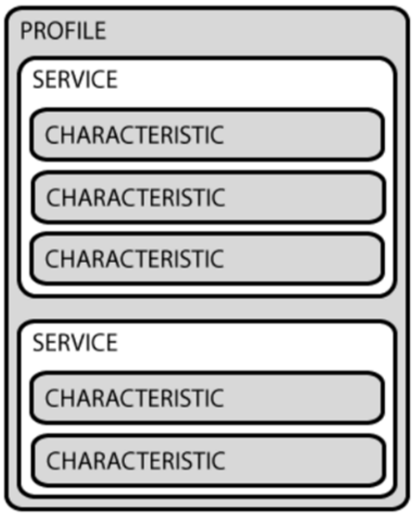


Figure 3.10: GATT Services and Characteristics
(Source: https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gatt)

3.5.4 *Roles and Responsibilities*

Here are the roles and responsibilities that apply when an Android device interacts with a BLE device:

- Central vs. peripheral. This applies to the BLE connection itself. The device in the central role scans, looking for advertisement, and the device in the peripheral role makes the advertisement.
- GATT server vs. GATT client. This determines how two devices talk to each other once they've established the connection.

This can be achieved by using Android phone and an activity tracker that is a BLE device. The phone supports the central role; the activity tracker supports the peripheral role [17].

Once the phone and the activity tracker have established a connection, they start transferring GATT metadata to one another. Depending on the kind of data they transfer, one or the other might act as the server. For example, if the activity tracker wants to report sensor data to the phone, it might make sense for the activity tracker to act as the server. If the activity tracker wants to receive updates from the phone, then it might make sense for the phone to act as the server [17].

3.5.5 GATT Characteristics and Attributes

While characteristics and attributes are sometimes used interchangeably when referring to Bluetooth low energy, characteristics can be considered as groups of information called attributes. Attributes are the information transferred between devices. Characteristics organize and use attributes as data values, properties, and configuration information. A typical characteristic is composed of the following attributes [22].

- Characteristic Value

data value of the characteristic

- Characteristic Declaration

descriptor storing the properties, location, and type of the characteristic value

- Client Characteristic Configuration

a configuration that allows the GATT server to configure the characteristic to be notified (send message asynchronously) or indicated (send message asynchronously with acknowledgment)

- Characteristic User Description

an ASCII string describing the characteristic

These attributes are stored in the GATT server in an attribute table. In addition to the value, the following properties are associated with each attribute.

- Handle

the index of the attribute in the table (Every attribute has a unique handle.)

- Type

indicates what the attribute data represents (referred to as a UUID [universal unique identifier]. Some of these are Bluetooth SIG-defined and some are custom.)

- Permissions

enforces if and how a GATT client device can access the value of an attribute [22].

3.5.6 GATT Transactions

An important concept to understand with GATT is the server/client relationship. The peripheral is known as the GATT Server, which holds the ATT lookup data and service and characteristic definitions, and the GATT Client (the phone/tablet), which sends requests to this server [18].

All transactions are started by the master device, the GATT Client, which receives response from the slave device, the GATT Server. When establishing a connection, the peripheral will suggest a 'Connection Interval' to the central device, and the central device will try to reconnect every connection interval to see if any new data is available, etc. The following diagram should illustrate the data exchange process between a peripheral (the GATT Server) and a central device (the GATT Client) [18].

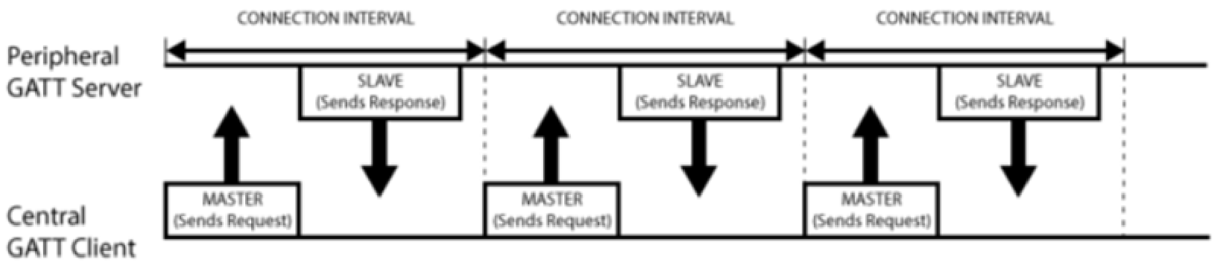


Figure 3.11: GATT Data Exchange Process
(Source: https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gatt)

3.5.7 *nRF51 profile log*

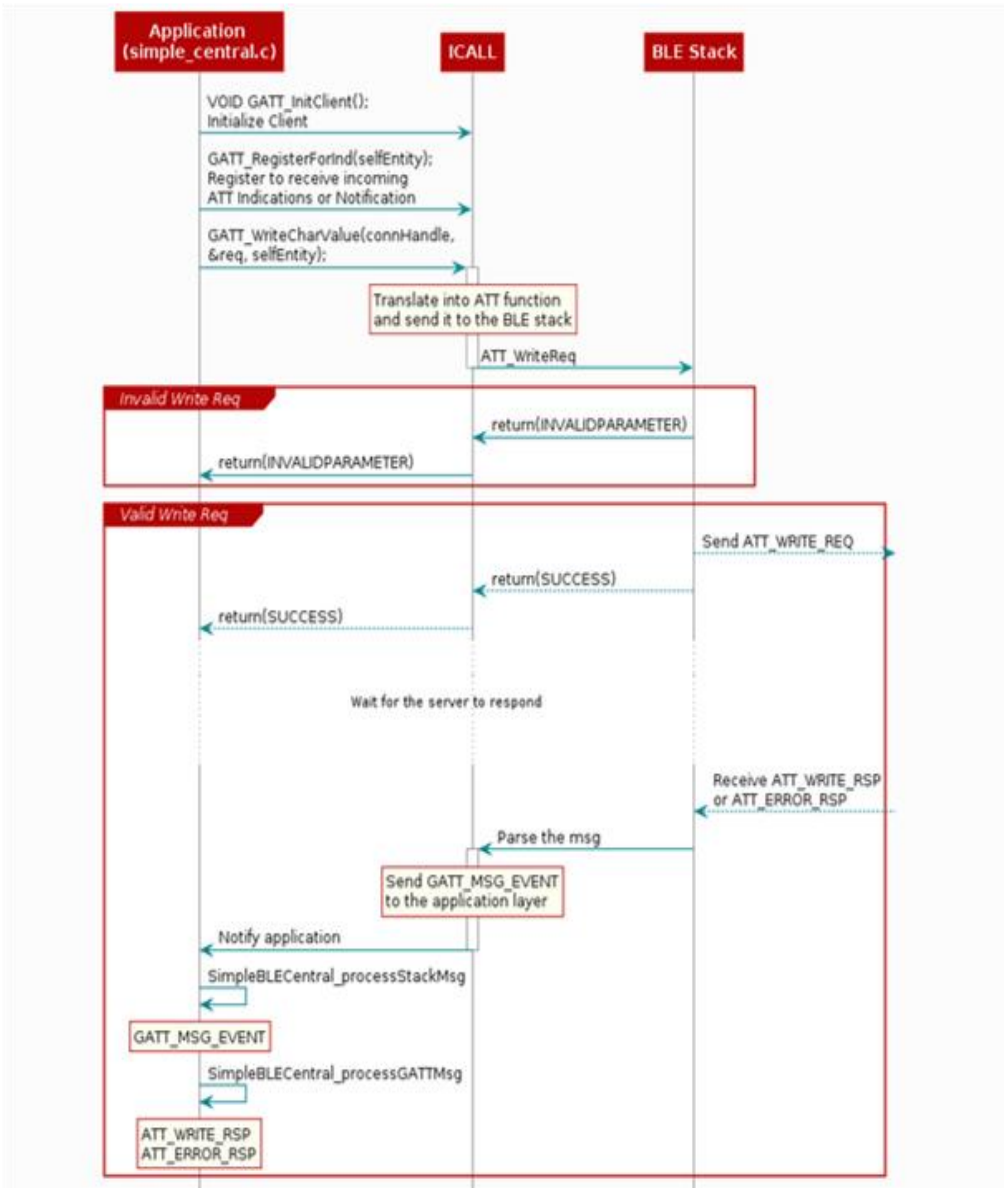
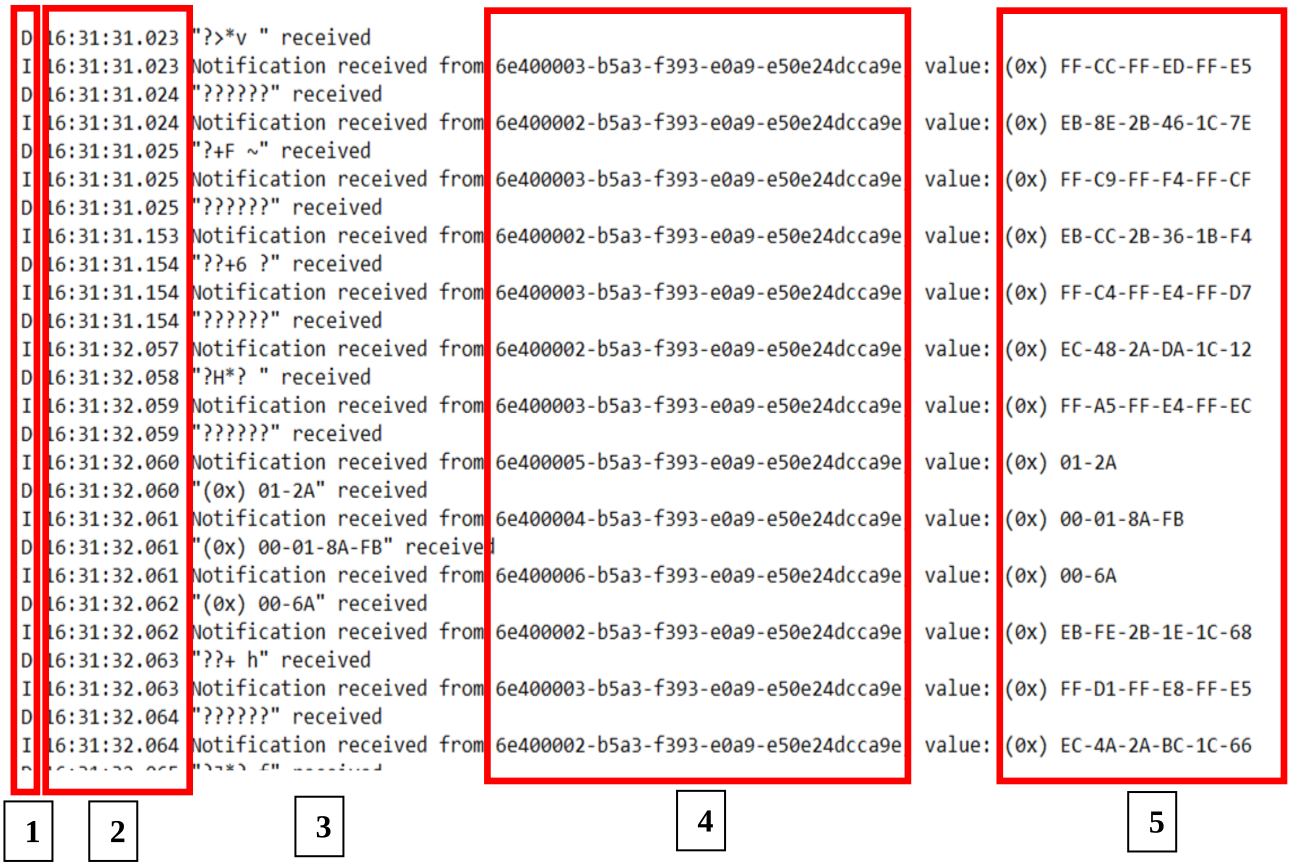

Application (simple_central.c)
ICALL
BLE Stack
VOID GATT_InitClient(); Initialize Client
GATT_RegisterForInd(selfEntity); Register to receive incoming ATT Indications or Notification
GATT_WriteCharValue(connHandle, &req, selfEntity);
Translate into ATT function and send it to the BLE stack
ATT_WriteReq
Invalid Write Req
return(INVALIDPARAMETER)
return(INVALIDPARAMETER)
Valid Write Req
Send ATT_WRITE_REQ
return(SUCCESS)
return(SUCCESS)
Wait for the server to respond
Receive ATT_WRITE_RSP or ATT_ERROR_RSP
Parse the msg
Send GATT_MSG_EVENT to the application layer
Notify application
SimpleBLECentral_processStackMsg
GATT_MSG_EVENT
SimpleBLECentral_processGATTMsg
ATT_WRITE_RSP ATT_ERROR_RSP

Figure 3.12: nRF51 Profile Log

(Source: http://dev.ti.com/tirex/content/simplelink_cc2640r2_sdk_1_40_00_45/docs/blestack/ble_user_guide/html/ble-stack-3.x/gatt.html)

3.5.8 nRF Log File Explanation



```
D 16:31:31.023 "?>*v " received
I 16:31:31.023 Notification received from 6e400003-b5a3-f393-e0a9-e50e24dcca9e  value: (0x) FF-CC-FF-ED-FF-E5
D 16:31:31.024 "??????" received
I 16:31:31.024 Notification received from 6e400002-b5a3-f393-e0a9-e50e24dcca9e  value: (0x) EB-8E-2B-46-1C-7E
D 16:31:31.025 "?+F ~" received
I 16:31:31.025 Notification received from 6e400003-b5a3-f393-e0a9-e50e24dcca9e  value: (0x) FF-C9-FF-F4-FF-CF
D 16:31:31.025 "??????" received
I 16:31:31.153 Notification received from 6e400002-b5a3-f393-e0a9-e50e24dcca9e  value: (0x) EB-CC-2B-36-1B-F4
D 16:31:31.154 "??+6 ?" received
I 16:31:31.154 Notification received from 6e400003-b5a3-f393-e0a9-e50e24dcca9e  value: (0x) FF-C4-FF-E4-FF-D7
D 16:31:31.154 "??????" received
I 16:31:32.057 Notification received from 6e400002-b5a3-f393-e0a9-e50e24dcca9e  value: (0x) EC-48-2A-DA-1C-12
D 16:31:32.058 "?H*? " received
I 16:31:32.059 Notification received from 6e400003-b5a3-f393-e0a9-e50e24dcca9e  value: (0x) FF-A5-FF-E4-FF-EC
D 16:31:32.059 "??????" received
I 16:31:32.060 Notification received from 6e400005-b5a3-f393-e0a9-e50e24dcca9e  value: (0x) 01-2A
D 16:31:32.060 "(0x) 01-2A" received
I 16:31:32.061 Notification received from 6e400004-b5a3-f393-e0a9-e50e24dcca9e  value: (0x) 00-01-8A-FB
D 16:31:32.061 "(0x) 00-01-8A-FB" received
I 16:31:32.061 Notification received from 6e400006-b5a3-f393-e0a9-e50e24dcca9e  value: (0x) 00-6A
D 16:31:32.062 "(0x) 00-6A" received
I 16:31:32.062 Notification received from 6e400002-b5a3-f393-e0a9-e50e24dcca9e  value: (0x) EB-FE-2B-1E-1C-68
D 16:31:32.063 "??+ h" received
I 16:31:32.063 Notification received from 6e400003-b5a3-f393-e0a9-e50e24dcca9e  value: (0x) FF-D1-FF-E8-FF-E5
D 16:31:32.064 "??????" received
I 16:31:32.064 Notification received from 6e400002-b5a3-f393-e0a9-e50e24dcca9e  value: (0x) EC-4A-2A-BC-1C-66
```

Figure 3.13: nRF Log File Explanation

Below table showcases the boxed columns.

Table 3.1: Log Profile Description

Column No	Category	Example Value
1	Info (log level)	I
2	Time Stamp	31:29.4
3	UUID Description	Notification received from
4	Sensor ID	6e400002-b5a3-f393-e0a9-e50e24dcca9e
5	Sensor Value	EC-50-29-F6-1B-AC

3.6 Multi-Device, Multi-Vendor Integrations

In addition to above proposed development, the prototype shall be capable of translating and filtering data from IOT sensors from multiple vendors. A sensor data log from Libelium products shall be considered in this prototype development.

Even though the prototype is not capable of accepting data directly from Libelium end nodes due to the connectivity limitations, the gateway application should be able to accept data from this Libelium sensor data log, filter and send alerts.

The sample sensor data log contains sensor data from 3 different Plug & Sense units connected with different types of sensors as shown in the table below.

Table 3.2: Libelium Nodes and Sensors

Plug & Sense Unit	**Sensor Type**
Smart Environment PRO	NO2, CO, O3, Temperature, Humidity and Atmospheric Pressure
Smart Cities PRO	Accelerometer, Temperature, Humidity and Atmospheric Pressure
Smart Agriculture	Luminosity

3.6.1 Smart Environment PRO

This sensor node is tailor made to suite most of the smart environmental solutions where measuring the air quality is a key requirement. This Plug & Sense is capable of accepting sensor data from multiple sensors as shown in the table 3.3 below.

