Polytechnic of Turin

Master of science program Biomedical engineering

Master Thesis

IoT and Telemedicine for anesthesia practices enabled by an Android application with cloud integration

Supervisors

Prof. Danilo Demarchi

Dr. MER Sandro Carrara

Co-Supervisor

PhD Francesca Stradolini

Candidate

Nadia Tamburrano

Academic Year 2017/2018

Alla mia famiglia

1

[1]This thesis was conducted in the Laboratory of Integrated Systems (LSI) of École polytechnique fédérale de Lausanne (EPFL)

Abstract

Internet of Things (IoT) is a new paradigm that is receiving a noticeable impact on different aspects of every-day life. Powerful networks can be fuelled by IoT solutions, interconnecting several heterogeneous objects through the Internet. This would guarantee a constant flux of exchanging information and sharing possibilities.

One of the most promising field where IoT is involved is in health-care. *medical Internet of Things* (mIoT) is really promising for different health-care environments, such as home-monitoring, hospital management, disease treatments, diagnosis and monitoring. Furthermore, the recent advances in *mobile Health* (mHealth) and Cloud Computing are paving the way to innovative portable-mIoT systems for remotely delivering and accessing health-care services, enabling telemedicine.

The aim of this thesis, conducted in the Laboratory of Integrated Systems (LSI) of École polytechnique fédérale de Lausanne (EPFL), is to present a mIoT cloud-based network for anesthesia on-line monitoring. Continuous monitoring in anesthesia is essential to guarantee adequate sedation levels. The developed system allows the anesthesiologist to simultaneously monitor up to three drugs concentration in patients' blood trough an Android application and to share them through a cloud-based solution. More than one patient can be monitored as the same time. The developed Android app continuously visualizes the data received via Wi-Fi from a *Therapeutic Drug Monitoring* (TDM) platform directly connected on the patient. A web application has been also realized to allow the medical doctor to remotely access and visualize the shared data.

The system can be used for different monitoring applications as well, thanks to the possibility to interface with any medical device that can wirelessy send the measured data.

Sommario

Internet of Things (IoT) è un recente paradigma che sta influenzando diversi aspetti della vita quotidiana. Grazie all'IoT, potenti reti possono essere create, in cui oggetti eterogenei sono interconnessi attraverso Internet, scambiando continuamente informazioni e coordinando decisioni.
Uno tra i più promettenti campi di applicazione dell'IoT è la medicina. *medical Internet of Things* (mIoT) può portare a miglioramenti in diversi campi medici, tra cui home-monitoring, management ospedaliero, nel trattamento delle malattie, diagnosi e nel monitoraggio. Inoltre, l'avanzamento in *mobile Health* (mHealth) e in Cloud Computing stanno portando allo sviluppo di innovativi e portatili sistemi mIoT per accedere e fornire i servizi medici anche da remoto, rendendo possibili applicazioni di telemedicina.
Lo scopo di questa tesi, condotta nel Laboratorio dei Sistemi Integrati(LSI) dell'École polytechnique fédérale de Lausanne (EPFL), è presentare una rete mIoT basata su cloud per il monitoraggio online durante l'anestesia. Monitorare continuamente durante l'anestesia è fondamentale per garantire adeguati livelli di sedazione. Il sistema sviluppato permette al medico anestetista di monitorare simultaneamente la concentrazione di anestetici nel sangue del paziente, grazie ad un'applicazione Android, e di condividere i dati su una piattaforma cloud. Più pazienti possono essere monitorati simultaneamente.
L'applicazione Android visualizza continuamente i dati ricevuti via Wi-Fi da una piattaforma per *Therapeutic Drug Monitoring* (TDM), connessa al paziente. Un'applicazione web è stata realizzata per consentire ad ogni medico autorizzato

di accedere e visualizzare i dati da remoto. Il sistema può anche essere utilizzato per applicazioni di monitoraggio diverse dall'anestesia, grazie alla possibilità di essere connesso a qualsiasi dispositivo in grado di trasmettere i dati via Wi-Fi.

List of Abbreviations

GA General Anesthesia

TIVA Total Intravenous Anesthesia

TCI Target Controlled Infusion

DOA Depth of Anesthesia

IoT Internet of Things

mIoT Medical Internet of Things

TDM Therapeutic Drug Monitoring

RPi Raspberry Pi

WE Working Electrode

RE Reference Electrode

CE Counter Electrode

CV Cyclic Voltammetry

DPV Differential Pulse Voltammetry

CA Chronoamperometry

IP Internet Protocol

TCP Transmission Control Protocol

Contents

List of Figures

Chapter 1

Introduction

Internet of Things (IoT) is becoming more and more important in daily lives. Nowadays the interconnection between objects is not only a desirable property, but it is a requirement for any competitive system currently in the market [28]. Data sharing can now be enabled by several types of smart-devices, *e.g.* smart-watches, tablets, smart-phones and others, thanks to their possibility to connect to the Internet. In addition, by exploiting on-line Cloud platforms, data can be accessed by everyone, at any time and from any place. These portable smart-devices play an increasing relevant role in IoT systems, thanks to their small dimensions, light weight, low costs and great number of integrated sensors [5].

During the last few years, IoT research has evolved widely in the field of healthcare. The existence of wearable, wireless and low power sensors, placed in strategical parts of the body, enables the creation of body networks, leading to the creation of monitoring systems for personalized medicine strategies [29]. Different types of sensors are able to simultaneously monitor different physiological parameters and to share the measured data by exploiting wireless connections [30]. Furthermore, with a secure storage, *e.g.* on cloud or on physical devices, the data measured by the patients' sensors can be collected and accessed by allowed medical doctors so that an appropriate and safe remote monitoring is ensured [31].

IoT-based systems can be adopted not only for remote monitoring in teleconsulting

and telemedicine applications, but also for providing real-time feedback in hospitals or in health-care clinical centers [29], paving the way towards the realization of a *Smart Hospital*, an interconnected environment with smart services, quickly accessible by physicians and patients [32].

Thanks to these systems, many improvements are achieved: (i) highly specialized

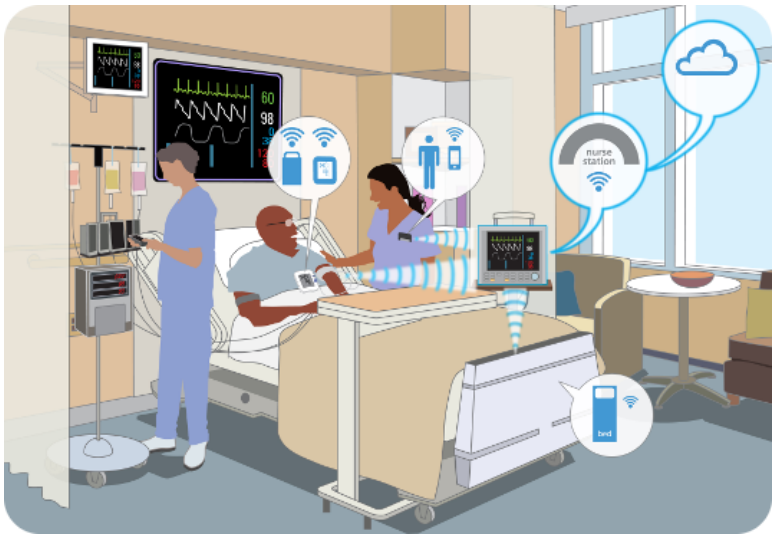

Figure 1.1: Sketch of the future Smart Hospital [1].

medical staff can access the patient's measured data at any time in case of critical health conditions, (ii) the quality of the health-care monitoring system is improved and can be guaranteed also in rural area, (iii) the costs for treatments and travels are reduced as a consequence of the continuous monitoring that ensures a personalized dose adjustment and the possibility of remote monitoring enabled by cloud support [30].

IoT health-care systems can have a great impact in several different environments, ranging from home-monitoring to management of hospital wards as well as for the remote control of patient's treatments, for diagnosis purposes and for monitoring of endogenous and exogenous compounds (*e.g.* to analyse the effects of drug therapies).

In this thesis, it will be described an IoT-based system to be adopted in anesthesia practices. The importance of such developed system relies on the fact that an

effective sedation is achieved only if a right balance of the cocktail drugs is ensured and personalized for each patient. Hence a continuous and simultaneous monitoring of the anesthetics' concentration in patient's veins is of vital importance to keep the desired level of sedation during the whole surgery [23].

The here presented IoT monitoring architecture has been developed in Android and based on Wi-Fi communication. The main core of the architecture is an Android app developed for the anesthesiologist. It offers an immediate and user-friendly visualization of the parameters received from TDM system directly connected with the patient. The TDM system, for the analysed anesthesia application, consists of electrochemical sensors able to detect the anesthesia level in the patient. A *Raspberry Pi (RPi)*-driven electronic system is interfaced with these sensors and it is in charge of sending the measured data to the doctor's tablet/smart-phone, where the Android application is running. A smart-watch, worn by the anesthesiologist and paired with the tablet, receives alert notifications by the Android app in case of critical clinical conditions (*e.g.* a measured values is out of its safe range). The system is enriched by a Cloud network, which guarantees a secure data storage and allows a remote visualization. The data on the Cloud are accessible at any time and from any place through a Web application.

The system can connect several different anesthetized patients at the same time so that the responsible anesthesiologist can continuously monitor their sedation level during the surgeries. This is beneficial if considering the fact that more than one surgery can be performed at the same time; therefore, this architecture can guarantee the efficient monitoring of each patient [33]. This would reduce the waiting list for surgical interventions.

Thanks to the flexibility of the system, the presented mIoT network can be also adopted in other health-monitoring applications where medical devices are able to send the measured data through Wi-Fi.

Chapter 2

Application: Continuous Monitoring in Anesthesia

The correct administration of therapeutic drugs is a critical issue due to the variability among patients' metabolism. A personalized therapy has to be guaranteed to increase the therapeutic effect of the drug and to reduce undesired side effects. This is especially valuable in case of difficult-to-handle and critical medication like anesthetics, where a specific balance between drugs must be maintained to guarantee the desired level of sedation. In this scenario continuous monitoring is crucial to preserve drugs' efficacy and to avoid awareness or toxicity phenomena [3].

2.1 General Anesthesia

General Anesthesia (GA) is defined as a drug-induced condition which makes the patient unresponsive to painful surgical stimuli [34]. It produces loss of consciousness, analgesia, amnesia and akinesia while maintaining cardiovascular, thermo-regulatory and respiratory stability [35].

Although the wide use of anesthetics, their specific action is not completely clear and well-known. As it is shown in (Fig. 2.1), the anesthetic effect manifests at different body's scales, ranging from molecular dimension to the system and

behavioral one. It is known that at microscopic level the anesthetics interact with neuronal membrane proteins that work as ion channel and neurotrasmitter receptors, causing structural or dynamic changes, but it is not known which of these molecules are particularly affected. [2]. Many studies have been focused on the anesthetics' interference with ion gates involved in the secretion of neurotrasmitters such as $GABA_A$ (Gama-Amino-Butiric-Acid) receptors, which regulate anxiety, vigilance, memory and muscle tension, nicotinic acetylchiline receptors (nACh) and NMDA (N-Methyl-D-Aspartate) receptors, which are involved in the pain processes. In addition these drugs may act on the GPCRs (G protein-coupled receptors), which influence the receptors of dopamine, acetylcholine, norephrine and opioids. More recently it has been shown an inibition effect on the "background potassium channels", which regulate cellular excitability [34].

Nowadays, even if it has been observed a significant decrease of the mortality

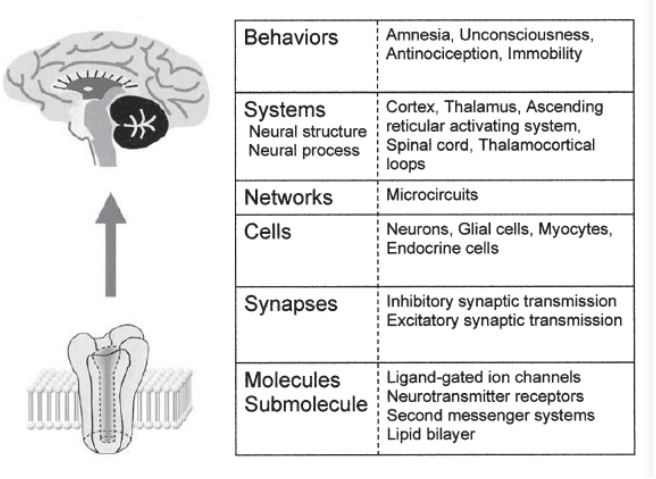

Figure 2.1: Anesthetics' effects on different biological organizational level [2].

caused by anesthetics drugs, thanks to the improvements in drug's monitoring techniques, risks associated to anesthesia and the side effects still represent a critical matter [2]. Among the severe complications that could rise as a consequence or during a surgery, there are awareness, myocardial infarction, pneumonia, allergy, pulmonary embolism to death, while some minor side effects post-surgery are nausea and *Postoperative Cognitive Dysfunction (POCD)*.

In particular, during the last few years, the side effect of intraoperative awareness, which occurs in up to 1 to 2 per 1000 patients [36], has been object of study in many researches, becoming a significant problem associated to general anesthesia [2]. Anesthesia awareness occurs when the state of unconsciousness of the patient during the surgery is not maintained; hence the patient may be feeling pain while not able to move and give an alarm to the medical doctor. As a consequence of this, the patient can have memory of the surgical experience once recovered leading to psychological disturbs [37], such as depression, anxiety and flashbacks. To avoid such undesired anesthesia effects due to over- or under-dosage of drugs, a continuous monitoring during the surgery represents a valid solution to prevent this undesirable inconvenience [36].
General anesthesia can be performed through two different techniques, which substantially differ in the way the anesthestic drugs are delivered to the patient and enter in the circulation. The anesthesist can decide whether to use the *Inhalation Anesthetics Technique*, in which volatile anesthetics are used and administered trough a vaporizer, or the *Total Intravenous Anesthesia (TIVA))*, in which anesthetic drugs are directly injected into the blood circulation [38].
The system proposed in this work is designed for the second technique, which is going to be analysed in detail.

2.1.1 Total Intravenous Anesthesia

For many years the inhalation anesthetics technique was considered more advantageous compared to TIVA.
Indeed, the vaporizer ensures an easy monitoring of the drug concentration in the blood for two reasons:

- the absorption of anesthetics gradually decreases as the equilibrium between alveolar and pulmonary capillary partial pressured is reached. Thanks to this, the anesthesiologist can control drugs' administration by setting a maximum partial pressure that can be achieved;

- the inhalation process can be monitored trough respiratory gas monitoring, which allows the anesthesiologist to measure the expired concentration of the inhaled agent.

On the contrary, when TIVA was introduced, there was not a mature monitoring system able to accurately control drugs' concentration in the blood during the infusion process [38]. In addition, the anesthetic agents that were available, as the *sodium thiopental*, caused many side effects and were active too long.
However, with time, other parameters, as the *Bispectral Index (BIS)* , have been kept under control while performing TIVA anesthesia. Hence, this novel approach has gained popularity in recent times and today it is considered an established technique, comparable to the traditional inhalation anesthesia approach [38]. This rapid rise is mainly due to the large number of studies that has been conducted about this technique, which has been focused on two main fields:

- availability of new drugs and opioids which are more efficient and with less side effects [39]. An important step has been made with the advent of *Propofol* (2,6-diisopropylphenol) in TIVA procedures. The advantages of this intravenous drug are related to its short action, which makes the after-surgery recovery faster, even in case of long infusions [40]. Nowadays, during TIVA, propofol is often associated to other drugs, such as *Remifentanil*, an esterase-metabolized opioid, and *Midazolam*, an esterase-metabolized benzodiazepine. This combination is efficient thanks to the different mechanism of action of each drug, which allows to achieve anesthesia with minimum toxicity and faster recovery [38];

- improvements in pharmacokinetic models and in computer technology, which paved the way towards new delivery and monitoring systems. Pharmacokinetics describes, using mathematics, how a particular drug interacts with the body. Since there is a mathematical relationship between drug's dose and its concentration in plasma, many researchers designed advanced pharmacokinetics models that can calculate the rate of drug dosing during TIVA

procedures [39].

2.1.1.1 Target Controlled Infusion

To achieve and maintain the desired level of sedation through intravenous anesthesia a cocktail of up-to-three drugs has to be injected to the patient. Since the last 30 years this was done manually by the anesthesiologist, with the risk on one hand of toxicity, due to an excessive amount of dose, on the other hand of under-dosing, which could lead to awareness [39].
The advent of *Target Controlled Infusion (TCI)* allowed to overcome some of the problems associated with manually-regulated TIVA. TCI is a computer-assisted intravenous administration of anesthetics which allows induction and maintenance of general anesthesia. As shown in Fig. 2.2, a TCI system is composed by a computer that controls an infusion pump trough which the anesthetics are injected. This system, implementing pharmacokinetic models, automatically determines the initial dose necessary to reach a certain concentration in the blood and the infusion rate to maintain it. The infusion rate is dynamically adjusted in response to clinical signs, as BIS values and EEG signal of the patient, which correlates the anesthetics' concentration to the amount of drug in the patient [41]. It has to be noticed that TCI is a tool to assist the doctor making faster and easier dosages calculations [41]. Therefore, the anesthesiologist firstly has to set the target blood concentration and during the surgery has to monitor the patient, adjusting the target concentration when it is needed [39].

2.2 Continuous Monitoring in Anesthesia

Continuous monitoring in anesthesia is absolutely necessary, considering that the effects of anesthetics show variability between patients, also due to their different genetic predisposition in metabolising the anesthetics, and that the pharmacokinetic models used in TCI systems are mathematical models that can not perfectly predict the drug concentration level since they do not take into

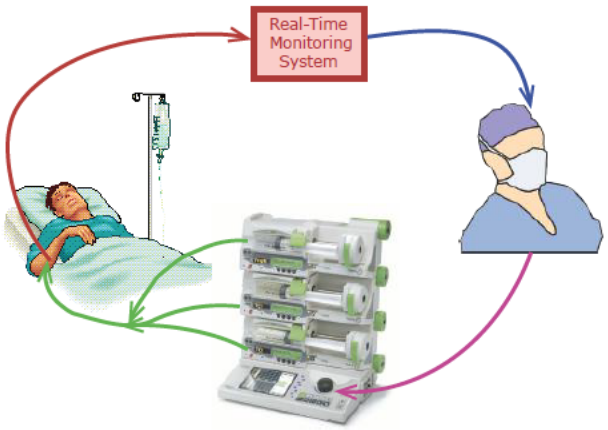

Figure 2.2: Target Controlled Infusion (TCI [3].

account the clinical history of the patient and his/her health conditions, which affects the drug efficacy [42].

An important parameter to assess the efficacy of anesthesia is the *Depth of Anesthesia (DOA)*. The definition of DOA is various and also depends from the anesthetic technique. For the inhalational technique, it is associated to the Minimum Alveolar Concentration (MAC), whereas for the intravenous technique is related to plasma drug concentration of anesthetics [42].

In this context, a problem is to determine accurately the DOA during the surgery. At the moment several methods are available:

- **Clinical Signs**: Traditionally some physiological parameters are used as sign of DOA, including blood pressure and heart rate, which can increase when anesthesia is inadequate [35], heart rate variability, sweating, pupil size, movement and respiration [43]. However these parameters show variability between patients and surgery type [3];

- **EEG monitoring**: Brain electrical activity is correlated to DOA. It has been noticed that EEG signal changes from low amplitude and high frequencies to large amplitude and low frequencies when anesthesia is established correctly [43]. In particular after the administration of anesthetics EEG shows a decrease in beta activity and an increase in alpha and delta activi-

ties with some periods of flatness [35]. Unfortunately also this method is uncertain due to EEG variability between different drugs and patients [3];

- **Bispectral Index**: Bispectral index (BIS) monitoring is a technology which allows the anesthesiologis to follow the anesthesia effects during the surgery [44]. BIS is a parameter obtained with a weighted sum of some EEG features. During the surgery the BI monitoring provides a number from 0 to 100, which reflects the DOA. In particular, a number from 40 to 60 is a sign of an appropriate level of anesthesia [43]. Recently this technique has obtained the approval from Food and Drug Administration (FDA) [44]. However, since it is evaluated in an indirect way, it may be affected by artefacts.

