Bachelor's thesis

Information Technology

Software Business

2014

Marko Patanen

CREATING ANDROID APPLICATION USING BLE SENSOR

– A knee rehabilitation monitoring system

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Software Business

2014| 37

Tiina Ferm

Marko Patanen

CREATING ANDROID APPLICATION USING BLE SENSOR

This thesis introduces and discusses Android development with a special focus on the use of external sensors as part of an application.

This thesis first introduces Android application development in general, going through the most common components in Android applications and then takes a closer inspection on development with an external sensor and the components supporting its use.

The aim of this thesis was to create an application with working low energy sensor connection and data storage. The data transfer between the application and the Android device is carried out via Bluetooth Low Energy (BLE) technology, therefore, an introduction to this technology is included in this thesis

The sensor used in this thesis is TI SensorTag and it is used to create an application for a knee rehabilitation monitoring system. The IDE of choice is Eclipse IDE with Android developer tools plugin. The application is targeted at the Android 4.4 operating system.

The major obstacle in the development process was lack of proper sources and guides which increased the development time significantly. Minor challenges originated from the integrated development environment which reportedly has some issues with Android development.

The application was created to a point where all the functionalities for using the sensor and saving data were completed. The implementation of the user interface was left for future development. However, a simple user interface was created for testing purposes. The application in itself offers little to none monetary value. In the future, a number of components will be extracted and shared online for future reference and for the benefit of other developers. Adding more functionalities and finalizing the user interface is left for future evaluation.

KEYWORDS:

Android, Eclipse, BLE, Bluetooth Low Energy, mobile application, Sensor Tag, OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikka| Ohjelmistoliiketoiminta

2014 | 37

Tiina Ferm

Marko Patanen

BLE-SENSORIA HYÖDYNTÄVÄN ANDROID -SOVELLUKSEN LUOMINEN

Työssä tutustuttiin Android -ohjelmistokehitykseen keskittyen erityisesti ulkoisten sensorien käyttöön osana sovellusta. Opinnäytetyössä tarkasteltiin Android-kehitystä ensin yleisellä tasolla. Tarkoituksena oli saada kuva yleisimmistä Android-sovelluskehityksen komponenteista. Tarkempaan tarkasteluun otettiin erityisesti ulkoiset sensorit ja niiden käyttöön liittyvät komponentit.

Opinnäytetyön tavoitteena oli luoda Android – sovellus, joka muodostaa yhteyden low energy -anturiin ja tallentaa anturidataa. Tiedonsiirtoon käytettävä teknologia on Bluetooth Low Energy, joka esitellään osana työtä.

Työssä käytettävä anturi oli TI Sensor Tag, jota käyttäen luotiin polven kuntoutuksen seurantajärjestelmä. Työssä on käytetty Eclipse IDE -ohjelmointiympäristöä, johon on liitetty Android Development Tools -lisäosa.

Suurin haaste työssä oli lähteiden ja oppaiden heikko tarjonta, mikä kasvatti työn tekemiseen kulunutta aikaa merkittävästi. Pienempiä ongelmia toi mukanaan käytetty IDE, jonka Android -kehitystyökaluissa on raportoituja ongelmia.

Sovellukseen luotiin kaikki toiminnallisuudet anturin käyttämiseksi ja tiedon tallentamiseksi. Lopullisen käyttöliittymän luominen jätettiin kuitenkin myöhempään kehitysvaiheeseen. Sovellukseen kuitenkin luotiin yksinkertainen käyttöliittymä testausta varten.

Sovelluksen käytettävyys sellaisenaan on heikko, eikä sillä ole juurikaan rahallista arvoa. Sovelluksesta irrotetaan tulevaisuudessa osia oppaiden luomista varten. Oppaat julkaistaan internetissä, tulevaa käyttöä sekä muita sovelluskehittäjiä silmälläpitäen. Tulevaisuudessa myös arvioidaan tarvetta lisätä toiminnallisuuksia sovellukseen.

ASIASANAT:

Android, Bluetooth Low Energy, BLE, Eclipse, Sensor Tag

Content

Pictures

LIST OF ABBREVIATIONS (OR) SYMBOLS

Abbreviation	Explanation of abbreviation (Source)
IDE	Integrated Development Environment
TI	Texas Instruments
ADT	Android Development Tools
BLE	Bluetooth Low Energy
LE	Low Energy
XML	Extensible Markup Language
GATT	Generic Atribute Profile

1 INTRODUCTION

There are over 30 000 different applications in Health and Fitness category in Google Play -service. The number of health applications and innovation for new health and wellbeing technologies is increasing [1]. According to estimates in 2015 there are 500 million people using mobile healthcare applications. The studies say that almost one in five smartphone users have at least one health application installed [2]. The increased number of smartphone and tablet devices has had a positive effect on sells of mobile health applications.

The aim for this thesis is to have a look at Android -mobile application development and how different sensors can be used along with Android application. For this purpose a mockup application is created. The purpose of the application is first to work as an example and to make it easier for patient or physiotherapist to monitor the rehabilitation of the knee. It is hoped that the application would motivate patients to take active part in rehabilitation process.

The software has been developed for Android -device and it uses TI Sensor Tag in order to demonstrate sensors. Sensors from other manufacturers are also covered briefly. The main point is to create Android –application that connects to sensor, is able to gather data, has a function that uses data, and finally shows user the end result.

2 TECHNOLOGIES FOR DEVELOPMENT

2.1 TI Sensor Tag

TI Sensor Tag is a sensor device developed by Texas Instruments. It is particularly designed as a base for smartphone application and peripheral developers. It utilizes wireless Bluetooth BLE technology. Sensor Tag is a circuit board that has 6 different sensors attached to it. It also has a Bluetooth transmitter to be able to communicate with other devices.

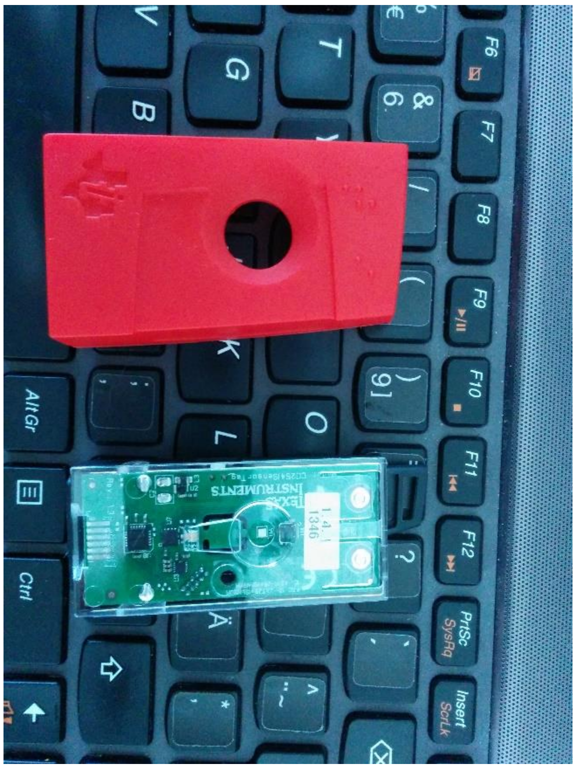


Image 1. Sensor Tag

The sensor found on Sensor Tag are: pressure sensor, temperature sensor, magnetometer, gyroscope, accelerometer and humidity sensor. For this thesis only gyroscope is used (Image 1).

