Abstract—Lately, the advancement in circuit technology combined with the design of low cost embedded devices have resulted in an infiltration of the latter into everyday humans' lives. To exploit the full potential of ubiquitous embedded devices, a network is used for their inter-communication, offering advanced real-time monitoring. This paradigm, known as Internet of Things (IoT), is steadily consolidated and promises to offer a wide variety of applications. However, with the adoption of IoT, new challenges arise, such as the design of architectures able to support the requirements of the new applications. Towards this goal, we explore a three layered architecture, able to acquire, process and store Healthcare data as well as to provide real-time decision making. We use ECG signal arrhythmia detection as our use case evaluation scenario, and compare different techniques for wireless communication, storage and data classification. Experimental results show that, our architecture provides real-time decision making, with an average delay of 15 μs and that different communication technologies achieved to provide up to 10% lower power consumption on the monitoring devices.

I. INTRODUCTION

The tremendous improvements in the domain of embedded devices over the past few years have led to new technologies enabling hardware to become smaller, cheaper and extended with communication capabilities. In this way, embedded devices are able to connect and interact with the environment. This new paradigm, also known as “Internet of Things” (IoT), is a network of objects capable of detecting and communicating information between each other.

This vision has spawned a lot of research efforts both in industry and academia and the outcome is a very diverse set of candidate IoT applications, expressing high requirements in processing resources, both near the user and at the Cloud level. These high requirements turned out to be contradictory to the core principles of IoT (Edge) nodes, which describe simple processing elements, low power consumption and fast user response. Inevitably, new architectures were presented, which include an intermediate layer of processing devices, bridging the gap between Edge and Cloud infrastructure and these devices were introduced as the Fog layer [3]. While the introduction of this layer effectively mitigated the processing requirements, it also imposed an extra perspective on the design of IoT applications, since their functionality must be partitioned in a three-tier processing architecture.

In this work, we present the development of a use case, medically oriented IoT application in order to explore, quantify and design alternatives considering an Edge-Fog-Cloud architecture. The use case was chosen due to the fact that IoT infrastructure has struck the attention of the healthcare sector in order to provide more efficient services and be able to manipulate vast amount of patient data and transform it into valuable information from which expert knowledge can be extracted. In addition, an effective medical application design in crucial in order to provide the necessary foundation for developing the Decision Support Systems [11] of the future, which should include high sophistication and increased resiliency given their life-critical aspect.

II. RELATED WORK

Over the past few years, much research has been conducted regarding the design and implementation of healthcare systems over two or even three layers. Authors in [17] propose a three layered architecture consisting of sensors for collecting and transmitting data, a mobile phone for displaying and processing data and the Cloud for storing data and take decisions based on them, using Artificial Neural Networks (ANNs). Although this work presents most of the major challenges of modern healthcare systems, such as communication latencies, power consumption and detection accuracy of the ANN, it does not explore and compare design alternatives regarding the communication methods used as well as the storage databases decision making algorithms. In [18], the authors present a two layered architecture which provides three offloading levels for processing data to the Cloud based on Hidden Markov Model. Each level offloads different amount of workload to be executed on the Cloud. Similarly with [17] this work lacks of comparing the different communication technologies, database storage alternatives and decision making algorithms.

Authors in [13] present a system architecture design based on three layers. The Edge layer is implemented by utilizing the Savvy ECG sensor in order to capture the ECG signal. Moreover, Cloud layer is responsible for ECG signal filtering and analysis but algorithms for the signal analysis are not presented. Lastly, the mobile application does not
have dynamic(real-time) plotting capabilities and there are no experimental results presented.

In [1], [12] two custom electronic circuit designs for ECG signal capturing, filtering and digitization are presented. In these works, only classic Bluetooth communication is supported and there is no implementation or design proposition for the Cloud layer.

Finally, a Cloud layer architecture design and implementation based on Hadoop framework is presented in [21]. The application responsible for transmitting data, comes with data encryption and compression features. The Cloud storage consists of MySQL database and non-relational distributed file system storage (HDFS). The ECG signal analysis algorithms running on the Cloud are integrated into MapReduce framework. Experimental results and Edge layer design are not presented.

