Mobile Sensor Gateway
A multi-purpose mobile sensor application

Linus Forsberg
Maximilian Falkenström

Main field: Computer Science
Program: Application development
Bachelor thesis
15 credits
Spring 2019
Supervisor: José Font
Examiner: Dipak Surie
Final seminar: 2019-05-31
Abstract

This thesis describes the process of creating a platform-independent mobile application for connecting mobile devices to wireless sensors using Bluetooth Low Energy, collecting data from connected sensors and uploading the collected data to a cloud storage service. As consumers and researchers use more sensors and other Bluetooth-devices, one could argue that there is a need for simpler and standardised solutions to working with these. A literature study has been conducted where information on related research has been collected and important information about the necessary software components has been evaluated. In combination with the literature study, an IT artefact has been developed in the form of a mobile application that has been tested according to collected requirements to ensure the application's functionality. The purpose of this work is to contribute with a clear scientific process over what is required to create a mobile application of this kind and what potential difficulties exist in present-day design of this type of applications. The results show that some manufacturers may not be following the standards for Bluetooth data communication, thus making it hard to write generalized methods for retrieving data from sensors of any type or manufacturer.

Keywords: Bluetooth Low Energy, Sensors, Cross platform mobile application, Cloud storage, Internet of things
Sammanfattning


Sökord: Bluetooth Low Energy, Sensorer, Multi-plattform mobilapplikation, Molnlagring, Sakernas internet
Contents

ABBREVIATIONS ................................................................................................. 1

1. INTRODUCTION ............................................................................................. 2
  1.1 AREA OF CONCERN .................................................................................. 2
  1.2 PROBLEM DEFINITION .......................................................................... 3
  1.3 PURPOSE ................................................................................................. 3
  1.4 COLLABORATION ................................................................................... 4
  1.5 RESEARCH QUESTION ........................................................................... 5
  1.6 RESEARCH METHODS ............................................................................ 6
    1.6.1 Design and creation ......................................................................... 6
    1.6.2 Literature study ................................................................................ 6
    1.6.3 System requirement analysis .......................................................... 7
  1.7 RELATED WORKS .................................................................................... 7
  1.8 EXPECTED RESULTS ............................................................................... 8
  1.9 SOFTWARE STACK .................................................................................. 9
  1.10 PRIVACY AND SECURITY ASPECTS .................................................... 9
  1.11 DISPOSITION .......................................................................................... 9

2. METHOD ............................................................................................................ 10
  2.1 LITERATURE REVIEW ............................................................................ 10
  2.2 DESIGN AND CREATION ........................................................................ 11
    2.2.1 Disadvantages of using Design and creation .................................... 13
  2.3 SOFTWARE ............................................................................................... 14
    2.3.1 Cross platform mobile applications ................................................. 14
    2.3.2 Programming languages ................................................................. 15
    2.3.3 Xamarin and Xamarin.Forms .......................................................... 15
    2.3.4 MVVM software architecture ......................................................... 16
  2.4 HARDWARE ............................................................................................... 17
  2.5 BLE PLUGIN LIBRARY ............................................................................. 18
  2.6 AZURE CLOUD STORAGE ..................................................................... 18
  2.7 SYSTEM REQUIREMENT ANALYSIS ....................................................... 18
  2.8 EVALUATION OF THE IT ARTEFACT ..................................................... 19

3. SOLUTION ......................................................................................................... 20
  3.1 THE SYSTEM OF THE IT ARTEFACT ....................................................... 20
  3.2 XAMARIN.FORMS APPLICATION .......................................................... 20
    3.2.1 Data models...................................................................................... 21
    3.2.2 User interface .................................................................................. 21
  3.3 NUGET PACKAGE MANAGER ..................................................................... 23
  3.4 ENTITY FRAMEWORK CORE .................................................................. 23
  3.5 APP PERMISSIONS .................................................................................. 24
  3.6 BLE PLUGIN ............................................................................................. 24
  3.7 AZURE CLOUD STORAGE ....................................................................... 26

4. RESULTS ............................................................................................................ 27
  4.1 ARTEFACT EVALUATION ......................................................................... 27
  4.2 BLUETOOTH CONNECTIVITY ................................................................. 27
4.3 Bluetooth standardisation........................................................................................................ 27
4.4 Customization to receive data from Joyway sensors.............................................................. 28
4.5 Connectivity and receiving data from Arduino cards.......................................................... 29
4.6 Uploading data to the cloud...................................................................................................... 29
4.7 Requirement testing.................................................................................................................. 30
  4.7.1 Requirement 1: Search for BLE devices.......................................................................... 31
  4.7.2 Requirement 2: Connect with BLE devices................................................................. 31
  4.7.3 Requirement 3: Receive data from BLE devices............................................................ 32
  4.7.4 Requirement 4: Date and time interval functionality.................................................... 32
  4.7.5 Requirement 5: Upload data to cloud storage............................................................... 33
  4.7.6 Requirement 6: Store data properly in the cloud.......................................................... 33
  4.7.7 Requirement 7: Cross-platform usage........................................................................... 33
4.8 Summary of results.................................................................................................................. 34

5. Analysis and discussion............................................................................................................ 34
  5.1 Comparing results to previous research.......................................................................... 34
  5.2 Results of conducting the literature review..................................................................... 35
  5.3 Evaluation of the IT artefact............................................................................................... 35
  5.4 Challenges in Bluetooth standardisation.......................................................................... 36
  5.5 Summary.............................................................................................................................. 36

6. Conclusions and further research.......................................................................................... 37
  6.1 Scientific contribution.......................................................................................................... 37
  6.2 Future research.................................................................................................................... 37

References..................................................................................................................................... 38
### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>ATT</td>
<td>Attribute Protocol</td>
</tr>
<tr>
<td>BLE</td>
<td>Bluetooth Low Energy</td>
</tr>
<tr>
<td>BLE Plugin</td>
<td>ACR Reactive Bluetooth LE Plugin (Library for communicating over BLE)</td>
</tr>
<tr>
<td>BSIG</td>
<td>Bluetooth Special Interest Group</td>
</tr>
<tr>
<td>BSN</td>
<td>Body Sensor Network</td>
</tr>
<tr>
<td>CRUD</td>
<td>Create, Read, Update and Delete</td>
</tr>
<tr>
<td>EF</td>
<td>Entity Framework</td>
</tr>
<tr>
<td>GATT</td>
<td>Generic Attribute Profile</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>MUEP</td>
<td>Malmö University Electronic Publishing</td>
</tr>
<tr>
<td>ORM</td>
<td>Object-Relational Mapping</td>
</tr>
<tr>
<td>SQL</td>
<td>Standard Query Language</td>
</tr>
<tr>
<td>UUID</td>
<td>Universal Unique Identifier</td>
</tr>
<tr>
<td>XAML</td>
<td>Extensible Application Markup Language</td>
</tr>
</tbody>
</table>
1. Introduction

The areas of Ubiquitous computing and the Internet of Things (IoT) are revolutionizing the society. A report suggests that 25 to 50 billion devices will be connected to the internet by 2020 [2]. The term ubiquitous computing was coined by Mark Weiser in 1991 and propose that computers should be embedded into the environment until they disappear [3]. This groundbreaking paper led to the emergence of IoT which has been called the fourth wave of digitalization and may affect all parts of society in the coming future [4]. IoT is a network which connects uniquely identifiable devices or “things” to the internet [5]. The devices possess the ability to sense the environment and possess potential programmability capabilities [5]. Information from the devices can be collected and the state of the devices can be changed [5].

1.1 Area of concern

Within the area of IoT there are many different application purposes, designed for making lives easier and more fulfilling for its users [6]. One of these purposes concern using sensors for collecting information about the user [7]. A word used for this kind of technology, which is worn on the body is the term “wearable” [8]. There are several types of applications, for example step-counters and activity monitors which measures heart rate etc. Many of these applications are using the built-in sensors that most mobile devices have today. However, there is also the possibility to connect the mobile device to external sensors which utilize Bluetooth Low Energy (BLE) for sending data from the sensor to the mobile device [9]. These sensors are small portable devices powered by batteries and can detect changes and/or events in their environment, for example temperature, humidity or pressure. Bluetooth is a global technology standard that allows devices to communicate with each other via radio links [9]. The latest enhancement of Bluetooth technology, BLE, is a lighter version of Bluetooth but with lower data transfer rates, among other things and thus gives a lower power consumption, which gives the devices a longer battery life with use of regular clock batteries [9].

