VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Jere Lyytinen

An Android Application for Communicating with 'Internet of Things' Devices

Tekniikka
2017

VAASAN AMMATTIKORKEAKOULU
Tietotekniikka

TIIVISTELMÄ

Tekijä	Jere Lyytinen
Opinnäytetyön nimi	An Android Application for Communicating with 'Internet of Things' Devices
Vuosi	2017
Kieli	englanti
Sivumäärä	44
Ohjaaja	Timo Kankaanpää

Opinnäytetyö suoritettiin Comsel System Oy:lle. Työssä toteutettiin sovellus, joka listasi ympärillä olevat Bluetooth Low Energy -laitteet, haki laitteen kuormasta Eddystone-viestin, joka sisälsi Comsel System Oy:n nimiavaruuden sekä instanssin. Ohjelma suodatti pois muut kuin Comselin laitteet Eddystone-viestin sisältämän nimiavaruuden perusteella, ja käytti Comsel-laitteen instanssissa olevaa UUID-tunnusta, jotta Comselin Corona-pilvipalvelusta saatiin haettua Comselin laitetta tukeva IP-osoite. Tällä IP-osoitteella vaihdettiin fyysinen linkki verkkorajapintaan, jolla suoritettiin CoAP-kutsut, joiden perusteella saatiin selville laitteen tyyppi sekä laitteesta haluttavat data-arvot. Lopuksi nämä listattiin Androidin näkymään käyttäjälle luettavaksi. Sovelluksen avulla yritys tai asiakas voi reaaliajassa seurata laitteiden keräämää tietoa ja huomata, jos jokin laite on esimerkiksi alkanut toimia virheellisesti.

Opinnäytetyössä käytettiin Comselin tarjoamaa Texas Instruments CC2650 sensor tagia sekä Android-puhelinta. Sovellus kehitettiin Java-ohjelmointikielellä Android Studiolla.

Lopputuloksena oli sovellus, joka toi näytölle listan löydetyistä Comselin Bluetooth Low Energy-beaconeista, sekä niiden reaaliajassa päivittyvistä arvoista. Sovelluksen esiin listaamia arvoja on tarvittaessa helppo muokata ja sovellus on tehokas ja hyödyllinen väline niin asiakkaalle kuin Comselille, ja sovellukseen on jo suunnitteilla jatkokehitystä.

Avainsanat Bluetooth Low Energy, Android, CoAP, Eddystone

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietotekniikka

ABSTRACT

Author	Jere Lyytinen
Title	An Android Application for Communicating with 'Internet of Things' Devices
Year	2017
Language	English
Pages	44
Name of Supervisor	Timo Kankaanpää

The thesis work was made for Comsel System Oy. The goal was to create an application that lists the nearby Bluetooth Low Energy -devices and their payloads that would contain the instance and the payload of the device. The application used the namespace to filter off the non-Comsel devices and then used the UUID-information obtained from the instance to connect to the Comsel's Corona cloud-service to obtain the IP address supporting the Comsel's device. Using the obtained IP, the application switched from Bluetooth to a network interface and performed CoAP-requests to find the type of the beacon and its measurement values and displayed them in the user interface to the user.

The beacon used in the thesis work was a Texas Instrument CC2650 sensor tag along with an Android phone offered by Comsel System Oy. The software was created using Java programming language in Android Studio environment.

The result was a software that displayed the list of found Comsel Bluetooth Low Energy -devices and their real-time values. Now it is easy to configure what data to display from the found devices and the finished software is an effective and handy tool for both the customer and Comsel System Oy. Further development plans for the software are already being designed.

Keywords Bluetooth Low Energy, Android, CoAP, Eddystone

SISÄLLYS

TIIVISTELMÄ
ABSTRACT

1 SUMMARY 8
1.1 ”Internet of Things” as a term 8
1.2 Thesis work and its stages 10
1.3 Comsel System Oy 10
1.4 Texas Instruments CC2650 Sensor Tag 11
1.5 Huawei Y511 CUN-L21 12
1.6 Android 13
1.7 Android Versions 14
1.8 Android’s Platform Architecture 15
2 PROGRAMMING ENVIROMENT AND LANGUAGE 17
2.1 Android Studio Structure 17
2.2 Android Studio Completion 18
2.3 Gradle Build System 18
2.4 Java programming language 18
3 SOFTWARE ANALYSIS 20
3.1 UML Use Case Diagram 20
3.2 Sequence diagram 21
3.3 General View 23
3.4 Functions 23
3.5 External Interface 24
3.6 Test Plan 24
4 THE IMPLEMENTATION 29
4.1 Android Manifest 29
4.2 XML user interface 29
4.3 Main Activity class 30
4.4 Eddystone 31
4.5 Eddystone-UID 31

4.6 HTTP REST API 32
4.7 Constrained Application Protocol 33
4.8 CoAP in the thesis work 35
4.9 Test Report 37
5 CONCLUSION 41
6 SOURCES 42

KUVIO- JA TAULUKKOLUETTELO

Figure 1. A demonstration of a city and its districts that could be connected to Iinternet of Things -network. /2/ 9
Figure 2. Texas Instruments CC2650 Sensor Tag. 12
Figure 3. Huawei Y5II CUN-L21. /5/ 13
Figure 4. The Android version numbers and their codenames. /7/ 14
Figure 5. Android Architecture. /8/ 16
Figure 6. Android project structure. 17
Figure 7. Use Case Diagram. 21
Figure 8. Sequence Diagram. 22
Figure 9. Android Manifest. 29
Figure 10. XML user interface. 30
Figure 11. Main Activity class. 31
Figure 12. Eddystone beacon and Eddystone-UID. /13/ 32
Figure 13. CoAP layer structure. 34
Figure 14. CoAP and HTTP co-operation. 34
Figure 15. A final view where the found device(s) and their values are being listed in the user interface. 36

Table 1. Functions 23
Table 2. External Interface 24
Table 3. Test Plan 25
Table 4. Test Report 37

LYHENTEET JA TERMIT

BLE	Bluetooth Low Energy. A wireless personal area network technology.
EDDYSTONE	Google's open Bluetooth Low Energy beacon profile.
RSSI	Received Signal Strength Indication. A measurement of the power present in a received signal
MAC	Media Access Control address in the unique identifier for network interfaces for communications at the data link layer.
UUID	Universally Unique Identifier. 128-bit number used in computer systems to identify information
JSON	JavaScript Object Notation is an open-standard file format that transmits data objects by using human-readable text
API	Application Programming Interface is a set of methods that communicate between software components.
COAP	Constrained Application Protocol is a protocol for constrained devices.
SSL	Secure Shell is a protocol for secured data transmitting. Most commonly used for a secured remote connection to another computer via a character based console.
HTTP	Hypertext Transfer Protocol is a protocol used by web browsers and WWW-services for transmitting data

1 SUMMARY

Purchasing and installing an Internet of Things (IoT) networked embedded system as an end consumer today can be prohibitively difficult because of the plethora of technologies involved and the often-convoluted setup processes. To ease the setup process, the tools need to be created and these tools need to be available to the end consumer, preferably by using the devices they already own, for example, smartphones.

This thesis work examines building an Android application to handle communication with IoT devices.

1.1 ”Internet of Things” as a term

Internet of Things is a network of objects that are connected to the internet and can collect and exchange data with embedded sensors and software. The internet connected, “Internet of Things” devices can be monitored and/or controlled remotely and are widely used today in companies and in the daily life. By estimate, over 34 billion IoT devices will be connected to the internet by 2020, of which 24 billion are IoT devices and 10 billion traditional computing devices such as smartphones and tablets. The phrase for the future is “Anything that can be connected, will be connected” and already today Internet of Things devices can be anything from cellphones to tablets, lamps, headphones or toasters and the list just keeps increasing. /1/

Internet of Things is the top adopt for businesses and companies due to its decreasing costs and increased productivity. In addition, Internet of Things keeps expanding to the new markets and developing new products. Furthermore, Internet of Things offers new chances for the government, public services and cities to build “smart cities” that will help us to reduce waste and improve efficiently and the way we work and live. (See Figure 1.)