The idea behind Sensor Tag is to be easily approachable and easy-to-use tool for developers. Using Sensor Tag requires no knowledge in electronics or embedded software. It supports updating the firmware wirelessly with smartphone or tablet device. [3]

The source code for example program for Sensor Tag is available online and the compiled program can be downloaded straight from Google Play Store [4].There are also other applications for Windows and iOS devices, however here we concentrate on Android applications.

2.2 Android

Android is an operating system developed by Android Inc. Google bought Android Inc. in 2005. Even before that Android was financially supported by Google.

Google has released the source code for Android under the Apache License 2.0 and therefore developers can modify and distribute it as they wish. However Android devices typically comes with combination of both open and closed source code.

Android is a software-stack which consists of several layers of components. The lowest layer is Linux kernel which acts as a core for the operating system. Application programming interfaces and libraries are laid on top of kernel. The top level consists of applications and supporting application framework. There are set of default applications on Android i.e. Calendar, Maps, Contacts, E-mail, and SMS. The applications for Android are programmed with Java language. [5]

Modified version of Android are developed by Android Open Source Community (AOSP). There are many different versions of modified Android operating system.

In this thesis special significance is put in BLE platform, since it is common and widely used platform for communication with sensors. Support for BLE platform has been included for operating system since Android version 4.3. This sets requirements for the device on which the application runs.

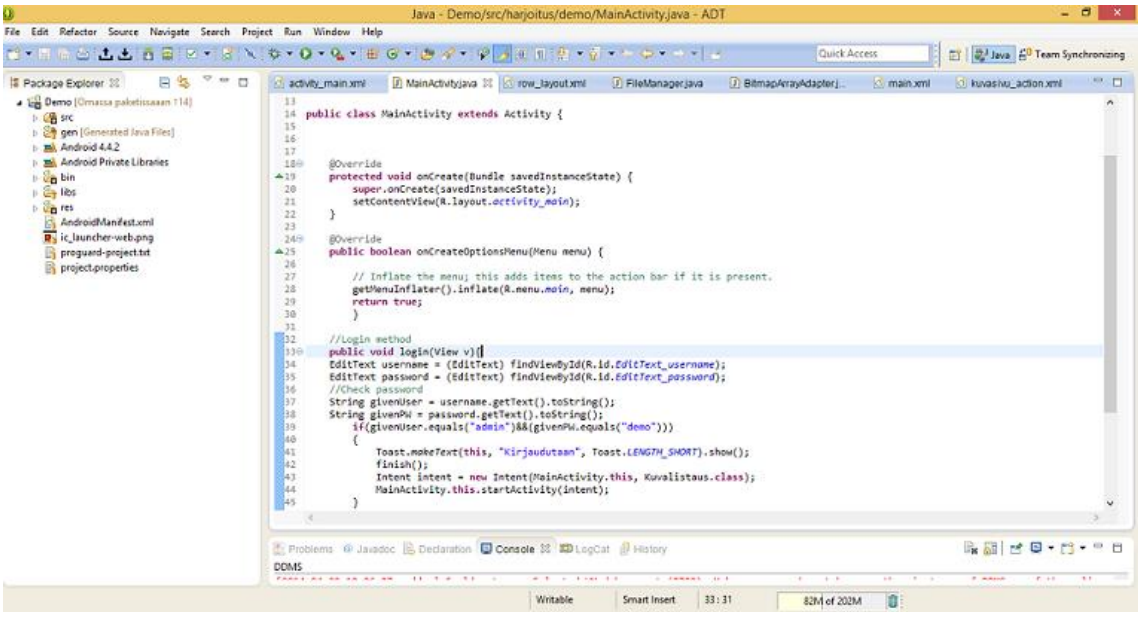
2.3 IDE

2.3.1 Eclipse

Eclipse IDE is used to build the software. Eclipse supports many programming languages, most importantly Java. Since 2001 Eclipse has been published under open source license and it is free software. In 2004 Eclipse Foundation took care of development of Eclipse. Before that Eclipse was developed by Object Technology International, which has been part of IBM from 1996. [6, 7]

Eclipse has been built in a way that it is extensible through plugins. Technically everything in Eclipse are plugins, except the base system. For most parts Java programming language is used to develop Eclipse [6]. The user interface is simple in spite of several installed plugins (Image 2).

The knee rehabilitation monitoring system is built using Android Development Tools which are available as a plugin for Eclipse IDE.

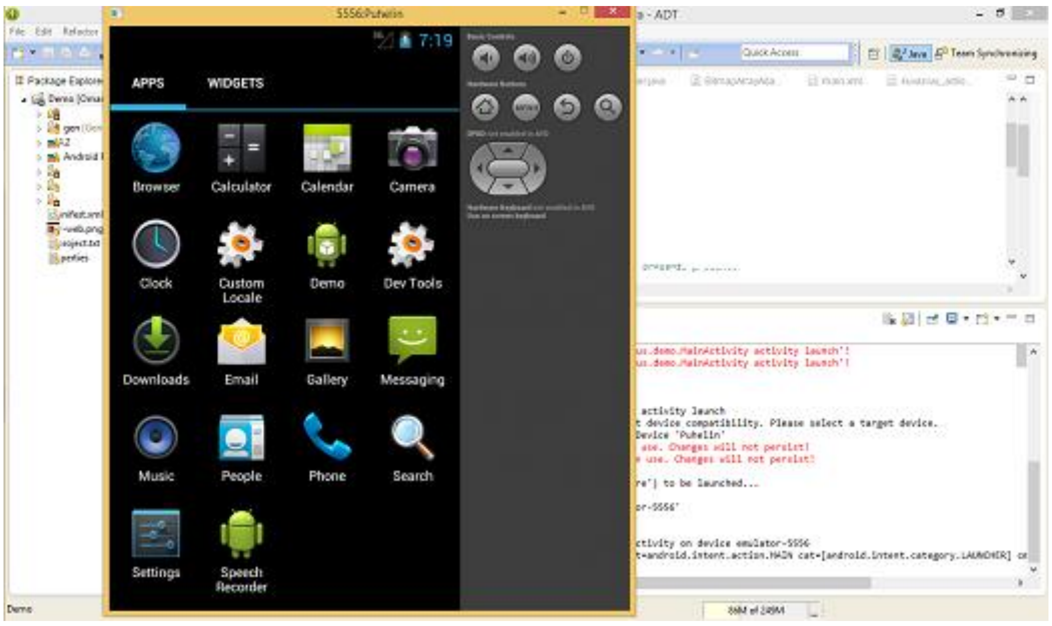


```
public class MainActivity extends Activity {



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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {

        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
        }

    //Login method
    public void login(View v){
    EditText username = (EditText) findViewById(R.id.EditText_username);
    EditText password = (EditText) findViewById(R.id.EditText_password);
    //Check password
    String givenUser = username.getText().toString();
    String givenPW = password.getText().toString();
        if(givenUser.equals("admin")&&(givenPW.equals("demo")))
        {
            Toast.makeText(this, "Kirjaudutaan", Toast.LENGTH_SHORT).show();
            finish();
            Intent intent = new Intent(MainActivity.this, Kuvalistaus.class);
            MainActivity.this.startActivity(intent);
        }
```