Wearable devices, connected by BLE, are becoming more and more common. A lot of the time the devices themselves are either sensors, or at least includes one or more sensors. A mobile application is often needed to collect data from the sensors and usually comes bundled with these devices, but the data is usually not exportable to usage outside of the application [10], [11]. Another technology which is growing steadily is cloud storage services [12]. Cloud storage services is a technology which makes it possible to send data from a device, for example a computer or a smartphone, to store it on servers, provided by different companies such as, Amazon [13], Google [14] or Microsoft [15]. By uploading the data from sensors to the cloud, storage can be freed from the device and the data can be accessed from anywhere with an internet connection. The technologies of using smartphone applications, BLE sensors and cloud storage creates new possibilities, of building custom applications, mobile or otherwise, that utilize data collected from sensors in different
ways. A user might build a health tracker with his or her own selection of sensors, or maybe combine the data from multiple sensors in order to get some other type of information. These technologies combined gives users a lot of flexibility in what kind of data that can be recorded and for what purposes it can be used.

1.2 Problem definition

The interest from the public for IoT and wearables is growing, as the public becomes more technically knowledgeable and the technology is getting better, cheaper and easier to use [16], [17]. There are ready-made solutions that can be used where sensors are utilized with accompanying applications [10], [11]. This, however, entails a limitation for users where one may need to use multiple applications to be able to use multiple sensors from different manufacturers.

A specific area that may have this problem in the near future is for example the health sector. Sensors could be used by medical doctors to track the status of a patient between visits, for example tracking heart rate, blood sugar levels etc. The problem in this scenario is that different patients may need different combinations of sensors and hospitals may over time need to purchase the same type of sensors from different manufacturers. This research aims to find out if there is a generic solution to this type of problem, one that isn’t concerned about what sensors are used or from what manufacturer and create a framework that can be followed.

1.3 Purpose

The interest from the scientific community is also growing fast as can be seen by the growth in the number of scientific articles dedicated to sensor applications in recent years. As figure 1 shows, the number of articles published that includes the keyword “sensor” in the ACM has almost doubled between 2008 and 2018 [18]. That data was gathered by searching for “sensor” on ACM and sorting by year to get the number of published articles for each year. It seems from reviewing the existing literature, that researchers are developing their own software from scratch which means a lot of time and resources that could be used for the actual research is put into software development. This could be mitigated if there existed a framework that could be utilized for quickly creating a generic multi-purpose sensor application. At the moment there doesn’t seem to exist a well described framework for creating a sensor application for general purpose use. The results from this thesis can therefore provide researchers with a framework and guidelines for conducting similar research projects where the researchers can use the gathered knowledge from this thesis in conducting their own research.
This thesis aims to be of interest for scientists that wants to conduct more research regarding BLE, IoT, sensors, wearables and the standards around these. It will also be of interest to industry professionals working in these fields as the software and hardware standards are crucial to enabling the type of applications that are suggested in this thesis. It may also be of interest to people of the general public that want to learn more about the devices they use.

1.4 Collaboration

The research work conducted in this thesis is achieved and funded in collaboration with CGI which is a world-wide software consulting company [19]. CGI presented an idea of an application that they wanted to have developed. The application that they had in mind is a mobile application for connecting to BLE sensors of any kind, regardless of manufacturer, which can collect data from the sensors and then upload the data to cloud storage. The fact that CGI wants this application to be developed, gives indication that there is a need from the industry for an application like this, which support the relevance of this thesis. The collaboration with CGI means that there are some points that needs to be considered regarding the development of the application. CGI prefers the code to be written in Microsoft’s programming language C# [20], and the cloud storage service used, to be Microsoft’s Azure Cloud Platform [15] as these are the main technologies being used inside the company.
1.5 Research question

This thesis focuses on the possibility of creating a cross-platform mobile application which functions on both the Android and iOS mobile platforms, for connecting the mobile device to external BLE devices, fetching data from the sensors as well as uploading the collected data to the cloud. During the process of developing this application the following research question is investigated:

- **How to create a cross-platform mobile application for connecting wireless Bluetooth low energy sensors regardless of manufacturer or type, to a mobile device and upload the collected data to the cloud?**

The research question is relevant because as people start to use more and more devices, whether they be wearables or IoT-devices, they need more and more apps to interact with these devices. Making it possible to connect to these devices and use the data from them with only one or two applications would help people in streamlining their experience.

Imagine having an mobile application that serves as a middleware and uploads the data from nearby devices to the cloud and being able to download one app that interacts with that data and provide the same or similar functionality to using all of the apps from the different manufacturers of the hardware. By creating a standard for how this data would be saved and retrieved from the cloud, developers would be able to develop mobile applications that enable the user to connect to their cloud data and use it in their applications. Users would be able to download an app from a developer or from another company than the manufacturer of the physical device, for example if the user did not enjoy the experience of the mobile app but liked the physical device.

If these standards were created and implemented, companies that want to focus on building hardware, like sensors and wearables, but aren’t interested in building software, would be able to build the hardware and companies that want to focus on software could build generic software that would work with any of the devices from the hardware companies. Today consumers are usually stuck with the application that comes from the manufacturer of the physical device. Users have different taste and not all users like the same apps. With an ecosystem like the one explained, a user could select a physical device that they like and use an application that they find intuitive and well designed. This system would also lower the barrier of entry as a person or company would be able to either build software or hardware and would not have to build both as is often the case today. By being able to build an application without worrying about hardware or vice versa, there will be more variety and more competition in the market.

Scientists could use this type of application to speed up the development portion of many scientific experiments as the scientists could use an application like this to get data from sensors in experiments and putting their time towards the actual experiment.
The goal of this thesis is to show whether this is possible and, importantly, what the limitations of today may be as well as create a framework for how it could be done. For this idea to be possible in the future, some special interest group, company or scientists will need to create the standard that will be adopted. Research may also be made into the surrounding technologies and hardware for this to happen and this thesis may expose some of the areas that requires more research.

1.6 Research methods

This thesis is based on a combination of two different research methodologies, the design and creation method as well as a literature study.

1.6.1 Design and creation

The main focus of this thesis is to develop a functioning IT artefact in the shape of a cross-platform mobile application, which can connect to BLE sensor devices, collect data from them and upload the data to cloud storage. When conducting research where software is developed, it is important to base the software development process in a well-established research methodology, such as the design and creation method, to legitimize the software development part of the thesis as properly conducted research. Design and creation is a research method which is well suited for the creation of new IT artefacts [21]. In this thesis the design and creation method is used to develop a cross-platform mobile application that offers a simple way for a user to connect a mobile device to any BLE sensor, regardless of manufacturer, gather the data from the sensor and passing that data along to a cloud service provider. The software development follows an iterative process which is further explained in the Method section. Even though the design and creation method is well suited for this thesis, there are some potential disadvantages that needs to be considered. Examples of these disadvantages are that it can be difficult to prove that the developed IT artefact is actual research and not just normal design and creation from an industry standpoint, and the scientific results might be hard to reproduce and generalize [21].

1.6.2 Literature study

In addition to the design and creation part of this thesis, a literature study needs to be conducted to establish the scientific basis on which the thesis stands on [21]. It is important to gather knowledge about what has already been done in this field of research, as well as learning about the different software frameworks and protocols that can be used to create an IT artefact of this type. Literature that is of interest for this thesis, is for example, similar research projects that has been conducted in the field of combining mobile application development with Bluetooth Low Energy sensors, and literature that explains the technologies that is needed for creating this particular kind of IT artefact. Gathering knowledge in form of a literature study,
combined with the design and creation research, will provide a basis for answering the research question stated earlier in this section.

