A human-centered Web-based tool for the effective real-time motion data collection and annotation from BLE IoT devices

Andreas Bardoutsos[1,2], Dimitris Markantonatos[1,2], Sotiris Nikoletseas[1,2], Paul G. Spirakis[1,3], and Pantelis Tzamalis[1,2]

[1]Computer Engineering and Informatics Department, University of Patras, Patras, Greece
[2]Computer Technology Institute and Press "Diophantus" (CTI), Patras, Greece
[3]Department of Computer Science, University of Liverpool, UK
bardoutsos@ceid.upatras.gr, markantonatos@ceid.upatras.gr, nikole@cti.gr, p.spirakis@liverpool.ac.uk, tzamalis@ceid.upatras.gr

***Abstract*—The effective utilization of real-world data is an integral part of any IoT monitoring or AI-assisted system. Thus, data collection and annotation is an important step towards the successful development and realization of such systems. Nevertheless, in order to create reliable datasets, current data collection and annotation methodologies often require a controlled environment while also the presence of the volunteer contributing to the process, or any subject for that matter, and an expert, monitoring the procedure, is mandatory. These processes are heavily restrained by the recent COVID-19 pandemic outbreak.**

To address such issues, in this paper we propose a human-centered Web-based dataset creation and annotation tool that utilizes the Web Bluetooth API. The user can effectively collect gestures from a nearby device that supports the BLE protocol, assign tags to the collected data, and store them remotely, in real-time. The data storage, as well as its annotation, can also be performed remotely by an expert stakeholder. An off-the-shelf wearable sensorial device has been used indicatively for our tool demonstration purposes. To the best of our knowledge, this is the first attempt that exploits the Web Bluetooth API capabilities for the development of a Browser-based real-time data collection, storage, and annotation tool. Our tool can be also expanded to other applications that use the sensing device with only minor configuration changes and is also operable through any smart-device that supports a Web-Browser. Furthermore, our tool's performance matches that of native applications'. Finally, the tool is successfully deployed and validated by integrating it into our ongoing ML platform that is related to allergic rhinitis gesture recognition.

***Index Terms*—the Internet of Thing (IoT), Web Bluetooth API, dataset creation, machine learning, mHealth, eHealth, data annotation**

I. Introduction

The world today is transforming through the technology of the Internet of Things (IoT), as more and more smart devices are incorporated into our everyday life. IoT enables sensing, reaction, and interaction of devices with the physical world, thus its adoption in the production and economic growth activities turns Industry 4.0 vision into a realization. The proliferation of IoT systems generates an enormous amount of data that relay information to everyone. This information is related to countless aspects of everyday life and almost everything that surrounds us, thus, leading to characterize our era as a "data-driven" one.

Some of those aspects are Human Activity Recognition (HAR) or Human Gesture Recognition (HGR) which target the discovery of human body movement patterns related to daily activities and gestures. Generally speaking, body movement is recorded by motion sensors, such as the accelerometer and gyroscope that nowadays are common both in smartphones and wearables. These sensors' data from such devices is either stored locally in the device and later are manually retrieved, or the data is streamed to a nearby device by utilizing a wireless transfer protocol such as WiFi or Bluetooth. Moreover, with the recent advances in Data Analysis and Machine Learning (ML), even tight, complicated, and heterogeneous movement patterns can be monitored, modeled, or even get recognized with high accuracy by exploiting the huge information that comes from this data [1].

Following the same design principles as Industry 4.0 for interoperability, decentralization, service orientation, and modularity, Healthcare services are also transforming towards Healthcare 4.0 in order to better respond to the constantly increasing demand, reduce the operational costs and provide higher quality treatment [2]. A fundamental aspect of Healthcare 4.0 is Physical Activity Recognition and Monitoring which includes HAR and HGR. More specifically, gaining insights into a patient's daily physical activity in a privacy non-intrusive manner is a corner-stone for concepts like smart health and smart rehabilitation, as well as Ambient Assisted Living (AAL) [3]. Other paradigms of IoT devices utilization in mHealth and eHealth systems include disease symptoms detection and monitoring [4].

The problem. Nevertheless, it is widely known that a mandatory process to the development of successful healthcare-related AI or monitoring systems, in order to yield better outcomes, is the creation of large and reliable datasets. Real-world data should be recorded as they are performed by

different subjects and in diverse attitudes, to ensure that the data exploitation procedure is not restrained to a specific movement occurrence. Furthermore, the data collection procedure usually requires expert supervision to guarantee that activities are performed acceptably, as well as to provide reliable annotation. Since pattern recognition is based on labeled datasets for training, adequate and reliable annotation is crucial for such systems performance. Additionally, certain concerns have to be underlined regarding the aforementioned procedure, especially for the HAR and HGR systems. The data heterogeneity is highly dependent on the availability of volunteers that contribute to performing each activity as well as how differently each individual performs different repetitions of the same activity. As a conclusion, an important component is the development an application for the efficient data transfer from a wearable device, its proper visualization, and storage to a centralized computer node.

Despite the fact that some device manufacturers provide applications for the efficiency of data collection, each use case varies widely in requirements for processing and annotation. In general, device manufacturers provide Application Programming Interfaces (APIs) that allow the development of such applications. The meeting of requirements however turns to a challenge, in terms of human and funding resources, when support for different platforms or devices is compelled. Hence, dataset creation turns to a demanding task, requiring the setup of a data collection protocol that includes the attendance of several subjects in controlled environments as well as ensuring the availability of specific devices and software development. Eventually, the COVID-19 outbreak has restrained any such research initiative that the presence of volunteers is required but has also triggered the increased necessity for remotely provided healthcare services.