- **Drug Concentration Monitoring**: another recent technique to evaluate the DOA is to measure anesthetic drug concentration in plasma. After the administration the concentration has an instant peak that progressively decreases as the drug goes to the target area. Knowing this, it is possible to monitor the DOA thanks to the pharmacokinetic relationship between the administered dose and the concentration in plasma [43].

Chapter 3

IoT in medical field: State-Of-The-Art

Internet of Things (IoT) is a recent paradigm born kind of ten years ago [6] to define the network of interconnected *Things*, such as physical devices, which interact with each other, cooperating and exchanging data through the *Internet* [45]. As a consequence of this, shared data are available from *anywhere* and at *any time*.

Related to the incoming of the IoT era, the concept of "Big Data" is outlined in parallel. To better understand it, the "three V's" words need to be introduced: (i) *Volume*, intended as a big quantity of data; (ii) *Variety*, meaning data heterogeneity; (iii) *Velocity*, due to data production rate [5].

Since several devices can be interconnected within an IoT network, an essential requirement is to have a reliable and organized structure to handle the big number of exchanged and shared data. The term "Big Data" refers to a huge amount of information, which need to be handled by tools with sufficient storage capacity and with an excellent computational power. To provide these crucial requirements, cloud computing seems to offer a great opportunity.

IoT is expected to gain more and more popularity in the next years, impacting on the everyday-life of the users. By 2020, IoT smart-objects are expected to

reach the number of 212 billions and *Machine-to-Machine (M2M)* traffic is going to constitute up to 45% of the entire Internet traffic. The pervasive impact that IoT systems are going to have in our lives is already evident by observing the fast spread of connected machines in the last 5 years, which registered an increase of 300% according to the McKinsey Global Institute [4].

The wide use of IoT networks represents also a great opportunity from an economical point of view. Its impact is estimated to be in range of 2.7 trillion to 6.2 trillion of dollars by 2025 [4].

An IoT system connects heterogeneous objects; hence it is of crucial importance to provide well-realized and flexible architectures. At the moment, even if a large number of architectures have been already realized, there is still any reference or standard model to adopt.

In literature one of the most implemented architecture is the *five-layers-model*, which appears to be accurate and, at the same time, generic enough to guarantee an efficient communication among several different IoT devices [4]. This IoT model consists of five layers:

1. **Objects Layer**: it is the base of the model. It consists of the physical sensors (e.g. accelerometer, thermometer, etc.) that measure parameters of interest and produce the "Big Data". The produced data are digitalized and sent to the upper layer;

2. **Object Abstraction Layer**: it transfers data produced by the previous layer to the Service Management layer through secure channels, exploiting technologies like 3G, WiFi, Bluetooth, ZigBee, *etc.*;

3. **Service Management Layer**: this layer, also called *Middleware Layer*, processes the received data and, by using devices' addresses, it delivers the required services over the network;

4. **Application Layer**: it provides the services to the user through an interface. Thanks to this layer, the user can interact with devices, *e.g.* smart-phones,

tablet, *etc.*.

5. **Business Layer**: it manages and monitors the whole IoT network; furthermore it organizes the generated data (*e.g.* with flowcharts, graphs, *etc.*), making them available for further analysis.

As shown in Fig. 3.1, IoT applications are various and are involved in several aspects of the every-day life. Some of the potential IoT application categories are:

Figure 3.1: IoT application fields [4].

Industry: In this field, IoT is adopted to improve efficiency, making the production processes more autonomous and computerized. Thanks to the interconnection among smart-devices, IoT increases the productivity by analyzing and controlling the flow of data and by solving production issues [4];

Agriculture: Involving smart farming networks that improve productivity, management and transparency for the consumers [28];

Transportation: IoT networks for traffic administration and for driving experience improvement [28];

Smart Home: Related to interconnection between home systems and appliances (*e.g.* heating systems, air contitioning, energy consumption monitoring, *etc.*). This enables an easier house management and it improves the every-day home experience [4];

Health-care: It enables a better monitoring of the patient, by improving the management of the health-care systems and by proposing advanced and novel solutions to reach medical wellness, diagnosis and treatments;

Education: It offers new educational tools, and it also simplifies of the teaching and learning processes by making them more interactive;

Life-style: It offers new entertainments (*e.g.* immersive virtual games), improving social interaction (*e.g.* social networks) and daily life.

All of these fields (and many others) are going to be deeply explored in the next future, paving the way towards the realization of a *Smart-City*: an interconnected environment with ubiquitous and smart services, quickly accessible by every citizen through a widespread smart-technology [4].

3.1 *Medical Internet of Things* (mIoT)

Fig. 3.2 shows the projected economic market of IoT by 2025. It is evident that health-care, along with manufacturing, is going to have the largest economic impact, by creating about $1.1/ $2.5 trillion annually by 2025 [4]. Indeed, terms like *Medical Internet of Things (mIoT)* and *Health Internet of Things (HIoT)* have gained popularity in health-care sector, during the last few years.
A mIoT system offers instrumentation, wireless networks and innovative ways for handling the health-care systems in the delivery of services by exploiting the interconnection between smart-device. These networks increases the quality and

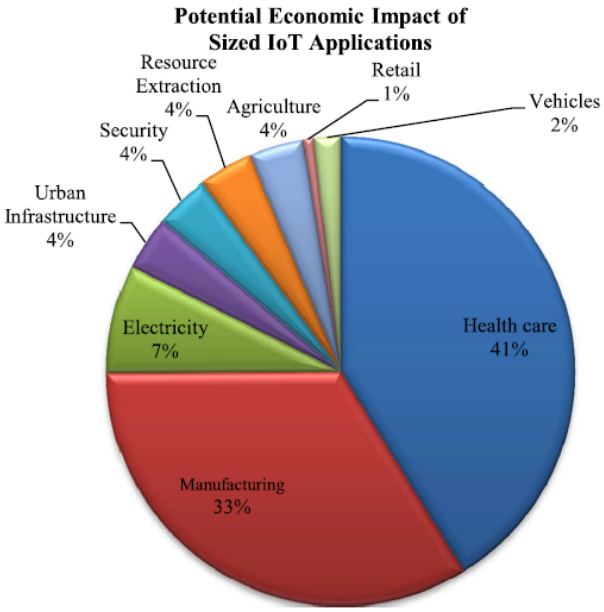

Figure 3.2: Projected marked of IoT by 2025 [4].

the availability of care services while reducing costs and inefficiencies [30].
The success of IoT is also fueled by the advances in sensor technology, which are becoming more and more smart, cheap and small in dimensions (either implantable or wearable), and to the large availability of wireless telecommunication channels, such as WiFi, Bluetooth, 3G, NFC, *etc.* [46].
mIoT potentially has a wide number of applications, which includes two general categories: (i) *IoT for smart hospital* and (ii) *IoT for personalized health-care*.

IoT for Smart Hospital

It includes all the IoT architectures that will bring, in a near future, to the realization of *Smart Hospitals* with better management of the services and more advanced ways for patient monitoring and therapies' delivery.
For instance, in the Smart-Hospital (Fig. 3.3) the patient will have medical records, prescriptions and therapies history stored in a cloud infrastructure; then the medical doctor will be able to access all of the patient's data by scanning an ID card. Furthermore, thanks to smart-devices such as smartphones and tablets, the ink-and-paper system, which is often not reliable, will be totally replaced by a

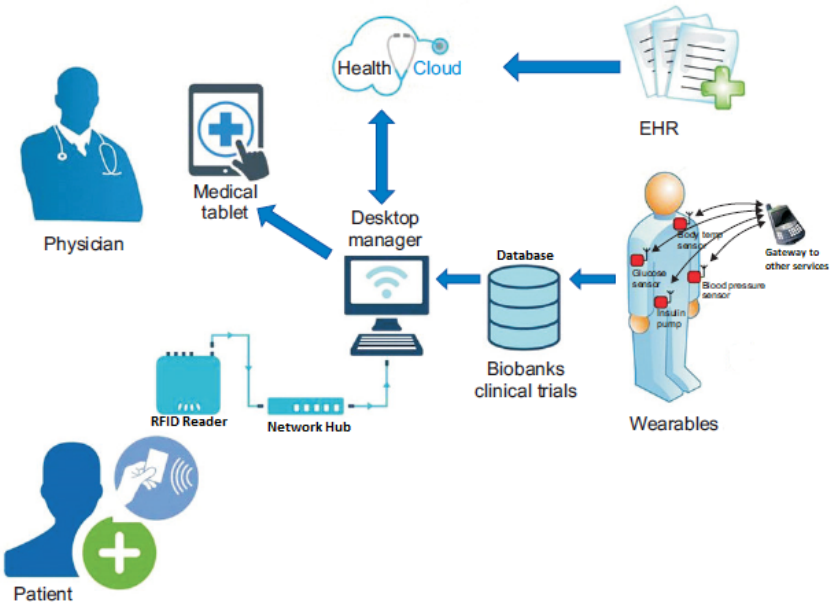

Figure 3.3: Example of an IoT architecture in a future Smart Hospital [5].

digitalized system [5], enabling a better access, recording and management of the medical data.

Another future scenario for this category is the *Ambulance Service* [6]. As shown in Fig. 3.4, firstly the patient will be able to share his/her current location by using a specific application; then the ambulance, using medical equipment (*ElectroEncephaloGram* (ECG), sphygmomanometer, thermometer, *etc.*) supporting WiFi communication, will continuously send measurements to the hospital. Therefore, the hospital will be able to start organizing the treatments before the patient's arrival according to the received data. Finally, traffic lights and traffic jam will be controlled to let the ambulance to quickly arrive to the hospital.

IoT for personalized health-care and telemedicine

IoT networks realized with interconnected wearable sensors enriched by mobile applications are enabling the continuous monitoring during everyday activities. This brings on one hand to prevent from health problems, and on the other hand to facilitate the diagnosis and the rapid responses in case of emergency [46].

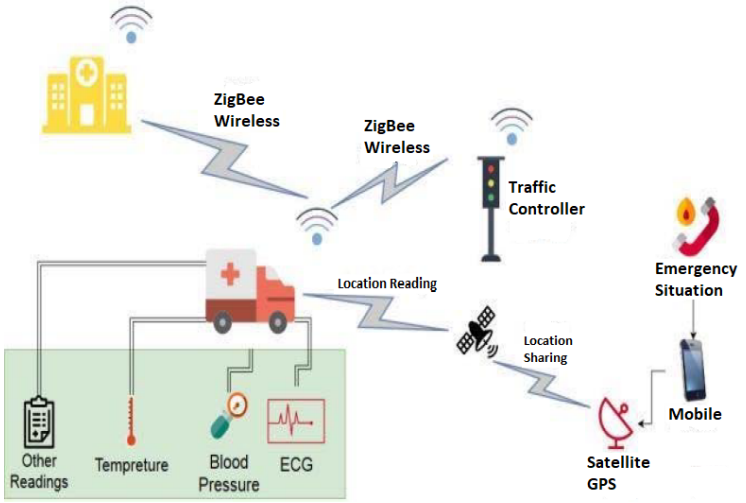

Figure 3.4: Example of an IoT architecture for emergencies management [6].

3.1.1 IoT and telemedicine

One of the most interesting application comprised in mIoT is the remote health-care and diagnostics, known as **Telemedicine** [7] or **Telehealth**.
Thanks to its advantages and to the improvements of the *Information and Communication Technology (ICT)* system, telemedicine is going to have a big influence in the health-care field, *e.g.* by enabling remote caring of the patients directly at their homes [47].
Indeed, according to United Nations, the number of people over 60 years old in the world reached 901 million in 2015 and projections show that it is going to increase to 1.4 billion in 2030 and to 2.1 billion in 2050 [48]. In addition, it has been observed an increase of chronic diseases, as *Chronic Heart Failure (CHF)* and Diabetes [48].
In this context, continuous remote monitoring will help to reduce hospitalization by allowing fast interventions in case of anomalies and, at the same time, it will provide the patients with a more independent daily life, which is highly desired especially by people affected by chronic diseases or by elders [48].
Moreover, telemedicine could be particularly helpful in rural areas and in underdeveloped countries, where there is not an adequate number of physicians and health

services, such as hospitals, medicines and diagnostic laboratories [49]. In this zones more expensive and inefficient health-care system are still provided. It is usual, in these places, that patients wait until suffering from serious symptoms before contacting doctor, and often at that point the patient needs specific treatments and specialists, who often are not available. Unfortunately, these limitations of the medical system are not solvable by any immediate improvement such as by building new infrastructures and facilities or by bringing a larger number of physicians [49].

In this context telemedicine could represent a great opportunity, by providing an easy way for exchange informations among specialized medical structures and by ensuring remote monitoring, remote diagnosis and telesurgery assistance [50]. For example, exploiting wearable sensors and telecommunication infrastructures, vital parameters could be monitored, collected and sent to the hospitals, becoming accessible to experts.

A typical model of this telemedicine enviroment is presented in [7] (Fig. 3.5). Bedside sensors measure and collect vital parameters of the patient. Data are then encoded and sent to an *Exchange Service* through WiFi. The *Exchange Service* can redirects data to a remote storage or it can handle requests from content service, including medical devices used by physicians (for example for remote data visualization).

However, there are still some issues that slow down the growth of telemedicine. First of all infrastructures problems, due to a not equally distributed bandwidth and telecommunication lines; secondly there are still acceptance issues, especially in rural areas, where the technology is not part of the daily life yet [50].

3.1.2 IoT and Cloud Computing

As mentioned before, IoT is strictly related to the concept of "Big Data": a huge amount of data is continuously produced requiring the need for powerful tools to handle them (data exchange, sharing, management, storage and safety)

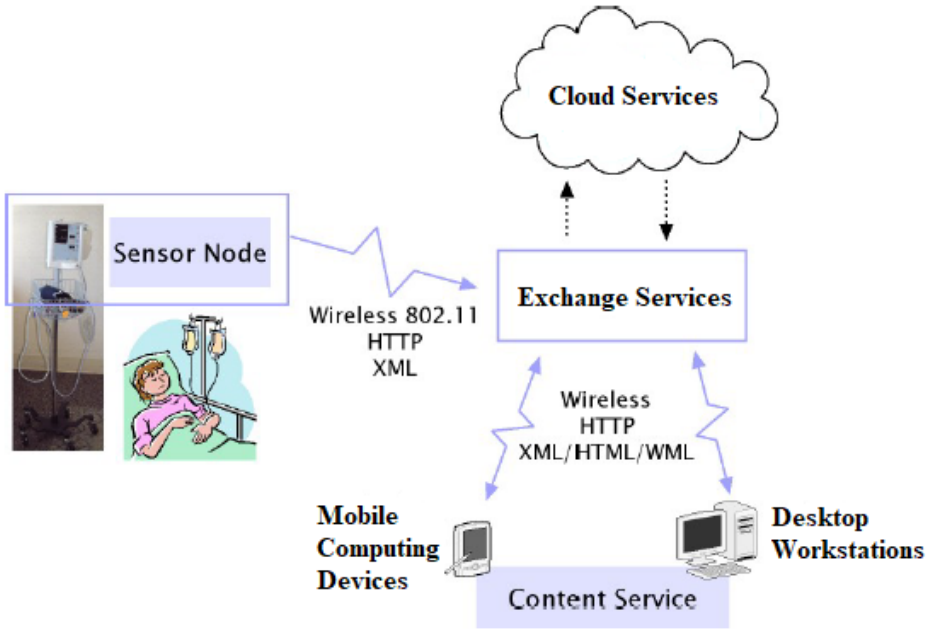

Figure 3.5: Proposed solution by [7] for an IoT architecture for telemedicine practices.

[51].

Recently, a new technology innovation subject has widely been investigated for these purposes: **Cloud Computing**.

According to the definition given by [52], Cloud Computing is an *"information technology service model where computing services (both hardware and software) are delivered on-demand to customers over a network in a self-service fashion, independent of device and location. The resources, [...], are shared, dynamically scalable, rapidly provisioned, virtualized and released with minimal service provider interaction [...]."*.

Thanks to the cloud, a massive number of data can be stored, shared and managed. Various and numerous are its advantages. It is *flexible*: multiple users can access data stored in a cloud trough the Internet from anywhere and at any time. It is *cheaper* than the current storage solutions: cloud's costs follow the formula "pay-per-use", in which the users pay only for what they use [52]. It can provide *immediate access* to resources. It is *secure*: data can be safely stored and protected,

thanks to efficient security measurements. Finally, it is *up-to-date*: cloud services are maintained and updated by the provider, hence users are always working with the latest platform [53].
As a consequence of these advantages offered by cloud-based solutions, recent mIoT systems are using these infrastructures for data storage and management. Cloud computing improves the efficiency in the use and coordination of medical resources and infrastructures; moreover, its scalability permits to expand or reduce the use of the cloud fitting the user's needs [53].
To sum up the main fields where medical applications can benefit from cloud computing are:

- IoT system for remote monitoring. Data are sent in real-time to a cloud infrastructure, allowing an immediate visualization;
- Patient's medical history can be stored on cloud. In this way every patient can access data and share them with the medical doctors, without the need of repeating medical exams;
- Telesurgery applications: surgeons are using cloud computing to control robotic surgery [50];
- Smart-Hospital: a cloud solution could be used for managing the administrative tasks, as patient admission, scheduling and monitoring [50].

3.1.3 Mobile health (mHealth)

An emerging part of the mIoT that could particularly take advantage from cloud computing is *Mobile Health (mHealth)* [31]. *The World Health Organization (WHO)* defines mHealth as a "*medical and public health practice supported by mobile devices, personal digital assistants (PDAs), and other wireless devices* [54]. In the last two decades there have been great proliferation of mobile applications for smart-devices, such as smart-phones, smart-watches and tablets, due to the improvements in technologies and to the reduction of the prices. The Fig. 3.6

shows this trend: smart-phone sales increased from 0.52 billion in 2011 to 1.5 billion in 2016. Smart-technology have penetrated in the society, impacting not only the communication, but also our habits [55].

Furthermore, the latest generation of smart-devices has impressive computational

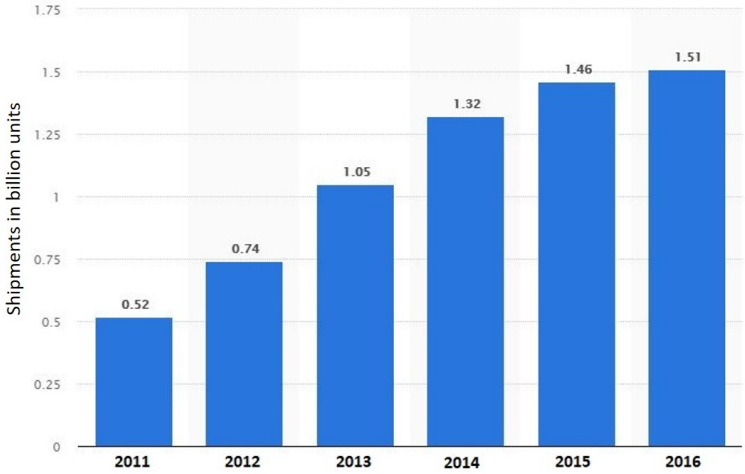

Figure 3.6: Smartphones shipments worlwide from 2011 to 2016 [8].

skills, storage capabilities and integrated sensors. All these features make these devices suitable for several application in health-care, making services portable and easily accessible [10]. A typical mHealth IoT-based architecture has been presented by [9](Fig. 3.7). The patient is wearing different types of sensors for monitoring purposes. The measurements are collected from an application running on the smart-device, and afterwards sent to a medical server (*e.g.* cloud support). Finally, physicians who are in charge of accessing the data, are enabled to remotely monitor the patient.