1.6.3 System requirement analysis

In the early stages of the design and creation process, a system requirement analysis is performed in collaboration with the collaborator CGI to gather information about the functionality needed for the application. During the software development process, the application is tested continuously against the requirements, to establish that it is functioning properly. Important aspects of the functionality are connecting the mobile device to nearby BLE sensors, collecting the data transmitted by the sensors and then transmitting the collected data to the cloud storage service.

1.7 Related works

There is a lot of previous research conducted in the areas of wearables and IoT concerning health and medical care [22]–[26]. One of these articles show how sensors could be used for following a patient’s rehabilitation process outside of the hospital [22]. Researchers developed a system which allows physicians to get a better overview of the rehabilitation process of a patient based on a variety of values collected through the system through fiber optics, Electrocardiography (ECG), Electroencephalography (EEG), Electromyography (EMG), stabilogram and biological feedback [22]. Clinical trials on patients showed that the system produced promising results and accelerated patients' rehabilitation process time, compared to patients who had undergone a conventional rehabilitation process [22].

Other research projects involved developing a prototype Body Sensor Network (BSN) which gathers information about the vital functions of a patient while the patient is in the home environment, using smart textile with sensors integrated into the fabric [23], [25].

Another research project involved body sensors connected to mobile devices and cloud services [24]. Data was collected from sensors that could be used to make clinical diagnoses as well as continuous physiological monitoring of patients [24]. The researchers arrived at a technical solution to some common problems that exist with sensors, such as limited storage, power supply and processing capabilities [24].

In a different paper, researchers presented a systematic review of the existing state of the art applications for IoT-enabled healthcare applications [26]. The researchers concluded that IoT healthcare has big potential for enabling faster and safer healthcare, lowering patient costs and improved patient-centered practice [26].

Some of the existing literature deals with the various techniques used in the field of IoT and the difficulties that exist for creating this type of applications [27], [28].
One paper describes the fundamental concepts and critical points in creating IoT devices, presenting projects involving a vegetable production and distribution system and a public transportation monitoring system [27].

Another paper describes the problems and solutions to building smartphone sensing applications that can run continuously over a long period of time without losing reliability and efficiency [28].

A great source of information about all things concerning Bluetooth Low Energy is the book “Inside Bluetooth Low Energy” by Gupta, which explains all the concepts needed for working with BLE devices [9].

There doesn’t seem to be much existing research done to specifically deal with applications that allows a user to connect with nearby BLE devices regardless of manufacturer and collect the data from it. Most previous research deals with specific sets of sensors chosen by the researchers. This means that the software built in these research projects are most likely purposely tailored to these specific sensors and not able to connect to a wider variety of sensors regardless of manufacturer. The research conducted in this paper will therefore differ in that sense that it tries to offer a solution that is more generic, and therefore can be used for any kind of purpose that can be of interest for a potential user of this kind of application. The lack of research on this specific topic does not mean that there are no existing similar applications on the market today. There are for example the nRF Connect application for Android and the LightBlue application for iOS, which can connect to many different types of BLE devices. What is lacking in this context, is research describing how to achieve this type of application from a scientific point of view, a framework for the software development process of creating such an application. This is the knowledge gap that this thesis intends to fill.

1.8 Expected results

The expected results from this research will be a cross-platform mobile application which connects BLE sensors of any type or manufacturer to a mobile device and lets the user choose available sensors to collect data from. The application then sends the collected data forward to a cloud storage service and stores it in a structured manner that another program can easily read. The development of the mobile application in combination with the literature study, which reviews the recent research in interesting areas connected to this topic will result in new knowledge in form of an answer to the formerly stated research question. This results in a software development framework which can be used as reference and gives the reader a scientific approach of how to create an application that could be used to collect data from any sensor. The use cases of such an application is infinite and could be of usage for both the scientific and industry communities.
1.9 Software stack

The cross-platform mobile application is built using Xamarin. Xamarin is a cross-platform mobile development platform owned by Microsoft, which serves as a framework for creating cross-platform mobile applications using the C# programming language [26]. The main selling point of Xamarin is that it allows software developers to share most of the code for the application between different systems [26]. This allows for building both Android and iOS applications without writing all the code twice, once for each specific platform.

1.10 Privacy and security aspects

This thesis focuses on creating the framework for how to create this type of application. The framework supports data security in the sense that developers that use it to build applications can choose to include security measures such as encrypting the data before it is sent to the cloud but it is not something that is included in this thesis as the focus is on how to create this type of application and not the security aspects of it. As for privacy, developers that use this framework to create an application could use different cloud services, host one themselves or maybe let the user define where they want their data to be sent. This means that it’s impossible for the authors to make any assumptions about the privacy aspect of the data storage.

1.11 Disposition

The next section presents a more detailed explanation of the research methods used to conduct this research: The design and creation method and the literature study, as well as the software frameworks and hardware used during the research. Following the method section is the solution section, which describes the developed IT artefact and its different parts in more detail. After the solution section comes the results section, were the gathered results from the research is presented. The section after that involves a discussion and analysis about the gathered results. The thesis ends with a conclusion and proposals to possible future research that could be conducted in the context of this thesis.
2. Method

This section focuses on describing the research methodologies used for conducting this thesis. The chosen research methods are presented and discussed, along with motivations to why the specific methods are well suited for this type of research as well as the potential risks of using design and creation as a research method and how these risks are mitigated. Information is presented about the process of deciding which software frameworks to use, the choice of software architectural pattern and a brief description of the hardware used for the research. The last subsection describes what functionality needs to be implemented in the cross-platform mobile application, in order to achieve its purpose of communicating with BLE sensors, receiving the collected data and uploading it to the cloud storage service.

2.1 Literature review

The first part of this thesis consists of conducting a literature review. The literature review is necessary for supporting the scientific base on which this thesis stands on [21]. Conducting a review on the existing literature makes it possible to establish that this topic is worth researching. The literature review gathers information related to the research question, to get a clear picture of what has already been done in related research areas and to make sure that this thesis is not repeating some other researchers work [21]. Conducting the literature review is also a way to get a better understanding of the technologies that can be used for creating this type of IT artefact and the methods that are used for developing such artefacts. The literature studied are mainly research papers from technical conferences and scientific journals which touch on the subject of using sensors for gathering data for various purposes [23], [25]. Other literature studied is documentation for the software frameworks used in the development of the IT artefact [9], [29][28], and books on research methodologies [21], [30].

The structure of the literature review is based on seven different activities as they are described by Oates in the book on research methodology [21]:

- Searching
- Obtaining
- Assessing
- Reading
- Evaluating
- Reviewing

When searching for useful material, mainly internet databases for academic publications such as Libsearch, IEEE Xplore and ResearchGate are used. The Malmö University Electronic Publishing (MUEP) database is also used to look at the thesis works of previous students. Various search terms are tested in different combinations of words to get a satisfiable amount of results. Some of the literature is also to be found using regular internet search engines such as Google and
DuckDuckGo. When searching for previous research articles, the keywords that give satisfying results are “Bluetooth Low Energy”, “Sensor”, “Health”, “Cloud” and “Smartphone”, as well as synonyms and acronyms of these such as “BLE”, “Bluetooth LE”, “Medical”, “Healthcare” etc. A majority of the research articles used in this thesis is found using some combination of these keywords.

A lot of material is also to be found by reading the sources from the articles which are of interest, a method known as “snowballing” [31]. Obtaining the publications is possible by using the student login credentials for Malmö university which gives access to downloading the literature in PDF format from the internet. When assessing the literature, it is important to ensure that the source is credible [21]. The reading material used in this thesis is therefore analyzed by asking critical questions about the authors and their work. For example, what research background does the authors have? Are the articles peer-reviewed and published in a trustworthy academic sense such as academic conferences or journals?

Some of the references used in this thesis are internet sources which are not published in scientific publications. These sources are not to be seen as replacements for the peer reviewed material, but rather as complementary source material for references to official documentation for the programming language and software frameworks used in this thesis.

