Middleware Architecture for Health Sensors Interoperability

Nawras Georgi, Aline Corvol, Regine Le Bouquin Jeannes

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 19, 2018, accepted April 29, 2018, date of publication May 14, 2018, date of current version June 5, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2835644

Middleware Architecture for Health Sensors Interoperability

NAWRAS GEORGI[1,2], ALINE CORVOL[3,4], AND RÉGINE LE BOUQUIN JEANNÈS[1], (Member, IEEE)
[1]Univ Rennes, Inserm, LTSI–UMR_S 1099, F-35000 Rennes, France
[2]AZNetwork, 61000 Alençon, France
[3]Service de gériatrie, CHU Rennes, 35000 Rennes, France
[4]Univ Rennes, CNRS, ARENES–UMR 6051, F-35000 Rennes, France

Corresponding author: Nawras Georgi (nawras.georgi@univ-rennes1.fr)

This work was supported by ANRT under Contract 915/2015.

ABSTRACT Quantified-self is a growing movement that stimulates users to monitor their health status for a better prevention of illnesses using a set of sensors. For reliable monitoring, it should be easy to retrieve sensors data despite the implemented communication protocol, standard or proprietary. In this contribution, we propose a middleware that can be used by software developers to retrieve data from sensors without worrying about the communication protocol implemented by those sensors. This middleware takes into consideration data category, that is, wellness or medical. It also handles sensors reliability and legal framework. Six sensors, which implement one standard protocol and two proprietary ones, have been integrated to the middleware. They offer to measure 24 different data types, such as weight, blood pressure, or heart rate. The middleware manages also users' privacy.

INDEX TERMS Biomedical monitoring, e-Health, interoperability, Internet of Things, wearable sensors.

I. INTRODUCTION

The number of connected sensors, also known as Internet of Things (IoT), is increasing exponentially [1] with projection of reaching 24 billion devices by 2020. In healthcare, the IoT market is expected to reach $409.9 billion by 2022 [2]. This expansion provides e-health and telecare sectors with the necessary tools to get developed. As a matter of fact, this market value may increase by $21.9 billion between 2014 and 2020 [3]. A recent movement, named the 4P medicine (Predictive, Personalized, Preventive and Participatory), has appeared [4]. It focuses on providing personalized health prevention instead of treating diseases [5] which can be done thanks to quantified-self. The latter consists in engaging any individual in the self-tracking of any kind of biological, physical, behavioral or environmental information [6]. Using a set of sensors, users can monitor themselves in order to improve their health. On the one hand, this quantification helps them improving their lifestyle, and, on the other hand, by sharing their health data with their practitioners, remotely and in real-time, they participate in improving their treatment [7]. However, such follow-up cannot be done efficiently using only sensors implementing the ISO/IEEE 11073 protocol [8], also known as Continua, which is the standard protocol for personal health devices and which is promoted by the Personal Connected Health Alliance (PCHA). The number of certified devices indeed kept decreasing over the last 8 years, from 10 devices certified in 2009 to only 1 device implementing the standard protocol in 2016 [9]. A recent survey [10] has shown that out of 92 health sensors, 86 implement proprietary protocols. Consequently, sensors with proprietary protocols should be considered for an extensive health monitoring.

In parallel, a new business model based on data has emerged. Thus, many sensors manufacturers privilege their own proprietary protocols to control their users' data. However, reliable health monitoring cannot be limited to one sensor. Therefore, interoperability becomes a crucial issue to exchange data from multiple types of sensors to provide better health monitoring.

In order to encourage interoperability in health systems, we propose a middleware, in the form of a software library, able to communicate with a set of sensors, using both standard and proprietary protocols, with the objective to access any data. This middleware is meant to be used by third-party developers, healthcare service applications or providers to retrieve patients' data, enabling them to ensure

TABLE 1. A comparison between the proposed middleware and related works middlewares.

Reference	Market-Ready Sensors	Standard Protocols	Proprietary Protocols	Runs on Smartphones	Handles Users' Privacy	Handles Sensors Reliability	Middleware
Georgi 2017 [9]	✓	✓	✓	✓			
Hofer 2015 [11]		✓		✓	✓		✓
Ramírez-Ramírez 2015 [12]		✓					✓
Mihaylov 2015 [13]		✓					✓
Catarinucci 2015 [14]					✓		✓
Rahmani 2015 [15]		✓				✓	✓
Gay 2015 [16]	✓	✓	✓	✓			
Proposed middleware	✓	✓	✓	✓	✓	✓	✓

interoperability with a maximum number of health sensors with minimal effort. By using such a library, developers do not need to implement each protocol independently and can focus on data analysis and processing.

In this contribution, we first present a state-of-the-art in Section II. Then, in Section III, we tackle the technical, privacy and legal specifications that are considered, as well as the architecture of our system and its operating mode. We present our results in Section IV and discuss them in Section V.

II. RELATED WORKS

Many studies have been done to improve interoperability between health sensors [11]–[15], [17].