Therefore, the potentiality of mHealth is the capability to deliver health-care services trough common devices, without the need of buying new *ad-hoc* instrumentation, except for external sensors. Further, mobile devices make the system portable and user-friendly, thanks to applications that can fit the user's technology experience.

Finally, by combining Cloud Computing with mHealth, powerful networks, enriched by huge storage capability and computational skills, are possible to be

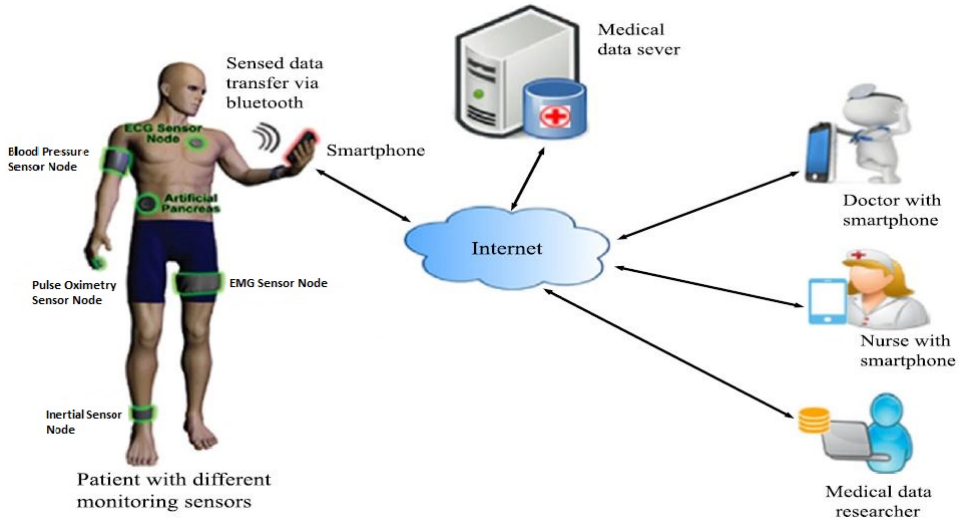

Figure 3.7: Medical health architecture proposed by [9].

realized.

3.1.4 Challenges for mIoT

The realization of an IoT network is characterized by several challenges that have to be faced. In particular, every IoT system has to meet some requirements:

- **Availability**: Data must be always available to the users, that can access simoultaneosly from different places. This requirement can be satisfied providing redundancy for critical services [4];
- **Reliability**: It is related to *availability* and it refers to the success rate in delivering services and providing data. Reliability can only be obtained with a strong underlying communication technology, otherwise lack of reliability could cause errors in transmission, long delays, loss of data [4, 7]. It could be very hard to ensure technical maintenance since the objects of the network are normally far one from the others [55];
- **Integrity**: Data have to be transmitted and received without any modifications [9];

- **Mobility**: An IoT network is expected to deliver services to several users geographically dislocated. Therefore, it is crucial to ensure the capability to continuously access the services by mobile devices [4];

- **Scalability**: An IoT network must be flexible, allowing to expand and contract the system based on the users' needs. There must be the possibility to add and remove services, functions and devices without damaging the system's performance. The difficulty relies on the fact that the network is constituted by several connected devices characterized by different communication protocols and hardware platforms [4];

- **Security and Privacy**: Data must be protected from unauthorized accesses. These requirements are probably the most tricky, since there is no standars in IoT security. In such heterogeneous environment, with different devices interconnected through the Internet, is not easy to guarantee security and privacy and at the same time maintain the flexibility and the usability [4].

All these challenges are even more evident in a health-care environment, where an unstable system could bring to wrong decisions, which could affect the wellness of the patient. Further, security and privacy must be absolutely addresses, since an mIoT system involves sensitive data.

3.1.4.1 Security challenges of mIoT

The discussion about security in mIoT systems is wide and controversial since they are based on interactions between numerous and heterogeneous devices, and security must be guaranteed at every communication level and for every device. Some of the security challenges of an mIoT system have been pointed out in [56]: *confidentiality*, that ensures the data can be read only by the receiver; *authentication*, thanks to which only allowed people can access data; *privacy*, which is about protecting personal information in the devices, communication,

storage and processing.
Another big challenge relies on the responsibility and legislation. mIoT systems involve several service providers and the *objects* of the network are situated in different Countries, with different data security jurisdiction.
Cloud computing in healthcare is an example of the complexity of this matter. Often Cloud services are offered by external providers, which guarantee for their performance. Therefore, in case of lack of security (*e.g.* in case of unavailability of patient's medical history, data modification or technical problems during telemedicine applications) the providers have to take their responsibility. In this context, it is also important the physical place in which data are stored and consequently the jurisdiction of the involved Country. There is not yet an international regulations to handle this issues, hence, in future, it will be important to cope with the expansion of IoT systems also from this point of view [47].

3.2 State of the Art

In this section some mIoT solutions that have been already presented in literature are going to be described.

3.2.1 Physiodroid

PhysioDroid, introduced by [10], is a mIoT system for remote and continuous monitoring of physiological and behavioural conditions.
As shown in Fig. 3.8, the system consists of:

1. **Wearable devices** for measuring physiological data and for transmitting them wirelessly via bluetooth. In particular, the monitoring is performed trough the *Equivital EQ01 system*, that is a multi-parametric and portable device (Fig. 3.9), mounted on a belt. The EQ01 can sense electrocardiogram signal (with two electrodes), the heart rate (trough the electrocardiogram), the respiration (trough an embedded strain gauge), the motion (with an

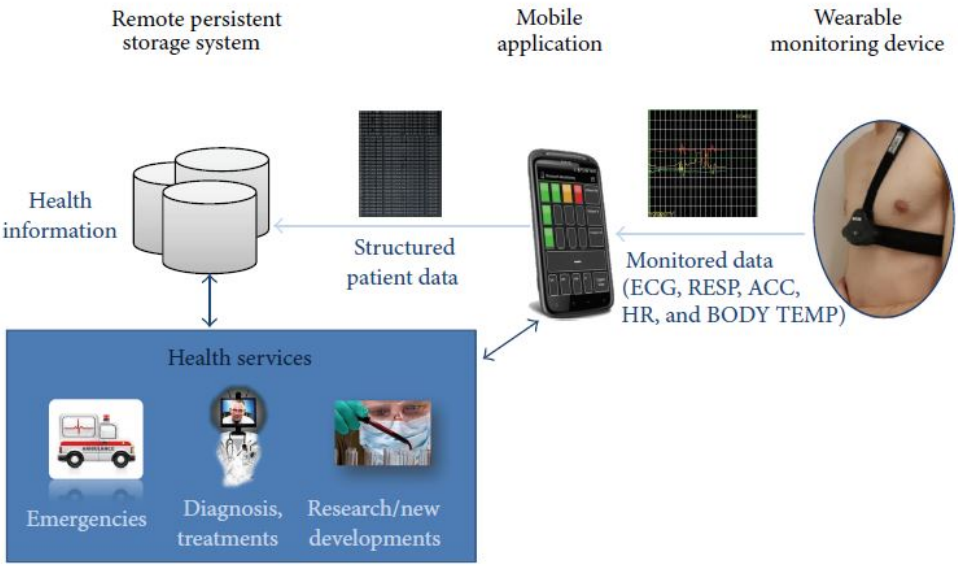

Figure 3.8: The PhysioDroid system [10].

accelerometer) and the body temperature (trough a thermometer).

This device also enables the data processing, as filtering and extraction of

Figure 3.9: Equivital device [10].

basic features; for example it can detect a set of postures and can extract the R-R interval from ECG signal;

2. **Physiodroid**, an Android application that, trough a mobile device (*e.g.* smart-phone), collects data received from the *Equivital* system, gives support for medical diagnosis, alerts in case of emergency, and offers the possibility to send data to a remote storage. In particular, the application, after a login page, handles the creation of a bluetooth communication channel with

the EQ01 system. After the binding, the application starts collecting data. *Physiodroid* offers two visualization modes: one for expert analyst and one for average user. The first one provides a real-time representation of data waveforms; the second one provides averaged values, *e.g.* hearth rate and temperature, and visual indicators about the status of the person. There is also the possibility to set an emergency call in case of critical values;

3. **Remote storage system** can store data from multiple users. Trough the application, the user can decide whether to send or not measured data to the server.

This personalized IoT-based architecture is particularly useful for home monitoring, *e.g.* for an elderly person. It can be also used for sport applications to monitor the body's response to physical efforts and trainings.

3.2.2 IoT m-GreenCARDIO Remote Cardiac Monitoring System

The mIoT system described by [11] is based on a mobile tele-electrocardiograph, the *m-GreenCARDIO*, together with an application, for mobile devices or desktop, and a cloud structure (Fig. 3.10).

The purpose of the system is to enable tele-diagnosis and home monitoring of cardiac patients, using the *electrocardiogram (ECG)* signal. The ECG is one of the most used instruments in medicine, due to the simplicity of the signal acquisition process and to the significant number of informations that can be extracted from it [11].

The *m-GreenCardio* system is the wearable module of the architecture, and its main components are:(i) a circuit for measuring ECG signal trough electrodes placed on the body of the person; (ii) a microcontroller for processing and managing the ECG data; (iii) a microSD card for saving the collected data (so in case of communication malfunctions the system can send the data after the communication

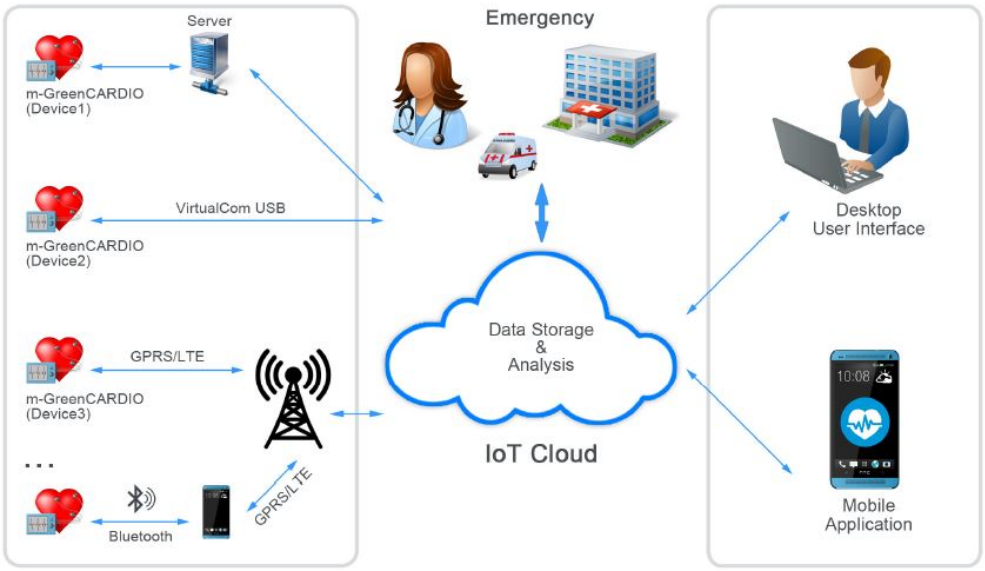

Figure 3.10: IoT architecture of the ECG monitoring system [11].

is re-established); (iv) an accelerometer, for controlling the right position of the device on the body; (v) bluetooth module, used to transmit collected data to the mobile application; (vi) a GPRS module, used to transmit data directly to the cloud, in case the patient does not have a smartphone or a tablet.
Thanks to the application, every patient can autonomously monitor his/her status. In addition, the application automatically sends received data to the cloud, from where doctors or nurses can monitor the patient and, in case of emergency, call an aid unit.

3.2.3 IoT Based Low-Cost Distant Patient ECG monitoring system

In [12] another mIoT system for distant ECG monitoring is presented. The system allows to measure ECG and then to send measured data to a cloud structure (Fig. 3.11). The peculiarity of this system is its inexpensiveness, compared to other ECG tele-monitoring system proposed in literature [12].

The system consists of three hardware modules, *Raspberry Pi (RPi)*, *Arduino* and *ECG Sensor module* for data measuring, processing and transmission, and

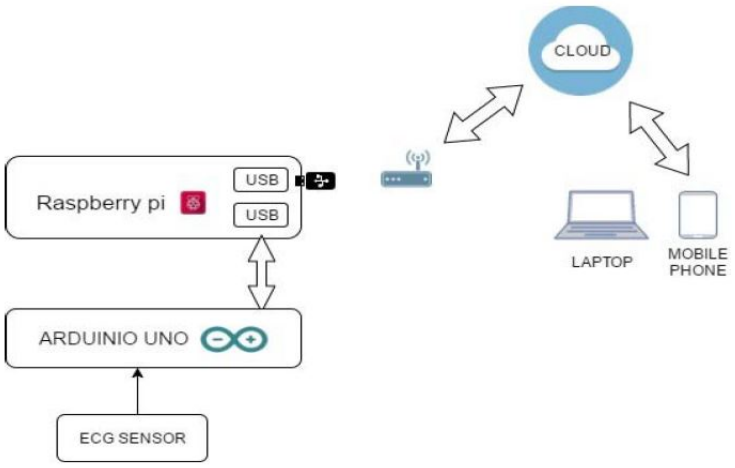

Figure 3.11: IoT architecture proposed by [12] for distant ECG monitoring.

a cloud platform. The RPi sends measured data to the Cloud via Wi-Fi. The cloud used is named *Bluemix*, owned by IBM, and it allows to plot in real-time the ECG signal and to memorize it for later analysis (Fig. 3.12).

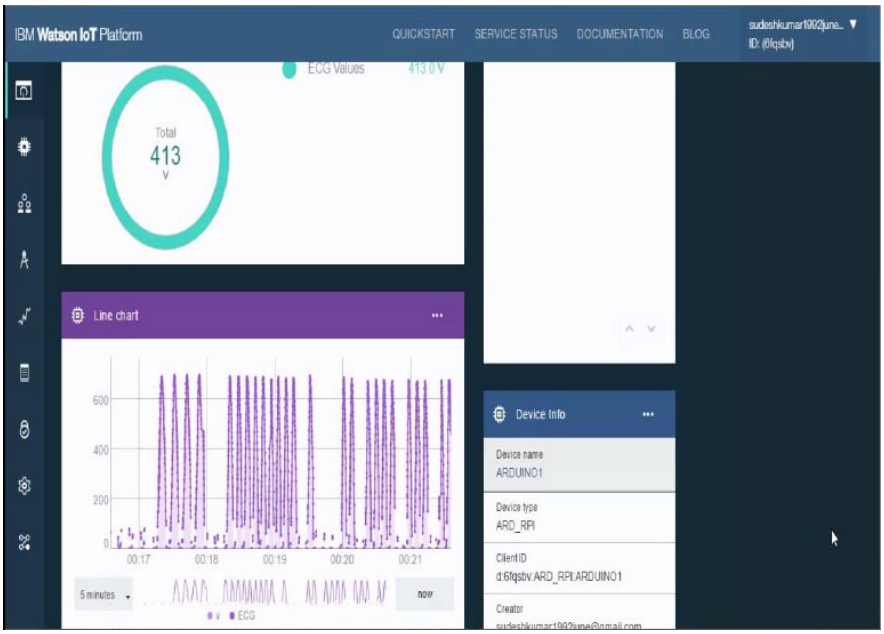

Figure 3.12: Ecg visualized on the cloud interface (adapted from [12])

3.2.4 mDurance: A Mobile Health System to Support Trunk Endurance Assessment

Banos et Al., present a mobile health system to support specialists during trunk endurance assessment procedures [13].
The systems consists of (i) wearable sensors for monitoring trunk posture and for detecting the *electromyographic (EMG)* signal produced by trunk muscles, and (ii) an Android application, called *mDurance*, that receives, processes and shows measured data. The communication between the mobile device, *e.g.* smart-phone, and the wearable sensors is done exploiting the bluetooth technology.
Fig. 3.13 shows some of the mDurance activities. After the login procedure, the specialist can select an existing patient or can register a new one by inserting the information of the patient. Then the specialist can either connect the selected profile with the wearable sensor and start the endurance test or visualize the historical data of the patient. The expert can also choose the type of the test to perform, for example visualizing the EMG signal or other parameters.

3.2.5 Ubiquitous Access to Cloud Emergency Medical Service

Emergency Medical Services (EMS) includes the health-care services provided by pre-hospital and in-hospital emergencies. In [57] the authors present a system for accessing *personal health records (PHR)* of the patients, using a cloud platform, to ensure a quick intervention in case of emergency. The system consists of three components:

- a **PHR platform**, composed by a user interface and a medical recorder repository stored in the cloud, through which the patients can access their own informations or the authorized physician can access some parts of the patient's medical data;
- **EMS application**, that consists of a data repository which contains emer-

Figure 3.13: Some of the activities of the *mDurance* app (Adapted from [13])

gency medical data. It can be used by authorized personnel, such as ambulatory or the emergency personnel;

- **portal**, that provides a web-based front-end for the EMS application. The portal can be accessed also from mobile phone, which can be useful in case of remote access to patient's information.

To sum up, although a relevant number of IoT systems for health-care have been already presented, there is still a very small number for physiological continuous monitoring provided [10]. Currently, none of the already implemented mIoT solutions systems has been realized with the aim to be adopted in continuous monitoring of anesthesia. In this thesis, I have worked to overcome this lack by proposing the first mIoT system for anesthesia practices.

Chapter 4

Materials and Methods

4.1 Android OS

The Android *Operating System (OS)* is one of the most popular and widely diffused OS for mobile devices [58]. Every day more than a million Android devices are activated all over the World [19].
In accordance with StatCounter, in 2017 Android has been the most used OS in the World (Fig. 4.1), surprisingly surpassing even the Windows, sign of the wide expansion of mobile devices in everyday life.

Android is an open-source platform and it was originally developed by Android

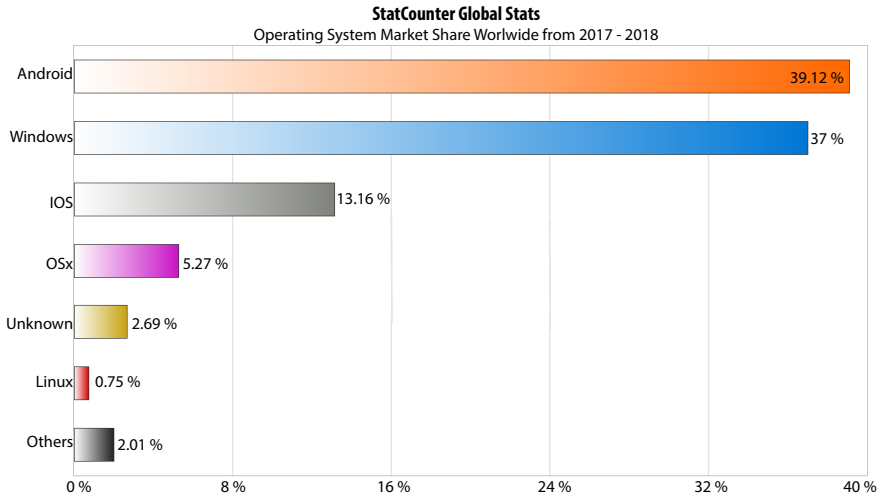

Figure 4.1: Operating System Market Share Worldwide in 2017 [14].

