Internet of Things 20 (2022) 100614

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot

Research article

IoT architecture for continuous long term monitoring: Parkinson's Disease case study

Onorato d'Angelis [a], Lazzaro Di Biase [b,c,d], Luca Vollero [a], Mario Merone [a,*]

[a] Unit of Computer Systems and Bioinformatics, Department of Engineering, Campus Bio-Medico di Roma, Via Alvaro del Portillo, 21, Rome, 00128, Italy
[b] Neurology Unit, Campus Bio-Medico University Hospital Foundation, Via Alvaro del Portillo, 200, Rome, 00128, Italy
[c] Unit of Neurology, Neurophysiology, Neurobiology, Department of Medicine, Campus Bio-Medico di Roma, Via Alvaro del Portillo, 21, Rome, 00128, Italy
[d] Brain Innovations Lab, Campus Bio-Medico di Roma, Via Alvaro del Portillo, 21, Rome, 00128, Italy

ARTICLE INFO

Keywords:
IoT
Telemedicine
Continuous long term monitoring
Parkinson's disease
e-Health

ABSTRACT

In recent years, technological advancements and the strengthening of the Internet of Things concepts have led to significant improvements in the technology infrastructures for remote monitoring. This includes telemedicine which is the ensemble of technologies and tools involved in medical services, from consultations, to diagnosis, prescriptions, treatment and patient monitoring, all done remotely via an Internet connection.

Developing a telemedicine framework capable of monitoring patients over a continuous long-term monitoring window may encounter various issues related to the battery life of the device or the accuracy of the retrieved data. Moreover, it is crucial to develop an IoT architecture that is adaptable to various scenarios and the ongoing changes of the application scenario under analysis.

In this work, we present an IoT architecture for continuous long-term monitoring of patients. Furthermore, as a real scenario case study, we adapt our IoT architecture for Parkinson's Disease management, building up the PDRMA (Parkinson's disease remote monitoring architecture). Performance analysis for optimal operation with respect to temperature and daily battery life is conducted. Finally, a multi-parameter app for the continuous monitoring of Parkinson's patients is presented.

1. Introduction

Nowadays Internet of Things (IoT) encompasses many areas of modern life, and, one of the most important area, is in healthcare monitoring system for providing effective emergency services to patients [1]. The use of cloud-based IoT in healthcare provides a wide range of applications and services for patient monitoring beyond the ability to access shared resources and a common infrastructure. However, there are still many issues with patient health monitoring using IoT platform (for a systematic review refers to [2]) and some of these are related to wearable devices. The main ones include: (1) **The compatibility of wearable devices**: depending on the patient's clinical condition, certain sensors must be used for telemonitoring, and not all sensors are always compatible with each other. Furthermore using multiple devices for data acquisition could result in patient discomfort. Few wearable medical devices have effectively integrated multiple functions [3]; (2) **The data accuracy**: two important parameters to

* Corresponding author.
E-mail address: m.merone@unicampus.it (M. Merone).

https://doi.org/10.1016/j.iot.2022.100614
Received 3 August 2022; Received in revised form 6 September 2022; Accepted 7 September 2022
Available online 11 September 2022

consider are specificity and sensitivity. The low specificity, in terms of the ability to correctly identify healthy subjects, may lead to overdetection of benign nonclinical related signals resulting in misdiagnosis. While, low sensor sensitivity, in terms of the ability to correctly identify sick people, increases the risk of false negatives, i.e. subjects who, despite having normal values, are still affected by the disease or condition being researched. This results in missed diagnosis and delayed treatment [4–6]; (3) **Battery life**: monitoring of certain clinical states requires signal extraction systems to have a very high sampling rate, combined with user interaction with the device to obtain direct information from the user. In addition, viewing data in real-time requires continuous transmission. This results in huge battery draining, forcing us to decrease the analysis window during the day. However, the efficiency of a telemonitoring system also depends on the length of the analysis window. Designing low-power consuming and high-energy storage wearable devices have always been a challenging issue [7,8]; (4) **Other issues** such as cost, low data collection, and processing efficiency, unstable human–computer interaction interfaces, and incomplete construction of big data health clouds need to be further improved [5,9,10].

An acceptable IoT architecture for monitoring must take these aspects into account and ensure an extended daily monitoring window to extract data useful for analysis. In literature, several proposed architectures provide respectable experience and try to solve some of these issues. Zhang Q. et al. in [11] proposed an environment-centric framework for the monitoring of warehouse designed specifically for complex and structured systems. This architecture includes only a simple network topology for the sensing layer. In the healthcare field is necessary to have meaningful data throughout the day, so it would be appropriate to use a more user-centric approach. For example, Zhang Y. et al. in [12], proposed a remote mobile health monitoring system with a mobile phone, web service capabilities, and a belt for heart rate monitoring. The main limitation was that the system was capable of only real-time monitoring of the patient's status, not professional analysis and instruction. Furthermore, some users were not enthusiastic about the idea of continuously wearing a monitoring device. The reason could be the overall discomfort caused by the belt for the heart rate monitoring. This is a crucial point to monitor over a large time window, you need a system that meets users' needs. So a possible solution is to use devices that are already part of the user's daily life. Regarding this, an interesting framework was developed by Kheirkhahan M. et al. in [13]. They developed a smartwatch-based framework for real-time and online assessment and mobility monitoring.

The proposed ROAMM framework includes a smartwatch application and server. However, while they have achieved great results, they encountered issues concerning battery life and that forced them to monitor only in certain time windows during the day. Especially in the field of telemedicine, having continuous monitoring, not limited to certain time windows, can be crucial in the identification of some clinical parameters or the identification of pathological states or dangers

The proposed paper aims to provide the elements for the creation of a possible framework for continuous long-term monitoring of patients, solving both some problems from a technical point of view and creating a framework that meets the needs of the user. So the proposed IoT architecture is user-centric and, it is adaptable for various application cases. As proof of concept, we adapt the proposed architecture to monitoring patients with Parkinson's disease. Through analysis of the battery of the monitoring device (smartwatch), we extend the monitoring window to a duration that covers the entire day of activity of the user. Finally, a multi-parameter app for continuous monitoring of Parkinson's patients is presented.

The rest of this work is organized as follows: the next section presents the proposed IoT Architecture for continuous and real-time monitoring; in Section 3 we present the Parkinson's Disease Remote Monitoring Architecture (PDRMA); in Section 4 we show and discuss the main aspects of PDRMA; finally, Section 5 provides concluding remarks.

2. Proposed general architecture

The developed architecture, is designed for daily long continuous monitoring sessions. To achieve this goal, on the one hand, the heart of the system, the edge system, has to facilitate the user experience, on the other hand, it must perform essential functions to preserve the battery life of wearable systems.

An overview of the architecture is shown in Fig. 1. The main elements of the proposed architecture are the edge system, the fog system, and the cloud system. The main functions of the edge system are the storage of data coming from sensors and the User Interface (UI), data communication between sensors, and from/to the fog. The fog system stores the data sent by the edge system, and performs other major functions such as acting as a bridge between cloud and edge in order to update configuration parameters in the latter. Finally, the cloud offers a web interface for data visualization and control database storage, AI analytics, and patient digital representation (Digital Twin Model).

Each component of the architecture has to handle three types of data: user data, management data, and control data. Thus, depending on the nature of the data, certain aspects need to be taken into account to avoid communication errors between the various parts of the system and to avoid losing relevant information. The types of data are explained in detail below:

- **User data:** these data are generated by the direct interaction, either through GUI or indirectly through sensors, of the user with the system, or are generated for consumption by the user of the system (alerts, warnings, . . .). One of the main problems of sensor data is in the transmission layer, which may incur network congestion. Congestion occurs when the amount of data to be transferred exceeds the effective network capacity. To avoid network congestion, the architecture introduces a flow controller that verifies the amount of data to send before every transmission. If necessary, related to buffering bounds, it has to divide the data to be sent in accordance with a policy of traffic shaping. Furthermore, a high sampling frequency leads to a substantial increase in memory occupation. Therefore the workflow controller, if there is an error in the transmission that does not allow the memory to be cleared, must interrupt data acquisition to avoid data loss and prompt the administrator.

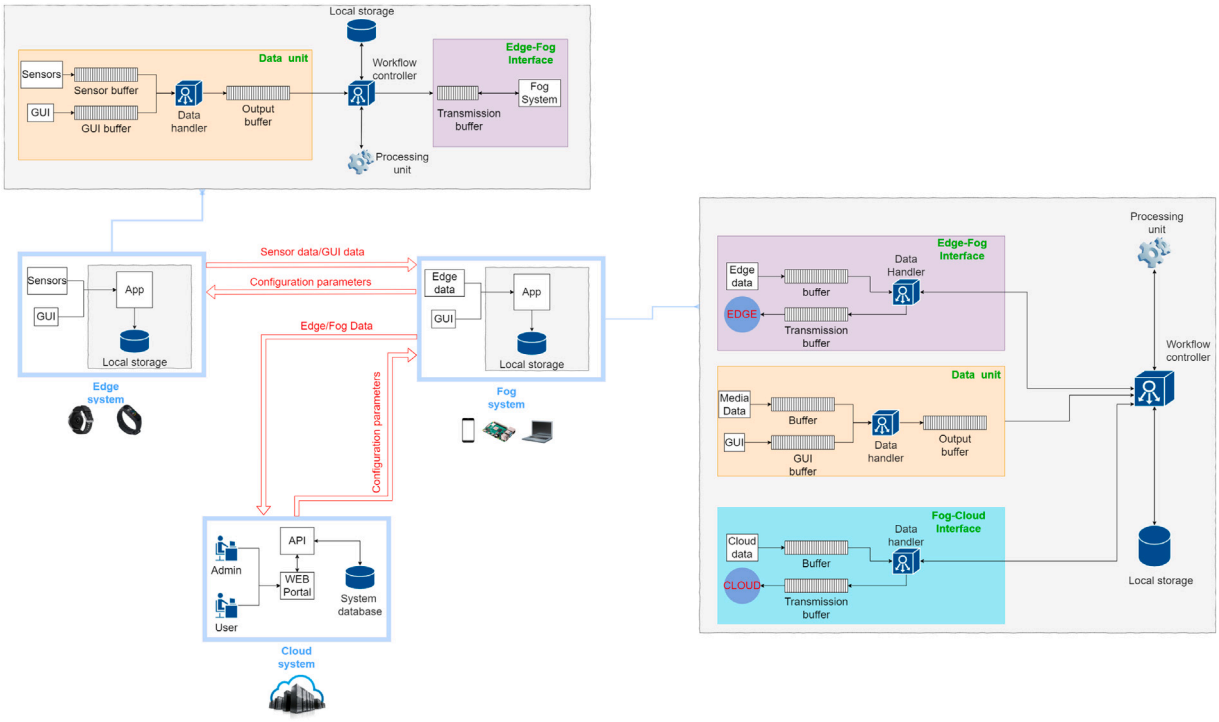


Fig. 1. Architecture overview: The figure shows the functional elements of the architecture, divided into edge, fog, and cloud. For the peripheral systems (edge and fog systems), the internal structure is shown in more detail.

Users directly interact with the system by entering data via a GUI: the administrator can configure the system to prompt the user at a specific instant of time to enter a data item, and the alerts handler carries out a control so that the parameters set by the administrator are respected. A crucial aspect of user input is the UI: UI should follow the principles of UX Design (User experience design).

User data also includes other unstructured data such as audio and video. Audio and video data, as in general unstructured data, do not have a predefined data model and therefore are not suitable for a traditional relational database. So, identified the component of the system that will have to generate this type of data, it will be necessary to provide a specific space on the server for their storage.

- **Management data:** they are the configuration parameters, namely parameters set by the administrator, such as the sensors to be used in the application and their sampling frequency. For this type of data, there is a need for a sensor identification method (each sensor will have its ID), and a workflow controller. This is to synchronize the data to a specific timestamp and to have a predefined logic for data acquisition.
- **Control data:** they are the device parameters, such as battery, transmission error, application error, etc. Keeping track of these parameters is important to alert the administrator in case of malfunctions and intervene promptly.

