2020-06

Bluetooth Performance Evaluation based on Notify for Real-time Body-Area Sensor Networks

Reich, O

http://hdl.handle.net/10026.1/17134

10.1109/metroind4.0iot48571.2020.9138199
2020 IEEE International Workshop on Metrology for Industry 4.0 & IoT
IEEE

Bluetooth Performance Evaluation based on Notify for Real-time Body-Area Sensor Networks

1st Olaf Reich, 2nd Erik Hübner, 3rd Bogdan Ghita, 4th Matthias Wagner, 5th Jörg Schäfer
Department of Computer Sciences and Engineering
Frankfurt University of Applied Sciences and University of Plymouth
Frankfurt am Main, Germany and Plymouth, United Kingdom
olaf.reich@fb2.fra-uas.de

***Abstract*—This work focuses on the analysis of Bluetooth Sensor Networks and the maximum number of Bluetooth sensors nodes for real-time human activity recognition (HAR). M:1 multi-pairing is used as network topology. The tests use notify for the Bluetooth data transmission. We investigated the quality of service (QoS) for the maximum number of Bluetooth connections with real-time sensor data based on inertial measurement units (IMU). We show how many Bluetooth sensor nodes can be simultaneously connected using different commercial off-the-shelf hardware and OS. On top of that we point out how the package loss behaves depending on the set time interval and used hardware. Our results show that a maximum number of 13 Bluetooth sensor nodes can be connected using an external Bluetooth dongle with the Linux OS. Connecting more than 10 Bluetooth sensor nodes at a frequency of 0.025 s tested in the *Thinkpad P53* in combination with the *DeLock Bluetooth dongle* being most useful. This combination allowed a maximum number of simultaneous connections by having only a package loss of about 1.2%.**

***Index Terms*—Bluetooth, sensor networks, wireless body area network (WBAN), quality of service (QoS), NINA, human activity recognition (HAR)**

I. Introduction

In the Internet of Things (IoT) and in Industry 4.0 the Bluetooth standard is a widely used communication protocol [1]. Personal devices like headsets, smartphones or smart homes are the most commonly usage for it. Additionally it is more and more applied in professional settings like smart factory or eHealth. Bluetooth is still actively being researched and developed by the Special Interest Group (SIG). The current clusters of research are real-world implementations, lab implementations and theoretical concepts [2]. The current Bluetooth specification has the version 5.2 [3] which includes features of earlier versions like the Bluetooth Low Energy (BLE) technology and mesh network capability. The introduction of these features yields new applications, possibilities and research fields for IoT [4]. An overview of the changes between the different Bluetooth versions can be found in the work of Yin et al. [2]. The standard defines as key features of Bluetooth technology its robustness, low power consumption and low cost [3]. These features make Bluetooth a promising candidate for the use of sensor networks in different research fields, like eHealth, position estimation and human activity recognition (HAR) to name a few. Regarding eHealth the Corona tracing apps are high potential Bluetooth candidates [5]. The quantity of information associated with the self-implementation of Bluetooth features in available hardware lead to challenges for the use of Bluetooth in real-time sensor networks. In the field of HAR for example, it can be important to have many Bluetooth sensors connected simultaneously and to transfer the data in a short measuring interval. Using BLE in practice, multiple factors can influence the performance. These factors are the operating system (OS), the Bluetooth library and the Bluetooth chip itself.

Furthermore it is important for the transmission to choose a proper connection mode from the Bluetooth standard. The Bluetooth version 5.0 allows to configure various types of procedures in order to retrieve values from connected devices [8]. One possibility is to configure indications which also require two steps. By using this procedure, the server inhabiting the desired characteristic value, sends an indication including the value to the connected client. It sends the indication to the client whenever its value changes. The client responds to the server with a confirmation. The usage of notifications as procedure is a another type. It is basically the same as described for indications but the server does not expect any confirmation from the client. Following, this procedure involves just one communication step from the server to the client. This is interesting for applications focusing on high performance. For example, the literature suggests a sampling rate of sensor data up to 50 Hz, indicating a period of 20 ms, to identify daily activities in HAR [16]. Thus, setting notifications as communication procedure to send data from a sensor to a computer can be a promising approach. In the present paper we first evaluate the maximum number of simultaneously wired connected sensors to a computer. On top of that, we point out how the package loss of the streamed data behaves based on different hardware and software configurations. We analysed a sampling data rate of 25 ms and 35 ms which lies in the suggested range of the literature. These results can be used to identify the quality of service of Bluetooth regarding high performance sensor readings as it is the case for example in HAR.