Our contribution. In this paper, a novel Web-based tool for the real-time data collection and annotation from IoT devices is introduced. The designed tool enables the effective data acquisition from BLE-enabled sensorial devices in order to facilitate dataset creation by utilizing the Web Bluetooth API, even remotely. The design choice to implement the tool with the assistance of this technology, is inspired by the fact that users do not have to install third-party applications in order to interact with their BLE devices but instead can access them through a platform-wide and very familiar way which is the Browser. Web Bluetooth API's performance (in terms of the sampling rate achieved) is on the same level as other solutions while also offering additional functionalities due to its Web-based nature. User data is annotated in real-time as far as the proper remote supervision by an expert is guaranteed while also, said expert, is offered the same feature. The proposed tool is accessible by any platform that supports a Web Browser and integrates Bluetooth connectivity (with only exception devices that run on IOS). The collected data is labeled and streamed to a centralized data warehouse for storage and further processing. Moreover, a non-personalized user profile is also maintained to provide later on enhanced insights for the subject as well as help the decision-making process by the corresponding stakeholders. Finally, the tool has experimentally been deployed for testing to an on-going development platform related to HGR for Allergic Rhinitis symptoms. The tool is heavily utilized by medical experts for the remote data collection and annotation of a dataset, consisting of Allergic Rhinitis gestures. While off-the-shelf components are used for the implementation of this tool, to the best of our knowledge, this is the first time that the technology of the Web Bluetooth API is applied in this context.

Roadmap The rest of the paper is organized as follows. Section II includes the related work where several approaches related to our own, are mentioned. In III, the components of our proposed tool are outlined. Section IV, contains the technologies utilized for the development of our tool. Sections V, VI, VII and VIII describe in detail the individual components of our tool along with our development methodology to materialize them. The deployed tool is highlighted in IX, where its utility is validated. Our tool's performance is evaluated in X. Our future work is described in the final section where the paper is concluded too.

II. Related Work

There is a variety of sensing and wearable devices that are selected to be integrated into monitoring and AI-assisted systems. Many options are offered such as Shimmer Sensing [5] or Mbientlab [6] that provide a variety of sensing devices. Each manufacturer though provides an Application Programming Interface (API) for interaction with the corresponding devices. Otherwise, an application may be used such as [7], for personnel not familiar with development processes. Android-enabled smartphones or wearables also support sensorial data recording by utilizing the Android sensors' API [8].

However, not only native applications are able to communicate with IoT devices. [9] proposed a framework that allows classic Web-based applications to immediately interact with nearby devices, a functionality officially introduced by Web Bluetooth Community Group [10]. Web Bluetooth API allows the immediate interaction of a Web-Browser with a BLE enabled device. A demonstration of the technology capabilities is presented in [11]. More specifically, a framework is developed for physical activity monitoring where the capability for remote software updating of nearby sensing devices is highlighted. Additionally, the capability of rapid development and deployment of application-oriented algorithms with minimal impact on available resources and continuous operation of the devices is outlined too. The functionalities that sensing and wearable devices offer, their non-intrusive nature due to their small form factor, and their ability to connect and communicate with nearby devices of higher processing capabilities, are the reasons that such devices are selected in many Healthcare 4.0 approaches.

In this paper, the issues that occurred during our previous data collection procedure presented in [12] are addressed. For the purpose of recognizing Allergic Rhinitis gestures, active allergic patients were invited to participate in a sequence of experiment sessions. The patients were prompted to perform

various identified allergic gestures spontaneously, under the inspection of medical experts. Metabase MMR motion sensor [13] was used for recording the patients' hand motion. However, the procedure was delayed and limited due to several challenges that arose, including software compatibility issues and familiarization of the medical personnel with the manufacturer provided software.

Data annotation was manually performed, at one experiment instance at a time, by encoding both data labels and patient-related information into the recordings' files names, which files' format was CSV. Cloud services that offer file-sharing and file-syncing functionalities (such as Google Drive) were used in order to update and distribute the collected datasets as well as provide access to them by the interested stakeholders. The aforementioned though delayed the data transfer procedures. Eventually, due to the COVID-19 outbreak, all data collection activities were restrained as the patients' attendance to specialized clinics is not encouraged.

The novelty of our tool. As a response to the ongoing situation, an efficient remote data collection and annotation tool has been designed and implemented. Through the utilization of the Web Bluetooth API the interaction with the patients' sensorial devices is decentralized, hence eliminating the requirement for the patients' attendance in a clinic, as long as an expert remote supervision is guaranteed. Since connectivity and data transmission is handled by the Web-Browser, platform-related issues are overridden. Additionally, the data labelling and patients' profiling procedures are accelerated significantly through a user-friendly Web interface (UI) that is easily operated. All the data is uploaded and stored to a remote database, hence enabling unified accessibility, updating, and immediate distribution. Finally, by exploiting the Web Bluetooth API capabilities, the tool can be easily expanded and integrated to data preprocessing algorithms while also supporting more BLE-enabled sensing devices. With minor configuration changes our tool can be adopted by similar HGR or HAR systems.

III. Tool Architecture

The designed tool is a human-centered data-collection tool oriented towards motion recording that is established as a Web Application. Access to the main interface of the application is only provided to users that have previously completed a registration and profile creation procedure. The tool utilizes the data collection from a wearable BLE device that directly communicates and exchanges data with the user's Browser. In particular, a stream of data is collected through the device's accelerometer and gyroscope sensors. Upon user's selection, the data can be stored in a centralized data warehouse while simultaneously is visualized to the user's UI through time-series plots. A data annotation functionality is also enabled, which labels the data that is about to be stored in real-time. This labeling occurs while the procedure is remotely monitored by an expert stakeholder. Last but not least, the procedure can be also directly remotely monitored by an expert stakeholder as a dashboard client is provided, through which access to similar time-series plots to the user's as well as a labeling functionality are enabled. The tool's architecture, according to Fig. 1, can be broken down into four main components which are the following:

The User Profiling Handling Component. This component enables sign in or registration capabilities to the users along with profile creation and update functionality. Additionally, user authentication is also managed by this component.

