











## Article

# RAMi: A New Real-Time Internet of Medical Things Architecture for Elderly Patient Monitoring

Olivier Debauche <sup>1,2,†,\*</sup> , Jean Bertin Nkamla Penka <sup>1,†</sup> , Saïd Mahmoudi <sup>1</sup> , Xavier Lessage <sup>1,3</sup> , Moad Hani <sup>1</sup> , Pierre Manneback <sup>1</sup> , Uriel Kanku Lufuluabu <sup>1</sup> , Nicolas Bert <sup>4</sup> , Dounia Messaoudi <sup>5</sup>  and Adriano Guttadauria <sup>1</sup> 

<sup>1</sup> Faculty of Engineering—ILIA/Infotech, University of Mons, 9 rue de Houdain, 7000 Mons, Belgium

<sup>2</sup> GxABT—BioDynE—DEAL/TERRA, University of Liège, 4000 Liège, Belgium

<sup>3</sup> CETIC, Applied Research Center, 6041 Charleroi, Belgium

<sup>4</sup> Ecole Centrale Marseille, 13451 Marseille, France

<sup>5</sup> Polytech Paris-Saclay, 91405 Orsay, France

\* Correspondence: olivier.debauche@umons.ac.be; Tel.: +32-65-374-059

† These authors contributed equally to this work.

**Abstract:** The aging of the world's population, the willingness of elderly to remain independent, and the recent COVID-19 pandemic have demonstrated the urgent need for home-based diagnostic and patient monitoring systems to reduce the financial and organizational burdens that impact healthcare organizations and professionals. The Internet of Medical Things (IoMT), i.e., all medical devices and applications that connect to health information systems through online computer networks. The IoMT is one of the domains of IoT where real-time processing of data and reliability are crucial. In this paper, we propose RAMi, which is a *Real-Time Architecture for the Monitoring* of elderly patients thanks to the Internet of Medical Things. This new architecture includes a Things layer where data are retrieved from sensors or smartphone, a Fog layer built on a smart gateway, Mobile Edge Computing (MEC), a cloud component, blockchain, and Artificial Intelligence (AI) to address the specific problems of IoMT. Data are processed at Fog level, MEC or cloud in function of the workload, resource requirements, and the level of confidentiality. A local blockchain allows workload orchestration between Fog, MEC, and Cloud while a global blockchain secures exchanges and data sharing by means of smart contracts. Our architecture allows to follow elderly persons and patients during and after their hospitalization. In addition, our architecture allows the use of federated learning to train AI algorithms while respecting privacy and data confidentiality. AI is also used to detect patterns of intrusion.

**Keywords:** real-time architecture; internet of things; internet of medical things; healthcare internet of things; edge AI; edge computing; data; apache; real-time; blockchain



**Citation:** Debauche, O.; Nkamla Penka, J.B.; Mahmoudi, S.; Lessage, X.; Hani, M.; Manneback, P.; Lufuluabu, U.K.; Bert, N.; Messaoudi, D.; Guttadauria, A. RAMi: A New Real-Time Internet of Medical Things Architecture for Elderly Patient Monitoring. *Information* **2022**, *13*, 423. <https://doi.org/10.3390/info13090423>

Academic Editor: Diogo Mattos

Received: 2 July 2022

Accepted: 29 August 2022

Published: 7 September 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

With the increase in the world's population and its aging, the traditional medical resource has reached its limits. Indeed, it is becoming difficult to get medical appointments, the price of consultations and treatments are expensive. In parallel, the development of the Healthcare Internet of Things (HIoT) allowed to propose a solution to achieve remote diagnostics, develop remote healthcare consultation, follow chronically ill and elderly patients. These new services are based on medical sensors, RFID tags, wireless networks, and applications used to connect medical sensors [1]. As cited by Kashani et al. [2], Health IoT systems (HIoT), in the face of a growing older population (affected by chronic diseases) and lack of access to medical resources, are a very promising solution. With the potential to provide health services to improve quality of life while reducing pressure on health systems. Research and experiments, such as the ones we propose, deserve to be evaluated on a real test bench.

Moreover, in [2], Girardi et al. mentioned that an inadequate or reduced data collection and low communication rates lead to diagnostic errors that impact the quality of healthcare [2].

Nowadays, medical sensors are widely used in operation and emergency rooms, and Intensive Care Unit (ICU) providing real-time monitoring of patients. While wearable sensors allow to reduce the treatment cost and personalized medical services thanks to remote consultation. With the deployment of the 5G, new applications will appear such as operation guidance, first aid vehicle diagnostic, and so on [3].

All these applications need to be collected, data need to be processed in real-time to analyze them and make a decision while respecting the sensitivity of the data and the privacy of the patients. Elsewhere, the global context shows that security is more than ever a point of attention at the level of processing architectures, especially for those dealing with sensitive information. Indeed, the data availability is crucial. Architectures must also guarantee the integrity and the confidentiality of data while remaining resilient to distributed attacks such as Distributed Denial of Service (DDOS), Man-in-the-middle (MITM), Botnets [4], which are groups of computers or devices under the control of an attacker that act to conduct malicious activities against a targeted victim via the network. By using distributed architectures, data are replicated and client-server architectures are no longer sensitive to this type of attack. These facts lead us to rethink the way we design architectures and secure data for critical infrastructures. Recent technological advances such as blockchain [5] and, more specifically, smart contract and confidential computing, allow us to reduce drastically the attack surface. The major contributions of this paper are as follows:

- We conceptualize and implement our IoT and AI architecture proposal for IoMT and we demonstrate its performance under real operating conditions;
- We experiment with the abnormal ECG detection-based on Machine Learning and annotation service in order to eliminate a part of false positive alerts.

The remainder of this paper is organized as follows. In Section 2, we present a brief overview of the state of the art in terms of real-time architecture in Internet of Medical Things (IoMT) in the form of a classification and we provide a comparison of pros and cons of all these architectures. In the Section 3, we present our original architecture. In Section 4, we evaluate performance of our architecture on real use cases. In Section 5, results are presented and afterwards discussed in Section 6. Finally, we conclude, and we draw outlines of our future works.

## 2. Literature Review

In this section, we consider architectures used to process IoT in real-time. Afterwards, we identified those that are specifically developed to address requirements of the Medical Internet of Things. We identified some existing elderly monitoring systems and, finally, we explained recent technological advances.

### 2.1. General Purpose Real-Time Architectures

The real-time processing of data is crucial in IoMT because human lives are at stake. To achieve this type of treatment, particular architectures must be put in place [6]. In this section, we investigate real-time architectures that have not been designed specifically for IoMT but that could be used in this domain.

Gaur et al. [7] defined a Smart City Architecture model for IoT. This architecture is a multi-level architecture. The level 1 is the data collection where raw data are collected from sensors and stored for further processing. The level 2 is data processing where raw data are transformed into a common format for processing. The level 3 is data integration and reasoning where data are classified and can be enriched with domain experts and uncertain reasoning. The level 4 is device control and alerts where processed data from level 3 can be used for customized services. The Smart City Architecture components

could be smart health, smart environment, smart energy, smart security, smart office and residential buildings, smart administration, smart industries, and smart transport [7].

Loria et al. [8] presented an in-house IoT architecture for a product named SeeYourBox. This architecture is built on two parts: processing and storing. To achieve that goal, the architecture is composed by three components: the gateway which received raw data from devices, the engine which analyzed the data and the databases to store the data. This architecture was designed to be very efficient regarding the SeeYourBox business and the expertise of their team regarding the technologies in IoT compared to the existing solutions such as Amazon AWS IoT or Microsoft Azure IoT [8].

Duan et al. [9] proposed an IoT architecture based on Quality of Service (QoS). Their architecture is based on the traditional IoT architecture of three layers: the perception layer, the network layer, and the third layer is a mix with the application and the service layer. The idea of this architecture is to ensure that the requirements of the end user of the IoT's application will become the QoS requirement of the application and the service layer. Then, this QoS requirement will be sent to the perception layer through the network layer. To achieve this, they also defined a QoS Management Facility for the three layers and two brokers: one in the network layer and the other in the perception layer [9].

The Table 1 gives a pros and cons analysis of general purposed real-time architectures selected.

**Table 1.** Comparison of general purpose real-time architectures.

Author	Pros	Cons	Year	Ref.
Duan et al.	This architecture focus on the improvement of the QoS of the IoT architecture.	The architecture is only theoretical.	2011	[9]
Gaur et al.	This architecture gives an overview of the Smart City architecture model.	It is only a general overview of the architecture for Smart City.	2015	[7]
Loria et al.	This architecture is an alternative to Amazon AWS IoT or Microsoft Azure IoT.	The architecture has been designed for one specific use case.	2017	[8]
Ta-Shma et al.	This architecture is build to deliver real-time decisions based on the mix of the knowledge of historical and new data from IoT streams.	The need of historical data could be a bottleneck for some use cases for real-time monitoring like health.	2018	[10]
Debauche et al.	This architecture is based on edge-cloud to deploy algorithms of AI and microservices.	This architecture was developed especially to run on constrained devices based on k8s.	2020	[11]

Ta-Shma et al. [10] proposed the *Hut Architecture* coupling historical data analytics and real-time processing in order to enlighten the real-time decision making on the basis of knowledge extracted from historical data and new data from IoT streams. This architecture is designed for per-event decisions or respond to events as they arrive by automated detection complex events in near real-time [10].

In 2020, Debauche et al. [11] described an architecture edge-cloud to deploy algorithms of AI and micro-services specifically developed to run on constrained devices using k8s [11].

## 2.2. Real-Time Architectures for IoMT

With the rise of 5G, the diversification of data sources has increased and the velocity of medical big data has also speed up [3].

In 2021, Delsate et al. [12] described The Institute of Analytics for Health (INAH), a platform for statistical and medical research which ensures that medical data are used in an ethical and secure way. Indeed, each data provider remains sovereign of its own data because a double pseudonymization prevents any possibility of patient identification [12].

In 2019, Debauche et al. [13] developed a three-layer architecture (Sensors, Fog, Cloud) for the monitoring of elderly and patients [13]. The Fog Layer is built around the association of Influxdb, Chronograf, and Kapacitor while a webservice allows the communication with the cloud layer. Data transmitted to Apache Kafka are temporarily stored in it before being ingested by the Druid Indexing Service. Apache Druid stores the data in the form of segments, in other terms, in sets of a few millions of rows of data stored on Hadoop File System (HDFS). Apache Ambari retrieved the Druid metric emitted to monitor it. Apache Kylin is used on Druid [13]. Nevertheless, with recent the versions of Apache Druid, it is more efficient to use Druid without Kylin or in alternative Kylin 4.0 with Parquet.

Sun et al. [3] reviewed the architecture of IoMT, the cloud-enabled IoMT, and the architecture of edge cloud IoMT and showed the potential of edge-cloud computing in the medical field. They argued that Edge Cloud computing is well adapted to 5G but brings a lot of security problems, threats, and privacy issues. The latter will be necessary for the edge level: the extension of hash and encryption algorithms, the mutual authentication of nodes, the development of communication security protocol, the thrust management of nodes, and distributed intrusion detection. In addition, a series of challenges still needs to be addressed such as cache strategy and cache update strategy, AI algorithms optimization at the edge level [3].

Nguyen et al. [14] proposed a new decentralized architecture for a cooperative hospital network using Blockchain called BEdgeHealth. This decentralized health architecture integrates a data offloading scheme and a data sharing scheme for distributed hospital networks with Mobile Edge Computing and blockchain [14]. The aim was to reduce the data latency retrieval and the data sharing overhead known with the existing sharing system.