Furthermore we analyse the influence of various parameters such as using different computers including changing the Bluetooth module, the number of Bluetooth sensor nodes. Section II evaluates the related work. The configuration and setup of the experiment including the used hardware will be shown in Section III. Section IV concludes the paper.

II. Related Work

Current research focusses on the theoretical specifications, features and architecture of Bluetooth [2], [9] and [10]. Since the introduction of mesh topology in Bluetooth version 5.0, studies have focused on mesh networks and their performance optimization. For example, there is research on investigating real-time communication in a mesh network [11]. Additionally, some extensive research about tuning power consumption in order to increase lifetime of a battery from 9.55 days to 2.32 years [12] can be found. Furthermore, as seen in the work of Dian et al. [13] the important part of time synchronization in a mesh network between BLE devices also gets addressed. They figured out that poor environmental conditions can lead to divergence of more than 17 µs in time synchronization. Nevertheless there has been research regarding the number of simultaneous Bluetooth connections and performance analysis. As stated by Gomez et. al. [14] the theoretical maximum number of simultaneous connected Bluetooth devices is between 2 and 5917. Moreover there is a minimum of 676 µs latency to receive sensor data. A practical study already investigated the performance of Bluetooth in combination with various *iPhones* [15]. A maximum of 14 Bluetooth devices could be connected at the same time resulting in a reliable transmission rate of 0.04 s.
In the research field of HAR, certain suggestions on how to adjust the data acquisition can be found in literature. For example the sampling rate should range from 20 Hz [16] to 50 Hz [17] in order to detect activities of daily life. But the necessary sampling rate is affected by the activity which shall be detected. Miezal et al. [18] use 7 IMUs with a sampling rate of 60 Hz to recognise walking, running and jumping. Moreover, Gutiérrez-Madroñal et al. [19] analysed fall-detection by a sampling rate of 100 Hz with a combination of 4 IMUs. Scheurer et al. [20] investigated 16 different human activities using 1 IMU with a sampling rate of 30 Hz.

III. Method and Environment

A. Software

We used Python as programming language for the experiment. Python, a scripting language, is fairly popular for data analysis, data acquisition and simple data processing. The simple scripting language allows for easy and fast programming. Additionally, the test setup can be adjusted quickly. It is possible to execute Python scripts on different platforms like MacOS or Linux. This facilitates the cross platform development resulting in a good comparison between different platforms. Nevertheless we had to use two distinct platform specific Bluetooth libraries, bluepy for Linux and Corebluetooth for MacOS.

B. Hardware

For the tests we varied the used hardware in order to get the best performing combination. As seen in table I the hardware on the receiver side consisted of different *MacBooks*, an *Intel NUC 8* and a *ThinkPad P53*. As the *MacBooks* and the *Intel NUC 8* have soldered Bluetooth modules we tried them as out of the shelve solutions. As the Bluetooth module of the *ThinkPad P53* was not soldered, we equipped the computer with various Bluetooth modules as seen in table II. This could reveal any differences in performance based on the hardware combination.

TABLE I
Overview of used computers with soldered Bluetooth moduls

Name	**Bluetooth version**	**Bluetooth module**	**OS**	**Bluetooth software version**
Intel NUC 8	4.2	Intel Wireless AC 8265	Ubuntu 18.04.3 LTS	-
MacBook Pro 2019	5.0	Broadcom 4364B3	Catalina 10.15.2	7.0.2f4
MacBook Pro 2018	5.0	Broadcom 4364B0	Catalina 10.15.2	7.0.2f4
MacBook Pro 2014	4.0	Broadcom 20702B0	Mojave 10.14.6	6.014d3
MacBook Air 2013	4.0	Broadcom 20702B0	Catalina 10.15.2	7.0.2f4

TABLE II
Overview of used commercial off-the-shelf Bluetooth moduls

Name	**Bluetooth version**	**Module**
Killer Wi-Fi 6 AX1650	5.0	Intel AX200
Killer Wireless-AC 1550	5.0	Intel 9260NGW
Intel AX200NGW	5.0	Intel AX200NGW
Intel AC 8265	4.2	Intel 8265
Intel 8260NGW	4.0	Intel 8260NGW
DeLock USB2.0 Bluetooth Adapter	4.0	Broadcom BCM20702A0