The Communication and Data Management Component. This component handles the communication with the wearable BLE device as well as its remote programming in order to initialize its sensors. Furthermore, the data transmitted by the device is decoded from its raw hexadecimal format to readable real numbers and then visualized in the main interface through appropriate time-series plots.

The Real-time Storage Component. In this component, the storage of the sensory data to a centralized data warehouse is handled as well as its annotation with the remote assistance of an expert stakeholder.

The Remote Monitoring Component. Finally, this component handles the user's device sensory data forwarding to the expert's dashboard.

The following Sections describe each functionality of the aforementioned components in detail.

IV. Utilized technologies and device

The tool is essentially a modern Web Application where cutting-edge technologies have been applied for its deployment. The front-end part is supported by JavaScript and HTML5 and the back-end by a NodeJS server which hosts our application and handles WebSocket connections. The WebSocket connections were implemented with the assistance of the NodeJS Socket.io library [14]. A NoSQL database was selected as it is preferred for storing the sensory time-series data which in our case was the MongoDB. The connection to the database is assisted by the NodeJS MongooseJS framework [15]. The communication between the Browser and the BLE wearable device is enabled via the Web Bluetooth API. In order to better understand the technology, an overview follows describing the Web Bluetooth API and the communication protocol BLE on which the API is based on.

Bluetooth Low Energy (BLE). Bluetooth Low Energy is a wireless personal area network technology introduced by the Bluetooth Special Interest Group (Bluetooth SIG) [16]. It is focused on low-power operation and aimed towards applications that are in need of such a feature. As detailed in [17] and [18], the BLE protocol stack consists of three main components. *a)* The *Controller* that is composed of the lowest layers in the BLE protocol stack. *b)* The *Host* that comprises higher layers such as, the *Generic Attribute Profile (GATT)* and the *Attribute Protocol (ATT)*, which are the two more relevant to our research purposes. The ATT defines the communication between two devices via specific data structures named *Attributes*, while the GATT is the layer that handles the Attributes (ATT) in order to make the device's functionalities accessible or expose them to another

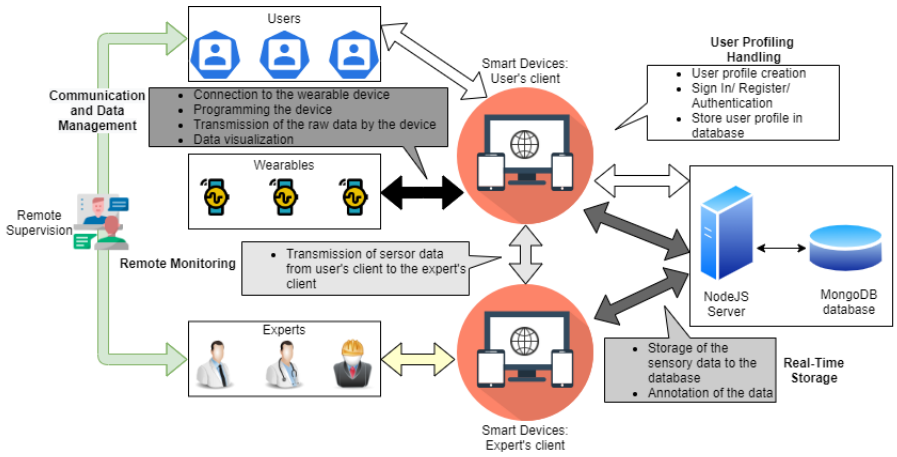

Fig. 1. High-level overview of the tool's architecture. Highlighted with a) white coloured arrows: The User Profiling Handling Component, b) black coloured arrows: The Communication and Data Management Component, c) grey coloured arrows: The Real-time Storage Component and d) light grey coloured arrows: The Remote Monitoring Component.

device. Other layers are the *Generic Access Profile (GAP)*, *Logical Link Control and Adaptation Protocol (L2CAP)* and the *Security Manager Protocol (SMP)*. An overview of the BLE protocol stack is depicted in Fig. 6 of Appendix B. *c)* The *Application* which defines the application that is utilising the Bluetooth Low Energy technology, in our case that application is our developed tool and in extension the Browser.

A more detailed overview of the main features of ATT and GATT is presented in the Appendix A as this analysis in not a main focus of our research but is essential to better understand the protocol that the Web Bluetooth API is based on.

Web Bluetooth API. The Web Bluetooth API [19] is a JavaScript technology which allows the communication of a Web Browser with a BLE device without the usage of any other intermediate technology. This technology, allows a Web Browser to manage the device's Bluetooth Module. Thus, the Browser is enabled to discover the nearby BLE device's GATT Services and Characteristics, read values from their Characteristics or enable their notification functionality, receive indications and notifications, and generally operate as a GATT *Client*. The Web Bluetooth API is integrated into the JavaScript libraries of the most popular Web Browsers (with the best compatibility existing with the Google Chrome browser).

Selected Device for Demonstration. The off-the-shelf device utilized for our research was the MetaMotionR sensor kit (MMR) [13]. The MMR encloses a variety of motion sensors such as 3-axial accelerometer, gyroscope and magnetometer sensors, and a variety of environmental ones such as temperature, barometric pressure and ambient light sensors. An API is offered as well, that enables sensor data recording, data transmission to a central device (over a BLE link) and sensor fusion capabilities. In particular, the sensors that our tool interacts with are the *accelerometer* and *gyroscope* sensors contained in Bosch's BMI160 IMU sensor [20], which recording range of values is $\pm 16\ g$ and $\pm 2000^{\circ}/s$ respectively at sampling frequencies of $6400Hz$ and $1600Hz$. The sampling frequencies can be filtered to the desirable output data rate through configuration changes by the developer. The sensor's support of the BLE connectivity as well its highly configurable accelerometer and gyroscope sensors are the reasons that the specific device is selected. As mentioned before, the use of this device is only indicative, since our tool can be adapted to work with any similar device that utilizes the BLE protocol.