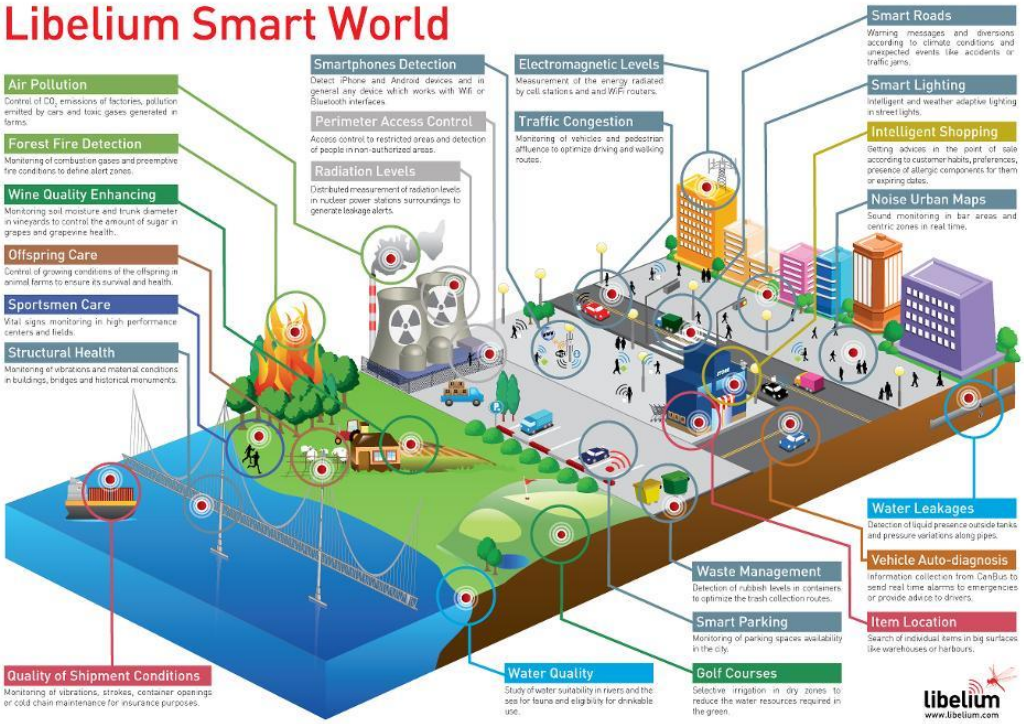

Figure 1. A demonstration of a city and its districts that could be connected to a Internet of Things -network. /2/

IoT is a door opener for endless opportunities and connections, and most of its uses are yet to be discovered. The Internet of Things will affect people in every day's life and be visible everywhere around us both indoors and outdoors. In a nutshell Internet of Things is our technological future. /2/

1.2 Thesis work and its stages

The thesis project has three distinct stages: locating and identifying the device in proximity of the user, obtaining the IP address from Comsel System Oy's Corona cloud service that supports the found Comsel device and, finally, switching into a network interface by using the obtained IP address and performing a CoAP request to identify the device and its values (such as the temperature from a heat beacon). These three steps are completed fully within the project.

The first step is achieved using the Bluetooth Low Energy beacon (BLE beacon) protocol. The BLE beacon protocol is a simple, non-IP based, one way communication protocol used to exchange a Universally Unique Identifier (UUID) from the device to the client. The second step is to involve communicating with a control system to the Comsel's Corona cloud service in order to translate the UUID into a network address (IP) to facilitate direct network communication between the Android application and the device. Finally, the third step is to use the IP-address to swap into a web interface to perform a CoAP request to obtain the device type and its values that would be useful to the company and the customer. Due to the timeframe the nice-to-have feature for direct communicating back and forth between the IoT devices and the Android Application is left out. However, this step is completed with the one way communication from the beacon to the application.

1.3 Comsel System Oy

Comsel System is a product design and development company specialized in developing and industrializing engineering for the energy factor. /3/ Comsel System Oy was founded in 1989 and today its headquarters are in Yliopistonranta, Vaasa. Initially the company was focused on planning and implementing electricity installations, data network services and automatic equipment, also known as embedded systems. In 1995 Comsel System Oy started

developing TCP/IP based remote acquisition and Internet of Things CI/OS products that consist both hardware and software. The main use of the CI/OS-products was to remotely read electricity-, district heating-, gas and water consumption data, known as communication products for automatic meter reading.

In 2000, Comsel System Oy delivered GPRS TCP/IP-based metering points to different projects in the Nordic countries, for example to two of the largest AMR-projects in Sweden (Fortum Sweden and E.On Sweden).

Year 2012 was a milestone year when Comsel System Oy launched three smart metering research & development projects, namely; Comsel Corona Service Hub, an Information Technology-infrastructure system for managing the equipment involved in smart metering and the measurement data, Comsel Zodiac Smart Metering Module, a communication module for stand-alone operations as well as integration into various meters, and Neuron, a sensor system that enables communication and exchanging useful information between devices and interact to form an useful application.

By 2016 Comsel Corona Service Hub and Comsel Zodiac Smart Metering Modules have delivered over 65 million electricity and district heating measurements from installations in the Nordic countries.

1.4 Texas Instruments CC2650 Sensor Tag

Texas Instruments is a technology company that designs and manufactures semiconductors, which are sold globally to manufacturers and electronics designers.

The sensor tag used in this thesis work is a Texas Instruments CC2650 Sensor Tag, which supports the wireless MCU that targets the Bluetooth remote control applications. It is mainly used due to its cost efficiency and ultralow power. It is a multi-standard device that supports wireless technology. In the thesis

work the application is built to discover this specific sensor tag, which contains the payload that is used to fetch the IP address of the beacon from Comsel System Oy's Corona cloud service using the UUID-information. /4/

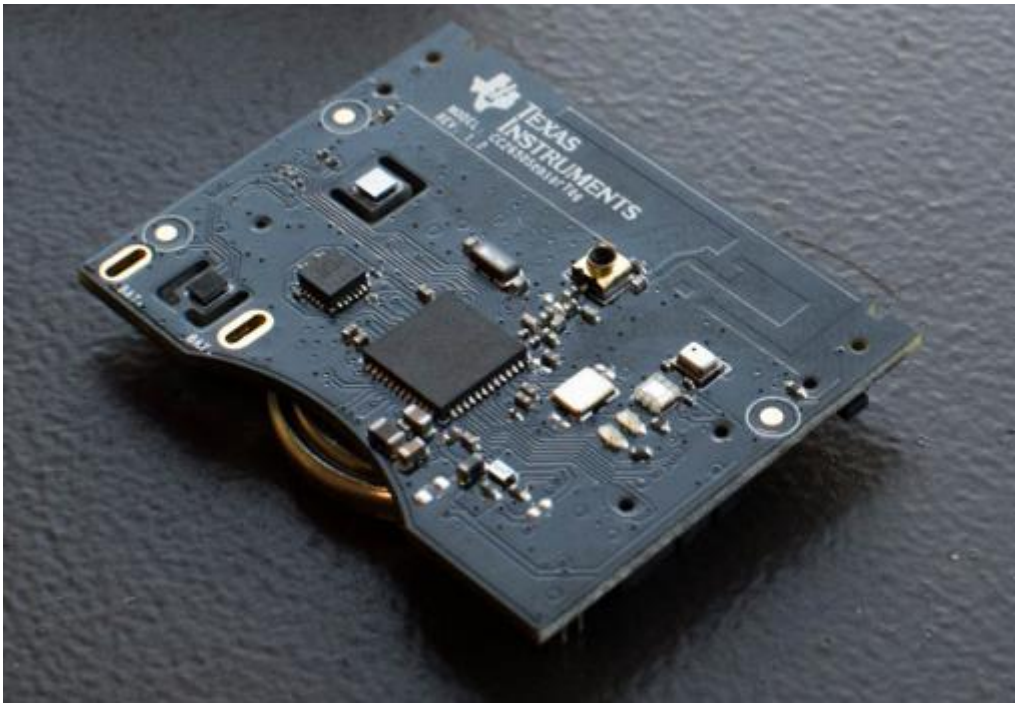

Figure 2. Texas Instruments CC2650 Sensor Tag.