In the next sections the edge, the fog, and the cloud system are analyzed in detail.

2.1. Edge system

The edge system manages data collection: sensors data and GUI data. The architecture of the edge system is summarized in Fig. 1. The main component is the flow controller, which synchronizes the operations of the data and transmission unit. The data unit collects data from sensors and GUI and through a data handler, organizes all these data in an output buffer. The flow controller deals with determining the exact time for saving this data to the local storage. Subsequently, when possible, the flow controller fetches data from local storage, processes them through the processing unit, and finally creates a transmission buffer to send the data to the fog system. In this way, we avoid network congestion, and the data, depending on its type, will be stored in the proper location.

The sequence diagram of the logic of the edge system is shown in Fig. 2. The main data flows are: (1) **Sensor and GUI data flow:** sensors data are collected based on sampling rate (stored on configuration parameters section of local storage) and they are stored in local storage through the data unit. Likewise, when the user interacts with the application and enters new data, this is sent to the data unit and stored locally. As soon as the workflow controller gives the command to send the data to the fog system,

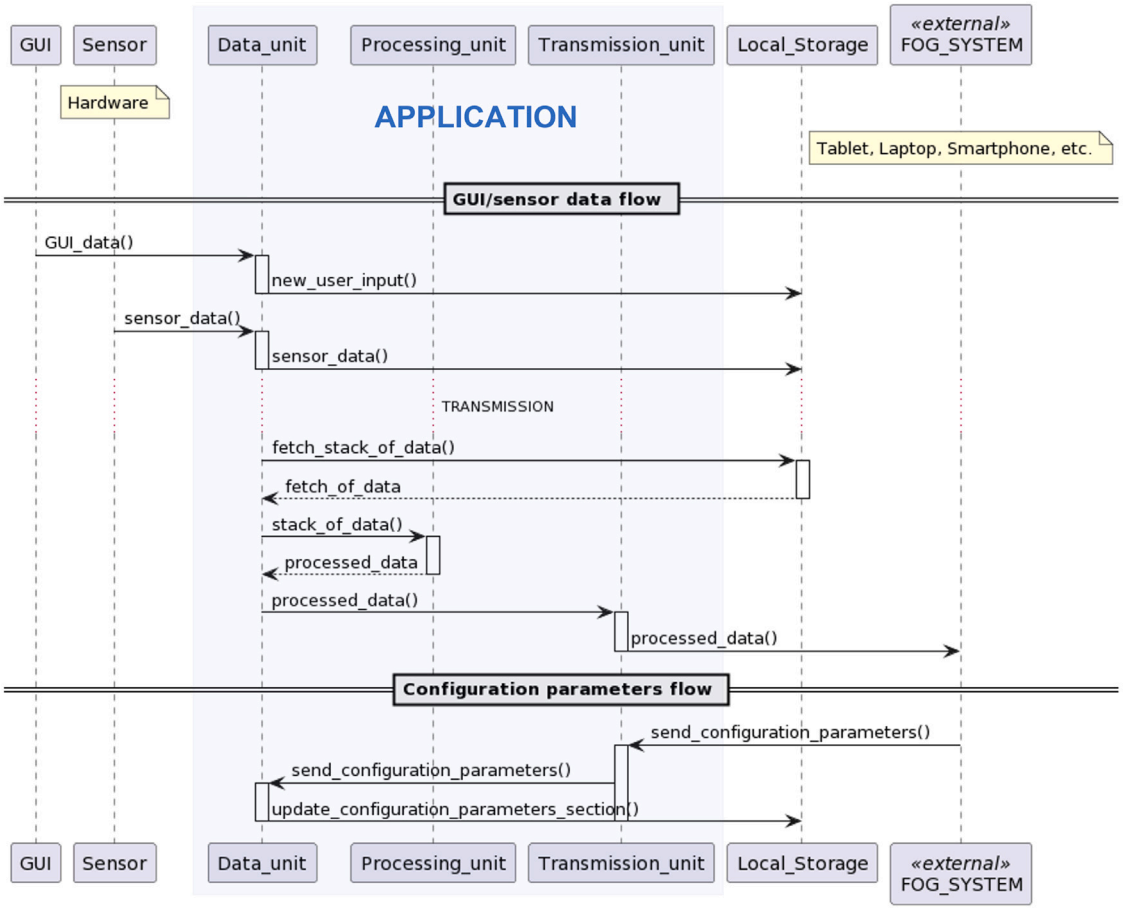


Fig. 2. Edge system sequence diagram. Data_unit, Processing_unit, and Transmission_unit are part of the application layer and they are controlled by the flow controller. GUI and Sensor are the sources of the edge system data.

a stack of data is fetched from the local storage, elaborated through the processing unit, and sent to the fog system through the transmission unit. (2) **Configuration parameters data flow:** the transmission unit receives configuration data from the fog system, and sends it to the data unit which provides for storing it in the local storage. A key aspect to consider is edge-side transmission technologies and policies, which significantly impact the system's ability to operate for extended service times.

2.2. Fog system

The fog system is the bridge between the edge and the cloud system, indeed, it locally saves the data sent by the edge system, and it periodically sends the data to the cloud database. Also, it might produce unstructured data such as audio and video to relieve the data load at the edge system. The architecture of the fog system is shown in Fig. 1. The main component is the flow controller, which synchronizes the operations of the edge–fog interface, the fog–cloud interface, and the data unit. Fig. 3 shows the data flow with respect to the operations carried out by the fog system: (1) **Edge system data flow:** when new data from the edge system are available, the data are reported to the data unit. Subsequently, the data are elaborated by the processing unit and stored in the local storage. As soon as the workflow controller gives the command to send the data to the cloud system, a stack of data is fetched from the local storage and is sent to the cloud system through the transmission unit; (2) **Media data flow:** when the audio or video file is recorded from the application, the media data are reported to the data unit and processed by the processing unit. Subsequently, they are saved in the local storage, in order to send them to the cloud system through the transmission unit; (3) **Configuration parameter data flow:** the cloud system sends new configuration parameters to the fog system, the transmission unit sends the configuration parameters to the data unit to save them in the local storage. The workflow controller takes care of getting the configuration parameters from the local storage and sending them to the data unit. The parameters are then sent to the transmission unit and finally sent to the edge; (4) **Authentication workflow:** the user enters authentication data through the GUI. The login data are sent to the data unit that sends them to the transmission unit. The transmission unit sends the login data to the cloud system, which responds with the authentication status. If the user is authenticated, the transmission unit prompts the flow controller to enable system operations.

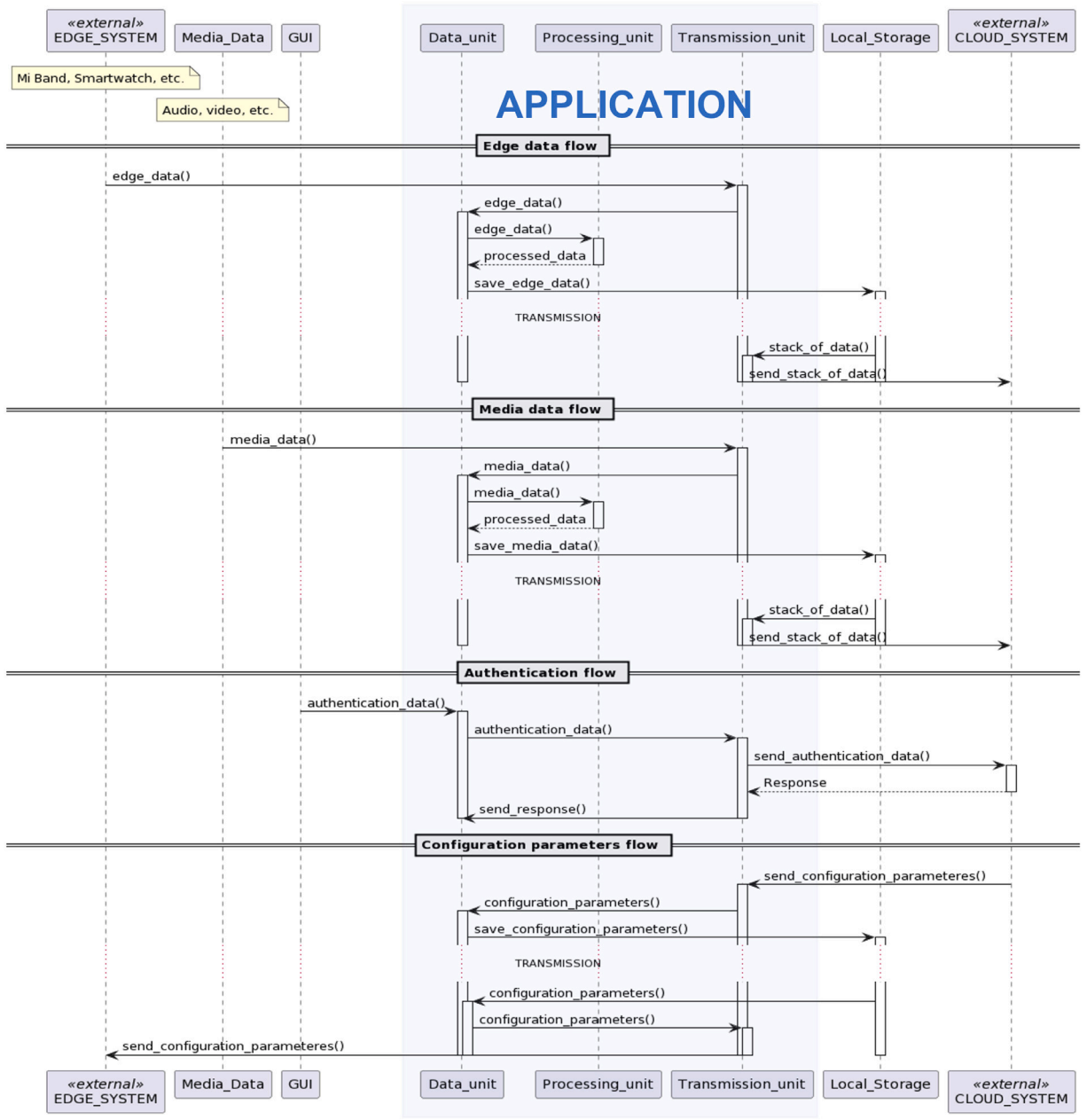


Fig. 3. Fog system sequence diagram. The Data_unit, Processing_unit, and Transmission_unit are part of the application layer and they are controlled by the workflow controller. GUI and Media_Data are the sources of the edge system data. The fog system communicates with both the edge system and the cloud system.

2.3. Cloud system

As shown in Fig. 1, the cloud system is composed of three main components: Web portal, API, and database. Each component has a specific role, the database stores fog and edge data and, for security reasons, is located in a virtual machine dislocated from the web portal and the API.

The API acts as a bridge between the web portal, the fog system, and the database. It implements functions such as authentication management (OAUTH2 or Apikey), database interaction, user and administrator functions management, etc.

Finally, the web portal is essentially the access point for registering with the system and viewing and downloading data. It implements a user and administrator interface: the user through the portal can read instructions written by the administrator and

upload consent for testing and data processing. While the administrator, through the portal, can configure edge and fog systems parameters.

The choice of database is crucial for healthcare applications. Each database has its pros and cons. From the study in [14], it was compared a NoSQL database (MongoDB) with MySQL database. It was observed that in some scenarios, MongoDB provided less response time compared to MySQL. But MySQL's response times are stable compared to MongoDB's in executing queries for a larger number of records. Choosing a better database for healthcare depends on the requirements of the application. A NoSQL database has been chosen for this architecture since most of the data are of an unstructured type.

The API and the database have to consider some rules for the correct functioning of the system and respect user privacy. The system has to implement a user identification method that does not use personal data, to retrieve data from the database through the API. Moreover, the personal data section in the database should be separated from the recorded data.

At the database level, two key aspects must be considered: structure flexibility and fast data retrieval. Regarding the structure, it has to be scalable and easily amenable to future changes and each collection should have supporting parameters that help the API to perform fast queries. To obtain fast data retrieval, the query performed by the API should be selective (it is necessary to clearly define the requirements before writing the query so that only the necessary information is received). A support tool to improve query performance, in NoSQL database, is the indexing method: an index supports a query when the index contains all the fields scanned by the query [15].