V. The User Profiling Handling Component

In this section, the component's functionalities are outlined. In particular, it offers full user management and user profiling capabilities. Every potential user is required to create an account to utilize the tool which maintains a high level of privacy. Additionally, every user matches her own user profile that contains integral information that can later on assist processes by the rest of our tool.

User Profile Creation and Sign-In/Register/Authentication. The main interface of our tool is only accessible by users who have successfully concluded a registration procedure. During registration, a potential user is asked to provide a variety of non-sensitive personalized information about herself regarding her physiology (i.e. weight, height) or demographic information (i.e. gender, age). Hence, users' profiles are created that assist in the demographic analysis of the participants and the more personalized filtration of the sensory data acquired through the tool. The selection of the data which is asked to be provided is use-case specific and highly depends on the intentions of the application that the data collection tool is integrated to but it can be adopted to any use-case with minor configurations. Every user is first greeted with an appropriate Web page which contains the corresponding widgets for the user to perform a login or a registration to the application. During these procedures, an authentication mechanism handles whether the specific request will be fulfilled or not.

Privacy. As always, applications that involve a multitude of users and require their provision of any kind of personal information are potentially intrusive. Thus, encryption tech-

niques in user's personal information, such as their passwords, are also implemented. Depending on the context of the application this tool is used for, the users might be asked to provide some sensitive information. In those cases, an extra step during registration is required by the user in order to successfully register to the tool which is agreeing to a Terms of Service and Conditions along with an EULA (End-User License Agreement) which describe in detail the functionality of the tool and data gathering, data sharing and data usage practices (an example of such a case is the tool's deployment test case described in Section IX).

VI. The Communication and Data Management Component

This component is the most important of our proposed tool. Through it, the "magic" of connecting a Web Browser with a BLE device is enabled. Its main purpose is to remotely program the device for the sensors' initialization and data transmission. The data is encoded as BLE notifications, thus, a decoding process is applied. While this component is part of the front-end of our application, it is important to mention that the code is executed on the Browser and can be seen by everyone, the method of obfuscation was applied for the reason of a security measure in order to protect the code visibility and access by potential malicious users.

A. Connection to the BLE device

Through this component, the connection to the BLE device is established. By selecting the appropriate widget from our main interface, a potential user will be prompted to select the device she wishes to connect to, as shown in Fig. 7 of Appendix B. The available devices which are depicted in Fig. 7, are filtered by name (the only devices shown are the ones that match a specific name), in order to only show the ones that are compatible with the tool.

A communication between a Browser and a BLE device follows a specific set of operations. Essentially, a Browser must navigate through the Server's GATT hierarchy with the assistance of the Web Bluetooth API, meaning, the GATT Profile, Services and Characteristics. These operations are based on the asynchronous event feature of JavaScript which is named *promise*, where a sequential procedure is applied for the *Client* to reach and discover all the levels of the *Server's* GATT hierarchy. In order to discover and connect to the desired GATT Service and its Characteristics, their names or 128-bit addresses must be known and provided.

Initially, the search for the BLE device is taking place. Upon success, a connection attempt to the device's GATT server occurs. In case this is successful, a connection to the Primary Service with a particular address that we provide is attempted. Then, in the case this concludes successfully as well, an attempt to acquire the Characteristics that the Primary Service contains takes place again, explicitly, according to the Characteristics' names or addresses. At any point, if a promise is rejected, the connection to the device fails. The user is being informed for every step of the process through appropriate notification messages in the main interface.

Analyzing the device's GATT architecture. As described before, the connection procedure requires the knowledge of the addresses or names of the Primary Service and its Characteristics which are intended for the connection to be established. Hence, a methodology must be applied to define the device's architecture.

Bluetooth specification has a group of predefined Services and Characteristics which standardize selected functionalities throughout BLE devices and are already appointed with specific names and addresses. However manufacturers often develop custom Services and Characteristics, as the predefined only cover a limited amount of functionalities, turning the interaction with them into a challenging task, especially because of the lack of proper guidance and documentation.

As the device manufacturer has implemented her own Services and Characteristics, the device's architecture must be analyzed to identify their respective custom addresses. The application "nRF Connect" [21] was utilized for these purposes. As shown in Fig. 8 of Appendix B, we concluded that the device contains a custom Primary Service which consists of two custom Characteristics (usually are named as Unknown due to the fact the no official name is appointed to them). By integrating the detected addresses discovered through the application, the connection to the BLE device can be finally completed.

Defining the Command and Notification Characteristics. In order to derive the exact role of each of the detected Characteristics, a BLE sniffer device [22] along with the "Wireshark" software was used to monitor the connection between the manufacturer's official application and the BLE module. By analyzing the transmitted packets between the device and the application, the purpose of each value configuration was determined. It was concluded that with the first of the two of the Unknown Characteristics found, as depicted in Fig. 8 of Appendix B, device setup is implemented, whereas the second, is responsible for transmitting the sensor readings to the Client device through notifications. Thus, the nomenclature of the two Unknown characteristics as the Command Characteristic and the Notification Characteristic respectively were appointed.

Sensor's initialization. Writing a specific set of commands to the BLE module's Command Characteristic enables the programming of the device to initiate and start a data stream from the desired sensors. The user also has the ability to initialize the desirable sensors with the optative sampling frequencies through a selection of widgets that exist in the main interface. Upon successful connection to the device and initialization by the user, the appropriate commands are written to the device via a function that performs a series of Web Bluetooth API's *writeValue* commands to the Command Characteristic. Every value that is about to be written on a Characteristic must be in a byte array format. The commands must be written with a specific order to the command Characteristic otherwise the sensor's initialization will not occur.