Image 2. Screenshot of Eclipse

2.3.1.1 Android Development Tools (ADT)

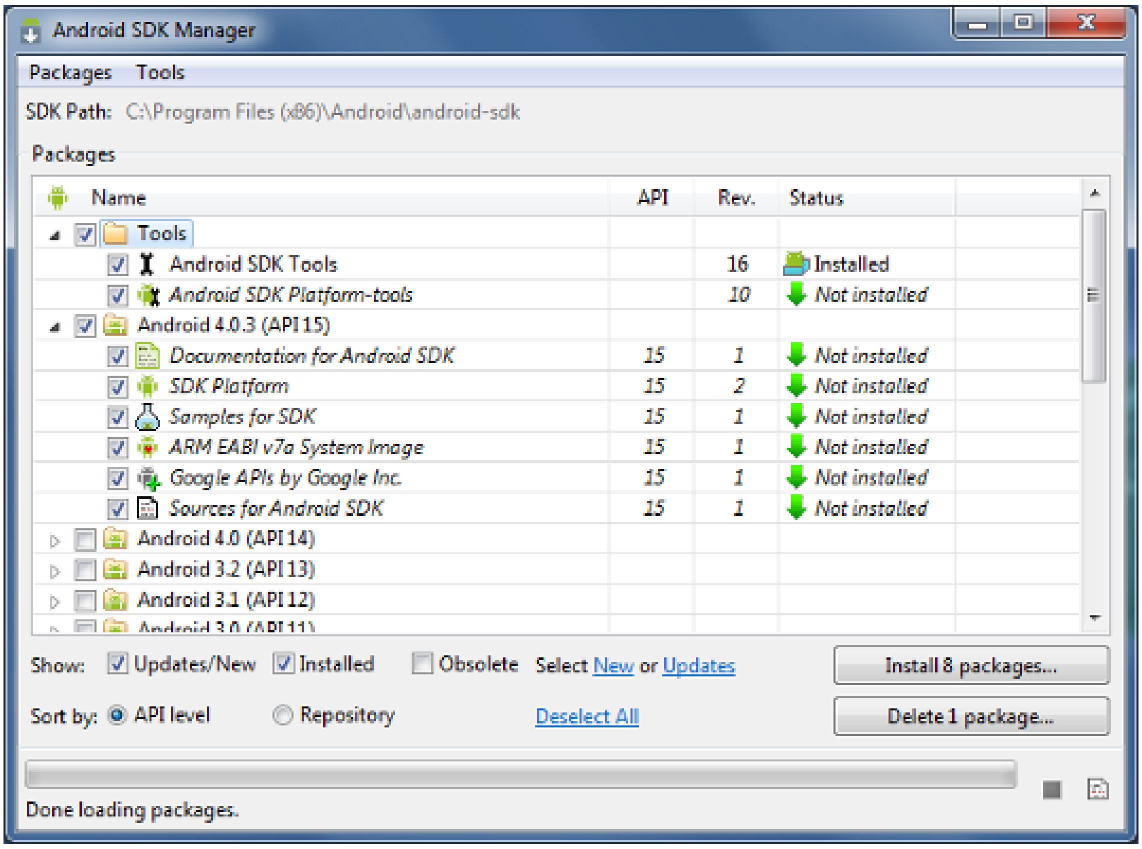
Android Development Tools are available as a plugin packet for Eclipse. It has software that enables development of Android software with Eclipse. It has several pieces of software to ease the process of Android development. For example it offers tools for developing graphical user interface (GUI) using drag-and-drop technique or by modifying XML file. The drag-and-drop technique generates XML -code and it is possible to use both techniques in combination. Furthermore the GUI can be previewed on different screen densities and sizes without launching the emulator.



```
us.demo.MainActivity activity launch'!
us.demo.MainActivity activity launch'!

 activity launch
t device compatibility. Please select a target device.
Device 'Puhelin'
 use. Changes will not persist!
e use. Changes will not persist!

re'] to be launched...

or-5556'

ctivity on device emulator-5556
t=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cm
```

Image 3. Screenshot of emulator

As mentioned it has Android emulator, as seen in image, which allows testing with virtual devices (Image 3). The emulator can emulate all Android devices but the system images needs to be downloaded separately (Image 4). The emulator helps to develop software to support wide base of devices. [8]



Android SDK Manager

Packages Tools

SDK Path: C:\Program Files (x86)\Android\android-sdk

Packages

Name	API	Rev.	Status
✓ Tools			
✓ Android SDK Tools		16	Installed
✓ *Android SDK Platform-tools*		*10*	*Not installed*
✓ Android 4.0.3 (API 15)			
✓ *Documentation for Android SDK*	*15*	*1*	*Not installed*
✓ *SDK Platform*	*15*	*2*	*Not installed*
✓ *Samples for SDK*	*15*	*1*	*Not installed*
✓ *ARM EABI v7a System Image*	*15*	*1*	*Not installed*
✓ *Google APIs by Google Inc.*	*15*	*1*	*Not installed*
✓ *Sources for Android SDK*	*15*	*1*	*Not installed*
Android 4.0 (API 14)			
Android 3.2 (API 13)			
Android 3.1 (API 12)			
Android 3.0 (API 11)			

Show: [✓ Updates/New] [✓ Installed] Obsolete Select New or Updates

Sort by: [✓ API level] Repository Deselect All

Install 8 packages...

Delete 1 package...

Done loading packages.

Image 4. Adding platforms with Android SDK Manager

ADT also has support for physical devices that makes installing software, debugging and testing possible. Physical devices can be plugged in to computer using USB -cable and developed application can be installed straight to device.

Like Eclipse also ADT-plugin is free and published under open-source-license. [8]

2.3.2 Android Studio

Another IDE for Android development is Android Studio, which was introduced in 2013. It is especially designed for Android development, in contrast Eclipse is a multiplatform IDE. It is released by Google and therefore it has integrated function for other Google services, for example Google Cloud Messaging.

Like Eclipses Android Development Tools it has graphical user interface for dragging and dropping layout components. Android Studio also allows previewing the application layout on multiple screens.

The creation of projects are much like in Android Development Tools. It is wizard based and creates all the necessary files for application to run. However it offers few features that are not included on ADT-plugin. For example it allows adding padding for launcher icons before creating project.

Since the tools in Android Studio are mainly the same as in ADT Eclipse was chosen over Android Studio. It was considered that Android Studio did not offer any major advantage over Eclipse and therefore it was better to use a tool that was already familiar. The second reason was that Android Studio is still an early access preview version and all the features are not implemented or are incomplete which increased the risk of problems during development. [9, 10]

2.3.3 Xamarin Studio

Xamarin Studio is an independent IDE that allows creation of iOS and Android. It was released in 2013. Like ADT-plugin and Android Studio it enables user interface design without XML.¨

However Xamarins approach is much different from ADT and Android Studio. Xamarin is designed to build applications for multiple platforms, Windows Phone; iOS and Android, with ease. Xamarin allows programming for Android with C#, instead of Java. Xamarin does have debugging features and syntax check like both IDEs mentioned above. [11]

Xamarin would have given value to project by allowing it to run on iOS and Windows Phones as well. However, since all the tutorials and source material were written in Java, that was the language of choice for the project. In addition Xamarin licensing could become problematic in case the development is taken further. On top of that it does not support Linux operating system which would have represented additional costs. [12]

2.4 Bluetooth

Bluetooth is a wireless communication technology developed by Ericsson in 1994. In the year 1998 Ericsson, Intel, Nokia, Toshiba, and IBM formed Bluetooth Special Interest Group to develop Bluetooth even further. The technology is a combination of both software and hardware [13, 14]. Originally it was developed to work as an alternative for RS-232 cables [13].