On the transmitter side we used 20 *Arduino Nano 33 BLE Sense* as sensor nodes. The devices are equipped with an ARM CPU, which operates with a frequency of 64 MHz and a working memory of 256 kB. In addition, the sensor nodes can be linked to a computer via an UART interface. We used the *Arduino Nano 33 BLE Sense*, since it characterises a typical Bluetooth sensor node for HAR. This means that there are several sensors on the boards which can be used for data generation. These include a typical 9 axis inertial sensor [21], barometric pressure [22] and temperature sensor [23]. The exact specifications of the sensors can be found in the respective data sheet. Although the *Ardunio Nano 33 BLE Sense* has additional sensors like microphone, gesture light and proximity we did not make use of them with regard to the HAR. The focus was on the Inertial Measurement Unit (IMU) data. The generated data can be packed in 48 bytes packets, representing the size of typical sensor readings, followed by sending them via Bluetooth to a client as it would also happen in a real application. A NINA-B306 chip is installed for Bluetooth communication. The chip uses version 5.0 and supports BLE. The NINA-B306 has an internal antenna and supports a maximum of 20 Bluetooth connections [24]. Furthermore, the sensor node can be battery operated which is necessary for HAR.

C. Configuration and Setup

Setup and experiment are divided into two steps. In the first step, called pretest, the maximum quality of service regarding data transmission via cable is evaluated. In the second step, called main-test, the quality of service regarding data transmission via BLE is evaluated. Until now, there has not been any effort done in improving the performance of the system by adjusting different Bluetooth specific parameters. This includes for example setting the connecting interval.

Pretest: For this purpose one Bluetooth sensor at a time was wired to a computer via the UART interface. Then, the Bluetooth sensor measured the sensor values consecutively at different given rates ranging from 0.015 s to 0.035 s. The Bluetooth sensor started its measurement automatically. After each measurement, the Bluetooth sensor transmitted the values to the computer. After 300 s the Bluetooth sensor automatically stopped measuring and sent a stop signal to the computer. To exclude any difference in performance across the Bluetooth sensors, the test was executed for 16 Bluetooth modules of the same type. To preclude an effect of consecutive data transmission based on the order, one Bluetooth sensor node was picked and the test was repeated in a modified version. Now, each transmission rate was recorded separately. Additionally, the size of the payload was changed from 44 bytes to 48 bytes, which simulates the size of realistic sensor measurements. The data was recorded at the computer using a Python script. To avoid any latencies, the data was successive written to a list, followed by writing to a file after the experiment was over. The difference between the expected amount of received packets, based on the duration of the experiment and transmission time, can then be compared to the real amount of received packages.

Main-test: The Wi-Fi of all used computers was deactivated during the experiment while only the Bluetooth interface was switched on. All other connections and applications have been terminated via the operating system. As in real use, the Bluetooth sensor nodes were operated with 9 Volt block batteries. The Bluetooth sensor nodes were positioned in a row opposite to the respective computer as shown in figure 2. There were no barriers between the computer and the Bluetooth sensor nodes. This means that the Line of Sight (LoS) was free.
The main-test was split in two sections. In the first section we discovered the maximum possible number of connected Bluetooth sensors which the computer could handle. In the second scenario, we evaluated the quality of service regarding data transmission via Bluetooth from the Bluetooth sensor nodes to the respective computer. Therefore, the maximum number of Bluetooth sensors was connected to the computer and the transmission rate was varied between 0.025 s and 0.035 s. This was repeated for all computers in table I. Each packet which the Bluetooth sensors sent to the computer had a size of 48 bytes for simulating realistic sensor measurements. The data was recorded at the computer using a Python script following the flow of control which can be seen in the activity diagram in figure 1. At the beginning the Bluetooth sensors were connected to the computer. After that, the computer subscribes for notifications at each Bluetooth sensor. In order to start the experiment, a command is sent from the computer to each Bluetooth sensor. From this time on each Bluetooth sensor sends packets to the computer for 300 s. After 300 s each Bluetooth sensor sends a stop signal to the computer. To avoid any latencies on the computer side, the data is successively written to a list. After the Python script received the stop signals from all Bluetooth sensors the list is written to a SQLite database to save the data persistently. The difference between the expected amount of received packets, based on the duration of the experiment and transmission time can then be compared to the real amount of received packages.

