Optimizing Bluetooth communication between medical devices and Android

L.Tanaskoska[1], A. Stamenkovska[2], S.Savoska[3], I. Jolevski[4], N. Blazheska– Tabakovska[5] and B. Ristevski[6]

[1,2,3,4,5,6] University of St. Kliment Ohridski, Faculty of ICT, Bitola, North Macedonia

[1]lencetanaskoska@gmail.com, [2]andrea.stamenkovska@gmail.com, [3]snezana.savoska@fikt.edu.mk,[4]ilija.jolevski@fikt.edu.mk, [5]natasa.tabakovska@fikt.edu.mk, [6]blagoj.ristevski@fikt.edu.mk

Abstract: Healthcare is one of the biggest users of emerging Information and communication technology, especially when the wide range of people have to use healthcare devices. With the concept of Internet of Things, these trends increase permanently as ubiquitous connected devices toward implementation of m-health concept. Usually devices are connected with Bluetooth that has many communication obstacles. Our research intends to provide some experience in optimizing Bluetooth communication between a couple of medical devices with Android application. Through usage of devices intended for short time tele-monitoring of the distance patient in the IPA 2 project Cross4all, we provide some extended experience for implementation of selected devices and gaining data suitable for wider use in the patients' Personal Health Record (PHR).

I. INTRODUCTION

Nowadays, health information technology is the trend of international development. Health information technology is applied information technologies to medicine and healthcare which supports health information management across computerized systems as well as secure exchange of health information between consumers, providers and monitors. These technologies have many benefits as the ability to store and retrieve data, enable communication between healthcare stakeholders, provide medication safety that is improved through the use of suitable secure technology and medication alerts, increasing ability for getting patients' health tracking data as well as patient access to their medical records, known as Personal Health Record (PHR) etc.

On one hand, the rapid development of information technology is attracting more and more medical institutions to accelerate the use of health information technology and assistive living technology [1]. Also, there is an increasing interest in the production of devices that enable monitoring of healthcare connected data such as blood pressure, blood sugar, heart rate etc. [2]. The rapid development of mobile applications provides a new development direction for the medical industry and pervasive computing [3]. As one of the most used mobile operating system in the world is Android, as an open source mobile operating system based on Linux platform, it is also the most used for development of mobile application toward the concept named mobile health (m-health) [2, 4]. With its open and convenient development mode has occupied an important position in the field of mobile applications. But the possibility of communication between Android devices and health monitoring devices known as BLE devices is due to Bluetooth Low Energy (BLE) [5].

Table 1. Services and characteristics for devices

Type	BLE, GATT
Medisana BU550 - Blood pressure monitor	One service One characteristic
Medisana PM 150 - Pulse Oximeter	One service Five characteristics
MediTouch 2 - Blood Glucose Monitor	One service Three characteristics

We have to highlight what is BLE device and how to communicate with them. Bluetooth low energy device is a device that communicates with other devices using Bluetooth [6]. It's a simple device which can send and receive data as a byte array. In order to read data from the device you must connect with it. To connect with it, there are steps that must be taken. First, you have to scan for the device. For that purpose, we are using the default Bluetooth scanning service from Android. For each discovered BLE device we are receiving event and data specific for that device. Once we discover the wanted device we connect with it. Next, we need to make the communication with the device or retrieve the data we need. Each BLE device has a communication interface that is exposed through services and characteristics. A characteristic is as a command or function, it's a place you read or write to fetch data or control the device. Each characteristic can also have one or more descriptors that tend to be access less frequently by application code. Services are

collections of characteristics, group of related commands/functions. Also, each service and characteristic has a universally unique id (UUID) that is used when setting up the communication with the BLE device [7]. There UUIDs are usually found in the device documentation provided by the manufacturer.