Hofer *et al.* [11] proposed an interoperable health system for monitoring obstructive pulmonary diseases named COMPASS. These authors developed a server-client RESTful architecture with a publish/subscribe mechanism running on Android devices with the aim of increasing the quality of care by collecting and analyzing sensors data in a long-term monitoring context. Collected data were sent to a remote server for storage and later processing. The authors used a prototype multi-sensor medical device from Biovotion, measuring several vital signs such as blood oxygenation, heart rate, skin temperature and steps count. Data were retrieved from the sensor using ISO/IEEE 11073 protocol and sent remotely using HL7 standard protocol [18]. However, despite the increasing number of commercialized health sensors, especially those using proprietary protocols, COMPASS does not support any of these sensors. Furthermore, the only device integrated to the platform is a prototype that is not commercialized, and thus not publicly available. Ramírez-Ramírez *et al.* [12] developed a hardware middleware device that handles data from a sensor and sends them to the cloud. The sensor is a prototype that measures body temperature and integrates a fall detector, and uses ISO/IEEE 11073 to communicate. Once retrieved by the middleware, these data are then transmitted to a remote server using HL7 protocol. Similarly to Hofer's work, Ramírez-Ramírez middleware does not implement any wide public sensor, neither standard nor proprietary. Mihaylov *et al.* [13] proposed a cloud-based platform for supporting elderly e-health services so that they can live at home much longer. It aimed at detecting abnormal behaviors in elderly lifestyle that might indicate a change in their health status. To achieve this goal, the middleware retrieved data from numerous sensors like pulse oximeters before transmitting them to a cloud-based server for a later processing. Implemented sensors used only the standard protocol ISO/IEEE 11073 and remote transmission was done using HL7 protocol. Catarinucci *et al.* [14] proposed an architecture for a monitoring system within hospitals, consisting in a wireless sensors network that implements a REST interface for data communication. It was able to communicate with RFID-based sensors. Due to its complex architecture, this type of solution is not adapted for personal health and home-based telemonitoring systems. Furthermore, it does not support market-ready sensors. Rahmani *et al.* [15] presented the concept of a smart e-health gateway that served as a bridge for medical sensors between the home and the hospital. It runs on a Pandaboard device running Ubuntu operating system. Similarly to previous works, commercialized sensors were not integrated.

Table 1 summarizes the characteristics of the current related studies. First of all, none of them supports market-ready health sensors, but only research devices. Most of these works are based on standard protocols like ISO/IEEE 11073 or HL7. Although HL7 is widely used, especially in hospitals, among commercialized personal health sensors, very few use the Continua standard to communicate whereas the number of devices implementing proprietary protocols is growing continuously. As a matter of fact, only 7% of personal health sensors implement the ISO/IEEE 11073 protocol [10]. Several studies demonstrated that it is possible to implement both standard and proprietary protocols [9], [16]. Georgi and Le Bouquin Jeannès [9] developed an Android application that retrieves data from sensors implementing the standard protocol ISO/IEEE 11073 such as weight scale and blood pressure monitor and from others implementing proprietary protocols (iHealth blood pressure monitor and Microsoft Band actimeter). Gay and Leijdekkers [16] also developed an Android application that integrates, besides health sensors, cloud services for managing data like Google Fit and Microsoft HealthVault. However, in these two studies, developers implemented each protocol separately in an application, which is time and cost consuming whereas it is possible to aggregate all these protocols into one middleware that can be reused and shared with other developers to encourage interoperability. Moreover, as we detail below, privacy and reliability are very important in healthcare. Despite this, few works took them into consideration.

III. METHODS AND MATERIALS

In this contribution, we propose a middleware integrating features mentioned in Table 1. We tackle users' privacy, in terms of data category (wellness or medical), as well as how data are retrieved (directly from the sensor or through cloud) and sensors reliability. This middleware can retrieve health data (such as blood pressure or weight) without worrying about which communication protocol sensors implement.

A. MIDDLEWARE SPECIFICATIONS

We first mention the middleware specifications, which can be divided into three categories: legal, privacy and technical.

1) LEGAL SPECIFICATIONS

In healthcare, sensors can be split into two categories: medical and wellness devices. In the European Union, medical devices are those certified by the European directive 93/42/CEE which aims to *"ensure a high level of protection for human health and safety, smooth operation of the single market and to achieve the results for which the devices are intended to"*. This directive defines four categories of medical devices (I, IIa, IIb, III in criticality order). Regarding wellness sensors, according to the Center for Devices and Radiological Health (CDRH), they are intended for wellness use with a very low risk to users' safety.

In France, any data retrieved from a medical device is considered as medical data. Therefore, only companies with health-data-hosting authorization can store them. This authorization ensures a high level of security, ethical, technical and economical requirements to secure medical data [19]. An administration linked with the French ministry of health, ASIP Santé (Agence des Systèmes d'Information Partagés de Santé), is in charge of the authorization process.

In order to identify whether data need special hosting, the middleware informs its users if measuring data is done with a medical device. In this case, they should subscribe to a medical data hosting service. It is important to mention that the middleware is only aimed at retrieving data from sensors to send them back to users. Therefore, it is their responsibility to get the final user consent to retrieve these data, to store them for later processing and, if necessary, to transmit them securely.

2) PRIVACY SPECIFICATIONS

Retrieving data from sensors can be done in two ways: directly from the sensor or from the manufacturer's cloud. In the latter case, the middleware users are no longer able to control who is accessing the data and hence to protect their privacy. As a matter of fact, a recent study examined four sensors manufacturers' privacy policies and concluded that *"... all of the services collect more data than necessary, and none properly specify who they are sharing user data with."* [20].

In order to ensure privacy, the middleware can apply a filter on the sensors list to consider according to the application requirements. Thus, if the final user decides to fully protect his data, developers should inform the middleware to ignore all cloud-based sensors, and to use only direct-communication sensors. Using this strategy, the user stays in control of his data since they are not transmitted to third-party clouds.

3) TECHNICAL SPECIFICATIONS

The middleware is able to handle both online and offline data, when this option is available. In fact, users can perform measurements without having their sensors connected to their smartphones. For instance, some blood pressure monitors can be used without smartphones since the measure is directly displayed on the device screen. In this case, measures are saved on the device itself until it is synchronized with the smartphone later on. The smartphone downloads these data from the sensor either to keep a history of the user's measures or for processing. Therefore, any available measure saved on the sensor can be retrieved when the middleware is connected to it.