III. SYSTEM ARCHITECTURE OVERVIEW

Our target system architecture is illustrated in Fig. 1, consisting of three distinct computation layers. The bottom one, referred to as Edge [7], includes an enormous number of computational devices of diverse characteristics, whose common feature is that they are located in close vicinity to the end user. They are also characterized by high mobility since they are portable or wearable, operate on battery power and are normally equipped with a wireless communication interface to be able to transmit data to higher communication layers.

The middle layer of the architecture is called Fog [3] and is responsible for gathering and processing data from the Edge nodes, thus alleviating the burden of uploading all the Edge-derived data to the Cloud. In addition, some applications with real-time requirements cannot tolerate the uploading latency to the Cloud and thus processing is migrated from the Edge to the Fog which involves low latency, short ranged communication.

The higher level of the architecture is the Cloud infrastructure which is responsible for data storage and analysis of the data uploaded from the combination of Fog and Edge levels. It also includes all the necessary functionality for this data to be accessed and manipulated by all interested third parties. Different from the belief that the Cloud has infinite resources, we consider that effective design is mandatory in this level and can result in various socio-economic benefits [19].

IV. SYSTEM IMPLEMENTATION DESIGN SPECIFICS

The presented system design is based on the three-layer architecture presented in Section III.

A. Edge Layer

1) Optimizing execution flow - DSE framework: On the target board, a diagnosis decision making algorithm must be locally executed. In this design campaign, we focus on classification algorithms, whose requirements are high accuracy and acceptable computational intensiveness in order to be executed at real-time. This implies the need for an optimization of the employed algorithm, a process which is hindered by the enormous design space of manually examining all the combinations of the available choices regarding feature extraction techniques and candidate algorithms. Consequently, there is the need of a systematic approach for the exploration of the aforementioned space and thus, we introduce a framework to examine it in an automated manner.

The input dataset of the exploration framework is composed of a collection of observations of the phenomenon whose behavior has to be captured using machine learning algorithms, combined with labels of the classification class they belong to. In addition, the designer designates a set of feature extraction mechanisms and machine learning algorithms to be explored.

The framework automatically iterates over the parameters designated by the application designer, thus producing the outcome of this exploration, which is statistics regarding the accuracy of the combinations of extracted features and classification algorithms. Accuracy is defined as the number of correctly predicted input feature vectors of the classifier against all the input data set.

2) Wireless communication design alternatives: We examined two modes regarding the short range wireless communication. Option 1 is the Classic Bluetooth protocol (BTC), where the Fog device operates as a server and the Edge device as a client, designed with the ability to connect to any compliant device by making use of the Service Discovery Protocol (SDP)[8]. Furthermore, the two devices communicate over the RFCOMM protocol, in which data packets are transmitted in the same order they arrived in the Bluetooth controller even if re-transmission is needed. This adds a small delay in data exchange but ensures data consistency and eliminates the need for packet ordering in server side. Furthermore, communication between the server-client devices is realized with the use of blocking or non-blocking sockets.

Option 2 is the Bluetooth Low Energy (BLE) wireless communication interface. The nature of the model is more object oriented, as the highest layer of its SW stack, i.e. the GATT profile, makes use of entities for data exchange. A typical message exchange process between the three layers of the design is shown in Fig. 2. The Fog device notifies the Edge device of its intention to read data and then begins to
transmit read requests. Then, the Edge device takes the role of the server and the Fog device the role of client. This design decision, takes advantage of the low power consumption of BLE in idle and advertising modes. The server device exposes the characteristics that contain the measured data to scanning devices that are interested to acquire it.