For the project Cross4all from IPA2 cloud based architecture is created [8] with development of PHR that have to be the main base of collecting valuable healthcare data for patients cross border. As a part of patients' PHR data, data collected from short time monitoring period have to be provided using m-health concept. So, we start to collect these data from selected devices connected to patient, sending then into cloud database. We worked on the Medisana devices equipped with Bluetooth. According to project' demands and m-health principle, we worked on optimizing the Bluetooth communication between these medical devices and Android device in order to gain better results from intended devices.

The paper is organized as follows. After Introduction, related works are taken into consideration. The proposed optimization processes are explained in the next section. Results and discussions are provided in the fourth section. Concluding remarks highlighted some future work intended for the project and propose some improvement of the concept.

II. Related Works

Considering that migration and traveling of the people increase all the time, people as patients are more and more mobile and change their need for healthcare. Many times, they use electronic health methods as e-health and m-health to satisfy their need for healthcare. There is an arising need of medical facts for patients also because of providing evidence based medicine [9, 10]. A lot of researchers' work provide a new concepts and methodologies in order to enable measuring instruments, devices and other equipment for this objective, especially devices according to the concept Internet of Things (IoT). One of the best solutions to support these new trends is collected healthcare data from all patients' healthcare activities, medical examinations, laboratory results, prescription and referral. All these data have to be stored in some healthcare digital repositories in electronic format as Electronic Health Records (EHR) or PHR [11], saved and accessible for the patients or selectively by medical staff [12]. All stored data and medical information play a positive role in medical practitioners' decision making for the patient healthcare.

The growth of using mobile and ubiquitous computing technologies contributes to increase the number of records regarding personal health of patients. IoT includes the development of wearable measurement sensors connected with Bluetooth, which are capable to capture and store health-related data, intended to store in PHR [3, 11]. As the mobility increasing, researchers give special attention to data security and privacy because of healthcare data sensitivity. Silva et al. proposed encryption solution, called data encryption for mobile health applications (DE4MHA), trying to guarantee the best confidentiality, integrity, and authenticity of m-health systems users' data [13].

Many efforts are made to create e-health and m-health application last two decade. Some authors highlighted some extensible, scalable, highly interoperable and customizable platforms that provide cloud based architecture, based on components and services architecture. It can be used to develop several remote mobile monitoring m-health systems [14]. With the arise usage of IoT concept, m-health is widely used as medical help for medical practitioner, especially for cardiovascular research and patients' health monitoring [4]. Some medical m-health/e-health vendors already have their own platform that are connected with their own cloud with included data encryption and protect data from wider usage. Bluetooth connection can be sometimes insecure and for this reason, they provide accounts with secure data control and Service oriented architecture [5].

M-health also has the potential to promote patient's self-management, especially for high risk patients as hypertension and cardiovascular disease, as a complement to the doctor's intervention [15]. They encourage greater participation in medical decision making. With cloud based m-health, the medicine is empowered with tools which bring data for evidence based medicine [1]. There are a lot of solutions for mobile monitoring as m-health solution. The concept MoMo (Mobile Monitoring) Framework is a solution that allows for patient mobile monitoring through mobile phones and biometric devices (blood pressure meter, glucometer and others) and is widely spread nowadays [16]. All of these devices usually have Bluetooth and some similar solution for data collection. So, optimization of such connection can give great results for data collection

III. Used Devices And Optimized Bluetooth Communication With Android Os

A. First Insights

At the beginning of the project we came across several personal healthcare devices produced by Medisana corporation: Medisana Blood Pressure Monitor BU550, Medisana Pulse Oximeter PM 150 and Medisana Blood Glucose Monitor MediTouch 2 listed in Table 1. Each of these devices supports different communication services and monitors different health related data. While trying to establish communication with the devices we encountered several problems. The first problem was that with some devices it was possible to connect only after the values had been measured, while for other it was possible at the beginning while the measurement was in progress. Another problem we encountered was unexpected disconnection of the devices during communication, and the last problem was that not all devices were using the UUIDs that were specified in GATT standard [17]. Due to the above problems we

had to make optimizations in order to have better and valid communication with the devices via BLE.