Razdan et al. [15] proposed an overview of emerging technologies in IoMT, mainly with a three-layer-based architecture: the Things layer, the Fog layer, and the Cloud layer. The Things layer consists of sensors, actuators, medical records, etc. The goal of this layer is to collect all the available data for a further treatment. The Fog layer is composed by local servers and gateway devices which are used for security and data integrity purposes. The Cloud layer is used for data storage and computation resources of the data. Through some case studies, they presented the possibility to use these technologies in IoMT. They used the Physically Unclonable Function (PUF) to authenticate devices to the network. They used the Software-Defined Networking (SDN) to connect the devices with no internet connectivity. Finally, they used the Blockchain to ensure data security and privacy [15].

Boutros-Saikali et al. [16] proposed Open mHealth, a platform built on the top of scriptr.io, providing multiple connectors allowing interaction with the API of healthcare device. The platform automatically converts data obtained from devices in Open mHealth or FHIR observations. The security is inherently managed by scriptr.io through fine-grained access control rules applied to operations and stored data [16].

Girardi et al. suggested to associate Smart Contract with a datalake. Smart contract is organized in three parts. The first one “Contract Registrar” contains an user ID and identity on the blockchain. The second: “Summary Contract allows patients to ensure the traceability of their medical records and, finally, the third “Patient-Provider Relationship Contract” manages the relationship between patients and healthcare provider. The data are stored outside the chain in a datalake. They argue that this combination allows to store a wide variety of data [17].

Papaioannou et al. [18] presented a categorization of the potential threats and a categorization of security countermeasures for IoMT. According to [18], the security objectives

in IoMT are: confidentiality, integrity, non-repudiation, authentication, authorization, and availability of data.

It should be noted, as mentioned by Zhang et al. [19], that with the generalization of the Internet of Things (IoT), including smart clothes, there is not yet a unified architecture capable of connecting all the smart objects in smart hospitals, except with Nb-IoT, which can be a serious alternative by allowing objects to connect to the Internet by connecting directly to the operators' base stations with a very narrow bandwidth of 200 kHz. This approach introduces edge computing, and Zhang et al. developed an infusion-monitoring system to monitor in real time the rate of drop and volume of drugs remaining during intravenous infusion [19].

IPv6 applications in IoT are the trend at the moment with the Community Medical Internet of Things (CMIoT), but as Lui et al. [20] mentioned, the interconnection of the different CMIoT components is the main problem to be solved. There are many key issues still need to be addressed, such as how to handle IoT-oriented IPv6 network routing.

Together with his team, they proposed to design a simplified protocol message format assuming that the functional requirements are met. The interconnection between the IPv6 network and the physical network can then be achieved more efficiently [20].

Ed-daoudy et al. [21] proposed an architecture for real-time health status prediction and analytics system using big data technologies. This architecture applies distributed machine learning model on streaming health data events ingested to Spark, streaming through Kafka topics. According to the author, their architecture can predict health status, send an alert message to care providers, and store the details in a distributed database, to perform health data analytics and stream reporting [21].

Yacchirema et al. [22] implemented an architecture which has two goals: detect and support of treatment of Obtrusive Sleep Apnea (OSA) of elderly people. Indeed, the OSA is an important sleep disorder which directly impacts the quality of life. In order to detect the OSA, they used a FOG Computing method implemented in a smart device located at the edge of the network. The second objective is achieved by batch data which enable a descriptive analysis that statistically details the behavior of the data and a predictive analysis for the development of different services. This architecture used Big Data Tools on Cloud Computing [22].

The Table 2 gives a pros and cons analysis of real-time architectures for IoMT selected.

**Table 2.** Comparison of real-time architectures for IoMT.

Authors	Pros	Cons	Year	Ref.
Boutros-Saikali et al.	They proposed a platform build on another platform scriptr.io which provides different connectors to use with the API of healthcare devices.	Their platform and security depend entirely on the scriptr.io platform which could be a weakness in securing people's data.	2018	[16]
Zhang et al.	They proposed a real-time edge computing architecture for an infusion monitoring system.	This architecture has been designed for a specific use case and they did not give a general architecture which can be applied to another use case.	2018	[19]
Lui et al.	They proposed to design a new simplified protocol message to solve the connection's issue between the IPv6 network and the physical network	This protocol is not recognized as a standard to use to solve this issue.	2018	[20]
Debauche et al.	This architecture is built based on different open source components.	Today, some components used are not necessary anymore (e.g., Apache Druid with Kylin)	2019	[13]
Ed-daoudy et al.	This architecture used distributed machine learning (ML) model on streaming health data events.	The predictions depend on the accuracy resulting of the training of the ML model.	2019	[21]
Yacchirema et al.	Their architecture used Big Data Tools on Cloud Computing to detect and treat OSA.	This architecture has been built specifically for OSA problem related to the elderly people.	2019	[22]
Sun et al.	They reviewed different IoMT architectures, and they argued that Edge Cloud Computing is well adapted to 5G.	There are a lot of security problems, threats, and privacy issues with that solution.	2020	[3]
Girardi et al.	Their architecture proposal is to associate a Smart Contract with a datalake.	The contract seems to be very complicated (three parts) to understand and to implement.	2020	[17]
Papaioannou et al.	They presented the security objectives in IoMT.	They did not present a real use case with at least one security objective and the possible solution.	2020	[18]
Nguyen et al.	This architecture uses the Blockchain component.	This architecture is specific to a hospital which wanted to share their data securely and by reducing latency.	2021	[14]
Razdan et al.	They proposed an overview of emerging technologies in IoMT.	They do not apply them to a real use case.	2021	[15]
Delsate et al.	Platform that centralizes data from multiple medical institutions for scientific research while preserving patient privacy through double pseudonymization.	Adoption remains dependent on acceptance by medical institutions and ethics committees.	2021	[7]



### 2.3. Elderly Monitoring Systems

Jangra et al. [23], in 2018, proposed a design of a real-time multilayered smart healthcare monitoring framework based on IoT. Their proposal is composed of a Personal Body Area Network (PBAN) composed by various type of sensors (temperature, pulse oximetry, smart watch, blood pressure, etc). Then, after some transformations, the data are sent to the cloud to be monitored, by a gateway which could be a smartphone. As a result of this architecture, they proposed a simulation in LABVIEW software of the body temperature acquired by a temperature sensor. The temperature range is between 36.3 °C and 37.0 °C [23].

Islam et al. [24] developed a smart healthcare monitoring system based on IoT. Their system is composed of three modules: the sensor module, the data processing module, and the web user interface. The data processing module is composed of ESP32 processor, which is the heart of their system. The ESP32 processor offers a full Linux system on a small platform for a very low price. Their results for the heart rate were in a range between 68–83 bpm, for the body temperature, in a range between 36.1 °C and 37.0 °C, and for the room humidity, in a range of 60% to 72% [24].

Ramírez López et al. [25] proposed an IoT architecture in healthcare monitoring to enhance acquisition performance of respiratory disorder sensors. Their architecture is composed by three modules: the Wireless Personal Area Network (WPAN) acquisition module composed by a body temperature sensor and a CMS50DL Pulse Oximeter. The transmission module is composed by Arduino UNO and Raspberry Pi Model 3B (RPI3B) programmable cards. The Hub-IoT (NoSQL DB, cloud broker) is used to store and to process data. As a result, they obtained for spO<sub>2</sub>, a range between 95% and 97%, for the heart rate, a range between 86–100 bpm, and for the body temperature, a range of 35.5 °C to 37.0 °C [25].

Neyja et al. [26] presented an IoT based e-health monitoring system which uses the ECG signal. This architecture is formed by heterogeneous devices (sensors, user equipment, etc.), gateway (smartphone), and the hospital database. They have also developed a hospital alert system which is supposed to generate an alert in case of detection of an abnormal heart rate activity. To test their architecture, they used the Physio Bank ATM toolbox to generate the ECG [26].

Ruman et al. [27] proposed an emergency health monitoring system based on IoT. This architecture is composed by sensors (body temperature sensor, heartbeat sensor, etc), the data processing software (Arduino, ESP8266) and the cloud to store data. To visualize data, they used Thingspeak online software. As a result, they obtained a body temperature range between 35.6 °C and 37.8 °C. For the ECG, they obtained a range between 65–73 bpm [27].

The Table 3 gives a pros and cons analysis of elderly monitoring systems selected.

**Table 3.** Comparison of elderly monitoring systems.

Author	Pros	Cons	Year	Ref.
Neyja et al.	They developed a hospital alert when their system detects an abnormal heart rate activity.	Their tests used a simulation toolbox to generate ECG signal.	2017	[26]
Jangra et al.	The architecture proposed more computation on the PBAN side	As result, they have a simulation in the LABVIEW software.	2018	[23]
Ramírez López et al.	Their architecture gives the possibility to visualize historical data through their web application.	Their architecture depends mainly on Arduino UNO and Raspberry Pi.	2019	[25]
Islam et al.	The architecture does not required too much IoT tools.	Their architecture depends entirely on the ESP32 processor.	2020	[24]
Ruman et al.	Their system needs few IoT tools.	The architecture depends strongly on Arduino and ESP8266.	2020	[27]

#### 2.4. Recent Technological Advances

Rahimi et al. [28] presented a study for healthcare services in the cloud environment. They presented the challenges, needs, benefits on using the cloud for healthcare services, but also, the strengths and weaknesses of the existing methods. According to the authors of [28], many articles have tried to improve reliability, scalability, cost, and data access but they have not investigated safety and privacy aspects. For some articles, which treated these aspects, they assumed that various security algorithms can be adopted to secure healthcare data, such as cryptographic, biometric authentication, access control, secure socket layer, hashing, watermarking, and K-mean clustering [28].

The *blockchain* is an alternative to the singular failure point of the cloud Architecture [29].

In 2018, Ali et al. [29] conducted a survey on the various uses of blockchain in IoT. They recommended using permissionless blockchains which are public, anonymous accessible, and decentralized with Proof-of-stack or proof-of-X consensus algorithms. In addition, sharing mechanisms in Ethereum and Tendermint (<https://tendermint.com>, accessed on 1 July 2022) offers superior performances and scalability for IoT applications. Nevertheless, IoT devices have limited resources and, consequently, cannot host a copy of the blockchain and are not able to validate new blocks of the blockchain [29].

Pelekoudas-Oikonomou et al. [30] presented the fact that despite the existing blockchain applications for IoT to enable security, there are not really much existing blockchain-based security mechanisms specifically designed for IoMT edge networks. They summarized the current blockchain-based security for IoMT based on smart-contracts which leveraging the physical unclonable function (PUF) as an additional authentication factor. They also presented some related works not designed for IoMT but which could be adopted to it due to similar capabilities and technical characteristics [30].

In 2022, Ruggeri et al. [31] proposed BCB-FaaS, a solution based on smart contracts on Ethereum blockchain and Function-as-a-Service to allow the interoperability of on-demand services and secure decentralized communications. Indeed, blockchain allows to resolve security issues, privacy, and trust concerns while ensuring data confidentiality, integrity, and availability in multi-stakeholder application environments. The authors used Ethereum to have a large number of nodes to resist to distributed attacks but in case of the medical domain, the use of a private blockchain such as Hyperledger Fabric or Sawtooth is preferable. The processing time by the blockchain must be also considered in critical systems such as healthcare [31].