2.2 Design and creation

Apart from conducting a literature review, the main focus of this research thesis consists of producing an IT artefact in the shape of a cross-platform mobile application. The functionality of the application consists of connecting a mobile device such as a Samsung Android phone or tablet or an Apple iPhone or iPad to any nearby Bluetooth Low Energy devices, gather the sensor data from the sensors and uploading it to a cloud storage service such as Microsoft’s Azure platform. When developing an IT artefact for research purposes it is necessary to follow well-established scientific methods for the software development process, in order to strengthen the research credibility of the developed artefact. A research strategy which is well suited and commonly used for developing IT artefacts, is the design and creation strategy [21]. It is a well-established and documented method used by researchers when creating IT artefacts and is the main reason for the decision to use it for this thesis. When planning and conducting design and creation research, it is important to follow the already established principles that exist for this method of research [21]. This design and creation research follow the “learning via making” process which is an iterative process that consists of five different steps (See figure 2).
The following presents an explanation of the five steps of the “learning via making” process as they are described by Oates followed by how they are used in this thesis:

- **Awareness** - “The recognition of a problem which should be solved” [21]. In this case the problem is: how to create a cross-platform, general purpose BLE sensor application.

- **Suggestion** - “A solution to how the problem can be solved is presented” [21]. A requirement analysis is performed which gathers information about what functionality should be implemented to solve the problem. A design is devised based on the requirements, for how an application which connects to BLE sensors could be implemented.

- **Development** - “The implementation of the IT artefact which solves the problem” [21]. The BLE sensor application is developed according to the necessary requirements gathered from the requirement analysis.

- **Evaluation** - “Judgement of the value of the IT artefact and how it diverge from the expectations” [21]. The BLE sensor application is evaluated with the help of testing to see that the application is working properly and that the necessary requirements are fulfilled.

- **Conclusion** - “The newly obtained knowledge is presented and possible further research is proposed” [21]. The acquired results from testing the application gives feedback on how the application functions and if it needs to be redesigned to make it function more properly. The data gathered from testing the application provides a basis from answering the research question stated in the introduction section.

When using the “learning via making” process in the development of the IT artefact, the steps are not strictly followed step-by-step, as a more fluid overlapping process
tends to be more productive for the research [21]. For example, more awareness of the problem can be gathered by thinking about different solutions and evaluating a design might lead to discovering new insights about the problem. This means that during the research, the steps are performed in a non-linear fashion and the five steps are iterated as many times as needed until the IT artefact is giving satisfiable results compared to the requirements.

2.2.1 Disadvantages of using Design and creation

There are some potential risks of using design and creation as a method to create new scientific material. These risks can, if they are not mitigated, lead to criticism of the validity of the research. In the book on research methods written by Oates there are several disadvantages which are presented below, together with motivations to how they are mitigated in this thesis:

- “You may be challenged to justify why your work is not just ‘normal’ design and creation” [21].

  What makes this thesis work research and not ‘normal’ design and creation is that it is providing the scientific community with a framework for developing a cross-platform general purpose sensor application which could be used as a reference to simplify research and development by other researchers in the future.

- “It is risky if you do not have the necessary technical or artistic skills. Enthusiasm is no substitute” [21].

  The necessary technical skills have been acquired in part through conducting a literature review, but also through several years of education on topics that are directly related to the research area, such as object oriented programming, mobile application development, software engineering, computer science and research methodology.

- “It can be difficult to generalize to different settings from the use of an IT artefact in a single situation” [21].

  The IT artefact developed during this thesis is built for general purpose use, which means that it can be used in different settings and not just a single situation.

- “The (apparent) success of an IT artefact may depend on the researchers being present – once they’ve gone, an IT method or system may not work so effectively” [21]

  The process of developing the IT artefact is described in sufficient detail so that it can be understood and reproduced by other people with knowledge about software engineering and mobile application development.
• “It may produce perishable research. Rapid advances in technology can invalidate the research results before they have been tried out in a real-life context or even before they have been written up and published” [21].

This research is built on several technologies which are steady on the rise, such as IoT devices, smartphones and cloud storage. These technologies are relatively new and gaining rapidly in popularity. There are no signs as of today that these technologies should become obsolete in many years to come which makes this research both relevant and long lasting.

2.3 Software

When discussing the application functionality with the collaborator at CGI, one of the requirements for the application was that it has to be built for Android, but it would be preferable if it works for iOS as well. It would not be possible to develop two separate applications during the limited time span that is set aside for this research. This means that it is necessary to move away from native Android or iOS development as these use different languages, and thus it is not possible to reuse any of the code. To be able to produce a working mobile application for both Android and iOS, the decision has been made to look into technologies that supports cross-platform development. When developing applications for multiple platforms, one of the pitfalls is the complications of getting access to the operating systems native application programming interfaces (API’s). The native API’s are used to gain access to a devices’ hardware specific functionality, such as the GPS or Bluetooth. As the application that is developed in this research makes use of the devices’ Bluetooth functionality, the pre requirement in researching possible cross-platform frameworks is therefore that there needs to be support for this functionality from within the framework.

2.3.1 Cross platform mobile applications

There are a few frameworks that can be used for cross-platform mobile application development that are also able to access native API functionality, some of them being: React Native, Xamarin.Forms and Cordova. The decision ultimately fell on Xamarin for a few different reasons. Cordova is web-based and does not compile down to native code which means it has worse performance. React Native and Xamarin both utilize the OS’s own native APIs and compile down to native code. This means that they are both performing fast in runtime and make it possible to reuse most of the code when developing applications for multiple hardware platforms [33]–[35]. The main reason for choosing Xamarin instead of React native is that Xamarin uses C# instead of JavaScript, which is better suited for the researchers as well as a requirement from CGI.
2.3.2 Programming languages

C# is a programming language which is very close to Java in its syntax [36]. This is a strong advantage since the researchers of this thesis have a lot of previous experience programming with Java, which implies a shorter learning curve. Both Java and C# are members of the C family of programming languages and both aim at being a simplified language derived from C++ by eliminating some of the complexities of C++ [36]. Other than being almost syntactically identical to Java, the languages also share the following traits [36]:

- Both languages compile to an intermediate format that is run in a managed environment.
- Each runtime environment provides support for automatic garbage collection.
- All classes in both languages descend from the Object class and are allocated on the heap when they are created.
- Everything must belong to a class, which means that there are no global functions or constants.
- Both languages abandoned multiple inheritances, although implementing multiple interfaces is allowed.
- Both C# and Java use exceptions for error handling.
- Arrays are bound checked.
- Inline code comments are used to produce API documentation.
- They both use packages/namespaces to avoid type collision.

The preference from the partner CGI to use C# in combination with the researchers’ former experience with Java and the similarities between the two languages suggests a shorter learning curve. Therefore, the decision is made to use C# for the application development part of this thesis.

2.3.3 Xamarin and Xamarin.Forms

There are two different types of Xamarin which can be used for mobile application development, Xamarin (also called Xamarin Native) and Xamarin.Forms. The latter makes it possible, not only to share the business logic between Android and iOS but also the graphical user interface (GUI). In basic Xamarin, the business logic is written as shared code, but the user interfaces are programmed separately for Android and iOS. This makes software development in basic Xamarin more time consuming and requires knowledge about designing UI for both Android and iOS.
Xamarin.Forms does have some drawbacks. For example, there is not the same level of fine-grain control, which may result in the GUI not looking pixel-perfect. It is also not the best choice when there is a lot of native API functionality needed for the functionality. For the kind of application that is developed during this thesis this could be a problem as the native Bluetooth functionality is needed for the application to function properly. Xamarin.Forms does not natively support cross-platform use of BLE connectivity. It has to be implemented separately for each platform. However, there are workarounds for this problem as there are third-party libraries that implements a common interface to communicate with BLE on both Android and iOS.

There are several third-party libraries which make it possible to set up a connection to external devices via BLE and they have basically the same type of functionality. The library that is used in this project is “ACR Reactive Bluetooth LE Plugin” (BLE plugin) [29] as it is a popular library and it is regularly maintained and updated.