1.5 Huawei Y511 CUN-L21

Due to the required Bluetooth connection, a physical phone is needed and the virtual phone environment cannot be used for testing as usually when developing an application. The phone used for building this application is a Huawei Y5II CUN-L21 smartphone with an Android version 5.1 (Lollipop). The phone model was launched in June 2016 and it offers a new platform (Lollipop) which supports the Eddystone Bluetooth Beacon -technology that is essential for this thesis work. The phone is provided by Comsel System Oy. To use the phone in development purposes, development settings must be turned on and the Bluetooth tracking and connecting have to be enabled. In addition,

the phone requires an address pair connection with the computer that is pushing the application into the device.

Figure 3. Huawei Y5II CUN-L21. /5/

1.6 Android

Android is Google's mobile operating system released in 2008. It is found on several devices, for example on TVs and tablets but is mainly known for being the most popular operating system for smartphones. Android uses a Linux kernel operating system and has a source code under Google's open source license. Most Android devices have a combination of both open and non-free software. In addition, Android applications, known as apps, can be downloaded from Google Store. By February 2017, over 2.7 million apps can be found Google Play. /6/

The responsible branch for founding and developing Android operating system is an Open Handset Alliance -consortium that includes software, hardware and telecommunicating companies that are aiming to evolve and increase the open standards for mobile devices.

In April 2017, after the breakthrough growth of smartphones in Asian countries, Android overtook Microsoft Windows as the most popular operating system for total internet usage across desktop, laptops, tablets and mobile combined. /6/

1.7 Android Versions

Android Mobile OS began their version history with the release of their alpha version (Android alpha) in 2007. The first commercial version, Android 1.0, released a year later. The versions 1.1 and 1.2 had no code names in them but since then the code names have been onfectionery-themed in alphabetical order, starting with the Android 1.5 version called ”Cupcake”. The latest major version released in August 2016 was the Android 7.0 version called ”Nougat”. (See Figure 4 below.)

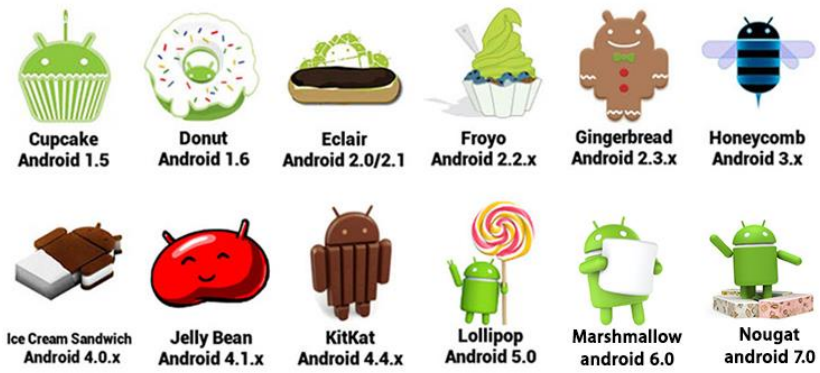

Figure 4. The Android version numbers and their codenames. /7/

1.8 Android's Platform Architecture

The Linux Kernel is the lowest level in the Android's platform architecture. It allows Android to take advantage of key security features and enables the manufacturers to develop hardware drivers for it by controlling the resource management. /8/

Together with the upper level called Hardware Abstraction Layer (HAL), Linux Kernel and HAL are also responsible for the communication with the physical devices. HAL consists of multiple library modules, which implements the interface for the specific types of hardware components.

Android Runtime (ART) is the third level in the architecture that consists the Java libraries. The devices which run with Android version 5.0 or higher have their apps running their own process with their own instances of the ART. The key features of this level are the Ahead-of-time and the Just-in-time compilations, the optimized garbage collection and the better debugging support that includes a sampling profiler and a better dialogic exception and a report system for crashes.

In addition, the third level contains the native C and C++ libraries that are required for the Android system's core components and services like ART and HAL.

The fourth level contains the Java API framework that the entire Android OS feature-set uses. These APIs form the building blocks that is required to create Android apps.

The fifth and highest level is the System Application level that contains the applications in Android, both core applications installed by the system such as the clock, calendar and email and the applications installed by the user.

