LIMBus: a lightweight remote monitoring system powered by iOS and BLE

Luís Marques Silva
DETI, University of Aveiro
Campus Universitário de Santiago
Aveiro, Portugal
lmsilva@ua.pt

José Maria Fernandes
DETI & IEETA, University of Aveiro
Campus Universitário de Santiago
Aveiro, Portugal
jfernan@ua.pt

Ilídio Castro Oliveira
DETI & IEETA, University of Aveiro
Campus Universitário de Santiago
Aveiro, Portugal
ico@ua.pt

ABSTRACT
Mobile devices, including off-the-shelf smartphones, are being used as data gateways for nearby sensors in end-to-end integration of monitoring solutions, mostly based in the Android OS. In this paper, we present LIMBus, a lightweight monitoring system supported in iOS mobile devices to perform the role of data collector and gateway. LIMBus uses Bluetooth Low Energy to connect to external sensors and MQTT for mobile to backend streaming. The proposed system has been implemented and is demonstrated for an eHealth monitoring scenario.

Categories and Subject Descriptors
Embedded and cyber-physical systems; ubiquitous and mobile computing systems and tools; publish-subscribe/event-based architectures; Bluetooth.

Keywords
Message brokering, Cyber-Physical Systems, monitoring framework, BLE, iOS, MQTT.

1. INTRODUCTION
The use of mobile devices as gateways for data collected by sensors has been widely acknowledged, including in the health domain [1][2]. Most solutions resort to Android based smartphones and still rely on Classic Bluetooth for sensors data aggregator and sensor connectivity, respectively. There is, however, a large market penetration of iOS devices that already support Bluetooth Low Energy (BLE) [3]. This presents an opportunity to explore these devices as energy-efficient and BLE compliant data-gathering gateways.

With this in mind, we developed the LIMBus system, leveraging on the Bluetooth Low Energy (BLE) capabilities, which allows an iPhone or iPad to act as a gateway for data being acquired by nearby sensors, to be remotely visualized.

2. ARCHITECTURE

2.1 System overview
The main components of LIMBus are (1) a mobile data aggregator, (2) the MQTT broker, (3) backend services and (4) a web application (Figure **1**).
The mobile data aggregator is an application running on an iOS device; it collects data from sensors wirelessly, using BLE, and relays the incoming data to the backend server, which runs a MQTT messaging broker [4]. The data aggregator uses the BLE protocol features to discover nearby sensors.
The backend services, running on a remote server, receive the incoming data via the MQTT broker. The broker notifies a registered listener process whenever it receives new sensor configuration or new sensor data. The listener stores the received data in a database for long term storage. The web application accesses this long-term storage and provides interactive support for structured visualization of the data coming from one or more sensors.

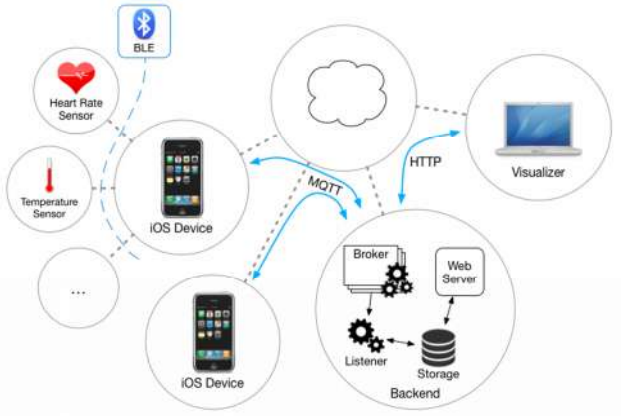

Figure 1 – LIMBus components overview.

2.2 Components interaction
Our architecture targets typical remote online monitoring use cases. The iOS application (LIMBus-Mob), acting as a daemon process, initializes the CoreBluetooth framework [5] as described in the Apple documentation [6]. It starts scanning for BLE nearby peripherals. When a peripheral is found, a connection is established to discover which services it provides. The information provided by the BLE peripheral is published by the application to the remote MQTT broker.

Afterwards, the LIMBus-Mob application waits for further (control) instructions from the backend to read data from one or more sensors. For this, LIMBus-Mob subscribes to a control topic on the MQTT broker from which it can receive commands: when an 'observe' command is raised in the backend for a given device, the LIMBus-Mob gets the message and asks the BLE framework to notify when the value of the corresponding characteristic is updated on the relevant sensor. When notifications are received, the new value is published to the MQTT broker.

3. SYSTEM IMPLEMENTATION

3.1 Sensors to aggregator BLE streams
In the BLE data model [3] each peripheral can have several services which, in turn, can have several characteristics. For instance, our test device (peripheral) provides two data streams: a

MOBIQUITOUS 2015, July 22-24, Coimbra, Portugal

DOI 10.4108/eai.22-7-2015.2260249

heart rate monitor and thermometer that, in BLE, are represented as two different services. In the specific case of the heart rate service, it allows retrieving the current heart rate measurement and the minimum and maximum within a time interval. These could be represented by 3 individual characteristics associated with the service. Each individual service and characteristic has its own unique identifier (UUID) associated. The service and characteristic UUIDs in BLE allow anyone to infer from any BLE peripheral the data semantics without knowing its inner-workings, based on standard BLE profiles.