The conclusion is that there is not a problem to use Xamarin.Forms in this application as there are workarounds for the native API pitfalls and the application does not rely on advanced graphics for making it function properly. These pitfalls could however be a problem in bigger projects with more custom GUI elements and more extensive native functionality. The code reuse that Xamarin.Forms allows, coupled with the fact that there is a simple GUI, makes it the best decision to use within this project.

2.3.4 MVVM software architecture

The software architectural pattern used for developing the application in Xamarin.Forms, is the MVVM design pattern [37], [38]. MVVM stands for Model-View-ViewModel and this design pattern splits the code for the user interface into three different parts, Model, View and ViewModel (See figure 3):

- Model classes which represents data coming from services or databases.

- View classes which corresponds to the visual representation of the Model data.

- ViewModel which can be viewed as a glue between the View and the Model. The ViewModel wraps data from the Model for being presented and modified in the View and controls the View’s interaction with other parts of the application.
The MVVM design pattern is the software architectural pattern that is recommended by Microsoft (the owners of Xamarin) and it provides a good way of keeping business-logic and the user interface decoupled.

### 2.4 Hardware

The application needs to be able to connect with external BLE devices. This is done by acquiring different BLE sensors from different manufacturers to be able to make sure that the application can in fact connect to BLE devices, regardless of brand. There exists a wide range of BLE sensor devices on the market today and the prices vary from $10 USD up to $50 USD per sensor. As this project is funded by CGI it is the budget set by them which governs what sensors could be used for this project. When researching which sensors that can be bought within the budget there are not many devices left to choose from. This means the cheapest possible sensors are used for this project. Other than the bought sensors, there are two Arduino cards borrowed from the thesis supervisor, which can be programmed to function as BLE sensors in this project. The following list of devices has been acquired for this research project (See figure 4):

- **(2) Arduino Genuino 101 cards.** Microcontrollers that contains sensors and a BLE chip which means they can be programmed to send data in the development phase [40].
- **(3) Joyway T-Sensors,** able to measure temperature, altitude and air pressure
- **(3) Joyway iBeacon accelerometer sensors**
The BLE devices by Joyway are driven by clock batteries and runs continuously until the batteries are depleted. The Arduino cards are not battery driven and need an external power source to run. For mobility purposes a charged power bank is connected to them to serve as a power source. When the BLE devices are powered and running they are advertising their existence to other Bluetooth devices such as smartphones or tablets.

2.5 BLE plugin library

The application that is developed during this thesis uses the BLE plugin library to be able to scan for nearby devices [29]. When the BLE devices are identified by the application it can then connect to them and start receiving the sensor data which is advertised wirelessly from the BLE devices. The application collects the data and processes it in order to be able to send it forward to the cloud storage service.

2.6 Azure cloud storage

The cloud storage provider chosen for this project is Microsoft Azure [15], partly because it is a preference from CGI, but also because it is well integrated into the Visual Studio development software that is used for creating the Xamarin.Forms application.

2.7 System requirement analysis

An important aspect of the research is making sure that the application meets the expectations in terms of receiving the sensor data from the BLE devices and sending it to the cloud, as well as how the data will be sorted in the cloud. A system requirement analysis is therefore done in collaboration with the contact at CGI to gather information about what functionality needs to be implemented in the application. The requirement analysis in this thesis is based on a simplified version.
of the requirements engineering process stated in the book written by Tsui et al (See figure 5).

(Figure 5. The requirements engineering process [41])

The technical requirements discovered from the performed requirement analysis are the following:

**R1.** Search for BLE devices that are within range.

**R2.** Connect with BLE devices that are within range.

**R3.** Receive data from connected BLE devices.

**R4.** Choose dates and time intervals for when the application should receive data from the connected BLE devices.

**R5.** Send the gathered data to the cloud storage service.

**R6.** Store the data in the cloud appropriately.

**R7.** Application runs on multiple mobile platforms such as Android and iOS.

### 2.8 Evaluation of the IT artefact

During the software development phase of this thesis there is unit tests continuously performed on the different components of the application to make sure that they are working properly. When the development phase is completed, a system evaluation is performed where the technical requirements of the application are compared with the actual performance of the IT artefact. This is done to be able to conclude that the performance of the IT artefact meets the system requirements stated earlier, thus providing proof of concept.
3. Solution

This section presents the implemented IT artefact and the different software components that it is made from. A brief overview of the system as a whole is presented following a more in-depth presentation of the components that make up the IT artefact.

3.1 The system of the IT artefact

As seen in figure 6, the system is made up from several different hardware and software components. There are BLE sensor devices (T-sensor, Accelerometer, Arduino) which transmits collected sensor data to the Xamarin.Forms smartphone application, which then forwards the data to the cloud storage. The sensors that are activated by the user are saved in a local database on the device.

(Figure 6. Visualization of the system)

3.2 Xamarin.Forms application

The central part of the system is the Xamarin.Forms smartphone application which is making it possible for an application developer to create an application for both the Android and iOS platforms using almost 100 % of the same C# code. The integrated development environment (IDE) used for creating the application is Visual Studio from Microsoft which is the standard development environment for Xamarin.

The application is made up from different components that has different functionality according to the MVVM software architectural pattern explained in the
Visible to the user are the different Views that are written in Extensible Application Markup Language (XAML) which is a tag-based language similar to XML and HTML. The Views written in XAML are used in Xamarin to define the user interface and bind the interface components, such as buttons and labels, to data and event callbacks. The logic of the Views is not placed in the Views itself but instead defined in the ViewModels which are bound by context to the Views. This means that the user interface itself is decoupled from the business logic of the application.

### 3.2.1 Data models

There are two different data models used in this application, a sensor model (see figure 7). The sensor model, which consists of properties holding information about a sensor, such as name, UUID, sensor data, date and time for when the application should be collecting data from it, and a boolean value stating if it is activated or not. The other model class is holding the interval time and dates for a sensor and it has a composite relationship with the sensor class.

![Data models diagram](Figure 7. The data models)

### 3.2.2 User interface

As seen in figure 8, the user interface is deliberately stripped down to keep it simple, with only the most essential functionality needed. The main page of the application holds some extra functionality in form of a switch for activating location tracking which can be sent to the cloud storage. This functionality is outside the scope for this thesis and was added as a bonus feature due to a request from CGI. The interface also shows the current location coordinates of the user and how long it has been active. This feature was added later in the development as a suggestion from CGI. The other components of the main page are the two buttons which are navigating the user to the other two pages of the application, where the user can activate sensors or view currently active sensors.
(Figure 8. The navigation of the user interface shown on a iOS device)
The Add / Remove sensors page is the page where the user can choose from a list of nearby BLE devices that are within range of connecting to the application. The devices that are of interest are then activated by switching the corresponding switch-buttons. A button called “Select all” has been added which lets the user activate or deactivate all of the sensors in one click. When the user clicks on the switch, a modal window is shown where the user can select the preferred date and time intervals for how long the sensor should remain active and how often data should be collected. CGI wanted to be able to read the sensor data on a per-minute basis, but the application could be changed to read much more frequently if that’s necessary. The limitations are in the physical Bluetooth devices and the Bluetooth communication protocol. The date range for which the application is collecting data from the BLE devices can be set with simple date-pickers. The BLE devices are continuously sending data, which can lead to unnecessary quantities of data collected, therefore an interval can be set with a slider which sets how often the application is supposed to pick up information from the sensors. When a sensor is activated, two things happen. The application starts receiving data from the sensor and the sensor is saved in a local database, which means that the application remembers what sensors are active if the application is shut down and later started again. When a sensor is deactivated, it is removed from the local database and the application stops receiving data from it. The last navigation page is showing the current activated sensors to the user in form of a list with the sensors and some information related to it, such as its name, UUID and the date and time intervals.