The security in architecture is achieved thanks to the encrypting of network transmission and stored data. However, when processed, the data appear in plain in the memory during the treatment. Today, the end-to-end security can be obtained with the *Confidential Computing* that uses hardware encryption capabilities of recent processors such Intel SGX, AMD Secure Encrypted Virtualization (SEV) or ARM Trust Zone which allow to process encrypted data in memory [32]. Trusted Execution Environments (TEEs) are the base of Confidential Computing [33]. Intel SGX is an extension of x86 architecture and a Software Development Kit (SDK). The Intel TEE allows execute the safe code in a protected region of the memory called enclave in order to protect code and data from disclosure or modification [34]. Intel SGX also provide a remote attestation mechanism, allowing to a remote application to verify the authenticity of an application thanks to the Intel Attestation Service (IAS) and the checking of the information received from the enclave. Both sides can then exchange sensitive and encrypted information thanks to the sharing of a symmetric key [34]. While the ARM TEE called TrustZone provides system-wide hardware isolation for trusted applications, in particular for IoT devices [34,35]. The secure zone runs a trusted OS providing isolation, integrity, system integrity, and running trusted applications [34]. The TEE helps protect against malicious applications, run the code in a secure world, and allow to isolate sensitive data with a low power consumption, and a low latency and low performance overhead [34]. Nevertheless, these environments remain vulnerable to side channel attacks of specific algorithms [33,36] or reverse-engineering attacks [33].



Valaderes et al. [33] described Confidential Computing technology and its implementation in the Edge-Fog-Cloud paradigm. The Edge level, on edge and IoT devices, ARM processors are mostly used. They implement the TrustZone technology which provides memory isolation, cryptographic acceleration, and trusted I/O. At the fog and cloud level, Intel SGX provides secure storage, memory isolation, cryptographic acceleration mechanism, and remote attestation procedure with third parties to create a trusted communication channel between the IoT/Edge and the Fog/Cloud servers [33].

Segarra et al. [35,37] proposed an adapted version of MQTT server Mosquitto, integrating TrustZone in order to improve the security in addition of the using of TLS or SSL. They implemented in MQT-TZ, a mutual handshaked TLS and an end-to-end two-layer encryption based on TEE, ACLs, and re-encryption of information in TrustZone with AES in CBC mode with 32-Byte keys [37]. The proposed modification to the Mosquitto MQTT server allows to prevent packet interception, man-in-the-middle attacks, unauthorized entities injection or subscription, and the spoofing of sensitive information [35].

Deep neural networks are a promising approach for the development of health support tools. However, the use of these particular models requires a large amount of training data to achieve critical performance. In addition, the data acquisition process is often problematic as it requires expert annotation and is subject to data protection legislation, which hinders the development of healthcare support tools. *Federated learning* is a solution to overcome this limited data availability while respecting data privacy. The Federated Learning generates a machine learning model trained with datasets distributed on multiple devices. The data and the model stay locally without any transmission of data, only the owner can obtain the trained model [36]. According to authors of [36], the computational formula of Federated Learning is

$$\text{Arg Min } L(a, b, c) = \sum P_k L_k(a, b, c) \quad (1)$$

where  $k$  = total number of clients;  $P_k$  = weight of the  $k^{\text{th}}$  client;  $a$  = input;  $b$  = output;  $c$  = parameter to be learned.

In our architecture, in order to ensure data confidentiality and privacy, medical data will therefore not be shared and will not leave their original location. Our deep learning algorithms will rely on federated learning instead of traditional learning. The main advantage for the members of the Coalition thus formed (partners) is that they do not exchange their local data. In exchange, the partners have to train a similar model on their local data. In particular, in federated learning, a consortium of actors shares their locally trained model weights and a central unit aggregates these.

Although it allows partners to share only the gradients and weights of their model, the architecture also raises several challenges to ensure privacy and confidentiality. Firstly, the pseudonymization of the training dataset and, secondly, the confidentiality of the models and gradients must be guaranteed in order to avoid any reverse engineering of the training dataset. Finally, the protection of the model against degradation by training on inadequate data.

There is a desire to solve the above challenges. Kaissis et al. [1] proposed a secure and privacy-preserving architecture. The latter uses federated learning to develop a global model resulting from the aggregation of weights transmitted by local hospitals.

However, given the privacy breaches of deep learning models, it is highly recommended to protect them from adversarial attacks, especially for applications using sensitive data. One of the solutions to this problem is to protect the model with homomorphic encryption, the particularity of which is that it is possible to perform operations directly on the encrypted data and, therefore, without having to decrypt them, as is usually the case. In our architecture, this encryption will be used specifically for weight sharing between neural networks.

### 3. RAMi Architecture

Our architecture proposition called RAMi is a **Real-Time Architecture** for the **Monitoring** of elderly patients thanks to the Internet of Medical Things. It contains an early warning system using the real-time data analysis using Machine Learning (ML) algorithms applied to real-time data and an annotation system to specifically tag individual patient’s alert on historical data to avoid false positive alerts.

Our architecture is based on open-source software to ensure its sustainability and the Independence of other platforms by a design which allows the replacement of a software brick which would disappear, or another software brick would be more efficient. Moreover, the fact of using software components with permissive open-source licenses makes it possible to value the developments made on the platform and even to close the code afterwards in view of a commercialization. In addition, changing software bricks allows the architecture to be adapted to the specific needs of particular use cases. The use of a datalake formed by an object storage, a service of indexing and discovery of the data allows to store large amounts of data and to exploit them on demand for energy saving purposes compared to a systematic treatment of the whole dataset before storage.

The architecture associates a cloud and a fog level of treatment to prevent and be independent of possible connection problems such as link congestion, temporary interruption of network connections that could be fatal for patients. In addition, false positives usually require human intervention, which is costly for health care facilities. The elimination of a part of them will allow to relieve the care structures by limiting the costs related to false alarms. We have chosen an architecture deployed in the form of containers to allow easier deployment, scaling, and management.

Figure 1 presents an overview of the main software bricks of the cloud part of RAMi architecture as well as the way data flow within the architecture. The data are transmitted to MQTT topic and then converted in message stored in the message queue. The real-time processing consumes and transforms the data train and predicts anomalies in data. Patterns detected are stored in the relational database in order to be annotated. The nursing staff can annotate each patient’s patterns into true and false positives using the eponymous service. All the data processed by the real-time processing are stored in the time series database. All data are stored in a deep storage Amazon S3 compatible. However, data from the relational database must be converted into objects beforehand. The conversion into an object makes it easier to search the data and display them in the dashboard. Finally, an in-memory cache system speeds up the most common queries on both the time series database and the dashboard system.

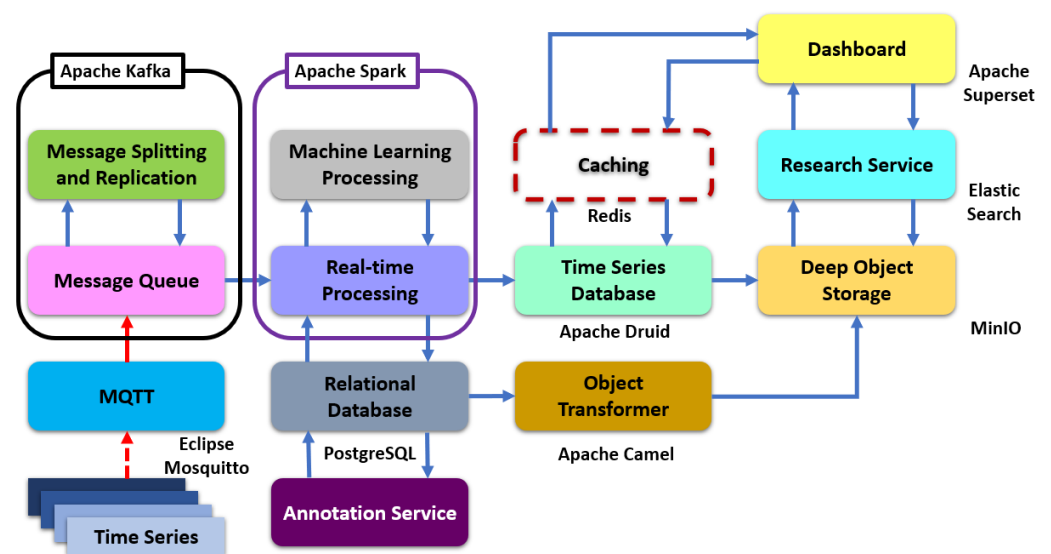
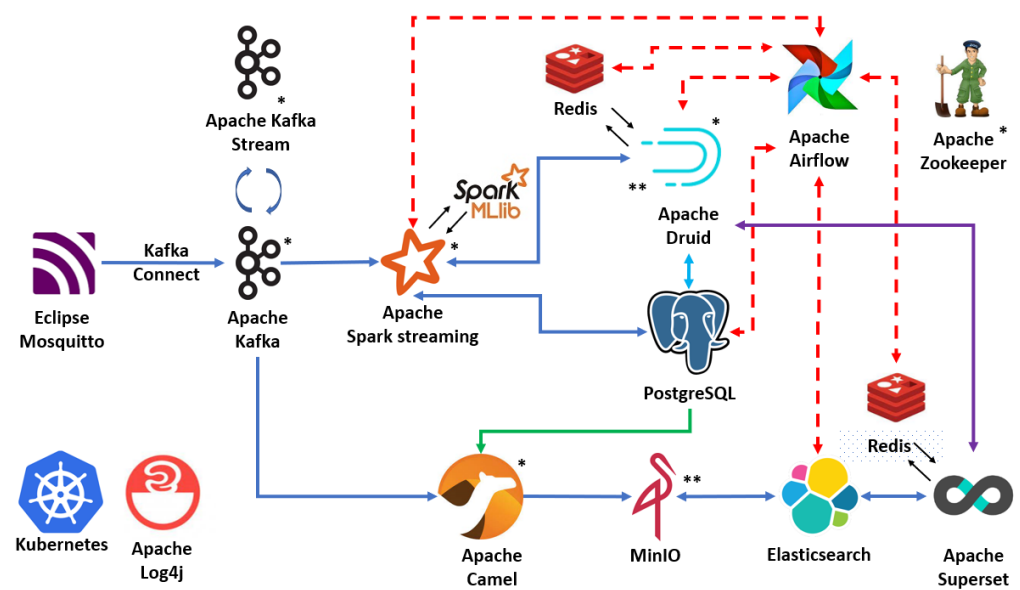


Figure 1. Conceptual diagram and principal software of the cloud part of the RAMi Architecture.

The cloud part of the architecture is built around ten software components (Mosquitto, Kafka, Spark, Camel, MinIO, ElasticSearch, Surperset, PostgreSQL, Druid, Redis, Zookeeper) which constitute the core of the architecture, a log system (Log4j), a monitoring system (AirFlow) and deployment, and an administration model (Kubernetes).

**Eclipse Mosquitto** ([mosquitto.org](https://mosquitto.org), accessed on 22 June 2022) is a MQTT server that collects data from sensors of origins and stores them in individual topics in binary, plain text or json form, **Apache Kafka** ([kafka.apache.org](https://kafka.apache.org), accessed on 24 June 2022), a message queue widely used to temporary store data before or after their processing. **Apache Spark Streaming** ([spark.apache.org](https://spark.apache.org), accessed on 28 June 2022) is implemented to process streams of data while the MLlib library achieves Machine Learning analysis. At the end, the data are stored in function of the type of data in a polystore composed of multiple databases (Apache Druid & PostgreSQL). **Apache Camel** ([camel.apache.org](https://camel.apache.org), accessed on 22 June 2022) is an integration framework that empowers integration of various systems consuming or producing data. **MinIO** ([min.io](https://min.io), accessed on 22 June 2022) is high-performance, S3 compatible object storage. **ElasticSearch** ([elastic.co](https://elastic.co), accessed on 24 June 2022) is a software using Apache Lucene ([lucene.apache.org](https://lucene.apache.org), accessed on 24 June 2022) for indexing and searching data. **Apache Airflow** ([airflow.apache.org](https://airflow.apache.org), accessed on 24 June 2022) is a platform to programmatically author, schedule, and monitor workflows. **Apache Superset** ([superset.apache.org](https://superset.apache.org), accessed on 25 June 2022) is a data exploration and visualization platform. **PostgreSQL** ([www.postgresql.org](https://www.postgresql.org), accessed on 28 June 2022) is a powerful, open-source object-relational database system. **Apache Druid** ([druid.apache.org](https://druid.apache.org), accessed on 27 June 2022) is a real-time database to power modern analytic applications. **Redis** ([redis.io](https://redis.io), accessed on 25 June 2022) is an open source, in-memory data store. **Apache Zookeeper** ([zookeeper.apache.org](https://zookeeper.apache.org), accessed on 25 June 2022) is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. **Kubernetes** ([kubernetes.io](https://kubernetes.io), accessed on 25 June 2022) is an open source system for automating the deployment, scaling, and management of components in a containerized architecture. **Apache Log4j** ([logging.apache.org/log4j/2.x](https://logging.apache.org/log4j/2.x), accessed on 25 June 2022) is a Java-based logging utility.