Inc., founded in California in October 2003 by Andi Rubin, Rich Miner, Nick Sears, and Chris White. The first aim of the Company was to develop an OS for digital cameras, but then they decided to expand their market, starting to think about Android as an adversary of Symbian and Microsoft Windows Mobile, that were the two most used mobile OS at that time [15].

In July 2005, Android Inc. and its employers were bought by Google, and in 2007 it was released as the *Android Open Source Project (AOSP)*. In that year was also founded the *Open Handset Alliance (OHA)*, a consortium of different techonology Companies, such as Google, NVIDIA, Intel, HTC, Qualcomm. OHA aimed to develop new technologies to reduce the time and the costs in development and distribution of mobile devices applications and services [58]. In 2007 the first Android *Software Development Kit (SDK)* was offered to the developers and phone manufacturers, and in 2008 the mobile phone "HTC Dream G1" (Fig. 4.2) was the first device with Android 1.0, in which there where services as HTML web browser, calendar, maps, Media Player, and Gmail. From 2008 up to now, there have been

Figure 4.2: HTC G1 Dream: the first mobile phone with Android, 2008 [15].

numerous Android updates (almost once a year), introducing new features and improving performances. During these years, Android has been established the tradition of naming each major release with the name of dessert in alphabetic order (Fig. 4.3): *Cupcake*, *Donut*, *Eclair*, *Froyo*, *Gingerbread*, *Honeycomb*, *Ice Cream Sandwich*, *Jelly Bean*, *KitKat*, *Lollipop*, *Marshmallow*, *Nougat*. In 2017 the latest version, Android Oreo (Android 8.0), was launched and in 2018 it is going

to be delivered to many devices.

Google has also created *Google Play*, a digital platform for the distribution of

Figure 4.3: Android release history [16].

Android applications.

4.1.1 Android Wear

Together with improvements in smart-devices, as smart-phones and tablet, from 2000 many researches have been focused on the design of *wearable* smart-devices, aiming to transform common objects, like watches, in systems capable to perform different interactive actions.

In 2014 OS were introduced in wearable devices. In September 2014 Apple announced its first smart-watch, the *Apple Watch*, with a dedicated OS, the *watchOS*. The Apple Watch, after the pairing with an iPhone smart-phone, was capable to execute different services, such as answering calls, sending messages, and monitoring fitness activities.

However, in March 2014, before the launch of the Apple Watch, Google had already announced a new version of Android OS designed for smart-watches and other wearable devices: **Android Wear** (Fig. 4.4).

After the pairing with a mobile phone running Android 4.3 or higher, this platform allows to integrate in the wearable device different Android features,

Figure 4.4: New generation smartwatch with Android Wear [17].

such as receiving real-time notifications, showing maps, news, emails, listening to music. It integrates also Google Assistant, which permits to literally speak with the wearable device asking for services. Using watches equipped with adequate sensors, Android Wear allows to monitor vital signals as hearth-rate, fitness exercises, burned calories *etc.*.

In 2016 Google announced Android Wear 2.0, which introduces new several features:

- *Stand-alone apps:* Android Wear 1.0 watch always needs to be paired with a phone to install applications; with Android Wear 2.0 the watch has direct access to the Play Store; hence the user can download all the available apps directly from it;
- *Network connection:* Android Wear 1.0 watch always needs the phone for making a network request. Through the companion app installed on the phone, the watch makes the request and then the received data are pushed to the watch trough Blutetooth or Wi-Fi. With Android Wear 2.0 the user can perform any network request directly from the watch;
- *Darker theme* to increase performances and to save power-consuming;
- *Wearable Recycler View*, which permits a better support for rounded displays;
- *Notification system:* a better and faster interaction between user and smart-

watch.

4.1.2 Android OS Architecture

The Android OS system architecture is organized as a five-layers stack (Fig. 4.5): *Linux Kernel*, *Libraries*, *Application Framework*, *Android Runtime*, *Applications*.

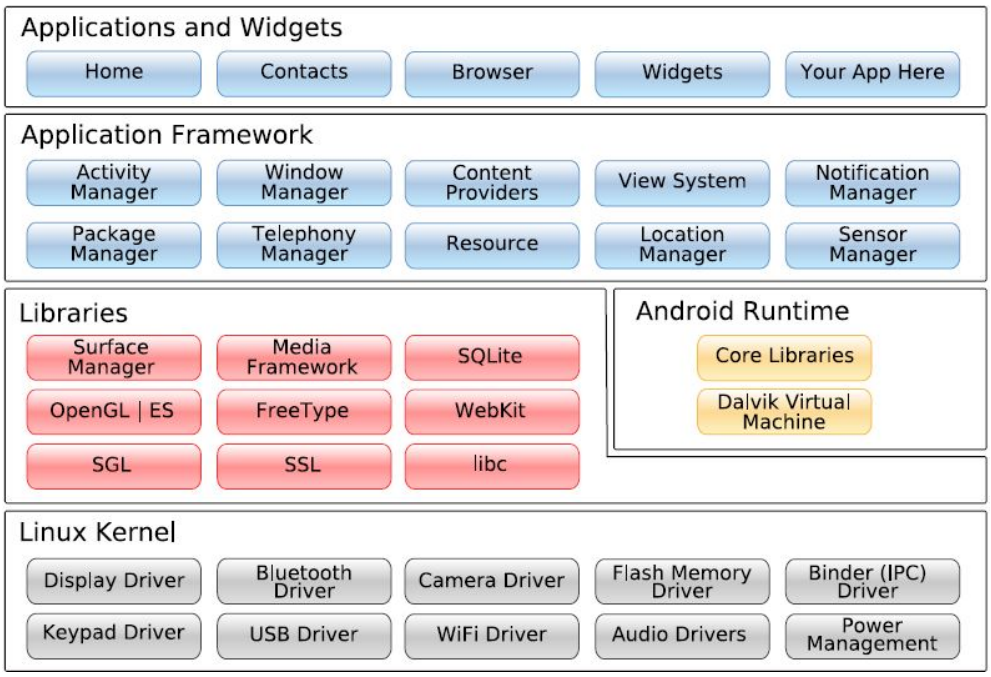

Figure 4.5: Architecture of Android OS [18].

- **Linux Kernel**: It is the base of Android OS. It is a 2.6 series Linux kernel driver, optimized for memory management, process management, networking and other services. Linux provides the hardware abstraction layer for Android, allowing to use it on a wide range of platforms. The Android user will never work directly with Linux [18], but has to keep in mind that every Android application runs on the Linux kernel;

- **Libraries:** This layer consists of libraries written in C and C++. Originally Android was designed to work on low powered *(Central Processing Unit (CPU)* and *Graphics Processing Unit (GPU)* devices, with limited memory, so the libraries on this layer ensure low-memory consumption. Further at

this level there is a Surface Manager, for screen access management, a Media Framework, for audio and vid in higher level applications;

- **Android Runtime**: It is used by the OS for converting the code, written from the developer in a high language (like Java), to machine code understandable for the processor.
  It includes Java Core Libraries and a Virtual Machine. Java Core libraries are similar to Java Standard Edition libraries (Java SE) and Java Mobile Edition (Java ME) libraries (used in Java programs), but they are optimized for Android.
  In Android, Java classes are converted in DEX bytecode, which is translated to machine code by virtual machines like ART or Dalvik runtime. Dalvik was used until Android 4.4 (Kitkat), but from Android 5.0 (Lollipop) it was replaced by ART, which uses less CPU and improves power performances and application's speed.

- **Application Framework**: It is placed above the libraries and runtime layers and it provides a number of high-level blocks for the creation of applications. The most important parts of this layer are:
  - *Activity Manager*, which controls the life cycle of applications;
  - *Content Providers*, which handles data that are going to be shared between the applications;
  - *Resource Manager*, that manages the resources, which are non-code pieces (as bitmap, layout and text string), that need to be used in the application;
  - *Location Manager*, that handles the system location services;
  - *Notification Manager*, which manages events as arriving messages, appointments and other alerts and notification for the user.

- **Applications**: It is the highest layer of the architecture, where are placed all the Android applications. An application is a program through which the

user can interact with the smart-device. It includes web browser, calendar, calculator an others.

4.1.2.1 Application components

Every Android application can exploit objects defined in the Android SDK. The most important are:

Activities Generally an application consists of several activities. An activity is a user interface where several different actions are available to let the user interact with the app. The Activity class creates a window as *User Interface (UI)*;

Intents Intents are used to start one component from another component in an Android application. Thus, if there is the need to start a component, such as a service, an activity or a Broadcast Receiver, an intent object must be created and passed to the Android Framework. There are two types of intents: *Explicit Intents*, when the component that has to be started is completed known; *Implicit Intents*, when the component is not specified, but is given the action that the developer wants, *e.g.* user's location, and by this intent the other application will interpret the required action to perform.

Services A service is an application component that runs in background and does not interact with the user. It can be used for lightening the UI and for background tasks, such as network communications;

Content Providers It can be used to manage access to data stored, and can handle the sharing with other applications.

4.1.2.2 Activity Lifecycle

When the user starts an application, *e.g.* Google Maps, Android starts it and bring it to the foreground [18]. From this application the user can also run other apps, *e.g.* web browser, or other parts of the same application and then he/she

can return to the previous screen. The Activity Manager handles and records all of these actions and parts on the *application stack*, putting in fore-ground the running activity.

Every application consists of different parts, some of them are associated to a screen interface and allow interaction with the user, while other activities develops background tasks.

Each activity has its own life-cycle, as shown in Fig. 4.6, and it can be in five different status: *Started*, *Running*, *Paused*, *Stopped*, and *Destroyed*. The activity

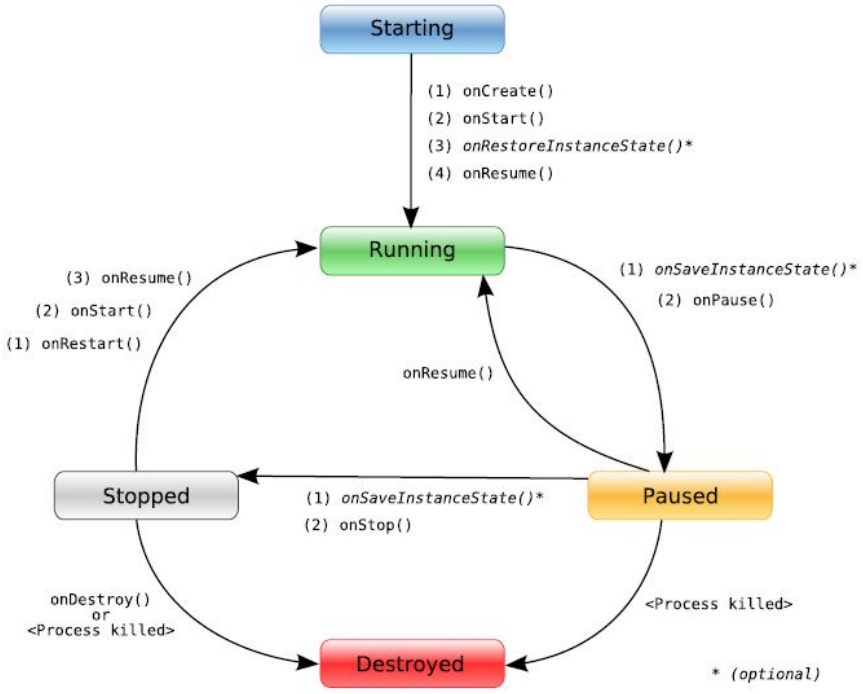

Figure 4.6: Activity Lifecycle [18].

class provides some callback methods trough which the developer can set how the activity has to behave, for example when the user leaves or returns to the activity. These callback methods are:

- *onCreate():* it is called when the activity is firstly created. In this method there are all the actions that the activity has to do in its lifecycle, for example the creation of the layout. This method takes one parameter, that can be *null* or some informations previously saved in the *onSaveInstanceState()*

method [18].

- *onStart():* when the activity enters in the *start state* this method is called and the activity becomes visible to the user;
- *onResume():* it is called when the activity interacts with the user;
- *onPause():* it is called when the user is going to leave the activity for a certain period of time. The activity enters in the *pause state* and goes in background, waiting for being resumed. In this state the user can not interact with the activity;
- *onStop():* the activity enters in the *stop state* and it is not visible to the user, but it still retains its state. It is called for instance when another activity covers the entire screen. From this state the activity can be resumed, and in this case the method *onRestart()* is called and the previous information and state are resumed, or the activity can be destroyed, calling the method *onDestroy()*;
- *onDestroy():* it is called before the activity is definitely destroyed;
- *onSaveInstanceState(Bundle):* it is called in order to save the state of the activity;
- *onRestoreInstanceState(Bundle):* it is called when the activity is going to be reinitialized in a previously saved state trough the *onSaveInstanceState()* method.

4.1.3 Android Studio

Android Studio is the official *Integrated Development Evironment (IDE)* for app development in Android or Android Wear [19]. It is a smart editor that offers code suggestions, refactoring and analysis tools, making the developing process faster and easier.

Figure 4.7: Android Studio Logo [19].

Each project in Android Studio is organized in modules to provide quick access to files: (i) Android app modules, (ii) library modules and (iii) Google App Engine module (Fig. 4.8).

The Android app module consists of three folders:

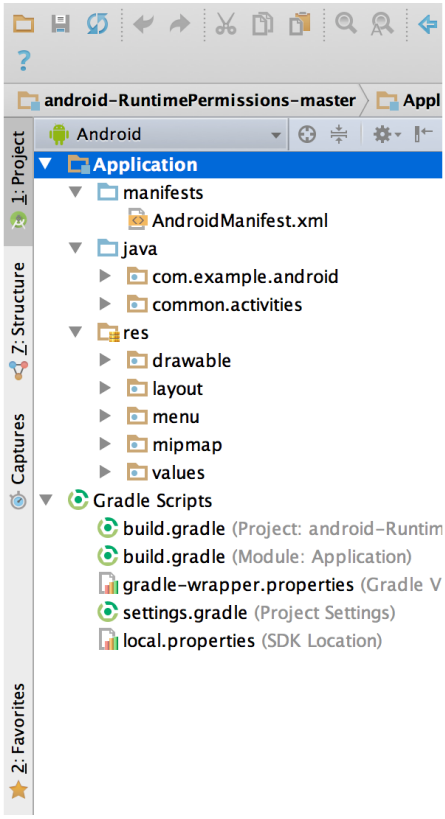

Figure 4.8: Project files in Android Studio [19].

- **manifest**: It contains the file *AndroidManifest.xml*. All the components of the Android application are declared in this file. The Android framework is able to consult the manifest file to know what are the various components of the application and in which way the developer invokes and uses them. Furthermore, it contains more details, like descriptions of the APi libraries,

description of the intent filters and the permissions list, if the application requires them from the user.

- **java**: It contains the Java source code files of the application.

- **res**: It includes folders for drawable, XML layout, values.

Another important feature of Android Studio is the possibility to test the application without connecting the phone, thanks to an Android Emulator that Android Studio make available (Fig. 4.9).

The main actor of the IoT system presented in this thesis is an Android ap-

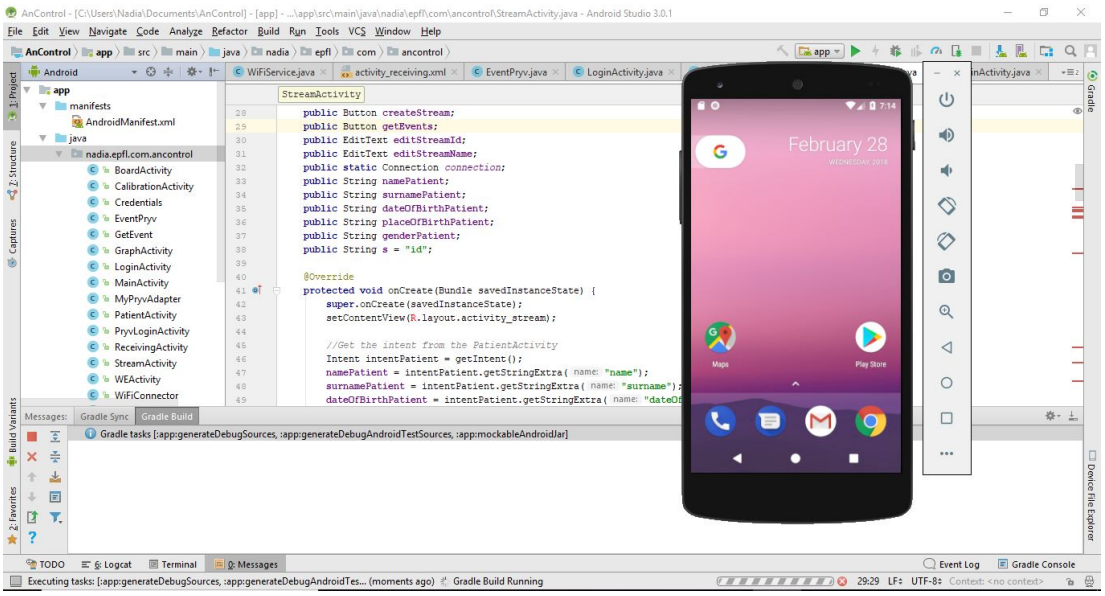

Figure 4.9: Android Studio main window and an Android phone emulated with Android Emulator.

plication, developed in Java language using Android Studio. Several Android components have been implemented, such as activities and services, to manage the interconnection with the all the elements of the IoT network.

The Android app is going to be described in detail in the next chapter.

4.2 Pryv Health Data Middleware

Pryv S.A. is a swiss company based at the EPFL Innovation Park, founded by Pierre-Mikael Legris. It offers a cloud platform that allows secure storage and secure transmission of personal health information.
Every user of Pryv has a personal account and can access the platform using the user-name and password. Different kind of data can be stored, manipulated and visualized on cloud.

Data in Pryv are organized in a modular structure, modeling individual data

Figure 4.10: Pryv Logo [20].

pieces as *events* and contextualizing them trough the *contexts* [20]:

- **Events**

  Events are the Pryv primary data units. An event is a "*timestamped piece of typed data, possibly with one or more attached files, belonging to a given context*" [20]. Depending on its type, an event can be a picture, a text, measured value, or other. Pryv offers a big number of types, which can be divided in two categories: *numerical types*, such as density, electric-current, energy, force, *etc.*, and *complex types*, which are more specific, such as audio, date, contact, position, video, *etc.*.

- **Contexts**

  The contexts are "*the circumstances in which events occur*" [20]. In particular, it is obtain combining two elements:

  - **Streams**: they are the fundamental part of the context. Every event belongs to a stream. Depending on the application, streams can follow

a hierarchical structure for a better data organization; indeed a stream can have some sub-streams [20];

– **Tags**: they provide additional informations to the context. Every event can be labelled with one or more tags, that can be plain text tag or typed tags.