B. Fog Layer

The application allows the user to enable Bluetooth connection and choose if he wants to pair with another device, selecting the type of connection to be initiated. In case of the BTC connection, the application opens a server socket and listens for incoming connection requests. When BLE communication is preferred, the application operates as a client so it has to scan for the available devices prior to initiating a connection. When a connection is established with the Edge device, a request is sent to receive its GATT records, discover the desired service and retrieve the characteristic associated with the intended data. In case of new data, the Edge device sends notifications that the characteristic data has changed and the Fog device reads the data values and stores them locally. The application supports both dynamic and static visualization of the acquired data. Finally, data are uploaded to Cloud in a JavaScript Object Notation (JSON) file format.

C. Cloud Layer

The Cloud layer infrastructure was designed for simple but efficient scaling and smooth integration with existing systems, without the need of exposing technical details of the lower layers of our system. To achieve that, we developed a RESTful web service [6] using the Java Servlet API, in order to receive data from the Fog layer and manage the database transactions of our server. During its life-cycle the servlet instances intercept incoming HTTP requests in order to insert or update database records or execute queries to send stored data back to the client for display. The web service is designed to be compatible with MongoDB and SQL databases. A non-relational database was deemed necessary during the design of the infrastructure in order to achieve high throughput in data transaction servicing and fast patient ECG record retrieval in emergency situations. Furthermore, because MongoDB is schema-less, continuous and consistent operation, in the case data format and relations alter in the future, is ensured.

V. USE CASE: ECG SIGNAL ARRHYTHMIA DETECTION.

To quantify the effectiveness of the presented framework we deal with the problem of arrhythmia detection in ECG signals. The framework operated on a large input data set consisting of more than 100000 heart beats from MIT-BIH database[10], where 75% was utilized as training set and the rest as testing. Every observation of the database is accompanied by its clinical status indication label as dictated by medical experts. Two possible classes are considered, one describing a Normal and one an Abnormal detected heart beat. The chosen features of the heart beat under investigation, as resulted from a feature extraction process, are directed as an input to a machine learning based decision making algorithm which concludes the diagnosis label of the beat.

VI. EXPERIMENTAL EVALUATIONS

A. Experimental Setup

The objective of the Edge layer is data acquisition by utilizing ECG sensors that can be located on the patients body and data transmission through a short distance wireless communication protocol. In this scope we chose to implement the Bluetooth monitor application on the Raspberry Pi 3 model B, featuring an 1.2GHz Quad-Core ARM Cortex-A53, 1GB RAM memory, 802.11 b/g/n Wireless LAN and Bluetooth 4.1 (BTC and BLE) connectivity.

The Raspbian Jessie Lite was used as operating system, with no graphical interface to minimize storage requirements and CPU utilization by the OS. The BTC application part was develop using Bluez stack [2], while the respective BLE part using a cross-compiled version of Qt Library [14].

To link the Edge layer with the remote server for further data processing and storage, an Android application was implemented to act as the Fog layer. The application was evaluated on a Sony Xperia Z smartphone, which operates on Android 5.1 OS Its hardware is based on Qualcomm APQ8064 Snapdragon S4 Pro chipset with a 1.5 GHz quad-core CPU, 2 GB RAM, Bluetooth Classic and Low Energy capabilities. The Cloud layer-remote server infrastructure was developed and implemented on an infrastructure similar to work of [20].

All experimental evaluations were conducted by transmitting values of ECG data as packets from the monitor application (Edge) to the smart phone (Fog). Furthermore, Android Studio was used both during the development and testing of the smartphone application, as it offers a wide range of tools to track CPU, GPU, memory management and network utilization of the target device while an application is running.

B. Employing the decision making framework in the case of ECG signal arrhythmia detection.

Regarding the feature extraction mechanisms, the Discrete Wavelet Transform was used as in [16], due to the fact that
the ECG signal exhibits variations in its period and thus the
multi-scale decomposition nature of the DFT is much more
capable of capturing these asymmetries in the time evolution
of the signal.