Figure 2 presents an overview of the architecture in which some links with Apache Zookeeper and MinIO are respectively replaced by single and double asterisks to improve readability. Zookeeper also ensures the replacement of failed instances of Apache Kafka and Apache Kafka stream, Apache Spark Stream, Apache Camel, Apache Druid. Airflow plays the role of monitoring of critical components of the architecture: Redis, Apache Spark Stream, PostgreSQL, Elasticsearch and Apache Druid. It is not used for the deployment of workflow in our architecture. Kubernetes is used for the deployments of architecture's components and ensures its scalability. While Log4j, a well-known library, is implemented to monitor incidents at key points in the architecture.



**Figure 2.** Cloud part of the RAMi Architecture.

Data produced by Medical Sensors or Medical Things are sent to MQTT topics. Kafka Connect interconnects MQTT with Apache Kafka and ensures one-to-one correspondence between an MQTT topic and an Apache Kafka topic. Following the nature of the sensors, data are transmitted individually on individual MQTT topics or on a common topic. In this last case, Apache Kafka Stream splits all kinds of data in separated thematic topics to facilitate their processing by Apache Spark later on. Apache Camel consumes raw data in Apache Kafka. Apache Camel transforms each one in an object and stores it on MinIO where it is directly available for ElasticSearch where it is indexed and available for search. MinIO plays at same time the role of Datalake. Apache Spark Streaming consumes data present in the Apache Kafka, annotates and enriches data while Apache Spark MLlib is used to clean and validate data before storing them. The time-series data are stored in Apache Druid in the form of groups of 4 to 7 millions of lines called segments stored onto MinIO which also plays the role of deep storage. While PostgreSQL is used to store, on the one hand, a relation between collected data and Metadata of segments is produced by Apache Druid. Apache Superset allows to analyze, on the one hand, raw data to obtain an overview of the evolution of the parameters and, on the other hand, query the Druid database-processed data refining trends and achieving more complex analyses.

Apache Airflow is used in our architecture to monitor workflows in Apache Druid, PostgreSQL, Redis, and ElasticSearch. Zookeeper coordinates Druid, Spark, and Kafka clusters. While Log4j allows to log all events at key components of the architecture to quickly detect anomalies and investigate failures. Redis plays the role of caching system for Apache Druid at broker level and for the Apache Superset. Finally, Kubernetes deploys easily all components of the architecture and news pods to scale of the architecture.

The proposed architecture is sufficiently flexible to easily replace most components with alternatives. In order to enlighten the readers on the possibilities of substitution, we propose some alternative software bricks that could be used to replace the components of the architecture. An alternative to Apache Druid, Apache Pinot ([pinot.apache.org](https://pinot.apache.org), accessed on 30 June 2022), a promising Real-time distributed OLAP datastore, designed to answer OLAP queries with low latency can be implemented. While Elasticserach can be replaced by Apache Solr ([solr.apache.org](https://solr.apache.org), accessed on 30 June 2022), a popular, blazing-fast, open-source enterprise search platform also built on Apache Lucene ([lucene.apache.org](https://lucene.apache.org), accessed on 29 June 2022). MinIO can be replaced by Apache Cassandra ([cassandra.apache.org](https://cassandra.apache.org), accessed on 30 June 2022) that is also compatible with Apache Druid in terms of deep storage.

Figure 3 presents a conceptual overview of the fog part of the architecture RAMi. Data are sent by sensors to MQTT server with TLS protocol where they are processed and temporarily stored locally and then transferred to the MQTT server in the form of timeseries to the MQTT server of the cloud architecture. The local storage allows to avoid loss of crucial data in case of temporary congestion or interruption of the network.

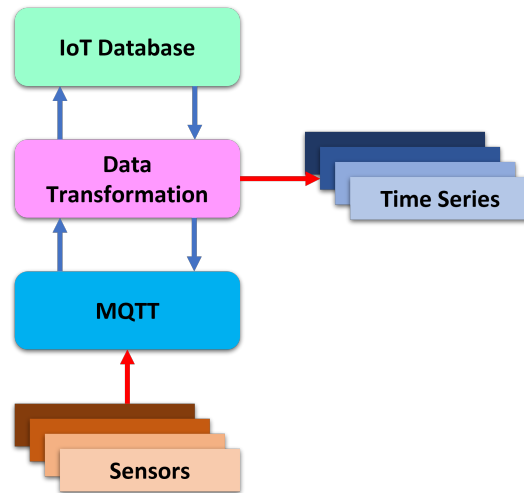


Figure 3. Conceptual diagram and principal software of the Fog part of the RAMi Architecture.

Figure 4 presents the architecture deployed on the gateway to clean, validate, and enrich data received from sensors by the Mosquitto MQTT Server.

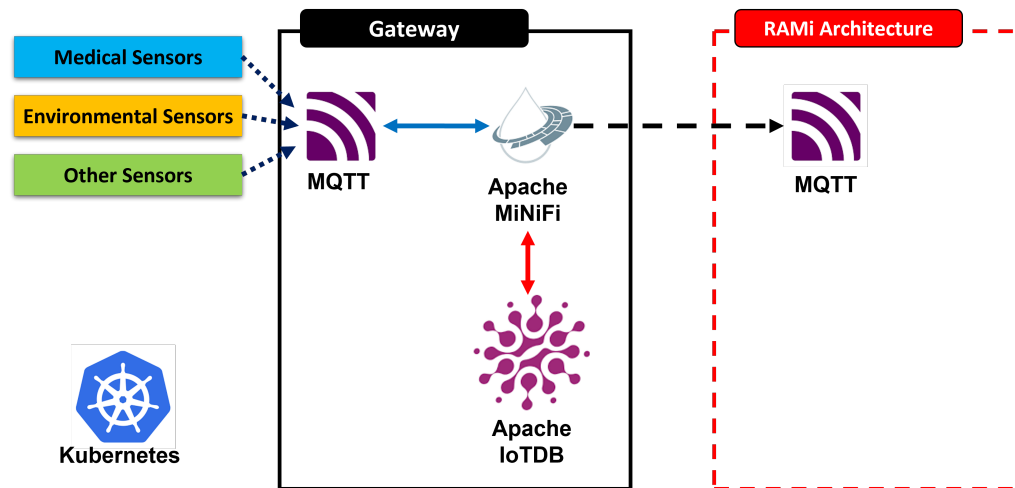


Figure 4. Fog part of the RAMi Architecture.

At the gateway level, devices send their data to the MQTT server of the gateway where the data are handled by Apache MiNiFi (a light version of Apache NiFi ([nifi.apache.org](http://nifi.apache.org), accessed on 1 July 2022) dedicated to constrained devices). We implemented Apache IoTDB to store temporary and local data.

A responsive mobile application was developed in Flutter/Dart to visualize data at the edge and cloud level. Figure 5 shows an overview of the interface data visualization of the developed application.

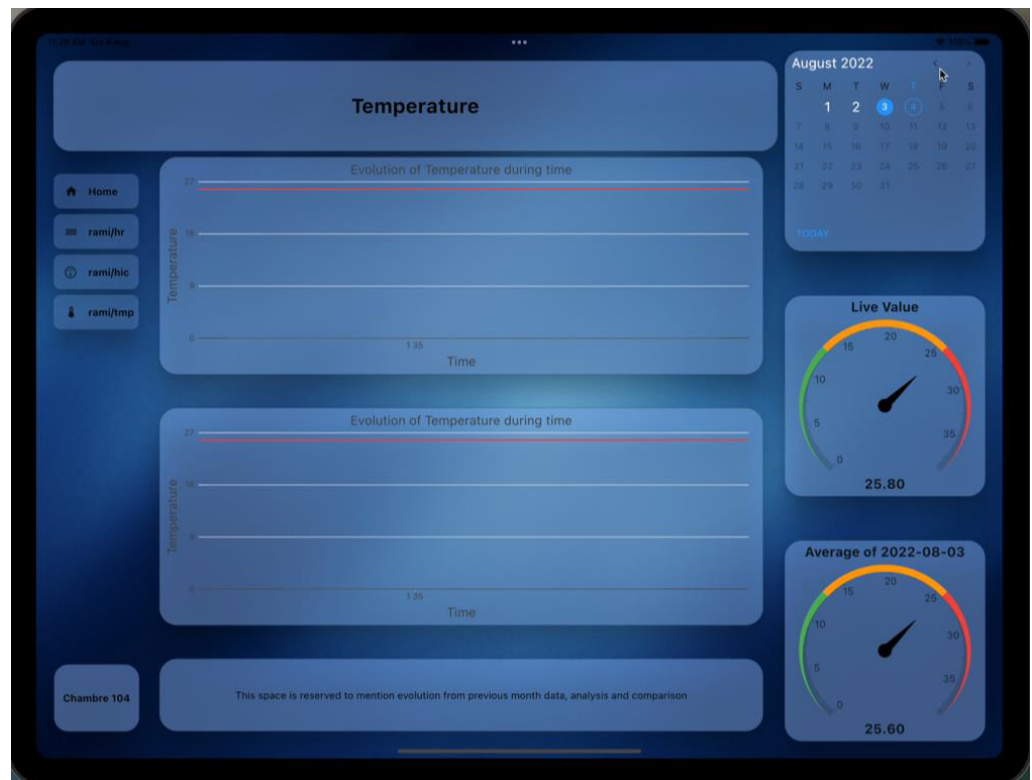


Figure 5. Mobile App to visualize data.

#### 4. Experimentation

The aim of our experimentation is to evaluate our architecture, on one the hand, on the basis of the ECG real-time abnormal detection and, on the other hand, on the quality of the patient environment thanks to the evaluation of Heat Index (HI). The two experiments are implemented on the same microcontroller equipped of four sensors. We used a Wemos D1 R32 board that has the same factor form as an Arduino UNO and is also programmable with Arduino IDE. On this board, the ATmega328 16 MHz is replaced by an ESP32 WROOM at 240 MHz with Wi-Fi, Bluetooth, and 4MB of flash memory.