B. Proposed optimization

Due to the growing popularity if M-health, a number of mobile applications have been developed that allow communication with BLE devices and can be found on the Google Play Store. However, most of them only communicate with one specific device and are not user friendly for elderly people and people with disabilities. For this purpose, we have developed application that will enable communication with multiple personal healthcare devices while being user friendly. We developed an application that provides a view of all the measurements made on the devices listed in the Table 1, then option to filter the measured data by day, week, month and graph representation of it, storing data in the cloud, additional analysis and statistics, as well as early detection and prevention of health disorders.

Medisana blood pressure Monitor BU550 connects works the same as any other BLE device, using the same protocol for connecting with the Android device as described above. So, the first step in the communication process is to verify that the device we are trying to connect supports Bluetooth and has granted permissions for Bluetooth and location by checking for BLUETOOTH, BLUETOOTH_ADMIN, ACCESS_COARSE_LOCATION and ACCESS_FINE_LOCATION permissions in Android OS. If these particular permissions are not granted we must request permissions in order to proceed, but if permissions are granted, the next step is to ensure that the user has enabled Bluetooth. Also, it checks that we are using BluetoothAdapter class from Android SDK. When we have granted permissions and enabled Bluetooth we can allow the user to scan for BLE devices. Using the startDiscovery method from BluetoothAdapter class we are starting the scan for BLE devices, but firstly we need to register broadcast receiver which will listen for data/packages from BLE devices and when receives a package will notify us that the BLE device has been discovered. When we register the receiver, we also set filter with action FOUND, so when the onReceive method would be called we could compare the action from the intent that we just received with action Found. If the action from the intent we received equals to action FOUND we can get the extra data that the intent is caring. The intent that we receive is holding an amount of data that might be useful to us. The data is stored in some type of map where each data part is associated with a key. The key that is associated with object that has data about the device that we discovered is "android.bluetooth.device.extra.DEVICE" or we can find it as static property in BluetoothDevice class as EXTRA_DEVICE. The associated object with this key is of type BluetoothDevice and has multiple properties that we can use such as the name of the device, its mac address, type, alias, alias name, battery level, if it's connected or encrypted and the most important property the array of UUIDs that this device is supporting.

Other than these useful properties the BluetoothDevice, class has methods that we need to use in order to get data from the BLE device such as createBond for creating a bond and pairing with the device and connectGath for actual connection with desired device. ConnectionGath method has multiple implementations with various numbers of parameters and we should choose the one implementation that suits the best. In this case we are using 'connectGatt (Context context, Boolean autoConnect, BluetoothGattCallback callback)' where the autoconnect is set to true and we are passing callback of BluetoothGattCallback type. It's best to have its own class for each device we want to implement. The custom class should extend BluetoothGattCallback class and override its methods such as onConnectionStateChange, onServicesDiscovered, onDescriptorWrite, onDescriptorRead, onCharacteristicRead and onCharacteristicChanged which we will use in the further communication.

As mentioned before each device has its unique identifiers called UUID that we use for communication. They can be service uuid, characteristic uuid or descriptor uuid. Each device has one or more services, each service has one or more characteristics and each characteristic has one or more descriptors. Medisana Blood Pressure Monitor BU550 has multiple services and the most important service for us is the blood pressure service with uuid: '00001810-0000-1000-8000-00805f9b34fb'. From this service the characteristics that we need is the one with uuid '00002a35-0000-1000-8000-00805f9b34fb' and in order to get data we'll need descriptor with uuid '00002902-0000-1000-8000-00805f9b34fb'. These uuid-s can be found in the BluetoothDevice object that we receive in onReceive method.

After we call connectGatt method we are waiting for data in the callback we passed as parameter. OnConnectionStateChange method will be first called to notify us that connection state changed. In this method we are receiving BluetoothGatt object which we will use to retrieve data, the current status of the ble device which can be disconnected (state 0), connecting (state 1), connected (state 2) and disconnecting (state 3) and the new state of the ble device. If we receive status 2 (connected), then we can call discoverServices method of the BluetoothGatt object that was received in onConnectionStateChanged method. This is an asynchronous operation so when the discovery is completed the onServiceDiscovered method will be called as the callback will be triggered by the operation.