Table 3.3: Smart Environment PRO Sensors
(Source: http://libelium.com/)

Sensor Socket	Sensor Type
A, B, C or F	CO, CO2, O2, O3, NO, NO2, SO2, NH3, CH4, H2, H2S, HCI, HCN, PH3, ETO, Cl2 air quality sensors
D	Particle matter
E	Temperature, Pressure, Humidity, Luminosity, Ultra-sound

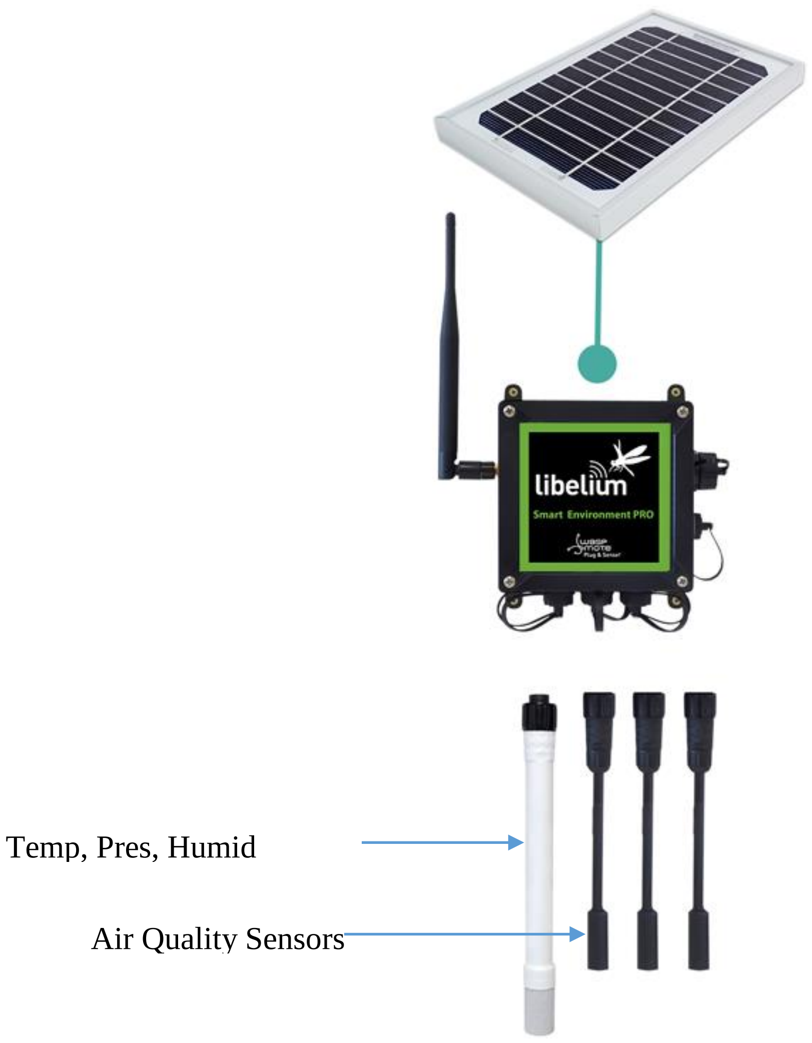


Figure 3.14: Smart Environment PRO Sensor Node
(Source: http://libelium.com/)

3.6.2 *Smart Cities PRO*

This sensor node consists of sensors which covers all most all of the smart city solutions and applications. In addition to the air quality monitoring sensors, this unit can accept noise level data and ultra sound distance levels which are used in deploying traffic management systems, city pollution level monitoring systems etc. Below table 3.4 illustrates the types of sensors accepted by this node.

Table 3.4: Smart Cities PRO Sensors
(Source: http://libelium.com/)

Sensor Socket	**Sensor Type**
A	Noise Level, Ultra Sound, Temperature, Pressure, Humidity, Luminosity

Sensor Socket	**Sensor Type**
B, C, F	CO, CO2, O2, O3, NO, NO2, SO2, NH3, CH4, H2, H2S, HCI, HCN, PH3, ETO, Cl2 air quality sensors, Temperature, Pressure, Humidity, Luminosity, Ultra Sound
D	Particle matter
E	Temperature, Pressure, Humidity, Luminosity, Ultra-sound

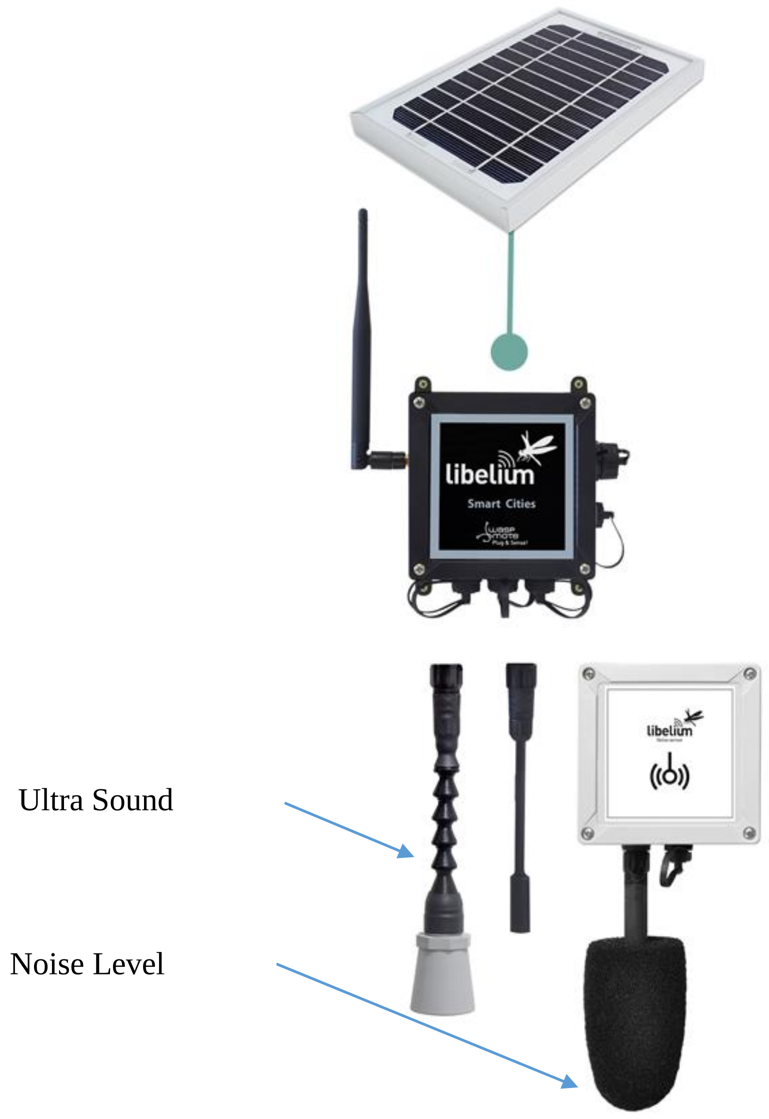


Figure 3.15: Smart Cities PRO Sensor Node
(Source: http://libelium.com/)

3.6.3 Smart Agriculture

Smart Agriculture Plug & Sense contains a set of sensors which measures soil quality, water quality and weather conditions etc., which is very useful to maintain the quality requirements of farm lands, livestock and even large crop fields. Lora based Sensor Nodes are commonly used in Smart Agriculture IOT deployments due to poor infrastructure conditions in traditional connectivity modes. The table 3.5 showcases the commonly used set of sensors with this sensor node.

Table 3.5: Smart Agriculture Sensors
(Source: http://libelium.com/)

Sensor Socket	**Sensor Type**
A	Anemometer, wind vane, pluviometer
B	Soil Moisture, Solar Radiation, Ultraviolet Radiation
C	Soil Moisture, Dendrometers
D	Soil Temperature, Temperature, Atmospheric Pressure, Humidity, Luminosity, Ultra Sound
E	Leaf Wetness, Soil Moisture
F	Temperature, Atmospheric Pressure, Humidity, Luminosity, Ultra Sound

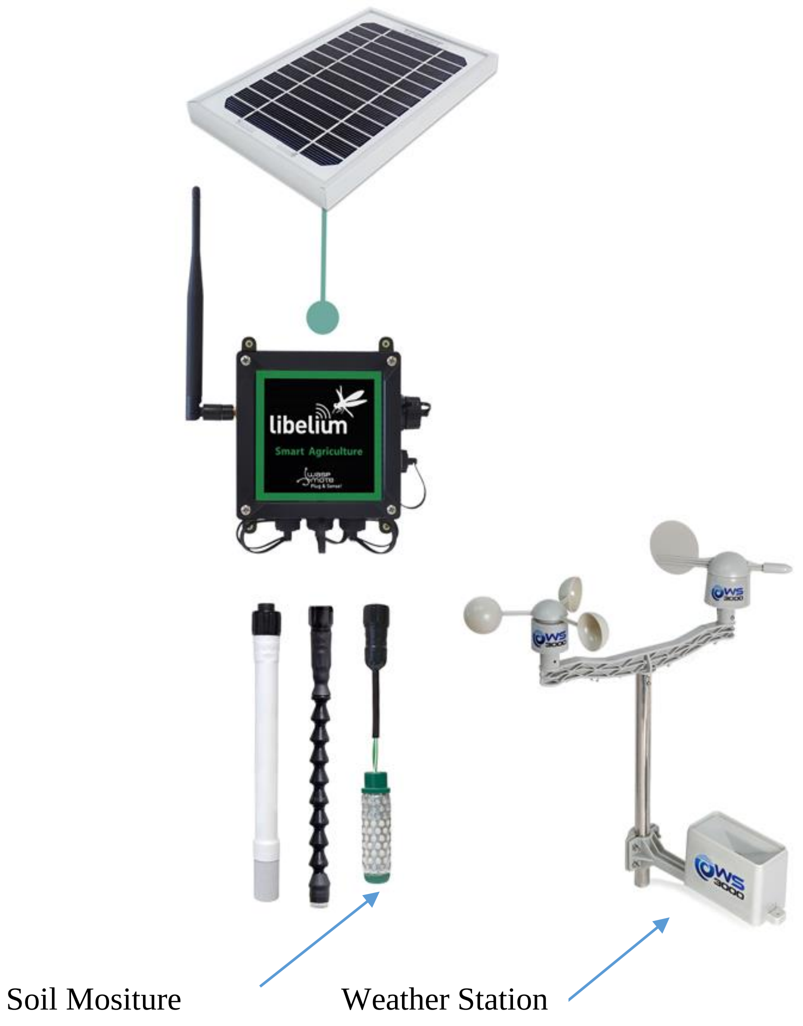
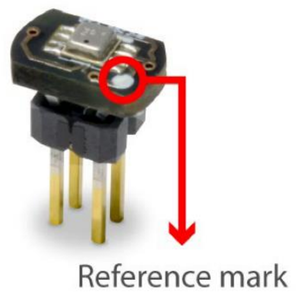


Figure 3.16: Smart Agriculture Sensor Node
(Source: http://libelium.com/)

3.6.4 Sensors in Sensor Data Log

This section briefly discusses the features and functionalities of the sensors that are being used in the sensor data log which are initially tabulated in table 3.2. Even though the Libelium sensors look rugged and bulky, that is purely the robust enclosure which protects them from harsh outdoor conditions. The actual sensor is quite small and fragile as shown in table 3.6 below.

Table 3.6: Libelium Sensors
(Source: http://libelium.com/)

Sensor Type	**Key Features**
Temperature, Pressure and Humidity(BME280) 	**Temperature sensor** - Operational range: -40 ~ +85 ºC - Full accuracy range: 0 ~ +65 ºC - Accuracy: ±1 ºC (range 0 ºC ~ +65 ºC) - Response time: 1.65 seconds (63% response from +30 to +125 °C). - Typical consumption: 1 µA measuring **Humidity sensor** - Measurement range: 0 ~ 100% of Relative Humidity (for temperatures < 0 °C and > 60 °C see figure below) - Accuracy: < ±3% RH (at 25 ºC, range 20 ~ 80%) - Hysteresis: ±1% RH - Operating temperature: -40 ~ +85 ºC - Response time (63% of step 90% to 0% or 0% to 90%): 1 second - Typical consumption: 1.8 µA measuring - Maximum consumption: 2.8 µA measuring **Pressure sensor** - Measurement range: 30 ~ 110 kPa - Operational temperature range: -40 ~ +85 ºC - Full accuracy temperature range: 0 ~ +65 ºC - Absolute accuracy: ±0.1 kPa (0 ~ 65 ºC) - Typical consumption: 2.8 µA measuring - Maximum consumption: 4.2 µA measuring

Sensor Type	**Key Features**
Luminosity Sensor	- Dynamic range: 0.1 to 40000 lux - Spectral range: 300 ~ 1100 nm - Voltage range: 2.7 ~ 3.6 V - Supply current typical: 0.24 mA - Sleep current maximum: 0.3 µA - Operating temperature: -30 ~ 70 ºC
NO2 Sensor	- Nominal Range: 0 to 20 ppm - Maximum Overload: 50 ppm - Long Term Sensitivity Drift: < -20% to -40% change/year in lab air, monthly test - Long Term zero Drif: < 20 ppb equivalent change/year in lab air - Response Time (T90): ≤ 60 seconds - Sensitivity: -175 to -450 nA/ppm - Accuracy: as good as ±0.1 ppm* (ideal conditions) - O3 filter capacity @ 2 ppm: > 500 ppm·hrs
CO Sensor	- Nominal Range: 0 to 25 ppm - Maximum Overload: 2000 ppm - Long Term Sensitivity Drift: < 10% change/year in lab air, monthly test - Long Term zero Drift: < ±100 ppb equivalent change/year in lab air - Response Time (T90): ≤ 20 seconds - Sensitivity: 220 to 375 nA/ppm - Accuracy: as good as ±0.1 ppm* (ideal conditions) - H2S filter capacity: 250000 ppm·hrs
O3 Sensor	- Nominal Range: 0 to 18 ppm - Maximum Overload: 50 ppm - Long Term sensitivity Drift: -20 to -40% change/year - Response Time (T90): ≤ 45 seconds - Sensitivity: -200 to -550 nA/ppm - Accuracy: as good as ±0.2 ppm* (ideal conditions)

3.6.5 Waspframes

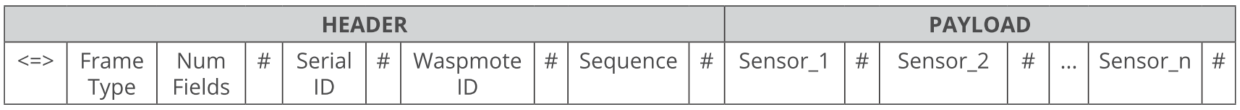
Waspframe is the official data frame format of Libelium where all the captured sensor data are being sent from the “Plug & Sense” Sensor nodes. The objective of having this data frame is to give a specific format to the sensor data frames. Meshlium, the Libelium gateway which is discussed in detail in section 2.5.4 is capable of decoding waspframes which has made IOT deployments easier and faster.