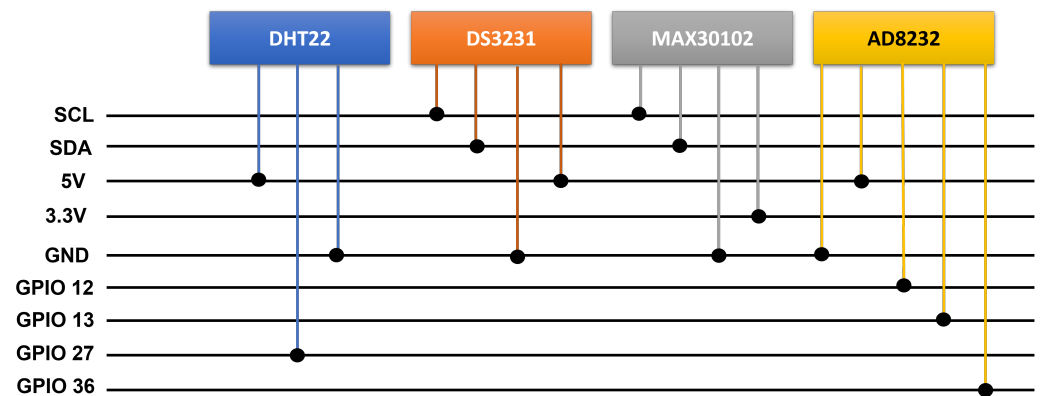
Both experiments use a common real time clock DS3231 connected to the microcontroller by the I<sup>2</sup>C protocol to retrieve a common base time which is the Linux time, i.e., the number of seconds elapsed since 1 January 1970. For the first experimentation, the Wemos UNO D1 R32 relates to an integrated signal conditioning block for ECG AD8232 and a High-Sensitivity Pulse Oximeter and Heart-Rate Sensor MAX30102 by I<sup>2</sup>C protocol. In the second experimentation, a DTH22 sensor measures ambient air temperature and relative humidity. The HI is calculated from ambient air temperature and relative humidity with the following equation:

$$HI = c_1 + c_2T + c_3R + c_4TR + c_5T^2 + c_6R^2 + c_7T^2R + c_8TR^2 + c_9T^2R^2 \quad (2)$$

where  $HI$  is the heat index ( $^{\circ}\text{C}$ );  $T$  is the temperature ( $^{\circ}\text{C}$ );  $R$  is the relative humidity (%);  $c_1 = -8.78469475556$ ;  $c_2 = 1.61139411$ ;  $c_3 = 2.33854883889$ ;  $c_4 = -0.14611605$ ;  $c_5 = -0.0123808094$ ;  $c_6 = -0.0164248277778$ ;  $c_7 = 2.211732 \times 10^{-3}$ ;  $c_8 = 7.2548 \times 10^{-4}$ ;  $c_9 = -3.582 \times 10^{-6}$ .

In the two experiments, the data are transmitted by Wi-Fi to the gateway with MQTT protocol with TLS. Figure 6 shows the connection diagram of the 4 sensors with pins of the microcontroller Wemos D1 R32.





**Figure 6.** Connection diagram of the sensors with the microcontroller.

The cloud architecture has been deployed on Contabo Cloud VPS XL (10 vCPU, 60 GB ram, 800 GB NVMe) with Ubuntu 20.04 LTS Server ([contabo.com/en/vps](https://contabo.com/en/vps), accessed on 1 July 2022).

The gateway is built around an Odroid M1-8GB powered by a Rockchip RK3568B2 containing a quad-core Cortex-A55, a Mali-G52 EE, and a 0.8 TOPS NPU. In addition, it is equipped with 8GB LPDDR4 RAM, a RJ45 Ethernet Port (10/100/1000), a M.2 NVMe M-Key PCIe3.0 2-Lane, a SATA3, and a eMMC Module Socket. An adapted version of Ubuntu 20.04 LTS with Kernel 4.19.219 and NPU support powered the Odroid M1.

We opted for the use of a medical node of our own making to facilitate the experimentation and to free us from the problems of data format, compatibility but also of proprietary transmission protocols. In the first experiment, we transmitted data continuously at high frequency (up to 120 Hz) to test the capability of the architecture to process data in real-time. In the second experiment, we evaluate the capability of the architecture to process several data flows in parallel. Temperature and relative humidity are transmitted at a frequency of 0.5 Hz while the data of heart rate and oxygen saturation are transmitted only when the patient's finger is in contact with the sensor in a data burst. These two experiments allow us to test the abilities of the architecture to process data in the three modes of transmission (continuous, at regular time intervals, and burst).

## 5. Results

This section presents results obtained at the end of the two experiments. These results are intended to demonstrate the ability of the architecture to operate in all three data transmission modes (continuous, at regular intervals, and burst mode).

### 5.1. Results of the First Experiment

In this first experiment, the ECG sensor is connected to the patient and sends data at high frequency (up to 120 Hz) continuously. The goals are to test the ability to process data retrieved at high frequency, on the one hand, and the effectiveness of the Machine Learning algorithm to detect anomalies on the other hand.

Figures 7 and 8 show ECG of patients detected by the Machine Learning (ML) as normal, while Figures 9 and 10 show ECG of patients suffering of cardiac diseases and detected abnormal by the ML algorithm.

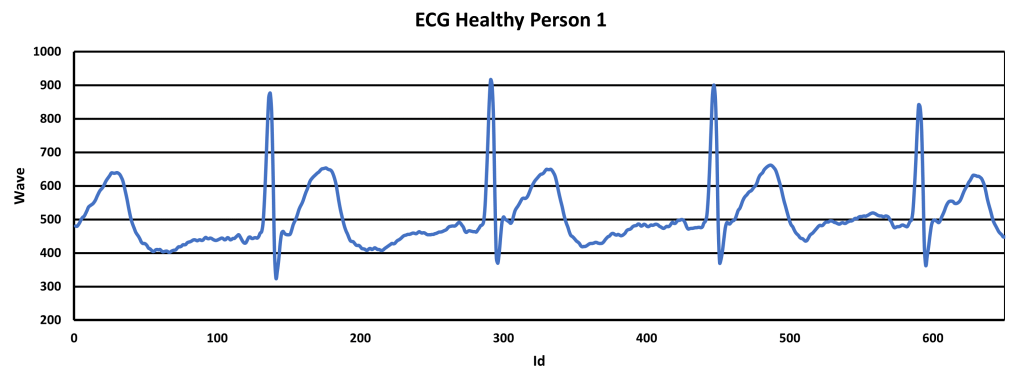


Figure 7. Evolution of values for ECG—Healthy person 1.

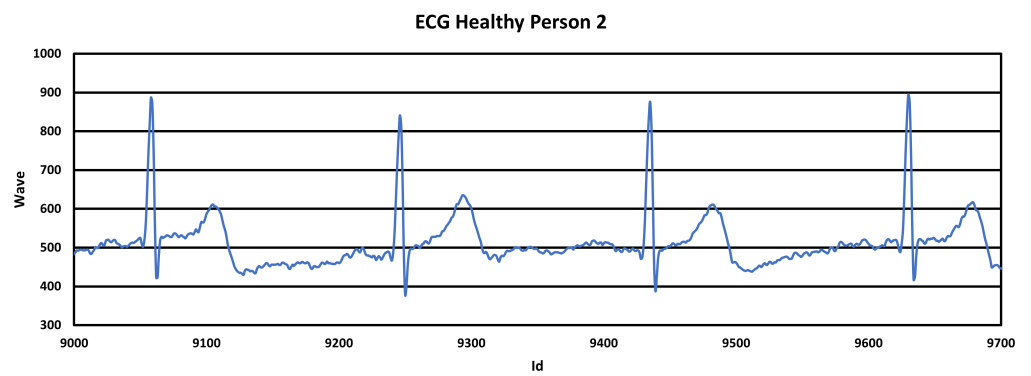


Figure 8. Evolution of values for ECG—Healthy person 2.

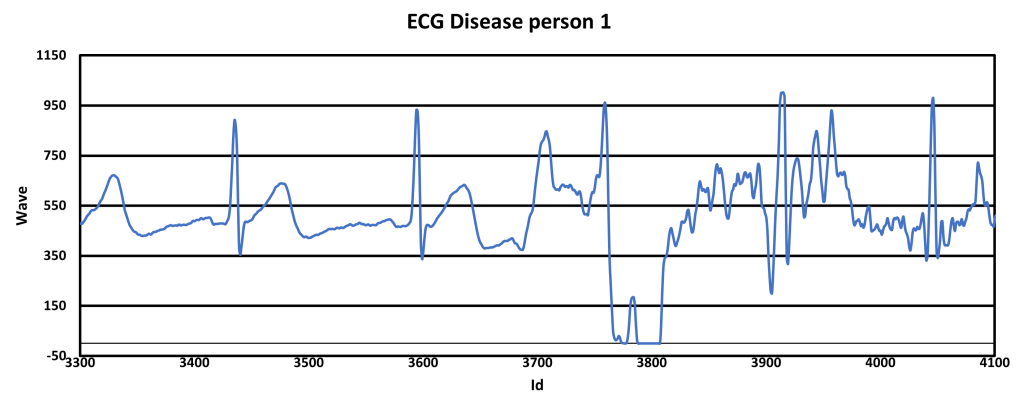


Figure 9. Evolution of values for ECG—Person with disease 1.

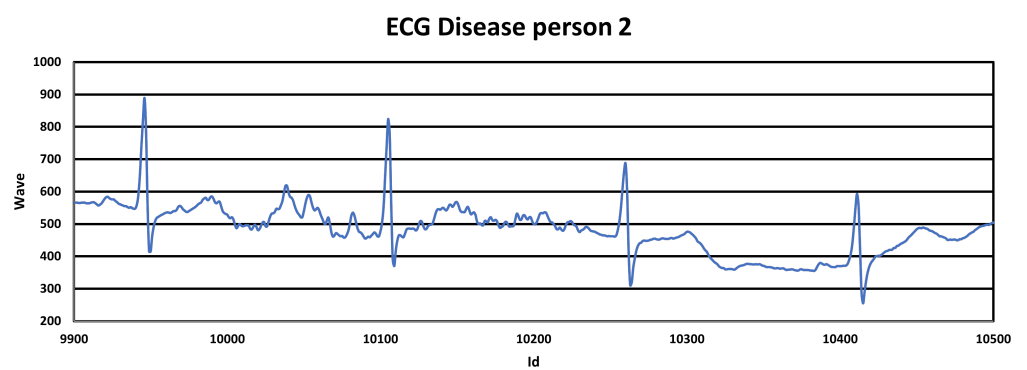


Figure 10. Evolution of values for ECG—Person with disease 2.

### 5.2. Results of the Second Experiment

The second experiment shows the functioning of the architecture for the calculation of the HI from the temperature and relative humidity data sent at regular intervals at the level of Apache Spark Streaming.

The following results are samples of data which have been continuously retrieved from the medical node described in the previous paragraph, during a period of twelve hours. Figures 11–13 show examples of results obtained after processing of temperature data. Figure 11 presents results of the average temperature. While Figure 12 describes the evolution of standard deviation of the mean temperature. Their values are between 0.046 °C and 0.1631 °C, which means that there is very little deviation around the mean value.

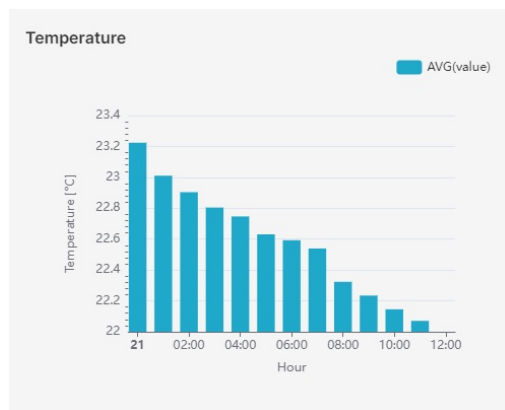


Figure 11. Average values for temperature.