Another point to consider is data reliability, since some sensors are more reliable than others. Consequently, when two or more sensors are able to measure the same type of data, the middleware starts by determining which is the most reliable one as detailed in section III-B2.

B. MIDDLEWARE ARCHITECTURE

1) OVERVIEW

Users request data from the middleware and the latter handles the communication with the available sensors to retrieve the requested data.

The increasing number of proprietary protocols implemented by health sensors is causing a critical heterogeneity. Therefore, the middleware has been designed to handle data type. Thus, when its users want to retrieve measures, they have to ask the middleware for a specific data type and not for a sensor model. For example, they can request a measure of heart rate but not a measure from Microsoft Band. This is a deliberate choice since, when monitoring health status, the data themselves must prevail. When they need to use a sensor from a specific manufacturer, developers should use the SDK or the API (Application Programming Interface) provided by this manufacturer.

2) OPERATING MODE

In this section, we detail the operating mechanism of the middleware as displayed in the flowchart of Fig. 1.

When a measure is requested, the middleware starts by determining the list of sensors able to perform this measure (Fig. 1 - ①). This list is retrieved from the middleware database. Then, users can ask the middleware to apply two filters on the sensors list. The first one deals with users' privacy, and removes from the list any sensor that sends its data to the manufacturer. Therefore, only sensors allowing direct communication are used. The second filter limits measures to

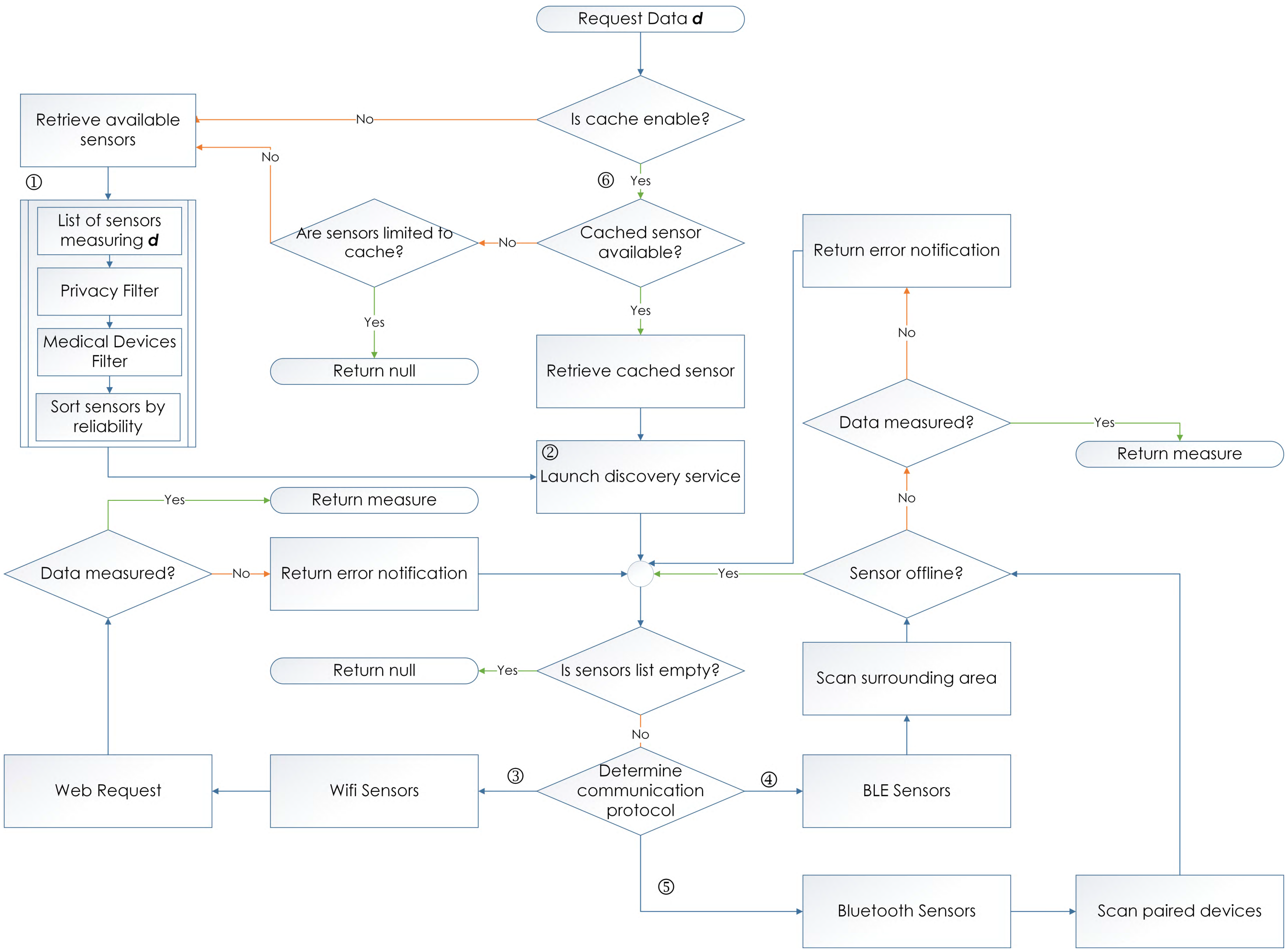

FIGURE 1. A flowchart describing how the middleware proceeds to measure data from sensors.

either medical devices or wellness devices. Of course, if users want to use any type of sensors, this filter should not be applied. If the middleware is used for a medical development goal, it can measure data with medical devices only. Similarly, if developers do not want to use a medical hosting offer to store users' medical data for economic reasons, the middleware is able to measure data from wellness devices only.