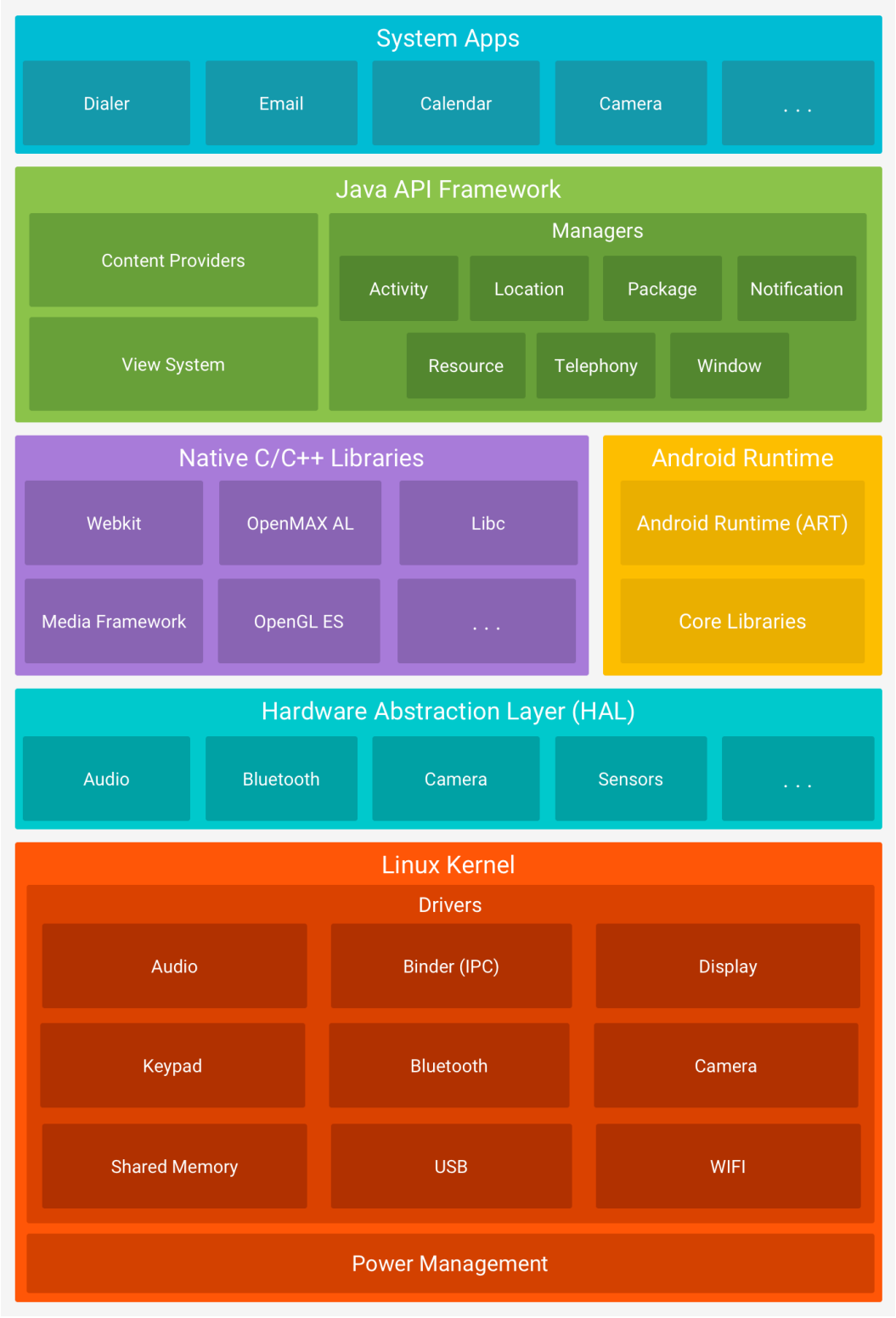

Figure 5. Android Architecture. /8/

2 PROGRAMMING ENVIROMENT AND LANGUAGE

2.1 Android Studio Structure

The thesis work is made on Android Studio. It is the Intellij IDEA based Integrated Development Environment for Android app development. /9/ It is structured to contain the Android app, the library and the Google App Engine modules. By default, it uses an Android project view as shown below. The build files are displayed under the Gradle Scripts and every app module contains the manifest, the Java and the res folders. The Manifests contains the AndroidManifest.xml -file which provides the information about the app to the Android System and it is required before the app's code can be run. The Java folder contains the Java code source files, including the Junit test files and codes. Lastly, the res-folder contains the UI-elements such as the XML-layout, UI-strings, bitmap images and stylesheets.

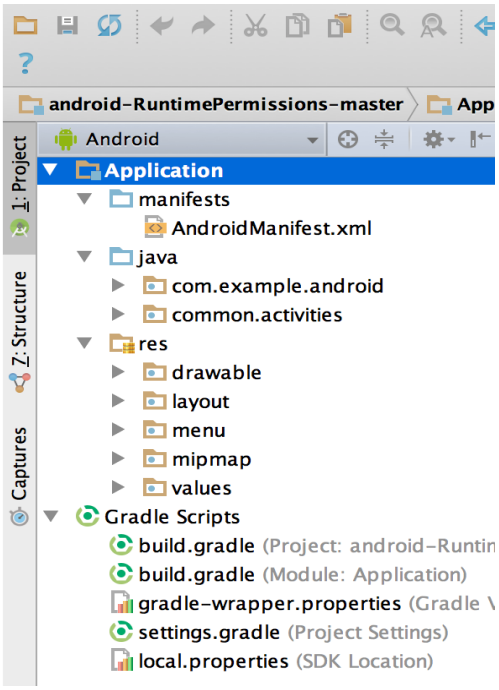

Figure 6. Android project structure.

2.2 Android Studio Completion

Android Studio uses three types of a code completion. The first one is the basic completion that displays the basic suggestions for the variables, the types, the methods and the expressions. The second one is a smart completion that displays the relevant options on the context, excluding the unnecessary parts. The last completion is the statement completion which completes the statement, adding the missing parts for the completion such as the parentheses, the formats and the braces.

2.3 Gradle Build System

Android Studio uses Gradle as the foundation of the build system. The build system is used for customizing, configuring and extending the build, to create multiple APKs with different features and to reuse the resources and the codes across the source. /10/ The build file is named in the project as build.gradle and it is the text file to configure the build with elements provided by the Android plugin. The Android automatically generates both the top-level build file and the module-level files for each file when importing an existing project into the program.

2.4 Java programming language

The thesis work is made with Java programming language. Java is a programming language and a computing platform first released by Sun Microsystems in 1995. /11/ The Java language is a C-language derivative, so the syntax rules look much like in C-programming. The code blocks are modularized into methods and delimited by braces, and the variables are declared before they are used.

The Java language uses packages which are Java's namespace mechanic. Inside the packages are the classes and inside them the methods, the variables, the constants and the other programming entities. In Java programming, the source code is

written in the .java files and then compiled. The compiler checks the code and then writes the bytecode into the .class files. The bytecode is a set of instructions targeted to run on a Java virtual machine (JVM). The JVM reads and interprents the .class files and executes the program's instructions on the native hardware platform for which the JVM was written to.

3 SOFTWARE ANALYSIS

This is a thesis work for Comsel System Oy. The goal is to create an Android Application for communicating with 'Internet of Things' devices by using a Bluetooth low energy beacon (BLE beacon).

3.1 UML Use Case Diagram

The software requires minimal user actions. Once the application is run, it will scan all the nearby BLE-devices, parse the Eddystone message from the payload, filter out the non-Comsel devices, obtain the device's IP address from Comsel's Corona cloud service by using the discovered UUID and perform the CoAP callbacks to obtain the device type and its measurement data using the obtained IP-address. In the end the application will list the found devices and their values in the user interface for the user. (See Figure 7 below.)

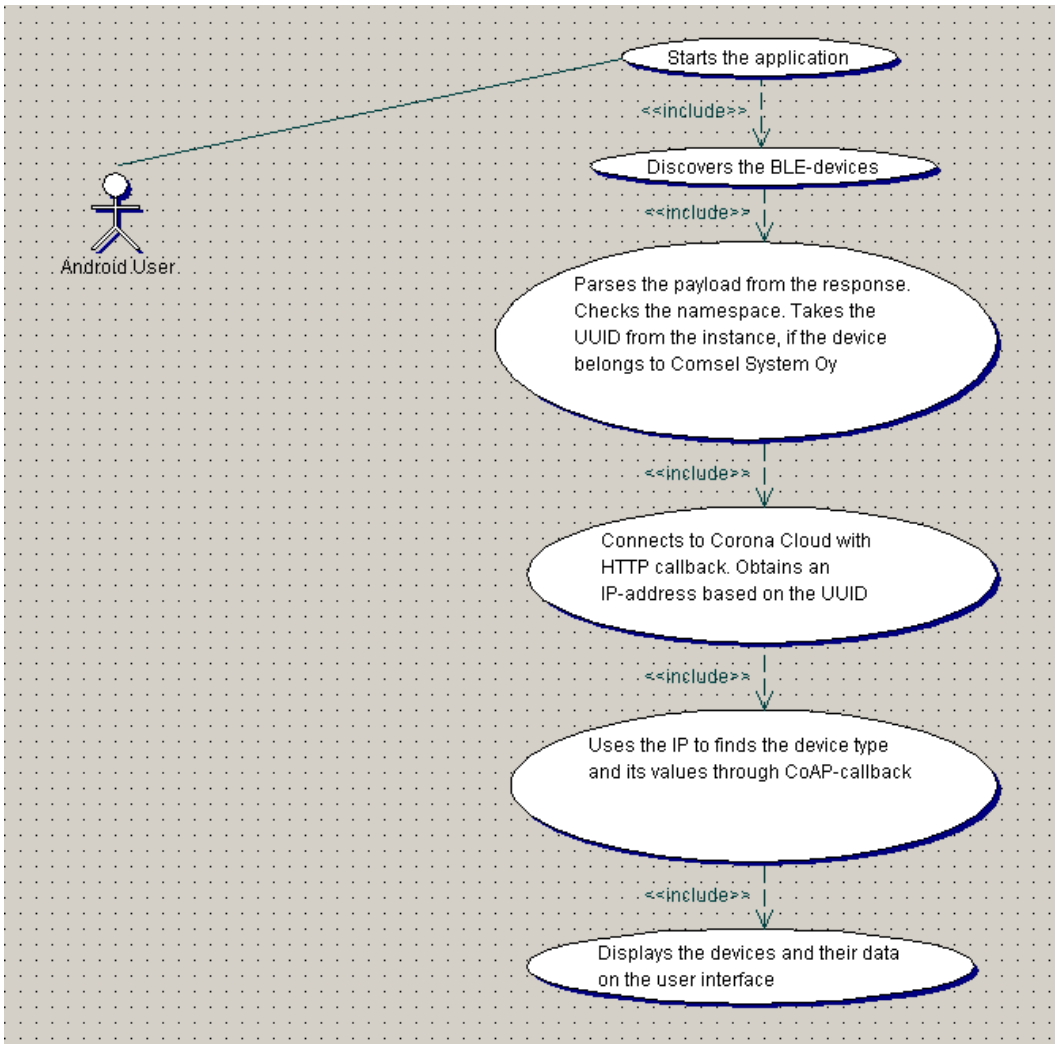

Figure 7. Use Case Diagram.