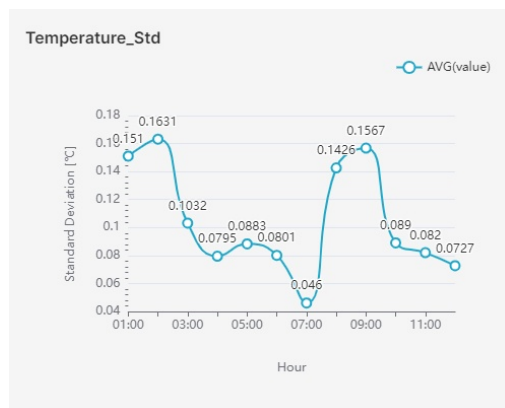
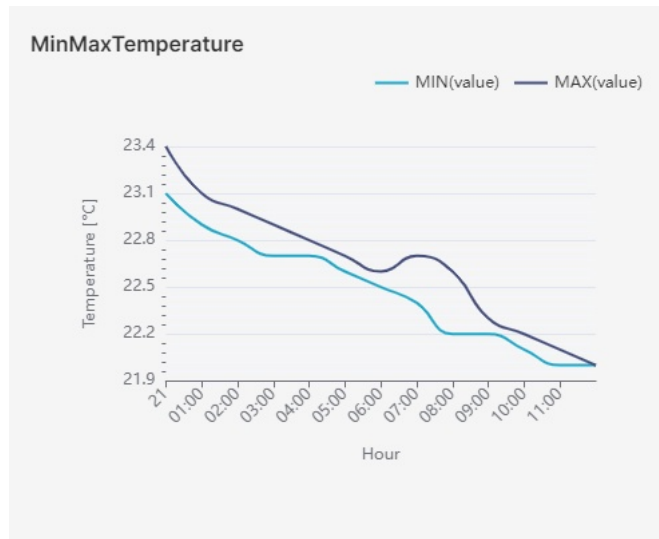


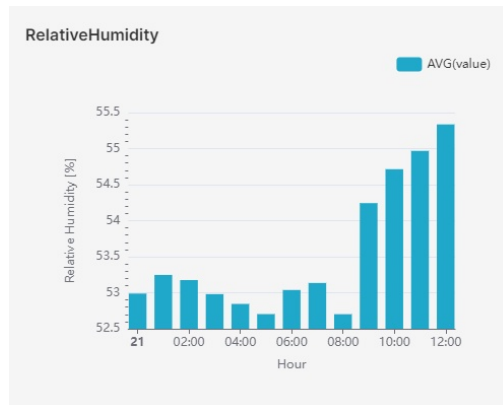
Figure 12. Standard Deviation on average values for temperature.

Figure 13 presents the evolution of minimum and maximum temperature. In this figure, the difference is less than one °C. The data covered only twelve hours, but we can conclude that the temperature that we received from our medical node is quite stable.

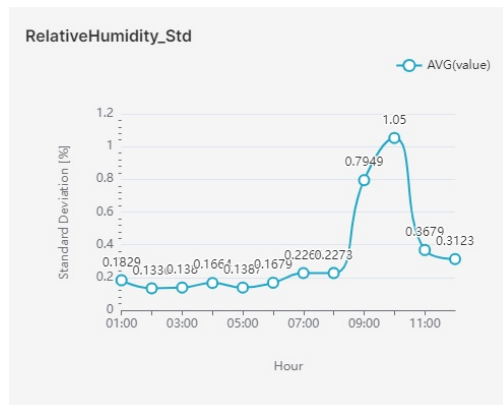


**Figure 13.** Evolution of Minimum—Maximum values of the temperature.

The results for the relative humidity are represented in Figures 14–16. Figure 14 represents the average values of relative humidity and Figure 15 represents the standard deviation of these average values. Their values vary between 0.1829% and 1.05%. In Figure 15, we can see that the relative humidity is quite stable from 1 h to 8 h. Then, we have a higher value. This is also visible in Figure 14 where the average value increases a little at the 8th hour.



**Figure 14.** Average values for relative humidity.



**Figure 15.** Standard Deviation of average values for relative humidity.

Figure 16 shows the evolution of minimum and maximum values of the relative humidity during the given time.

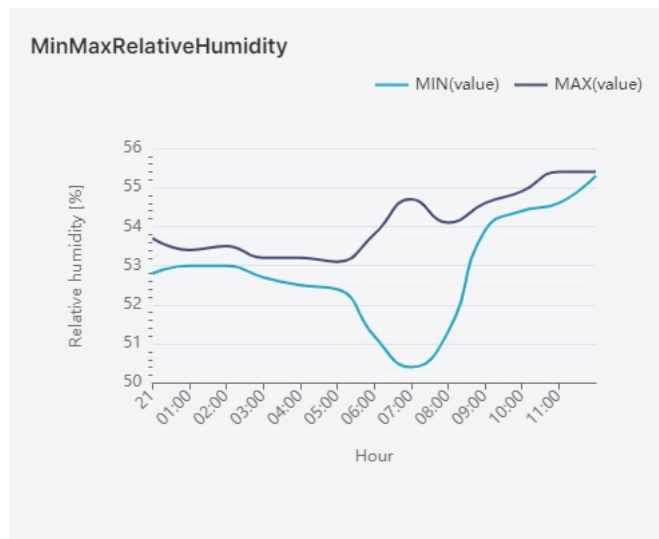


Figure 16. Minimum—Maximum values for relative humidity.

Figures 17–19 represent the result of the heat index. Figure 18 shows the variation of the head index around the average values represented in the Figure 17. We can see that the heat index is not so stable as the relative humidity. Figure 18 presents variations between 0.0461 °C and 0.1779 °C which still represent very small variations of the temperature.

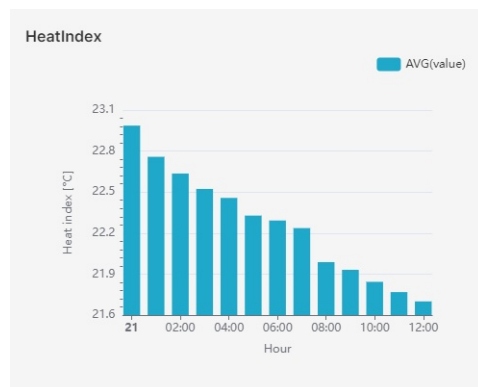


Figure 17. Average values for heat index.

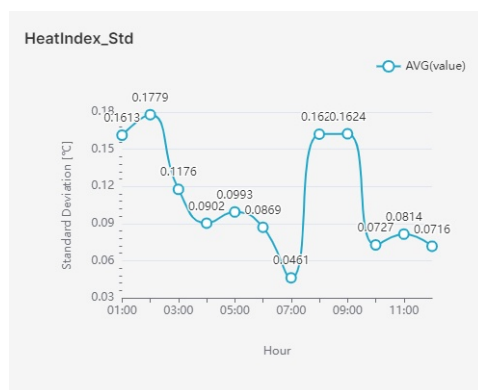
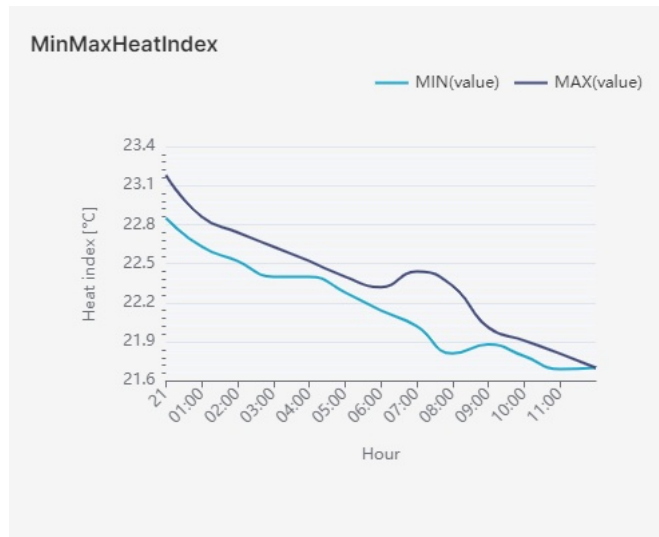


Figure 18. Standard Deviation of average values for heat index.

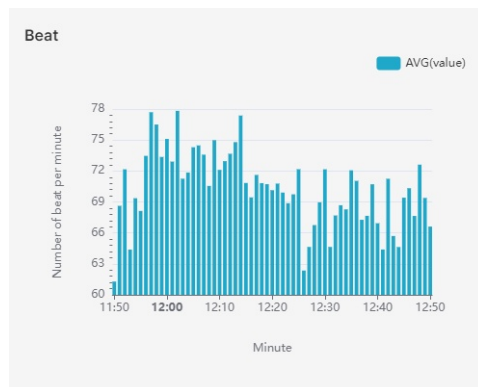
Figure 19 shows the evolution of minimum and maximum values of the heat index during the given time. This curve is very similar to that of Figure 13.



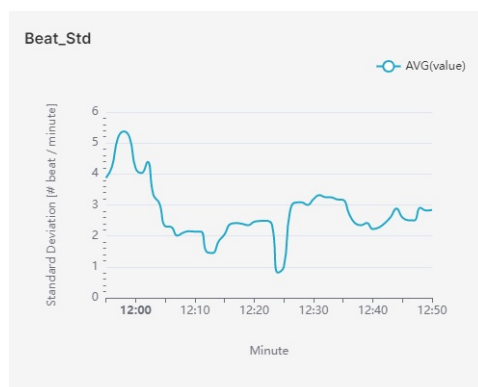
**Figure 19.** Evolution of Minimum and Maximum values of the heat index.

In a second experiment, we experimented with the burst mode thanks to the transmission of beat rate per minute and oxygen saturation of hemoglobin by pulse oximetry (spO2). The data are only transmitted when the patient puts his finger on the infrared sensor (AD8232).

Figures 20–22 are samples of the results of the number of beats per minute measured during one hour from our medical node. In Figure 21, we can see the standard deviation of the average beat values represented in Figure 20. Instead of the other parameters, the standard deviation on beat varies between 0.8139 and 5.37 beats per minute. This difference could be explained by the fact that the X-axis in Figure 21 is expressed in minutes instead of hours as the other graphics.



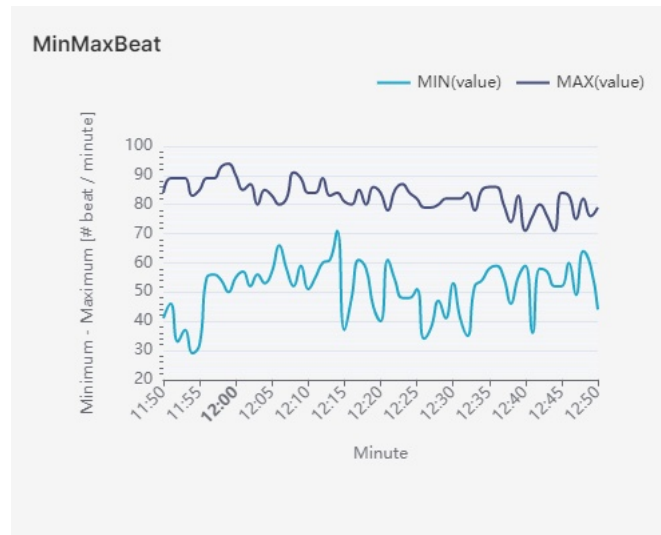
**Figure 20.** Average values for beat.



**Figure 21.** Standard deviation on average beat values.



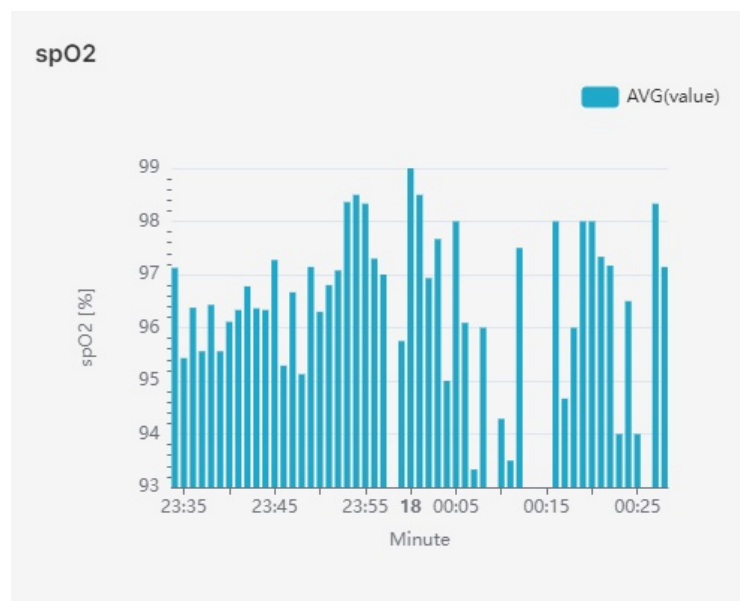
Figure 22 shows the evolution of minimum and maximum values of beat during the given time. Instead of the other minimum and maximum's graphics, we can clearly see the difference between the minimum and the maximum compared Figure 19, for example.