3.3 NuGet package manager

When developing the application in Xamarin.Forms there is a need for adding extra functionality from both third-party libraries and additional libraries from Microsoft that is not installed from the beginning in Visual Studio. Microsoft has simplified the process of adding these extra libraries in form of the NuGet Package Manager, which is an open-source package manager for the Microsoft development platform [42]. The NuGet Package Manager is installed as part of the Visual Studio Environment and automates the process of installing and updating libraries to be able to use them in Visual Studio. When a library has been installed, its functionality is simply accessed by adding the “using” keyword in the C# file that needs the library functionality.

3.4 Entity framework core

As mentioned earlier, the active sensors are stored in a local database on the smartphones’ internal memory. The database is created using the Entity Framework (EF) which is installed as a library via the NuGet Package Manager. EF is an open source object-relational-mapping software (ORM), developed by Microsoft [43]. In this application EF is used to create and make changes to an SQLite database. SQLite is a relational database management system which is common to use for storing data locally in mobile applications. The SQLite database is created locally in
the mobile device and makes it possible to store and modify data related to the application using Create, Read, Update and Delete operations (CRUD). In order to create and communicate with an SQLite database there is usually a need to write Standard Query Language (SQL) statements, which is the language used for managing SQLite databases. In this application however, EF is used which simplifies the creation of the database and communication with CRUD operations without writing any SQL code. The need for the local SQLite database is crucial for this project so that the application can remember which sensors are activated since the last session, so the user won’t have to reactivate them every time the application is restarted. The date and interval information for each sensor is also saved in the local database. Other data that is stored locally is the sensor data that has not yet been sent to the cloud storage. This is done so that no sensor data is lost, for example if the smartphone is offline or otherwise unable to upload the data to the cloud storage. When the sensor data is successfully uploaded to the cloud storage it then automatically erases it from the local database, preventing unnecessary quantities of data cluttering up the memory of the smartphone.

3.5 App permissions

In order for the application to gain access to specific hardware and software capabilities of the mobile device, such as Bluetooth, device location and storing data locally, the user needs to allow the application’s usage of these features. The ability for the user to choose whether to grant access or to deny an application’s access to these features is there for protecting the privacy of the user. The process of gaining access to application permissions is implemented differently on both Android and iOS. This means that platform specific code needs to be implemented separately for this application to function properly. An employee and Xamarin evangelist at Xamarin named James Montemagno has released a library for solving this specific problem called Plugins.Permissions which is installed from the NuGet Package Manager [44]. The library makes it easy with a few lines of code, to allow the user to decide whether to let the application use the needed features or not. In this application the permissions that are needed from the user are the following:

- Location of the device
- Bluetooth
- Local storage
- Ability for application processes to run in the background

3.6 BLE plugin

Accessing the native Bluetooth functionality of both the Android and iOS operating systems is necessary to make it possible for the application to communicate with the external BLE devices. This functionality is as mentioned earlier, not possible with Xamarin.Forms out of the box. A third-party library must be installed to be able use this functionality without writing platform specific code for both Android and iOS.
The library chosen for the application is the BLE plugin library which has all the functionality needed for communicating with BLE devices [29]. The BLE plugin library is installed to the application project via the NuGet Package Manager in Visual Studio. When using Bluetooth in an application, permission must be granted from the user. When permission is granted from the user, the application can then scan the environment for Bluetooth devices to connect to. This is done by calling the Scan-function in the BLE plugin library (see figure 9) which then receives the devices as a list of objects which are displayed to the user in the user interface.

```csharp
public void RefreshSensors()
{
    var scanner = CrossBleAdapter.Current.Scan().Subscribe(scanResult =>
    {
        ObservableCollection<Sensor> sensorList = selectSensorsViewModel.SensorList;
        foreach (Sensor s in sensorList)
        {
            if (s.uuid.Equals(scanResult.Device.Uuid.ToString()))
                return;
            if (s.string.IsNullOrEmpty(scanResult.Device.Name))
                return;
        }
        byte[] serviceData = new byte[scanResult.AdvertisementData.ServiceData.Count];
        if (scanResult.AdvertisementData.ServiceData.Count > 0)
        {
            for (int i = 0; i < scanResult.AdvertisementData.ServiceData.Count; i++)
                serviceData[i] = scanResult.AdvertisementData.ServiceData[i][1];
        }
        Sensor sensor = new Sensor(scanResult.Device.Name, scanResult.Device.Uuid.ToString(), false, null, serviceData);
        selectSensorsViewModel.SensorList.Add(sensor);
        BLEDevices.Add(scanResult.Device);
    });
}
```

(Figure 9. The RefreshSensors method)

The BLE devices are identified by the application via their specific UUID (Universal Unique Identifier) [45]. The application is then able to receive data from the devices via Generic Attribute Profile (GATT), which defines the way BLE devices transfer data back and forth using Services and Characteristics [30]. GATT uses a generic data protocol called Attribute Protocol (ATT), which store Services and Characteristics [30]. The mobile device application and the BLE devices has a Client Server relationship where the application requests a service from the BLE device, which then sends back the requested Characteristic value (see figure 10).

(Figure 10. GATT Client and Server example [30])
3.7 Azure cloud storage

When a sensor is activated via the user interface the application starts receiving messages from the sensor which is then stored in the local database. The collected data is then sent to the Microsoft Azure cloud storage with regular intervals as Json messages (See figure 11).

```csharp
public class UploadSensorDataService
{
    private static DeviceClient _deviceClient;
    private readonly static string _connectionString = "HostName=SensorGatewayHub.azure-devices.net;DeviceId=SensorDeviceId;SharedAccessKey=sharedDeviceKey;"
    public UploadSensorDataService()
    {
        _deviceClient = DeviceClient.CreateFromConnectionString(_connectionString, TransportType.Mqtt);
    }
    public static async void SendDeviceToCloudMessagesAsync(TSensorData sensorData)
    {
        while (true)
        {
            // Create JSON message
            var telemetryDataPoint = new
            {
                temperature = sensorData.Temperature,
                pressure = sensorData.Pressure,
                altitude = sensorData.Altitude,
                battery = sensorData.Battery
            };
            var messageString = JsonConvert.SerializeObject(telemetryDataPoint);
            var message = new Message(Encoding.ASCII.GetBytes(messageString));
            // Send the telemetry message
            await _deviceClient.SendEventAsync(message);
            Console.WriteLine("(0) > Sending message: (1)", DateTime.Now, messageString);
        }
    }
}
```

(Figure 11. The class for uploading sensor data to the Azure cloud storage)

The Azure functionality of the application is obtained via Microsoft’s Azure library for Xamarin which is installed via the NuGet package manager. All that is needed for connecting the application to Azure, is a connection string which can be obtained when the Azure cloud storage has been configured. The data can then be sent via a method call to an IoT-hub that is set up in Azure [46]. The job of the IoT-hub in this case is to redirect the incoming data to the correct Azure-service for that data. In this case the data is sent to a Stream Analytics service [47]. Stream Analytics is the service that reads the incoming data and converts it to readable data and then sends that data to Power BI [48]. Power BI enables visualization of the data in real time through the web interface or the desktop app. This is a good enough implementation to show that sending the data works well.
4. Results

This section presents the results of the conducted research of this thesis. The first subsection involves a discussion about the most important findings that were gathered from developing the mobile application. Then follows a presentation of the evaluation of the IT artefact and how the system managed to meet the requirements presented earlier in the method section.

4.1 Artefact evaluation

The results from testing the developed IT artefact shows that the mobile application is functioning as expected on both Android and iOS devices. The application is able to search for and connect to BLE devices that are in range, but as to be explained further, certain conditions has to be met from the BLE devices for the application to be able to connect to them and receive the data properly.

4.2 Bluetooth connectivity

Connecting to and receiving data from the BLE devices is possible, as long as the BLE devices themselves are programmed to follow the Bluetooth standard for data communication which is described earlier in the solution section. When developing and testing the mobile application, there were some difficulties when connecting and receiving data from the Joyway sensors. One of the difficulties was that the application was unable to connect with the Joyway sensors. The sensors were showing up in the list of nearby devices, but it was not possible to establish a connection with the sensors and the mobile application.