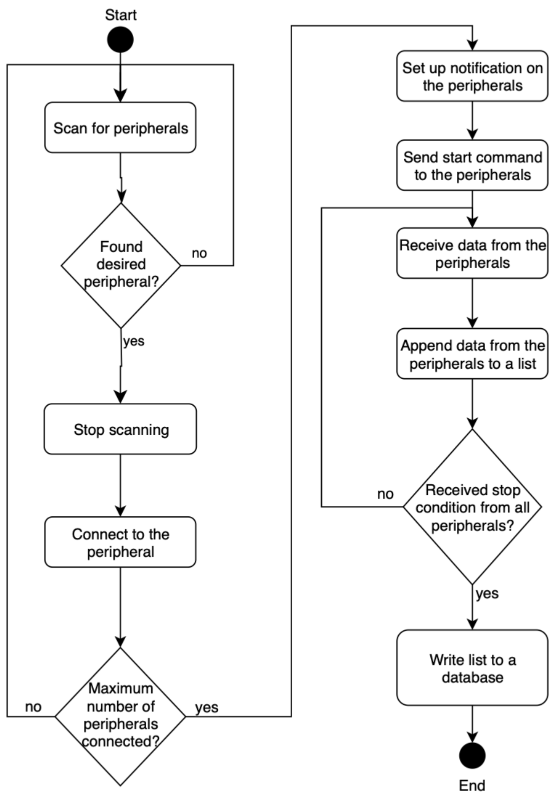

Fig. 1. The flow control of the program executed on the computers. *In the beginning, the computer connects to the Bluetooth sensor nodes. After all Bluetooth sensor nodes are connected, the computer sets up notifications on the Bluetooth sensor nodes, followed by a start command. The computer records the received data in a list until the stop signals of all Bluetooth sensor nodes arrived. The resulting list is then written to a database.*

TABLE III
COMPUTERS AND BLUETOOTH MODULES WITH THE RESULTING PACKAGE LOSS DEPENDING ON SETTED IMU NUMBER AND TRANSMISSION TIME.

Computer	Bluetooth module	IMUs	Transmission time [s]	Expected packets	Received packets	Package loss [%]
MacBook Air 2013	Broadcom 20702B0	10	0.025	120,000	77,064	35.78
MacBook Air 2013	Broadcom 20702B0	10	0.035	85,714	69,879	18.47
MacBook Pro 2014	Broadcom 20702B0	10	0.025	120,000	71,454	40.46
MacBook Pro 2014	Broadcom 20702B0	10	0.035	85,714	67998	20.67
MacBook Pro 2014	DeLock	10	0.035	85,714	67839	20.85
MacBook Pro 2018	Broadcom 4364B0	10	0.035	85,714	68568	20.00
Intel NUC 8	Intel 8265	10	0.025	120,000	119265	0.61
Intel NUC 8	Intel 8265	10	0.035	85,714	85,653	0.07
ThinkPad P53	DeLock	13	0.025	156,000	70,188	55.01
ThinkPad P53	DeLock	13	0.035	111,428	111,395	0.03
ThinkPad P53	DeLock	12	0.025	144,000	142,268	1.20
ThinkPad P53	DeLock	10	0.025	120,000	95,874	20.11
ThinkPad P53	DeLock	10	0.035	85,714	85,708	0.01
ThinkPad P53	Intel 8265	10	0.025	120,000	119,915	0.07
ThinkPad P53	Intel 8265	10	0.035	85,714	85710	0.00
ThinkPad P53	Killer 1550	10	0.025	120,000	116,434	2.97
ThinkPad P53	Killer 1550	10	0.035	85,714	85,525	0.22
ThinkPad P53	Killer 1650x	10	0.025	120,000	119,735	0.22
ThinkPad P53	Killer 1650x	10	0.035	85,714	85,714	0.01
ThinkPad P53	Intel AX200ngw	10	0.025	120,000	119,611	0.32
ThinkPad P53	Intel AX200ngw	10	0.035	85,714	85,552	0.19
ThinkPad P53	Intel 8260ngw	10	0.025	120,000	31,437	73.80
ThinkPad P53	Intel 8260ngw	10	0.035	85,714	31,427	63.34