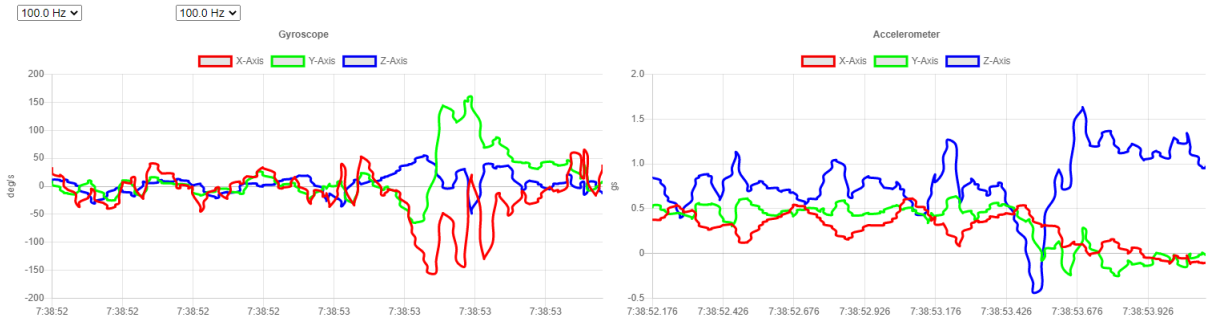

Fig. 2. An overview of the time-series plot visualized in the user's main page of the application during a sampling session. Both sensors are at $100Hz$ sampling rate.

B. Decoding notifications

After the successful initialization of the device's sensors, the Notification Characteristic transmits raw data readings through a stream of notifications. Each new one sensor's reading corresponds to a new notification. All values from Notifications consist always of array buffers (raw data values encoded in hexadecimal format). These values follow various encoding methods with one of the most common one being the IEEE-754 standard. The encoding methods though, vary amongst sensors and manufactures, while many times custom ones are utilized. Hence, a decoding mechanism is applied every time a new notification is transmitted. Through our research, we discovered that these decoding processes do not bottleneck the performance of the tool as these kinds of conversions occur rather quickly in the JavaScript environment while no considerable resources are wasted. Thus, despite the very high sampling rate of the accelerometer and gyroscope sensors, no values are lost during the decoding process.

C. Dashboard and Data Visualization

The main interface offers a variety of functionalities to a potential user. The user can select the sensors she desires to initiate the recording from and at which sampling frequency via appropriate widgets that exists in the main interface as depicted in Fig. 9 of Appendix B. The gyroscope and the accelerometer sensors are pre-configured to operate at their highest recording range of values which are $\pm 16\ g$ and $\pm 2000°/s$ respectively.

With every selection, the appropriate time-series plots are visualized in the main interface as depicted in Fig. 2. The data visualization library that is used for the plotting functionality is *Chart.js* [23]. The decoded sensor readings are visualized to these time-series plots dynamically, meaning that as time moves forward, the new values are dynamically visualized while the older ones are being discarded. Additionally, the y-axis of the plots also dynamically interacts according to the incoming values. The aforementioned, enhance the real-time monitoring experience of the user's motions. Finally, important to note is that once the sensors are enabled no further configuration changes can be sent to the device by the device's design. This ensures that the recording session will have the desirable settings and even in the case that it does not (wrong labeling or undesired configuration), it can be easily identified.

VII. The Real-Time Storage Component

The purpose of the component is to manage the storing of the recorded decoded values to a centralized data warehouse for further processing by an external application.

After the sensor's initialization phase, the user can start and stop the storing session through the corresponding widget in the main interface, at any point. The data which is transmitted by the device during this time-frame is collected into batches. Every batch is temporarily stored in our application's front-end and once a threshold of the values that it contains is reached, it is then being sent to the back-end (NodeJS server) for storing to the central data warehouse. The batches are identified by the same identification (session_ID) as long as they are produced during the same session (between a start and stop selection by the user) and also carry the timestamp that they were created. The batches mechanism was implemented in order to improve the overall performance of the storing functionality as storing sensors' values individually to the database would create a sizeable overhead. Before the initiation of a storing session, the user is also capable of labeling the data, according to the remote supervision of an expert stakeholder by typing the label into the corresponding widget. That supervision occurs during the whole time frame that the session lasts to ensure that the movements performed during the session correspond to the desirable ones.

Finally, aside from the sensory data, a variety of other information related to the session as well as the identification of the user performing it are also added to the batch. As a result, this additional data may expedite post collection processes and data analysis by the corresponding stakeholders. Each batch matches a new document in the MongoDB database as it is presented in Fig. 10 of Appendix B.

VIII. The Remote Monitoring Component

As mentioned earlier, this component is responsible for the user's device decoded sensory data forwarding to the expert's dashboard and handles the communication transactions between the two clients. The expert's dashboard is similar in format and functionality to the user's main interface. Upon

user's selection (via the corresponding widget on the user's main interface), a user enables a remote monitoring functionality and provides access to the data recorded by her device to an expert. The data is forwarded from the user's client to the expert's client through a WebSocket connection, which is specifically established for each user, and are visualized to the expert's dashboard through time-series plots in almost real-time. Consequently, the expert has the ability to initiate the storing session as well as annotate it utilizing the respective components that offer similar functionalities to a user, which were described earlier.