Bluetooth uses radio transmissions to send and receive data. The range for Bluetooth can be over 100 meters but the usual range used is around 10 meters. The frequency of Bluetooth is 2,4GHz – 2,485GHz. This frequency is referred to as ISM-frequency. ISM stands for industrial, scientific, medical. In most countries this frequency is unlicensed and available to use. [13]

2.4.1 Bluetooth Low Energy

Bluetooth low energy is a wireless technology for data transfer. It is particularly developed to be energy efficient. BLE enables devices to work for extended periods even when smaller batteries are used. All the major operating systems have a support for BLE. Many devices, such as smartphones and tablets, are able to communicate with BLE devices without the need to modify software or hardware.

The lower power consumption is achieved by using smaller data packets to send data and by sending data for shorter periods. The number of channels used is smaller for BLE than it is for standard Bluetooth. On top of that the architecture has been simplified for BLE. [14, 15]

When BLE technology came to market many sport and wellbeing companies started to use this technology since it is energy efficient and easily connected to smartphones. Nowadays BLE technology has spread into medical, wearables and home automation. [16]

The first devices supporting BLE technology was released by Apple, which brought BLE support as native to its operating system. This expedited the work of developers since they no longer had to think, how to make application to communicate with operating system. For Android devices BLE support came in version 4.3 API Level 18, which was released in July 2013. [17, 18]

3 DEVELOPING ANDROID APPLICATION

3.1 Creating new project

ADT bundle makes creating new android application project easy since it has a wizard for it. The wizard creates and HelloWorld application which allows developer to see what elements are included in very basic Android application.

First the wizard creates a MainActivity.java file. In this file the functions for activity are programmed. At very least the file contains onCreate() method that dictates what layout file is going to be used.

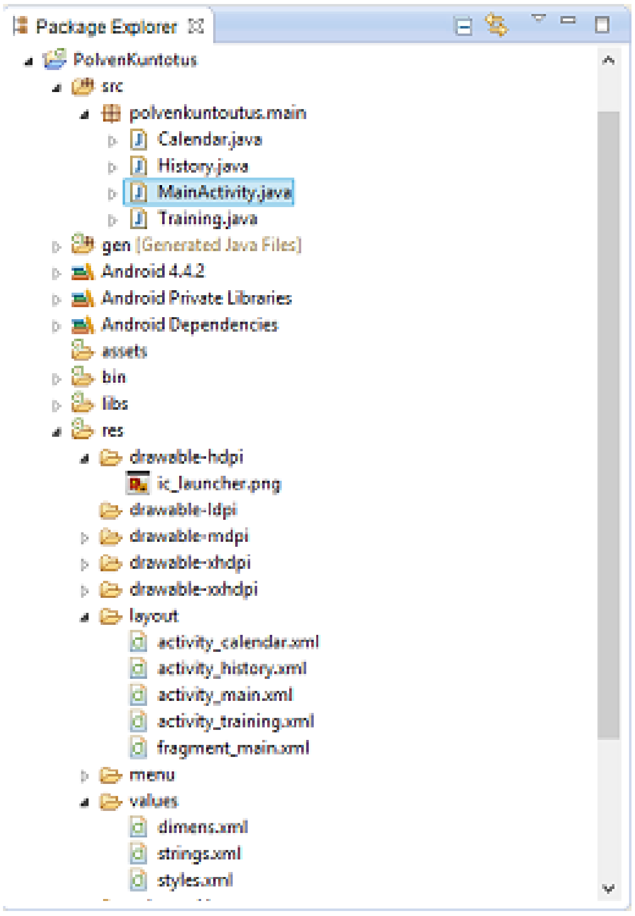
Secondly the wizard creates the layout file. There is a special folder to hold all the layouts named “layout”. The layout files define the user interface screen and each individual screen is defined in its own XML file. The user interface - screens are called activities.

The wizard also creates the folders and proper folder tree (Image 5). Different folders hold different elements of application. For example images, text and layout files are all in their respective folders which are all created by wizard.

3.1.1 Android Manifest file

Android Manifest file is also created. Manifest file is XML file that is mandatory for every Android application. The manifest files has information about the application that system needs before it can start executing the code. Manifest file contains the application permissions information and which permissions it needs to function. When downloading new applications from Google Play store these permissions are shown to user before downloading. That way user can monitor wich applications have access to for example his contacts or messages.

In addition to that it declares the minimun Android API version application requires to run and all the libraries that nees to be linked to application. Manifest file declares the JAVA package which is used also as unique indentifier for the application.



```
Package Explorer
PolvenKuntotus
  src
    polvenkuntoutus.main
      Calendar.java
      History.java
      MainActivity.java
      Training.java
  gen [Generated Java Files]
  Android 4.4.2
  Android Private Libraries
  Android Dependencies
  assets
  bin
  libs
  res
    drawable-hdpi
      ic_launcher.png
    drawable-ldpi
    drawable-mdpi
    drawable-xhdpi
    drawable-xxhdpi
    layout
      activity_calendar.xml
      activity_history.xml
      activity_main.xml
      activity_training.xml
      fragment_main.xml
    menu
    values
      dimens.xml
      strings.xml
      styles.xml
```

Image 5. File tree of Android application.

The different components of application are also listed in manifest file. For example the activities (user interface screens), content providers and broadcast receivers. [19]

3.2 Creating Activities

3.2.1 Layout

Android supports several different types of layout containers. Layout container by definition controls how the elements are arranged and positioned. Different layout containers have different ways of arranging elements. [20]

There are two main types of layout containers. For the first ones you have to specify each component on XML -file separately. Linear Layout and Relative Layout are this type of layout containers.

Relative Layout lets designer to specify how the children objects are positioned relative to each other while Linear Layout adds all children to single row or column. For both of these all objects must be declared separately.

The other main type of layout container uses adapter to lay the objects. This kind of layout doesn’t have to be static but can take information from database or array. Examples of this type of layout container is Grid View and List View.

It is also possible to nest layout containers so that the layout container holds another layout container. [21]

3.2.1 Different types of components

There are several different types of components that can be put into layout. Many of these are more or less similar to each other. For example there are many types of text fields that are specifically designed to contain specific type of information for example plain text, address or password. The similar kind of components are combined under one umbrella term that describes the main behavior of component.