Sensors are retrieved in line with the used hardware. For instance, if the hardware running the middleware does not support Bluetooth Low Energy (BLE), sensors using it are ignored. Similarly, if an Internet connection is unavailable at the time of current measurement, cloud-based sensors are not used.

Once established, this list is sorted in terms of reliability. As a matter of fact, all sensors do not present the same level of confidence, some types being more reliable than others. Therefore, they are sorted so that the most accurate one is ranked the highest and accessed first. Sensors are first ranked regarding their types. For instance, upper arm blood pressure monitors are more reliable than wrist blood pressure ones [21] since the latter overestimate the blood pressure measure [22]. Hence, the middleware sorts upper-arm blood pressure monitors first. If no study allows identifying such rules, then this step is skipped. Thereafter, sensors are sorted based on their technical characteristics, provided by devices manufacturers. When sensors are equally reliable, the one allowing a direct communication is privileged. If several sensors allow it, the one that does not require any interaction with the user is privileged. If this case is met for more than one sensor, the sensors are sorted randomly.

Another aspect that is taken into consideration is continuous measurement. Indeed, it can be important to monitor a health data type over a period of time. When users request a continuous measure, the middleware retrieves only sensors able to perform such measure, then sorting sensors with regard to their accuracy is applied as mentioned above. For example, a heart rate can be measured continuously with a pulse oximeter but not with a blood pressure monitor. Thereafter, a discovery service is launched to retrieve data from the established sensors list. The service iterates this list until data is measured from one of the sensors (Fig. 1 - ②). The most accurate sensor is used first. It starts by determining the appropriate SDK or API to use. Based on the communication technology used by sensors, we distinguish three processes:

- If the sensor communicates *via* Wi-Fi (Fig. 1 - ③), a web request is performed to retrieve data. This request is intended either to the sensor or to its manufacturer's cloud.
- If the sensor uses BLE (Fig. 1 - ④), the middleware scans the surrounding area looking for sensors.
- Finally, if the sensor uses a classic Bluetooth connection (Fig. 1 - ⑤), the middleware searches for it among paired devices.

Once a connection is established, the measure is performed and data is returned to the final user. If the connection to the sensor has been established but no measurement can be obtained, due to a sensor dysfunctioning for example, an error notification is returned and the middleware proceeds with the next sensor existing on the list established by the discovery service. It is necessary to inform users that a measure could not be retrieved despite an established connection because the sensor might be damaged. In the case where sensors need manual activation, the middleware notifies developers that the user needs to perform some action on the sensor. In this case, developers should ask the user to activate the sensor using the application User Interface (UI). If no sensor is found to perform the measure, a null object is returned.

When a cloud-based sensor is used for the first time, a notification is sent to the end-user informing him that his data are handled by a third-party provider. This choice is made so he can be fully aware of his privacy.

To optimize the middleware performance, a cache function is available (Fig. 1 - ⑥). It stores the used sensors to improve the middleware performance. If it is enabled, the middleware tries to re-establish a connection to the last used sensor before launching the discovery service. If it fails, the sensors list is retrieved as mentioned above. If the healthcare application requirements are limited, the requested sensors can be preloaded in the cache. The discovery service can also be limited to only sensors available in the middleware cache. Any interaction between the application and the middleware is done asynchronously, preventing the middleware from freezing the application user interface.

The middleware has a prefilled database with all necessary information for its functioning. It includes, amongst others, sensors types, models, their communication protocols and the list of data types they can measure.

3) COMPONENTS

The middleware is composed of a number of components as illustrated in Fig. 2. It is designed as a software library, therefore intended to be embedded in third-party applications. The communication service component is the communication interface between the middleware and other applications. When the middleware users request data, as detailed in Section III-B2, it is done through this component, and when the measure is realized, the communication service returns the result to users.

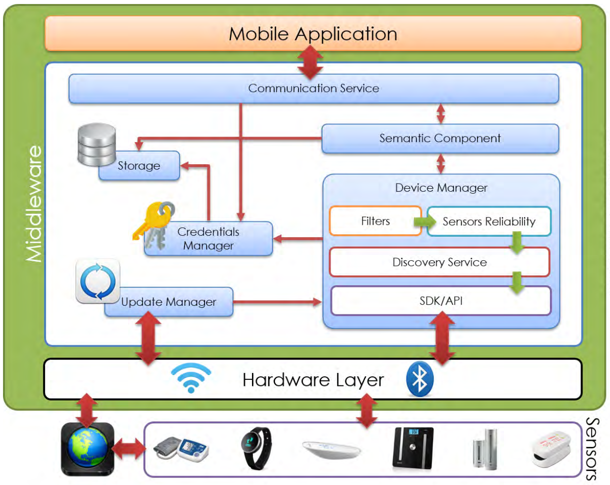

FIGURE 2. A diagram illustrating the middleware components.

As mentioned in III-B1, when data are requested, they are requested as a data type. Hence, the middleware needs to determine which sensor can be used to perform this measure. The semantic component handles this point and transmits a list of possible sensors to the device manager. The latter manages the measurement process. It starts by applying the filters described above, sorting sensors based on their reliability then launching the discovery service in order to determine which sensor the final user handles. It also identifies which communication protocol the sensor implements to decide which SDK or API to enable. Since some sensors require an identified user, the device manager communicates with a credentials manager to get the login ID that should be used. The credentials manager stores credentials provided by the application developer. The communication between the device manager component and sensors is done through the hardware layer of the operating system running the middleware.

The middleware has also an update manager component in charge of updating sensors firmware. Regarding updating the middleware, to add new sensors for instance, developers should upgrade it in their development project, as they would update any software library used as a dependency.