Regarding the logic of the cloud system, Fig. 4 describes the sequence diagram of data flows carried out by the cloud system: (1) **Edge/fog data flow:** the edge/fog data are sent to the API, that process the request and checks the user authentication status. If the user is authenticated, it sends the data to the Virtual Machine (VM) that hosts the database. Here the database is updated. If the user is not authenticated, an error is sent to the fog system; (2) **Authentication data flow:** the authentication data sent by the fog system, are processed by an API function that executes a check in the database and responds with the outcome of the operation; (3) **User/administrator interaction flow:** the user or the admin interacts with the web portal. The API receives and processes the request from the web portal and checks if the user is authenticated. If the user is authenticated, the API retrieves the data from the VM and makes the data available on the web portal. If the user is not authenticated, an error is reported; (4) **Configuration parameters data flow:** the administrator set the configuration parameters for the users through the web portal, the API processes the request, and if the admin is authenticated and it is verified the status of administrator, inserts in the VM these parameters and pulls out the edge address of the users. Finally, the fog systems are prompted with new parameters.

3. Case of study: Parkinson application

Parkinson's disease (PD) is a chronic progressive neurodegenerative disorder, characterized by the presence of predominantly motor symptoms (bradykinesia, rest tremor, rigidity, and postural disturbances). It is also associated with a variety of non-motor symptoms, which, together with late-onset motor symptoms (such as postural instability and falls, speech and swallowing difficulties), are to date challenging symptoms to treat for the neurologist. PD has taken over almost 10 million people based on the statistics provided by the World Health Organization [16].

In the management of Parkinson's disease, it is well known that there are several problems related to both diagnosis (error rate reaches 30% [17]) and therapy management, mainly related to motor fluctuations and dyskinesias. A solution to improve these aspects is certainly the continuous monitoring of the patient through wearable sensors and telemedicine systems. Remote monitoring allows care to be brought to the patient's home and this makes Parkinson's disease care more patient-centered [18] with personalized therapies [19]. This is in line with the trend in medicine now moving toward precision medicine.

A valuable tool for the remote monitoring of PD patients is the inertial system. The inertial systems are now integrated into wearable sensors and smart devices and it has been proved their impact in the evaluation of motor abnormalities. In literature there are several studies where the usage of wearable sensors is peculiar in clinical monitoring of PD patients and, in particular, these systems are used to monitor patient's cardinal motor symptoms like bradykinesia [20–23], rigidity [24–28], and tremor [29–31] or gait, posture, balance axial motor symptoms [32–38]. For example, in [39] has been proved the wearable electronics impact as valid support in assessing a correct evaluation of PD in its early stage. In [40] has been proved that motor anomalies in PD can be detected through analysis of keystroke dynamics during typing on smartphone touchscreens. LeMoyne et al. in [41], demonstrate the capability of the iPhone wireless accelerometer to quantify Parkinson's disease tremor attributes. However, in most of these studies, data are offline analyzed, and embedded devices are used only for the acquisition.

In order to allow an online analysis of data, and thus continuously monitor the patient, there is a need for a system that allows the visualization of data in real-time. Regarding this, several solutions have been proposed: in Shahar Cohen et al. study [42], a mobile application and an Internet of Things (IoT) platform have been developed in order to support large-scale studies of objective, continuously sampled sensory data from people with PD. Even though the system seems to work reasonably well, they encountered complications in smartphone and smartwatch device pairing and battery drainage. As mentioned in Section 1, battery life is one of the technological barriers in healthcare wearables. One possible solution is presented in the study of Paola Pierleoni et al. [43], where, in order to develop a smart inertial system for 24 h Monitoring, a Power Management Unit with a capacity of 850 mAh is used. Although many of these methods have yielded acceptable results, from a long-term monitoring perspective, the monitoring device must be appropriate for the patient's daily life. Therefore, it must be a wearable device that facilitates the user experience and is easily integrated into daily life. This means that in some respect, it should not be alienating to the patient and should not require too much of the patient's attention on many tasks throughout the day. This study proposes a patient-centered solution to aid in symptom monitoring through a telemedicine system.

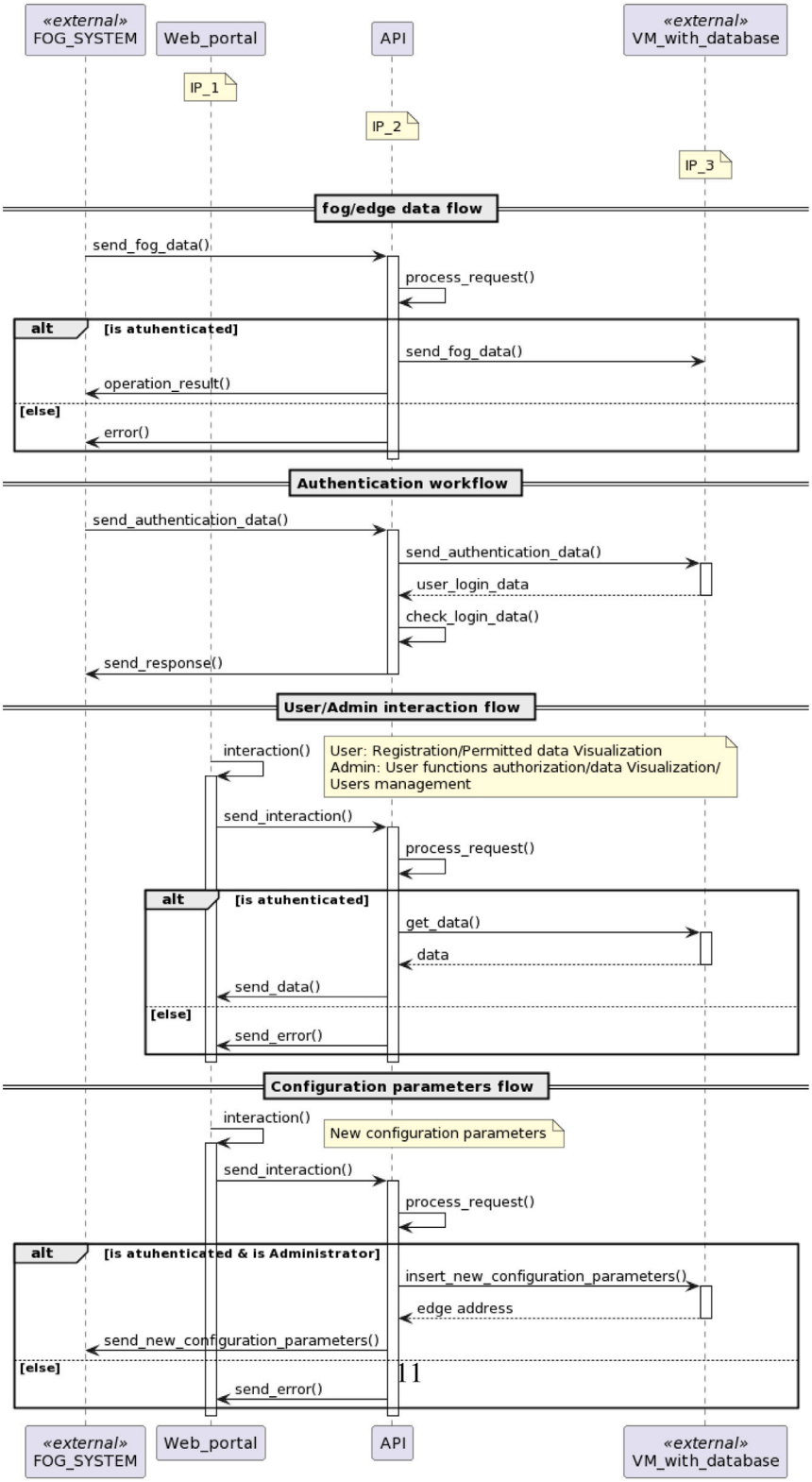


Fig. 4. Cloud system sequence diagram. The main components of the cloud system are the API, the Web portal and the Virtual Machine with the database of the system Moreover the User/Admin interaction from the Web Portal is described.

3.1. Recording data of edge/fog system

To develop a framework for PD and implement the architecture described in Section 2, there is a need for a match among application-specific data to be extracted and data sources devised in the architecture definition. A fundamental datum is the motor status of the patient, correlated to motor symptoms. In Parkinson's disease the main motor symptoms are:

- Cardinal motor symptoms:
  - Bradykinesia: slowness of initiation of voluntary movement with a progressive reduction in speed and amplitude of repetitive actions;
  - Rigidity: an increase in muscle tone at rest leading to a resistance to passive movements;
  - Tremor: is an involuntary, rhythmic movement that affects a part of the body and it is caused by the rapid and alternating contraction and relaxation of muscles.
- Motor complications:
  - Motor fluctuations:
    - 'ON' time is when antiparkinsonian treatments with the drug levodopa are working well and the symptoms are controlled;
    - 'OFF' time is when antiparkinsonian treatments are no longer working well and symptoms such as tremor, rigidity, and bradykinesia re-emerge.
  - Dyskinesia: are unintended, involuntary movements induced by levodopa (the main Parkinson's disease therapy) that typically occur during ON-time.

In order to acquire these data, accelerometer, magnetometer and gyroscope have been used as sensor sources and are easily integrated into the architecture presented in Section 2. The feasibility of using accelerometers to estimate the severity of symptoms and motor complications in patients with Parkinson's disease has recently been demonstrated [44] and previous works used gyroscopes to measure velocity and stride length, joint angle of lower limbs, the angular velocity of trunk rotation, and angular displacement of trunk motions [45]. In the study, the gyroscope is used with the scope of estimating the angular velocity of hand rotation.

Therefore, acquiring information about the patient's perceived state can help the doctor in the right therapy. A graphical interface has been implemented for the insertion of the perceived motor state.

As mentioned previously some symptoms, such as 'ON-OFF' swings, are related to taking medications. So it is crucial to have a history of drugs taken.

Additional information such as daily meals and sleep are acquired through graphical interfaces. Other data that are helpful in assessing the patient's clinical status include audio and video data. Videotaping of patients is a qualitative and quantitative method useful for the analysis of motion disorders [46]. In addition, has been proved that Parkinson's disease (PD) is characterized by speech and voice symptoms that invariably impact the PD patient's ability to communicate effectively. Described as hypokinetic dysarthria [47], it has been estimated to be present in between 70 and 89% of PD patients [48]. The framework utilizes a smartphone to get these data.

3.2. PDRMA (Parkinson's Disease Remote Monitoring Architecture)

PDRMA mirrors the architecture presented in Section 2 and adapts it to the application case. Additional consideration should be made about the method of data transmission between smartphone and smartwatch. The smartwatch, to preserve battery life, requires an energy-efficient transmission technology. So, the system uses a Bluetooth Low Energy (BLE) transmission between these two devices. The synergy between good performance and ubiquitous diffusion makes BLE an excellent candidate for a great variety of applications and, among these, in the medical field for e-health applications [49–51]. In addition [52] shows BLE protocol and IMU sensor are used for the early diagnosis of Parkinson's disease.

Whereas, the smartphone uses a high throughput data-intensive communication technique (HTTP requests) to update the database with the smartwatch's data. In Fig. 5 the sequence diagram of the logic of BLE and HTTP requests are shown. Regarding the BLE protocol, the smartwatch takes the role of server while the smartphone takes the role of client. The smartwatch exposes a service and a characteristic, with write and read permissions to send data and receive the therapeutic plan. Write and read operations must be synchronized, so the smartphone implements a flow controller. If there is an update of the therapeutic plan, it is taken from the local storage, and a write request to the characteristic exposed by the server is performed. The smartwatch saves the therapeutic plan in local storage and sends a notification to the smartphone to start the transmission of the sensor data. So the smartphone sends a characteristic read request, and the smartwatch sends sensor data. If the transmission has a positive outcome, the data are deleted from the smartwatch's local storage.

The data collected by the smartwatch are then sent via HTTP requests to the cloud and stored in the database. The API takes care of saving the data in the right way.