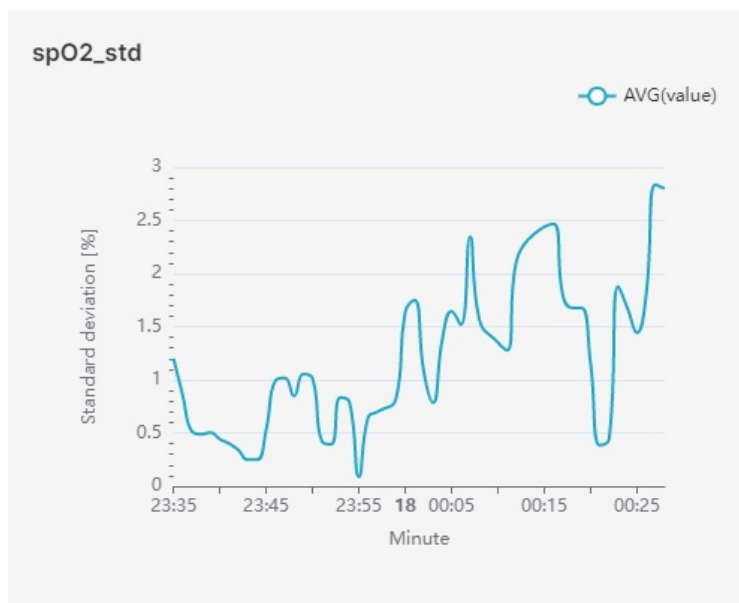
**Figure 22.** Evolution of minimum and maximum values of beats per minute.

This difference comes from the fact that for beats, we are observing data for one hour, while we are observing the minimum and maximum of the other parameters for a twelve-hour period.

Figures 23–25 are samples of the results spO2 rate expressed in % measured during one hour from our medical node. In Figure 24, we can see the standard deviation of the average spO2 values represented in Figure 23. Instead of the other parameters, the standard deviation on spO2 varies between 0.0888 and 2.83% per minute. This difference can be explained by the fact that the x-axis in Figure 23 is in minutes instead of hours as the other graphics.

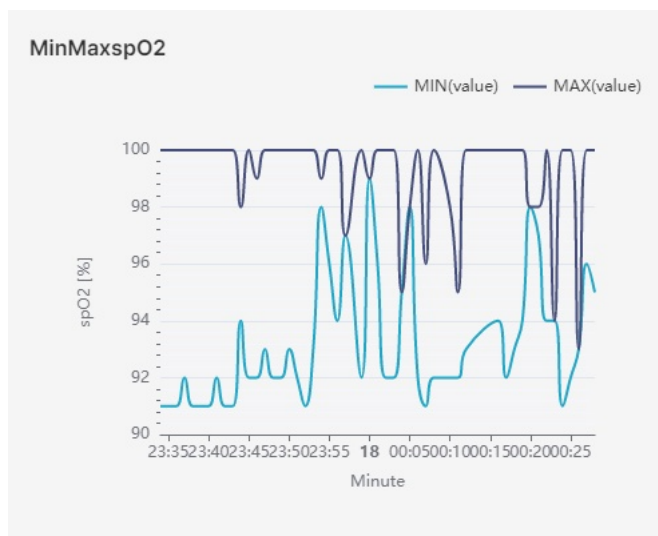


**Figure 23.** Average values for spO2 rate.



**Figure 24.** Standard deviation of average spO2 rate values.

Figure 25 shows the evolution of the minimum and maximum values of spO2 during the given time. Instead of the other minimum and maximum graphs, we can clearly see the difference between the extremes.



**Figure 25.** Evolution of Minimum and Maximum values for spO2 rate.

This difference indeed comes with the fact that for spO2, like heartbeat, we analyzed data for one hour while we observed the minimum and maximum for the other parameters over a period of twelve-hour period.

In the end, we tested our architecture with the data transmitted by our medical node. ECG Machine Learning algorithm performance was evaluated from data acquired on two healthy patients (Figures 7 and 8) and on two other patients suffering of cardiac diseases (Figures 9 and 10). The continuous transmission of data has demonstrated the ability of the architecture to collect, ingest, and store information at high speed and to detect certain cardiac anomalies in real time using Machine Learning. Environmental data (air temperature and relative humidity) were transmitted at a frequency of 1 Hz. The HI is calculated on the basis of temperature expressed in °C and relative humidity expressed in %. An extract of the analysis achieved with Apache Superset is shown in Figure 11 to Figure 13 for temperature, Figure 14 to Figure 16 for humidity, and Figure 17 to Figure 19

for heat index, respectively. Beat rate per minute and spO<sub>2</sub> rate acquired by MAX10302 and transmitted at each change of value at a variable frequency between 1 Hz to 3 Hz. One overview of these data is presented in Figure 20 to Figure 22 and Figure 23 to Figure 25.

These first results allow to demonstrate the correct functioning of our architecture. Nevertheless, the architecture must also be tested with commercial sensors to demonstrate its ability to work with a wide variety of sensors producing data at different frequencies and in various formats. In addition, the architecture must also be assessed with several sensors which send their data synchronously, to demonstrate its ability to manage and process data without loss of performance.

## 6. Discussion

In our experimentation, we have evaluated the ability of the architecture to ingest and process three ways of sending data. We used for our experiments a Wemos D1 R32 board equipped with 4 sensors (DHT22, DS3231, MAX30102, AD8232) which measures air temperature/relative humidity, date/time, heartbeat/spO<sub>2</sub>, and ECG, respectively.

The authors of [23] demonstrated that local processing of data from various sensors is necessary before sending them to the cloud. Humidity and beat rate values measured in our experiment is like those obtained in [24]. They also showed the ability of the ESP32 to process data. The authors of [25] measured spO<sub>2</sub> and heart rate and their values are comparable to those measured in our experiments. The authors of [26] used a smartphone as gateway and sensors to detect heart anomalies but they tested their system with generated ECG data. While the authors of [27] showed the ability to achieve the processing of ECG data with Arduino and ESP8266.

In the first experiment, we evaluated our Machine Learning algorithm on several ECGs to evaluate its ability to detect anomalies. Our experiment was performed with 2 normal ECGs and 2 abnormal ECGs. Moreover, the transmission of an ECG signal has demonstrated the ability of the architecture to collect, process, and store data in real-time.

In the second experiment, we evaluated the ability of the architecture to manage data arriving at regular intervals of time or in burst mode.

In the end, our architecture has been tested with our own sensors to alleviate problems of compatibility and interoperability problems. Indeed, experimentation in a medical context requires a series of administrative steps and prior agreements with equipment manufacturers to have access to raw or pre-processed data from their sensors. Moreover, this architecture alone does not allow for complete patient follow-up. Indeed, the development of connectors to interact with the architectures present in hospitals before and after hospitalization must still be carried out to ensure a complete follow-up from home to the hospital and vice versa. To achieve this, it will be necessary to support and certify the architecture for various formatting and data storage standards to ensure security, privacy, and system interoperability. These processes are lengthy and will likely require several more years of work.

Indeed, we must implement security measures and best practices to apply on RAMi, including—but not limited—to “MQTT”, “Data”, and “Application”.

MQTT: A firewall with advanced configuration is needed for a connection in the MQTT broker. As MQTT uses TCP, we can block all “UDP”, then study the blocking of “ICMP”, block all ports not used by MQTT, except the default ones (8883 and 1883), block everything but the IP address range of the known clients, and use SSH instead of root access.

Data: For maximum protection of personal data, measures can be implemented by pre-processing the data—i.e., before ingesting it—such as anonymization, which is to make re-identification of the data difficult (if not impossible), and pseudonymization, which is reversible and could take the form of using advanced cryptographic or hash methods with key or salting, encryption, tokenization, and other relevant techniques.

Application: It is necessary to update software and keep all libraries and packages up to date. We can opt to use SELinux which prevents programs from acting differently after an update. Thus, even if an attacker manages to replace a program with a harmful

version, nothing will change afterwards. In addition, an interesting piece of software called “Fail2Ban” checks several log files on a Linux computer, including the SSH log, and looks for brute force attacks. It depends on iptables and updates the rules automatically if malicious clients are found.

Finally, the architecture has been tested on a small scale and needs to be tested for use in production or on a larger scale.

## 7. Conclusions and Future Direction

In this paper, we presented RAMi, a new platform Fog/Cloud to process medical data to monitor elderly, bedridden, and convalescent patients. Our architecture is designed to detect patterns of anomalies in the data in real-time but also to eliminate false positives by integrating the particularities of the patients.

Moreover, we have proposed alternatives to most of the software bricks that compose the architecture so that it can be adapted to the needs or particularities of other use cases. The fact of being relatively independent of the software bricks allows us to protect ourselves from the disappearance or abandonment of some of them over time and thus make the architecture hopefully more sustainable.

In the end, the architecture is limited to real-time processing of time series data, which allows for the collection of most of the data produced by the sensors of convalescent or elderly patients. Currently, the architecture is not designed to process and import old patient data.

Our experiments have proven the proper functioning of our architecture and the ability of the node to correctly process the data streams from the ECG sensor and other sensors. The values measured by the sensors are in similar ranges to those in the literature.

In future work, we will extend our architecture to all kind of medical data. With this aim, we will implement Fast Healthcare Interoperability Resources (FIRH) v3 proposed by Health Level Seven International (HL7) to describe data and improve their interoperability. The interoperability between data will allow to develop new applications. The development of a connector will also allow to interconnect INAH [12] and RAMi in the future. In addition, special attention will be given to the security of the platform to ensure resilience in case of attack.

In order to improve the security of our architecture, we plan to replace the MQTT server where information is transmitted in clear text by a version integrating TrustZone, as proposed by Segarra et al. [35,37].

Finally, our medical node will be equipped with additional sensors to obtain alerts related to dust (GP2Y1010AU), CO<sub>2</sub> (MH-Z14), CO, and NO<sub>2</sub> (MICS-4514), as suggested in [38]. In addition, other devices may be considered to complement the sensors and meet specific needs or diseases among those identified by [39]. In parallel, we develop an edge computing architecture to allow interaction and data exchange between medical infrastructures [40].

**Author Contributions:** Conceptualization, O.D., J.B.N.P.; methodology, J.B.N.P.; software, D.M.; validation, A.G., N.B. and J.B.N.P.; formal analysis, U.K.L.; investigation, N.B., M.H.; resources, A.G.; data curation, U.K.L.; writing—original draft preparation, O.D., J.B.N.P., X.L.; writing—review and editing, S.M.; visualization, D.M.; supervision, S.M and P.M.; project administration, O.D.; funding acquisition, S.M. and O.D. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was partially funded by Infortech and Numediart Institutes of UMONS and the APC was funded by MDPI.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Acknowledgments:** The authors would like to express their gratitude to Meryem Elmoulat accepting to edit the writing of this paper.

