Using Bluetooth on Android Platform for mHealth Development

Evgeny Stankevich, Ilya Paramonov
Yaroslavl State University
Yaroslavl, Russia
{stankevich.evg, ivparamonov}@gmail.com

Abstract

There are many mobile devices running Android Operating System, which makes this platform attractive for mHealth development. Cross-platform approach can be used to reduce the cost of such systems.

In this paper we discuss possibility of use the Android platform as one of the target platforms in cross-platform development of applications which uses Bluetooth to transmit medical data. In particular, we highlight problems with the use Bluetooth from native code on Android platform, and propose our solution of this problem.

Index Terms: mHealth, Android, Bluetooth.

I. Introduction

Mobile health care (mHealth) is an area of electronic health (eHealth) directed towards provision of health care services and information via mobile technologies, such as mobile phones and Personal Digital Assistants (PDAs) [1]. The use of mobile and wireless technologies to support the achievement of health objectives has the potential to transform the face of health service delivery across the globe [2]. There are several directions of application development in this area: illness guides, disease and epidemic outbreak tracking, remote diagnostic and treatment support of patients.

There are many different mobile platforms, and each of them usually provide its own development tools. So, it is rather expensive to develop applications for each mobile platform separately. By using cross-platform framework and libraries for application development, we can reduce the cost of mobile healthcare systems, because applications or separate modules of application based on such frameworks can be easily ported across platforms.

At the present time, there are many mobile devices running Android OS, which makes this platform attractive for development of mHealth applications.

Our problem was porting of the existing application for remote ECG monitoring from Maemo to Android platform. We also used experience, we gained during the porting, to elaborate a common solution for development of cross-platform applications involving communication via Bluetooth with the use of Qt framework. In this paper we present our approach and architectural solution we developed.

The paper is structured as follows. Section 2 highlights some issues of cross-platform mHealth development involving Bluetooth technology. Section 3 describes the ECG monitoring application for Maemo developed in MSU, which we ported to Android, the problems we faced during the porting Android, and the idea of our solution. The architecture of the solution is described in section 4. The major results of the work are summarized in the conclusion.

II. Issues of Cross-platform mHealth development

When we develop complex mHealth applications, on the one hand, it is desirable to support as many mobile devices and platforms as possible, and on the other hand, as many medical equipment of the same type as possible. Support for many devices and platforms is very important, because it allows more people to use our application. To support multiple mobile platforms there are frameworks and libraries which allow to develop cross-platform applications. The main benefit of cross-platform frameworks is that we can implement a significant part of application as platform-independent that can be used without any changes on any supported platform.

Cross-platform development becomes difficult, when one of the target platforms is Android, because there are no official cross-platform development tools for this platform. There are two development tools for Android: Android SDK and Android NDK. Android SDK uses Android platform API in Java. Android NDK was initially intended for performance-critical parts of application, it allows to implement them in native code with the use of C and C++ programming languages. In the latest versions of NDK it is possible to create pure native applications. Necessitas project [3] uses NDK to allow Qt-based development on Android platform. So, it makes porting of existing Qt applications possible.

A lot of medical equipment uses Bluetooth for data transfer. But there is no cross-platform Bluetooth API exists [8]. So if we would like to target an application for multiple platforms, and it uses Bluetooth, we can not implement Bluetooth communication in the same way for all the platforms.

To make mobile health care system more available, it also very important to develop mHealth applications which should be interoperable with medical devices of the same type (pulse oximeters, blood pressure monitors, and others) produced by different vendors. To provide interoperability there are standards for medical equipment.

As an example, for equipment using Bluetooth, there is Health Device Profile (HDP), which was developed to avoid dependencies on a specific stacks and in some cases on the operating systems used [4]. HDP is designed to facilitate transmission and reception of medical device data. HDP allows a device to be identified. It enables identification of the manufacturer, product id (type of equipment), product version, etc.

There are several implementations of Bluetooth stack supporting HDP profile, for example Bluetooth Toshiba stack [5] and BlueCode+ [6]. Android operating system uses Bluez stack, which supports HDP as well. Bluez also used on Maemo platform and on other Linux-based systems. Because of these platforms uses the same Bluetooth stack implementation, we could implement Bluetooth communications the same for them, using the same API. Such approach could help to build common solution for development of cross-platform applications which uses Bluetooth.