Once the service is discovered we need to setup the characteristic and the descriptor in order to properly receive data. We are using BluetoothGat object that we receive as a parameter within the onServiceDiscovered method to call setCharacteristicNotification method so we can enable

notifications for the characteristics that we need to retrieve data. After that we need to enable indication or notification on the descriptor. Which value will be set depends on the properties of the characteristic we are using to get data. For Medisana Blood Pressure Monitor's descriptor we are setting enable notification as value.

When it's all setup the only remaining thing is to wait for the callback to trigger onCharacteristicChanged method. In this method we are receiving BluetoothGatt object and BluetoothGattCharacteristic object. BluetoothGattCharacteristic object's property value consist the data that we are waiting to receive. Calling the getergetValue will give us byte array with the data of the measurement of the device.

Because we need to parse the data into readable form, we need to parse the byte array into hex string. The hex string contains a lot of information, but the ones that we need are systolic, diastolic and heart rate value. For systolic value we need to get the second and the third character from the hex string and parse it to integer, for diastolic value we need to get the sixth and seventh character from the hex string and parse it to integer, and for the heart rate value we need to get the 28th and 29th character from the hex string and parse it to integer. The onCharacteristicChanged method will be called as many times as there are records in the memory of the device, so the best practice is to wait a couple of seconds after each data received. If there is no following onCharacteristicChanged call it means that we have received the last and the latest measurement.

Although the main course of the communication with the Medisana Pulse Oximeter is relatively the same as Medisana Blood Pressure Monitor, there are some differences. The main difference is that the Pulse Oximeter for transmitting data uses specific service with UUDS outside of the ones specified with GATT standards code, the UUID of that service is 'ba11f08c-5f14-0b0d-1080-007cbe3d328f'. For the Medisana Blood Pressure communication we were using only one characteristic with one descriptor, but to setup the pulse oximeter we need multiple characteristics and they all use the same descriptor (uuid: '00002902-0000-1000-8000-00805f9b34fb').

The characteristics with following UUIDs: '0000cd02-0000-1000-8000-00805f9b34fb', '0000cd03-0000-1000-8000-00805f9b34fb', '0000cd01-0000-1000-8000-00805f9b34fb', '0000cd20-0000-1000-8000-00805f9b34fb' only need their descriptors to be set to enable sending notification by setting the value to the byte array {0x01, 0x00}. We need to make sure that the characteristics will be set one at a time, so we will need to wait for onDescriptorWrite method to be called so we can setup the descriptor of the next characteristic. When we are finished with the three characteristics on the last onDescriptorWrite call we need the characteristic with UUID '0000cd04-0000-1000-8000-00805f9b34fb'. This characteristic is specific because we need to write specific byte array {(byte) 0xaa, (byte) 0x55, (byte) 0x04, (byte) 0xb1, (byte) 0x00, (byte) 0x00, (byte) 0xb5} by setting the value property to the characteristic and then using the writeCharacteristic method of GATT object to write the data which will allow the device to send data to the android application. After all is set, we are waiting for onCharacteristicsChanged method call. This method will be called only once as we receive only the currently measured data as byte array from which the third byte is the saturation value and the fourth byte is heart rate value. We need to parse the received data to hex string and then to integer in order to get the real value.

Figure 2. Synchronizing data between medical device and mobile app

The communication with MediTouch 2 - Blood Glucose Monitor is very similar with Medisana Blood Pressure Monitor, because unlike the Pulse Oximeter we do not need to write data using the characteristics we need to enable the notifications of the characteristics descriptors (one at a time) and wait for onCharacteristicsChanged method to be called. For communication we are using the service with UUID '00001808-0000-1000-8000-00805f9b34fb' and characteristics with UUIDs '00002a18-0000-1000-8000-00805f9b34fb', '00002a34-0000-1000-8000-00805f9b34fb' and '00002a52-0000-1000-8000-00805f9b34fb'. The onCharacteristicsChanged will be called as many times as the number of measurements in the device memory so we should use the same practice as the one for Medisana Blood Pressure Monitor. The last method call will provide the current measurement as byte array.