4) OUTPUT RESULT

If no connection can be established with a sensor for measuring the requested data, a null object is returned. Otherwise, a measure object is returned containing the following data related to the measure:

- Type: the requested data type (weight, blood pressure...).
- Value: the requested data value.
- Unit: the requested data unit (kg for weight, mmHg for blood pressure).
- Timestamp: the requested data measure time, enabling the healthcare application to know when the measurement was done and therefore if it is a live measurement or not. If the sensor gives the measure time, it is

TABLE 2. Sensors supported by the middleware.

				Communication		
Sensor N°	Manufacturer	Type	Model	Technology	Protocol	Category
1	A&D	Weight Scale	UC-355PBT-Ci	Bluetooth	Standard	Wellness
2	A&D	Blood Pressure Monitor	UA-767PBT-Ci	Bluetooth	Standard	Medical
3	iHealth	Blood Pressure Monitor	BP7	Bluetooth	Proprietary	Medical
4	iHealth	Oximeter	PO3M	Bluetooth	Proprietary	Medical
5	iHealth	Actimeter	AM3S	Bluetooth	Proprietary	Wellness
6	Netatmo	Weather Station	Weather Station	Wi-Fi	Proprietary	Wellness

TABLE 3. Data measured through middleware on two devices: Samsung Galaxy Tab S2 (Android 6.0.1) and Google Nexus 7 (Android 5.1.1). Green = successful measures, Red = failed measures. Sensors column refers to Table 2.

Data Type	Data Unit	Sensors (cf. Table 2)	Galaxy Tab S2 (with Internet)	Galaxy Tab S2 (without Internet)	Nexus 7 (without BLE)
Weight	Kg	1	🟩	🟩	🟩
Systolic Blood Pressure	mmHg	2, 3	🟩	🟩	🟩
Diastolic Blood Pressure	mmHg	2, 3	🟩	🟩	🟩
Heart Rate	Beats Per Minute	3, 4	🟩	🟩	🟩
Blood Oxygen Saturation	%	4	🟩	🟩	🟥
Blood Pressure Pulse Wave	-	3	🟩	🟩	🟥
Arrhythmia	Boolean	3	🟩	🟩	🟥
Hemodynamic Stability Diagnosis	Boolean	3	🟩	🟩	🟥
Steps Number	Steps	5	🟩	🟩	🟥
Calories	kCalories	5	🟩	🟩	🟥
Total Calories (with BMR)	kCalories	5	🟩	🟩	🟥
Sleep State	-	5	🟩	🟩	🟥
Noise	dB	6	🟩	🟥	🟩
CO2	Parts Per Million	6	🟩	🟥	🟩
Pressure	mBar	6	🟩	🟥	🟩
Humidity	%	6	🟩	🟥	🟩
Temperature	Celsius	6	🟩	🟥	🟩
Wind Angle	Degree	6	🟩	🟥	🟩
Wind Strength	Km/h	6	🟩	🟥	🟩
Gust Angle	Degree	6	🟩	🟥	🟩
Gust Strength	Km/h	6	🟩	🟥	🟩
Current Rain Quantity	mm	6	🟩	🟥	🟩
Rain Quantity over last 1 hour	mm	6	🟩	🟥	🟩
Rain Quantity over last 24 hours	mm	6	🟩	🟥	🟩

used; otherwise, the current time is used. The Network Time Protocol (NTP) is used to ensure that real time is used.

- Resolution: the sensor accuracy as indicated by the manufacturer in the sensor data sheet.
- Privacy: this field is a Boolean. It allows users to know if data have been retrieved directly from the sensor (true) or from the manufacturer's cloud (false).
- Medical data: this field is a Boolean. It tells users if the retrieved measure is a medical data and therefore if it requires an authorized company to host it.

The users of the middleware can request an instance of the sensor type used for the measurement. In this case, they can get the following data (when applicable):

- Sensor type (weight scale, oximeter…).
- Sensor model.
- Battery level.

If a continuous measure is requested or if the measure is an array of single values, then a measure result is sent back to users each time a new value is available. The measure time can be predetermined at launch, or stopped manually by users when they estimate that it is time to.

IV. RESULTS

We developed this middleware for the Android platform. It supports OS versions starting API 19 (KitKat 4.4), since older versions are no longer supported [11], [23]. The middleware is programmed using Java language, and is interoperable with the six sensors described in Table 2. With these sensors, healthcare application can retrieve up to 24 data, described in Table 3. To test our middleware, we implemented an Android application integrating this middleware to retrieve health measures, as would other developers do for their developments. This application stores measured data in a local database and then displays them on a chart.

In a recent work [9], we used iHealth API to retrieve data. Since then, iHealth has published an SDK enabling direct communication with its sensors. This has the benefit of allowing direct control on sensors such as blood pressure monitors, actimeters or oximeters. In fact, when we used the API to retrieve data, we could not measure live data neither control the sensors (e.g. interrupt the measure, modify the sensor settings). This lack forces the user to use two applications, the manufacturer's one and the healthcare service one, thereby reducing the application ease of use. Now with the available SDK, we can directly control the sensors and therefore measure live data. Regarding A&D sensors, they implement the ISO/IEEE 11073 standard protocol which is natively supported by Android since API 14 through BluetoothHealth class. To retrieve data from Netatmo sensor, we implemented its API to fetch measures from the manufacturer's cloud. Therefore, an Internet connection is required.