III. Porting of Existing Qt Based Application to Android

Our problem was porting of the existing application for remote ECG monitoring to Android platform. The application was developed with the use of Qt framework for Maemo platform in Moscow State University [7]. It uses Alive Heart and Activity Monitor to acquire data. The Bluetooth protocol is used for communication, and the connectivity is based upon Bluez library.

To use the system, user connects the electrodes of the ECG monitor to the body, switches on the monitor, and starts the application. The application asks the user to select the monitor if there are several of them. After the monitor has been selected the application creates

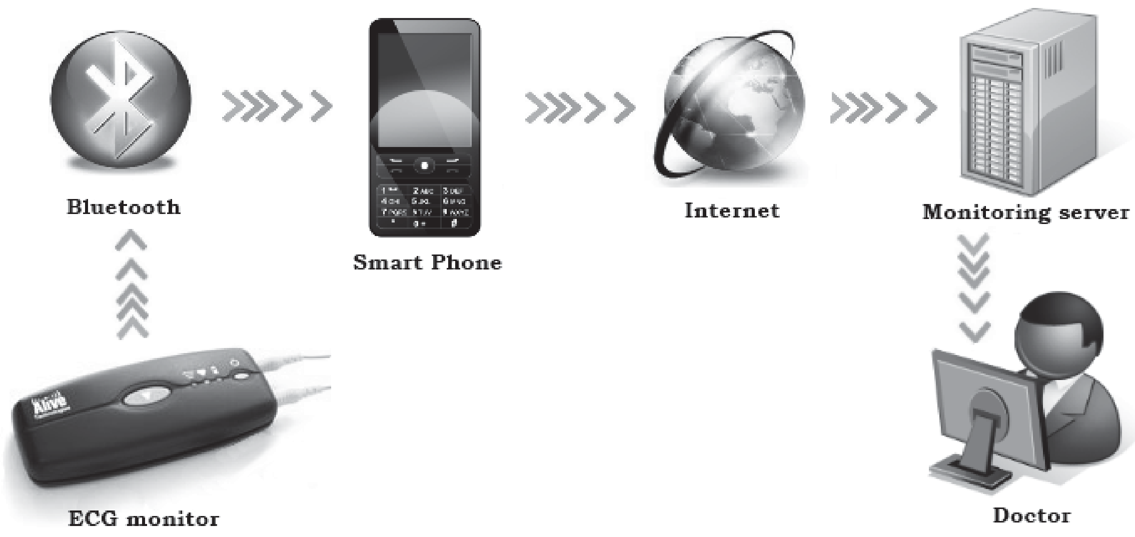

Fig. 1. Architecture of the ECG monitoring system developed in MSU.

RFCOMM socket (emulated serial port), connects to the monitor, and starts to receive ECG and accelerometer (ACC) data.

Using special algorithm the application determines QRS complexes and beat-to-beat (RR) intervals of ECG signal, then calculates heart rate. Electrocardiogram, ACC data and the heart rate are shown on the screen. Then the collected and calculated data is sent to the remote server for further analysis. The treating doctor can use these data to monitor the condition of the patient. The architecture of the ECG monitorng system is depicted in fig. 1.

We used facilities of the Necessitas project for porting of the application to Android platform. Each application built with the use of Necessitas contains two parts. The first part is implemented in C/C++ languages. The second part is the Java module. To integrate Qt/C++ and Java parts of the application Android NDK is used.

Essential part of the Necessitas is Ministro service, pure Java application, which manages the installation process of the Qt libraries. When the Qt based application starts, it searches for the service, if the service is not installed, the application will ask user to install it using Android Market. When the service in installed, the application requests modules from it and waits for the callback. Ministro then downloads all the necessary libraries, if they are missing, callbacks the application with the list containing all the necessary libraries to run. The application loads libraries provided by Ministro service and starts.

We used the HTC Desire mobile phone running Android 2.2 for porting experiments. When using Necessitas project for the application porting, we encountered a problem with using of the existing mechanism of Bluetooth communication.