In brief, there are three main data frame formats such as,

- ASCII Frames
- Binary Frames
- Tiny Frames

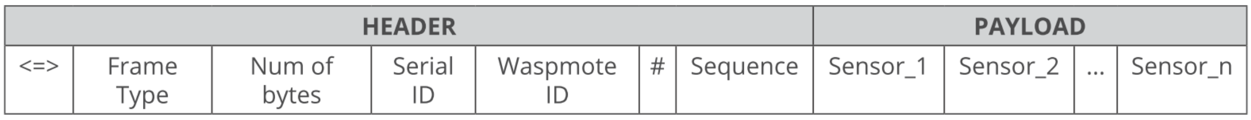
Table 3.7: Waspframe Structure
(Source: http://libelium.com/)

ASCII Frames – Data frame format followed by Plug & Sense nodes and can be decoded at the Meshlium Gateway



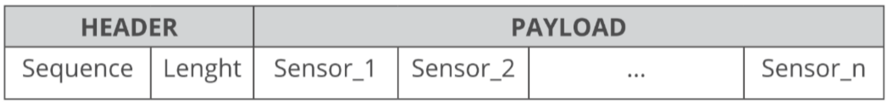
HEADER										PAYLOAD						
<=>	Frame Type	Num Fields	#	Serial ID	#	Waspmote ID	#	Sequence	#	Sensor_1	#	Sensor_2	#	...	Sensor_n	#

Binary Frames – A more compressed data frame format followed by Plug & Sense nodes and can be decoded at the Meshlium Gateway



HEADER							PAYLOAD			
<=>	Frame Type	Num of bytes	Serial ID	Waspmote ID	#	Sequence	Sensor_1	Sensor_2	...	Sensor_n

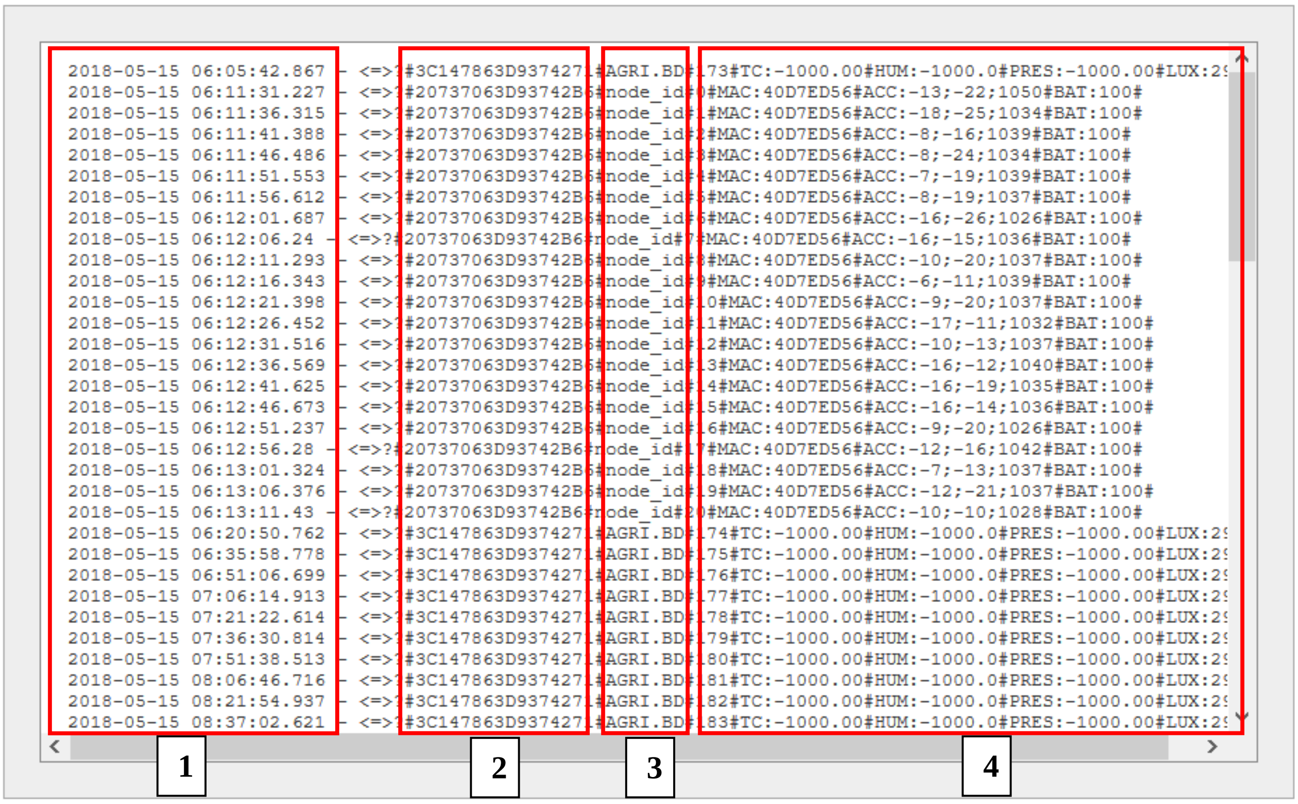
Tiny Frames – Designed to send sensor data via low bit-rate protocols with short payload size commonly used to send data to LoRAWAN and SigFox base stations. Cannot be decoded at the Meshlium end.



HEADER		PAYLOAD			
Sequence	Lenght	Sensor_1	Sensor_2	...	Sensor_n

3.6.6 Sample Sensor Data

Below listed are some of the raw sensor data that are being pushed as waspframes in ASCII data format from Plug & Sense sensor nodes.



```
2018-05-15 06:05:42.867 - <=>?#3C147863D9374271#AGRI.BD#173#TC:-1000.00#HUM:-1000.0#PRES:-1000.00#LUX:2
2018-05-15 06:11:31.227 - <=>?#20737063D93742B6#node_id#0#MAC:40D7ED56#ACC:-13;-22;1050#BAT:100#
2018-05-15 06:11:36.315 - <=>?#20737063D93742B6#node_id#1#MAC:40D7ED56#ACC:-18;-25;1034#BAT:100#
2018-05-15 06:11:41.388 - <=>?#20737063D93742B6#node_id#2#MAC:40D7ED56#ACC:-8;-16;1039#BAT:100#
2018-05-15 06:11:46.486 - <=>?#20737063D93742B6#node_id#3#MAC:40D7ED56#ACC:-8;-24;1034#BAT:100#
2018-05-15 06:11:51.553 - <=>?#20737063D93742B6#node_id#4#MAC:40D7ED56#ACC:-7;-19;1039#BAT:100#
2018-05-15 06:11:56.612 - <=>?#20737063D93742B6#node_id#5#MAC:40D7ED56#ACC:-8;-19;1037#BAT:100#
2018-05-15 06:12:01.687 - <=>?#20737063D93742B6#node_id#6#MAC:40D7ED56#ACC:-16;-26;1026#BAT:100#
2018-05-15 06:12:06.24 - <=>?#20737063D93742B6#node_id#7#MAC:40D7ED56#ACC:-16;-15;1036#BAT:100#
2018-05-15 06:12:11.293 - <=>?#20737063D93742B6#node_id#8#MAC:40D7ED56#ACC:-10;-20;1037#BAT:100#
2018-05-15 06:12:16.343 - <=>?#20737063D93742B6#node_id#9#MAC:40D7ED56#ACC:-6;-11;1039#BAT:100#
2018-05-15 06:12:21.398 - <=>?#20737063D93742B6#node_id#10#MAC:40D7ED56#ACC:-9;-20;1037#BAT:100#
2018-05-15 06:12:26.452 - <=>?#20737063D93742B6#node_id#11#MAC:40D7ED56#ACC:-17;-11;1032#BAT:100#
2018-05-15 06:12:31.516 - <=>?#20737063D93742B6#node_id#12#MAC:40D7ED56#ACC:-10;-13;1037#BAT:100#
2018-05-15 06:12:36.569 - <=>?#20737063D93742B6#node_id#13#MAC:40D7ED56#ACC:-16;-12;1040#BAT:100#
2018-05-15 06:12:41.625 - <=>?#20737063D93742B6#node_id#14#MAC:40D7ED56#ACC:-16;-19;1035#BAT:100#
2018-05-15 06:12:46.673 - <=>?#20737063D93742B6#node_id#15#MAC:40D7ED56#ACC:-16;-14;1036#BAT:100#
2018-05-15 06:12:51.237 - <=>?#20737063D93742B6#node_id#16#MAC:40D7ED56#ACC:-9;-20;1026#BAT:100#
2018-05-15 06:12:56.28 - <=>?#20737063D93742B6#node_id#17#MAC:40D7ED56#ACC:-12;-16;1042#BAT:100#
2018-05-15 06:13:01.324 - <=>?#20737063D93742B6#node_id#18#MAC:40D7ED56#ACC:-7;-13;1037#BAT:100#
2018-05-15 06:13:06.376 - <=>?#20737063D93742B6#node_id#19#MAC:40D7ED56#ACC:-12;-21;1037#BAT:100#
2018-05-15 06:13:11.43 - <=>?#20737063D93742B6#node_id#20#MAC:40D7ED56#ACC:-10;-10;1028#BAT:100#
2018-05-15 06:20:50.762 - <=>?#3C147863D9374271#AGRI.BD#174#TC:-1000.00#HUM:-1000.0#PRES:-1000.00#LUX:2
2018-05-15 06:35:58.778 - <=>?#3C147863D9374271#AGRI.BD#175#TC:-1000.00#HUM:-1000.0#PRES:-1000.00#LUX:2
2018-05-15 06:51:06.699 - <=>?#3C147863D9374271#AGRI.BD#176#TC:-1000.00#HUM:-1000.0#PRES:-1000.00#LUX:2
2018-05-15 07:06:14.913 - <=>?#3C147863D9374271#AGRI.BD#177#TC:-1000.00#HUM:-1000.0#PRES:-1000.00#LUX:2
2018-05-15 07:21:22.614 - <=>?#3C147863D9374271#AGRI.BD#178#TC:-1000.00#HUM:-1000.0#PRES:-1000.00#LUX:2
2018-05-15 07:36:30.814 - <=>?#3C147863D9374271#AGRI.BD#179#TC:-1000.00#HUM:-1000.0#PRES:-1000.00#LUX:2
2018-05-15 07:51:38.513 - <=>?#3C147863D9374271#AGRI.BD#180#TC:-1000.00#HUM:-1000.0#PRES:-1000.00#LUX:2
2018-05-15 08:06:46.716 - <=>?#3C147863D9374271#AGRI.BD#181#TC:-1000.00#HUM:-1000.0#PRES:-1000.00#LUX:2
2018-05-15 08:21:54.937 - <=>?#3C147863D9374271#AGRI.BD#182#TC:-1000.00#HUM:-1000.0#PRES:-1000.00#LUX:2
2018-05-15 08:37:02.621 - <=>?#3C147863D9374271#AGRI.BD#183#TC:-1000.00#HUM:-1000.0#PRES:-1000.00#LUX:2
```

Figure 3.17: Libelium Sensor Data Logs
(Source: http://libelium.com/)

Below table explains the boxed fields.

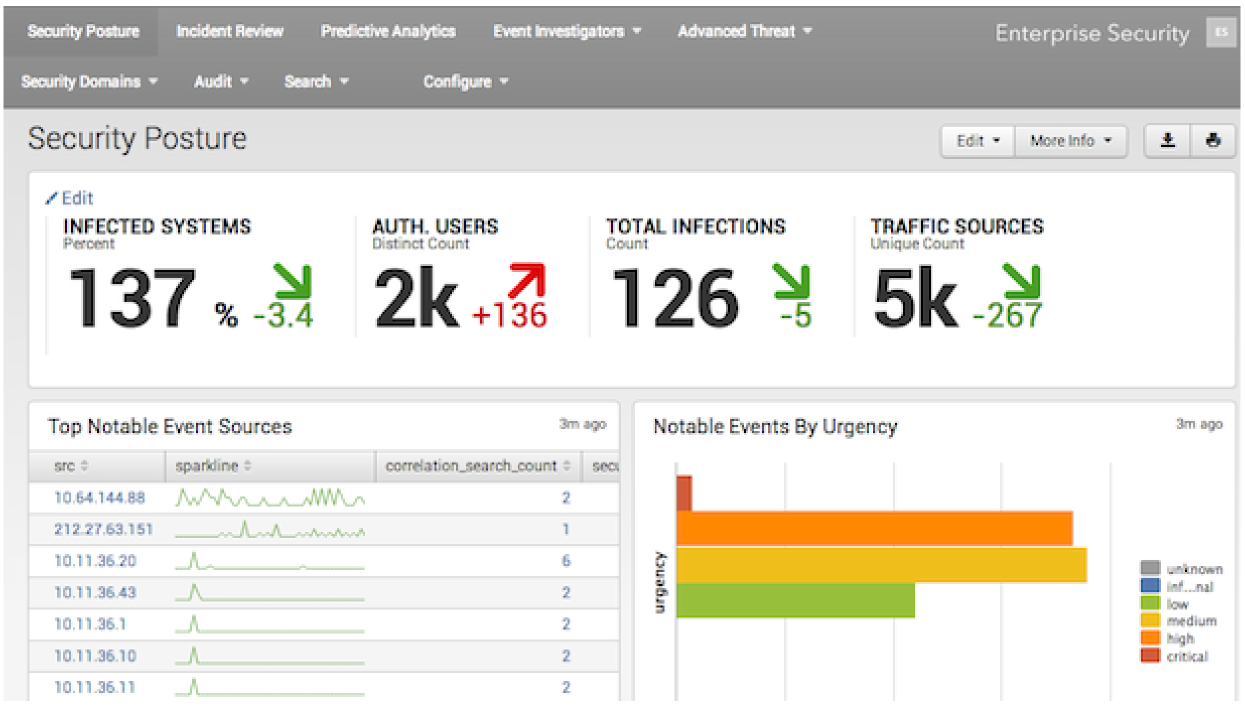
Table 3.8: Waspframe Explanation

Column No	Category	Example Value
1	Time Stamp	2018-05-15 06:11:31.227
2	Plug & Sense Secret ID	20737063D93742B6
3	Plug & Sense ID	node_id
4	Sensor ID and Sensor Value	ACC:-13;-22;1050#BAT:100

3.7 Splunk IOT Cloud Platform

Splunk is a software platform to search, analyze and visualize the machine-generated data gathered from the websites, applications, sensors, devices etc. Splunk is capable of capturing machine/ sensor data from multiple sources in various data formats which can be processed and be visualized.

Figure 3.18 showcases the powerful visualization capabilities of the Splunk application.



src	sparkline	correlation_search_count
10.64.144.88		2
212.27.63.151		1
10.11.36.20		6
10.11.36.43		2
10.11.36.1		2
10.11.36.10		2
10.11.36.11		2

Figure 3.18: Splunk Visualizations

3.8 Alerts and Notifications

One of the two main functionalities of the proposed prototype gateway is the ability to send alerts and notifications from the gateway itself which is not commonly seen in IOT gateways, however will add much more value to it considering most of the real-world deployments. The prototype developed shall be capable of sending alerts via multiple methods such that the dependency on each method is reduced. Further the prototype shall be capable of defining what type of notifications and alerts to be sent on each method.

The selected techniques are,

- *Email alerts:* To send scheduled logs and reports on the data collected from sensor nodes
- *SMS alerts:* To send real-time alerts for immediate actions
- *App notifications:* To send real-time alerts for immediate actions

Figure 3.19 illustrates the proposed setup on sending notifications.

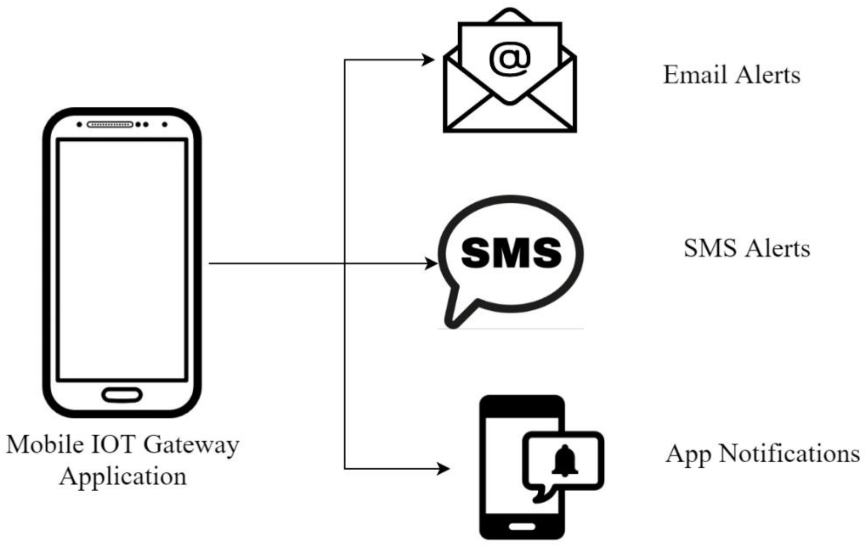
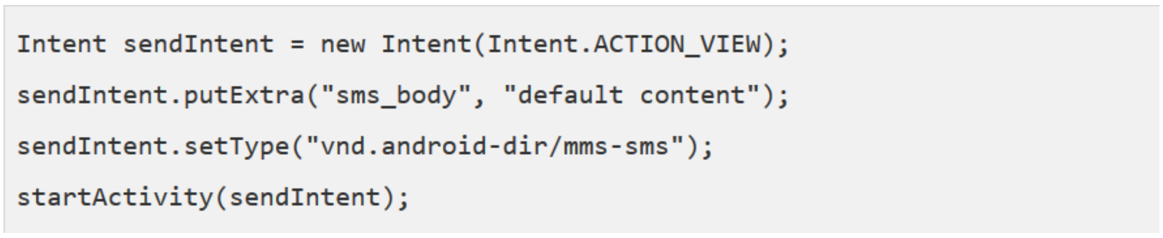


Figure 3.19: Alert and Notifications

3.8.1 SMS Alerts

The SmsManager API or built-in SMS applications can be used to send SMS from Android devices. The SmsManager API will be used in the prototype to send SMS based on the filtered data.

Basically, below method shall be used in the prototype android app to enable this feature.



```
Intent sendIntent = new Intent(Intent.ACTION_VIEW);
sendIntent.putExtra("sms_body", "default content");
sendIntent.setType("vnd.android-dir/mms-sms");
startActivity(sendIntent);
```

Figure 3.20: SmsManager Method

SMS alerts shall be used to send immediate and critical information to remote users who are pre-defined or ad hoc based in the gateway. As the application is capable of identifying any unusual values among the filtered data, the users can define the type alerts they want to be alerted with such as, sudden fluctuations in temperature, humidity and luminosity etc.

3.8.2 Email Alerts

This technique shall be used in sending scheduled reports on sensor activity, logs such as gateway user logs and connection logs etc. An existing email client will be used in the prototype application to send emails to pre-defined or ad hoc email clients. To do that an ‘activity’ needs to be written in the code which launches one of the email clients installed the mobile gateway. An ‘intent object’ is used to write this activity as shown in Figure 3.21 below.

```
Intent emailIntent = new Intent(Intent.ACTION_SEND);
```

```
emailIntent.setData(Uri.parse("mailto:"));
emailIntent.setType("text/plain");
```

Figure 3.21: Email Method

3.8.3 App Notifications

Google Cloud Messaging service is used to create app alerts from the prototype application to alert any critical alerts similar to the SMS alerts. This is a free service that enables developers to send messages between servers and client apps. This includes downstream messages from servers to client apps, and upstream messages from client apps to servers [20].