Unlike A&D, both iHealth and Netatmo sensors require developer credentials to communicate their data. Netatmo requires in addition the credentials of the final user who should provide them through the developer's application. Developers can communicate these data to the middleware credentials manager that uses SharedPreferences. It is a key-value interface for storing data in a given context. Since the context used by the middleware is supplied by the developers' application, both sides can access the same data. The middleware cache is also based on the SharedPreferences interface. The database used by the middleware is SQLite, which is also supported natively by Android.

Asynchronous communications are done using IntentServices. These components are designed to perform long operation in a worker thread. They have the benefit of stopping themselves once the requested task is done allowing optimizing the use of the platform resources. The communication between different components of the middleware is done using the ResultReceiver interface.

Since API 23 (Android Marshmallow 6.0), users grant permissions to an application at runtime rather than at installation time. In order to respect users' privacy, the middleware only requests necessary permissions for the required SDK instead of requesting all permissions at once.

Table 4 details the number of lines of code a developer needs to write when using our middleware to implement a specific protocol compared to a self implementation of the protocol. The middleware has an average cyclomatic complexity of 1.92 and a total cyclomatic complexity of 435. Its static memory footprint (compiled bytecode) is 778.8 KB.

TABLE 4. Implementation effort when using the middleware versus without it.

Protocol	Middleware (LoC)	Java (LoC)	Effort Ratio
ISO/IEEE 11073	32	717	4.46%
iHealth	39	158	24.68%
Netatmo	41	733	5.59%
The 3 protocols	44	1608	2.74%

This application was tested on two devices: a Samsung Galaxy Tab S2 running Android 6.0.1 that was tested with and without Internet connection, and a Google Nexus 7 running Android 5.1.1 with an Internet connection. This configuration made it possible to test the middleware adaptation to the chosen hardware. The 24 supported data types were measured. Table 3 shows the result of each measure. Green color means data measure was successfully performed, red color when it fails which is due either to a non-adapted hardware (Nexus 7 does not support BLE), or to the absence of an Internet connection (Galaxy S2) required by cloud sensors. Sensors column makes reference to Table 2. Regarding Fig. 3, it presents an example of heart rate measures retrieved using the middleware.

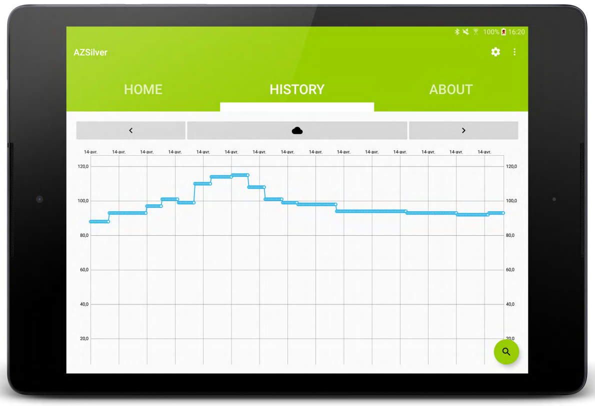

FIGURE 3. A line chart displaying heart rate measures.

V. DISCUSSION

We developed the middleware for Android because this platform represents 84.82% of smartphones [24]. Fig. 4 illustrates the evolution of mobile platforms among smartphones since 2009. A recent survey [10] showed that out of 92 health sensors, 96% of them use Bluetooth and/or Wi-Fi to communicate their data. Hence, we choose to focus on implementing these two wireless technologies. Other communication technologies could also be used, but they are not as widely adopted as Bluetooth and Wi-Fi. For instance, infrared port is supported by very few sensors, just as ANT wireless network technology.

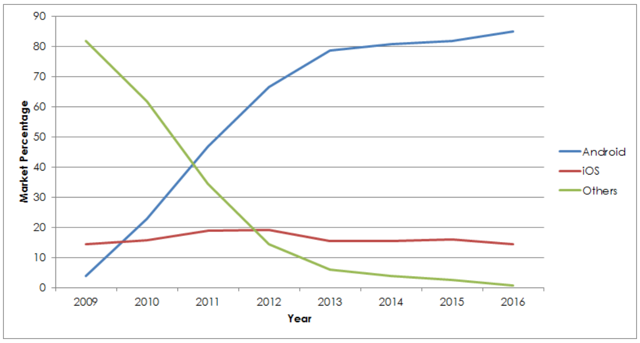

FIGURE 4. An illustration of the progress of Android platform compared with others.

Nexus 7 is a device that does not support BLE, therefore it cannot retrieve data measured by BLE sensors. Therefore, failed measurements in Table 3 are due to the inability of the Nexus 7 device to communicate over BLE. Regarding the Samsung device, since it does not have an Internet connection, it could not retrieve data from the Netatmo weather station which is a cloud-based sensor (cf. Table 2 - Sensor 6). Currently, the middleware does not retrieve any data from services like Google Fit or Apple Health because no explicit

```
CommunicationService nManager = new CommunicationService(this, mHandler);
nManager.requestMeasure(CommunicationService.TYPE_TEMPERATURE);
```

Listing 1. Example of data request code using the middleware.

```
private Handler mHandler = new Handler() {
  @Override
  public void handleMessage(Message msg) {
    switch (msg.what) {
      case SensorsManager.MEASURE_OK:
        Measure nMeasure = Parcels.unwrap(((Bundle) msg.obj).getParcelable(Measure.MEASURE));
        Log.d(App.TAG, ”Final result: ” + nMeasure.getValue() + nMeasure.getUnit());
        break;
    }
  }
};
```

Listing 2. Example of data reading code using the middleware.

data are available about the saved measures such as their category (medical or wellness), the sensor used to perform the measure, its accuracy, etc.

The developed application based on the middleware allows the user to retrieve his health data and to store them locally maintaining privacy. It offers an alternative solution to commercial applications.