A better explanation of the Pryv data structure is provided by Fig. 4.11, that is an example of data hierarchy in a Pryv account: inside the account there is a number of streams, in this case three; every stream has some events inside. In this case the stream1 has two substreams, with some events. For instance the

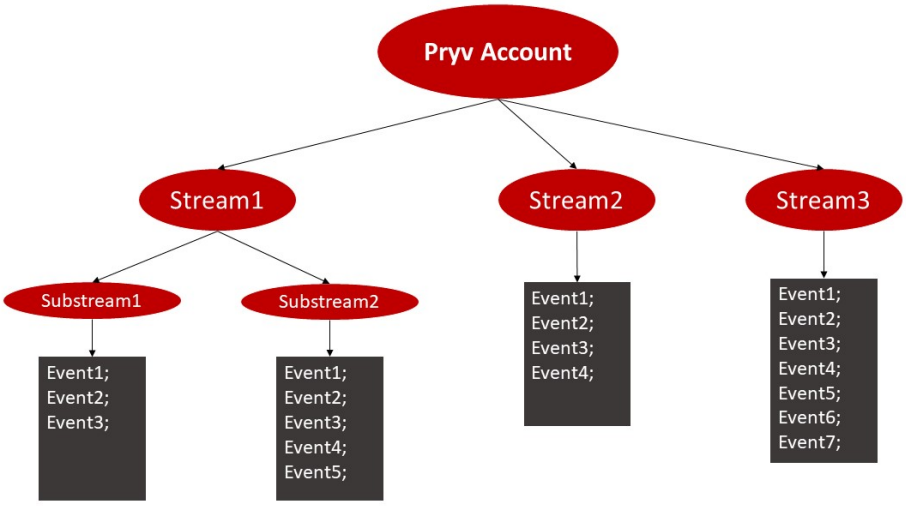

Figure 4.11: Example of a Pryv data hierarchy.

streams1 could be named as "physiological signals" and the substream1 cloud be named as "blood pressure"; each event is a measured value.
Stored data into the account are private by default, but some of them can be shared if the account's owner wants to allow external accesses. There are three types of accesses:

- **Shared accesses**, used for person-to-person sharing. They allow an authorized person to access a specific set of data. The owner can also specify

permission levels (*e.g.* read-only, read-write). The access is obtained by presenting the access' key, that is a *token* which can be transmitted via different communication channels [20];

- **App accesses**, used for the apps which do not need full, unrestricted access to the user's data. They grant access to a specific set of data and/or with limited permission levels (e.g. read-only), according to the app's needs; App accesses require the app to be registered on the Pryv account [20];

- **Personal accesses**, used by apps that need to access all of the user's data and/or manage account settings. With this type of access full permissions are given, including management of other accesses. Personal accesses require the app to be registered as a Pryv app [20].

Three different libraries are available for interaction with Pryv: Javascript (browser & Node.js), Apple Cocoa (iOS & OSX), Java (Android & desktop).
In this thesis the Pryv cloud platform has been used to securely store patient's measured data. Using the Pryv Java library for Android, the Android application establishes a secure connection with Pryv and it shares data to the patient's Pryv account. The streams and events structure has been used to better organize patient's information in the account.

Chapter 5

Architecture of the IoT network for continuous monitoring

The purpose of this thesis was to develop a mHealth system as an IoT cloud-based solution for anesthesia on-line monitoring during surgeries.
As shown in Fig. 5.1, the system mainly consists of three parts: (i) *Therapeutic Drug Monitoring (TDM)* sensing platforms, directly connected with the patient; (ii) an Android application; (iii) a cloud-based system with a Web Application.

The TDM platforms are able to continuosly measure the anesthetics' concen-

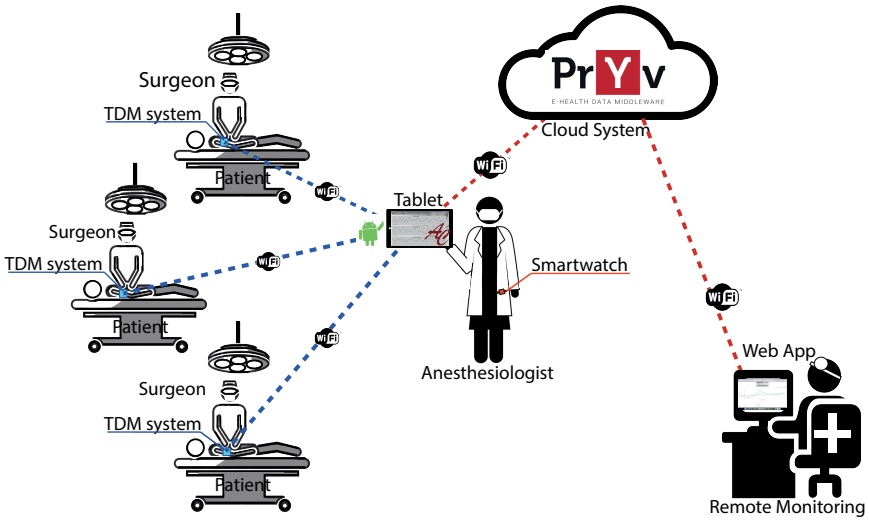

Figure 5.1: Sketch of the IoT cloud-based monitoring network [21].

tration in patient's blood. Exploiting Wi-Fi, the TDM system transmits data to the dedicated Android application, running on the anesthesiologist's smart-device, such as tablet or smart-phone. Thanks to the app, the anesthesiologist can simoultaneously visualize and monitor the drugs concentration from multiple patients. Furthermore, it is possible to share the received data on the cloud, enabling teleconsulting and remote visualization through a web application. Finally, the system includes a smart-watch to notify the doctor in case of an emergency [21]. The main objective of this architecture is to provide a safer patient monitoring by keeping the medical specialist always connected with all the patients under his/her responsibility. In other words, the doctor is allowed to freely move among the surgery rooms without loosing the control over all the anesthetized patients [21].

Investigation of Different IoT Architectures

In the first phase of this study, different IoT architectures have been evaluated for the network. In particular, three have been taken into consideration:

1. In the first architecture, sketched in Fig. 5.2.a, the TDM platform sends data both to the cloud and to the Android application, which only visualizes data. This configuration was rejected because the TDM system must handle a lot of transmissions, making the work for the TDM heavier. In addition, this configuration is not very convenient for the anesthesiologist that firstly has to handle the cloud connection in the TDM platform for each patient and after he/she has to use the Android application to monitor the patients;

2. The second configuration is represented in Fig. 5.2.b. As in the first option, the TDM platform directly sends data to the cloud. During the surgery the anestesiologist can remotely monitor the patient through a web application that is connected with the cloud, and that continuously displays the data. This configuration was not chosen because the interconnection TDM-cloud

requires dedicated libraries and, additionally, in case of connection problems with the cloud or in case of Internet unavailability the anesthesiologist is unable to monitor the patient and to communicate with the TDM system. Further, to use this IoT system the patient must give his/her agreement to the cloud sharing, otherwise it can not be used;

3. The third configuration (Fig. 5.2.c and Fig. 5.1) was chosen, because it offers the possibility to handle all the patients' and cloud connections trough the Android application. This enables on one hand to manage the system trough the Android app without making the TDM platform heavier, and on the other hand to have a more stable system, thanks to the possibility to use other communication technologies (*e.g.* bluetooth) in case of Internet unavailability to not lose the patients' data.

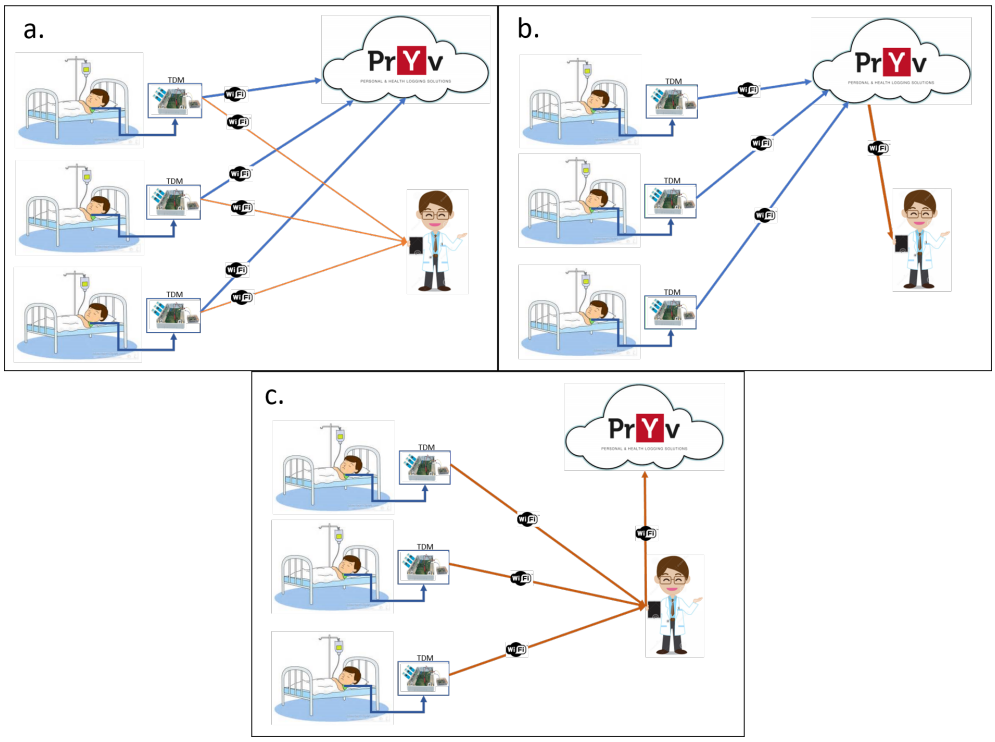

Figure 5.2: Sketch of the different IoT architectures taking into consideration for the monitoring system in anesthesia practices.

Hereafter, the main parts of the network are described in detail.

5.1 TDM System

5.1.1 Electrochemical sensors

Electrochemical sensors are considered excellent candidates for the realization of monitoring systems, thanks to their small dimensions and to the accurate and rapid detection process [23]. They operate by reacting with the molecule of interest and producing a current proportional to its concentration.
The most common configuration for performing an electrochemical measurement uses three electrodes (Fig. 5.3):

1. **Working Electrode (WE)**, which contains the chemical recognition system and a transducer for converting the chemical response into a current [59];

2. **Reference Electrode (RE)**, which has a stable and well-known potential to be used as reference for reading the other cell potentials;

3. **Counter Electrode (CE)**. which is adopted to close the current circuit. RE and WE are not enough for a correct measurement of the current, because a part of the applied voltage drops due to the resistance of the analyzed solution. Hence, the CE electrode is use to gathers the current from the RedOx reaction [60].

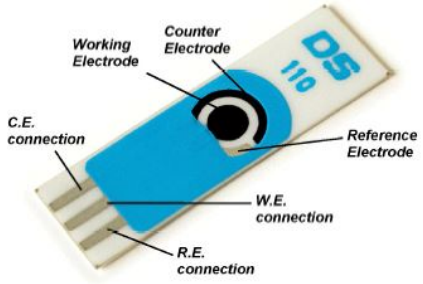

Figure 5.3: Electrochemical cell provided by a screen printed electrode [22].

Another fundamental component to perform electrochemical measurements is the *potentiostat*, that applies stable potentials between WE and RE and gathers the

current produces by the reaction of the WE with the analyzed solution through the CE [23].

By changing the applied potential waveform it is possible to perform different electrochemical techniques. Among these, three are preferred for drug sensing:

1. **Chronoamperometry (CA)**: In this technique a step-function potential is applied between WE and RE and the resulting current is registered and plotted as a function of time, as shown in Fig. 5.4.

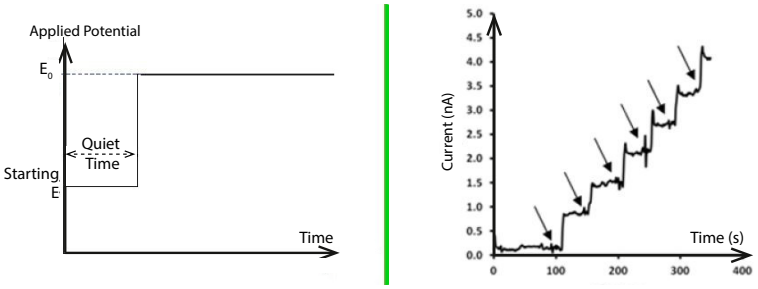

Figure 5.4: Chronoamperometry [23].

2. **Cyclic Voltammetry (CV)**: This technique applies a linear weep potential ramp potential between WE and RE (triangular shape), as shown in Fig. 5.5. The potential varies at a fixed rate from a V_1 value to a V_2 value. When the voltage reaches V_2 the scan is reversed and the voltage is swept back to V_1 [24]. This voltage variation produces a current variation which is plotted towards the applied potential obtaining a voltammogram.

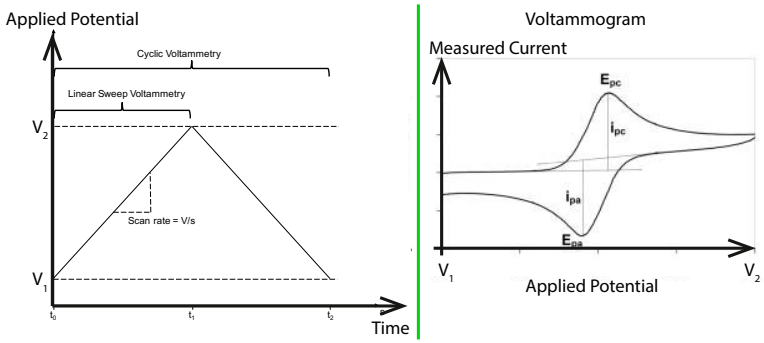

Figure 5.5: Cyclic Voltammetry [24].

3. **Differential Pulse Voltammetry (DPV)**: this technique is similar to the CV, but the applied potential waveform is realized by superimposing potential impulses of fixed amplitude on the triangular shape typical of the CV, as shown in Fig. 5.6.

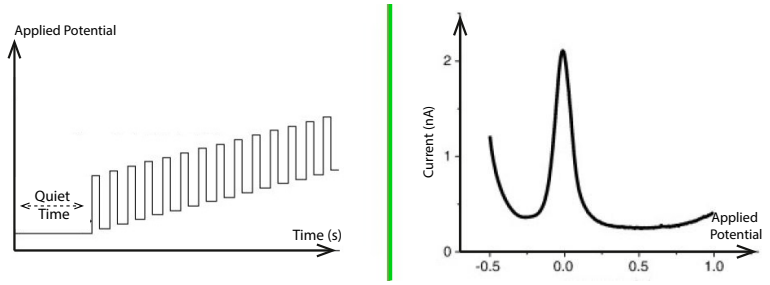

Figure 5.6: Differential Pulse Voltammetry (Adapted from [23]).

5.1.2 Portable and multichannel TDM system

The TDM platform, presented in [23, 61], consists of two main blocks (Fig. 5.7):

1. the **electrochemical sensing platform**, which includes a potentiostatic circuit and three electrochemical cells [21]. Each of them has three electrodes (WE, RE, CE) that are connected to a custom-built potentiostatic electronic circuit. The potentiostat ensures a constant potential between WE and RE and collects the current through the CE [3]. This system is capable to perform different electrochemical techniques, such as CA, CV and DPV, and it is designed to detect simultaneously up to three different compounds at the same time plus pH and temperature parameters, thanks to independent channels on the board.

2. the **RPi-based electronic system**, that is the central part of the TDM system. It enables the potentiostatic measurements and sets hardware parameters of the electrochemical sensing platform [23]. Furthermore, by exploiting Wi-Fi, the RPi continuously sends measured data to the Android Application.

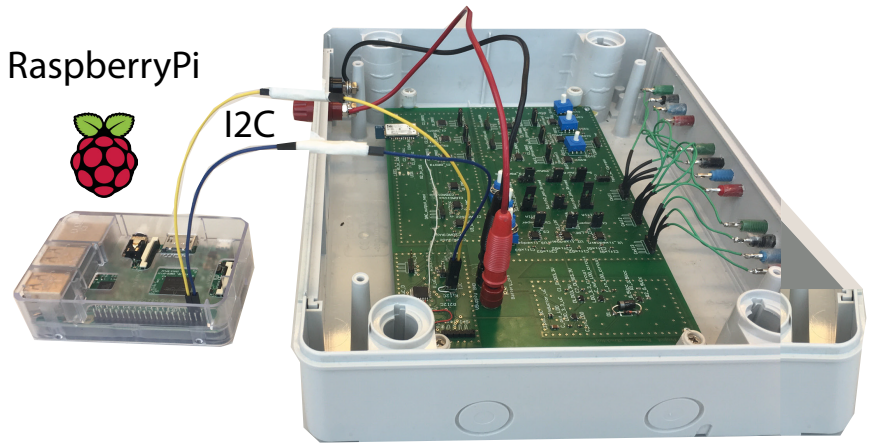

Figure 5.7: Main parts of the TDM system: Raspberry Pi (RPi) and electrochemical platform; i2c communication is exploited as serial bidirectional transmission between the electronic platform and the RPi (Adapted from [23]).

This TDM system enables the monitoring of the anesthetized patient by simultaneously detecting the concentration of up to three different anesthetic compounds. This is done by exploiting the relation between the measured current and the analyte concentration.

5.2 Android application: AnControl

One of the main objectives of this thesis was the development of the Android application, named as *AnControl*, running on the anesthesiologist's tablet or smartphone. The application establishes a bidirectional communication, via Wi-Fi, with the TDM platform. It enables the continuous plotting of the measured data allowing the anesthesiologist to monitor the anesthetics' concentration in patients' blood over time. In addition, trough AnControl, the medical doctor can manage the cloud sharing, enabling data transmission to the Pryv platform.

AnControl consists of seven *Activities* and one *Service* in background; hereafter they are going to be analysed in detail.

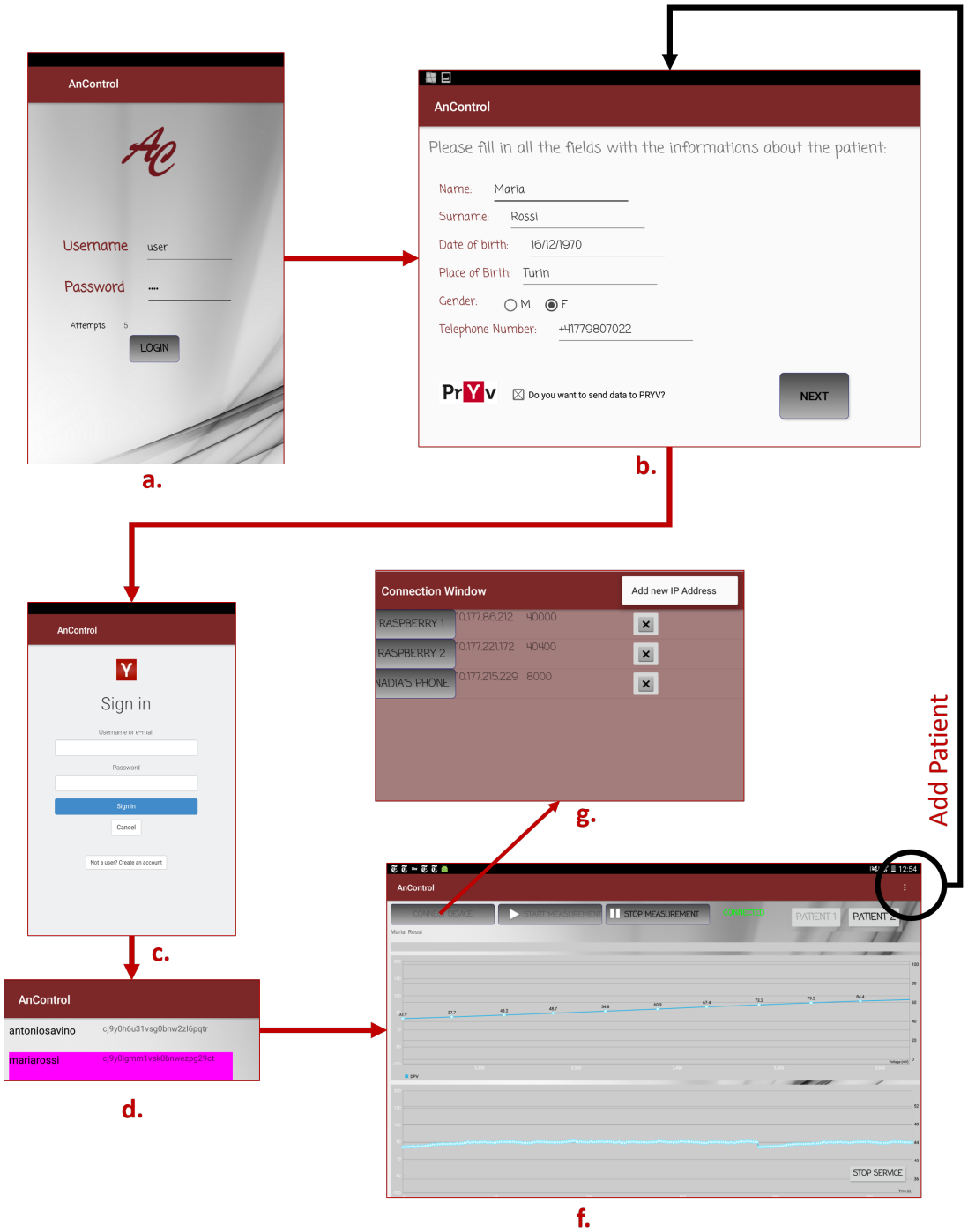

Figure 5.8: Main activities of AnControl: a. *Login Activity*; b. *Patient Activity*; c. *Pryv Activity*; d. *Token Activity*; e. *Monitoring Activity*; f. *Device Activity*;

Login Activity

The *Login Activity*, in Fig. 5.8.a, is the first activity of the application. It guarantees the access only to authorized doctors by requesting an user-name and a password.