For example, a lightweight downstream message could inform a client app that there is new data to be fetched from the server, as in the case of a "new email" notification. For use cases such as instant messaging, a GCM message can transfer up to 4kb of payload to the client app. The GCM service handles all aspects of queueing of messages and delivery to and from the target client app [20].

A GCM implementation includes a Google connection server, an app server in your environment that interacts with the connection server via HTTP or XMPP protocol, and a client app [20].

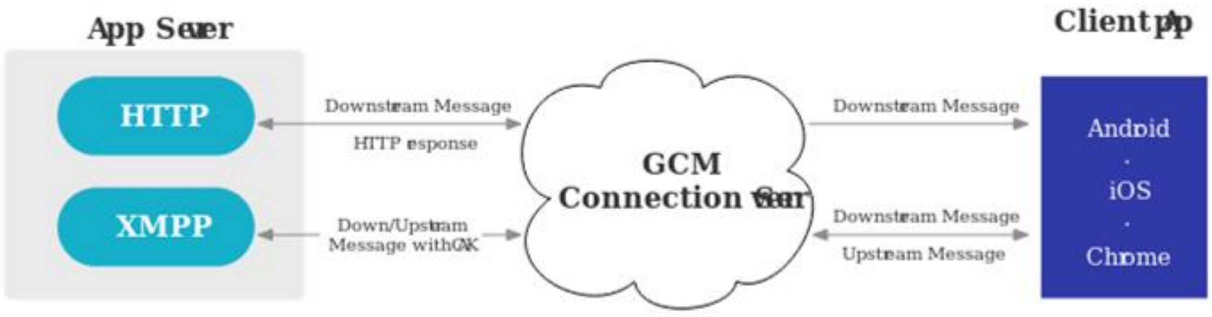


Figure 3.12: GCM Architecture
(Source: https://developers.google.com/cloud-messaging/gcm)

- Google **GCM Connection Servers** accept downstream messages from your app server and send them to a client app. The XMPP connection server can also accept messages sent upstream from the client app and forward them to your app server[20].
- On your **App Server**, you implement the HTTP and/or XMPP protocol to communicate with the GCM connection server(s). App servers send downstream messages to a GCM connection server; the connection server enqueues and stores the message, and then sends it to the client app. If you implement XMPP, your app server can receive messages sent from the client app [20].
- The **Client App** is a GCM-enabled client app. To receive and send GCM messages, this app must register with GCM and get a unique identifier called a registration token[20].

Chapter 4

4 RESEARCH IMPLEMENTATION

4.1 Introduction

This chapter discusses the prototype implementation based on the solution architecture developed above and following the methodology described in previous chapters. The actual components used, and the technologies involved will be discussed in this chapter aligning with the component setup as illustrated in Figure 4.1 below.

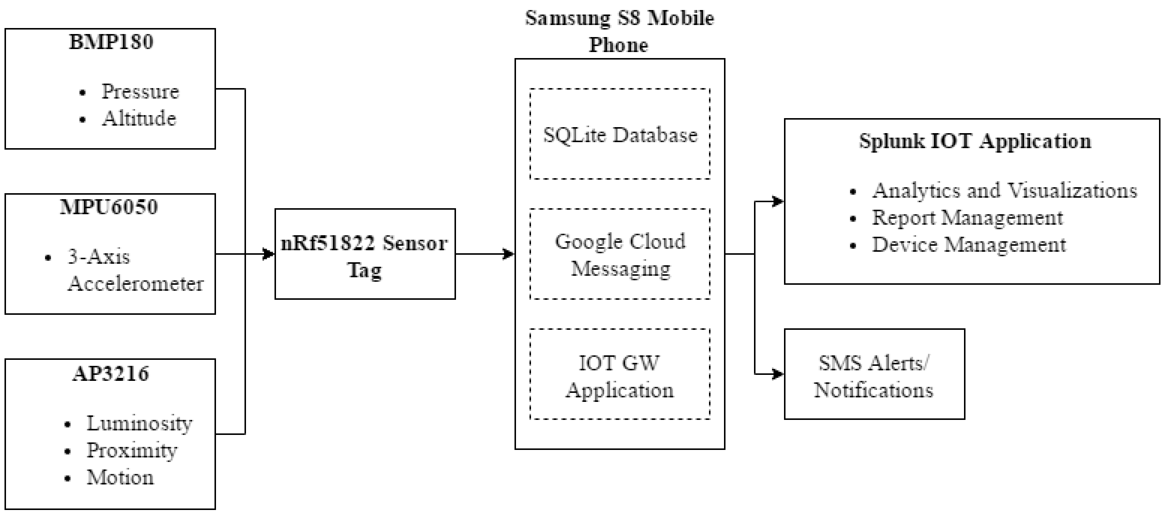


Figure 4.1: Implementation Setup

The implementation will be discussed based on five key steps identified in the development process such as,

- Step 1: Collecting data
- Step 2: Processing and filtering data
- Step 3: Logging filtered data
- Step 4: Alerting and Sending notifications
- Step 5: Analyzing and Visualizing data

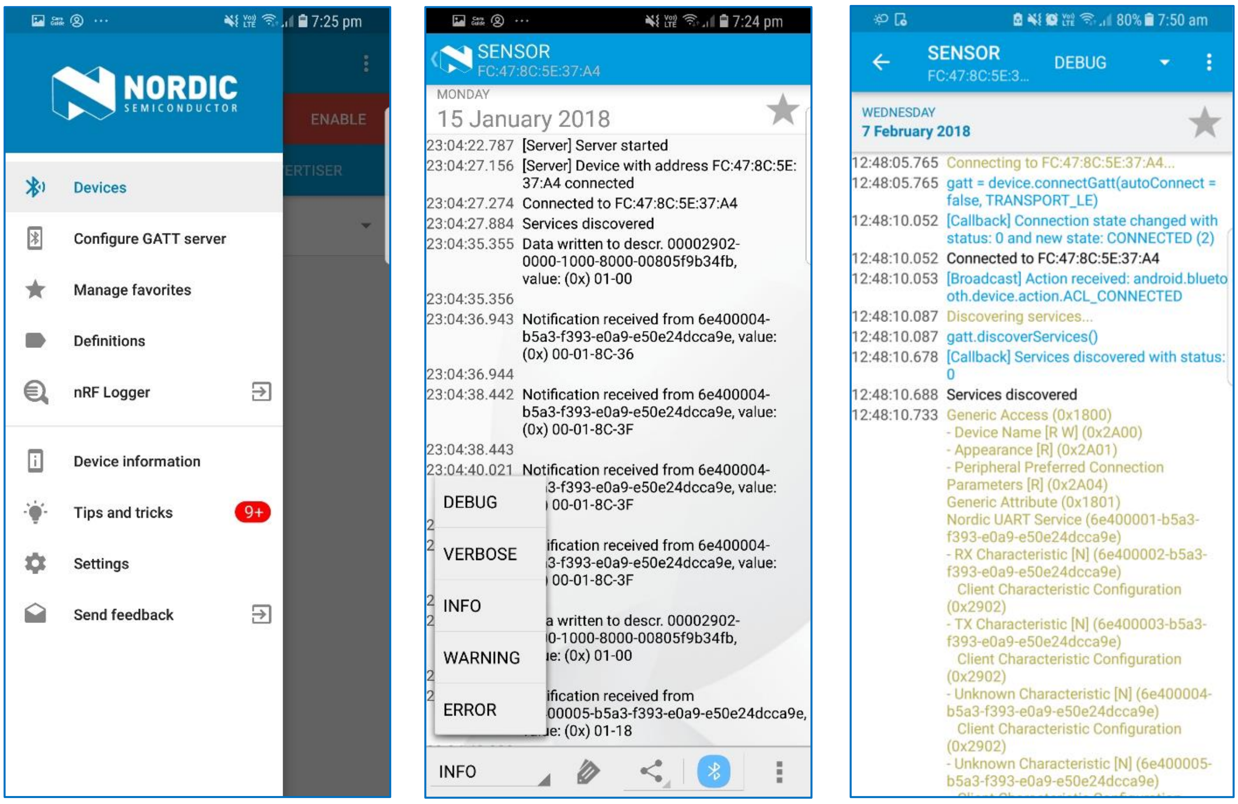
4.2 Step 1: Collecting Data

Collecting the data generated from the nRF51822 sensor tag is the first step identified in the development process. As the sensor tag uses BLE 4.0 wireless network technology to transfer sensor data from the sensor chips, a mobile device which supports BLE 4.0 or above had to be used. Therefore, Samsung S8 Android mobile phone with latest Bluetooth 5.0 version was used in this prototype development.

4.2.1 BLE Scanner Android Application

nRF Connect BLE Scanning Android Application was used as the app [23] to detect and connect with the sensor tag which has the nRF logger application in-built to capture the readings from nRF chipsets.

Another main reason to use the nRF logger application is that the availability of an API [24] which allows the users to log customized logs to the nRF database.



```
23:04:22.787 [Server] Server started
23:04:27.156 [Server] Device with address FC:47:8C:5E:
             37:A4 connected
23:04:27.274 Connected to FC:47:8C:5E:37:A4
23:04:27.884 Services discovered
23:04:35.355 Data written to descr. 00002902-
             0000-1000-8000-00805f9b34fb,
             value: (0x) 01-00
23:04:35.356
23:04:36.943 Notification received from 6e400004-
             b5a3-f393-e0a9-e50e24dcca9e, value:
             (0x) 00-01-8C-36
23:04:36.944
23:04:38.442 Notification received from 6e400004-
             b5a3-f393-e0a9-e50e24dcca9e, value:
             (0x) 00-01-8C-3F
23:04:38.443
23:04:40.021 Notification received from 6e400004-
              3-f393-e0a9-e50e24dcca9e, value:
              00-01-8C-3F
2
2             ification received from 6e400004-
              3-f393-e0a9-e50e24dcca9e, value:
              00-01-8C-3F
2
2             a written to descr. 00002902-
              0-1000-8000-00805f9b34fb,
              e: (0x) 01-00
2
2             ification received from
              00005-b5a3-f393-e0a9-e50e24dcca9e,
              e: (0x) 01-18
```

```
12:48:05.765 Connecting to FC:47:8C:5E:37:A4...
12:48:05.765 gatt = device.connectGatt(autoConnect =
             false, TRANSPORT_LE)
12:48:10.052 [Callback] Connection state changed with
             status: 0 and new state: CONNECTED (2)
12:48:10.052 Connected to FC:47:8C:5E:37:A4
12:48:10.053 [Broadcast] Action received: android.blueto
             oth.device.action.ACL_CONNECTED
12:48:10.087 Discovering services...
12:48:10.087 gatt.discoverServices()
12:48:10.678 [Callback] Services discovered with status:
             0
12:48:10.688 Services discovered
12:48:10.733 Generic Access (0x1800)
             - Device Name [R W] (0x2A00)
             - Appearance [R] (0x2A01)
             - Peripheral Preferred Connection
             Parameters [R] (0x2A04)
             Generic Attribute (0x1801)
             Nordic UART Service (6e400001-b5a3-
             f393-e0a9-e50e24dcca9e)
             - RX Characteristic [N] (6e400002-b5a3-
             f393-e0a9-e50e24dcca9e)
               Client Characteristic Configuration
             (0x2902)
             - TX Characteristic [N] (6e400003-b5a3-
             f393-e0a9-e50e24dcca9e)
               Client Characteristic Configuration
             (0x2902)
             - Unknown Characteristic [N] (6e400004-
             b5a3-f393-e0a9-e50e24dcca9e)
               Client Characteristic Configuration
             (0x2902)
             - Unknown Characteristic [N] (6e400005-
             b5a3-f393-e0a9-e50e24dcca9e)
```

Figure 4.2: nRF Connect Android App
(Source: https://www.nordicsemi.com/eng/Products/Nordic-mobile-Apps/nRF-Connect-for-Mobile)

Further this enables developers to debug their apps even not connected to the computer. Further log sessions may be also shared by customers to obtain detailed information which is being used in this prototype development process. As shown in Figure 3.18 above, this logger application supports six log levels such as,

- Debug
- Verbose

- Info
- Application
- Warning
- Error

The sensor data has been logged under ‘Debug’ logs as shown in Figure 3.18 which has been exported for data processing.

4.3 Step 2: Processing and filtering data

A python-based script is used in the prototype to extract the data from the data logs captured at the IOT gateway applications.

4.3.1 nRF51 Sensor data log

nRF51 tag provides five sensor values such as *acceleration, gyroscope, atmospheric pressure, temperature and the ambient light information*. Once the sensor tag is active and connected, all the information can be exported in one GATT file. These raw data values are in HEX numerical format with a proprietary template which cannot be easily translated into meaningful data by a traditional IOT platform.

Therefore, a Python based application was developed to translate these data in the GATT profile into readable and meaningful information. The UUID of each sensor was used to segment the data chunks into five categories which were later converted into readable values. Below table 4.1 contains the UUIDs of each sensor of the sensor tag used.

Table 4.1: Sensor UUIDs

Service Type	UUID
Accelerometer	6e400002-b5a3-f393-e0a9-e50e24dcca9e
Gyroscope	6e400003-b5a3-f393-e0a9-e50e24dcca9e
Atmospheric Pressure	6e400004-b5a3-f393-e0a9-e50e24dcca9e
Temperature	6e400005-b5a3-f393-e0a9-e50e24dcca9e
Ambient Light	6e400006-b5a3-f393-e0a9-e50e24dcca9e

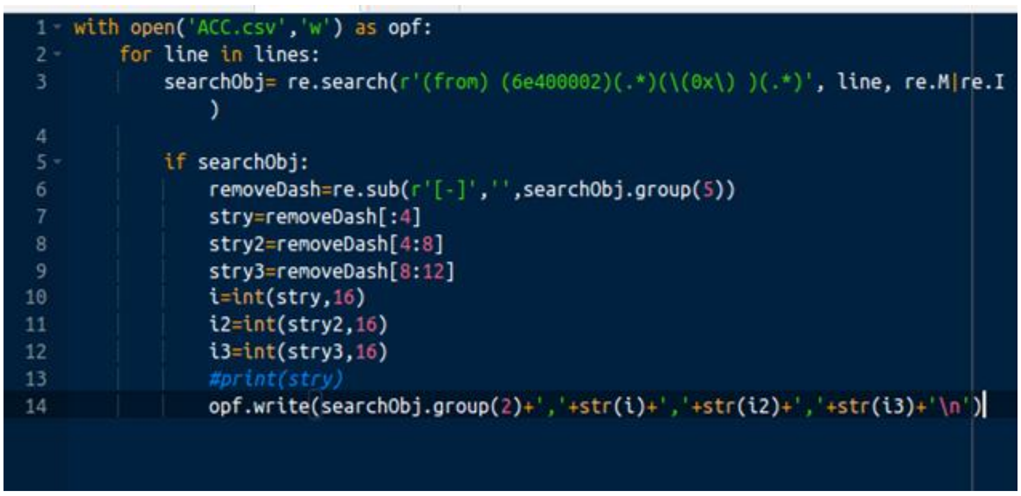
And this Python application created five excel files to five services/ sensor types.

If considered the accelerometer and gyroscope data as an example where the data has been sent in 6 bytes (three 16-bit values),

- Accelerometer value: 38-37-38-F0-0C-E5

$$\underbrace{38\text{-}37}_{X}\ \underbrace{38\text{-}F0}_{Y}\ \underbrace{0C\text{-}E5}_{Z}$$

AXIS	HEX	DECIMAL
X	0x3837	14391
Y	0x38F0	14576
Z	0x0CE5	3301

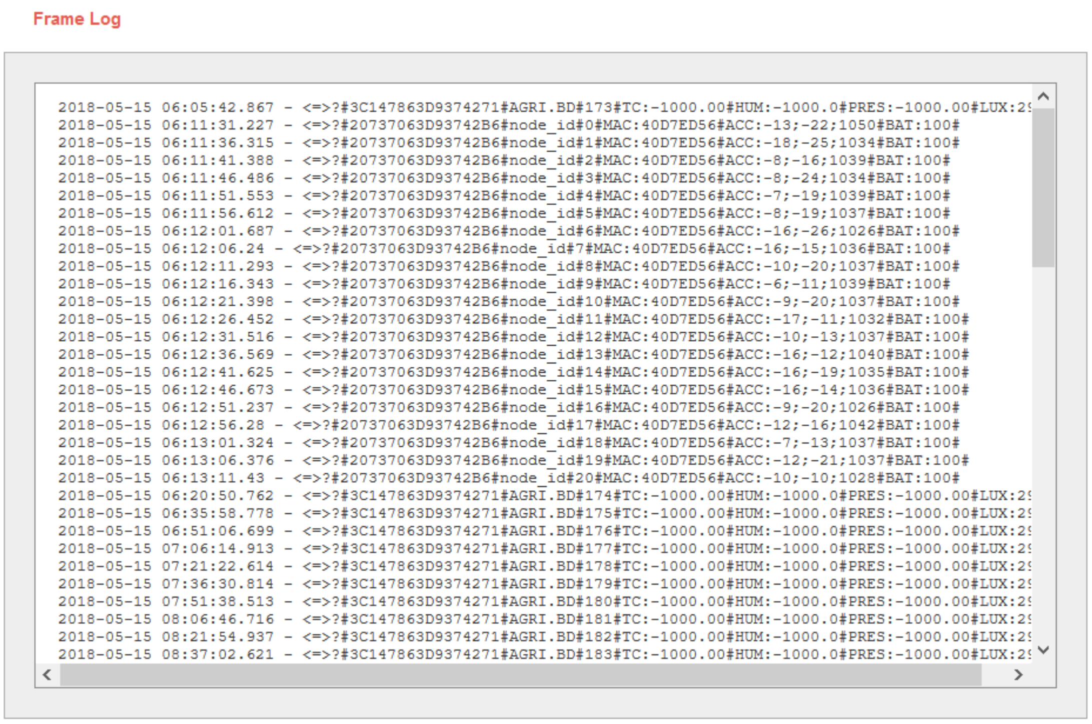


```python
with open('ACC.csv','w') as opf:
    for line in lines:
        searchObj= re.search(r'(from) (6e400002)(.*)(\(0x\) )(.*)', line, re.M|re.I
            )

        if searchObj:
            removeDash=re.sub(r'[-]','',searchObj.group(5))
            stry=removeDash[:4]
            stry2=removeDash[4:8]
            stry3=removeDash[8:12]
            i=int(stry,16)
            i2=int(stry2,16)
            i3=int(stry3,16)
            #print(stry)
            opf.write(searchObj.group(2)+','+str(i)+','+str(i2)+','+str(i3)+'\n')
```

Figure 4.3: Python Code

4.3.2 Libelium Sensor data log

As shown in Table 4.2 the Libelium sensor data log consists data from three plug and sense nodes connected to multiple sensors and the python script translates the data which are in Waspframes to readable meaningful information. Figure 4.4 shows a sample of logs which were received to the Meshlium gateway in Waspframes. These logs were used in the Python Script as the prototype gateway is not capable of accepting data from Libelium products due to the limitations in the connectivity technologies.