As a base of the DWT Daubechies, bi-orthogonal and reverse
bi-orthogonal mother wavelets were examined. For this case
study we used up to 4 levels of decomposition as in [16] and
the capabilities of the presented framework were utilized to
check combinations of coefficients originating from different
levels of decomposition, in an effort to discover the features
which most accurately capture the arrhythmia detection de-
cision making problem. Given that we have two channels
of incoming ECG signal, a heart beat is described of 8 * 2 = 16 set of wavelet coefficients. Due to the fact that all
combinations of these sets would not only create an immense
design space but also create sets of input feature vectors
greater in size compared to the original signal, we narrowed
the exploration down to combinations of up to 2 subsets of
coefficients of all levels of decomposition. Finally, as far as
candidate decision making algorithms are concerned, Artificial
Neural Networks[9], Support vector machines[5] and Random
Forests[4] were examined.

In Fig. 3, we present the summarized results regarding the
classification accuracy of the examined couples of wavelet
transformation and classification algorithm. In all sub-figures,
the X axis represents combinations of one or two of the sets of
wavelet coefficients (both detailed and approximate) produced
by each mother wavelet. As stated, these standalone sets are
16 in total and the rest 120 examined sets are combinations
of them. The Y axis represents the classification accuracy
achieved by the examined couple of feature extraction mech-
anism and classifier.

The figures show that in general all classifiers exhibit
high accuracy, of more than 93% in every case. The highest
classification accuracy is achieved by the SVM classifier
reaching 99.57%. Regarding the ability of the DWT to capture
the classification problem under investigation, it is evident
that it is a quite appropriate candidate. The overview of the
figures also shows that certain examined combinations of
coefficients significantly fare worse than others as for example
combinations 9 to 12 (approximate coefficients of levels 2 to 4
of channel 2 of the ECG signal). This suggests, that these parts
of the transformation fail to capture the distinct characteristics
of the investigated problem. Summing up, we consider SVM
classifier as the best choice given that it provides the maximum
accuracy, while its structure allows for optimizations aiming
at execution with real-time constraints [15].

The optimized version of the SVM model, derived from the
analysis as the one with the highest classification accuracy, was
used as the core model in order to quantify the execution of
different SVM configurations on the target embedded device.
The goal of the experiment was to evaluate how the target
device behaves, in case of a scaling in the computational
requirements of the classifier. In order to produce realistic
SVM models, we used the ECG signals of MIT-BIH database
as the starting point in order to create ECG signals of higher
or lower sampling rate, i.e. more or less detailed monitoring
of the ECG activity. These signals, resulted in SVM models
of different computational requirements which increase as
the sampling frequency increases. The available sampling
frequencies were 180, 360 (original), 720, 1440 and 2000 Hz.
Fig. 4 summarizes the average required time per heart beat
to be diagnosed, on the target Edge device. We observe that
due to our offline analysis, the derived SVM models are of
reasonable computational intensiveness and thus the embedded
device is capable of processing them in a few μs, even for the
highest sampling frequency of ECG.

C. Bluetooth connection evaluation

The performance of the Bluetooth was quantified in terms
of packet transmission delay, CPU utilization and power
consumption, based on experiments of up to 1000 packet
uploading from the the Raspberry Pi (Edge) to the smart phone

![Fig. 4: SVM execution latency w.r.t. input ECG sampling rates](image-url)
Initially, the BTC link was evaluated, by measuring the packet delay between two successive received packets. Assuming that the first packet was received at $t_{packet1}$ and the second at $t_{packet2}$, then the delay is $t_{delay} = t_{packet2} - t_{packet1}$. The experiment presented in Fig. 5 tries to capture the variation in the transmission delay of the protocol, in a real-life scenario with increasing distance between transmitter and receiver as well as with the inclusion of obstacles like walls. In addition, all experiments were conducted in a non-isolated environment with Wi-Fi networks present, so interference from Wi-Fi signals is included in the measured delay. In X axis the values indicate the distance, while the word Wall implies an obstacle in the transmission path. The results are presented in mean value and standard deviation format and show a high variation, especially when obstacles are involved.

We followed a slightly different approach in calculating packet delay for the BLE communication. We took advantage of the GATT profile’s specifications, so we measured the delay between the client’s GATT read requests and the server’s GATT read responses. Let $t_{request}$ be the time client sends the request and $t_{response}$ the time it receives the server’s response, then the delay is calculated from the formula $t_{delay} = t_{response} - t_{request}$. We followed the same testing procedure as in BTC communication.