Every day the smartphone checks if there is an update of the therapeutic plan through the API and if that is the case, the new therapeutic plan is taken from the database and updated in the local storage of the smartphone.

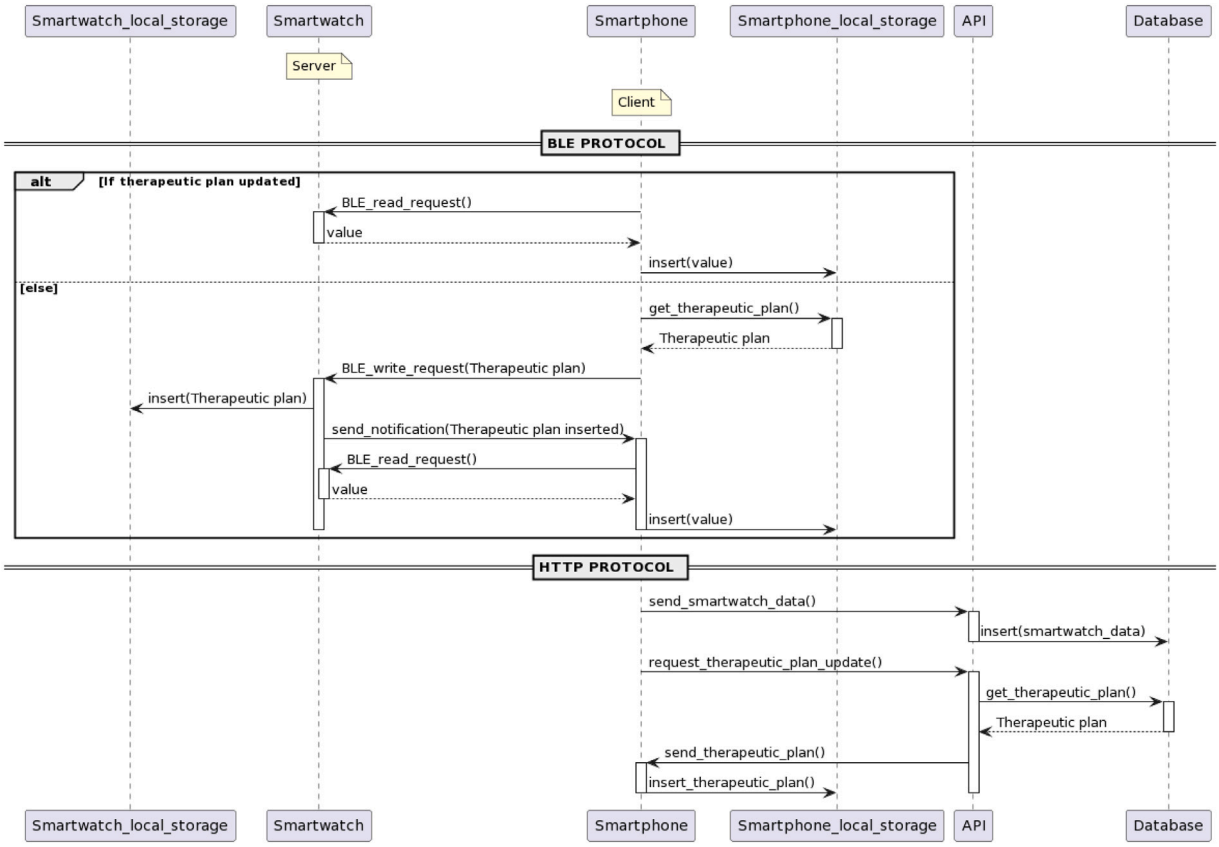


Fig. 5. Communication protocols sequence diagram. The smartwatch and the smartphone communicate through a BLE connection, in order to have an efficient and low energy connection. While the smartphone and the cloud system communicate through the HTTP requests.

3.3. Edge/fog system choice

The edge system has been chosen in such a way as to avoid patient discomfort, have accuracy in acquired data, preserve battery life, have a device that can be easily integrated with various types of systems, and finally, it has a low economic cost. So the device chosen for the edge system is a smartwatch, the Samsung Galaxy watch4. The fundamental characteristics that justify the choice are:

- Low-cost device (210$);
- Large local storage (16 Gb) and large RAM size (1,5 Gb);
- Built-in accelerometer, gyroscope, and magnetometer;
- Support of both low transmission rate (BLE) and high transmission rate (WiFi) technologies;
- Battery autonomy of approximately 40 h (in watch mode).

Regarding the data accuracy of sensors, some studies have proven that this type of device can be used for individual activity recognition, major body movement location detection, activity intensity detection, and locomotion detection tasks [53].

To facilitate communication with the smartwatch, a Samsung smartphone was used for the fog system.

Patient inputs data. Along with sensor data, the smartwatch collects data from patient input. The application handles four types of input: motor diary, taken drugs, sleep time information, and meal time information.

1. **Motor diary:** the database stores two types of information. Every hour an alert asks the patient to enter the perceived motor state. The state inserted by the patient is stored in local storage with the relative timestamp. If there is an intra-hour change in motor status, the patient can access the motor diary UI and enter the new status and, this new status is stored in another section of the database with the relative timestamp.
2. **Meal time information:** when the patient has a meal, he has to open the meal activity and confirm the action. In this way, the corresponding timestamp of the action is stored in local storage as meal time.

3. **Drugs information:** the therapeutic plan is stored in local storage. Each drug has a relative timestamp that refers to the time the drug should be taken by the patient. An alert is generated for each drug's timestamp, and, through the UI (slider with all drugs), the patient selects the drug and confirms the assumption. All drugs and related information (name, time, and quantity) are visible in the drugs section of the application. Drugs taken during the day are colored green, while others are red;
4. **Sleep information:** through the sleep UI, the patient can select sleep modality or wake-up mode. This way in the local storage saves the time the patient goes to sleep and the time he wakes up. At the moment the patient selects sleep mode, the sensors stop and no further data is recorded. However, by default, sensors record from 8:00 a.m. to 8:00 p.m., regardless of patient entry.

4. Results and discussion

4.1. Smartwatch application

As mentioned in 3.1, an accelerometer, magnetometer, and gyroscope are used to estimate the severity of symptoms and motor complications. The best choice for the logic of the architecture, and to avoid battery consumption is to run the sensors in the background, even when the user is not interacting with the application. So in the system, a background service is implemented (listed in the android documentation as a foreground service) and it performs operations that are noticeable to the user and show a status bar notification. So that users are actively aware that the app is performing a task in the foreground and is consuming system resources.[1]

To increase the responsiveness of the app, the application implements a multithread approach. Multithreading in an interactive application may allow a program to continue to run even if part of it is blocked or is performing a long operation, thus increasing responsiveness for the user. This allows us to optimize communication with the fog system, as we could have one thread for communication and one for user interaction with the application. Also, a multithread approach allows sharing its resources such as memory, data, files, etc. So a single application may have several threads within the same address space using resource sharing. This aspect allows us to better manage the database interactions.

In the smartwatch application, there are three concurrent threads: the main thread for user interactions with the application, the I/O thread for database interaction the network thread for data transmission. The application logic is shown in Fig. 6. When the application is launched, a foreground service and the main thread are initialized. The foreground service initializes the I/O thread and the network thread, which will start to record sensor data and transmit them to the smartphone. As soon as the user interacts with the application to enter a value, the main thread passes the data to the foreground service that will provide to save them in the local storage. The internal logical components of the foreground service block follow the logic presented in Section 2.1. While the flow of configuration parameters is represented by the therapeutic plan.

4.1.1. Battery considerations

As mentioned early the objective is to ensure the longest continuous monitoring time of the patient without the need of replacing or recharging the battery of the wearable device, and hence, some aspects must be considered to avoid excessive battery consumption. The first aspect to consider is the sampling rate of the sensors, as the sampling rate has a high impact on the battery consumption. In [13] is proved that with a Samsung Gear S3, with a sampling rate of 10 Hz and GPS on, the battery level after 2 h of use drops from 380 mAh, on a full charge, to 76 mAh.

GPS data are not important, in general, for these types of applications, so we have decided to turn off the GPS and avoid its unnecessary battery consumption. Also, the data are passed via BLE, so we can turn off the Wi-Fi (an item that can consume a lot of battery power). The parameter on which we have control is the sampling rate: considering that tremor has frequency components up to 12 Hz [54] and dyskinetic movements may have significant frequency components up to 8 Hz [55], in order to evaluate motor symptoms, for the Nyquist–Shannon sampling theorem we have to implements a sampling frequency of at least 30 Hz.

So, having a constraint on the sampling rate, we could think about acting on the transmission policy between the smartwatch and the smartphone, being data transmission is also involved in battery consumption. In particular, we can provide windows of data transmission between the two devices, and windows in which transmission does not occur. However, not having a continuous transmission, could lead us to lose the real-time condition. As mentioned earlier, the sampling rate of the sensors is 30 Hz. If each row of data (consisting of the x, y, and z values of each sensor used, the timestamp, and, possibly, values entered by the user at that related timestamp) takes up a space of 230 bytes, we store an amount of 24,84 MB per hour. The communication protocol chosen between the smartphone and the smartwatch is the BLE protocol. Considering the BLE ATT Maximum Transmission Unit (MTU), namely, the maximum length of an ATT packet, allows us to have a maximum frame size of 512 bytes, we could send two rows of data at once without exceeding the limit. If we choose a policy of continuous data transmission, we could send 49,68 MB per hour, and this allows us to have a real-time condition (the data on the server will be available in real-time). At the point when we choose to adopt a data transmission policy, which restricts data to be sent only in predetermined time windows, we cannot guarantee the real-time condition, having a size limit on the outgoing data for each sending. Therefore there is a need to do data operations to ensure that the same amount of data is always sent. Specifically, we performed these operations:

[1] As reported in https://developer.android.com/guide/components/foreground-services.

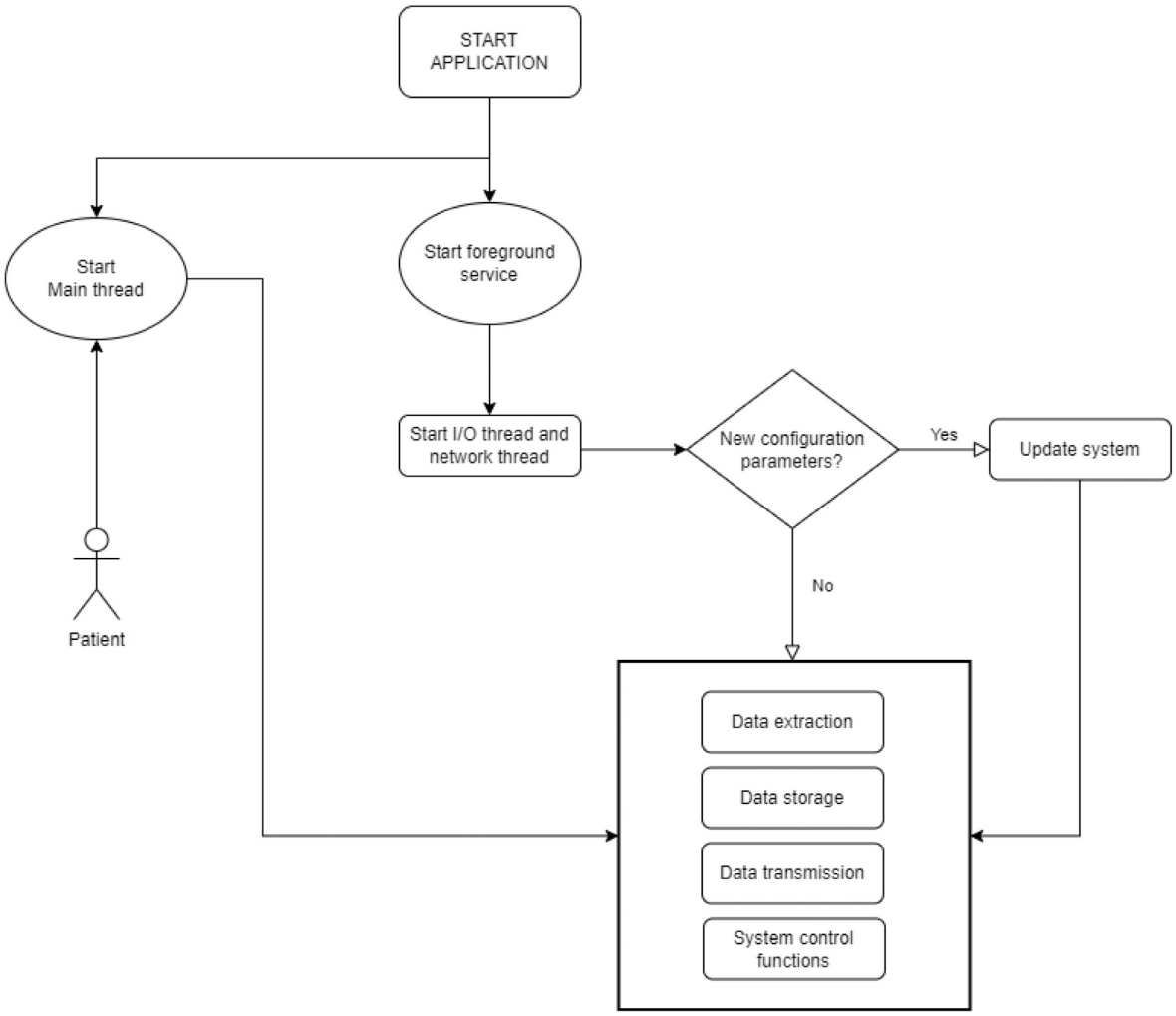


Fig. 6. Smartwatch application flow chart diagram.