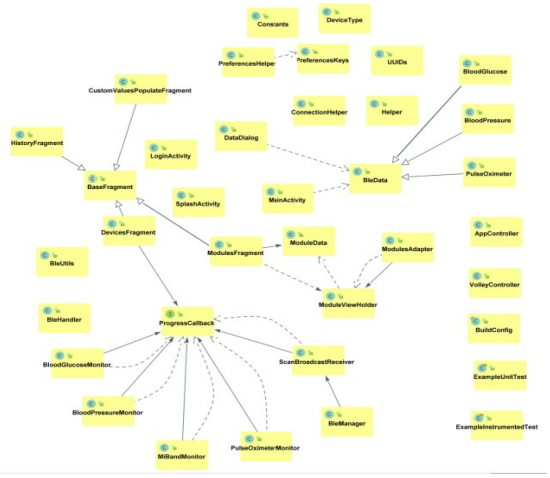

Figure 1. UML Class diagrams used for the application

As we mentioned before, many classes already made by the device vendors are used. Some of the classes are made for this purpose. Fig. 1 shows the class diagram created for the purpose of this research and class connections. The most important class is BleManager class which consists of methods that help with the communication and the most important thing is that it is a singleton so we will be sure that the communication with different devices will not be mixed up. This class has startScanPeriodically public method that allows us to scan by device type and as long as it's needed to find the device. The list of the devices is managed by DevicesFragment class and the user can choose device by clicking on a list, which invokes the startScanPeriodically method. Periodical scan is managed by the BleHandler class which extends Andorid's Handler class and schedules repeatedly stopping and starting the Bluetooth scan. Before the scan has started we are using ScanBroadcastReceiver class to register a receiver for the specified device. The ScanBroadcastReceiver class has onReceive method that handles whether the received device is the one that we need. Once the device is discovered we are using a class of BluetoothGattCallback type to handle whole communication. For each device we have created BloodGlucoseMonitor, BloodPressureMonitor, MiBandMonitor and PulseOximeterMonitor classes that implement all needed BluetoothGattCallback methods. BleDeviceState enumeration and BleUtils class help within the communication process. Also, we have created ProgressCallback interface in order to pass data to the devices fragment or where is needed and to control the progress that is displayed. For collecting the data we have one general type BleData class and classes for each measurement such as BloodGlucose, BloodPressure, PulseOximeter that extend the BleData class. When the data is received we are using DataDialog class for visual display.

IV. Results And Discussions

Integrating different BLE connected healthcare devices is a difficult task since the different ways of organizing the communication protocol and entities per device and per vendor. Although the Android ecosystem is trying to overcome this multiverse with introduction of the Google FIT API, vendors still are not in unison. On the other side the users of such devices are forced to use different healthcare application per device and thus having their healthcare measurement data scattered over several apps and clouds. Creating customizations per device has led us in creating software structures and Java classes that can be used as templates in future integrations of other BLE based healthcare measurement devices.

V. Conclusion

The research we made enables development of mobile healthcare applications with customized and optimized communication stack and procedures with different BLE enabled healthcare devices (Fig. 2). It will contribute to a better and easier access to personal healthcare devices and use by elderly people and people with disabilities, then better and easier access to measured data by doctors and users themselves, reducing costs and providing better healthcare services to all patients.

This paper may be a potential starting point of more broadly defining the design and templates for the medical data transfer data classes and BLE communication classes for Android OS.

Acknowledgment

This research is supported by the EU and North Macedonia within the IPA project CROSS4ALL (Cross-border initiative for integrated health and social services promoting safe ageing, early prevention and independent living for all), IPA 2 – 2014-2020 (Cross4all-CN1-S01.2-SC015).