3.2.1.1 User to application communication components

Text Fields as mentioned have several different types from multiline to numeric password. However the main purpose of Text Fields is to allow user to input text into application. Text Fields also support the basic text editing functions like copy, paste and cut. [22]

Buttons are also used for user communicating with application. Buttons can have text to describe the function of button. An image can be used too. There are two ways to link button and desired action. These are by linking the button to method in the activitys XML file by setting value on “android:onClick” parameter. The other way to do it is to link it in code by setting up and event handler for the button. In both cases the method must be defined in activity that hosts the layout. [23]

Radio Buttons and Checkboxes both allow user to select a value from multiple values. The difference between these two is that with Checkbox user can (usually) select multiple options while Radio Buttons limit the number of selectable options to one. [24, 25]

If there are only two options, which cannot exists in parallel (for example feature on/off) there is Toggle Button. Like with regular buttons, it is possible to link the button to action in either the XML or programmatically. [26]

Pickers are designed to let user choose date or time (Image 6). With Pickers the date and time are correctly formatted and valid which simplifies the use of data. [27]

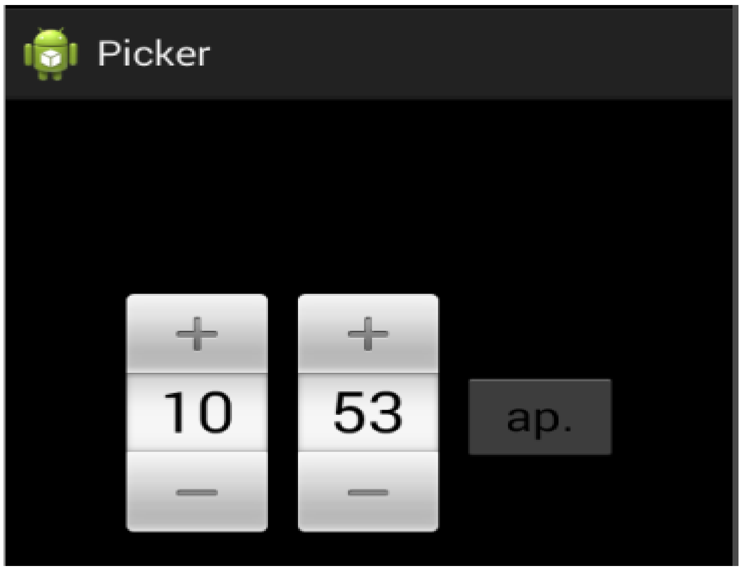


Image 6. Example of Picker

Final component in input category is a spinner which shows a dropdown menu when clicked. It is not possible to link selected choice to method via XML file and therefore it must be done programmatically. [28]

3.2.1.2 Application to user communication

Dialog is a pop-up window that gives user some information (e.g. warnings). It can also ask user to take action before proceeding. [29]

Toast is a small text-box that promptly pops up in order to give user information. This cannot require user to take action and they only appear on screen for pre-set time. [30]

Notification becomes visible in notification area on Android. This is not in the boundaries of regular application user interface. Notifications can be shown to user even if the application is only running in background. [31]

The regular user interface information could be seen as application-to-user communication. However only components giving user information outside the layout are included.

3.3 Database

Android has a built-in database implementation. SQLite is lightweight relational database management system. It is free to use and implement to different applications. SQLite is integrated into host application and therefore there is no need for separate database management. Therefore user doesn’t necessarily know that database is in use. SQLite support several different programming languages and it is embedded in many well-known applications and operating systems. SQLite’s embedded nature makes it well-fit solution for embedded systems. [32]

4 COMPONENTS FOR SENSOR TAG COMMUNICATION

Components listed above are common components with Android development and typically at least a part of them can be found implemented in every Android application. The following components are less common. In this thesis they are introduced mainly from the point of Sensor Tag implementation. However, they are not Sensor Tag specific and can be used for other purposes also.

Here the BLE connection is described in more detail and the details in creating a maintaining a connection are introduced.

4.1 GAP Role

GAP acts as basis for all other Bluetooth profiles. The connection establishment and discovery between two devices are defined by GAP. There are four GAP roles which are central and peripheral, observer and broadcaster. Here we only focus on central and peripheral, since the broadcaster and observers never actually enter connection.

For example in this thesis the Sensor Tag acts as peripheral device and Android-device as central. In practice this means that the sensor tag implements GATT server which stores data that the client (Android application) reads. [33, 34]

4.1.1 GATT

GATT is an abbreviation of Generic Attribute profile. It is used by Bluetooth Smart devices. GATT specifies the transportation and storing operations and common framework for ATT. The data format on GATT server is also dictated by this. In low energy devices it is used for discovering services on device. The two roles of GATT are client and server. [34]

GATT client acts as a client which reads data from the server. The client is the device which uses the data for some application (Image 7).

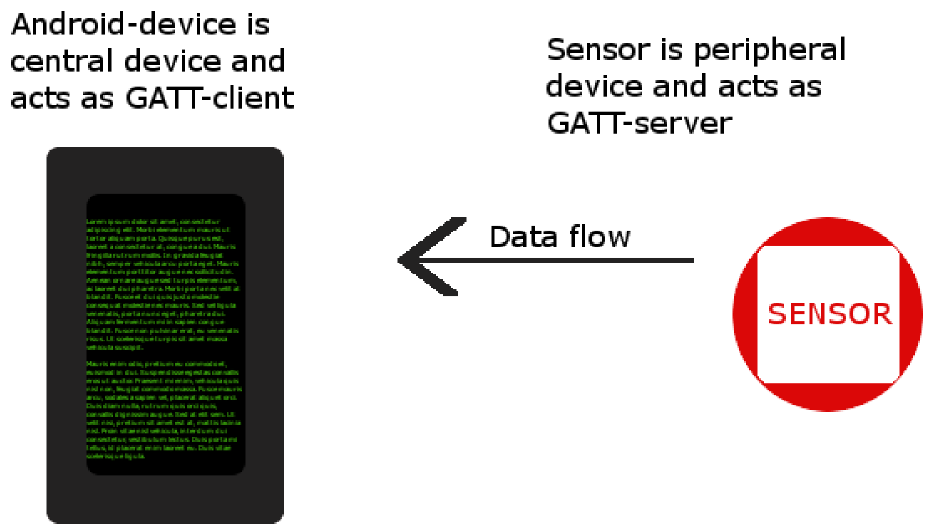

Android-device is central device and acts as GATT-client
Sensor is peripheral device and acts as GATT-server
Data flow
SENSOR

Image 7. The GAP and GATT roles and data flow

GATT client sends protocol requests to server. Server accepts the requests and can send notifications and other information to client. The server offers characteristics and services to client. Characteristics are values that are used in services. Together with value access and display information the value can be retrieved by client. Services are functions and accompanying data that are associated with different kind of behaviors. For example in Sensor Tag all different sensors are a separate service and characteristics are the values obtained from this services. [34, 35]

4.2 Intents

In Android development Intents are objects that allows application to request another app component to take action. Intents can be used to start a service or to deliver broadcast or start an activity. Switching from one activity user interface to another can be done through Intents. Intents can be of two different types, explicit and implicit.

Explicit intents are commonly used by application to start another component in itself. The component is called with its specific name. Immediately after creation the intent starts the component determined in intent. However, explicit intent is difficult to use to request action from external application since the name of component is not known. For these cases there are implicit intents.

Unlike explicit intents implicit intents is handled by Android system which looks for the appropriate component in other applications to take action. This method eliminates the need to know the exact component name and use general action instead. Implicit intents make use of intent filters which are defined in manifest file and describe what kind of intents the application takes care of. The system goes through manifest files of applications in search for appropriate intent filter. It is also possible that Intent matches two or more components. In this situation the system prompts user to choose which application to use. [36]

4.2.1 Broadcast Receivers and Broadcast Intents

Broadcast intents are used to communicate between different application components. In addition Android system uses them to notify applications about system events. For example connected devices and chargers cause a broadcast intent to be sent. Via broadcast intents applications can also send messages to each other.