- Vectorization: The data samples were encoded into JSON objects. To optimize their transmission over the EDGE channel, these objects underwent a squeezing process, with the removal of implicit information, the keys of the elements, before transmission from the smartwatch, and later reinsertion, after the reception on the smartphone;
- Data compression: we use the Deflate algorithm that allows lossless data compression [56];

By performing these operations, we reduced the size of the data sent by 90%. This allows us to adopt the policy in which transmission occurs in predefined time windows, while still maintaining the real-time condition.

In order to evaluate battery consumption with this type of policy, we performed various tests by varying the time windows for sending data. The smartphone systematically alternates time windows of data requests and time windows in which it is in sleep mode on transmission. In particular, three different policies were evaluated by dividing a one-hour time window into two sub-windows: one window in which data are sent, and one in which no transmission occurs. The subwindows evaluated are:

1. 10 min of sending data, and 50 min when there is no transmission (sleep mode)
2. 20 min of sending data, and 40 min when there is no transmission (sleep mode)
3. 30 min of sending data, and 30 min when there is no transmission (sleep mode)

To have a metric for comparison, a baseline corresponding to battery consumption under idle transmission was evaluated. Fig. 7 shows the graph of battery power consumption where the x-axis represents the time expressed in hours:minutes:seconds while the y-axis is the battery percentage.

Table 1 shows the results in detail. We report for the three modes of sending time (ST) and sleep mode (SM) the average battery drain and the mean difference from baseline. The results of Table 1 and Fig. 7 show that increasing the sending time, increases the battery drain.

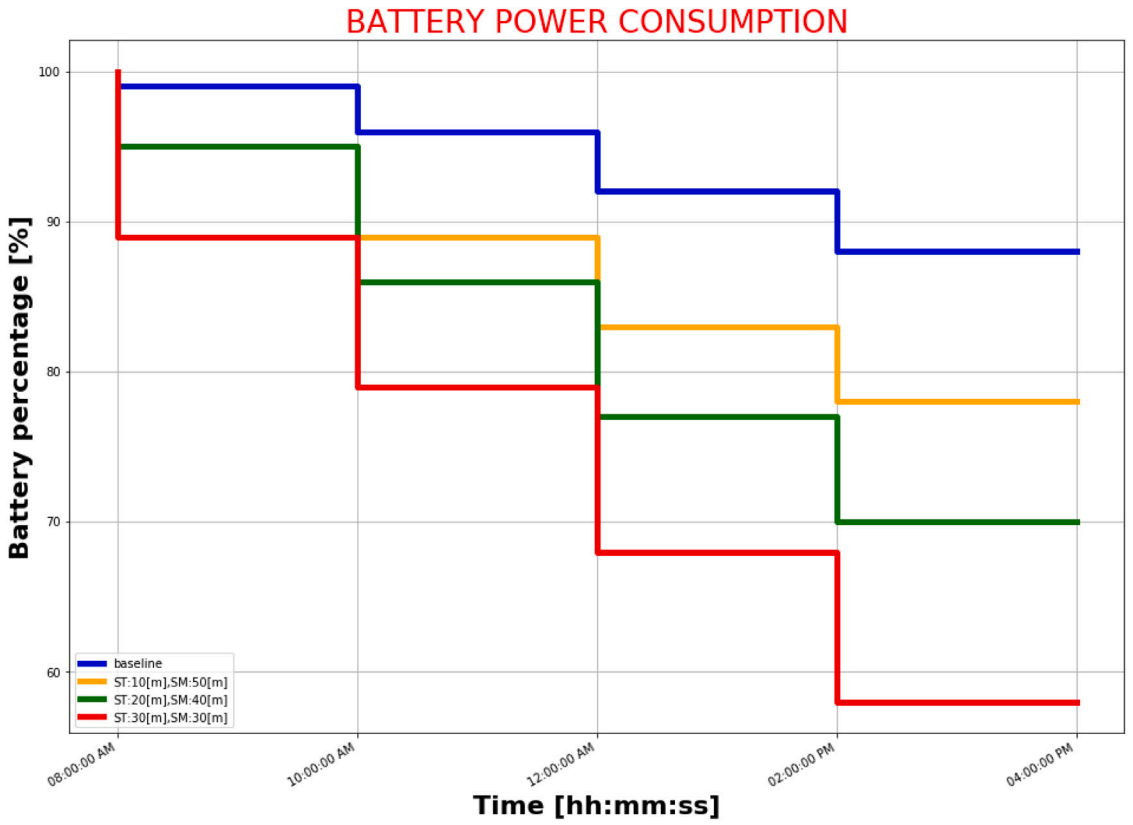


Fig. 7. Battery test. ST stands for "sending time", while SM stands for "sleep mode". Measures are in minutes. The *y*-axis represents the battery residual charge level. Different colors correspond to different transmission policies.

Table 1
Battery drain measurements: ST stands for "sending time", while SM stands for "sleep mode". Both measures are in minutes.

Mode		Average battery drain every two hours	Mean difference from baseline every two hours
ST[m]	SM[m]		
10	50	5.5%	2.5%
20	40	7.5%	4.0%
30	30	10.5%	7.5%

Table 2
Battery draining and device temperature based on HTTP frequency requests.

Frequency [request/hour]	Average Temperature [C°]	Average battery consumption [%]
Baseline	24.7 ±0.7	0.63 ±0.48
12	27.1 ±0.7	2.90 ±0.51
12	25.9 ±0.5	0.91 ±0.51
5	24.2 ±1.6	0.80 ±0.56

However, in some architectures, you may have a configuration in which there is no fog system. In this regard, we wondered whether it is possible to optimize battery life in an architecture where the smartwatch directly sends data to the cloud via HTTP requests. We hypothesized that device temperature may impact battery draining. Specifically, depending on the frequency with which HTTP requests are executed, increasing battery temperature also increases battery draining. We performed four tests: an HTTP request every 5 s, an HTTP request every 5 min, and an HTTP request every 10 min. Each test was performed by extracting the temperature and battery level value every 20 min. All tests were executed with GPS and Wi-Fi turned on and to have a metric for comparison, a baseline corresponding to battery consumption without data transmission was evaluated. Fig. 8 shows the graph of battery power consumption and the temperature trend. For each test the *x*-axis represents the time expressed in minutes, the *y*-axis represents the battery level in percentage, while the second *y*-axis represents the temperature in Celsius.

Table 3
Comparison of battery lifespan of some architectures in the literature.

Paper	Sensors employed and device type	Data transmission method	Battery lifespan
Gonzalez C. et al. [57]	Internal inertial system and an external sensor node (Huawei Watch 2 with a battery capacity of 420 mAh)	Wi-Fi connection and BLE	From 3.5 to 5 h
Robert Wu et al. [58]	Optical sensor, accelerometer, and gyroscope (LG Watch Urbane Smartwatch Android Wear with a battery capacity of 410 mAh)	The smartwatch sends sensor information to the smartphone (method not specified)	Battery life of 16 h (the smartwatch recorded 2 out of every 10 min)
Anna L. Beukenhorst et al. [59]	Accelerometer, gyroscope, and magnetometer (Huawei Watch 2 with a battery capacity of 420 mAh)	The watch was permanently prevented from transmitting data (in "airplane mode") until it was connected to a charger overnight	From 10 to 12 h
Virginia LeBaron et al. [60]	Accelerometer and optical sensor(Wear OS Fossil Sport Watch with a battery capacity of 350 mAh)	The smartwatch sends sensor information to a laptop (method not specified)	7 h
Matin Kheirkhahan et al. [13]	Accelerometer, gyroscope, GPS, and optical sensor (Samsung Gear S2 and S3 with a battery capacity of 380 mAh)	Wi-Fi or 4G network connection	12 h alternating non-wear times and wear times: from 8 am to 8 pm participants were prompted at four random times with a minimum three-hour gap between two consecutive prompts
PDRMA	Accelerometer, magnetometer, and gyroscope (Samsung Galaxy watch4 with a battery capacity of 361 mAh)	Wi-Fi/BLE	12 h