Patient Activity

In the *Patient Activity* the doctor has to fill in the personal data of the patient (name, surname, date of birth, *etc.*). To this aim a Java class, named as `Patients.java`, was realized to include as mandatory attributes: (i) the patient data, (ii) the information about the patient's TDM platform (*IP address* and *port*); and it was completed by coding some foundamental methods to set and get the single attributes.
This activity creates a Patients Java object for each patient and all these Objects are stored in an ArrayList, named as `patients_list`, to keep track of the connected patients. In this way, all the patients' data are included in the list, making the access to the information easier.
Using *Intents* (described in pag. 57), the patients' ArrayList, is passed from this activity to the next one, in order to keep memory of the monitored patients.

```
// Creation of the Intent
Intent in = new Intent(this, PryvLoginActivity.class);
//Two information are carried by the intent: the number of monitored
    patients and the patient's ArrayList
in.putExtra("NUMBER_OF_PATIENTS", patients_list.size());
in.putParcelableArrayListExtra("PATIENTS", patients_list);
startActivity(in);
```

In addition, if desired, the anesthesiologist can enable the cloud sharing by checking the dedicated box in the actual Activity layout, visible in Fig. 5.8.b.

Pryv Activity

If the sharing has been previously selected, the doctor is redirected to the *Pryv Activity*, where he/she has to sign-in in the cloud platform (Fig. 5.8.c). This is made through a *webView*, which is an Android Widget for accessing web pages, in this case the Pryv login page.

Token Activity

The *Token Activity* (Fig. 5.8.d) appears after the Pryv activity, in case of sharing enabled. Since the data are stored in the patient's cloud account, this activity is needed in order to associate the patient's cloud account with the patient's information registered in the application.
To do this, the doctor has to select the patient's cloud account from a downloaded list of patients that have allowed the doctor to access data. For this purpose, it has been implemented the method `retrieveSharings` to download the list from the Pryv platform. This method establishes the Pryv connection to the doctor's Pryv account and allow to retrieve the patient's list.

```
private void retrieveSharings() {
   //Creation of the Pryv Connection, using the doctor's credentials
      (inserted in the Pryv activity)
   Connection connection = new Connection(PryvTokenActivity.this,
      Credentials.getUsername(), Credentials.getToken(),
      PryvLoginActivity.DOMAIN, false, new DBinitCallback());

   //Creation of the Stream
   Stream patientsStream = new Stream("patients", "patients");

   //Setting a filter for downloading only the last 10 patients'
      information stored in the Stream patientStream
   Filter filter = new Filter();
```

```
    filter.addStream(patientsStream);
    filter.setLimit(10);
    //getting the last 10 events (patients)
    connection.events.get(filter, new GetEventsCallback() {
    @Override
    public void cacheCallback(final List<Event> list, Map<String,
        Double> map) {
       PryvTokenActivity.this.runOnUiThread(new Runnable() {
       @Override
       public void run() {
          for (Event event : list) {
             String content = event.getContent().toString();
             try {
                //Retrieving username and token from the downloaded event
                JSONObject jsonEvent = new JSONObject(content);
                String username = jsonEvent.getString("username");
                String token = jsonEvent.getString("token");
                //adding the downloaded item to a list called events
                    (displayed on the screen)
                events.add(new Patients(username, token));
                myPryvAdapter.notifyDataSetChanged();
             } catch (JSONException e) {
             }
          }
       }
    });
}
```

In this way an unequivocal *key-token* is linked with the monitored patient. This procedure is going to be better analysed in the section 5.4.

Monitoring Activity, Device Activity and AddDevice Activity

The *Monitoring Activity* is the main activity of *AnControl*, where the connection with the TDM systems can be managed, the measurements can be started/stopped, new patients can be added to the network, the status of the connection can be monitored and, most importantly, the patients' received data can be continuously visualized. There are two activities associated with the *Monitoring Activity*:

1. **Device Activity**: The first thing to do in the *Monitoring Activity* is to establish the connection by clicking on the "Connect Device" button (Fig. 5.9.a). This will let the *Device Activity* to appear. The *Device Activity*, in Fig. 5.9.b, is necessary to associate the TDM device with the *patient object* created in the app. In order to do this, the doctor has to select the correct TDM platform from a list of paired TDM devices. After the selection, the device's information (Id, IP address and Port) is sent back, through an intent, to the *Monitoring Activity* and added to the `patient_list` ArrayList:

```
//Creation of an intent with some notification tags, the device
    name, IP address and port
Intent intent = new Intent();
   intent.putExtra("FROM", "DeviceActivity");
   intent.putExtra("CONNECTION REQUEST", connectionRequest);
   intent.putExtra("DEVICE NAME", intentName_add);
   intent.putExtra("DEVICE IP", intentIP_add);
   intent.putExtra("DEVICE PORT", intentPort_add);
setResult(MonitoringActivity.DEVICE_CHOOSEN, intent);

//Adding the device's information to the patient_list
patientsArrayList.get(nrPatientFromPopActivity)
        .setIPserver(deviceIPForConnection);
patientsArrayList.get(nrPatientFromPopActivity)
        .setPORTserver(devicePortForConnection);
```

2. **AddDevice Activity**: The doctor can also add a new device in the *AddDevice Activity* (Fig. 5.9.c), by inserting the name, ip address and port number of the TDM system.

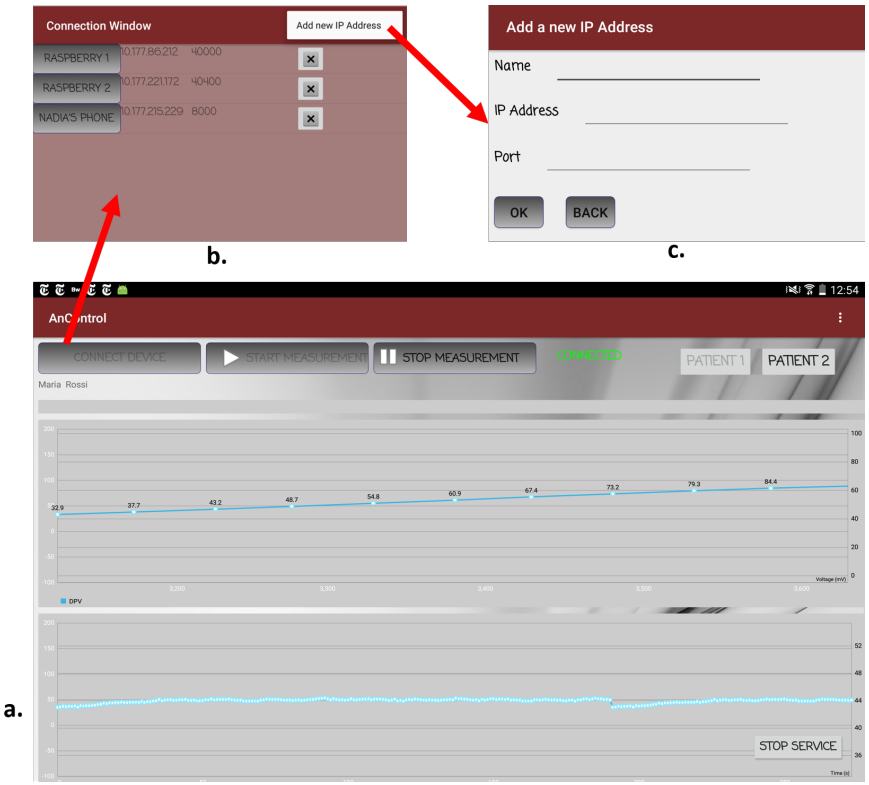

Figure 5.9: Detailed view of: a. Graph Activity; b. Device Activity; c. AddDevice Activity

After the device selection, the application tries to establish the connection with the patient's TDM and, in case of success, the anesthesiologist is allowed to start the measurements.

The *Monitoring Activity* consists of different interfaces, one for each patient; the doctor can switch between these interfaces by pressing the button with the patient's name of interest. Each interface can display up to three charts, depending on the number of drugs that the anesthesiologist wants to monitor. The charts are able to visualize not only drug concentration as a function of time, but also

CV, DPV and CA plots.

For the data plot, the library *MPAndroidChart* was chosen, because of its simplicity and good performance. This library offers different chart types; the *LineChart* type was used in the *Monitoring Activity*. In addition, this library enables interactions with the charts trough touch-gesture, to customize axes, to highlight values and it has a big amount of personalization features, such as colors, text font, legend, and others.

The method:

```
public void addEntryChart(LineChart chart, double current, double
    xAxis, String technique) {
   LineData data = chart.getData();
   if (data != null) {
      //If technique as CV and DPV are chosen, plot current as a
          function of voltage
      if (technique.equals("CV") || technique.equals("DPV")) {
         data.addEntry(new Entry((float) appliedvoltage, (float)
             current), 0);
         data.notifyDataChanged();
      } else if (technique.equals("CA")){
         //If CA is chosen, plot current as a function of time
         data.addEntry(new Entry(data.getEntryCount(), (float)
             current), 0);
         data.notifyDataChanged();
      }
      //Notification for the chart: data are changed
      chart.notifyDataSetChanged();
      //Limit number of visible entries
      chart.setVisibleXRangeMaximum(500);
      // move to the latest entry
      chart.moveViewToX((float)appliedvoltage);
   }
```

```
}
```

continuously receives current values, *voltage-time* values, drug's name and type of the technique (CV, CA, DPV or concentration) and plots them in the Linechart, changing axis' label depending on the chosen technique. Drug's name is important, because each Linechart is associated to an anesthetic, such as Propofol, Midazolam or Paracetamol and every data must go in the right chart.

The *Monitoring Activity* is strictly related to the WiFi *Service* (presented in the next paragraph), because between these two app components there is a bidirectional communication; they continuously exchange messages about various aspects, such as buttons pressed, received data and connection status. Therefore, in the *Monitoring Activity* different actions are defined as answer of different received message from the *Service*, and messages are generated and sent to the *Service*.

For instance, some messages that are sent from the *Monitor Activity* to the *Service* are related to the interaction user-UI, such as buttons pressing; depending on the id of the button pressed, a different message is sent to the service, as can be seen in the following code:

```
public void onClick(View v) {
   switch (v.getId()) {
      //If connect device button is pressed
      case R.id.connectDevicePatient1:
                intentForDeviceActivity = new
                    Intent(MonitoringActivity.this,
                    DeviceActivity.class);
                intentForDeviceActivity.putExtra("PATIENT", 0);
                startActivityForResult(intentForDeviceActivity,
                    CHOOSE_DEVICE_FOR_CONNECTION_PATIENT1);
      break;
      //If Stop Service button is pressed
```

```
        case R.id.button_StopService:
            Log.d(TAG, "service stopped");
            Intent in = new Intent(this, WiFiService.class);
                    stopService(in);
        break;
        //If stop measurement button is pressed
        case R.id.stopMeasurement:
            Log.d(TAG, "stop_measurement_patient1 pressed");
            MsgToWifiService("STOP MEASUREMENT", true);
        break;
        //If start measurement button is pressed
        case R.id.startMeasurement:
            Log.d(TAG, "start_measurement_patient1 pressed");
            MsgToWifiService("START MEASUREMENT", true);
        break;
    }
}
```

WiFi Service

The *WiFi Service* is the core of *AnControl*. It allows to perform tasks in background without making the UI activities heavier, because all the connections, data processing and data management are processed there.
After the TDM system has been selected, the *WiFi Service* handles the connection with it and manages the received data, converting them from binary values to double values. These actions are going to be explained in the next chapter.
In case of enabled cloud sharing, the *WiFi Service* also handles the connection with the Pryv platform. For this purpose it was used the Pryv Android/Java library, which enables the establishment of a secure connection, guaranteed trough the use of the *Hypertext Transfer Protocol Secure (HTTPS)*, and offers classes for creating Streams and Events. In particular, patient's data are stored in the

patient's Pryv account in a hierarchical structure, as shown in Fig. 5.10. There are four streams: one with the personal data of the patient and three with the name of the monitored drugs, *e.g.* Propofol, Midazolam and Paracetamol. For each stream, data are tagged and divided depending on the monitoring technique and *AnControl* creates events, ordered by upload date, with the data values.

The Pryv connection is established trough the implemented method `CreatePryvConnection`:

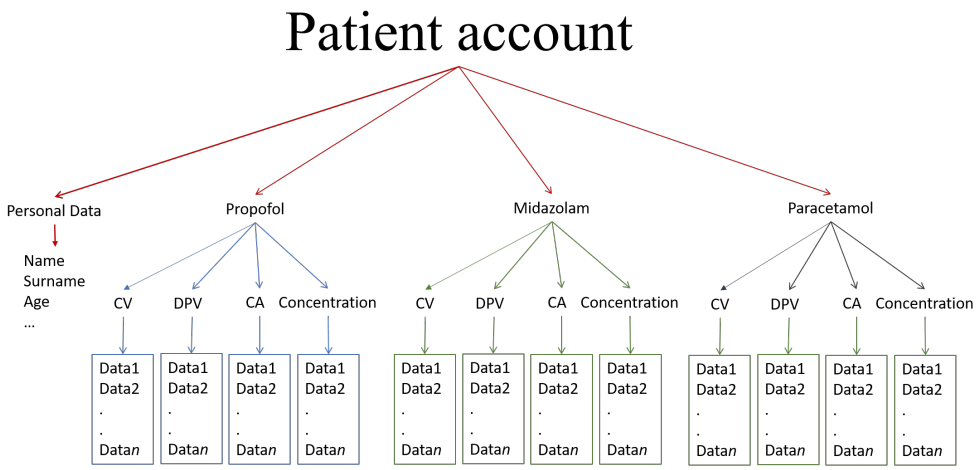

Figure 5.10: Sketch of the hierarchical structure in the cloud.

```
public static Connection CreatePryvConnection(String pryvUsername,
    String pryvToken) {
   connection = new Connection(context, pryvUsername, pryvToken,
      "pryv.me", true, new DBinitCallback());
   return connection;
}
```

that requires as input the patient's user-name and the token, generated when the the patient allows the doctor to visualize his/her surgery data. After the connection with Pryv, the *WiFi Service* creates a stream for each drug monitored and one for the patient's personal data:

```
public static String CreatePryvStream(String streamId, String
    streamName, Connection connection) {
```

```
    //Creating Pryv Stream
    Stream testStream = new Stream(streamId, streamName);
    Filter scope = new Filter();
    scope.addStream(testStream);
    connection.setupCacheScope(scope);
    connection.streams.create(testStream, new StreamsCallback())
    return streamId;
}
```

Finally the *Service*, while receiving data from the TDM platform, creates an event (which goes in the correct stream) for each received value, containing the y-value and the x-value of the data. For instance, in case of CV or DPV data (current as a function of voltage), the method `SendToPryv` creates an event containing the current value and the voltage value as follow:

```
public static void SendToPryv(double currentReceived,
     double xAxisReceived, String streamIdReceived,
     String technique, Connection connection) {
    Event events = new Event();
    events.setStreamId(streamIdReceived);
    if (technique.equals("DPV") || technique.equals("CV")) {
        events.setType("numset/voltammetry");
        JSONObject object = new JSONObject();
        try {
            object.put("current",currentReceived);
            object.put("voltage",xAxisReceived);
        } catch (JSONException e) {
            e.printStackTrace();
        }
        String Data = object.toString();
        events.setContent(Data);
    }
```

```
    connection.events.create(events, new EventsCallback()
}
```

In addition, the *WiFi Service* stores all the received data also in the external memory of the smart-device, to guarantee an appropriate recording of the patient's data even in case of cloud unavailability [21]. For each patient and for each monitored drug, the *WiFi Service* creates Excel files with all the measured data, organized by surgery's date.

5.2.1 Smart-watch

A smart-watch, worn by the anesthesiologist, enriches the architecture by warning the doctor in case of emergency.
The smart-watch is paired with the smart-device on which the Android app is running and, if a measured concentration is out-of-the-safe-range for a certain period of time, the Android app pushes an alert to the smart-watch. The alert consists of a strong vibration and a pop-up visible on the screen of the smart-watch with the information about the out-of-range drug and the name of the critical patient (Fig. 5.11). For this purpose, in the WiFi Service it was implemented the

Figure 5.11: Screenshots of the smart-watch alert view

following method `SendToSmartwatch`, that handles the smart-watch notifications

in case of emergency:

```
public static void SendToSmartwatch (int nr_patient, String drug) {
   int notificationId = 001;
   String patientName = patients_list.get(nr_patient).getName() + " " +
       patients_list.get(nr_patient).getSurname();
   NotificationCompat.Builder notificationBuilder = new
       NotificationCompat.Builder(context).setSmallIcon(R.mipmap.ic_launcher).setContent
       of range " + " " + patientName + " " + "On " + drug);
   notificationBuilder.setVibrate(new long[]{1000, 1000, 1000, 1000,
       1000, 1000, 1000});
   NotificationManagerCompat notificationManager =
       NotificationManagerCompat.from(context);
   notificationManager.notify(notificationId,
       notificationBuilder.build());
}
```

5.3 Cloud Solution and WebApp

As already said in section 4.2, the Pryv middle-ware solution was adopted as back-end cloud system.

If the cloud sharing is selected by the doctor in the *Patient Activity* of the Android app, the *AnControl* application will also continuously send the received data to the patient's cloud account in addition to the data visualization on the smart-device. A web application, from now on referred as WebApp, can be used to manage sharing and to access the shared data enabling remote visualization. Indeed, the WebApp, developed in Javascript, HTML and CSS, offers different user-friendly interfaces for patients and doctors, trough which health data can be shared while respecting security regulations.

In particular, the patient interface allows to share patient's surgery data with the doctor, as in Fig. 5.12.a; the WebApp for the doctor shows the list of all the

patients that have shared their data streams, as in Fig. 5.12.b.