4.3 Bluetooth standardisation

After trying many possible attempts at solutions to connecting the application with the Joyway sensors, the conclusion was drawn that the Joyway sensors were not following the standard for data communication between BLE devices. After some correspondence with the Joyway company, the source code for Joyway's own mobile application made for their T-sensor was acquired for analysis. The source code provided valuable insight which showed that the sensors were indeed sending data in different way than it is intended according to the Bluetooth standard. When analysing the acquired source code it was shown that the sensors were sending data continuously, without a need for an established Bluetooth connection. However, the sensors did not transmit the collected data in separate GATT services as described in the solution section, instead they were sending its data in a service which is supposed to hold data about the time and date. Instead of customizing their own
services for each sensor value (temperature, air pressure and elevation) as suggested by the Bluetooth standard, they had crammed all the sensor values in the time service. This meant that the application had to be tweaked in order to receive data from these particular devices.

4.4 Customization to receive data from Joyway sensors

As mentioned, the application had to be customized to be able to receive data from the sensors manufactured by Joyway. The following subsection describes how the data was extracted from the sensors. Firstly, the application had to be told which bytes were holding the different sensor values (see figure 12).

```csharp
public static TSensorData ParseSensorData(byte[] bytes)
{
    if (bytes.Count() <= 13 || bytes[1] != 22)
    {
        return null;
    }

    float temperature = ParseSensorData(bytes[2], bytes[3], bytes[4]) / 100.0f;
    float pressure = ParseSensorData(bytes[5], bytes[6], bytes[7]) / 100.0f;
    float altitude = ParseSensorData(bytes[8], bytes[9], bytes[10]) / 100.0f;
    int battery = bytes[11] & 0xFF;

    return new TSensorData(temperature, pressure, altitude, battery);
}
```

(Figure 12. The method for parsing the array of bytes received by the Joyway T-sensor)
The application also had to perform calculations from the parsed bytes in order to obtain the correct sensor values as shown in figure 13.

```csharp
private static int ParseSensorData(byte b1, byte b2, byte b3)
{
    if((b1 & 0x80) == 0)
    {
        return (b1 & 0x80) << 16 | (b2 & 0xFF) << 8 | b3 & 0xFF;
    }

    int i = b1 << 16 | b2 << 8 | b3;

    List<int32> bits = new List<int>();

    for(int j = 0; j < 20; j++)
    {
        bits.Add((0x1 & i >> j) == 0 ? 1 : 0);
    }

    i = 0;

    for(int j = 0; j < 20; j++)
    {
        i = (int)(i + Math.Pow(2.0D, j) * bits[j]);
        i++;  
        i = 0 - i;
    }
    return i;
}
```

(Figure 13. The method for obtaining the sensor values from the bytes parsed from the T-sensor time service)

4.5 Connectivity and receiving data from Arduino cards

The Arduino cards were connecting to the smartphone application as intended and the application was able to collect the data from them as soon as a connection had been established. As the Arduino cards were programmed correctly from the beginning in accordance to the Bluetooth standard, this gives indication that the application can fulfill its purpose as a general-purpose application, as long as the BLE devices are programmed correctly.

4.6 Uploading data to the cloud

The application managed to upload the collected sensor data from the BLE devices to the Azure cloud storage. Even though the application was sending the sensor messages according to the chosen intervals, it was not always received straight away
in the cloud. However, this was not problematic as all the messages sent from the application arrived sooner or later and no messages were lost.

4.7 Requirement testing

The requirement analysis performed in the early stages of this thesis work, presented the following requirements that needed to be met, for the application to fulfill its goal as a cross-platform, general purpose sensor application:

**R1.** Search for BLE devices that are within range.

**R2.** Connect with BLE devices that are within range.

**R3.** Receive data from connected BLE devices.

**R4.** Choose dates and time intervals for when the application should receive data from the connected BLE devices.

**R5.** Send the gathered data to the cloud storage service.

**R6.** Store the data in the cloud appropriately.

**R7.** Application runs on multiple mobile platforms such as Android and iOS.

The following tables (see tables 1.1 - 1.7) gives a visualization of how the final IT artefact responds to the requirements stated earlier. Preceding each table there is a description about each requirement test, and how it affects the applications’ functionality and performance.
4.7.1 Requirement 1: Search for BLE devices

R1. (see table 1.1) The first requirement was successfully met on all the devices tested and the application had no difficulty in discovering all BLE devices as well as any other nearby Bluetooth devices.

<table>
<thead>
<tr>
<th>BLE devices</th>
<th>Requirement met</th>
<th>Requirement not met</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arduino Genuino 101 (device 1)</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>Arduino Genuino 101 (device 2)</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>Joyway T-sensor (device 3)</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>Joyway T-sensor (device 4)</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>Joyway T-sensor (device 5)</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>Joyway Accelerometer (device 6)</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>Joyway Accelerometer (device 7)</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>Joyway Accelerometer (device 8)</td>
<td>○</td>
<td></td>
</tr>
</tbody>
</table>

4.7.2 Requirement 2: Connect with BLE devices

R2. (see table 1.2) When connecting to BLE devices the Arduino cards was able to connect without problems. The devices from Joyway, however was unable to connect with the application. This however is not a problem when it comes to receiving data from the Joyway devices, as they use a different approach in how the communication is implemented. The Joyway sensors are transmitting sensor data without a connection needed and any nearby device can therefore receive it.

<table>
<thead>
<tr>
<th>BLE devices</th>
<th>Requirement met</th>
<th>Requirement not met</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arduino Genuino 101 (device 1)</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>Arduino Genuino 101 (device 2)</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>Joyway T-sensor (device 3)</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>Joyway T-sensor (device 4)</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>Joyway T-sensor (device 5)</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>Joyway Accelerometer (device 6)</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>Joyway Accelerometer (device 7)</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>Joyway Accelerometer (device 8)</td>
<td>○</td>
<td></td>
</tr>
</tbody>
</table>
4.7.3 Requirement 3: Receive data from BLE devices

R3. (see table 1.3) When it comes to the ability to receive sensor data from the BLE devices the application was able to collect data from both the Arduino devices and the Joyway T-sensors. The application was not able to receive any data from the Joyway accelerometers.

<table>
<thead>
<tr>
<th>BLE devices</th>
<th>Requirement met</th>
<th>Requirement not met</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arduino Genuino 101 (device 1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arduino Genuino 101 (device 2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Joyway T-sensor (device 3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Joyway T-sensor (device 4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Joyway T-sensor (device 5)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Joyway Accelerometer (device 6)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Joyway Accelerometer (device 7)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Joyway Accelerometer (device 8)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.7.4 Requirement 4: Date and time interval functionality

R4. (see table 1.4) The application managed to fulfill its requirement to select a start date and end date as well as a time interval for when the sensor data collection should occur on all tested mobile devices.

<table>
<thead>
<tr>
<th>Mobile devices</th>
<th>Requirement met</th>
<th>Requirement not met</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple iPhone 6s (iOS 12.0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apple iPhone X emulator (iOS 12.1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oneplus X (Android 6.0.1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Huawei P20 (Android 8.0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Android emulator (Android 8.0)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4.7.5 Requirement 5: Upload data to cloud storage

**R5.** (see table 1.5) Requirement no. 5 was successfully met as the application was able to upload the collected data to the Microsoft Azure cloud storage on all tested physical devices.

<table>
<thead>
<tr>
<th>Mobile devices</th>
<th>Requirement met</th>
<th>Requirement not met</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple iPhone 6s (iOS 12.0)</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>Apple iPhone X emulator (iOS 12.1)</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>Oneplus X (Android 6.0.1)</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>Huawei P20 (Android 9.0)</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>Android emulator (Android 8.0)</td>
<td>○</td>
<td></td>
</tr>
</tbody>
</table>

4.7.6 Requirement 6: Store data properly in the cloud

**R6.** (see table 1.6) The application was able to upload the collected sensor data to the Azure cloud storage which proves that requirement no. 6 was successfully met.