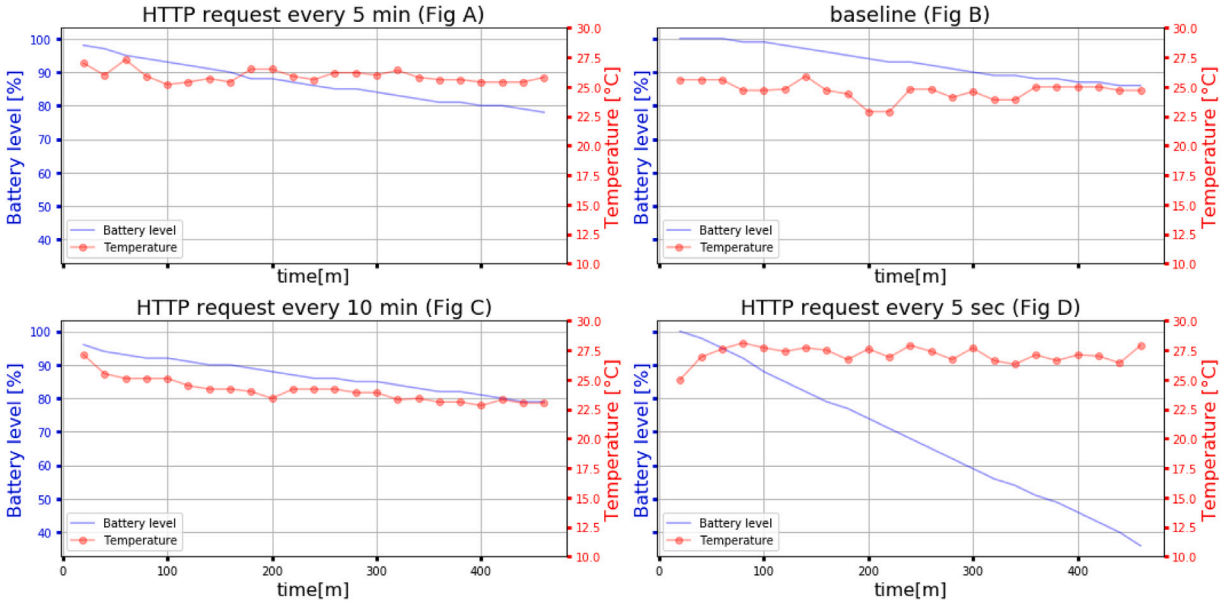
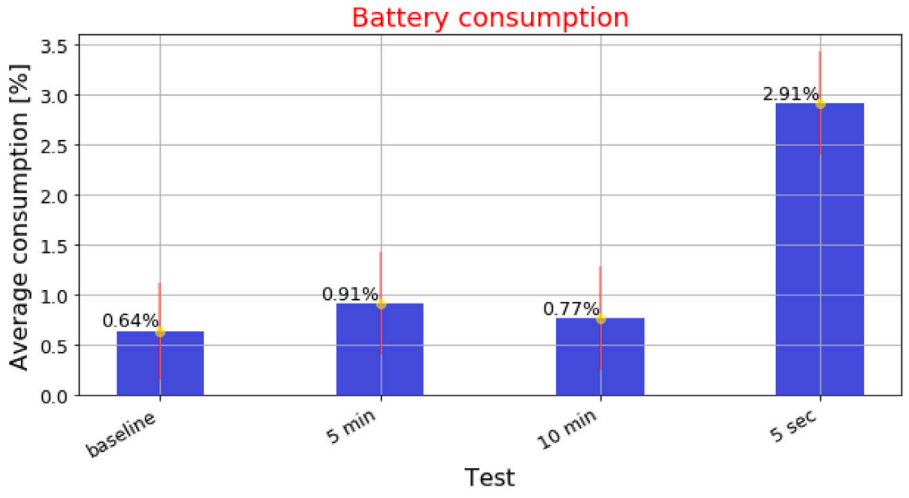
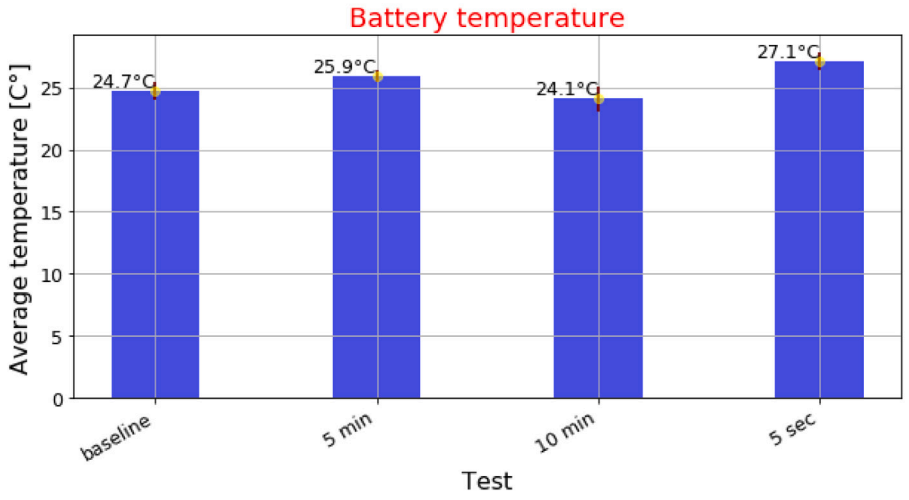


Fig. 8. Battery temperature: For each test, the x-axis represents the time expressed in minutes, the y-axis represents the battery level in percentage, and the second y-axis represents the temperature in Celsius. The red line refers to temperature, while the blue line refers to battery level. Baseline refers to a test where no HTTP requests are performed. Fig A refers to the test in which HTTP requests are performed every 5 min, Fig B represents the baseline, Fig C refers to the test in which HTTP requests are performed every 10 min and Fig D refers to the test in which HTTP requests are performed every 5 s.

It is clear from the graphs that the frequency of sending HTTP requests, has an impact on device temperature, increasing the battery draining. In order to explain the results obtained in Fig. 8, in terms of the average consumption and average temperature every 20 min, Bar plots are shown in Fig. 9 where y-axis represents the average consumption in percentage, while the x-axis the tests. Table 2 summarizes the results. To optimize this aspect, one would need to have continuous control over the temperature of the device and change the frequency of HTTP requests (also increasing or decreasing the throughput of data sending) in order to have over time an average temperature equal to the ambient temperature.



(a) Average consumption every 20 minutes



(b) Average temperature every 20 minutes

Fig. 9. Bar plot temperature study: Fig. 9(a) shows the bar plot relative to average battery consumption in all tests performed. *Y*-axis represents the average consumption in percentage, while the *x*-axis the tests; Fig. 9(b) shows the bar plot relative to the average temperature in all tests performed. *Y*-axis represents the average temperature in Celsius, while the *x*-axis the tests. For each bar, the exact value is reported on the top of the bar.

Specifically, the application monitors the ambient and the device temperature. These values are sent to the processing unit where, having defined a threshold above which it should not rise relative to the ambient temperature, it sends the configuration parameters to the workflow controller, which will set them in the transmission unit. If the device temperature is above the threshold the frequency of HTTP requests is set to 5 requests/hour.

Comparing the proposed approach with similar architectures using smartwatches, we can easily ascertain its place in the spectrum of monitoring solutions. These solutions with the most critical parameters for comparison are shown in Table 3, which shows that although there are solutions that provide daily monitoring, none allow this while ensuring continuity and real-time updating of results. In [58], the smartwatch's battery life is sustained for 16 h by alternating two-minute recordings every 10 min. In contrast, in [13], a battery lifespan of 12 h is achieved by alternating periods when the smartwatch is being worn and periods when it is not. In both solutions, daily monitoring is ensured but at the expense of monitoring continuity. Conversely, in [59], a solution is presented that allows continuous 12-hour monitoring in which, however, data transfer is postponed when the wearable device is put on charge. In this solution, the immediate data transfer is completely penalized, destroying the possibility of real-time patient monitoring. The PDRMA solution, in contrast to literature solutions, allows up to 12 h of continuous, real-time patient monitoring.

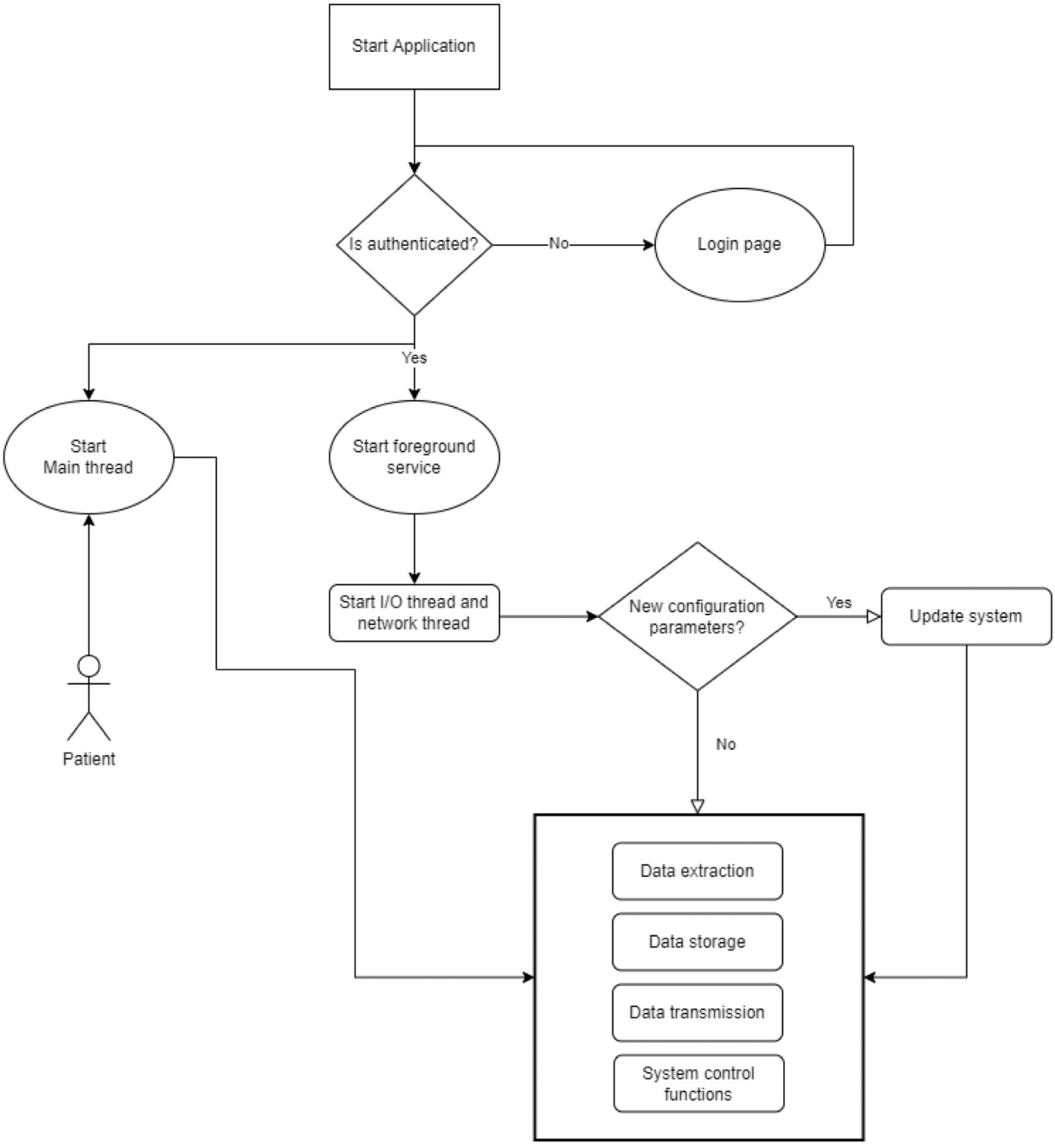


Fig. 10. Smartphone application flow chart diagram.

4.2. Smartphone application

The smartphone application is used to store smartwatch data in order to send them to the cloud. Additionally, it implements audio and video sections which in turn are stored in local storage and then sent to the cloud. To avoid data loss and obtain and make the system stable, the application implements multithread operations. There are three concurrent threads: the main thread for application level, the I/O thread for database interaction, and the network thread for data transmission.

As with the smartwatch application, a foreground service has been implemented, in this case, the foreground service handles the BLE communication with the smartwatch and the HTTP communication with the cloud service. The logic of the application is shown in Fig. 10. The application check if the user is authenticated. Following, if authenticated, a thread for data transmission