3.2 Sequence diagram

The sequence diagram shows the sequence of actions done by the application. (See Figure 8 below.) After the user runs the application, the app will create an empty list view into the user interface. After that the MainActivity method in the DeviceScanActivity class will start the BluetoothScanner class to scan the Bluetooth Low Energy -beacons around. When a device is found, the class will return it into the DeviceScanActivity main class which checks if it belongs into the Comsel's namespace. If it does, it will parse its instance to obtain the UUID.

The UUID is sent as a HTTP-callback into the Comsel's corona cloud service where the best quality IP-address is parsed from the JSON and returned into the MainClass activity. The class then performs a CoAP-request over IPv6 network to obtain the device's type from the CoAP. Based on the returned device type, the app will perform a call to obtain the values wanted for the specific type of a beacon. The type of the beacon and the measurement data are listed into the listview of the user interface and the user can find the found Comsel System Oy beacons, their types and their values from it.

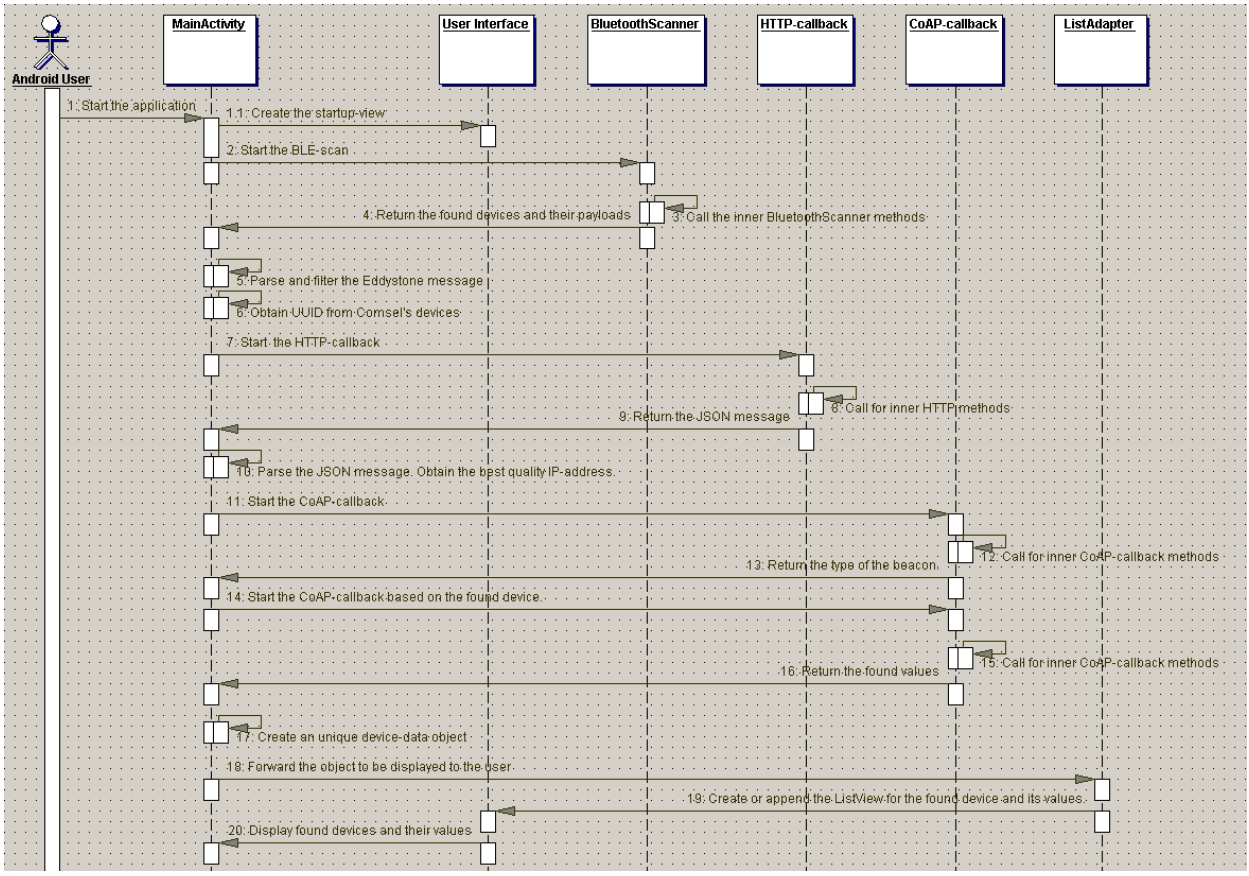

Figure 8. Sequence Diagram.

3.3 General View

The project is divided into three phases; 1) locating and identifying the device in proximity of the user by using Bluetooth Low Energy beacon and fetching its universally unique identifier (UUID) from its payload, 2) establishing an IP network link by communicating with Domain Name System to translate the UUID into an IP address and 3) fetching the device's type and its important data with the CoAP requests and printing them to the user. If time had allowed, the application could have performed direct application layer communication between the application and the device but this step is left out.

3.4 Functions

The functions and the functionality of the application were created at the start of the thesis work. Number 1 in the Priorize tab means a number 1 priority function and it is a must have function in order to get the final thesis work result accepted by Comsel System Oy. (See Table 1 below.)

Table 1. Functions

Reference	**Description**	**Priorize**
F1	Setting up an Android application	1
F2	Picking up a Bluetooth Low Energy protocol	1
F3	Extract the payload's "Universally unique identifier"	1
F4	Translate the Universally unique identifier into a network address	1
F5	Use the network address to perform a CoAP request	1
F6	Filter out the unnecessary data and create a request to the key values	2
F7	Display the fetched device type and the data to the user	1
F8	Perform direct application layer communication between Android ap-	4

	plication and the device	

3.5 External Interface

The external interface includes the external parts in the application that were not coded in the application but came from the outside of the application. For example the Comsel System Oy's Corona cloud service was the target for the HTTP-call instead of being created in the code itself. (See Table 2 below.)

Table 2. External Interface

Reference	**Description**
I1	Bluetooth low energy beacon (BLE beacon) using Android API
I2	Universally unique identifier (UUID)
I3	Comsel Cloud Service
I4	CoAP

3.6 Test Plan

The test plan was created to guarantee that each part in the application performs as intended and throws a defined error or a permission response to the user if some part needed adjusting or an input from the user. (See Table 3 below.)