Regarding the implementation effort, the middleware spares developers from implementing by themselves each protocol. The benefit increases with the rising of the number of implemented protocols as detailed in Table 4. As a matter of fact, when using the middleware, only 2.74% of the code amount is needed compared to the non-use of the middleware. Even if this is implementation-specific, it gives an order of magnitude of the implementation effort. Listing 1 shows the necessary code to request a specific type of data whereas Listing 2 shows how data are read. This benefit is even more significant as smartphone applications size keeps increasing. We analyzed 250 applications obtained from Google Play through a search with the keyword *“health monitoring”*. Eighteen applications were excluded because they were outside the scope (servers and battery monitoring). Among the rest of the applications, we considered only those having more than 10 000 downloads resulting in 115 applications among 232. Forty-five percent of these applications communicate with at least one health sensor (weight scale, blood pressure monitor, etc.). The average size of these applications is 33.51 MB. This means that the middleware represents 2.3% of an application size which is not significant regarding the saved implementation effort.

Thanks to the small number of implemented sensors, the middleware retrieves requested data quickly. However, with the increasing number of supported devices, the searching time to find the appropriate sensor might become significant. For this reason, we implemented the cache function. The used sensors are stored in the middleware memory and can be reused saving search time. Thus, developers are encouraged to preload in the cache the sensors their users are likely to use when possible. Currently, it needs on average 0.045 second to determine the appropriate sensor to use. The time to obtain sensor data depends on the need of the user’s intervention to activate the sensor. As a matter of fact, sensors n°1 (A&D weight scale), n°2 (A&D blood pressure monitor) and n°4 (iHealth oximeter) need to be activated manually by the user in order to retrieve the measures. For sensor n°3 (iHealth blood pressure monitor), the middleware needs around 37 seconds to perform the measure. This delay is normal and required for the device cuff to inflate and deflate in order to determine the user’s blood pressure. For sensor n°5 (iHealth actimeter), it needs 5.6 seconds to establish a connection and to retrieve current data, and 0.412 second to retrieve measures from sensor n°6 (Netatmo weather station).

We also used several software analysis tools to ensure a high quality code of the middleware. This enabled us to maintain it and upgrade it easily in the future. First, we used the Checkstyle tool. This ensures for developers willing to use it that the middleware code respects coding standards and conventions and, therefore, is easily understandable. PMD software was also used to provide a clean and readable code. The Android Studio default tool, Lint, was also used to look for optimizations and identify potential errors.

In this work, we developed a middleware, as a software library, that improves interoperability with health sensors implementing both standard and proprietary protocols. It allows developers to communicate with a greater diversity of sensors and to manipulate data without worrying about which communication protocol to use in order to achieve a better health monitoring. It is also able to take account of the used material characteristics, legal requirements and users’ privacy.

Currently, sensors are sorted based on the technical characteristics provided by their manufacturers and based on their type. However, several studies in the literature have already revealed that the reliability varies with sensors. As an example, a recent study showed that a Fitbit Charge HR was less accurate than a Polar H7 for measuring heart rate [25]. In a future work, we propose to test the veracity of the sensors accuracy regarding the technical characteristic provided by the manufacturers. We also plan to increase the number of interoperable sensors supported by the middleware on the one hand, and to make it compatible with other platforms than Android on the other hand.

ACKNOWLEDGMENT

The authors would like to thank Pr. Dominique Somme and Pr. François Taïani for their fruitful discussions, and Julien Modolo for comments.

REFERENCES

[1] Business Insider Australia. (Feb. 2014). *The Internet of Everything: 2014*. Accessed: May 23, 2017. [Online]. Available: http://www.businessinsider.fr/us/the-internet-of-everything-2014-slide-deck-sai-2014-2

[2] *Internet of Things (IoT) in Healthcare Market Report 2022*, Grand View Res., San Francisco, CA, USA, May 2016.

[3] 2016. *Five Telemedicine Trends Transforming Health Care*. Accessed: Nov. 2015. [Online]. Available: https://www.foley.com/five-telemedicine-trends-transforming-health-care-in-2016/

[4] L. Hood, "Systems biology and P4 medicine: Past, present, and future," *Rambam Maimonides Med. J.*, vol. 4, no. 2, p. e0012, Apr. 2013. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3678833/

[5] L. Hood and M. Flores, "A personal view on systems medicine and the emergence of proactive P4 medicine: Predictive, preventive, personalized and participatory," *New Biotechnol.*, vol. 29, no. 6, pp. 613–624, Sep. 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1871678412000477

[6] M. Swan, "The quantified self: Fundamental disruption in big data science and biological discovery," *Big Data*, vol. 1, no. 2, pp. 85–99, Jun. 2013. [Online]. Available: http://online.liebertpub.com/doi/abs/10.1089/big.2012.0002

[7] G. Appelboom, M. LoPresti, J.-Y. Reginster, E. S. Connolly, and E. P. L. Dumont, "The quantified patient: A patient participatory culture," *Current Med. Res. Opinion*, vol. 30, no. 12, pp. 2585–2587, Dec. 2014. [Online]. Available: http://dx.doi.org/10.1185/03007995.2014.954032

[8] L. Schmitt, T. Falck, F. Wartena, and D. Simons, "Novel ISO/IEEE 11073 standards for personal telehealth systems interoperability," in *Proc. Joint Workshop High Confidence Med. Devices Softw. Syst. Med. Device Plug-and-Play Interoperability (HCMDSS-MDPnP)*, Jun. 2007, pp. 146–148.

[9] N. Georgi and R. Le Bouquin Jeannès, "Wellness sensors and proprietary protocols, a solution for health monitoring?" in *Proc. IEEE EMBS Int. Conf. Biomed. Health Inf. (BHI)*, Feb. 2017, pp. 301–304.