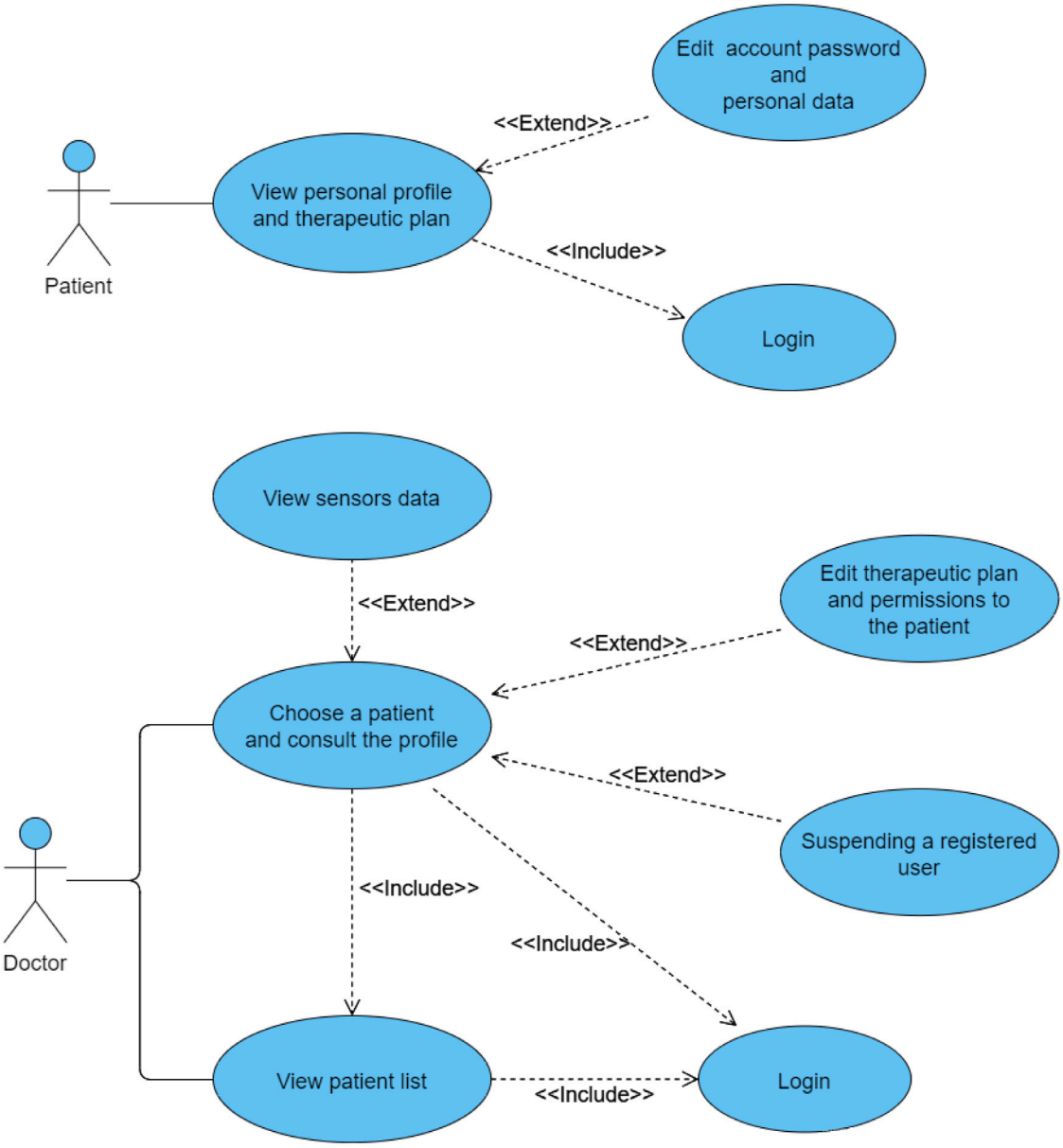


Fig. 11. Web Portal Use Case Diagram: the main entities are the user and the doctor. The include relationship adds additional functionality not specified in the base use case while the extend relationship describes additional behavior that can incrementally augment the behavior of the base use case.

(network thread) and a thread for database operations (I/O thread) are initialized. The main thread handles the application level. If a parameter update is notified, sends configuration parameters to the smartwatch, the system is updated and operations are resumed.

Transmission methods and data storage mirror the logic presented in Section 2.2.

4.3. Parkinson cloud system

The cloud system is composed by Web portal, API, and database. Each component is located on a different server and the API acts as a bridge between the systems and the database. In order to transmit patient data between systems, for security reasons, each patient is assigned a commercial code of 16 bytes. This avoids the transmission of personal data to authorize services access. The logic of the cloud system mirrors the logic presented in Section 2.3. In PDRMA for fog and edge data, we consider audio and video acquired by smartphone and data acquired by smartwatch while, for configuration parameters we consider therapeutic plan data.

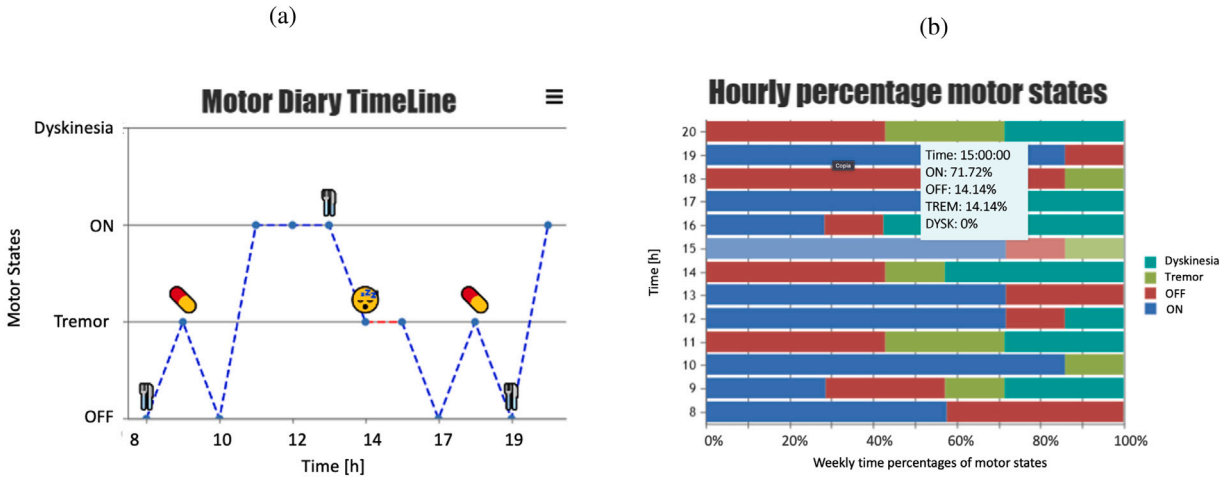


Fig. 12. Fig. 12(a) represents the timeline of the motor diary. *X*-axis represents the daytime, while the *y*-axis is the motor state. Each time may have an associated action and, in the plot, it is represented by an emoticon (meal, drug, and sleep). The red line represents the window time in which the user sleeps. Fig. 12(b) represents the weekly distribution of motor states. For example, the context menu shows that in one week at 3:00:00 pm, the user 71.72% percent of the time entered into the motor diary that perceived the state of ON.

Regarding the Web portal, Fig. 11 shows a use case diagram, which explains its logic: the physician, upon login, can view the list of patients and therefore, by selecting one, can view the data collected by the smartwatch, the personal data and the therapeutic plan with the possibility to modify the latter. Moreover, he could suspend a registered patient.

While the user, upon login, can view the personal profile (with the possibility of editing it) and the therapeutic plan.

An example of the patient details viewable by the doctor is shown in Fig. 12: the doctor can easily view the timeline of the user's motor states and their weekly statistics.

4.4. PDRMA reliability analysis

The architecture, depending on the use case, may rely on only edge or cloud systems. Indeed, choosing to implement a complex architecture composed of an edge, fog, and cloud system make it possible to compartmentalize any issues that arise while using the monitoring system.

Edge system. Common problems with the edge system are related to connectivity for sensor data transmission and battery consumption. The smartwatch enables transmission via two protocols BLE and HTTP. Thus, in case of network problems, data is sent to the fog system via BLE. The fog system subsequently sends the data to the cloud. In case of inability to transmit the data with both protocols, the smartwatch stores the data in its internal memory of 16 GB. If the smartwatch collects 24 MB/h of data, it will collect 288 MB of data in 12 h, so we will have a large margin before it runs out of memory.

Fog system. The fog system ensures that even if the HTTP protocol of the edge system is unavailable, the doctor will still be able to view data in real-time, thanks to transmission via BLE between the smartwatch and the smartphone. Also, in case the smartwatch runs out of battery or has a malfunction, the patient will still be able to consult the therapeutic plan and fill out the motor diary on the smartphone. However, the smartphone may also experience battery drain or connectivity problems, but data are retained thanks to the large local storage.

Cloud system. In case of cloud system failure, data are preserved by the internal storage of the fog and edge system. Restored the cloud service, the system can automatically recover pending transmissions and restore regular functionality.

5. Conclusions

In this paper, we proposed a general IoT architecture for real-time and continuous monitoring of patients in healthcare applications. Following a thorough requirements analysis and the identification of the main issues and technological limitations, we designed all the components (edge, fog, and cloud) and the architecture model needed for the continuous long-term monitoring of frail people and patients.

As analyzed in this work, one of the main limitations is related to wearable devices, and in particular, smartwatches, when extended time window monitoring is an application requirement, due to their limited battery life. The problem becomes more pronounced if all sensors simultaneously collect data at high sampling rates. One solution is to decrease the sampling rate, but this is not always possible, especially for medical applications. Indeed, it is unthinkable to decrease the sampling rate without losing

crucial information for all data types. To solve this problem, we showed that by using a stable and energy-efficient transmission technology (i.e. BLE) and including a proper data transmission policy, battery life can be significantly extended.

Eventually, since there are applications that require a direct connection between the edge and the cloud systems, we analyzed also this scenario. This type of architecture translates into implementation susceptible to higher battery consumption due to the required transmission methods. We demonstrated that the frequency of HTTP requests can impact the device temperature and this translates into increased battery drainage. Daily battery life can be improved through the implementation of a temperature control algorithm that varies the frequency of HTTP requests thereby allowing the device to re-establish the ambient temperature.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] A.M. Rahmani, T.N. Gia, B. Negash, A. Anzanpour, I. Azimi, M. Jiang, P. Liljeberg, Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach, Future Gener. Comput. Syst. 78 (2018) 641–658.
[2] M. Hassanalieragh, A. Page, T. Soyata, G. Sharma, M. Aktas, G. Mateos, B. Kantarci, S. Andreescu, Health monitoring and management using internet-of-things (IoT) sensing with cloud-based processing: Opportunities and challenges, in: 2015 IEEE International Conference on Services Computing, IEEE, 2015, pp. 285–292.
[3] A. Salim, S. Lim, Recent advances in noninvasive flexible and wearable wireless biosensors, Biosens. Bioelectron. 141 (2019) 111422.
[4] G.R. Burmester, Rheumatology 4.0: big data, wearables and diagnosis by computer, Ann. Rheum. Dis. 77 (7) (2018) 963–965.
[5] M.M. Rodgers, G. Alon, V.M. Pai, R.S. Conroy, Wearable technologies for active living and rehabilitation: Current research challenges and future opportunities, J. Rehabil. Assist. Technol. Eng. 6 (2019) 2055668319839607.
[6] L. Segura Anaya, A. Alsadoon, N. Costadopoulos, P. Prasad, Ethical implications of user perceptions of wearable devices, Sci. Eng. Ethics 24 (1) (2018) 1–28.
[7] I. cheol Jeong, D. Bychkov, P.C. Searson, Wearable devices for precision medicine and health state monitoring, IEEE Trans. Biomed. Eng. 66 (5) (2018) 1242–1258.
[8] B. Lin, Wearable smart devices for P4 medicine in heart disease: Ready for medical cyber-physical systems? OMICS: J. Integr. Biol. 23 (5) (2019) 291–292.
[9] R.-C. Qian, Y.-T. Long, Wearable chemosensors: A review of recent progress, ChemistryOpen 7 (2) (2018) 118–130.
[10] I. Sim, Mobile devices and health, N. Engl. J. Med. 381 (10) (2019) 956–968.
[11] Q. Zhang, Y. Wang, G. Cheng, Z. Wang, D. Shi, Research on warehouse environment monitoring system based on wireless sensor network, in: 2014 9th IEEE Conference on Industrial Electronics and Applications, IEEE, 2014, pp. 1639–1644.
[12] Y. Zhang, H. Liu, X. Su, P. Jiang, D. Wei, Remote mobile health monitoring system based on smart phone and browser/server structure, J. Healthcare Eng. 6 (4) (2015) 717–738.
[13] M. Kheirkhahan, S. Nair, A. Davoudi, P. Rashidi, A.A. Wanigatunga, D.B. Corbett, T. Mendoza, T.M. Manini, S. Ranka, A smartwatch-based framework for real-time and online assessment and mobility monitoring, J. Biomed. Inform. 89 (2019) 29–40.
[14] S. Rautmare, D. Bhalerao, Mysql and NoSQL database comparison for IoT application, in: 2016 IEEE International Conference on Advances in Computer Applications, ICACA, IEEE, 2016, pp. 235–238.
[15] R. Chopade, V. Pachghare, Mongodb indexing for performance improvement, in: ICT Systems and Sustainability, Springer, 2020, pp. 529–539.
[16] World Health Organization, Neurological Disorders: Public Health Challenges, World Health Organization, 2006.
[17] K.A. Jellinger, G. Logroscino, G. Rizzo, M. Copetti, S. Arcuti, D. Martino, A. Fontana, Accuracy of clinical diagnosis of parkinson disease: A systematic review and meta-analysisauthor response, Neurology 87 (2) (2016) 237–238.
[18] E.R. Dorsey, F.P. Vlaanderen, L.J. Engelen, K. Kieburtz, W. Zhu, K.M. Biglan, M.J. Faber, B.R. Bloem, Moving parkinson care to the home, Mov. Disorders 31 (9) (2016) 1258–1262.
[19] L. di Biase, G. Tinkhauser, E. Martin Moraud, M.L. Caminiti, P.M. Pecoraro, V. Di Lazzaro, Adaptive, personalized closed-loop therapy for Parkinson's disease: biochemical, neurophysiological, and wearable sensing systems, Exp. Rev. Neurotherapeut. 21 (12) (2021) 1371–1388.
[20] D.A. Heldman, D.E. Filipkowski, D.E. Riley, C.M. Whitney, B.L. Walter, S.A. Gunzler, J.P. Giuffrida, T.O. Mera, Automated motion sensor quantification of gait and lower extremity bradykinesia, in: 2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, IEEE, 2012, pp. 1956–1959.
[21] J. Stamatakis, J. Ambroise, J. Crémers, H. Sharei, V. Delvaux, B. Macq, G. Garraux, Finger tapping clinimetric score prediction in Parkinson's disease using low-cost accelerometers, Comput. Intell. Neurosci. 2013 (2013).
[22] S. Summa, J. Tosi, F. Taffoni, L. Di Biase, M. Marano, A.C. Rizzo, M. Tombini, G. Di Pino, D. Formica, Assessing bradykinesia in Parkinson's disease using gyroscope signals, in: 2017 International Conference on Rehabilitation Robotics, ICORR, IEEE, 2017, pp. 1556–1561.
[23] L. di Biase, L. Raiano, M.L. Caminiti, P.M. Pecoraro, V. Di Lazzaroo, Artificial intelligence in Parkinson's disease—symptoms identification and monitoring, in: Augmenting Neurological Disorder Prediction and Rehabilitation using Artificial Intelligence, Elsevier, 2022, pp. 35–52.
[24] P. Angeles, M. Mace, M. Admiraal, E. Burdet, N. Pavese, R. Vaidyanathan, A wearable automated system to quantify parkinsonian symptoms enabling closed loop deep brain stimulation, in: Annual Conference Towards Autonomous Robotic Systems, Springer, 2016, pp. 8–19.
[25] L. Di Biase, S. Summa, J. Tosi, F. Taffoni, M. Marano, A. Cascio Rizzo, F. Vecchio, D. Formica, V. Di Lazzaro, G. Di Pino, et al., Quantitative analysis of bradykinesia and rigidity in Parkinson's disease, Front. Neurol. 9 (2018) 121.
[26] T. Endo, R. Okuno, M. Yokoe, K. Akazawa, S. Sakoda, A novel method for systematic analysis of rigidity in Parkinson's disease, Mov. Disorders: Off. J. Mov. Disorder Soc. 24 (15) (2009) 2218–2224.
[27] Y. Kwon, S.-H. Park, J.-W. Kim, Y. Ho, H.-M. Jeon, M.-J. Bang, S.-B. Koh, J.-H. Kim, G.-M. Eom, Quantitative evaluation of parkinsonian rigidity during intra-operative deep brain stimulation, Biomed. Mater. Eng. 24 (6) (2014) 2273–2281.
[28] L. Raiano, G. di Pino, L. di Biase, M. Tombini, N.L. Tagliamonte, D. Formica, Pdmeter: A wrist wearable device for an at-home assessment of the Parkinson's disease rigidity, IEEE Trans. Neural Syst. Rehabil. Eng. 28 (6) (2020) 1325–1333.
[29] G. Deuschl, P. Krack, M. Lauk, J. Timmer, Clinical neurophysiology of tremor, J. Clin. Neurophysiol. 13 (2) (1996) 110–121.