Android platform uses Bluez library for Bluetooth communication. There are two ways to interact with Bluez: use the API, provided by Bluez library, for C/C++ code or use Bluez D-Bus daemon called hcid. D-Bus is a cross-platform inter-process communication system. It makes it simple and reliable to code a “single instance” application or daemon, and to launch applications and daemons on demand when their services are needed.

When we used Bluez library API in the application on our device, we could not create

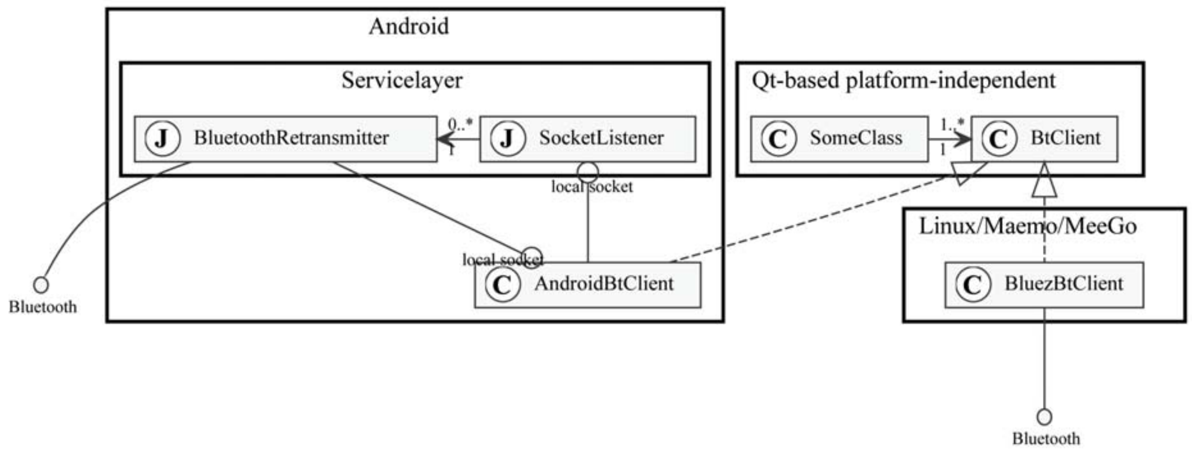

Fig. 2. Architecture of the proposed solution. Rounded "C" denotes the classes implemented in C/C++ (native code). Rounded "J" denotes Java classes.

the RFCOMM socket from the C++ code. During the creation of the socket we received the error code which corresponds to the message "No route to host". We also found out that this problem does not appear in all devices. There is an application using Bluetooth from the native code in the same way and there is a list of supported and unsupported devices for this application [9]. This lead us to the conclusion that, thus it is possible to use the same mechanism of Bluetooth communication on Android as on Maemo platform, but it narrows down the list of supported devices. Unfortunately, our test device (HTC Desire) is in the list of unsupported devices.

To communicate with the remote Bluetooth device from native code we implemented a special component in Java, which interacts with the Bluetooth device via Android Java API and allows to transmit all requests and data to and from the remote device to other processes by means of local (Unix) socket. The Unix socket is used as control and streaming data channel, so any application can send and receive data via Bluetooth regardless of the language it is developed in. We called the corresponding Java module Bluetooth Retransmitter.

IV. Architecture of the Proposed Solution

We applied the modular approach during the development of our application. All functionality was divided into two parts: platform-dependent and platform-independent.

The platform-dependent module implements Bluetooth communication functionality, it is responsible for the data transmission with the remote device via Bluetooth. The platform-independent part is responsible for the data processing and the visualization. This module uses Bluetooth module to receive ECG data, then determines QRS complexes of ECG, calculates the heart rate (HR), and visualizes ECG and HR on the screen.

The application developed with the use of Necessitas consists of two parts implemented in Java and C++ languages. Usually the Java part is used only to bootstrap C++ code, but it also can be used to host all the code, related to the communication via Bluetooth and the local socket publication. So, in our case, we implemented all functionality in the single application with no necessity to split it into two separate processes.