Table 3. Test Plan

ID	Goal	Params/class	Test Procedure
A1	DeviceScanActivity's onCreate method creates the Listview for the devices	DeviceScanActivity, onCreate	Test that the DeviceScanActivity's onCreate method creates the Listview for the devices
A2	Application checks if the Bluetooth is enabled and alerts if not	DeviceScanActivity, onCreate	Test that the application checks if the Bluetooth is enabled. If not, it throws an alert.
B1	Blescanner starts	DeviceScanActivity, onCreate	Test that the scanner is starting to scan the devices
B2	Bluetooth asks for location access if not already enabled	DeviceScanActivity, requestCoarseLocationPermission, onRequestPermissionResult	Test that if the location access is not given, the phone will perform a request for it.
B3	BleScanner adds found devices to its list, updating the list periodically and adding new devices into it too	BleScanner	Test that the BleScanner adds a found device to the list and keeps adding a new found devices into it aswell by using a random generated number ending for both the found device and its value.

C1	DeviceScanActivity parses the Eddystone message into a namespace and an instance	DeviceScanActivity, onNewScanResultList	Test that the application parses the payload properly into a namespace and an instance
C2	DeviceScanActivity checks if the namespace matches with Comsel's namespace	DeviceScanActivity, onNewScanResultList	Test that non-Comsel devices will not be parsed forward. The Namespace must match to that of Comsel's
C3	Comsel device's message is parsed to obtain the UUID	DeviceScanActivity, onNewScanResultList	Test that the devices matching to Comsel's namespace have their instance parsed to obtain the UUID
D1	DeviceScanActivity calls for HTTP class to perform a HTTP callback using UUID as its parameter	DeviceScanActivity, onNewScanResultList	Test that the HTTP-class starts properly and the call is working
D2	HTTP-class checks for connection and alerts if connection fails	HttpHandler	Test that the connection is enabled and an alert is displayed if not
D3	DeviceScanActivity gives Comsel Corona address for the HTTP-callback	DeviceScanActivity, onNewScanResultList	Test that the address is correct and the callback functions
D4	HTTP-callback uses IP,	DeviceResponse, HttpHandler,	Check that the JSON-

	type, quality and error as its parameters and parses the JSON from Comsel Corona address	CallbackFunction, IP, type, quality, (error)	message in Corona cloud has the right parameters
D5	HTTP-class returns the best quality IP-address to DeviceScanActivityclass	DeviceResponse, HttpHandler, CallbackFunction	Test that the IP returned comes from the JSON property that has the highest ”Quality” value
E1	DeviceScanActivity uses the IP-address to create a CoAP call to check the device type	DeviceScanActivity, onNewScanResultList	Test that the IP-address is used when initializing the CoAP-request
E2	CoAP class checks that IPv6 is enabled	AsyncBleCoapClient	Test that the IPv6 is enabled and alert if not
E3	CoAP callback performs a CoAP request and returns the device type	AsyncBleCoapClient, CoapCallback	Test that the correct type of a device is returned
E4	DeviceScanActivity selects the correct address to obtain the data for the specific type of a device found	DeviceScanActivity, GetDeviceData	Test that the path for the device’s data request is correct and is based on the device type returned previously
E5	CoAP callback performs a CoAP request and returns the device data	AsyncBleCoapClient, CoapCallback	Test that the data is returned and parsed right
E6	DeviceScanActivity adds the found results into the	DeviceScanActivity, SetDevi-	Test that the parsed data is forwarded to the

	user interface	cedata	user interface
F1	User interface displays all the found devices and their wanted values in a listview, appending new devices to the list when found.	Layout, activity_main, listitem_device	Test that the listview in the layout displays the type of the device and its values
F2	Listview appends new devices to the list when found and displays their values.	Layout, activity_main, listitem_device	Test that the listview appends a new device to the list when a new device is found, displaying its type and a value correctly
F3	Listview updates the device and value list	Layout, activity_main, listitem_device	Test that the listview updates the value of the found devices.
F4	The application keeps scanning for BLE-devices without crashing or shutting down as long as the application is open and active	BleScanner, Layout, activity_main, listitem_device	Test that the application keeps scanning for BLE-devices without crashing or shutting down as long as the application is open and active

4 THE IMPLEMENTATION

The thesis work is made for Huawei Y5II CUN-L21 Android phone using Android Studio. The phone was offered along with the sensor tag by Comsel System Oy.

4.1 Android Manifest

Android manifest is used in the Android Studio project to specify the required properties for the application such as the user permission to connect to the internet and enable using the bluetooth and the coarse location.

When the application is being run, the phone will also ask these permissions from the user.

```
package="com.example.jerel.bluetoothscanner">
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>

<permission android:name="android.permission.BLUETOOTH" android:label="BLUETOOTH" />
<permission android:name="android.permission.BLUETOOTH_ADMIN" />
<permission android:name="android.permission.ACCESS_COARSE_LOCATION" />


<uses-permission android:name="android.permission.INTERNET"/>
```

Figure 9. Android Manifest.

4.2 XML user interface

The user interface components are created with a XML-structure where the attributes and the view are being defined. The XML-structure in this project is divided into a linear layout that contains an inner layout and a textview for both the device name and its value(s). The XML-components are specified more specifically in the values-folder that contains the colors, the strings and the styles for the layout.

```
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textStyle="bold"
            android:text="Device:"/>

        <TextView
        android:id="@+id/device_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="start|bottom"
        android:layout_marginStart="4dp"
        android:textColor="@android:color/black" />
</LinearLayout>
```

Figure 10. XML user interface.

4.3 Main Activity class

Android API offers to the BluetoothScanner class the functionality to scan and communicate with both Bluetooth- and Bluetooth Low Energy- devices.

The user interface components are created and the Bluetooth Low Energy Beacon -scanner are initialized in the MainActivity class' OnCreate method. The Bluetooth Scanner constantly scans for the nearby beacons and once a device is found, it is added into the list view to display its basic data resources. The most important resources to display in this thesis work were the MAC address and the payload containing the Eddystone message. The Eddystone message includes a 10 bytes long namespace and a 6 bytes long instance that was, in this case, UUID of the Comsel beacon device. This data was transferred as a HTTP REST call to Comsel's Corona cloud to obtain the best quality IP-address of the beacon.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    list = (ListView) findViewById(R.id.found_devices);
    deviceValues = new HashMap<String, String>();
    m_ble = new Ble(this);
    m_bleScanner = new BleScanner(this);
    http = new HttpHandler();
    coapClient = new AsyncBleCoapClient(this);


    if (!m_ble.isSupported()) {
        Toast.makeText(this, "This device does not support BLE", Toast.LENGTH_LONG).show();
        finishAffinity(); // Exit application
    }