**Conflicts of Interest:** The authors declare no conflict of interest.

### Abbreviations

The following abbreviations are used in this manuscript:

ACL	Access Control List
AES	Advanced Encryption Standard
AI	Artificial Intelligence
API	Application Programming Interface
ARM	Advanced RISC Machines
BCB	Block Chain Based
CBC	Cipher Block Chaining
CMIOT	Community Medical Internet of Things
DDOS	Distributed Deny of Service
ECG	Electrocardiography
FaaS	Function-as-a-Service
FIRH	Fast Healthcare Interoperability Resources
HDFS	Hadoop Distributed File System
HI	Heat Index
HIoT	Healthcare Internet of Things
HL7	Health Level Seven International
IAS	Intel Attestation Service
ICMP	Internet Control Message Protocol
ICU	Intensive Care Unit
IDE	Integrated Development Environment
INAH	The Institute of Analytics for Health
IoT	Internet of Things
IoMT	Internet of Medical Things
LTS	Long Term Support
MEC	Mobile Edge Computing
MITM	Man-in-the-middle
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
NPU	Neural processing unit
OSA	Obtrusive Sleep Apnea
PUF	Physically Unclonable Function
PBAN	Personal Body Area Network
QoS	Quality of Service
RFID	Radio Frequency Identification
SDK	Software Development Kit
SDN	Software-Defined Networking
SEV	Secure Encrypted Virtualization
SGX	Software Guard Extensions
SSL	Secure Sockets Layer
spO2	Peripheral oxygen saturation
TCP	Transmission Control Protocol
TEE	Trusted Execution Environment
TLS	Transport Layer Security
TOPS	Total Operations Processing System
UDP	User Datagram Protocol
WPAN	Wireless Personal Area Network

### References

1. Kaissis, G.A.; Makowski, M.R.; Rückert, D.; Braren, R.F. Secure, privacy-preserving and federated machine learning in medical imaging. *Nat. Mach. Intell.* **2020**, *2*, 305–311.

2. Kashani, M.H.; Madanipour, M.; Nikravan, M.; Asghari, P.; Mahdipour, E. A systematic review of IoT in healthcare: Applications, techniques, and trends. *J. Netw. Comput. Appl.* **2021**, *192*, 103164.
3. Sun, L.; Jiang, X.; Ren, H.; Guo, Y. Edge-cloud computing and artificial intelligence in internet of medical things: Architecture, technology and application. *IEEE Access* **2020**, *8*, 101079–101092.
4. Debauche, O.; Trani, J.-P.; Mahmoudi, S.; Manneback, P.; Bindelle, J.; Mahmoudi S.A.; Guttadauria, A.; Lebeau, F. Data Management and Internet of Things: A Methodological Review in Smart Farming. *Internet Things* **2021**, *14*, 100378.
5. Debauche, O.; Mahmoudi, S.; Manneback, P.; Lebeau, F. Cloud and Distributed Architectures for Data Management in Agriculture 4.0: Review and Future Trends. *J. King Saud Univ. —Comput. Inf.* **2021**, *in press*.
6. Nkamla Penka, J.B.; Mahmoudi, S.; Debauche, O. An Optimized Kappa Architecture for IoT Data Management in Smart Farming. *Int. J. Ubiquitous Syst. Pervasive Netw.* **2022**, *17*, 59–65.
7. Gaur, A.; Scotney, B.; Parr, G.; McClean, S. Smart City Architecture and its Applications Based on IoT. *Procedia Comput. Sci.* **2015**, *52*, 1089–1094.
8. Loria, M.P.; Toja, M.; Carchiolo, V.; Malgeri, M. An efficient real-time architecture for collecting IoT data. In Proceedings of the 2017 Federated Conference on Computer Science and Information Systems (FedCSIS), Czech Republic, 4–7 September 2017; pp. 1157–1166.
9. Duan, R.; Chen, X.; Xing, T. A QoS Architecture for IOT. In Proceedings of the 2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing, Dalian, China, 19–22 October 2011; pp. 717–720.
10. Ta-Shma, P.; Akbar, A.; Gerson-Golan, G.; Hadash, G.; Carrez, F.; Moessner, K. An Ingestion and Analytics Architecture for IoT Applied to Smart City Use Cases. *IEEE Internet Things J.* **2018**, *5*, 765–774.
11. Debauche, O.; Mahmoudi, S.; Mahmoudi, S.A.; Manneback, P.; Lebeau, F. A new edge architecture for ai-iot services deployment. *Procedia Comput. Sci.* **2020**, *175*, 10–19.
12. Delsate, T.; Lessage, X.; Boukhebouze, M.; Ponsard, C. INAH: The Ethical & Secure Platform for Medical data Analysis. *ERCIM NEWS* **2021**, *126*, 20–21.
13. Debauche, O.; Mahmoudi, S.; Manneback, P.; Assila, A. Fog IoT for Health: A new Architecture for Patients and Elderly Monitoring. *Procedia Comput. Sci.* **2019**, *160*, 289–297.
14. Nguyen, D.C.; Pathirana, P.N.; Ding, M.; Seneviratne, A. Bedgehealth: A decentralized architecture for edge-based iomt networks using blockchain. *IEEE Internet Things J.* **2021**, *8*, 11743–11757.
15. Razdan, S.; Sharma, S. Internet of Medical Things (IoMT): Overview, Emerging Technologies, and Case Studies. *IETE Tech. Rev.* **2021**, 1–14.
16. Boutros-Saikali, N.; Saikali, K.; Abou Naoum, R. An IoMT platform to simplify the development of healthcare monitoring applications. In Proceedings of the 2018 Third International Conference on Electrical and Biomedical Engineering, Clean Energy and Green Computing (EBECEGC), Beirut, Lebanon, 25–27 April 2018.
17. Girardi, F.; De Gennaro, G.; Colizzi, L.; Convertini, N. Improving the healthcare effectiveness: The possible role of EHR, IoMT and blockchain. *Electronics* **2020**, *9*, 884.
18. Papaioannou, M.; Karageorgou, M.; Mantas, G.; Sucasas, V.; Essop, I.; Rodriguez, J.; ymberopoulos, D. A Survey on Security Threats and Countermeasures in Internet of Medical Things (IoMT). *Trans. Emerg. Telecommun. Technol.* **2022**, *33*, e4049.
19. Zhang, H.; Li, J.; Wen, B.; Xun, Y.; Liu, J. Connecting intelligent things in smart hospitals using NB-IoT. *IEEE Internet Things J.* **2018**, *5*, 1550–1560.
20. Liu, C.; Chen, F.; Zhao, C.; Wang, T.; Zhang, C.; Zhang, Z. IPv6-Based architecture of community medical internet of things. *IEEE Access* **2018**, *6*, 7897–7910.
21. Ed-daoudy, A.; Maalmi, K. A new Internet of Things architecture for real-time prediction of various diseases using machine learning on big data environment. *J. Big Data* **2019**, *6*, 1–25.
22. Yacchirema, D.C.; Sarabia-Jácome, D.; Palau, C.E.; Esteve, M. A smart system for sleep monitoring by integrating IoT with big data analytics. *IEEE Access* **2018**, *6*, 35988–36001.
23. Jangra, P.; Gupta, M. A Design of Real-Time Multilayered Smart Healthcare Monitoring Framework Using IoT. In Proceedings of the 2018 International Conference on Intelligent and Advanced System (ICIAS), Kuala Lumpur, Malaysia, 13–14 August 2018; pp. 1–5.
24. Islam, M.M.; Rahaman, A.; Islam, M.R. Development of smart healthcare monitoring system in IoT environment. *SN Comput. Sci.* **2020**, *1*, 1–11.
25. Ramírez López, L.J.; Rodríguez Garcia, A.; Puerta Aponte, G. Internet of things in healthcare monitoring to enhance acquisition performance of respiratory disorder sensors. *Int. J. Distrib. Sens. Netw.* **2019**, *15*, 1–10.
26. Neyja, M.; Mumtaz, S.; Huq, K.M.S.; Busari, S.A.; Rodriguez, J.; Zhou, Z. An IoT-based e-health monitoring system using ECG signal. In Proceedings of the GLOBECOM 2017–2017 IEEE Global Communications Conference, Singapore, 4–8 December 2017; pp. 1–6.
27. Ruman, M.R.; Barua, A.; Rahman, W.; Jahan, K.R.; Roni, M.J.; Rahman, M.F. IoT based emergency health monitoring system. In Proceedings of the 2020 International Conference on Industry 4.0 Technology (I4Tech), Pune, India, 13–15 February 2020; pp. 159–162.
28. Rahimi, M.; Navimipour, N.J.; Hosseinzadeh, M.; Moattar, M.H.; Darwesh, A. Cloud healthcare services: A comprehensive and systematic literature review. *Trans. Emerg. Tel. Tech.* **2022**, *e4473*, 1–27.



29. Ali, M.S.; Vecchio, M.; Pincheira, M.; Dolui, K.; Antonelli, F.; Rehmani, M.H. Applications of Blockchains in the Internet of Things: A Comprehensive Survey. *IEEE Commun. Surv. Tutor.* **2018**, *21*, 1676–1717.
30. Pelekoudas-Oikonomou, F.; Zachos, G.; Papaioannou, M.; de Ree, M.; Ribeiro, J.; Mantas, G.; Rodriguez, J. Blockchain-based security mechanisms for IoMT Edge networks in IoMT-based healthcare monitoring systems. *Sensors* **2022**, *22*, 2449.
31. Ruggeri, A.; Celesti, A.; Fazio, M.; Villari, M. An Innovative Blockchain-Based Orchestrator for Osmotic Computing. *J. Grid Comput.* **2022**, *20*, 1–17.
32. Russinovich, M.; Costa, M.; Fournet, C.; Chisnall, D.; Delignat-Lavaud, A.; Clebsch, S.; Vaswani, K.; Bhatia, V. Toward confidential cloud computing. *Commun. ACM* **2021**, *64*, 54–61.
33. Valadares, D.C.G.; Will, N.C.; Spohn, M.A.; de Souza Santos, D.F.; Perkusich, A.; Gorgônio, K.C. Confidential computing in cloud/fog-based Internet of Things scenarios. *Internet Things* **2022**, *19*, 100539.
34. Valadares, D.C.G.; Will, N.C.; Caminha, J.; Perkusich, M.B.; Perkusich, A.; Gorgônio, K.C. Systematic Literature Review on the Use of Trusted Execution Environments to Protect Cloud/Fog-Based Internet of Things Applications. *IEEE Access* **2021**, *9*, 80953–80966.
35. Segarra, C.; Delgado-Gonzalo, R.; Schiavoni, V. MQT-TZ: Hardening IoT Brokers Using ARM TrustZone. In Proceedings of the 39th International Symposium on Reliable Distributed Systems (SRDS 2020), Shanghai, China, 21–24 September 2020.
36. Dhiman, G.; Juneja, S.; Mohafez, H.; El-Bayoumy, I.; Sharma, L.K.; Hadizadeh, M.; Islam, M.A.; Viriyasitavat, W.; Khandaker, M.U. Federated Learning Approach to Protect Healthcare Data over Big Data Scenario. *Sustainability* **2022**, *14*, 2500.
37. Segarra, C.; Delgado-Gonzalo, R.; Schiavoni, V. MQT-Tz: Secure MQTT Broker for Biomedical Signal Processing on the Edge. *Stud. Health Technol. Inform.* **2020**, *270*, 332–336.
38. Taştan, M.; Gökozan, H. Real-time monitoring of indoor air quality with internet of things-based E-nose. *Appl. Sci.* **2019**, *9*, 3435.
39. Uddin, M.Z.; Khaksar, W.; Torresen, J. Ambient sensors for elderly care and independent living: A survey. *Sensors* **2018**, *18*, 2027.
40. Debauche, O.; Mahmoudi, S.; Nkamla Penka, J.B.; Lessage, X.; Manneback, P.; Guttadauria, A. Towards a New Edge Computing Architecture for Internet of Medical Things. In Proceedings of the 3rd international conference on Embedded & Distributed Systems, EDiS'2022, Nanjing, China, 17–18 December 2022.