Frame Log

```
2018-05-15 06:05:42.867 - <=>?#3C147863D9374271#AGRI.BD#173#TC:-1000.00#HUM:-1000.0#PRES:-1000.00#LUX:2
2018-05-15 06:11:31.227 - <=>?#20737063D93742B6#node_id#0#MAC:40D7ED56#ACC:-13;-22;1050#BAT:100#
2018-05-15 06:11:36.315 - <=>?#20737063D93742B6#node_id#1#MAC:40D7ED56#ACC:-18;-25;1034#BAT:100#
2018-05-15 06:11:41.388 - <=>?#20737063D93742B6#node_id#2#MAC:40D7ED56#ACC:-8;-16;1039#BAT:100#
2018-05-15 06:11:46.486 - <=>?#20737063D93742B6#node_id#3#MAC:40D7ED56#ACC:-8;-24;1034#BAT:100#
2018-05-15 06:11:51.553 - <=>?#20737063D93742B6#node_id#4#MAC:40D7ED56#ACC:-7;-19;1039#BAT:100#
2018-05-15 06:11:56.612 - <=>?#20737063D93742B6#node_id#5#MAC:40D7ED56#ACC:-8;-19;1037#BAT:100#
2018-05-15 06:12:01.687 - <=>?#20737063D93742B6#node_id#6#MAC:40D7ED56#ACC:-16;-26;1026#BAT:100#
2018-05-15 06:12:06.24 - <=>?#20737063D93742B6#node_id#7#MAC:40D7ED56#ACC:-16;-15;1036#BAT:100#
2018-05-15 06:12:11.293 - <=>?#20737063D93742B6#node_id#8#MAC:40D7ED56#ACC:-10;-20;1037#BAT:100#
2018-05-15 06:12:16.343 - <=>?#20737063D93742B6#node_id#9#MAC:40D7ED56#ACC:-6;-11;1039#BAT:100#
2018-05-15 06:12:21.398 - <=>?#20737063D93742B6#node_id#10#MAC:40D7ED56#ACC:-9;-20;1037#BAT:100#
2018-05-15 06:12:26.452 - <=>?#20737063D93742B6#node_id#11#MAC:40D7ED56#ACC:-17;-11;1032#BAT:100#
2018-05-15 06:12:31.516 - <=>?#20737063D93742B6#node_id#12#MAC:40D7ED56#ACC:-10;-13;1037#BAT:100#
2018-05-15 06:12:36.569 - <=>?#20737063D93742B6#node_id#13#MAC:40D7ED56#ACC:-16;-12;1040#BAT:100#
2018-05-15 06:12:41.625 - <=>?#20737063D93742B6#node_id#14#MAC:40D7ED56#ACC:-16;-19;1035#BAT:100#
2018-05-15 06:12:46.673 - <=>?#20737063D93742B6#node_id#15#MAC:40D7ED56#ACC:-16;-14;1036#BAT:100#
2018-05-15 06:12:51.237 - <=>?#20737063D93742B6#node_id#16#MAC:40D7ED56#ACC:-9;-20;1026#BAT:100#
2018-05-15 06:12:56.28 - <=>?#20737063D93742B6#node_id#17#MAC:40D7ED56#ACC:-12;-16;1042#BAT:100#
2018-05-15 06:13:01.324 - <=>?#20737063D93742B6#node_id#18#MAC:40D7ED56#ACC:-7;-13;1037#BAT:100#
2018-05-15 06:13:06.376 - <=>?#20737063D93742B6#node_id#19#MAC:40D7ED56#ACC:-12;-21;1037#BAT:100#
2018-05-15 06:13:11.43 - <=>?#20737063D93742B6#node_id#20#MAC:40D7ED56#ACC:-10;-10;1028#BAT:100#
2018-05-15 06:20:50.762 - <=>?#3C147863D9374271#AGRI.BD#174#TC:-1000.00#HUM:-1000.0#PRES:-1000.00#LUX:2
2018-05-15 06:35:58.778 - <=>?#3C147863D9374271#AGRI.BD#175#TC:-1000.00#HUM:-1000.0#PRES:-1000.00#LUX:2
2018-05-15 06:51:06.699 - <=>?#3C147863D9374271#AGRI.BD#176#TC:-1000.00#HUM:-1000.0#PRES:-1000.00#LUX:2
2018-05-15 07:06:14.913 - <=>?#3C147863D9374271#AGRI.BD#177#TC:-1000.00#HUM:-1000.0#PRES:-1000.00#LUX:2
2018-05-15 07:21:22.614 - <=>?#3C147863D9374271#AGRI.BD#178#TC:-1000.00#HUM:-1000.0#PRES:-1000.00#LUX:2
2018-05-15 07:36:30.814 - <=>?#3C147863D9374271#AGRI.BD#179#TC:-1000.00#HUM:-1000.0#PRES:-1000.00#LUX:2
2018-05-15 07:51:38.513 - <=>?#3C147863D9374271#AGRI.BD#180#TC:-1000.00#HUM:-1000.0#PRES:-1000.00#LUX:2
2018-05-15 08:06:46.716 - <=>?#3C147863D9374271#AGRI.BD#181#TC:-1000.00#HUM:-1000.0#PRES:-1000.00#LUX:2
2018-05-15 08:21:54.937 - <=>?#3C147863D9374271#AGRI.BD#182#TC:-1000.00#HUM:-1000.0#PRES:-1000.00#LUX:2
2018-05-15 08:37:02.621 - <=>?#3C147863D9374271#AGRI.BD#183#TC:-1000.00#HUM:-1000.0#PRES:-1000.00#LUX:2
```

Figure 4.4: ASCII Logs

As the first step, the waspframes were filtered based on the Wasp secret ID of the Plug & Sense to isolate the data coming from different nodes. Secondly the data was filtered based on the sensor to capture the sensor values. Below table illustrates secret IDs against the Plug & Sense nodes and the relevant sensors.

Table 4.2: Sample Sensor Data

Wasp Secret ID	Plug & Sense Node	Sensor	Sensor Value
20737063D93742B6	Smart Cities	BAT	96
20737063D93742B6	Smart Cities	ACC	-16;-15;1036
3C147863D9374271	Smart Agriculture	TC	24.18
3C147863D9374271	Smart Agriculture	HUM	34.6
3C147863D9374271	Smart Agriculture	PRES	100974.7
3C147863D9374271	Smart Agriculture	LUX	64
3C147863D9374271	Smart Agriculture	BAT	96
3C147863D9374271	Smart Agriculture	ACC	8;-146;1039
5A2C7063D93742AA	Smart Environment PRO	NO2	11.779
5A2C7063D93742AA	Smart Environment PRO	O3	8129.225
5A2C7063D93742AA	Smart Environment PRO	CO	129.361
5A2C7063D93742AA	Smart Environment PRO	TC	22.54

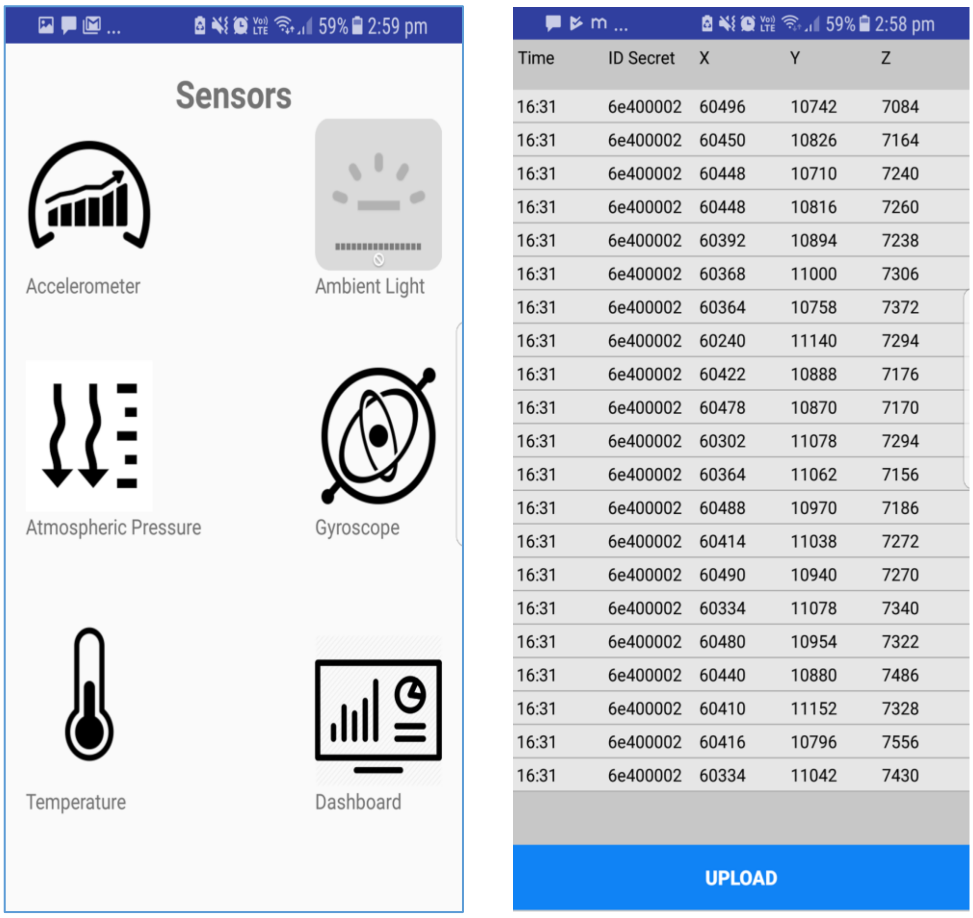
Wasp Secret ID	Plug & Sense Node	Sensor	Sensor Value
5A2C7063D93742AA	Smart Environment PRO	HUM	36.4
5A2C7063D93742AA	Smart Environment PRO	PRES	99589.66
5E6167057C10547E	Smart Cities PRO	BAT	3
5E6167057C10547E	Smart Cities PRO	ACC	-24;-39;969

4.4 Step 3: Logging filtered data

The Python application would automatically store the converted and filtered sensor values in five .csv files which were logged again in an android application. This application was used to carryout further actions using the readable sensor values.

As shown in Figure 4.5, there are five sensor icons in the application to import the accelerometer, ambient light, atmospheric pressure, gyroscope and temperature sensor values. User can select each sensor by clicking on it. Then the user will be redirected to a second interface where user can upload the .csv file generated from the Python application.

Sensor information will be loaded into a grid and a popup notification "*Data Imported*" will pop-up to acknowledge the successful import.

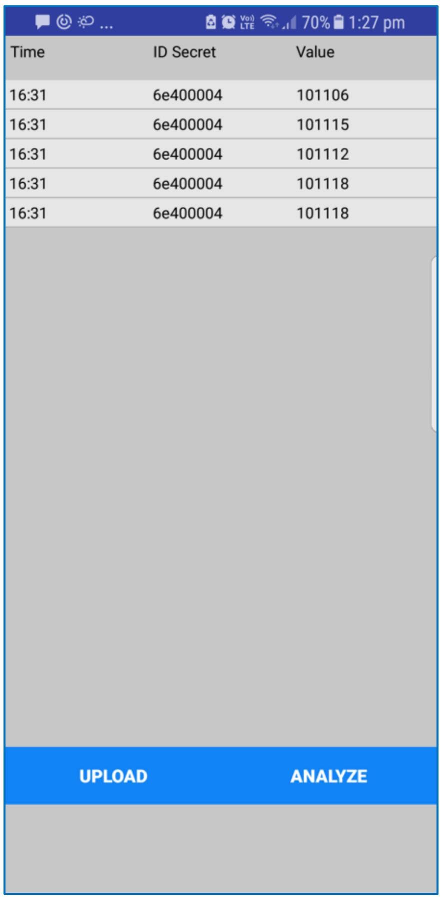


Time	ID Secret	X	Y	Z
16:31	6e400002	60496	10742	7084
16:31	6e400002	60450	10826	7164
16:31	6e400002	60448	10710	7240
16:31	6e400002	60448	10816	7260
16:31	6e400002	60392	10894	7238
16:31	6e400002	60368	11000	7306
16:31	6e400002	60364	10758	7372
16:31	6e400002	60240	11140	7294
16:31	6e400002	60422	10888	7176
16:31	6e400002	60478	10870	7170
16:31	6e400002	60302	11078	7294
16:31	6e400002	60364	11062	7156
16:31	6e400002	60488	10970	7186
16:31	6e400002	60414	11038	7272
16:31	6e400002	60490	10940	7270
16:31	6e400002	60334	11078	7340
16:31	6e400002	60480	10954	7322
16:31	6e400002	60440	10880	7486
16:31	6e400002	60410	11152	7328
16:31	6e400002	60416	10796	7556
16:31	6e400002	60334	11042	7430

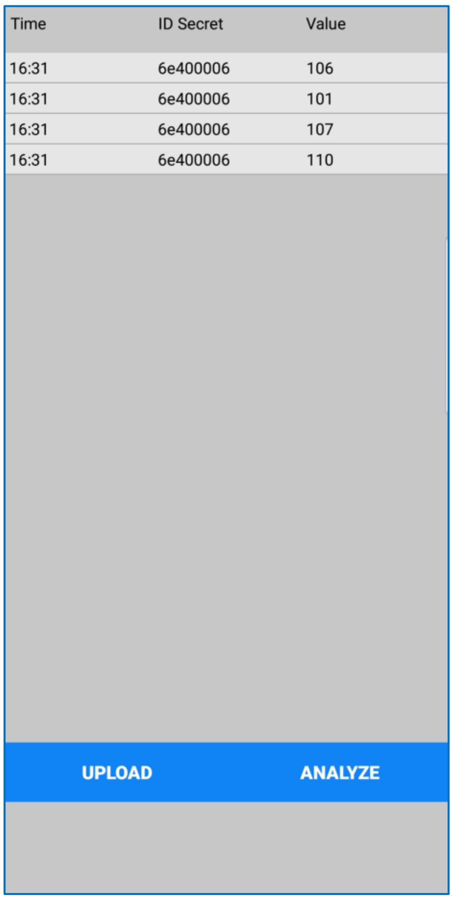
Figure 4.5: Sensor Data Processing App

4.5 Step 4: Alerting and Sending notifications

In addition to data evaluation, this application is equipped with sending alerts and notifications via three modes as discussed earlier. When evaluating the sensor values via the app, user will get two buttons for *upload* and *analyze* as shown in Figure 4.6.



Time	ID Secret	Value
16:31	6e400004	101106
16:31	6e400004	101115
16:31	6e400004	101112
16:31	6e400004	101118
16:31	6e400004	101118



Time	ID Secret	Value
16:31	6e400006	106
16:31	6e400006	101
16:31	6e400006	107
16:31	6e400006	110

Figure 4.6: Android App - Optional Actions

If the user wants to analyze the information user can click on the analyze button which will read the sensor values and if there is an any value which has exceeded the pre-defined accepted range, the app will prompt three options to send the alert as shown in Figure 4.7.