Regarding the CPU utilization of our Linux application, it was measured on the target board in 100 milliseconds intervals. The tests where conducted for both BTC and BLE operations and Fig. 6 summarizes the results of scanning, advertising and data transfer for both protocols. The figure is also annotated with critical points of operation.

Lastly, power consumption of the board was estimated via an external USB power measuring device. The board operates at a constant 4.98V voltage and assuming that $I$ is the current drawn from its power supply, the power consumption is approximated as $P = V \times I$. The idle consumption of the board was measured at 2390.4mW, while the consumption for the various operations of the two examined wireless communication protocols are reported in Table I.

### Table I: Classic Bluetooth vs BLE Power Consumption

| Operation        | Classic Bluetooth Consumption (mW) | Low Energy Bluetooth Consumption (mW) |
|------------------|------------------------------------|--------------------------------------|
| Scanning/Advertising | 2589.8                             | 2390.4                               |
| Transfer         | 2589.6                             | 2390.4                               |

---

**D. Android Application**

The profiling of the Android application, was performed using the built-in tools of Android Studio. Fig. 7 presents the CPU and memory utilization for both BLE and BTC connections. As far as BLE is concerned, from seconds 18 to 25 the application is scanning for devices and there is a rise in CPU usage attributed to frequency hopping algorithm of the Bluetooth adapter. At the 30th second, we observe a connection event and subsequent data reception. During this event there is a spike in CPU utilization as the application exchanges information with the remote device in order to set up their connection link.

As far as BTC is concerned, at the 6th second the application initiates the Bluetooth server service, triggering a spike in CPU utilization. Consequently, it listens for incoming connections and during this period CPU usage is almost 0%. At the 37th second the connection is established, leading to another, smaller spike and then data transfer is commenced.

**E. Webservice and Database Comparison**

The MySQL schema consists of two tables, one for patients’ information and one for acquired ECG data of every patient. On the contrary, in MongoDB no schema is needed, as NoSQL databases can handle dynamic data. The absence of relation representation between the data permit storing of records with no fixed fields or pre-defined data types. In addition, the concept of related tables is completely dropped in favour of on-demand document creation for data organization. In order to evaluate the performance of the databases, POST requests were sent to the web service. The performance was evaluated in terms of the required execution latency to insert or update one patient’s record into the database.

In all MySQL queries, the ID column was used as index and we tried different configurations in order to improve performance. Firstly, the database was just only indexing, but in subsequent tests the web container’s connection pooling feature was enabled, which creates and manages database connections and distributes them to the clients. Lastly, the application source code was improved by sending all the INSERT and UPDATE operations as batch, which means that no changes are committed to the database until all the write operations are completed. We also measured the required time to retrieve one patient record from the database.

MongoDB was evaluated using the same tests. One difference is that connection pooling is enabled by default and is managed directly by the database instead of the web container resulting in much better throughput and connection speed compared to MySQL. The patients’ IDs were used as index and no further performance improvement actions were taken. Table II summarizes the execution latency of the aforementioned experiments for both databases. We observe that due to the nature of the store biomedical signal, MongoDB heavily outperforms MySQL database in INSERT and UPDATE operations but fares significantly worse in the SELECT query, even for the unoptimized version of MySQL. After evaluating the experimental results for both databases we can safely conclude...
the ECG signal data should be handled by MongoDB for faster ECG measurement updating and analysis.

VII. CONCLUSION

In this paper, we introduced an integrated three layered architecture able to acquire and process patient’s data. Using ECG signal arrhythmia detection as our use case scenario, we compare several design alternatives both for communication between devices as well as data classification and storage. SVM classifier proves to be the optimal solution for data classification, considering the confined processing capabilities of embedded devices. Our architecture manages to provide real time decision making with an average delay of 15 μs. In addition, experimental results showcase that Bluetooth Low Energy achieves up to 10% less power consumption compared to classic Bluetooth, on a Raspberry Pi board. Finally, for data storage, results showed that MongoDB can process queries up to 178 times faster than a MySQL database.