[30] G. Di Pino, D. Formica, J.-M. Melgari, F. Taffoni, G. Salomone, L. di Biase, E. Caimo, F. Vernieri, E. Guglielmelli, Neurophysiological bases of tremors and accelerometric parameters analysis, in: 2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob), IEEE, 2012, pp. 1820–1825.
[31] L. Di Biase, J.-S. Brittain, S.A. Shah, D.J. Pedrosa, H. Cagnan, A. Mathy, C.C. Chen, J.F. Martín-Rodríguez, P. Mir, L. Timmerman, et al., Tremor stability index: a new tool for differential diagnosis in tremor syndromes, Brain 140 (7) (2017) 1977–1986.
[32] S.T. Moore, H.G. MacDougall, W.G. Ondo, Ambulatory monitoring of freezing of gait in Parkinson's disease, J. Neurosci. Methods 167 (2) (2008) 340–348.
[33] J.C. Schlachetzki, J. Barth, F. Marxreiter, J. Gossler, Z. Kohl, S. Reinfelder, H. Gassner, K. Aminian, B.M. Eskofier, J. Winkler, et al., Wearable sensors objectively measure gait parameters in Parkinson's disease, PLoS One 12 (10) (2017) e0183989.
[34] J. Tosi, S. Summa, F. Taffoni, L. di Biase, M. Marano, A.C. Rizzo, M. Tombini, E. Schena, D. Formica, G. Di Pino, Feature extraction in sit-to-stand task using M-IMU sensors and evaluatiton in Parkinson's disease, in: 2018 IEEE International Symposium on Medical Measurements and Applications (MeMeA), IEEE, 2018, pp. 1–6.
[35] A. Suppa, A. Kita, G. Leodori, A. Zampogna, E. Nicolini, P. Lorenzi, R. Rao, F. Irrera, L-DOPA and freezing of gait in Parkinson's disease: Objective assessment through a wearable wireless system, Front. Neurol. 8 (2017) 406.
[36] C.M. Letizia, S.S. Ahmar, L. Ricci, V. Di Lazzaro, et al., Gait analysis in Parkinson's disease: An overview of the most accurate markers for diagnosis and symptoms monitoring, 2020.
[37] À. Bayés, A. Samá, A. Prats, C. Pérez-López, M. Crespo-Maraver, J.M. Moreno, S. Alcaine, A. Rodriguez-Molinero, B. Mestre, P. Quispe, et al., A "HOLTER" for Parkinson's disease: Validation of the ability to detect on-off states using the REMPARK system, Gait Posture 59 (2018) 1–6.
[38] J. Barth, J. Klucken, P. Kugler, T. Kammerer, R. Steidl, J. Winkler, J. Hornegger, B. Eskofier, Biometric and mobile gait analysis for early diagnosis and therapy monitoring in Parkinson's disease, in: 2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, IEEE, 2011, pp. 868–871.
[39] M. Ricci, G. Di Lazzaro, A. Pisani, N.B. Mercuri, F. Giannini, G. Saggio, Assessment of motor impairments in early untreated parkinson's disease patients: the wearable electronics impact, IEEE J. Biomed. Health Inf. 24 (1) (2019) 120–130.
[40] T. Arroyo-Gallego, M.J. Ledesma-Carbayo, A. Sánchez-Ferro, I. Butterworth, C.S. Mendoza, M. Matarazzo, P. Montero, R. López-Blanco, V. Puertas-Martin, R. Trincado, et al., Detection of motor impairment in Parkinson's disease via mobile touchscreen typing, IEEE Trans. Biomed. Eng. 64 (9) (2017) 1994–2002.
[41] R. LeMoyne, T. Mastroianni, M. Cozza, C. Coroian, W. Grundfest, Implementation of an iphone for characterizing Parkinson's disease tremor through a wireless accelerometer application, in: 2010 Annual International Conference of the IEEE Engineering in Medicine and Biology, IEEE, 2010, pp. 4954–4958.
[42] S. Cohen, L.R. Bataille, A.K. Martig, Enabling breakthroughs in Parkinson's disease with wearable technologies and big data analytics, Mhealth 2 (2016).
[43] P. Pierleoni, A. Belli, O. Bazgir, L. Maurizi, M. Paniccia, L. Palma, A smart inertial system for 24h monitoring and classification of tremor and freezing of gait in Parkinson's disease, IEEE Sens. J. 19 (23) (2019) 11612–11623.
[44] S. Patel, K. Lorincz, R. Hughes, N. Huggins, J. Growdon, D. Standaert, M. Akay, J. Dy, M. Welsh, P. Bonato, Monitoring motor fluctuations in patients with Parkinson's disease using wearable sensors, IEEE Trans. Inf. Technol. Biomed. 13 (6) (2009) 864–873.
[45] G. Grimaldi, M. Manto, Tremor: from pathogenesis to treatment, Synth. Lect. Biomed. Eng. 3 (1) (2008) 1–212.
[46] T. Zhang, G. Wei, Z. Yan, M. Ding, C. Li, H. Ding, S. Xu, Quantitative assessment of Parkinson's disease deficits, Chin. Med. J. 112 (09) (1999) 812–815.
[47] F.L. Darley, A.E. Aronson, J.R. Brown, Differential diagnostic patterns of dysarthria, J. Speech Hear. Res. 12 (2) (1969) 246–269.
[48] J.A. Robbins, J.A. Logemann, H.S. Kirshner, Swallowing and speech production in Parkinson's disease, Ann. Neurol. 19 (3) (1986) 283–287.
[49] A.H. Omre, S. Keeping, Bluetooth low energy: wireless connectivity for medical monitoring, J. Diabetes Sci. Technol. 4 (2) (2010) 457–463.
[50] A.R. Fekr, K. Radecka, Z. Zilic, Design and evaluation of an intelligent remote tidal volume variability monitoring system in e-health applications, IEEE J. Biomed. Health Inf. 19 (5) (2015) 1532–1548.
[51] X. Fafoutis, A. Vafeas, B. Janko, R.S. Sherratt, J. Pope, A. Elsts, E. Mellios, G. Hilton, G. Oikonomou, R. Piechocki, et al., Designing wearable sensing platforms for healthcare in a residential environment, EAI Endorsed Trans. Pervasive Health Technol. 3 (12) (2017).
[52] J.P. Amaro, S. Patrão, F. Moita, L. Roseiro, Bluetooth low energy profile for MPU9150 IMU data transfers, in: 2017 IEEE 5th Portuguese Meeting on Bioengineering, ENBENG, IEEE, 2017, pp. 1–4.
[53] A. Davoudi, A.A. Wanigatunga, M. Kheirkhahan, D.B. Corbett, T. Mendoza, M. Battula, S. Ranka, R.B. Fillingim, T.M. Manini, P. Rashidi, Accuracy of samsung gear s smartwatch for activity recognition: Validation study, JMIR MHealth UHealth 7 (2) (2019) e11270.
[54] R.J. Elble, Tremor: clinical features, pathophysiology, and treatment, Neurol. Clin. 27 (3) (2009) 679–695.
[55] J. Jankovic, Motor fluctuations and dyskinesias in Parkinson's disease: clinical manifestations, Mov. Disorders: Off. J. Mov. Disorder Soc. 20 (S11) (2005) S11–S16.
[56] P. Deutsch, DEFLATE compressed data format specification version 1.3, Tech. rep., 1996.
[57] F.J. González-Cañete, E. Casilari, A feasibility study of the use of smartwatches in wearable fall detection systems, Sensors 21 (6) (2021) 2254.
[58] R. Wu, D. Liaqat, E. de Lara, T. Son, F. Rudzicz, H. Alshaer, P. Abed-Esfahani, A.S. Gershon, et al., Feasibility of using a smartwatch to intensively monitor patients with chronic obstructive pulmonary disease: prospective cohort study, JMIR MHealth UHealth 6 (6) (2018) e10046.
[59] A.L. Beukenhorst, M.J. Parkes, L. Cook, R. Barnard, S.N. van der Veer, M.A. Little, K. Howells, C. Sanders, J.C. Sergeant, T.W. O'Neill, et al., Collecting symptoms and sensor data with consumer smartwatches (the Knee OsteoArthritis, Linking Activity and Pain Study): protocol for a longitudinal, observational feasibility study, JMIR Res. Protocols 8 (1) (2019) e10238.
[60] V. LeBaron, R. Alam, R. Bennett, L. Blackhall, K. Gordon, J. Hayes, N. Homdee, R. Jones, K. Lichti, Y. Martinez, et al., Deploying the behavioral and environmental sensing and intervention for cancer smart health system to support patients and family caregivers in managing pain: Feasibility and acceptability study, JMIR Cancer 8 (3) (2022) e36879.