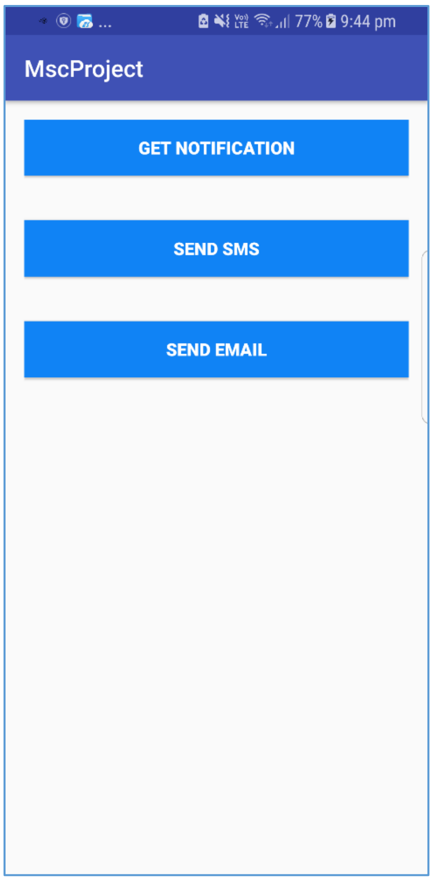
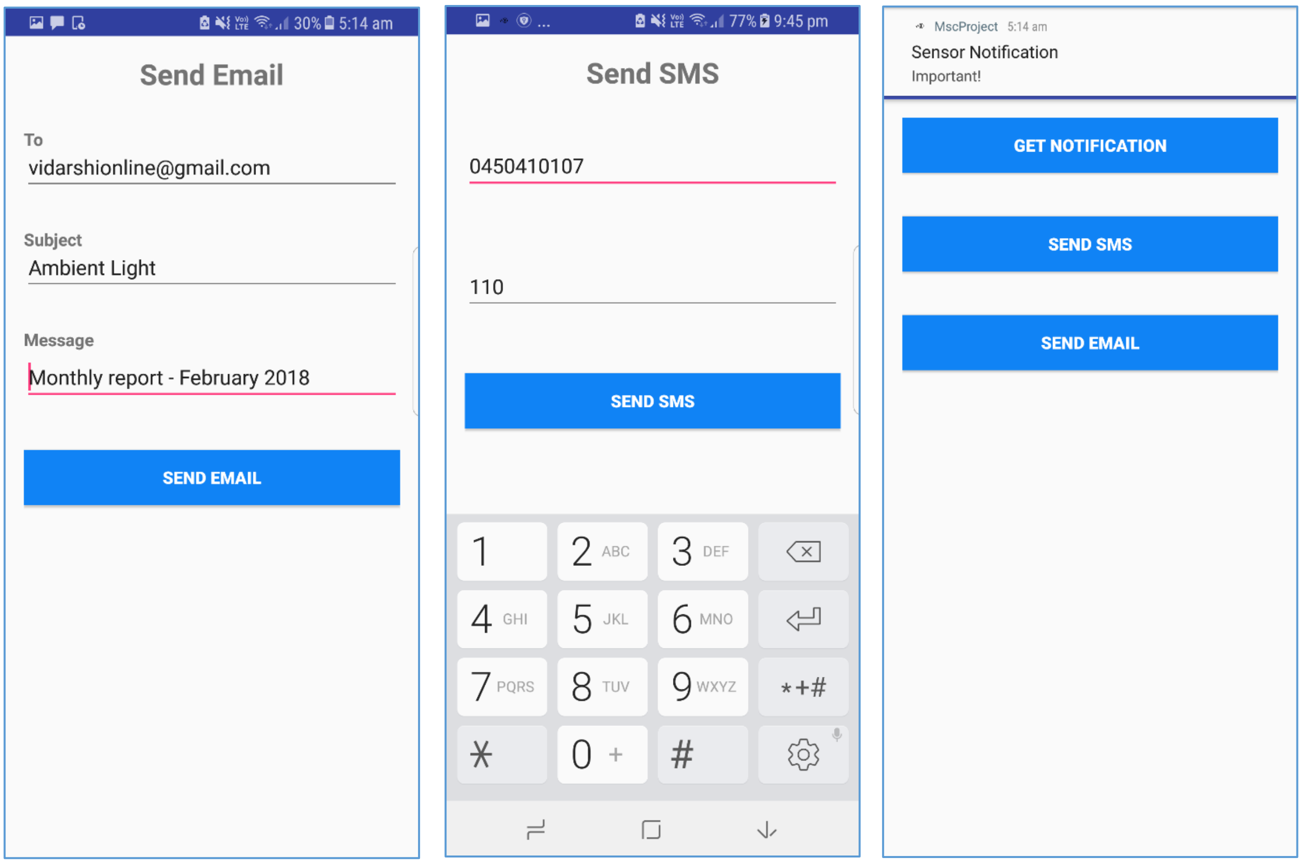


Figure 4.7: Android App – Alerts and Notifications

After that the user can select the preferred option from the three options given and proceed accordingly as shown in the figure 4.8 below.


Send Email

To
vidarshionline@gmail.com

Subject
Ambient Light

Message
Monthly report - February 2018

Send SMS

0450410107

110

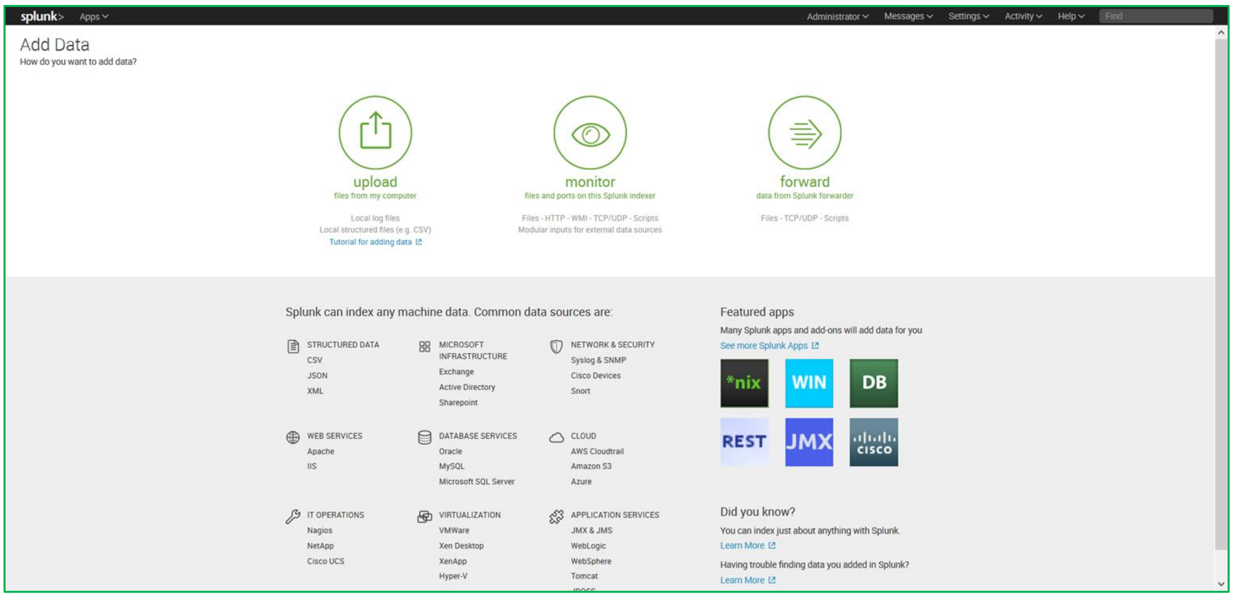
MscProject 5:14 am
Sensor Notification
Important!

Figure 4.8: Android App - Alerts and Notifications

4.6 Step 5: Analyzing and Visualizing data

As briefed in Methodology section, Splunk data analytics platform was used to analyze and visualize the sensor data which were processed and filtered from the android app and the Python application.

Step 1: When you open the Splunk home interface it has an option to add data. User can click on the icon and move to the next step.



Add Data

How do you want to add data?

upload
files from my computer
Local log files
Local structured files (e.g. CSV)
Tutorial for adding data

monitor
files and ports on this Splunk indexer
Files - HTTP - WMI - TCP/UDP - Scripts
Modular inputs for external data sources

forward
data from Splunk forwarder
Files - TCP/UDP - Scripts

Splunk can index any machine data. Common data sources are:

STRUCTURED DATA: CSV, JSON, XML
MICROSOFT INFRASTRUCTURE: Exchange, Active Directory, Sharepoint
NETWORK & SECURITY: Syslog & SNMP, Cisco Devices, Snort
WEB SERVICES: Apache, IIS
DATABASE SERVICES: Oracle, MySQL, Microsoft SQL Server
CLOUD: AWS Cloudtrail, Amazon S3, Azure
IT OPERATIONS: Nagios, NetApp, Cisco UCS
VIRTUALIZATION: VMWare, Xen Desktop, XenApp, Hyper-V
APPLICATION SERVICES: JMX & JMS, WebLogic, WebSphere, Tomcat

Featured apps

Many Splunk apps and add-ons will add data for you

See more Splunk Apps

Did you know?

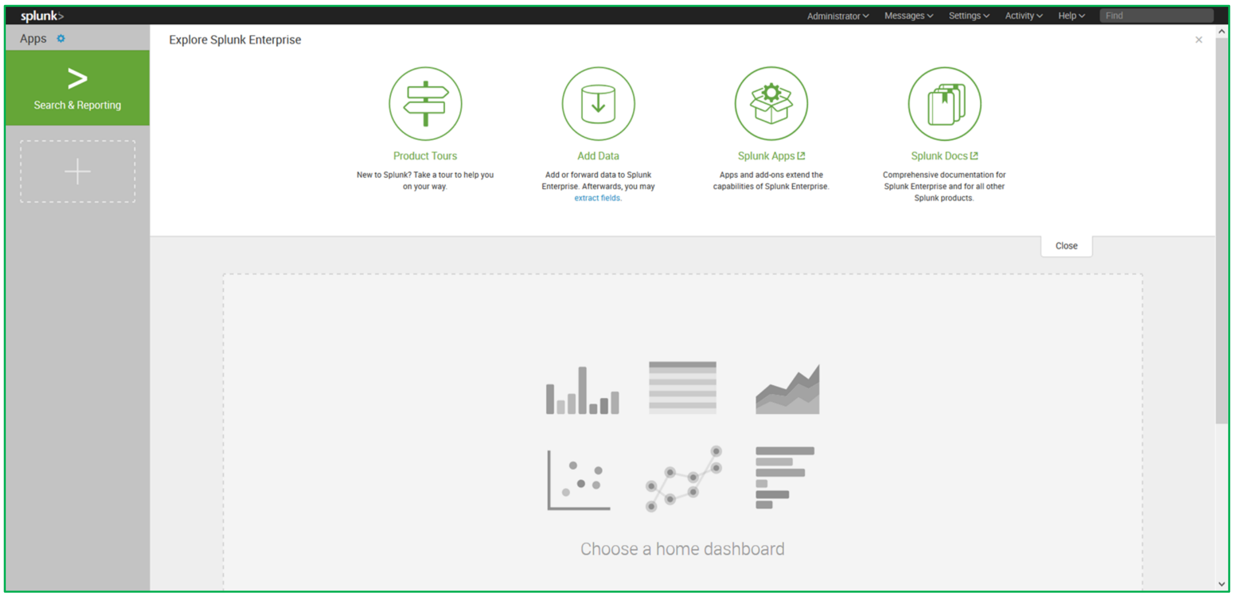
You can index just about anything with Splunk.
Learn More

Having trouble finding data you added in Splunk?
Learn More

Figure 4.9: Splunk Data Upload

Step 2: In the second interface user needs to click on the upload button and upload the .csv file or the txt file.

Even though a .csv upload is carried out in this prototype development, Splunk is capable of collecting data via various techniques such as MQTT, HTTP event collectors, database queries and real-time log file reading via Splunk agents etc.



New to Splunk? Take a tour to help you on your way.

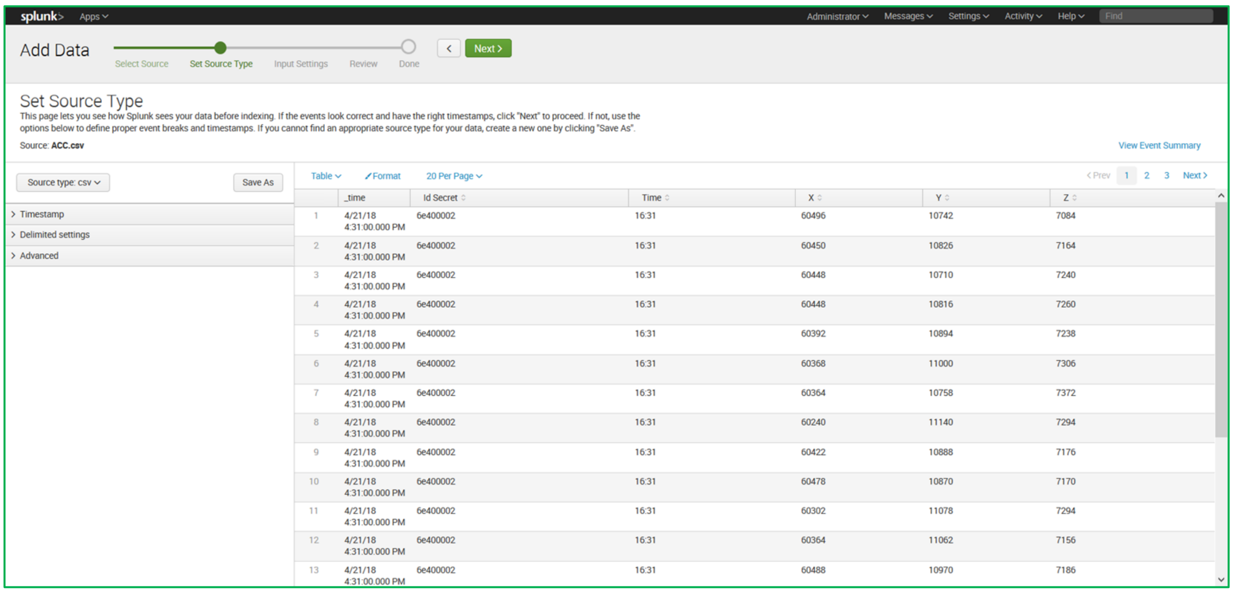
Add or forward data to Splunk Enterprise. Afterwards, you may extract fields.

Apps and add-ons extend the capabilities of Splunk Enterprise.

Comprehensive documentation for Splunk Enterprise and for all other Splunk products.

Figure 4.10: Splunk Data Upload

Step 3: The uploaded sensor data can be seen in the next interface.



Set Source Type

This page lets you see how Splunk sees your data before indexing. If the events look correct and have the right timestamps, click "Next" to proceed. If not, use the options below to define proper event breaks and timestamps. If you cannot find an appropriate source type for your data, create a new one by clicking "Save As".

Source: **ACC.csv**

	_time	Id Secret	Time	X	Y	Z
1	4/21/18 4:31:00.000 PM	6e400002	16:31	60496	10742	7084
2	4/21/18 4:31:00.000 PM	6e400002	16:31	60450	10826	7164
3	4/21/18 4:31:00.000 PM	6e400002	16:31	60448	10710	7240
4	4/21/18 4:31:00.000 PM	6e400002	16:31	60448	10816	7260
5	4/21/18 4:31:00.000 PM	6e400002	16:31	60392	10894	7238
6	4/21/18 4:31:00.000 PM	6e400002	16:31	60368	11000	7306
7	4/21/18 4:31:00.000 PM	6e400002	16:31	60364	10758	7372
8	4/21/18 4:31:00.000 PM	6e400002	16:31	60240	11140	7294
9	4/21/18 4:31:00.000 PM	6e400002	16:31	60422	10888	7176
10	4/21/18 4:31:00.000 PM	6e400002	16:31	60478	10870	7170
11	4/21/18 4:31:00.000 PM	6e400002	16:31	60302	11078	7294
12	4/21/18 4:31:00.000 PM	6e400002	16:31	60364	11062	7156
13	4/21/18 4:31:00.000 PM	6e400002	16:31	60488	10970	7186

Figure 4.11: Sensor data on Splunk

Step 4: Advanced analytics and visualizations can be carried out based on the uploaded sensor data.