Fig. 2. Test setup for the main-test: *In order to always have the same test conditions, the devices were positioned on a template (light brown). Multiple battery powered Bluetooth sensor nodes are equally positioned in a row. To avoid slipping between the tests, the nodes were attached to the template. Each space between two nodes was approximately 6 cm and the computer (not visible in the picture) was located about 100 cm from the nearest node. The space between the computer and the nodes was kept free.*

IV. RESULTS

The results can be divided into two parts. The first part evaluated the maximum quality of service of data transmission via cable. Only one Bluetooth sensor node at a time was tested. As expected all Bluetooth sensor nodes performed equally regarding the package loss. Recording the packages consecutively starting with 0.035 s, followed by 0.025 s and 0.015 s resulted in a maximum package loss of one packet at the transmission time of 0.025 s and 0.015 s. This results in a relative package loss of 0.005% when expecting 20,000 packets (0.015 s) and 0.008% when expecting 12,000 packets (0.025 s). Recording the results separately for each transmission time in a separate experiment and changing the payload size did not lead to a change in the results.

The second part reveals the maximum number of simultaneously connected Bluetooth sensor nodes. Afterwards it evaluates the quality of service of Bluetooth regarding the package loss during data transmission. Therefore, different sampling rates were tested. On the basis of the first part, in the next step we connected one Bluetooth sensor node via Bluetooth with the computer. The result indicated already a package loss of 2.42 % at a transmission time of 0.015 s. Changing the transmission time to 0.025 s decreased the package loss to 0.01%. Thus, we tested the package loss with a maximum number of connected Bluetooth sensor nodes and different hardware combinations at a transmission time of 0.025 s and 0.035 s. As seen in table III the maximum number, namely 13, of simultaneously connected Bluetooth sensor nodes could be achieved using the *ThinkPad P53* in combination with the *DeLock* Bluetooth module . In this setting, the package loss increased from 0.03% to 55.01% when changing the transmission time from 0.035 s to 0.025 s. Without using the *DeLock* Bluetooth module all combinations achieved a maximum of 10 simultaneously connected Bluetooth Sensor nodes. Looking at the MacBooks it is remarkable that even the lowest package loss lies at 18.47%. This was reached at a transmission time of 0.035 s using the MacBook Air 2013. Modifying the transmission time to 0.025 s led to a package loss of 35.78%. The comparison of the performance of the different used Bluetooth modules on the *ThinkPad P53* revealed interesting results. The *Intel 8625* Bluetooth module had the lowest overall package loss, namely 0.07% at a data transmission of 0.025 s and no notable package loss at 0.035 s.

V. Conclusion

We showed that a maximum of 13 Bluetooth sensor nodes can simultaneously be connected and provide real-time sensor data using a sampling rate of 0.035 s with an acceptable package loss of 0.03 %. This result was achieved with the constellation of a *ThinkPad P53* running an Ubuntu 18.04.3 LTS OS and the Bluetooth dongle *DeLock*. If a higher sampling rate is needed, our results show that a sampling rate of 0.025 s with 12 IMUs is working with a tolerable package loss of 1.2 % with the same setup. The notification is a good alternative to indication for high performance sensor networks. There are big differences in performance between Bluetooth modules.

Different requirements emerge depending on the human activity to be detected. In order to work out the limit under practical circumstances, we found the maximum number of simultaneously connected IMUs at the highest possible sampling rate. Based on our focus to use commercial off-the-shelf hardware and OS, the results reveal the settings and parameters which can be used. Nevertheless, further work needs to be done. It is recommended that deeper studies analyse the delay time between the signal acquisition and the signal transmission.

References

[1] K.-H. Chang, "Bluetooth: a viable solution for IoT? [industry perspectives]," IEEE Wireless Communications, vol. 21, no. 6, pp. 6–7, 2014.

[2] J. Yin, Z. Yang, H. Cao, T. Liu, Z. Zhou, and C. Wu, "A Survey on Bluetooth 5.0 and Mesh: New Milestones of IoT," ACM Transactions on Sensor Networks (TOSN), vol. 15, no. 3, p. 28, 2019.

[3] Special Interest Group (SIG), "Bluetooth Core Specification v 5.2," Bluetooth version 5.2 standard, Volume 0, December 2019.

[4] Special Interest Group (SIG), "Bluetooth Technology Website." https://www.bluetooth.com/. [Access] 2019-12-02.