Broadcast receiver, as the name suggests, is a component that listens to specific broadcast intents. Broadcast receiver allows a piece of code to be executed on demand. Broadcast receivers must be registered for them to start listening. A good practice is also to unregister receiver when it is no longer required. [36]

4.3 Handlers

Android runs its application on one single thread, the user interface, main thread. This however seizes the execution of user interface code in cases where there is a lengthy process to be executed. Therefore these processes are taken care of in background thread. Many components also run their processes in background. This allows the user interface to work while there are on-going action in the background. The downside of background processing is that the user interface is not updated when the process is run on different thread which are not allowed to communicate with main thread.

Handler makes it possible for background thread to send message and runnable objects to main thread and in this way communicate with it. The handlers are attached to the thread they are created. Then the handler can be used to post to the original thread from a different thread. This is commonly used to update user interface after a process in background is complete. Other way to use handler is to schedule tasks to be executed after a defined period of time. [37]

5 IMPLEMENTATION

The implementation process started with recognizing what exactly the application should do and how. Since the focus is on sensors and how to make sensors communicate with Android that is naturally considered the most important aspect. Incidentally that was also the subject that needed the largest amount of research to be done.

5.1 Sensors

The original idea was to use TI Sensor Tag, however research on other sensor manufacturers was done to give general idea what options there are and can a better solution be identified.

5.1.1 Arduino with iProtoXi -motion sensor

One option was to use iProtoXi motion sensor which included accelometer and gyroscope needed for application. The 3-axis gyro is the most important sensor and iProtoXi -motion sensor has that included with two interrupts, which would notify sensor orientation. [38]

On top of that iProtoXi -series has a Bluetooth Low Energy module which allows the connection with Android device.

These sensors would have been connected to Arduino Uno microcontroller. However the solution was discarded mainly because of three significant defects:

- The amount of work that had to be done to implement these sensors with Arduino. In scope of this thesis it was considered to take much focus from Android-development and put it on microcontrollers instead.
- The power consumption. The use of Arduino would have increased the power consumption of the complete device. That would have taken away much usability of the device

- Final reason to exclude iProtoXi –motion sensor was the overall physical size of the sensors and Arduino Uno. The implementation would have taken significantly more space and reduced the usability.

5.1.2 Other sensors-microcontroller combinations

The reasons not to use iProtoXi – motion sensor also applied to majority of other sensor solutions. Also number of products that would implement both, the sensors and BLE on same board, or easily combined, was low. The increase in size and power consumption were unavoidable and therefore the original choice, Sensor Tag, was considered superior.

5.2 Mockup Activities

After the concept was clear and on paper it was time to start programming. The programming started with creating mockup activities to give the overall idea of Main Window and moving from one activity to another.

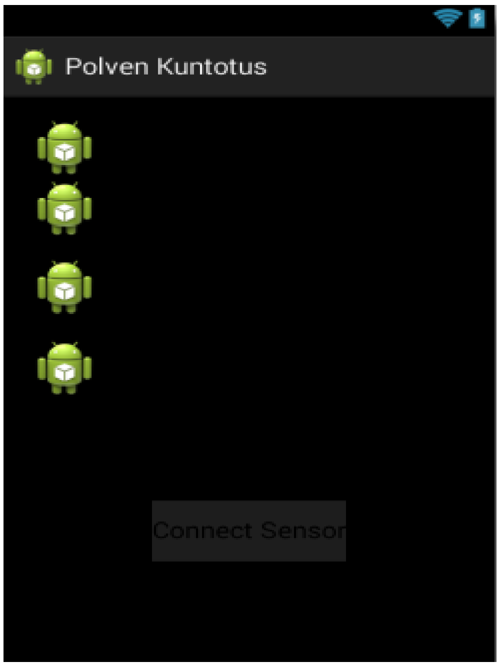


Image 8. Mockup of Main Window

As can be seen in image 8 the user interface is not implemented (Image 8). The mockups serve their purpose, because the implementation was starting from functionality and the user interface would be finalized last.

The pictures of android robots are placeholder images and are later replaced with correct ones. The images also work as buttons to change activity to another mockup user interface.

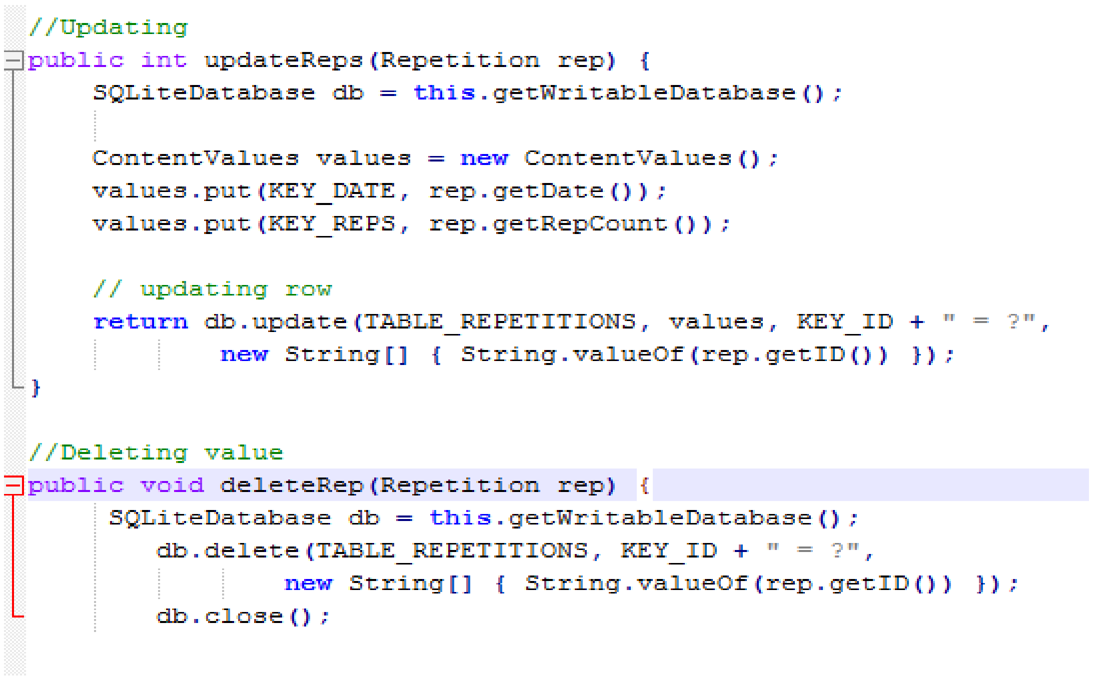
5.3 Database

A database is created to enable application to store data. Android supports SQLite databases on itself. Database consists of three different components.

First component is not actually part of database, but it is the class that calls the database functions. The values gathered from sensors are saved in database. Also the number of repetition user does on any given day is stored.

The second component is the database object. For simplicity a class that has a constructor for data and defines the format for it is created. This also allows the third component, the database handler to handle data as objects.

As stated the third component is the handler that has the functions database has. Database handler has create, read, update and delete functions. (Image 9)



```
//Updating
public int updateReps(Repetition rep) {
    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put(KEY_DATE, rep.getDate());
    values.put(KEY_REPS, rep.getRepCount());

    // updating row
    return db.update(TABLE_REPETITIONS, values, KEY_ID + " = ?",
            new String[] { String.valueOf(rep.getID()) });
}

//Deleting value
public void deleteRep(Repetition rep) {
     SQLiteDatabase db = this.getWritableDatabase();
        db.delete(TABLE_REPETITIONS, KEY_ID + " = ?",
                new String[] { String.valueOf(rep.getID()) });
        db.close();
```

Image 9. Database handler functions

5.4 Initial Sensor Setup

Setting up the Sensor Tag consists of several steps. This is the part where most of the work had to be done in terms of seeking information and studying the implementation. Guides and tutorials for the use of Sensor Tag was almost non-existent and therefore large amount of work had to be done studying the source code of example program and Sensor Tags user guide.