    if (!m_ble.isEnabled()) {
        Toast.makeText(this, "Please turn Bluetooth ON to scan devices", Toast.LENGTH_LONG).show();
        finishAffinity();
    }
    else {
        m_bleScanner.start();
    }
```

Figure 11. Main Activity class.

4.4 Eddystone

Eddystone is an open Bluetooth 4.0 protocol from Google that supports both Android and iOS. Eddystone support in the Android SDKs is based on a single method called Eddystone discovery which provides proximity estimations and works when the application is being active. The beacon system supports multiple data packet types; Eddystone-UID and Eddystone-URL. /12/ The first one (Eddystone-UID) is used in this thesis work.

4.5 Eddystone-UID

Eddystone-UID contains an identifier of a beacon. An app on the phone uses the identifier to trigger an event. Eddystone-UID is 16 bytes long and split into the namespace containing 10 bytes, and into the instance consisting 6 bytes. The instance is what differentiates individual beacons from each other. In the thesis

work the Eddystone-UID contains Comsel System Oy's namespace and the instance contains the specific UUID that is used to obtain the device's IP-address from Comsel's Corona cloud.

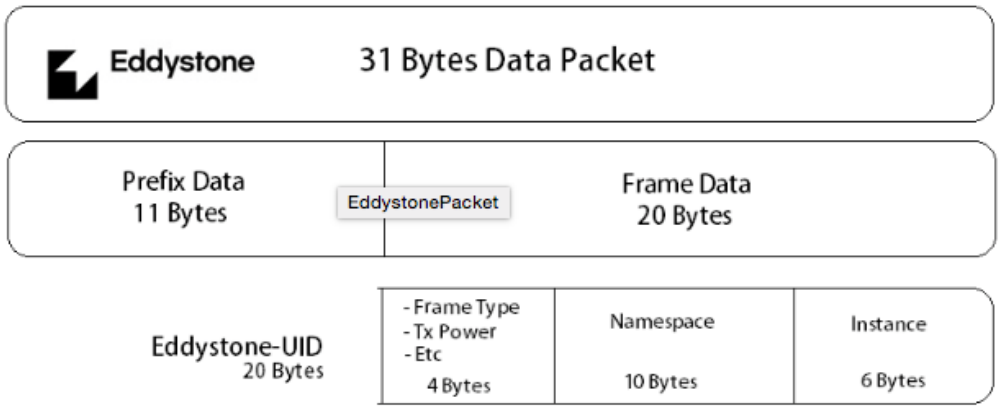

Figure 12. Eddystone beacon and Eddystone-UID. /13/

4.6 HTTP REST API

HTTP REST API provides a typical CREATE, READ, UPDATE DELETE (CRUD) API. GET has no payload, only an URL that defines which resource is wanted to be accessed. POST and PUT contain the payload and the user saves data from the client to the server. DELETE is for removing an existing resource from the server. The data is usually saved into a database.

Comsel Corona cloud provides a stateless application programming interface (API) which uses the HTTP REST API. The REST API is a stateless service. It means that the user or the server session state do not affect the data return performed by the API. When the client requests for the resource, the current state of the resource obtained with GET is always returned and it is not affected by the current state of the client or the server. The current resource is stated as it is in the database.

The HTTP calls are made asynchronously, which means that a new thread is allocated for the use of the process and so that the user interface updates and operates

while the HTTP call is being performed. When the HTTP call finishes, the applications callback function is executed and the response data can be handled in the function without interruptions.

4.7 Constrained Application Protocol

Constrained Application Protocol (CoAP) is a software protocol for simple electronic devices (such as power sensors, switches, valves and, as in this thesis work, beacons) to communicate over the Internet. CoAP supports translating HTTP for simplified integration with the internet. Importantly, CoAP also has a multicast support which is important for the Internet of Things -devices, which have less memory and power supply than the traditional internet devices. /14/

CoAP is mapped over IP-protocol and employs a two layer structure which consists a message layer and a request/response layer (see the picture below). The message layer supports a Confirmable (CON), a non-confirmable (NON), an Acknoledgement (ACK) and a reset (RST) types of messages. The message layer is for re-transmitting the lost packets and the request/response layer for methods like GET, PUT, POST and DELETE. (See Figure 13 below.)

Figure 13. CoAP layer structure.

The implementation of CoAP in the thesis work is Java based nCoap, which uses CoAP version RFC 7252 and includes both client and server sites. To gain access to HTTP, CoAP has to use a proxy over the IPv6 network. The URI scheme for accessing CoAP is ”coap://” (similar to http's ”http(s)://”). /15/

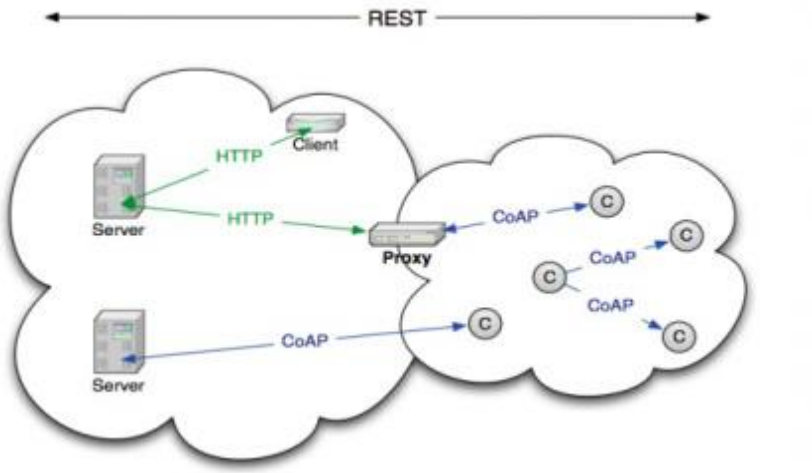

Figure 14. CoAP and HTTP co-operation.

4.8 CoAP in the thesis work

An nCoap-library in this thesis work implements a CoAP protocol. It offers a CoAP protocol implementation; functions for sending a CoAP-request and an observation and a subscription for the CoAP-resources. When the client subscribes to the server's resource, the server transfers and updates the resource to the client. This way the CoAP supports a two-way communication unlike HTTP. In addition, the server notifies about the status of the resource, meaning the client doesn't need to keep making callbacks to ask for it.

Like the HTTP call, CoAP call needs to be asynchronous so that the software can operate while the CoAP request are being made over the IPv6 address network. When the response arrives, the application performs a callback-function that uses the data to parse the response, to catch the beacon type and to fetch the wanted beacon data that is then displayed in the user interface.

The format for the CoAP request is coap://<url>/to/resource/<device's ip address/ + the path to get the device type or the measurement data. The Comsel cloud offers the CoAP server that is used to perform these CoAP requests. The response received tells what type of a device the beacon is.

Once the type of the beacon is recognized, the application performs another request to get the measured data based on what type of a device the found beacon is. For example, a district heat beacon receives a temperature measurement and a room humidity beacon a humidity level.

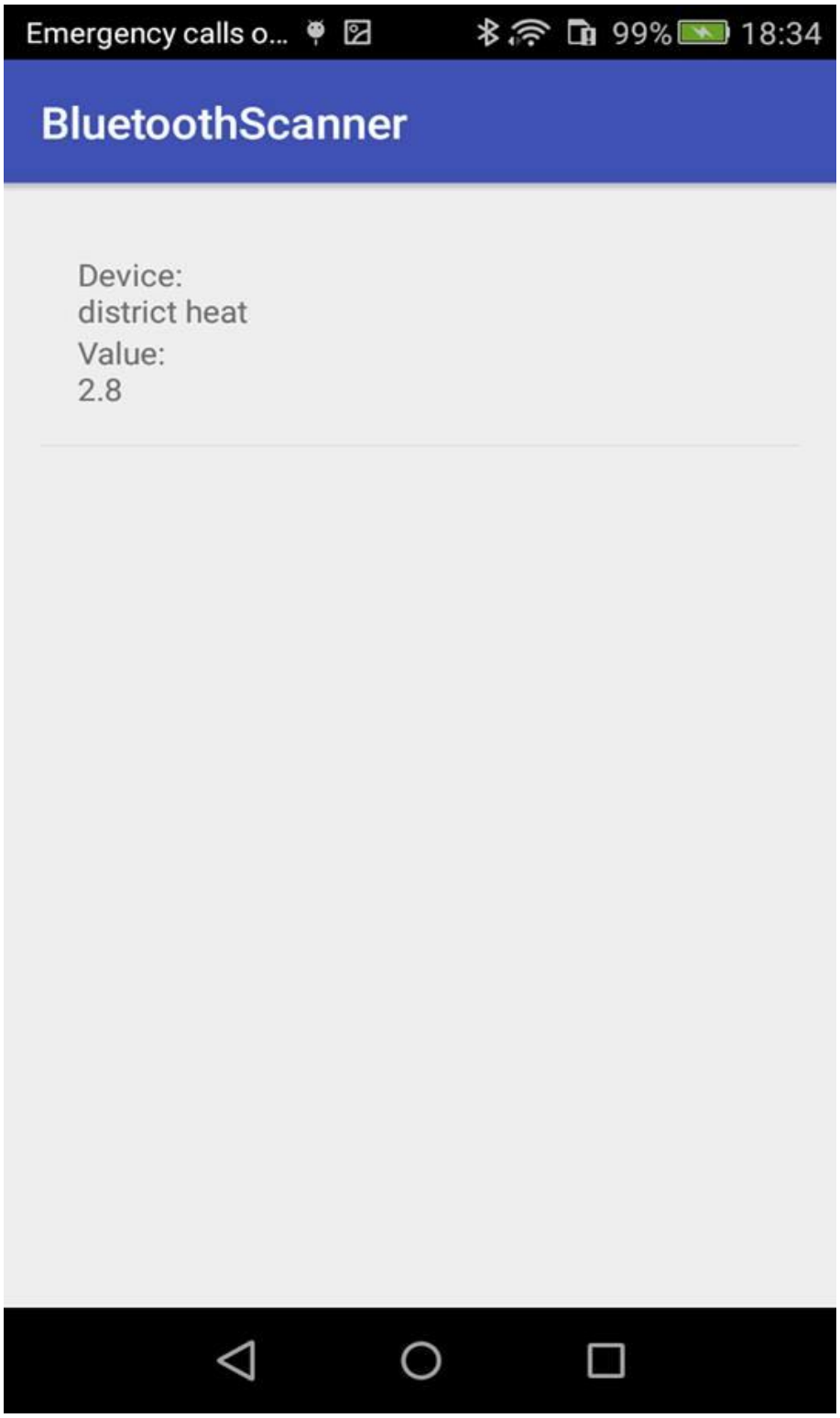

Figure 15. A final view where the found device(s) and their values are being listed in the user interface.

4.9 Test Report

In the end it is tested that all the steps in the test plan were tested and found working. Table 4 below shows a full list of the tests performed. All the tests performed turned out to be working as intended.

Table 4. Test Report

ID	Test **Procedure**	**Test Date**	**Test Result** **(1 success, 0 fail)**