TABLE II: MySQL Test Times in milliseconds

| Queries | MySQL-No Performance Improvements (ms) | MySQL-Batch and Pooling Enabled (ms) | MongoDB (ms) |
|---------|----------------------------------------|---------------------------------------|----------------|
| INSERT  | 4.3903.44                              | 32.82                                | 31.29          |
| UPDATE  | 1180.56                                 | 53.22                                | 30.98          |
| SELECT  | 14.14                                   | 10.83                                 | 26.74          |

that MySQL is better suited for patient and medical personnel information storing, as this is not a write intensive use case, whereas the ECG signal data should be handled by MongoDB for faster ECG measurement updating and analysis.

REFERENCES

[1] R. Agarwal and M. Parmar. Android based electrocardiogram monitoring system. *International Journal of Engineering Science*, 3683, 2016.

[2] V. Auletta et al. Performance evaluation of web services invocation over bluetooth. In *Proceedings of the ACM international workshop on Performance monitoring, measurement, and evaluation of heterogeneous wireless and wired networks*, pages 1–8. ACM, 2006.

[3] F. Bonomi et al. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16. ACM, 2012.

[4] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[5] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[6] R. Fielding and R. Taylor. Principled design of the modern web architecture. *ACM Transactions on Internet Technology*, 2(2), 2002.

[7] P. Garcia Lopez et al. Edge-centric computing: Vision and challenges. *ACM SIGCOMM Computer Communication Review*, 45(5):37–42, 2015.

[8] E. A. Gryazin. Service discovery in bluetooth. *Group for Robotics and Virtual Reality. Department of Computer Science. Helsinki University of Technology, Helsinki, Finland. Published at NEC CiteSeer, Scientific Literature Digital Library*, 2006.

[9] J. J. Hopfield. Artificial neural networks. *Circuits and Devices Magazine, IEEE*, 4(5):3–10, 1998.

[10] G. Moody and R. Mark. The impact of the mit-bih arrhythmia database. *Engineering in Medicine and Biology Magazine, IEEE*, 20(3), 2001.

[11] M. A. Musen et al. Clinical decision-support systems. In *Biomedical informatics*, pages 663–674. Springer, 2014.

[12] C. Rodriguez et al. Wireless ecg based on bluetooth protocol: design and implementation. In *Proc. of IEEE int. conf. on Information Technologies in Biomedicine*, 2006.

[13] A. Stojmenski et al. A mobile application for ecg detection and feature extraction. 2016.

[14] N. O. Toolkit. The qt company’s qt.

[15] V. Tsoutsouras et al. An exploration framework for efficient high-level synthesis of support vector machines: Case study on ecg arrhythmia detection for xilinx zynq soc. *Journal of Signal Processing Systems*, pages 1–21, IEEE, 2017.

[16] E. D. Uebiyi. Ecg beats classification using multiclass support vector machines with error correcting output codes. *Digital Signal Processing*, 2017.

[17] X. Wang et al. Enabling smart personalized healthcare: a hybrid mobile-cloud approach for ecg telemonitoring. *IEEE journal of biomedical and health informatics*, 18(3):739–745, 2014.

[18] X. Wang et al. A hidden markov model based dynamic scheduling approach for mobile cloud telemonitoring. In *Biomedical & Health Informatics (BHI), 2017 IEEE EMBS International Conference on*, pages 273–276, IEEE, 2017.

[19] A. J. Younge et al. Efficient resource management for cloud computing environments. In *Green Computing Conference, 2010 International*, pages 357–364. IEEE, 2010.

[20] G. Zervakis et al. Performance-power exploration of software-defined big data analytics: The agile cloud backend. In *Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS), 2016 International Conference on*, pages 312–319. IEEE, 2016.

[21] B. Zhou et al. Cloud-based dynamic electrocardiogram monitoring and analysis system. In *Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), International Congress on*, pages 1737–1741. IEEE, 2016.