5.4.1 The Layout

First task was to create user interface for development purposes. In the core functionality rather than usability. This user interface is not implemented for the final application.

Development user interface allowed to create the background functionality. It only implemented 2 buttons, for starting and stopping Bluetooth device scan, and a List View in which the found devices would be listed by their MAC code.

List View was made clickable for selecting a device. Clicking item on list also started enabling connection on selected device.

5.4.2 Bluetooth Connection

Since Android 4.3 the Bluetooth low energy has been supported and there are specific methods for establishing this kind of connection.

The connection establishing has 3 steps.

1. Android API has a class that includes Bluetooth adapter through which basic Bluetooth tasks are done. First Bluetooth is enabled on device in case it is disabled. Turning Bluetooth on is done by calling Intent on Bluetooth adapter that prompts user to turn Bluetooth on from Settings. Intent starts an external application which takes care of the rest.
2. Finding low energy devices with Bluetooth adapters LeScan method. The method then scans for low energy devices and returns the found devices. Adding the items on the list is done in the by creating a callback that does the adding after devices are found. This creates a connection with device, however not much can be done with that connection.
3. Connecting to GATT -server is the final phase of enabling connection. This also the phase that enables sensor to be used and gather data from it.

5.4.2.1 Gathering data – GATT server callback

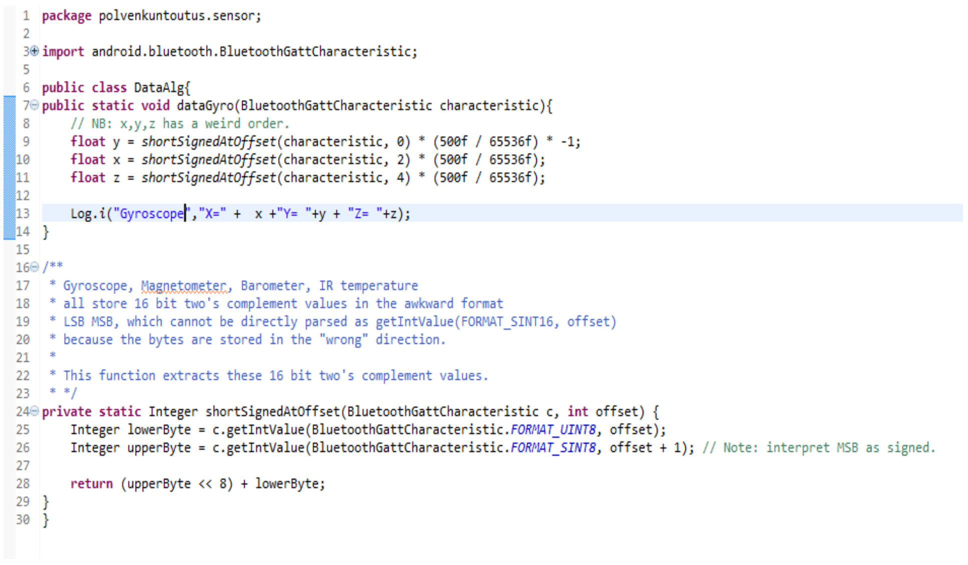
Turning the sensor on and gathering data from sensor are done with callback. The steps for turning the sensor on a gathering data are listed below.

1. First check if the GATT server is connected.

2. After that discover services on device. In this case all the different sensors on Sensor Tag are an individual service. The service discovery must be done before the sensors can be turned on.
3. After discovery the sensor is turned on. This is done by writing a code in byte format to characteristics configuration data space on Sensor Tag.
4. After the sensor is turned on the initial value from sensor is read from data space on Sensor Tag. This way it is also made sure that the sensor is turned on and gathering data.
5. To get data continuously the Sensor Tag has notification function, which sends data on specified time periods and allows application to be updated. In this step the notification function is turned on by writing description to data space of selected sensor.
6. Now the device picks up data and Sensor Tag notifies application on regular intervals. However the data from sensor is raw data and has to be handled before it is on human understandable form. A special class is used to turn data on human readable form (Image 10). This algorithm is provided in Sensor Tag User Guide.
7. Finally data is human readable and can be written out. The data are set on TextView box on user interface

5.5 Combining the components

The components are all functional and the final phase is to combine them and finalize the user interface. The user interface is intentionally left out of this thesis since it is out of the scope.



```java
package polvenkuntoutus.sensor;

import android.bluetooth.BluetoothGattCharacteristic;

public class DataAlg{
public static void dataGyro(BluetoothGattCharacteristic characteristic){
    // NB: x,y,z has a weird order.
    float y = shortSignedAtOffset(characteristic, 0) * (500f / 65536f) * -1;
    float x = shortSignedAtOffset(characteristic, 2) * (500f / 65536f);
    float z = shortSignedAtOffset(characteristic, 4) * (500f / 65536f);

    Log.i("Gyroscope","X=" +  x +"Y= "+y + "Z= "+z);
}

/**
 * Gyroscope, Magnetometer, Barometer, IR temperature
 * all store 16 bit two's complement values in the awkward format
 * LSB MSB, which cannot be directly parsed as getIntValue(FORMAT_SINT16, offset)
 * because the bytes are stored in the "wrong" direction.
 *
 * This function extracts these 16 bit two's complement values.
 * */
private static Integer shortSignedAtOffset(BluetoothGattCharacteristic c, int offset) {
    Integer lowerByte = c.getIntValue(BluetoothGattCharacteristic.FORMAT_UINT8, offset);
    Integer upperByte = c.getIntValue(BluetoothGattCharacteristic.FORMAT_SINT8, offset + 1); // Note: interpret MSB as signed.

    return (upperByte << 8) + lowerByte;
}
}
```

Image 10. Class transferring data to human readable form.

When combining components the data gathered from the sensor is stored in the database. After transferring data to human readable form, in addition to showing it, the database handler is called and the data is stored to the database for use later on. From database the data can be read and used for whatever purposes easily with the database handler introduced above.

5.5.1 Configuring sensor

Before the sensor can start counting repetitions it has to be configured properly. Configuration takes place before starting any new exercise. The configuration is done by prompting the user to take the starting position of specific movement. Then user is prompted to make repetition. During the movement the sensor reads values and takes the highest and lowest value. This value is stored and the initial value is calculated by formula:

$$initial\ value = startingpoint + (0.05 * highest\ value).$$

From the highest value the “end value” is calculated by equation:

$$endvalue = 0.9 * highest\ value.$$

This is to give user some buffer as it is likely that the user will not make the final repetitions as thoroughly as the calibrating one.

One repetition is added whenever the value read from sensor goes lower than initial value, goes up to end value, and goes back to initial value. The repetitions are determined simply by integers. After finished set the number of repetitions are stored in the database together with the date. This way the number of repetitions can be monitored.

6 SUMMARY

The aims set for this thesis were to familiarize with Android development and use of external sensor. These goals were met. However the final product is not complete as the design of user interface was left for later implementation.

The most common components introduced were chosen because they are visible and provide visible feedback to user. This lowers the bar for beginner to start developing own program. However programming skills and understanding are necessary to create functional application.