A1	Test that the DeviceScanActivity's onCreate method creates the Listview for the devices	10.06.2017	1
A2	Test that the application checks if the Bluetooth is enabled. If not, it throws an alert.	10.06.2017	1
B1	Test that the scanner is starting to scan the devices	10.06.2017	1
B2	Test that if the location access is not given, the phone will perform a request for it.	10.06.2017	1
B3	Test that the BleScanner adds a found device to the list and keeps adding a new found devices in-	10.06.2017	1

	to it aswell by using a random generated number ending for both the found device and its value.		
C1	Test that the application parses the payload properly into a namespace and an instance	10.06.2017	1
C2	Test that non-Comsel devices will not be parsed forward. The namespace must match to that of Comsel's	10.06.2017	1
C3	Test that the devices matching to Comsel's namespace have their instance parsed to obtain the UUID	10.06.2017	1
D1	Test that the HTTP-class starts properly and the call is working	10.06.2017	1
D2	Test that the connection is enabled and an alert is displayed if not	10.06.2017	1
D3	Test that the address is correct and the callback	10.06.2017	1

	functions correctly		
D4	Check that the JSON-message in Corona cloud has the right parameters	10.06.2017	1
D5	Test that only the best quality IP-address is returned	10.06.2017	1
E1	Test that the IP-address is used when initializing the CoAP-request	10.06.2017	1
E2	Test that the IPv6 is enabled and throw an alert if not	10.06.2017	1
E3	Test that the correct type of a device is returned	10.06.2017	1
E4	Test that the path for the device's data request is correct and is based on the device type returned previously	10.06.2017	1
E5	Test that the data is returned and parsed right	11.06.2017	1
E6	Test that the parsed data is forwarded to the user interface	11.06.2017	1

F1	Test that the listview in the layout displays the type of the device and its values	11.06.2017	1
F2	Test that the listview appends a new device to the list when a new device is found, displaying its type and the value correctly	11.06.2017	1
F3	Test that the listview updates the value of the found devices.	11.06.2017	1
F4	Test the app keeps scanning for devices without crashing or shutting down	15.08.2017	1

5 CONCLUSION

The result of the thesis work is an application that can discover the nearby Bluetooth Low Energy beacons and parse the namespace and the instance from the payload's Eddystone message. If the namespace matches to the Comsel System Oy's namespace, the UUID contained in the instance is then used to perform a HTTP call to obtain the best quality IP-address supporting the Comsel network interface by parsing a JSON in Comsel's Corona cloud. With the obtained IP-address, the application performs a CoAP-request to first obtain the device type and then the desired measurement data with a CoAP-callback. Lastly, the result is displayed in the user interface to the user.

The thesis work and its three steps required more than expected in the beginning. The biggest step in the thesis work was to be able to scan the Bluetooth Low Energy -beacons and display them in the user interface the way that the payload message could be parsed into a namespace and an instance. In addition, CoAP protocol was a completely new protocol that required a lot of research and understanding to be used correctly.

One of the biggest changes and stepbacks during the thesis work was that the beacon system was an Eddystone beacon system instead of a normal Bluetooth Low Energy -beacon system. This meant that the thesis work had to be done using a newer 5.1 Nougat Android version and several parts in the code had to be rewritten to support this version. Nevertheless, the thesis work was finished in time and Comsel System Oy was very satisfied with the result. All the required steps were completed in time.

For the future development, the nice to have feature for direct application level communication between the device and the application could be implemented. In addition, now the application only displays the key values wanted from the specific types of beacons. The further developed application could include selecting a specific device from the list to display more specific and detailed values from it.

6 SOURCES

/1/ IoT

Reference 28.07.2017

http://www.businessinsider.com/what-is-the-internet-of-things-definition-2016-8?r=US&IR=T&IR=T

/2/ IoT

Reference 28.07.2017

https://www.forbes.com/sites/jacobmorgan/2014/05/13/simple-explanation-internet-things-that-anyone-can-understand/#265c16ad1d09

/3/ Comsel System Oy

Reference 26.08.2017

http://comsel.no/about.html

/4/ Texas Instrument CC2650 Sensor Tag

Reference 28.07.2017

http://www.ti.com/product/CC2650

/5/ Huawei 5YII Cun 121

Reference 28.07.2017

http://imged.pl/huawei-y5ii-cun-l21-czarny-polska-dystrybuc-sklep-2-18811957.html

/6/ Android operating system

Reference 28.07.2017

http://gs.statcounter.com/press/android-overtakes-windows-for-first-time

/7/ Android versions

Reference 28.07.2017

https://www.jsys.co/android-flavors-and-its-features/

/8/ Android Structure

Reference 30.07.2017

https://developer.android.com/guide/platform/index.html#linux-kernel

/9/ Android Structure

Reference 30.07.2017

https://developer.android.com/guide/platform/index.html#linux-kernel

/10/ Android Studio

Reference 01.08.2017

https://developer.android.com/studio/intro/index.html

/11/ Java

Reference 01.08.2017

https://www.java.com/en/download/faq/whatis_java.xml

/12/ Eddystone

Reference 02.08.2017

http://developer.estimote.com/eddystone/

/13/ Eddystone

Reference 02.08.2017

https://support.kontakt.io/hc/en-us/article_attachments/203776115/Eddystone-Data-Packets-for-Adrian_1.png

/14/ CoAP

Reference 03.08.2017

http://www.networxsecurity.org/members-area/glossary/c/coap.html

/15/ CoAP

Reference 03.08.2017

https://zaidmufti.wordpress.com/2016/08/23/coap-an-application-layer-protocol-for-smart-dust/