<table>
<thead>
<tr>
<th>Cloud storage service</th>
<th>Requirement met</th>
<th>Requirement not met</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Azure</td>
<td>○</td>
<td></td>
</tr>
</tbody>
</table>

4.7.7 Requirement 7: Cross-platform usage

**R7.** (see table 1.7) As shown in the table, the application runs on various mobile operating systems such as Android versions 6.0.1, 8.0 and 9.0 as well as iOS versions 12.0 and 12.1. The Bluetooth functionality, however, is not supported on the emulated systems.

<table>
<thead>
<tr>
<th>Mobile devices</th>
<th>Requirement met</th>
<th>Requirement not met</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple iPhone 6s (iOS 12.0)</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>Apple iPhone X emulator (iOS 12.1)</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>Oneplus X (Android 6.0.1)</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>Huawei P20 (Android 9.0)</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>Android emulator (Android 8.0)</td>
<td>○</td>
<td></td>
</tr>
</tbody>
</table>
4.8 Summary of results

The research involved in the development of the general-purpose sensor application has led to interesting new knowledge in form of information about the difficulties in developing this type of mobile applications. The results show that manufacturers may not be following the standards for Bluetooth data communication, thus making it hard to write generalized methods for retrieving data from sensors of any type or manufacturer.

5. Analysis and discussion

This section of the thesis presents a discussion and analysis of the results gathered during the thesis work. The discussion focuses on how the gathered results relates to previous existing research, how the results are used to answer the research question and presents the contributions of this research.

5.1 Comparing results to previous research

The previous design and creation research that has been conducted in this field is based on using specific sets of sensors which are chosen beforehand by the researchers for solving specific problems. This means that the software created to communicate with these sensors is tailored to work only with these particular sensors. The main difference between this research project and earlier research, is that this thesis focuses on creating software that is able to communicate with any sensor, regardless of manufacturer or type which is a different kind of angle used for the creation of this type of IT-artefacts.

The results of this research could be used for reference and guidelines by other researchers that has a need for using sensors in collecting different types of data for their research. The purposes for this type of software which can gather sensor data from any kind of sensor are plentiful. For example it can be used in healthcare research for gathering information about a patient’s health, in sports research for perfecting an athletes performing abilities, or for other usages such as measuring temperature, humidity or any type of sensor data that could be of interest for research purposes. This means that the scientific contribution of this thesis is more generalized in that it provides a framework for creating a mobile application for connecting to any BLE sensor, regardless of manufacturer or type and not contributions in form of gathered results from data collected from specific sets of sensors.
5.2 Results of conducting the literature review

The focus of this thesis has been to determine if it is possible to create a cross-platform mobile application for connecting to BLE sensors of any type and manufacturer and to present the process of implementing such an application.

Gathering and analysing information from the existing literature gave the answer that it is indeed possible to create such an application with the help of current software development platforms and third-party libraries. The immense amount of information resources both digital and physical, gives software developers the information needed for creating an application of this type. Useful information amongst others, is the official documentation from the Xamarin website and the official Bluetooth standard, describing how to set up connection of BLE devices and how data communication should be implemented properly.

One of the potential criticisms that could be raised with the conducted literature review is that there could have been more peer-reviewed sources in comparison to the sources gathered from the internet. This could lead to criticism about the scientific trustworthiness. It has however been a well-considered decision to include these internet sources as they provide necessary information that could not be found elsewhere. This information should not be seen as replacements for the peer reviewed literature but rather as complementary sources for giving extra information to the reader about areas such as programming languages, software frameworks and third-party libraries. As these internet sources are often referring to official documentation or reports that are conducted by well-established corporations, the decision was made to include these sources even though it could become a point of critique from the scientific community.

Overall the coverage of literature on the interesting areas concerning this subject was satisfiable. However, none of the reviewed literature seemed to address the current problems that exist with standardisation of BLE devices from the manufacturers.

5.3 Evaluation of the IT artefact

The application that was built had to be tested continuously during the software development process to measure that the functionality had been implemented properly. As shown in the result section of this thesis, the application had no problems to discover BLE devices in the nearby surroundings. However, when it came to connecting to the BLE devices, the application failed to establish a connection with several of the devices. The Arduino cards which had been programmed were able to connect to the mobile device, which supported the theory that the connection functionality had been implemented properly within the application. Even though the Arduino cards were able to connect, the application was unable to establish connections with the devices made from the manufacturer Joyway. This led to suspicions about how the Bluetooth functionality software was implemented in these devices. After examining the source code for Joyway's own
Android application it was concluded that they did not follow the standard protocol for data communication. These findings suggest that the reality of developing an application that could connect to any type of sensor does not match the perfect world scenarios that the literature provides. In the real world, the functionality of this application is relying on factors that cannot be controlled such as the programming of the hardware from the manufacturers.

Evaluating the other requirements such as uploading data to the cloud and storing data, were straightforward and gave the results that were expected. One could argue that the evaluation of the developed IT artefact could be tested more extensively with a wider range of BLE devices and more specified requirements that could be tested, however the results should be satisfiable enough to provide proof of concept, given the limited time and resources set aside for this project.

5.4 Challenges in Bluetooth standardisation

The results from the design and creation research is giving indications that not all manufacturers are following the standard for how Bluetooth communication should be implemented. This is troublesome when developing an application for collecting data from any type of sensor regardless of manufacturer and one of the more interesting findings of this research. The fact that the Joyway sensors are not following the Bluetooth standard for data communication is problematic for the generalization properties of the mobile application created in this research. It means that it is not possible to receive data from BLE devices regardless of manufacturer if the manufacturers of BLE devices or not following the Bluetooth standard. The fact that the sensors manufactured by Joyway did not connect and were unable to transfer data the way it is intended, gives a clear indication that there is a great ongoing challenge, in terms of standardising BLE sensors manufactured by different manufacturers.

If this research project had a bigger budget, it would be possible to test the mobile application on a wider variety of devices from different manufacturers to evaluate to what extent other BLE device manufacturers are following the Bluetooth standard. That would be necessary to determine if this problem is widespread amongst manufacturers of BLE devices or limited to the devices manufactured by Joyway.

5.5 Summary

As the results show, it is possible to create an application for sensors which can be used for multiple purposes, but it has some obvious negative aspects given the state of the BLE devices that exist today. The application that was created during this research performs well in a perfect world scenario, where all software developers and hardware manufacturers follow the standards that are set in place by entities such as the Bluetooth Special Interest Group (BSIG). Unfortunately, the results give indications that there is much work to be done in standardising the BLE devices so
that they all function the way that the Bluetooth standard suggests. Without the standardisation of BLE devices, an application like the one that has been developed during this research is not as useful as it could be.

6. Conclusions and further research

This research set out to answer if it is possible to develop an IT artefact in form of a multi-purpose, cross-platform mobile application for connecting with BLE sensor devices, gather sensor data from the sensors and uploading it to a cloud storage service. To answer this question, a literature study was conducted to gather information about what has been done in this field and to learn more about the different software frameworks that can be used to create an application with this type of functionality. Following the literature study, an IT artefact was produced with design and creation using established software development methods. By testing the IT artefact it was shown that an application like this can be created, but as of today it is not able to connect to all types of devices as some manufacturers of BLE devices do not follow the data communication standard stated by the official Bluetooth documentation.

6.1 Scientific contribution

The scientific contribution from this research is the described process of creating this IT artefact, which contains useful information for researchers and other readers on what to consider in developing a cross-platform, multi-purpose, sensor application. Other interesting findings that contribute to the scientific community is that this research shows that there are clear indications that there is an ongoing challenge in terms of standardising BLE sensors manufactured by different manufacturers and that there is more research needed in this area.

6.2 Future research

Interesting future research in this field would be to conduct a study which compares many different BLE devices by different manufacturers to see how the manufacturers has chosen to implement their data communication protocols and how many that differ from the standard and in what ways. There could be interviews conducted with manufacturers to be able to better understand their choices and intentions. How can manufacturers around the world be convinced to follow the standards and ultimately to make development of similar applications easier for developers. This is important for software developers, end users of this growing technology and to make research easier and more efficient in the future.
References