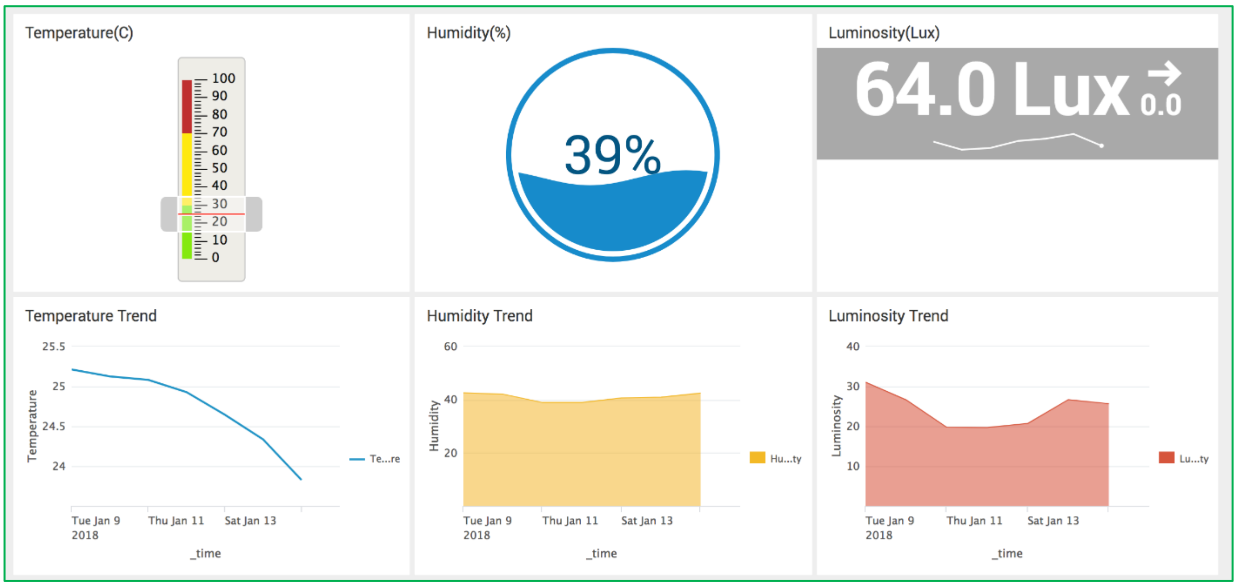


Figure 4.12: Splunk Visualizations

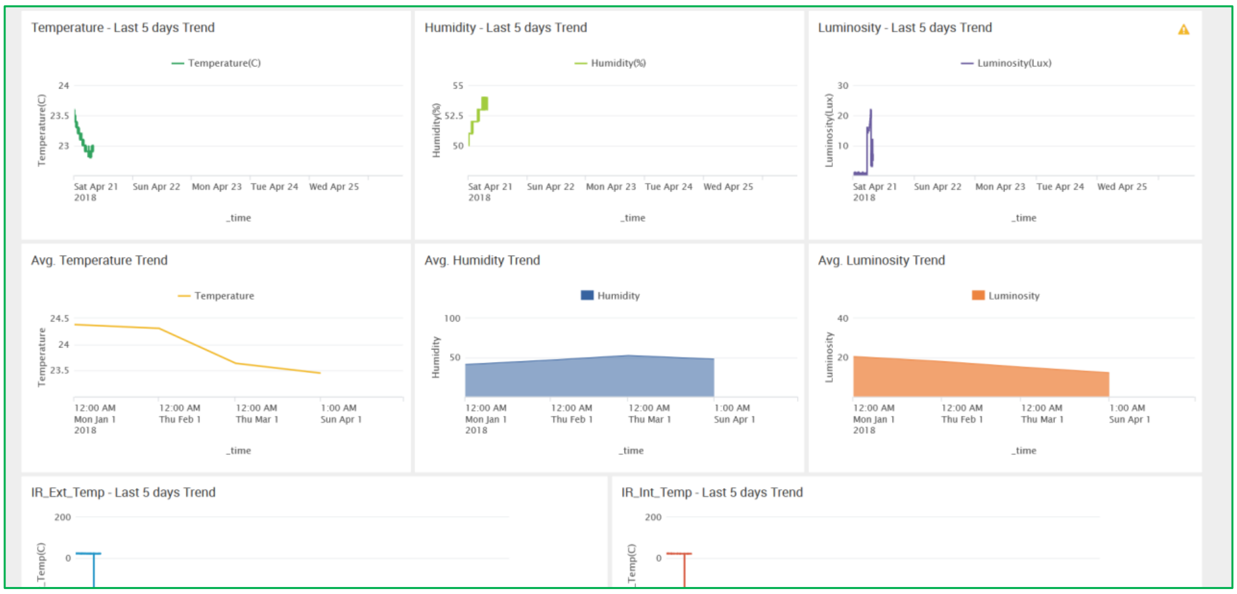


Figure 4.13: Splunk Visualizations

Chapter 5

5 RESEARCH EVALUATION

5.1 Introduction

This chapter evaluates the performance of the implemented prototype as explained in Chapter 4. Further it discusses the Pros and Cons of having a gateway with the features such as data filtering and real-time alerting against a traditional IOT gateway which connects the things with the IOT cloud platform. These were the key two challenges that were identified under research objective.

5.2 Data Filtering

As emphasized in Chapter 4, the prototyped gateway is capable of translating raw data into meaningful information by itself where as in a traditional IOT gateway scenario, all the raw data will be processed in the cloud platform.

There are two key competitive advantages identified when the data filtering happening in the gateway itself.

5.2.1 Cost

Multiple standard cost factors can be identified in a cloud-based deployment.

- *Number of API Calls*

The pricing model of most of the cloud platforms has a fee based on the number of API calls made with the cloud. Therefore, more API calls are made with the cloud, in other words more data are being sent to the cloud, the cost will increase. This cost can be drastically reduced by filtering the data pushed to the cloud which can be done by this prototype developed.

- *Resource Usage*

Even though the cloud platforms seem cheaper with compared to on-premise deployments, it becomes more expensive when the deployment is scaling up to cater the higher amount of data being processed. More processing power and storage space will be required which will improve the subscription drastically.

However, if the data is being filtered in a middle-node before the cloud similar to the prototype developed, this scaling up cost can be reduced. As shown in table 5.1, the total size of all five filtered files is significantly less than the actual raw file which are being used in the prototype development.

Table 5.1: Storage Space Calculation

Log File	Size (bytes)	Total Size (bytes)
Before Filtering		
- Raw log file		**21,143**
After Filtering		
- Accelerometer Log	1,474	
- ALS Log	80	
- Atmospheric Pressure Sensor Log	115	
- Gyroscope Log	1,433	
- Temperature Log	84	**3,186**

$$\begin{aligned} Cloud\ storage\ space\ saved\ (bytes) &= 21143 - 3186 \\ &= 17{,}957\ bytes \end{aligned}$$

$$\begin{aligned} Storage\ space\ saved\ (\%) &= \frac{17957}{21143} * 100 \\ &= \mathbf{84.93}\%\backslash \end{aligned}$$

- *Data Usage*

In most of the IOT deployments, the connectivity between the gateway and the cloud platform is setup via cellular networks. And more data is shared via the network, more data usage cost will incur. Considering a large scale IOT deployment with hundreds of sensors, continuously pushing data to the cloud, the data usage cost would be significant. If the data is filtered in the gateway and only usable data is being pushed, this cost shall be relatively very low which is available in the prototype development.

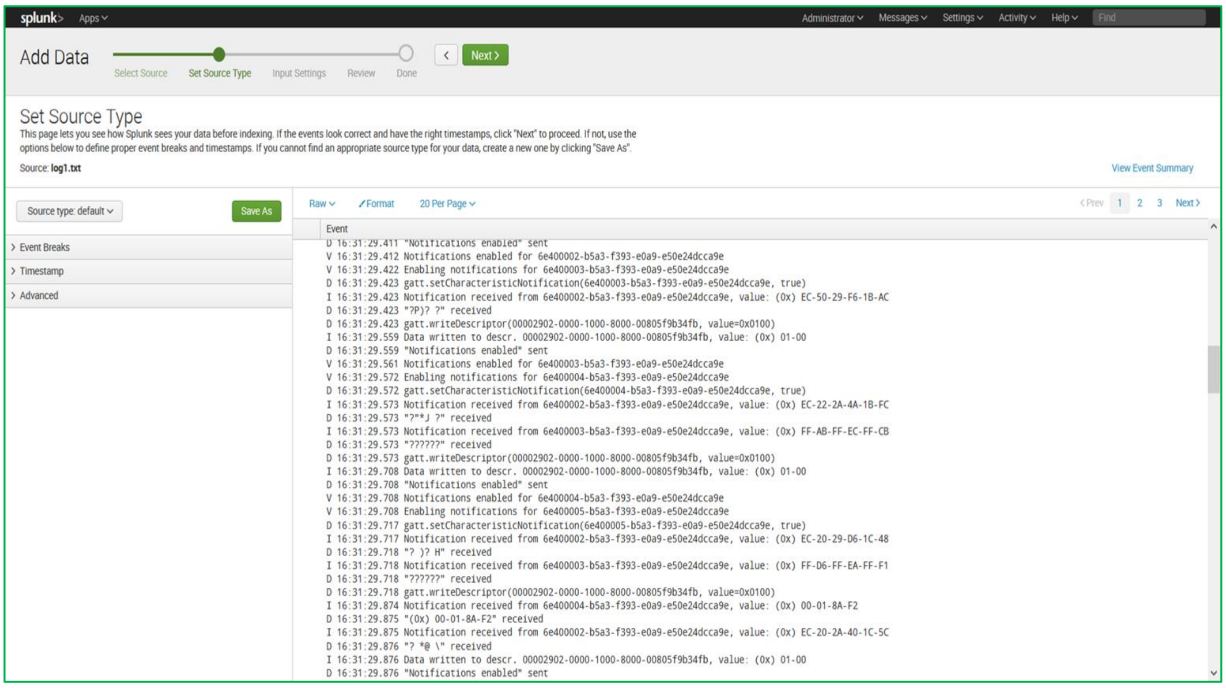
5.2.2 Latency

This is another key factor to be considered in any IOT deployment.

- *Data Processing*

As the amount of data being pushed to the cloud is increasing, more processing will be required to process them. If the processing power is not improved considering the higher cost, the latency will increase which will have major negative implications. However, if the data is filtered, less data will be pushed to the cloud which will be able to process data in real-time with minimal processing power consumption.

As shown in Figure 5.1 below, the Splunk interface will see the raw data in a very complex raw format which will take more time to understand and translate the data into more valuable information.



Set Source Type

This page lets you see how Splunk sees your data before indexing. If the events look correct and have the right timestamps, click "Next" to proceed. If not, use the options below to define proper event breaks and timestamps. If you cannot find an appropriate source type for your data, create a new one by clicking "Save As".

Source: log1.txt

```
D 16:31:29.411 "Notifications enabled" sent
V 16:31:29.412 Notifications enabled for 6e400002-b5a3-f393-e0a9-e50e24dcca9e
V 16:31:29.422 Enabling notifications for 6e400003-b5a3-f393-e0a9-e50e24dcca9e
D 16:31:29.423 gatt.setCharacteristicNotification(6e400003-b5a3-f393-e0a9-e50e24dcca9e, true)
I 16:31:29.423 Notification received from 6e400002-b5a3-f393-e0a9-e50e24dcca9e, value: (0x) EC-50-29-F6-1B-AC
D 16:31:29.423 "?P)? ?" received
D 16:31:29.423 gatt.writeDescriptor(00002902-0000-1000-8000-00805f9b34fb, value=0x0100)
I 16:31:29.559 Data written to descr. 00002902-0000-1000-8000-00805f9b34fb, value: (0x) 01-00
D 16:31:29.559 "Notifications enabled" sent
V 16:31:29.561 Notifications enabled for 6e400003-b5a3-f393-e0a9-e50e24dcca9e
V 16:31:29.572 Enabling notifications for 6e400004-b5a3-f393-e0a9-e50e24dcca9e
D 16:31:29.572 gatt.setCharacteristicNotification(6e400004-b5a3-f393-e0a9-e50e24dcca9e, true)
I 16:31:29.573 Notification received from 6e400002-b5a3-f393-e0a9-e50e24dcca9e, value: (0x) EC-22-2A-4A-1B-FC
D 16:31:29.573 "?"*J ?" received
I 16:31:29.573 Notification received from 6e400003-b5a3-f393-e0a9-e50e24dcca9e, value: (0x) FF-AB-FF-EC-FF-CB
D 16:31:29.573 "??????" received
D 16:31:29.573 gatt.writeDescriptor(00002902-0000-1000-8000-00805f9b34fb, value=0x0100)
I 16:31:29.708 Data written to descr. 00002902-0000-1000-8000-00805f9b34fb, value: (0x) 01-00
D 16:31:29.708 "Notifications enabled" sent
V 16:31:29.708 Notifications enabled for 6e400004-b5a3-f393-e0a9-e50e24dcca9e
V 16:31:29.708 Enabling notifications for 6e400005-b5a3-f393-e0a9-e50e24dcca9e
D 16:31:29.717 gatt.setCharacteristicNotification(6e400005-b5a3-f393-e0a9-e50e24dcca9e, true)
I 16:31:29.717 Notification received from 6e400002-b5a3-f393-e0a9-e50e24dcca9e, value: (0x) EC-20-29-D6-1C-48
D 16:31:29.718 "? )? H" received
I 16:31:29.718 Notification received from 6e400003-b5a3-f393-e0a9-e50e24dcca9e, value: (0x) FF-D6-FF-EA-FF-F1
D 16:31:29.718 "??????" received
D 16:31:29.718 gatt.writeDescriptor(00002902-0000-1000-8000-00805f9b34fb, value=0x0100)
I 16:31:29.874 Notification received from 6e400004-b5a3-f393-e0a9-e50e24dcca9e, value: (0x) 00-01-8A-F2
D 16:31:29.875 "(0x) 00-01-8A-F2" received
I 16:31:29.875 Notification received from 6e400002-b5a3-f393-e0a9-e50e24dcca9e, value: (0x) EC-20-2A-40-1C-5C
D 16:31:29.876 "? *@ \" received
I 16:31:29.876 Data written to descr. 00002902-0000-1000-8000-00805f9b34fb, value: (0x) 01-00
D 16:31:29.876 "Notifications enabled" sent
```

Figure 5.1: Splunk with Raw Data

5.3 Real-time Alerting

Another key feature available in the prototype developed is, its ability to push alerts and notifications from the gateway itself. Since the prototyped gateway is capable of filtering the data in itself, more value can be added by developing it to send alerts real-time.

5.3.1 Latency

Some current architectures attempt to route all commands, including rules processing, scenes and groups through the cloud, which can introduce unwanted latency. However, having an on-premises IOT gateway ensures response to sensor input is perceived as immediate which can be crucial in a critical IOT deployment.

The prototype built can alert and push notifications from itself using GCM technology. This largely helps to improve overall performance of the IOT deployment which will improve trust in the overall IOT system.

Further the ability to send Email alerts directly from the gateway has made it much faster to send information to remote clients. Further if triggers are automated to act based on the email content, the value added is much higher.

5.3.2 Uninterrupted, Internet-Independent Operation

While cloud-based control and monitoring of on-premises IoT devices will continue to be essential, attempting to move all intelligence and routing responsibilities into the cloud has so far proven to be ineffective [21]. This is due to the latency issues mentioned above, but a larger problem involves the need for a constant connection.

Although internet connectivity and up-time have made great advances, outages and other interruptions still occur frequently. It simply isn't feasible for users and managers to risk losing control of the *things.*

However, if the on-premise gateway is capable of alerting and pushing notifications by itself, then the damage can be controlled until the cloud connectivity is established.

This is where the SMS alerting feature largely adds value to the overall solution, as it is completely using a different connectivity technology to alert remote users, because sometimes app notification might not be sufficient based on the application. Further this helps to save costs on data and also opens an opportunity to leverage on optimized built-in apps.

Chapter 6

6 DISCUSSION

6.1 Introduction

This chapter will be focused on the outcome of the prototype developed against what's been identified as the research problem. Further it will discuss the new findings, challenges that were come across while developing the prototype. And the possible improvements are discussed afterwards.

6.2 Closing the Research Gap

In brief, the key challenges that we identified which we not present in traditional IOT gateways which we planned to be solved were,

- *Processing large data volumes in the Gateway*
- *Developing a Gateway with optimized power consumption*
- *Developing a low cost, smarter IOT gateway*

By using an Android mobile device as a gateway which are optimally designed to last for long hours with the option of continuous charging if closer to a power source, the problem of looking for a device with optimum power consumption was sorted. Similarly, a mobile device which is water proof and can withstand rugged conditions can be purchased at a cost much less than a traditional IOT gateway. For an example, a Libelium IOT gateway would cost more than $2000 where as a rugged mobile device with a high-end processor can be purchased easily for less than $1000.

As these mobile devices come with good quality processors along with well sufficient storage, they can easily absorb lots of data coming from many sensors simultaneously without hindering the performance. A prototype android app was built along with a Python application to convert and filter data coming from the Bluetooth sensors connected.

6.3 Improving the Prototype

Even though it is not being identified as a research gap initially, a new feature was added to the IOT Gateway prototype using GCM APIs where it can push alerts from the gateway itself in real-time without waiting for the IOT cloud to push alerts. However, few key improvements are identified which need to be done for the prototype to make it more productize-able.

- *Have one android application for everything.*

  Because in the current deployment, two different android applications have been used to collect data and to push real-time notifications. Further a python application has been developed to translate the raw data into meaningful information.

- *Develop an interface to configure the connected end devices*

  This is a standard feature in most of the traditional IOT gateways. The prototype has limited features in this regard. Having a proper interface will be crucial in a large scale IOT deployment because, commissioning a large network is one of the most time consuming and difficult operations in the network life cycle. Commissioning starts with making sure devices are part of the network. Further below features such as,

  - Being able to control devices, groups, or scenes on a schedule
  - Being able to take input from one device on the system and use that to control another device, group or scene.

  should also be available in the proposed interface.