[5] C. Troncoso et.al., "Decentralized Privacy-Preserving Proximity Tracing", https://github.com/DP-3T/documents/blob/master/DP3T%20White%20Paper.pdf. [Access] 2020-04-27.

[6] L. la Blunda, D. Corral-Plaza, M. Wagner, G. Ortiz and I. Medina-Bulo, "Distributed real-time based Human activity analysis System," Proceedings of the 16th International Conference on Applied Computing, Cagliari (Italy), November 2019.

[7] D. Corral-Plaza, O. Reich, E. Hübner, M. Wagner and I. Medina-Bulo, "A Sensor Fusion System Identifying Complex Events for Localisation Estimation," Proceedings of the 16th International Conference on Applied Computing, Cagliari (Italy), November 2019.

[8] Special Interest Group (SIG), "Bluetooth Core Specification v 5.0," Bluetooth version 5.0 standard, Volume 0, December 2016.

[9] M. Collata, G. Pau, T. Talty and O. k. Tonguz, "Bluetooth 5: A concrete step forward toward the IoT," IEEE Communications Magazine, vol. 56 no.7, pp 125-131, 2018.

[10] P. Ray and S. Agarwal, "Bluetooth 5 and Internet of Things: Potential and Architecture," International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES), pp. 1461-1465, 2016.

[11] L. Leonardi, G. Patti and L. L. Bello, "Multi-Hop Real-Time Communications Over Bluetooth Low Energy Industrial Wireless Mesh Networks," IEEE Access journal, vol. 6, pp. 26505-26519, 2018.

[12] A. Liendo, D. Morche, R. Guizzetti and F. Rousseau, "BLE Parameter Optimization for IoT Applications," IEEE International Conference on Communications (ICC), pp. 1-7, 2018.

[13] F. J. Dian, A. Yousefi and K. Somaratne, "Performance evaluation of time synchronization using current consumption pattern of BLE devices," IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), pp. 906-910, 2018.

[14] C. Gomez, J. Oller and J. Paradells, "Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology," Sensors journal, no 12, pp. 11734-11753, 29 August 2012.

[15] O. Reich, E. Hübner, B. Ghita, M. Wagner and J. Schäfer, "Performance Evaluation of Bluetooth in a Wireless Body Area Network for Practical Applications," 11th IEEE Sensor Array and Multichannel Signal Processing Workshop, 2020.

[16] I. Cleland, B. Kikhia, C. Nugent, A. Boytsov, J. Hallberg, K. Synnes, S. McClean and D. Finlay, "Optimal placement of accelerometers for the detection of everyday activities," Sensor journal, vol. 13, pp. 9183-9200, 2013.

[17] D. Curone, G. M. Bertolotti, A. Cristiani, E. L. Secco and G. Magenes, "A real-time and self-calibrating algorithm based on triaxial accelerometer signals for the detection of human posture and activity," IEEE transactions on information technology in biomedicine, vol. 14, pp. 1098-1105, 2010.

[18] M. Miezal, B. Taetz, and G. Bleser, "Real-time inertial lower body kinematics and ground contact estimation at anatomical foot points for agile human locomotion," in 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 3256–3263, IEEE, 2017.

[19] L. Gutiérrez-Madroñal, L. La Blunda, M. F. Wagner, and I. Medina-Bulo, "Test event generation for a fall-detection iot system," IEEE Internet of Things Journal, vol. 6, no. 4, pp. 6642–6651, 2019.

[20] S. Scheurer, S. Tedesco, K. N. Brown, and B. O'Flynn, "Human activity recognition for emergency first responders via body-worn inertial sensors," in 2017 IEEE 14th International Conference on Wearable and Implantable Body Sensor Networks (BSN), pp. 5–8, IEEE, 2017.

[21] STMicroelectronics, "iNEMO inertial module: 3D accelerometer, 3D gyroscope, 3D magnetometer," LSM9DS1 datasheet, Revision 3, March 2015.

[22] STMicroelectronics, "MEMS nano pressure sensor: 260-1260 hPa absolute digital output barometer," LPS22HB datasheet, Revision 6, June 2017.

[23] STMicroelectronics, "Capacitive digital sensor for relative humidity and temperature," HTS221 datasheet, Revision 4, August 2016.

[24] ublox, "NINA-B3 series: Stand-alone Bluetooth 5 low energy modules," UBX-17052099 datasheet, Revision R06, May 2019.