References

[1] Kravets A., Shcherbakov M., Kultsova M.&Shabalina O. (Editors), Creativity in Intelligent Technologies and Data Science, First conference, CIT&DS 2015 Volgograd, Russia, September 15–17, 2015 Proceedings, Springer, pp.

[2] Berlo A.and all (Editors), Ambient Intelligence – Software and Applications, 4th International Symposium on Ambient Intelligence (ISAmI 2013), Springer (2013)

[3] Aleksov V., Savoska S., Trajkovic V., Applications of pervasive computing and mobile services in health care, In proceedings of ISGT 2018 conference, Sofia November 2018, pp.

[4] Chaudhry M.S. Sarwar, MuthiahVaduganathan, Stefan D.,Anker, Stefano Coiro, Lampros Papadimitriou, Joel Saltz, Elinor R. Schoenfeld, Richard, L. Clark, Wilfried Dinh, Frank Kramer, Mihai Gheorghiade, Gregg C. Fonarow, Javed, Butler , Mobile health applications in cardiovascular research. Ijca (2017), doi:10.1016/j.ijcard.2018.06.039

[5] Developers web site, https://developer.android.com/guide/topics/connectivity/bluetooth-le, Accessed 10.9.2019

[6] PUBMED, US national library of medicine, https://www.ncbi.nlm.nih.gov/pubmed/28508815, Accessed 5.8.2019

[7] Bluetooth web site for UUID, https://www.bluetooth.com/specifications/assigned-numbers/16-bit-uuids-for-members/, Accessed 10.8.2019

[8] Savoska S., Jolevski I., Architectural Model of e-health System for Support the Integrated Cross-border Services, In proceedings of ISGT 2018 conference, Sofia November 2018

[9] Dagnano S. and Macherone A., Editors, Unraveling Exposome, © Springer international publishing, 2019; Smith M.T. and all, Chapter 1: Using Exposomics to Assess Cumulative Risks from Multiple Environmental Stressors, pp.3-22

[10] Andreu-Perez J, Poon CC, Merrifield RD, Wong ST, Yang GZ. Big data for health. IEEE J Biomed Health Inform 2015 Jul;19(4):1193-1208. [doi: 10.1109/JBHI.2015.2450362] [Medline: 26173222]

[11] Roehrs A. and all, Personal Health Records: A Systematic Literature Review, Journal of Medical Internet Research, 2017;19(1):e13, doi:10.2196/jmir.5876

[12] Gay K. and all, Electronic Health Record Error Prevention Approach Using Ontology in Big Data, 2015 IEEE 17th International Conference on High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), and 2015 IEEE 12th International Conf on Embedded Software and Systems (ICESS), DOI: DOI 10.1109/HPCC-CSS-ICESS.2015.168

[13] Silva B.M.C.and all, Towards a Cooperative Security System for Mobile-Health Applications,

[14] Ángel Ruiz-Zafra, Kawtar Benghazi, Manuel Noguera, and José Luis Garrido, Zappa: An Open Mobile Platform to Build Cloud-Based m-Health Systems (in: BerloA.and all (Editors), Ambient Intelligence – Software and Applications, 4th International Symposium on Ambient Intelligence (ISAmI 2013), Springer (2013))

[15] Omboni S. and all, Telemedicine and M-Health in Hypertension Management: Technologies, Applications and Clinical Evidence, High Blood Press Cardiovasc Prev DOI 10.1007/s40292-016-0143-6, Springer, 2016

[16] Villarreal1 V. and all, A Systematic Review for Mobile Monitoring Solutions in M-Health, J Med Syst (2016) 40:199 DOI 10.1007/s109, 6-016-0559-5, Springer, 2016

[17] GATT standards web site, http://dev.ti.com/tirex/content/simplelink_cc2640r2_sdk_1_40_00_45/docs/blestack/ble_user_guide/html/ble-stack-3.x/gatt.html# , Accessed 10.8.2019