The biggest challenge turned out to be the amount of studying needed about sensors and the low amount of sources about their use. In retrospective the biggest obstacles were the ones that taught the most and improved application design skills.

Not all obstacles were won by persistence and hard work. The IDE stopped working for no apparent reason for few days, and then as suddenly as it stopped it started working again. The source of the problem nor the solution was never found.

It is noteworthy that the number of integrated development environments is low, and not all of them support all major operating systems or additional tools. Therefore the total costs of tools are difficult to estimate.

The future of the project is still open. The application will be finished but whether it is finished as is or with more functionalities will be left for future evaluation. In any case the application will not be monetized.

REFERENCES

[1] Michael Essany, Mobile Health Care Apps Growing Fast in Number, mHealthWatch, 15.4.2013, Retrieved 28.4.2014, Available at: http://mhealthwatch.com/mobile-health-care-apps-growing-fast-in-number-20052/

[2] Fox, Susannah; Maeve, Duggan; Mobile Health 2012, 8.11.2012, Pew Internet & American Life Project, Retrieved 28.4.2014, Available at: http://www.pewinternet.org/~/media//Files/Reports/2012/PIP_MobileHealth2012_FINAL.pdf

[3] Texas Instruments, Bluetooth low energy SensorTag, Retrieved 25.3.2014 Available at: http://www.ti.com/ww/en/wireless_connectivity/sensortag/index.shtml?INTC=SensorTag&HQS=sensortag

[4] Texas Instruments Wiki 2014, Simplelink SensorTag, Retrieved 26.3.2014, Available at: http://processors.wiki.ti.com/index.php/Bluetooth_SensorTag?INTC=33SensorTag&HQS=sensortag-wiki

[5] Saha, Amit Kumar, A Developer's First Look At Android, in Linux For You, January 2008, Available at: http://tailieuandroid.googlecode.com/svn-history/r8/trunk/Andoid--tech.pdf

[6] Burnette, Ed, Eclipse IDE Pocket Guide, O'Reilly, August 2005

[7] The Eclipse Foundation, Where did Eclipse come from, Retrieved 28.4.2014, Available at: http://wiki.eclipse.org/FAQ_Where_did_Eclipse_come_from%3F

[8] Developer Tools, Android Developer's Guide, Retrieved 26.3.2014, Available at: http://developer.android.com/tools/index.html

[9] Getting Started with Android Studio, Android Developer Tools, Retrieved 21.5.2014, Available at: http://developer.android.com/sdk/installing/studio.html

[10] Norbye, Tor, Android Studio 0.5.8 Released, 8.5.2014, Retrieved 21.5.2014, Available at: http://tools.android.com/recent/androidstudio058released

[11] Xamarin Inc, Xamarin Studio, Retrieved 21.5.2014. Available at: http://xamarin.com/studio

[12] Xamarin Inc. Pricing, Retrieved 19.5.2014 Available at: https://store.xamarin.com

[13] Bluetooth SIG, Fast Facts, Retrieved 25.3.2014, Available at: http://www.bluetooth.com/Pages/Fast-Facts.aspx

[14] Tervakangas, Sallamaari, 2013, Bluetooth low energy -kehitysympäristö, Retrieved 26.3.2014 Available at: http://www.theseus.fi/handle/10024/64917

[15] Texas Instruments, 2013, Texas Instruments CC2540/41 Bluetooth® Low Energy Software Developer's Guide v1.3.2, Retrieved 26.3.2014, Available at: http://www.ti.com/lit/ug/swru271f/swru271f.pdf

[16] Bluetooth SIG, Bluetooth Smart, Retrieved 25.3.2014, Available at: http://www.bluetooth.com/Pages/Bluetooth-Smart.aspx

[17]Bluetooth SIG, Bluetooth Smart Devices, Retrieved 26.3.2014, Available at: http://www.bluetooth.com/Pages/Bluetooth-Smart-Devices.aspx

[18] Bluetooth low Energy, Android Developer's Guide, Retrieved 26.3.2014 Available at: http://developer.android.com/guide/topics/connectivity/bluetooth-le.html

[19] App Manifest, Android Developer's Guide, Retrieved 5.5.2014, Available at: http://developer.android.com/guide/topics/manifest/manifest-intro.html

[20] layout container, Glosbe, Retrieved 5.5.2014, Available at: http://glosbe.com/en/en/layout%20container

[21] Layouts, Android Developer's Guide, Retrieved 5.5.2014, Available at: http://developer.android.com/guide/topics/ui/declaring-layout.html

[22] Text Fields, Android Developer's Guide, Retrieved 5.5.2014, Available at: http://developer.android.com/guide/topics/ui/controls/text.html

[23] Buttons, Android Developer's Guide, Retrieved 5.5.2014, Available at: http://developer.android.com/guide/topics/ui/controls/button.html

[24] Checkboxes, Android Developer's Guide, Retrieved 5.5.2014, Available at: http://developer.android.com/guide/topics/ui/controls/checkbox.html

[25] Radio Buttons, Android Developer's Guide, Retrieved 5.5.2014, Available at: http://developer.android.com/guide/topics/ui/controls/radiobutton.html

[26] Toggle Buttons, Android Developer's Guide, Retrieved 5.5.2014, Available at: http://developer.android.com/guide/topics/ui/controls/togglebutton.html

[27] Pickers, Android Developer's Guide, Retrieved 5.5.2014, Available at: http://developer.android.com/guide/topics/ui/controls/pickers.html

[28] Spinners, Android Developer's Guide, Retrieved 5.5.2014, Available at: http://developer.android.com/guide/topics/ui/controls/spinner.html

[29] Dialogs, Android Developer's Guide, Retrieved 5.5.2014, Available at: http://developer.android.com/guide/topics/ui/dialogs.html

[30] Toasts, Android Developer's Guide, Retrieved 5.5.2014, Available at: http://developer.android.com/guide/topics/ui/notifiers/toasts.html

[31] Notifications, Android Developer's Guide, Retrieved 5.5.2014, Available at: http://developer.android.com/guide/topics/ui/notifiers/notifications.html

[32] Simon, Jonathan, Head First Android Development, O'reilly, 2011

[33] Apple Inc., About Core Bluetooth, 2013, Retrieved 19.5.2014, Available at: https://developer.apple.com/library/ios/documentation/NetworkingInternetWeb/Conceptual/CoreBluetooth_concepts/AboutCoreBluetooth/Introduction.html

[34] Bluetooth SIG, Technology Overview - GATT, Retrieved 19.5.2014, Available at: https://developer.bluetooth.org/TechnologyOverview/Pages/GATT.aspx

[35] Ole Morten, Forum Answer, Nordic Developer Zone, 2.7.2013, Retrieved 19.5.2014, Available at: https://devzone.nordicsemi.com/index.php/what-is-a-client-and-server-in-ble

[36] Intents and Intent Filters, Android Developer's Guide, Retrieved 19.5.2014, Available at: http://developer.android.com/guide/components/intents-filters.html

[37] Handlers, Android Developer's Guide, Retrieved 14.5.2014, Available at: http://developer.android.com/reference/android/os/Handler.html

[38] iProtoXi, iProtoXi-Liikesensori, Retrieved 18.5.2014, Available at: http://iprotoxi.fi/index.php?option=com_content&view=article&id=107&Itemid=613&lang=fi