IX. Deployment and Validation

The proposed tool is already being integrated into our Machine Learning platform that aims to classify Allergic Rhinitis gestures with a gesture recognition approach using wrist-worn devices [12]. As mentioned in Section II, the methodology of acquiring the data for the creation of our gestures datasets was not optimal. The recent pandemic outbreak has also elevated the need for an efficient and easily usable tool that allows the proper data collection and annotation, even remotely with the appropriate expert supervision. These reasons manifest a highly suitable scenario for our tool.

An overview of the deployed tool's architecture is highlighted in Fig. 3. During registration, a user profile for the allergic rhinitis patient is created. Patient profiles provide demographic and allergic condition information about the patients which further assist personalized solutions. This only occurs if the user has agreed to the EULA and Terms of Services and Conditions. An indicative patient profile is shown in Table I of Appendix B. Through the main interface of the application, the user initializes the gyroscope and accelerometer sensors of the compatible device while she can monitor her motion in real-time. While the transmission of the data is still active, a real-time storing capability is offered in the form of a session as well as the simultaneous annotation of this session. This procedure can be remotely monitored, through any telecommunication platform of choice, by an expert physician in order to be executed appropriately and accurately. A text-based manual is also provided for the user through the main interface delivering appropriate guidance for the application usage and explaining which labels correspond to the specific gestures, thus, providing further assistance in the correct annotation process. Physicians also have the ability to directly monitor the session and even annotate it, without any user input, through their dashboard. The annotated sensory data along with additional metadata, such as the time of the session streamed and stored to the platform's centralized data warehouse and the identification of the patient participating in the procedure, offer immediate accessibility by corresponding stakeholders for further processing.

Finally, it is worth mentioning that the data collection process was immediately expedited by the integration of the tool to our AI platform. Consequently, our tool enables a procedure of data acquisition that no longer requires a controlled environment where the patient and the doctor must

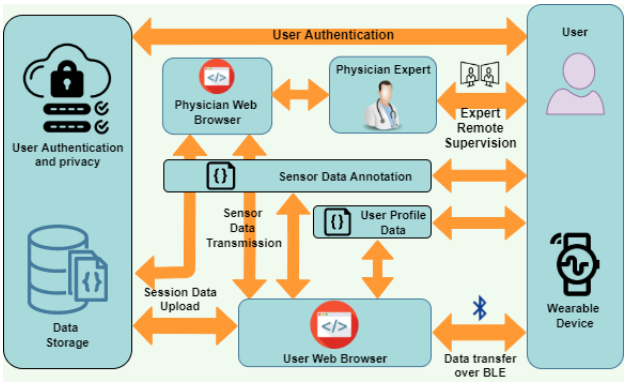

Fig. 3. An overview of the Deployment Architecture.

be present. Last but not least, our tool is accessible by all the smart-devices that integrate a Web Browser, thus minimizing the need for specific hardware from the patients.

X. Performance Evaluation

A sample rate test was conducted to compare our tool's performance with that of a native application's. The maximum theoretical number of samples for a 10-second recording session with both accelerometer and gyroscope sensors enabled at a 100Hz recording frequency, is 2000 samples (a 1000 samples respectively). This configuration happens to be the maximum output of the device by design (when these two sensors are enabled), thus, no comparison was performed with a higher throughput. As depicted in Fig. 4, both our tool utilizing the Web Bluetooth API and the native application can match it. Specifically, as it is shown in Fig. 4, the sample rate of our tool is identical to the theoretical maximum. The native application's sample rate is only slightly lower because between the logging screen and the initial configuration screen, exists a confirmation step during which the actual samples are recorded and logged in a CSV file but they are not included to the number of samples reported by the application. This can be confirmed by the fact that the number of samples logged in the CSV file produced by the application, matches the theoretical maximum as well.

From the above it is concluded that there are no performance differences between our developed tool and a native application. The sensor's battery consumption and the compute resources dedicated to the application or the tool, are also in similar levels. In addition, the tool's visualization of the sensory data is noticeably more responsive than that of the native application's while also the fact that no data are lost during storage, enhances our tool's viability as a respectable solution for these purposes, offering additional functionalities without any significant trade-offs.

XI. Conclusion and Future Work

In this paper, we present a novel Web Application that has the ability to effectively connect a BLE module to a smartphone, a tablet or a personal computer with the assistance of the Web Bluetooth API with no performance trade-offs when compared to a native application. To the best of our

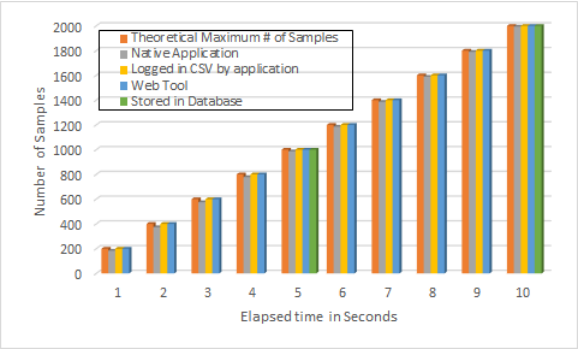

Fig. 4. Sample rate comparison between the native application and our Web-based tool. Samples are stored in the database in batches of a 1000 samples.

knowledge this is the first tool that utilizes this new technology and offers the ability to efficiently create and annotate datasets, based on data recorded from BLE enabled IoT devices even remotely, in a multi-client and human-centered environment. Finally, its successful deployment and use in a real world scenario through the integration to our existing AI platform for gesture recognition related to allergy detection [12], proved that it is a quite beneficial tool for data collection usages.

An additional enhancement to our tool would be the development of a functionality which allows the user or the expert to mark and then cut only the desirable parts of the movement which better represent the movement she wishes to save. The aforementioned will enable the creation of even clearer and more reliable datasets that only contain the desirable movement, reducing that way the time consumed by time-series data segmentation produced from a session. Another future step in our work would be our tool to support additional wrist-worn BLE modules from different manufacturers or even modules that are being worn in different parts of the body. In this way, data could potentially be collected from a variety of different sensors offering a more complete representation of a user's activity, health, and physiology. With the proper integration into future AI assisted or remote monitoring eHealth and mHealth systems, our proposed tool could become a valuable addition to the data collection practices that will be needed, therefore allowing rapid development and deployment processes of such systems while also enhancing their robustness.