By clicking on one of the patients' sharing, the doctor can visualize the cor-

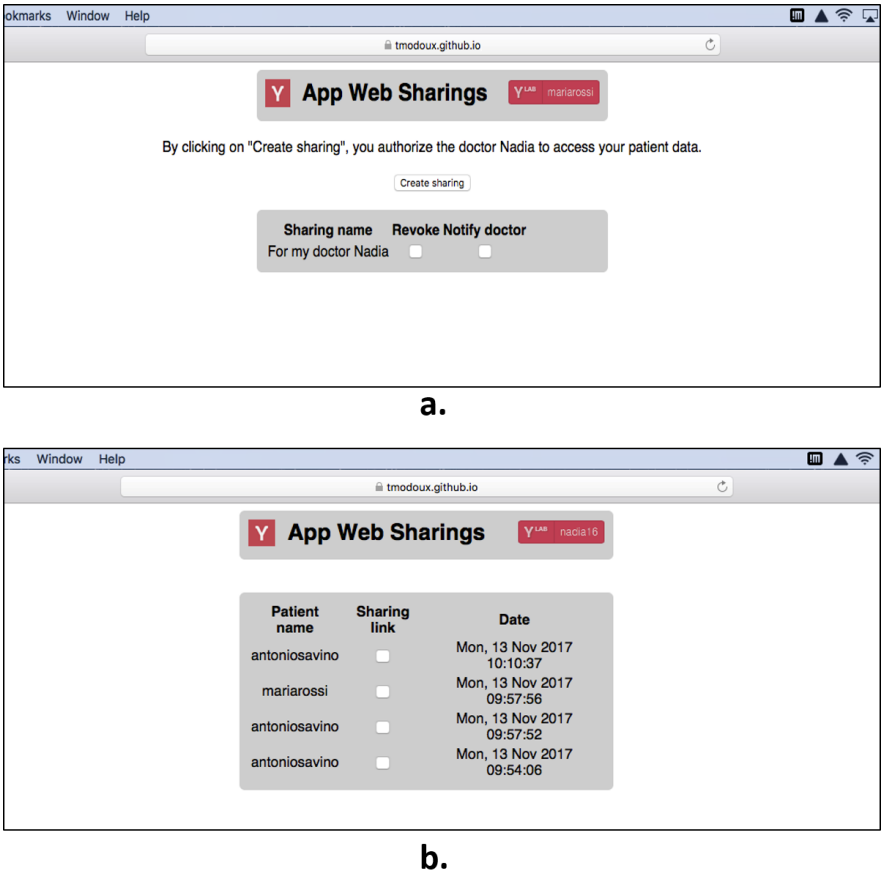

Figure 5.12: Screenshots of the WebApp: a. patient's interface; b. doctor's interface

responding data, thanks to an interface with multiple graphs, developed using *Ploty javascript library* (Fig. 5.13). Thanks to this solution, a medical specialist geographically far from the hospital area can remotely visualize the data during or after the surgery. This can be of extreme advantage in case of rural areas or underdeveloped Countries.

5.4 Standardized Procedures

To set the developed mIoT system, both patient and doctor have to follow some procedures, summarized as workflow in Fig. 5.4.

Figure 5.13: Screenshots of the WebApp data visualization tool

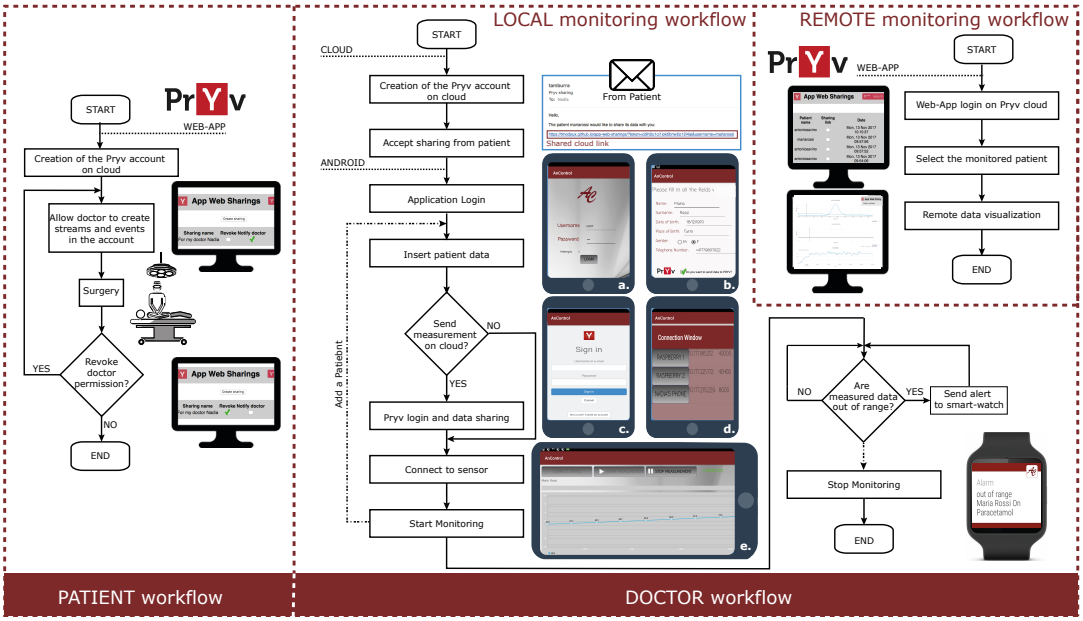

Figure 5.14: Work-flow charts for patient-side and doctor-side procedures. In particular, doctor-side can be subdivided in local and remote monitoring [21].

5.4.1 Patient-side Work-flow

In regulated environments such as health-care patient privacy must be always guaranteed. Data sharing and data access on cloud can be done only with prior consent of the patient. For this reason, it has been implemented a procedure for

the patient in order to enable cloud sharing. The patient firstly has to sign in with his/her Pryv account (or register a new one), then the patient can decide to share a subset of data with a doctor. To do this, in the previously described WebApp, a consent form is presented to the patient which, by clicking on a button, explicitly authorizes the doctor to access his/her data during the surgery. This action results in the creation of a shared access/token between the patient and the doctor. The patient can revoke the authorization at any time.

5.4.2 Doctor-side Work-flow

Local Monitoring Work-flow

After the patient's authorization, the doctor is notified trough an email with the patient's name and a sharing-link, which is created by concatenating the WebApp url with the sharing key/token. By clicking on this link, the doctor, after signing-in for his/her Pryv's account, accepts to take the mentioned patient in charge and the cloud setup pre-surgery is set.
During the surgery, thanks to the Android app, the anesthesiologist can continuously monitor anesthetics concentration for all the patient under his/her responsibility. To access *AnControl*, the doctor firstly has to sign in trough the *Login Activity* (Fig. 5.4.a). Afterwards, the doctor has to insert patient's personal data in the *Patient Activity* (Fig. 5.4.b) and can select a check-box to enable the cloud sharing. If the cloud sharing is enabled, the anesthesiologist is asked to sign-in also on his/her Pryv account (Fig. 5.4.c) and to link the patient information with the Pryv sharing. Then the doctor has to connect *AnControl* with the TDM system, selecting it among the list shown in the *Device Activity* (Fig. 5.4.d). At this point the anesthesiologist can start the measurements and visualize data in the *Monitoring Activity* (Fig. 5.4.e). For each measured drug, if out-of-range values are measured for a certain period of time, an alert is sent to the smart-watch worn by the doctor. While monitoring, the doctor can always add patients to the network, following the same steps previously described [21].

Remote Monitoring Work-flow

Any allowed medical doctor can remotely access data measured during a surgery, *e.g.* for teleconsulting, or after, *e.g.* for data analysis. To do this, the allowed doctor access the WebApp, which is also used by the patient, but, after signing-in, the interface is different: it presents a list of accepted patients sharing, alongside with a creation date and a link to a visualization tool.
These multiple graphs allow the doctor to easily supervise each patient by navigating through the data in time. The visualization tool is another simple web application that plot patient data in graphs using *Plotly javascript library* and it knows exactly which data to plot and from which patient is sent since it consider only the sharing link as input.

Chapter 6

Protocol and Validation

In this chapter it is going to be described (i) the data transfer communication technology used for the developed mIoT system, (ii) the protocol used for the data transmission between the TDM platform and the Android application and (iii) the validation of the system.

6.1 Network communication technology

Data can be transmitted from one device to another using different types of communication protocols.

The mIoT system presented in this thesis relies on the Wi-Fi communication, which is currently one of the most dominant and widely spread data transfer technology. More specifically, Wi-Fi is a networking technology based on the IEEE 802.11 standard, that allows the devices connected in a network to communicate each other wirelessly. It commonly uses the 2.4 GHz and the 5.8 GHz *Radio Frequency (RF)* bands.

If the *Wireless Local Area Network (WLAN)* is connected to the Internet trough a router, Wi-Fi technology can be used to provide Internet access to the devices connected to the network; if the router is not connected to the Internet, *e.g.* via *Digital Subscriber Line* (DSL), it is still possible to have Wi-Fi connection, but without Internet access.

To enable the bidirectional communication among different devices, a communication *protocol* has to be defined. A communication protocol defines a set of rules that needs to be respected by the peers who are going to exchange information through the network.

The *ISO/OSI* model, which is a globally accepted communication standard, structures the overall transmission into seven *layers* that are organized in a stack. Each layer contains protocols and performs specific functions are useful for the transmission. Fig. 6.1 shows the data-flow through the stack structure. Starting from the transmitter.side, data goes from the upper layer of the stack towards the lower layer; while for the receiver-side, data goes from the lower layer towards the upper layer.

The seven layers are:

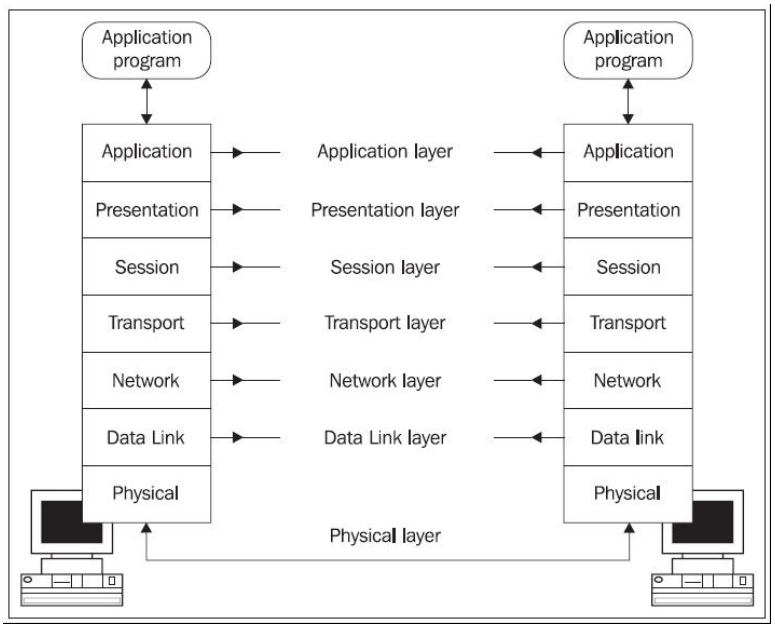

Figure 6.1: ISO/OSI model [25].

1. **Physical Layer** that regards the interaction between the device and transmission medium. It converts digital data into a bit stream for the transmission over the network;

2. **Data Link Layer** is responsible for detecting and correcting errors that

may occur during the transmission. Therefore, when a received packet can not be corrected, this layer is responsible to send again the data;

3. **Network Layer** handles data transmission across the network. It selects a path, which exploits one or more networks, to reach the destination and it manages the traffic control. However it does not guarantee the reliability of the transmission. The *Internet Protocol (IP)* runs at this layer;

4. **Transport Layer** ensures that messages are delivered without errors and in the correct sequence [26]. At the transmitter-side this layer splits messages into smaller segment suitable for the network layer; at the receiver side, this layer builds again the segments. Protocols such as *Transmission Control Protocol (TCP)* and the *User Datagram Protocol (UDP)* operates at this layer;

5. **Session Layer** controls the network connections. It is responsible for establishment, maintenance and termination of the network session;

6. **Presentation Layer** translates data from the application layer into the format that is going to be used across the network. At the transmitter-side it performs data compression, data encryption, data formatting, while on the receiver-side it translates the data back to the format that is understandable for the application layer;

7. **Application Layer**. This layer is responsible for providing access interfaces and services trough which the user can use the network, *e.g. Hypertext Transfer Protocol (HTTP)*, *Simple Mail Transfer Protocol (SMTP)*.

Highly related to the ISO/OSI model is the *Internet protocol stack*, which is made up of four layers and each of them includes one or more layers of the ISO/OSI model (Fig. 6.2). The Internet stack is similar to the ISO/OSI model, but it defines a set of communication protocols used on the Internet. It is also known as *TCP/IP model*, since the foundational protocols are the TCP and the IP.

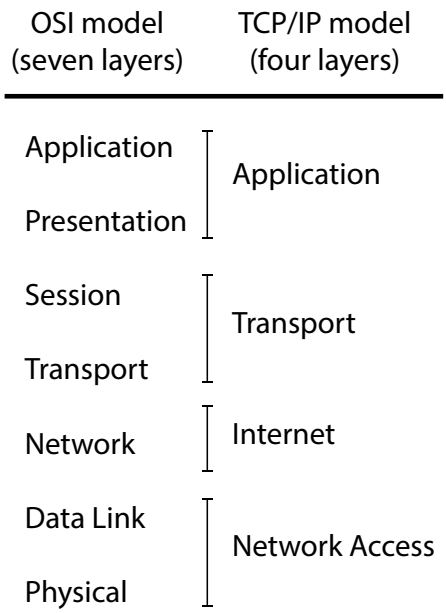

Figure 6.2: TCP/IP model [26].

The IP is a network layer protocol that contains the information of the receiver address and performs routing functions by choosing the path over which the data are sent. An IP address is a 32-bit address which uniquely identifies every host connected to the Internet. According to the IP address, IP decides if the packet delivery has to be done directly or indirectly. Direct delivery occurs when the transmitter and the receiver are in the same physical network, otherwise the deliver is indirect because the packet goes from router to router through a number of networks until it reaches the final destination [26].

IP is an unreliable and connectionless protocol, since the data delivery is not guaranteed and the data packets travel trough different paths which may lead to out of sequence transmission errors [26].

To reach reliability, IP has to be paired with a standard protocol, such as TCP, that provides a secure data transfer, thanks to flow/error controls. At the transmitter-side, TCP creates data packets and sends them to the lower layer; on the receiver-side, TCP receives and controls packets, extracts data and puts them in the correct order and finally delivers them as a byte-stream to the receiving application program [26]. TCP has also a mechanism to verify the reception and

to manage communication problems: in this case data are stored in a buffer and sent back when suitable conditions are restored

A typical architecture of the application layer, that it has been also used in the development of the here presented mIoT system, is the *client-server architecture*. In this configuration there is an always-on server, that provides the services to the client whenever required. The server can establish connections with many clients, but a client can only communicate with servers.

Each TCP packet contains: (i) the port numbers of the sending and the receiving applications, (ii) the transmitter and the receiver IP addresses. These parameters uniquely identify each connection [26]. The combination of the port number and the IP address is called *Socket*. The application sends the messages to the transport layer trough the socket, that can be opened and closed for each message (*non-persistent connections*) or it can remain active for all the time of the transmission (more than one message). In this case the socket remains open until it is manually closed or if it is not used for a certain period of time (*persistent connections*) [62].

6.2 System Communication Protocol

The client-server architecture has been used for establishing the Wi-Fi communication between the TDM system and the Android application. In particular, the RPi of the TDM platform is the server and the app is the client that requests the measurements.

The Android app handles the connection and manages the data in the *Wi-Fi Service* (5.2). After the selection of the patient's TDM platform in the *Device Activity*, the service performs the connection request. Knowing the IP address and the port number of the TDM, the application requires the establishment of the socket trough this code:

```
socketThread = new Socket(patients_list.get(nr_patient).getIPserver(),
    patients_list.get(nr_patient).getPORTserver());
```

A socket is opened for each patient.

After the socket is opened, the application can send messages to the server trough the developed `send2RPiMessage` method:

```
public void send2RPiMessage(byte c[], Socket socketThread1) throws
    IOException {
   OutputStream outputStream1 = socketThread1.getOutputStream();
   //Send the connecting request to the server
   outputStream1.write(c);
}
```

where `c` is the byte value of the message.

As shown in Fig. 6.3, some steps have to be done after the socket is open. To

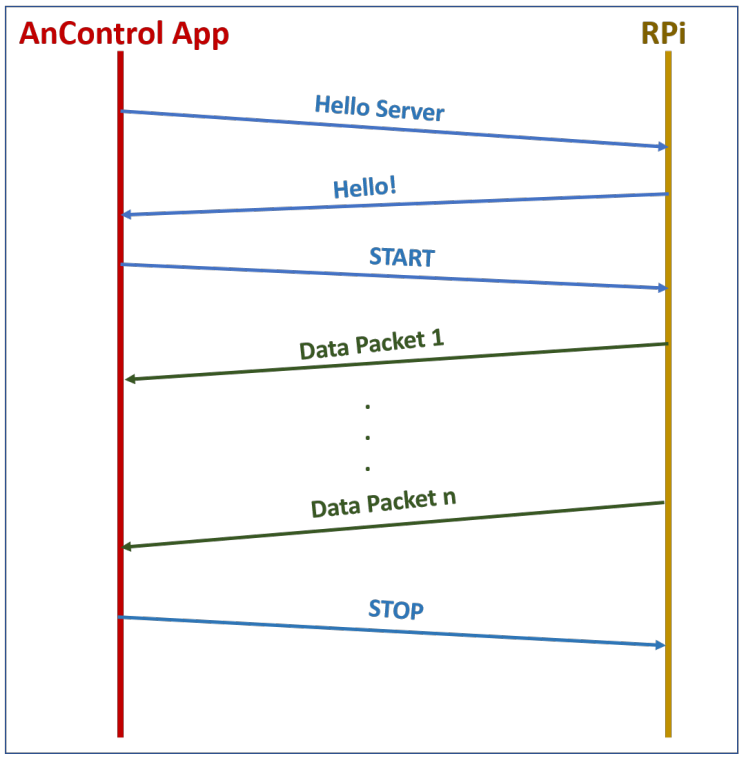

Figure 6.3: Communication protocol between AnControl and the TDM platform.

verify that the socket is correctly opened, immediately after the opening request

the application sends to the server the message "Hello server" and, if the server answers "Hello", it means that the connection is established and that the server is ready to send measured data. Obviously, the messages that are exchanged between the server and the client to verify the socket can be chosen in a arbitrary way. To read the messages received from the server, the following code is used:

```
//Reading Server's answer
serverOnline = new
    DataInputStream(socketHashMap.get(nr_patient).getInputStream());
byte[] received_bytes = new byte[6];
serverOnline.readFully(received_bytes);
```

When the anesthesiologist presses the *start-measurement* button in the *Monitoring Activity*, the message "START" is sent to the server. This query starts the measurements and enables the transmission of the measured data to the app.
As shown in Fig. 6.4, each measured data is sent in a six-bytes packet, following a specific protocol:

1. the first byte contains information about the monitored drug (such as Propofol, Midazolam or Paracetamol) and the adopted technique (which can be concentration per time, CA, DPV or CV);

2. the second and the third bytes contain the measured current value;

3. the fourth and the fifth bytes contain the applied voltage value. This value is considered just in case of CV and DPV. In case of concentration or CA the time value is considered;

4. the last byte is the *stop byte*, that identifies the end of the data packet.

When the anesthesiologist presses the *stop-measurements* button in the *Monitoring Activity*, the application sends the message "STOP" to the server and the data transmissio ends. However, whenever the doctor presses again the *start-measurement* button, the server restarts the transmission of the measurements.

If the doctor presses the button *stop-service* in the *Monitoring Activity*, the socket is definitively closed.

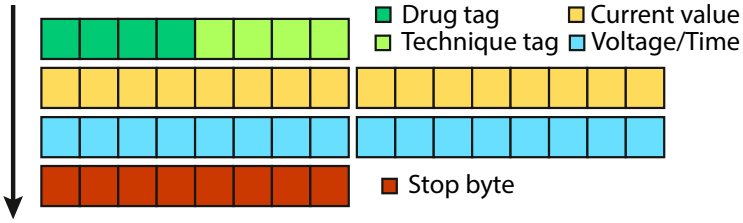

Figure 6.4: Transmitted data packet between RPi-driven TDM system and Android app [21].