- *Collect data from other connectivity technologies which comes out-of-the-box with mobile devices such as Wi-Fi, 3G and 4G*

  The prototype built can capture data only via BLE, whereas an IOT gateway could connect to *things (end nodes)* and collect data via multiple connectivity technologies. For an example, the Libelium gateway, “Meshlium” supports multiple connectivity options such as GPRS, 3G, 4G, Wi-Fi, Ethernet, Zigbee, SigFox and LoRaWAN etc. However, these options will be only limited to BLE, Wi-Fi, GPRS, 3G and 4G, because, a generic mobile device is being used in this development.

Chapter 7

7 CONCLUSION

7.1 Introduction

This chapter summarizes the overall research with concluding on how the prototype developed helped to solve the research problems. Further this chapter recommends possible features that can be added to the developed product in the future.

7.2 Conclusion

Even though there are various types of IOT gateways available in the market, most of them are having different feature limitations which has limited the potential of an end-to-end true IOT deployment. The relationship between the cost against the functionalities in always a negative relationship.

However, utilizing the mobile device as the gateway with more value additions is the objective of this research which was illustrated using a prototype built using Android based applications. And value additions such as data filtering and real-time alerting from the gateway itself were developed. Data filtering was executed via a python-based application which converted the HEXA-Decimal values pushed from nRF51 Sensor tag via blue-tooth to the mobile device. Real-time alerting was accomplished by leveraging on the GCM services.

After which the evaluation was done comparing the prototype against a traditional IOT gateway functionality and the possible cost savings achievable, because so far, the biggest bottle-neck for a large scale IOT deployment is the significant costs involved.

7.3 Recommendations

There are commonly used Wireless connectivity IOT technologies such as LoRaWAN, Zigbee, SigFox and NB-IOT etc which cannot be used directly with a mobile device as the hardware does not support such technologies. Therefore, it is recommended to develop hardware addons which enables such capabilities in a mobile device such that the use of Mobile devices can be expanded.

Further it is recommended to custom built an android based gateway removing the unnecessary cost components such as display, speakers and other hardware components and improve the features and functionalities needed for an effective gateway.

Finally, it is recommended to build an Android based OS tailor made for IOT deployments by removing unnecessary OS level features. Even now Microsoft has built OS versions such as Microsoft IOT Core and Microsoft Embedded Enterprise which can run in IOT devices. And there are Android OS versions custom built to run in Digital signage devices. Therefore, this is a possibility which will be addressed in coming years.

REFERENCES

[1] Gartner, "Internet of Things." [Online]. Available: https://www.gartner.com/it-glossary/internet-of-things/ .[Accessed: 20-Dec-2017].

[2] Prism Tech, "Messaging Technologies for the Industrial Internet and the Internet of Things Whitepaper." [Online].Available: http://www.prismtech.com/sites/default/files/documents/Messaging-Whitepaper-051217.pdf. [Accessed: 22-Dec-2017].

[3] S. Bandyopadhyay, M. Sengupta, S. Maiti and S. Dutta, "Role of middleware for Internet of Things : a Study," International *Journal of Computer Science & Engineering Survey (IJCSES)*, Vol.2, No.3, 2011

[4] S. Zarghami, "Middleware for Internet of Things." Master Thesis, University of Twente, Netherlands, 2013.

[5] M. Gregório, R. Santos, C. Barros and G. Silva, "Problems in Adopting Middleware for IoT : A Survey," *The Tenth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies,* 2016.

[6] Microsoft Azure, "What is cloud computing?" [Online]. Available: https://azure.microsoft.com/en-au/overview/what-is-cloud-computing/ . [Accessed: 25-Dec-2017].

[7] J. Soldatos, "Cloud Computing Basics." [Online]. Available: https://www.kdnuggets.com/2017/05/internet-of-things-iot-cloud.html . [Accessed: 30-Dec-2017].

[8] M. Rouse, "SmartPhone Sensor." [Online]. Available: http://whatis.techtarget.com/definition/smartphone-sensor. [Accessed: 30-Dec-2017].

[9] M. Rouse, "Sensor data." [Online]. Available: http://internetofthingsagenda.techtarget.com/definition/sensor-data. [Accessed: 30-Dec-2017].

[10] M. Hausenblas, "Smart phones and the Internet of Things." [Online]. Available: https://mapr.com/blog/smart-phones-and-internet-things/ . [Accessed: 30-Dec-2017].

[11] J. Walker, "Smartphones with IoT Apps are the Best Gateways for Remote Monitoring & Data collection." [Online]. Available:http://www.vensi.com/smartphones-with-iot-apps-are-the-best-gateways-for-remote-monitoring-data-collection/ . [Accessed: 5-Jan-2018].

[12] M. Elkhodr, S. Shahrestani, H. Cheung, “A Middleware for the Internet of Things,” *IEEE International Conference on Data Science and Data Intensive Systems*, 2015

[13] Finoit Technologies, “Top 15 Sensor Types Being Used in IoT.” [Online]. Available: https://www.finoit.com/blog/top-15-sensor-types-used-iot/ . [Accessed: 5-Jan-2018]

[14] B. Kang, H. Choo, “An experimental study of a reliable IoT gateway,” *The Korean Institute of Communications Information Sciences*, 2017

[15] M. Mohan “How middleware plays a crucial role in IoT.” [Online]. Available: http://www.computerworld.in/feature/how-middleware-plays-crucial-role-iot . [Accessed: 7-Jan-2018]

[16] D. Blouin, “Mobile Devices are the Gateway to the Internet of Things.” [Online]. Available: https://www.cmswire.com/cms/internet-of-things/mobile-devices-are-the-gateway-to-the-internet-of-things-026531.php. [Accessed: 10-Jan-2018]

[17] Android Developers, “Bluetooth low energy overview.” [Online]. Available: https://developer.android.com/guide/topics/connectivity/bluetooth-le.html . [Accessed: 10-Jan-2018]

[18] K. Townsend, “Introduction to Bluetooth Low Energy.” [Online]. Available: https://cdn-learn.adafruit.com/downloads/pdf/introduction-to-bluetooth-low-energy.pdf . [Accessed: 12-Jan-2018]

[19] Nordic Semiconductor, “Product Specification.” [Online]. Available: https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF51822. [Accessed: 13-Jan-2018]

[20] Google Cloud Messaging, “Google Cloud Messaging: Overview.” [Online]. Available: https://developers.google.com/cloud-messaging/gcm. [Accessed: 05-Feb-2018]

[21] M. Smith, “The Four Benefits of Using a Gateway in Your IoT Design.” [Online]. Available: https://www.ecnmag.com/blog/2017/03/four-benefits-using-gateway-your-iot-design. [Accessed: 07-Feb-2018]

[22] BLE Stack User’s Guide, “Generic Attribute Profile (GATT).” [Online]. Available: http://dev.ti.com/tirex/content/simplelink_cc2640r2_sdk_1_50_00_58/docs/blestack/ble_user_guide/html/ble-stack-3.x/gatt.html . [Accessed: 09-Feb-2018]

[23] Nordic Semiconductor, “nRF Connect for Mobile.” [Online]. Available: https://www.nordicsemi.com/eng/Products/Nordic-mobile-Apps/nRF-Connect-for-Mobile [Accessed: 15-Feb-2018]

[24] GitHub, Inc., “NordicSemiconductor/nRF-Logger-API.” [Online]. Available: https://github.com/NordicSemiconductor/nRF-Logger-API [Accessed: 15-Feb-2018]

[25] Libelium, “Waspmote Plug & Sense” [Online]. Available: http://www.libelium.com/products/plug-sense/ [Accessed: 20-Feb-2018]

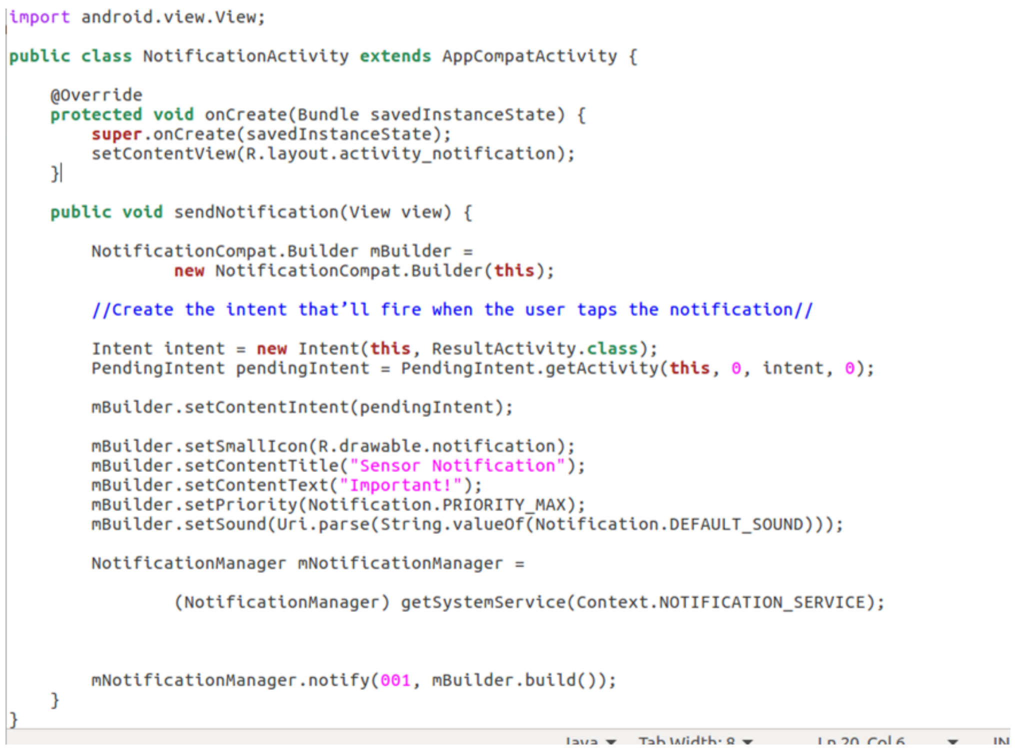
[26] Advantech, “IoT Integration Gateway” [Online]. Available: http://advantech-bb.com/product-technology/iot-and-network-edge-platforms/smartswarm-341/ [Accessed: 20-Feb-2018]

[27] Khan M.H., Shah M.A., *“*Survey on security threats of smartphones in Internet of Things,*” International Conference on Automation and Computing (ICAC); 7-8 September; University of Essex Wivenhoe Park Colchester, UK.* 2016

[28] Furnell, S., “Mobile Security: A pocket guide”[Online]. Available : http://www.books24x7.com.libproxy.library.wmich.edu/marc.asp?bookid=34445 [Accessed: 28-April-2019]

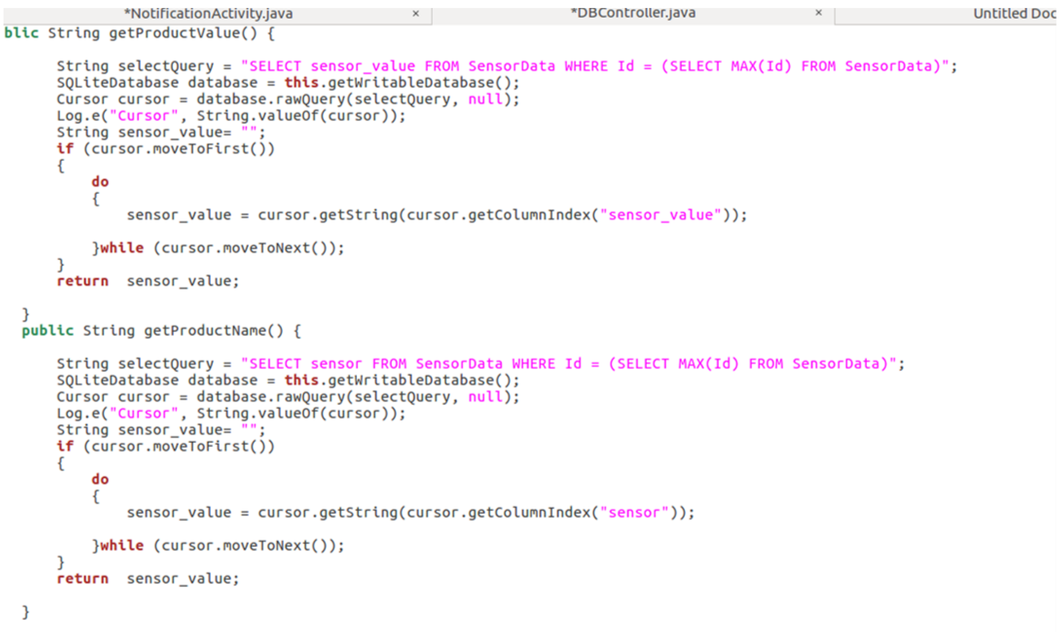
APPENDIX A: Real-time Application Source Code

A.1 Notification.java

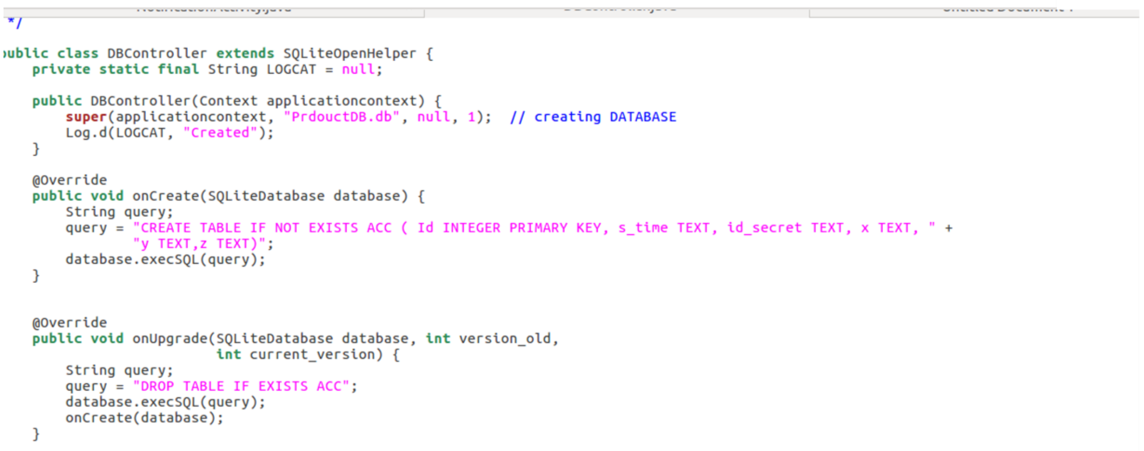


```java
import android.view.View;

public class NotificationActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_notification);
    }

    public void sendNotification(View view) {

        NotificationCompat.Builder mBuilder =
                new NotificationCompat.Builder(this);

        //Create the intent that’ll fire when the user taps the notification//

        Intent intent = new Intent(this, ResultActivity.class);
        PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, intent, 0);

        mBuilder.setContentIntent(pendingIntent);

        mBuilder.setSmallIcon(R.drawable.notification);
        mBuilder.setContentTitle("Sensor Notification");
        mBuilder.setContentText("Important!");
        mBuilder.setPriority(Notification.PRIORITY_MAX);
        mBuilder.setSound(Uri.parse(String.valueOf(Notification.DEFAULT_SOUND)));

        NotificationManager mNotificationManager =

                (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);


        mNotificationManager.notify(001, mBuilder.build());
    }
}
```

A.2 DB_Controller.java



```
blic String getProductValue() {

      String selectQuery = "SELECT sensor_value FROM SensorData WHERE Id = (SELECT MAX(Id) FROM SensorData)";
      SQLiteDatabase database = this.getWritableDatabase();
      Cursor cursor = database.rawQuery(selectQuery, null);
      Log.e("Cursor", String.valueOf(cursor));
      String sensor_value= "";
      if (cursor.moveToFirst())
      {
          do
          {
              sensor_value = cursor.getString(cursor.getColumnIndex("sensor_value"));

          }while (cursor.moveToNext());
      }
      return  sensor_value;

  }
  public String getProductName() {

      String selectQuery = "SELECT sensor FROM SensorData WHERE Id = (SELECT MAX(Id) FROM SensorData)";
      SQLiteDatabase database = this.getWritableDatabase();
      Cursor cursor = database.rawQuery(selectQuery, null);
      Log.e("Cursor", String.valueOf(cursor));
      String sensor_value= "";
      if (cursor.moveToFirst())
      {
          do
          {
              sensor_value = cursor.getString(cursor.getColumnIndex("sensor"));

          }while (cursor.moveToNext());
      }
      return  sensor_value;

  }
```



```
*/

ublic class DBController extends SQLiteOpenHelper {
   private static final String LOGCAT = null;

   public DBController(Context applicationcontext) {
       super(applicationcontext, "PrdouctDB.db", null, 1);  // creating DATABASE
       Log.d(LOGCAT, "Created");
   }

   @Override
   public void onCreate(SQLiteDatabase database) {
       String query;
       query = "CREATE TABLE IF NOT EXISTS ACC ( Id INTEGER PRIMARY KEY, s_time TEXT, id_secret TEXT, x TEXT, " +
               "y TEXT,z TEXT)";
       database.execSQL(query);
   }


   @Override
   public void onUpgrade(SQLiteDatabase database, int version_old,
                         int current_version) {
       String query;
       query = "DROP TABLE IF EXISTS ACC";
       database.execSQL(query);
       onCreate(database);
   }
```