[10] N. Georgi and R. Le Bouquin Jeannès, "Proposal of a health monitoring system for continuous care," in *Proc. 4th Int. Conf. Adv. Biomed. Eng. (ICABME)*, Oct. 2017, pp. 1–4.

[11] T. Hofer, M. Schumacher, and S. Bromuri, "COMPASS: An interoperable personal health system to monitor and compress signals in chronic obstructive pulmonary disease," in *Proc. 9th Int. Conf. Pervasive Comput. Technol. Healthcare (PervasiveHealth)*, May 2015, pp. 304–311.

[12] R. Ramırez-Ramırez, M. Cosıo-Leon, D. Ojeda-Carreno, M. Vazquez-Briseno, and J. I. Nieto-Hipolito, "Designing a gateway IEEE1451-HL7 for E-health telemonitoring services," in *Proc. Int. Conf. Comput. Syst. Telematics (ICCSAT)*, Oct. 2015, pp. 1–6.

[13] M. Mihaylov, A. Mihovska, S. Kyriazakos, and R. Prasad, "Interoperable eHealth platform for personalized smart services," in *Proc. IEEE Int. Conf. Commun. Workshop (ICCW)*, Jun. 2015, pp. 240–245.

[14] L. Catarinucci *et al.*, "An IoT-aware architecture for smart healthcare systems," *IEEE Internet Things J.*, vol. 2, no. 6, pp. 515–526, Dec. 2015.

[15] A. M. Rahmani *et al.*, "Smart e-health gateway: Bringing intelligence to Internet-of-Things based ubiquitous healthcare systems," in *Proc. 12th Annu. IEEE Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2015, pp. 826–834.

[16] V. Gay and P. Leijdekkers, "Bringing health and fitness data together for connected health care: Mobile Apps as enablers of interoperability," *J. Med. Internet Res.*, vol. 17, no. 11, p. e260, Nov. 2015. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4704968/

[17] L. Pescosolido, R. Berta, L. Scalise, G. M. Revel, A. D. Gloria, and G. Orlandi, "An IoT-inspired cloud-based Web service architecture for e-Health applications," in *Proc. IEEE Int. Smart Cities Conf. (ISC)*, Sep. 2016, pp. 1–4.

[18] M. Yuksel and A. Dogac, "Interoperability of medical device information and the clinical applications: An HL7 RMIM based on the ISO/IEEE 11073 DIM," *IEEE Trans. Inf. Technol. Biomed.*, vol. 15, no. 4, pp. 557–566, Jul. 2011.

[19] (2011). *L'agrément des Hébergeurs de Données de Santé à Caractère Personnel, esante.gouv.fr, le portail de l'ASIP Santé*. [Online]. Available: http://esante.gouv.fr/services/reperes-juridiques/lagrement-des-hebergeurs-de-donnees-de-sante-a-caractere-personnel

[20] *Consumer Protection in Fitness Wearables*, Forbrukerradet, Oslo, Norway, Nov. 2016.

[21] G. Mancia *et al.*, "2013 ESH/ESC Guidelines for the management of arterial hypertension: The task force for the management of arterial hypertension of the European society of hypertension (ESH) and of the European society of cardiology (ESC)," *J. Hypertension*, vol. 31, no. 7, pp. 1281–1357, Jul. 2013.

[22] P. Palatini, D. Longo, G. Toffanin, O. Bertolo, V. Zaetta, and A. C. Pessina, "Wrist blood pressure overestimates blood pressure measured at the upper arm," *Blood Pressure Monitoring*, vol. 9, no. 2, pp. 77–81, 2004. [Online]. Available: http://journals.lww.com/bpmonitoring/Fulltext/2004/04000/Wrist_blood_pressure_overestimates_blood_pressure.4.aspx

[23] *Introduction to Android Bar Android Developers*. Accessed: Jul. 2, 2017. [Online]. Available: https://developer.android.com/guide/index.html

[24] (2017). *Smartphone OS Market Share Worldwide 2009–2016 Bar Statistic*. [Online]. Available: https://www.statista.com/statistics/263453/global-market-share-held-by-smartphone-operating-systems/

[25] R. Wang *et al.*, "Accuracy of wrist-worn heart rate monitors," *JAMA Cardiol.*, vol. 2, no. 1, pp. 104–106, Jan. 2017. [Online]. Available: http://jamanetwork.com/journals/jamacardiology/fullarticle/2566167

NAWRAS GEORGI received the Engineering degree in biomedical engineering from Ecole Supérieure d'Ingénieurs de Rennes, France, in 2015. He is currently pursuing the Ph.D. degree with the Laboratoire Traitement du Signal et de l'Image, Inserm U1099, Université de Rennes 1, France. His research topics are health monitoring and telecare.

ALINE CORVOL was born in 1977. She received the Ph.D. degree in medical ethics from Université Paris Descartes in 2013. She became a Geriatrician in 2007 after a fellowship in internal medicine. She is currently a French Medical Doctor with the University Hospital, Rennes, France, where she is responsible for the memory clinic, and an Associate Researcher with ARENES (UMR 6051).

RÉGINE LE BOUQUIN JEANNÈS received the Ph.D. degree in signal processing and telecommunications from Université de Rennes 1, France, in 1991. She is currently a Professor with Laboratoire Traitement du Signal et de l'Image, Université de Rennes 1, and also with the Centre de Recherche en Information Biomédicale sino-français, a French–Chinese Laboratory associated with Institut national de la santé et de la recherche médicale. Her research activities mainly focus on biomedical signals processing and modeling in the field of epilepsy and brain connectivity, on quality evaluation, telemedicine, and telemonitoring applications.