In our system, after discovering the sensors, LIMBus-Mob announces them to the backend. The backend maps each sensor (i.e. peripheral+service+characteristic UUIDs) to a system wide identifier handle (currently an integer). This handle is kept locally and returned to LIMBus-Mob, allowing to map that stream to a single topic (e.g. "/users/<username>/data/<handle>") without needing to use the full peripheral+service+characteristic key, thus significantly reducing the traffic produced.

3.2 Aggregator to backend messaging

LIMBus makes use of the 'mosquito' [7] MQTT broker. Mosquito controls what can be read/written using Access Control Lists (ACL), so that an unauthorized user cannot access data belonging to another user. Different users contexts are mapped to different topics (e.g. "/users/<username>"). All communications for a specific user are done via subtopics.

The listener on the backend connects to the MQTT broker and subscribes to the appropriate topic (e.g. subscribing to "/users/#" allows to receive all messages published to subtopics of "/users"), to receive the incoming data and store them in a database (PostgreSQL). The messages include information about new devices, peripherals, services, characteristics and characteristics values. The data stored can be visualized in real-time on the web application (developed with Python/Django). The visualization environment also allows issuing commands to the iOS device (e.g. activating or deactivating specific sensor data streams).

4. E-HEALTH TESTBED

To demonstrate the application of the system, we use a simple eHealth monitoring setup (Figure 2), comprising physiological sensors included in the Cooking-Hacks e-Health Sensor Platform [9], in conjunction with the Nordic Semiconductor nRF51 DK mbed board [10] to enable the required BLE capabilities. In this setup, the sensors connect wirelessly to an iPad that relays the data to a (remote) laptop. We are able to stream the heart rate and body temperature of a subject to the backend and visualize it in real-time using the web application.

5. DISCUSSION

The LIMBus system is an in-house solution that successfully takes advantage of the energy-efficient BLE protocol to enable iOS mobile devices to act as aggregators for wireless sensors. The systems uses the Mosquito lightweight message brokering protocol for mobile to backend streaming, as an alternative to more verbose protocols. The use of well-defined protocols, such as BLE and MQTT, facilitates the integration of different sensors and the evolution of backend services, with little adaptation. The eHealth demonstrator shows that it is possible to remotely monitor patient physiologic signals using common iOS BLE-capable devices, low-cost sensors and lightweight messaging.

6. ACKNOWLEDGMENTS

This work was partially funded by FEDER through the COMPETE programe and by the Portuguese Government through the FCT - Foundation for Science and Technology, projects Incentivo/EEI/UI0127/2014 and UID/CEC/00127/2013, (IEETA/UA, www.ieeta.pt), CMUP-ERI/FIA/0031/2013, and PTDC/EEI-ELC/2760/2012.

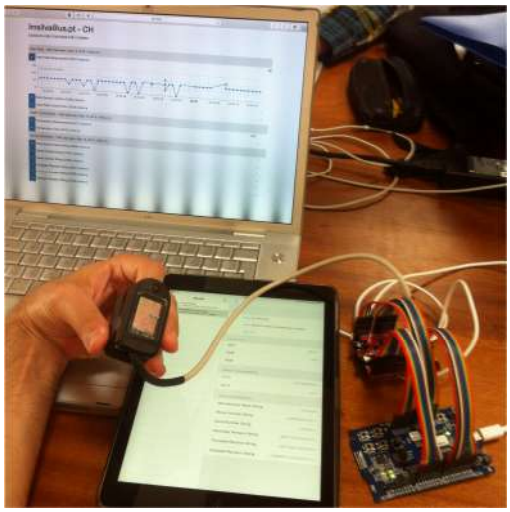

Figure 2 – Sample setup showing the heart rate being acquired and visualized in real-time on the web.

7. REFERENCES

[1] M. J. Morón, R. Luque, and E. Casilari, "On the capability of smartphones to perform as communication gateways in medical wireless personal area networks.," Sensors (Basel)., vol. 14, no. 1, pp. 575–594, 2014.

[2] A. Ghose, C. Bhaumik, D. Das, and A. K. Agrawal, "Mobile healthcare infrastructure for home and small clinic," Proc. 2nd ACM Int. Work. Pervasive Wirel. Healthc. - MobileHealth '12, vol. 2, p. 15, 2012.

[3] Bluetooth Core Specification Version 4.2. [Online]. Available: https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=282159.

[4] A. Stanford-Clark and A. Nipper, "MQ Telemetry Transport," *MQ Telemetry Transport*, 2013. [Online]. Available: http://mqtt.org/.

[5] Core Bluetooth Framework Reference. [Online]. Available: https://developer.apple.com/library/ios/documentation/CoreBluetooth/Reference/CoreBluetooth_Framework/.

[6] Core Bluetooth Programming Guide. [Online]. Available: https://developer.apple.com/library/ios/documentation/NetworkingInternetWeb/Conceptual/CoreBluetooth_concepts/AboutCoreBluetooth/Introduction.html.

[7] Mosquitto Open Source MQTT v3.1/v3.1.1 Broker. [Online]. Available: http://www.mosquitto.org.

[8] P. Leach, M. Mealling, and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace," *Internet Soc.*, vol. RFC 4122, pp. 1–32, 2005.

[9] Cooking Hacks e-Health Sensor Shield. [Online]. Available: http://www.cooking-hacks.com/ehealth-sensors-complete-kit-biometric-medical-arduino-raspberry-pi.

[10] Nordic Semiconductor nRF51 DK. [Online]. Available: https://www.nordicsemi.com/eng/Products/nRF51-DK/.