Furthermore, another major enhancement would be to have this tool implemented as a back-end, alleviating the need of obfuscating the code execution that, for the moment, is being run on the front-end. Last but not least, the tool can be adapted to work in various fields beyond the scope of Healthcare, in systems that harness and are in need of movement data, with only minor configuration changes. For example, by simply changing the required profile information that is provided by the users and the labeling process (e.g. monitoring the movements of an animal) can create a whole new aspect for our tool. Finally, adding support for a new device would be relatively easy as the process is well documented and researched as outlined in the previous sections of our paper.

ACKNOWLEDGMENT

This research has been co-financed by the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH – CREATE – INNOVATE (project code: T1EDK 02436, project name: Personal Allergy Tracer).

REFERENCES

[1] M. Kim, J. Cho, S. Lee, and Y. Jung, "Imu sensor-based hand gesture recognition for human-machine interfaces," *Sensors (Switzerland)*, vol. 19, no. 18, pp. 1–13, 2019.
[2] G. Aceto, V. Persico, and A. Pescapé, "Industry 4.0 and health: Internet of things, big data, and cloud computing for healthcare 4.0," *Journal of Industrial Information Integration*, vol. 18, p. 100129, 2020.
[3] J. Qi, P. Yang, Z. Deng, Y. Zhao, A. Waraich, and Y. Yang, "Examining sensor-based physical activity recognition and monitoring for healthcare using internet of things: A systematic review," *Journal of Biomedical Informatics*, vol. 87, 08 2018.
[4] B. Fyntanidou, M. Zouka, A. Fourlis, A. Apostolopoulou, P. Bamidis D, A. Billis, K. Mitsopoulos, and P. Angelidis, "IoT-based smart triage of Covid-19 suspicious cases in the Emergency Department," in *2020 IEEE Global Communications Conference (GLOBECOM)*, 2020, pp. 8–13.
[5] "Shimer Sensing," http://www.shimmersensing.com/, (accessed 2021).
[6] "Mbientlab," https://mbientlab.com/, (accessed 2021).
[7] "MetaBase Application," https://mbientlab.com/tutorials/MetaBaseApp.html, (accessed 2021).
[8] "Android API," https://developer.android.com/guide/topics/sensors, (accessed 2021).
[9] J. P. Espada, V. García-Díaz, R. Crespo, O. Martínez, B. C. P. García-Bustelo, and J. M. C. Lovelle, "Using extended web technologies to develop bluetooth multi-platform mobile applications for interact with smart things," *Inf. Fusion*, vol. 21, pp. 30–41, 2015.
[10] J. Yasskin, "Web bluetooth community group charter." [Online]. Available: https://www.w3.org/community/web-bluetooth/web-bluetooth-community-group-charter/
[11] J. Wåhslén and T. Lindh, "A javascript web framework for rapid development of applications in iot systems for ehealth," in *2018 IEEE 20th International Conference on e-Health Networking, Applications and Services (Healthcom)*, 2018, pp. 1–6.
[12] X. Aggelides, A. Bardoutsos, S. Nikoletseas, N. Papadopoulos, C. Raptopoulos, and P. Tzamalis, "A gesture recognition approach to classifying allergic rhinitis gestures using wrist-worn devices : a multidisciplinary case study," in *2020 16th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2020, pp. 1–10.
[13] "MetaMotionR sensor kit," https://mbientlab.com/metamotionr/, (accessed 2021).
[14] "Socket.io," https://socket.io/, (accessed 2021).
[15] "MongooseJS ODM library," https://mongoosejs.com/, (accessed 2021).
[16] "Bluetooth Low Energy," https://www.bluetooth.com/learn-about-bluetooth/radio-versions/, (accessed 2021).
[17] C. Gomez, J. Oller Bosch, and J. Paradells, "Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology," *Sensors (Basel, Switzerland)*, vol. 12, pp. 11 734–53, 12 2012.
[18] M. Afaneh, *Intro to Bluetooth Low Energy: The Easiest Way to Learn BLE*. Amazon Digital Services LLC - KDP Print US, 2018. [Online]. Available: https://books.google.gr/books?id=0UhjvwEACAAJ
[19] "Web Bluetooth API," https://www.w3.org/community/web-bluetooth/, (accessed 2021).
[20] "BMI160 IMU," https://www.bosch-sensortec.com/products/motion-sensors/imus/bmi160.html, (accessed 2021).
[21] "nRF Connect application," https://www.nordicsemi.com/Software-and-tools/Development-Tools/nRF-Connect-for-mobile, (accessed 2021).
[22] "nRF sniffer for bluetooth LE," https://www.nordicsemi.com/Software-and-tools/Development-Tools/nRF-Sniffer-for-Bluetooth-LE, (accessed 2021).
[23] "Chart.js," https://www.chartjs.org/, (accessed 2021).
[24] "Bluetooth GATT Specifications," https://www.bluetooth.com/specifications/ gatt/, (accessed 2021).
[25] L. Leonardi, G. Patti, and L. Lo Bello, "Multi-hop real-time communications over bluetooth low energy industrial wireless mesh networks," *IEEE Access*, vol. PP, pp. 1–1, 05 2018.

APPENDIX A
ATT & GATT

The ATT defines the communication between two devices. In more detail, a device that maintains a set of attributes and their respective values, named the *Server*, is able to expose them by notifying or indicating to a peer device, named the *Client*. An attribute is a data structure that contains information handled by the protocol running on top of it, the GATT. The Server's exposed attributes can be accessed by the *Client* by discovering, reading, and writing to them.