6.3 System validation

The system has been validated by simulating a network with two anesthetized patients, as in Fig. 6.5.

Each patient's TDM is simulated trough a RPi. A tablet Galaxy Note pro 12.2in with Android 5.1.1 Lollipop paired with a Sony Smart-watch 3 has been used for the anesthesiologist. Finally a laptop for accessing the Pryv WebApp has been included. Adopting this architecture, the following features has been validated:

- The Android app can simultaneously communicate via Wi-Fi with the RPi and with the cloud solution;
- Multiple patients can be monitored at the same time, allowing the anesthesiologist to easily follow different surgeries in parallel;
- The patient access in the WebApp and shares a subset of data with the doctor. An email with the link to the shared data is automatically sent to the doctor. Then, it has been also tested the revoke of the patient sharing;
- After receiving the email from the patient, the doctor use the share link to access the WebApp and visualize the cloud data;

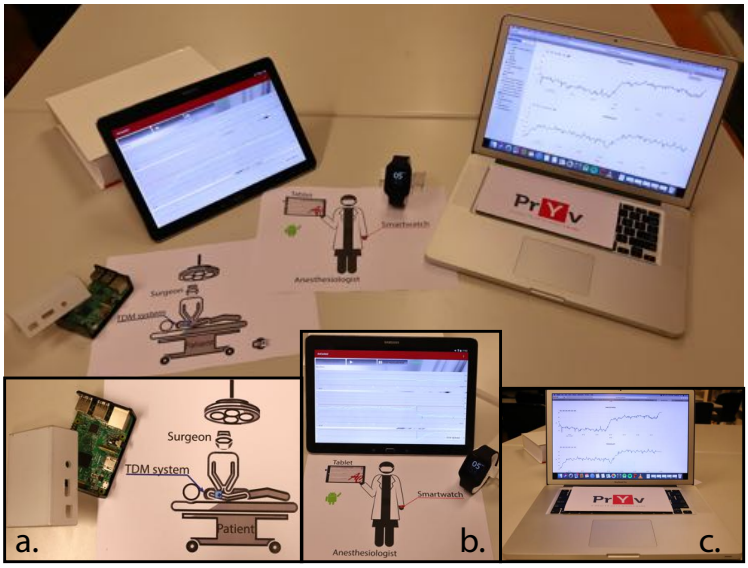

Figure 6.5: Validation setup. Full monitoring architecture on top, while single actors are on bottom. a.) The patient connected with the TDM system, b.) the anesthesiologist holding a tablet, paired with a smartwatch and c.) the Pryv cloud system [27].

- The Android application continuously receives data from the RPi and continuously plots data on the charts;
- In case of out-of-range measured parameters from one connected patient, an alert notification is sent to the smart-watch worn by the doctor. The notification consists in a strong vibration and a pop-up displaying the name of the critical patient, and the out-of-range parameter. The alert is then sent when more than n received values exceed the physiological threshold. Both the n value and the threshold depend on the measured parameter;
- It has been evaluated the latency time for an event creation call on Pryv API as under 100 ms, which is comparable to a standard HTTP request without any overhead. This parameter is influenced by the quality of Internet connection, the server location and the size of the payload (Switzerland and events with double-precision floating-point numbers, respectively in this case).

Chapter 7

Conclusions and Future Works

In this thesis a novel mIoT cloud-based network for anesthesia monitoring is presented. Its design, realization and validation have been widely described and analysed in the present work. The importance of this system relies on the fact that, during anesthesia practices, a correct drug dosing have to be continuously guaranteed to avoid undesirables side-effects on the sedated patient. Therefore, guaranteeing a continuous monitoring of the patient undergoing anesthesia is of vital importance.

The central actor of the system is an Android application, running on the anesthesiologist's tablet/smart-phone, that bidirectionally communicate, via Wi-Fi, with a RPi-driven TDM platform that continuously monitor the anesthetics' concentrations in patient's blood. By using the application, the doctor is enabled to simultaneously monitor multiple patient under his/her responsibility. The system is enriched with a cloud back-end solution, that enables remote monitoring trough a web application. In this way teleconsulting functions are also offered. The cloud sharing is enabled trough the Android app, that can securely share the measured data on the cloud platform. A standard communication protocol, such as HTTPS, guarantees secure transmission of sensible data. Furthermore a smart-watch, worn by the anesthesiologist, is used to alert the doctor in case of out-of-range parameters, ensuring a rapid intervention.

To guarantee a secure and reliable architecture, along with the observance of privacy regulations, some procedures have been developed and validated. A login procedure in the Android app allows protection from unauthorized accesses. An *ad-hoc* cloud solution for health data management, offered by Pryv, guarantees privacy protection and a secure storage of the patient's data. Moreover, since sensitive patient's personal data are involved, it has been designed a procedure to get the patient's permissions for cloud storage and for remotely accessing data.
The system has been validated by simulating a network with two anesthetized patients. Each patient was simulated trough a RPi that was continuously sending real measured data trough Wi-Fi. The Android application was verified to continuously plot the received data on charts and to successfully alert the doctor by pushing an alert to the smart-watch in case of out-of-range. Finally, the sharing to the cloud platform and the remote access trough the developed web application has been validated as well.
Although in literature there is a relevant number of IoT systems for health-care applications, there is still a small number for continuous monitoring in medical environments. In this context, the presented mIoT cloud-based system is extremely flexible and provides a useful tool to be used in various health-care monitoring applications. Indeed, the flexibility and portability of the developed architecture allow the connection with any medical device that can wirelessy send data.

7.1 Future Works

Further analysis can be done on the enhancement of the security level for the Wi-Fi communication between the TDM platform and the Android application. Furthermore, it would be nice to include additional protections in case of Wi-Fi unavailability, for example by temporarily switching to other communication technologies, such as bluetooth or by developing ad-hoc backup transmission procedures.

Chapter 8

Acknowledgments

Firstly, I would like to thank my supervisor in EPFL, Dr. MER Sandro Carrara, for the opportunity to work with him and with his wonderful team. His guidance and suggestions helped me in all the time of developing and writing of this thesis.
I would also like to express my sincere gratitude to the PhD student at EPFL Francesca Stradolini for her constant presence, support and guidance, during and after my period in Lausanne. Her dynamism, smartness and creativity encouraged me to always improve and to never give up if things don't always work as the they should. It has been a real pleasure to work with her.
I thank all the brilliant people of the Laboratory of Integrated Systems (LSI) for the warm welcoming, for all the incredible days spent together, inside and outside the laboratory, and for being a little family during my period in Lausanne.
Last but not least, a very special gratitude goes to my parents, my brother and sister for always supporting me, not only during the thesis, but during my entire life.

Bibliography

[1] "www.rfpage.com/how-wireless-technology-used-medicine-healthcare/," 2017.

[2] Y. Ishizawa, "Mechanisms of anesthetic actions and the brain," *Journal of Anesthesia*, vol. 21, no. 2, pp. 187–199, 2007.

[3] F. Stradolini, T. Elboshra, A. Biscontini, G. De Micheli, and S. Carrara, "Simultaneous monitoring of anesthetics and therapeutic compounds with a portable multichannel potentiostat," *Proceedings - IEEE International Symposium on Circuits and Systems*, vol. 2016-July, pp. 834–837, 2016.

[4] A. Al-fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things : A Survey on Enabling Internet of Things : A Survey on Enabling Technologies , Protocols and Applications," *Ieee*, vol. 17, no. November, pp. 2347–2376, 2015.

[5] D. V. Dimitrov, "Medical internet of things and big data in healthcare," *Healthcare Informatics Research*, vol. 22, no. 3, pp. 156–163, 2016.

[6] S. I. Lakkis and M. Elshakankiri, "IoT based emergency and operational services in medical care systems," in *2017 Internet of Things Business Models, Users, and Networks*, pp. 1–5, IEEE, nov 2017.

[7] C. O. Rolim, F. L. Koch, C. B. Westphall, J. Werner, A. Fracalossi, and G. S. Salvador, "A cloud computing solution for patient's data collection in health care institutions," *2nd International Conference on eHealth, Telemedicine,*

and Social Medicine, eTELEMED 2010, Includes MLMB 2010; BUSMMed 2010, no. ii, pp. 95–99, 2010.

[8] "www.statista.com/topics/840/smartphones/."

[9] M. Wazid, S. Zeadally, A. K. Das, and V. Odelu, "Analysis of Security Protocols for Mobile Healthcare," *Journal of Medical Systems*, vol. 40, no. 11, 2016.

[10] O. Banos, C. Villalonga, M. Damas, P. Gloesekoetter, H. Pomares, and I. Rojas, "PhysioDroid: Combining Wearable Health Sensors and Mobile Devices for a Ubiquitous, Continuous, and Personal Monitoring," *Scientific World Journal*, vol. 2014, 2014.

[11] I. Zagan, V. G. Gaitan, A. I. Petrariu, and A. Brezulianu, "Healthcare IoT m-greenCARDIO remote cardiac monitoring system - concept, theory of operation and implementation," *Advances in Electrical and Computer Engineering*, vol. 17, no. 2, pp. 23–30, 2017.

[12] P. Singh and A. Jasuja, "IoT Based Low-Cost Distant Patient ECG Monitoring System," in *2017 International Conference on Computing, Communication and Automation (ICCCA)*, pp. 1330–1334, 2017.

[13] O. Banos, J. A. Moral-Munoz, I. Diaz-Reyes, M. Arroyo-Morales, M. Damas, E. Herrera-Viedma, C. S. Hong, S. Lee, H. Pomares, I. Rojas, and C. Villalonga, "MDurance: A novel mobile health system to support trunk endurance assessment," *Sensors (Switzerland)*, vol. 15, no. 6, pp. 13159–13183, 2015.

[14] "StatCounter Global Stats - http://gs.statcounter.com/os-market-share#yearly-2017-2018-bar."

[15] Wikipedia, "Android - https://en.wikipedia.org/wiki/Android_(operating_system)," 2017.

[16] C. Gunther, "Android Updates - www.gottabemobile.com/android-p-news-features-release/," 2018.

[17] M. Simon, "Watches to Android Wear 2.0 | Greenbot - https://www.greenbot.com/article/3181373/android/fossil-updates-its-three-q-watches-to-android-wear-2-0.ht."

[18] E. Burnette, *Hello, Android - Introducing Google's Mobile Development Platform*. 2009.

[19] Google Android, "Introduction to Android | Android Developers - http://developer.android.com/about/index.html," 2015.

[20] "Pryv Web Page - https://pryv.com/."

[21] F. Stradolini, N. Tamburrano, T. Modoux, A. Tuoheti, D. Demarchi, and S. Carrara, "IoT for Telemedicine Practices enabled by an Android TM Application with Cloud System Integration," in *ISCAS Conference, 2018*.

[22] "www.dropsens.com/en/screen_printed_electrodes_pag.html."

[23] F. Stradolini, A. Tuoheti, P. M. Ros, D. Demarchi, and S. Carrara, "Raspberry pi based system for portable and simultaneous monitoring of anesthetics and therapeutic compounds," *Proceedings - 2017 1st New Generation of CAS, NGCAS 2017*, pp. 101–104, 2017.

[24] D. Andrienko, "Cyclic Voltammetry," *Cyclic Voltammetry*, pp. 1–12, 2008.

[25] "ISO/OSI Network Model - http://www.shankysportal.com/blog/iso_osi_network_model/2014-02-13-36."

[26] M. Y. Rhee, *Wireless Mobile Internet Security*. John Wiley & Sons, Ltd, 2013.

[27] F. Stradolini, N. Tamburrano, T. Modoux, A. Tuoheti, D. Demarchi, and S. Carrara, "Live Demonstration : An IoT Cloud-Based Architecture for Anesthesia Monitoring," in *Live Demonstration, Iscas 2018*, 2018.

[28] A. Kamilaris and A. Pitsillides, "Mobile phone computing and the Internet of Things: a survey," *IEEE Internet of Things (on print)*, vol. 3, no. 6, pp. 1–13, 2016.

[29] C. Otto, A. Milenković, C. Sanders, and E. Jovanov, "System architecture of a wireless body area sensor network for ubiquitous health monitoring," *Journal of Mobile Multimedia*, vol. 1, no. 4, pp. 307–326, 2006.

[30] U. Satija, B. Ramkumar, and S. M. Manikandan, "Real-Time Signal Quality-Aware ECG Telemetry System for IoT-Based Health Care Monitoring," *IEEE Internet of Things Journal*, vol. 4, no. 3, pp. 815–823, 2017.

[31] M. T. Nkosi and F. Mekuria, "Cloud computing for enhanced mobile health applications," *Proceedings - 2nd IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2010*, pp. 629–633, 2010.

[32] M. Thangaraj, P. P. Ponmalar, and S. Anuradha, "Internet Of Things (IOT) enabled smart autonomous hospital management system - A real world health care use case with the technology drivers," *2015 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, pp. 1–8, 2015.

[33] M. J. Brown, A. Subramanian, T. B. Curry, D. J. Kor, S. L. Moran, and T. R. Rohleder, "Improving operating room productivity via parallel anesthesia processing," *International Journal of Health Care Quality Assurance*, vol. 27, no. 8, pp. 697–706, 2014.

[34] G. Kotsovolis and G. Komninos, "Awareness during anesthesia: How sure can we be that the patient is sleeping indeed?," *Hippokratia*, vol. 13, no. 2, pp. 83–89, 2009.

[35] E. N. Brown, R. Lydic, and N. D. Schiff, "General Anesthesia, Sleep, and Coma," *The new england journal o f medicine*, 2010.

[36] M. M. Ghoneim, "Awareness during Anesthesia," *Anesthesiology*, vol. 92, no. 2, p. 597, 2000.

[37] G. A. Mashour and M. S. Avidan, "Intraoperative awareness: Controversies and non-controversies," *British Journal of Anaesthesia*, vol. 115, no. March, pp. I20–I26, 2015.

[38] T. D. Egan, "Total Intravenous Anesthesia Versus Inhalation Anesthesia: A Drug Delivery Perspective," *Journal of Cardiothoracic and Vascular Anesthesia*, vol. 29, no. S1, pp. S3–S6, 2015.

[39] J. Sandham, "Total intravenous anaesthesia (TIVA) - www.ebme.co.uk/articles/clinical-engineering/95-total-intravenous-anaesthesia-tiva."

[40] F. Kivlehan, E. Chaum, and E. Lindner, "Propofol detection and quantification in human blood: the promise of feedback controlled, closed-loop anesthesia," *The Analyst*, vol. 140, no. 1, pp. 98–106, 2015.

[41] F. Guarracino, F. Lapolla, C. Cariello, A. Danella, L. Doroni, R. Baldassarri, A. Boldrini, and M. L. Volpe, "Target controlled infusion: TCI," *Minerva Anestesiologica*, vol. 71, no. 6, pp. 335–337, 2005.

[42] N. Jagannivas and D. Hepsiba, "Control Of Anaesthesia Concentration Using Model Based Controller," *International Journal of Innovative Research in Technology*, vol. 1, no. 10, pp. 237–244, 2014.

[43] N. Jagannivas and D. Hepsiba, "Control Of Anaesthesia Concentration Using Model Based Controller," *International Journal of Innovative Research in Technology*, vol. 1, no. 10, pp. 237–244, 2014.

[44] J. P. Abenstein, "Is BIS monitoring cost-effective?," *Proceedings of the 31st Annual International Conference of the IEEE Engineering in Medicine and Biology Society: Engineering the Future of Biomedicine, EMBC 2009*, pp. 7041–7044, 2009.

[45] L. Atzori, A. Iera, G. Morabito, and A. Diee, "The Internet of Things: A survey," *Computer Networks xxx*, 2010.

[46] M. Paschou, E. Sakkopoulos, E. Sourla, and A. Tsakalidis, "Health Internet of Things: Metrics and methods for efficient data transfer," *Simulation Modelling Practice and Theory*, vol. 34, pp. 186–199, 2012.

[47] H. Alami, M.-P. Gagnon, and J.-P. Fortin, "Telehealth in Light of Cloud Computing : Clinical , Technological , Regulatory and Policy Issues," *Journal of the International Society for Telemedicine and eHealth*, vol. 4, no. April, p. 5, 2016.

[48] H. H. Nguyen, F. Mirza, M. A. Naeem, and M. Nguyen, "A review on IoT healthcare monitoring applications and a vision for transforming sensor data into real-time clinical feedback," *Proceedings of the 2017 IEEE 21st International Conference on Computer Supported Cooperative Work in Design, CSCWD 2017*, pp. 257–262, 2017.

[49] M. D. Lakshmi and J. P. M. Dhas, "An open source private cloud solution for rural healthcare," *2011 - International Conference on Signal Processing, Communication, Computing and Networking Technologies, ICSCCN-2011*, no. Icsccn, pp. 670–674, 2011.

[50] P. Matlani and N. D. Londhe, "A cloud computing based telemedicine service," *IEEE EMBS Special Topic Conference on Point-of-Care (POC) Healthcare Technologies: Synergy Towards Better Global Healthcare, PHT 2013*, pp. 326–330, 2013.

[51] D. Thilakanathan, S. Chen, S. Nepal, R. Calvo, and L. Alem, "A platform for secure monitoring and sharing of generic health data in the Cloud," *Future Generation Computer Systems*, vol. 35, pp. 102–113, 2014.

[52] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, and A. Ghalsasi, "Cloud computing - The business perspective," *Decision Support Systems*, vol. 51, no. 1, pp. 176–189, 2011.

[53] R. Wooten, R. Klink, F. Sinek, Y. Bai, and M. Sharma, "Design and implementation of a secure healthcare social cloud system," *Proceedings - 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2012*, pp. 805–810, 2012.

[54] B. Martínez-Pérez, I. De La Torre-Díez, and M. López-Coronado, "Mobile health applications for the most prevalent conditions by the world health organization: Review and analysis," *Journal of Medical Internet Research*, vol. 15, no. 6, 2013.

[55] M. N. Boulos, S. Wheeler, C. Tavares, and R. Jones, "How smartphones are changing the face of mobile and participatory healthcare: An overview, with example from eCAALYX," *BioMedical Engineering Online*, vol. 10, pp. 1–14, 2011.

[56] A. B. Pawar, S. Ghumbre, and Ieee, "A survey on IoT applications, security challenges and counter measures," *2016 International Conference on Computing, Analytics and Security Trends*, pp. 294–299, 2016.

[57] V. Koufi, F. Malamateniou, and G. Vassilacopoulos, "Ubiquitous access to cloud emergency medical services," in *Proceedings of the 10th IEEE International Conference on Information Technology and Applications in Biomedicine*, pp. 1–4, 2010.

[58] P. Gilski and J. Stefanski, "Android OS: A Review," *TEM Journal*, vol. 4, no. 1, pp. 116–120, 2015.

[59] F. Faridbod, V. K. Gupta, and H. A. Zamani, "Electrochemical Sensors and Biosensors," *International Journal of Electrochemistry*, vol. 2011, pp. 1–2, 2011.

[60] S. Carrara, *Bio/CMOS Interfaces and Co-Design*. Springer, 2012.

[61] S. L. Ntella, F. Stradolini, A. Tuoheti, D. Demarchi, A. A. Hatzopoulos, and S. Carrara, "Architecture and Procedures for pH and Temperature Monitoring in Medical Applications," in *IEEE Sensors Conference*, pp. 7–9, 2017.

[62] J. F. Kurose and K. W. . Ross, *Computer networking: a top-down approach*. Pearson, 6th ed., 2012.

Appendix A

AnControl Codes

The git repository containing the AnControl project's codes is available at the following url: https://github.com/NadiaTam/AnControl.git