The architecture of our solution is depicted in fig. 2.

BtClient is an interface for all Bluetooth communication classes. It contains methods responsible for the data transmission and the remote device connection management: establishment, disconnection and status retrieval. According to the operating system used the platform independent module will use the platform-specific implementation of BtClient interface.

BluezBtClient class is a platform-specific implementation of BtClient interface for the GNU/Linux platform. It uses the libbluetooth library (Bluez API in C programming language) to create RFCOMM socket, connects to the remote Bluetooth device and retrieves data from it.

AndroidBtClient class is the implementation of BtClient interface for Android platform. To provide access to Bluetooth device, it uses SocketListener class, which manages the instances of BluetoothRetransmitter class. BluetoothRetransmitter is a component responsible for the direct communication with remote device. The class uses Android Java API to establish connection and transfer data with Bluetooth device.

The communication between AndroidBtClient class and SocketListener class is done via a Unix socket. The second class listens to the commands coming from the socket. There are two types of commands:

1) Establish a connection with the remote device;
2) Send data to the remote device.

First command consists of the MAC address of the Buletooth device. When request for connection comes, SocketListener creates an object of BluetoothRetransmitter class for the device and puts it into the pool. BluetoothRetransmitter object opens a Unix socket with the name coinciding with the passed address. When request for sending data comes, SocketListener passes data to the appropriate BluetoothRetransmitter object, which sends it to the device. When the data comes from the device, corresponding object writes it to the bound socket.

The communication protocol with the SocketListener class from the point of view of AndroidBtClient class includes the following steps:

1) Create a listening socket with the name coinciding with the MAC address of the Bluetooth device;
2) Send the address of the device to the listening socket the SocketListener class;
3) If the connection succeeded, all data from the device are received via the socket created at the first step.

V. Conclusion

The Android platform is very attractive for mHealth development because of its growing popularity. In this paper we discussed the peculiarities of cross-platform development of mHealth applications using Qt framework if Android is one of the target platforms. We highlighted problems with the implementation of Bluetooth communication using native code on Android platform, which complicates the use of Android NDK for mHealth application development. We proposed the solution for the problem.

Our approach allows to organize communication with remote Bluetooth devices from the native code on all devices running Android platform. This solution is applicable for all types of devices, because it uses the official Android Java API for Bluetooth communication and the local sockets API for interprocess communication. The approached was used and proved to be working during the porting of the mHealth application form Maemo platform to Android.

For now the implemented module allows only to initiate Bluetooth connection from native code. It does not supports the publication of Bluetooth service, which may be necessary to enable communication with other types of remote devices. Our future work will be the implementation of Bluetooth service publication.

Acknowledgment

The authors would like to thank Oleg Medvedev and Maxim Yatsovsky for assistance in formulation of the problems and also for counseling in some medicine and technical issues.

References

[1] "World Health Organization. Global Observatory for eHealth", http://www.who.int/goe/mobile_health/en/index.html, [Sep. 16, 2011]
[2] World Health Organization, "mHealth: New horizons for health through mobile technologies: Second global survey on eHealth.", Geneva: WHO, 2011, p. 1, http://www.who.int/entity/goe/publications/goe_mhealth_web.pdf
[3] "Necessitas.", Internet: http://sourceforge.net/p/necessitas/home/necessitas/, [Sep. 14, 2011]
[4] R. Latuske, "Bluetooth Health Device Profile (HDP)." *ARS Software GmbH*, p. 1.
[5] "Toshiba Bluetooth portal", Internet: http://aps2.toshiba-tro.de/bluetooth/index.php?page=toshiba/supported-pc-bt-prof iles
[6] "BlueCode+ Stollman E+V GmbH", http://www.stollmann.de/en/stacks/bluetooth/bluecode.html
[7] "Demo Connected Health", 9th Conference of Open Innovation Community FRUCT and First Regional Summit Russia–Finland, 2011, p. 180
[8] A. S. Huang, R. Rudolph, "Bluetooth essentials for programmers." *Cambridge University Press*, 2007, p. 179
[9] "App Compatibility.", Internet: http://android.ccpcreations.com/wiicontroller/wc-compatibility