As mentioned before, GATT is the Layer that is built on top of ATT and defines a framework that utilizes ATT and describes the way two BLE devices exchange packets. GATT includes a hierarchical data structure which (from the top level down) consists of the *Profile*, the *Services* and the *Characteristics*. A GATT Profile defines the way that the ATT is used to perform the various actions regarding the attributes contained in the server (discover, read, write, notify, indicate) and it consists of Services that are specific to a use case. Services are composed of Characteristics or references to other Services. A *Characteristic* consists of a set of data which includes a value (the value of the reading or the state of the functionality that the Characteristic corresponds to, for example the battery level of a device or the reading of an accelerometer sensor), properties (specific functionalities that describe if the Characteristic's value can be read, written etc.), and optionally a Client Characteristic Configuration Descriptor (CCCD) through which a *Client* device can configure a *Server's* Characteristic (enable/disable notifications). Services and Characteristics are identified by a name or an 128bit address. For example, a BLE *Server* device exposes a "battery service" that contains the Characteristic "battery level" that can be read and optionally notified [24]. A *Client* can read the battery level of the device via a request or enable notifications of the Characteristic by writing on its CCCD. An overview of a GATT profile's hierarchy is depicted in Fig. 5.

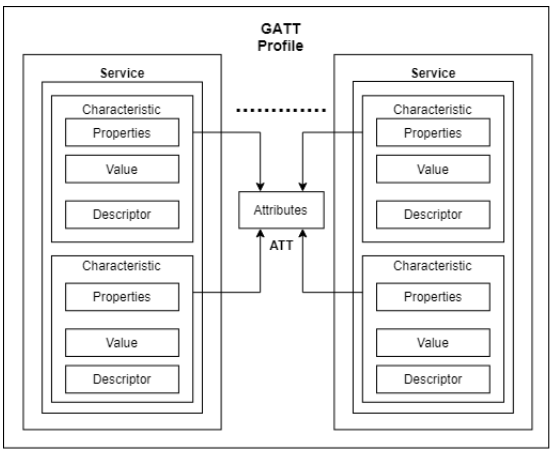

Fig. 5. The GATT profile's hierarchy.

APPENDIX B
IMAGES & TABLES

TABLE I
TABLE OUTLINING A PATIENT PROFILE

Fields	Possible Values
Name	Text
Last Name	Text
Email	Email
Password	Text / Symbols / Integers
Height	Integer
Weight	Integer
Dominant Hand	Left / Right
Age	Integer
Gender	Male/ Female
Family member that has history of Allergy	Father/Mother/Siblings
Allergic Rhinitis	Yes / No
Allergic Asthma	Yes / No
Allergic Conjuctivitis	Yes / No
How often my allergy affects me	Rarely / Sometimes / Often / Extremely often
Way of receiving Immunotherapy	Sublingual / Injection / Other / Not receiving
Date started receiving Immunotherapy	Date
Weeks of receiving Medicine	Integer
Allergens that affect me	Selection of allergens from the USA region
My symptoms	Selection of common allergen symptoms

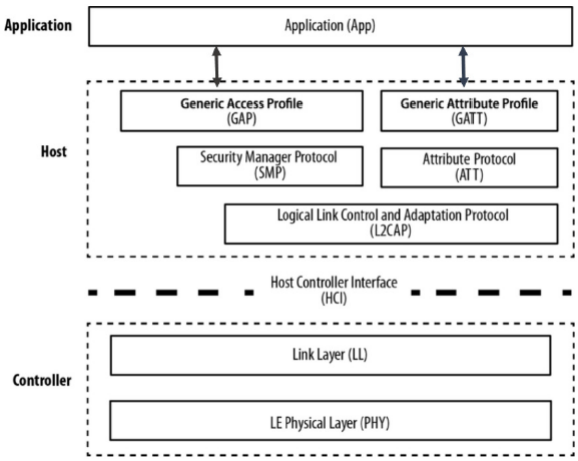

Fig. 6. An overview of the BLE protocol stack. Figure acquired from [25].

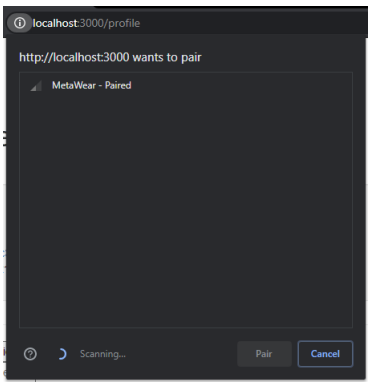

Fig. 7. The pop-up window showing the available Bluetooth Low Energy devices.

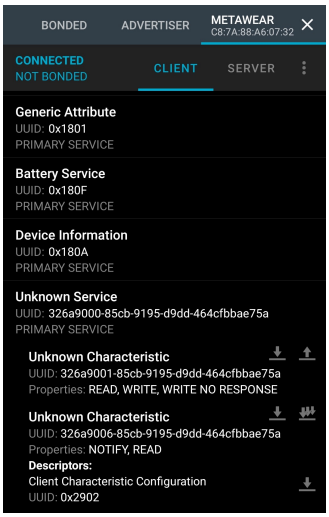

Fig. 8. The "nrf Connect" Application showing the custom Service and Characteristics of the selected module.

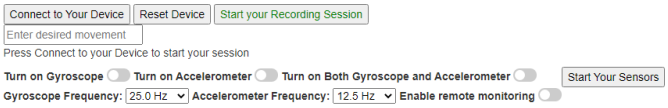

Fig. 9. Screenshot of the widgets the user can interact with to enable the device's sensors.

Fig. 10. Example of a document in the MongoDB database containing the sensory data.