Design and Power Consumption Analysis of a NB-IoT End Device for Monitoring Applications

Sofia Paiva
DTx - Digital Transformation Colab
Campus de Azurém
Guimarães, Portugal
a78838@alunos.uminho.pt

Sérgio Branco
Department of Industrial Electronics
Algoritmi Center, University of Minho
Braga, Portugal
asergio.branco@gmail.com

Jorge Cabral
Department of Industrial Electronics
Algoritmi Center, University of Minho
Braga, Portugal
jcabral@dei.uminho.pt

***Abstract*—As the number of connected "things" increases at a very fast pace, the Internet of Things (IoT) ecosystem expands and nowadays covers a vast number of application domains, providing a large portfolio of solutions that are based on an evolving system, from the physical sensors (end devices) to the Cloud. When designing battery-powered end devices, previous research has identified several challenges such as wireless connectivity, battery lifetime, embedded intelligence, security and privacy concerns, and costs (modem unit, communication link and maintenance, among others). This paper focuses on the design and development of battery-powered IoT devices in which NarrowBand Internet of Things (NB-IoT) is used to provide seamless wireless connection, reduce power consumption, enhance communication coverage and minimize maintenance costs. The paper describes a typical use case where an Arm® Cortex®-M0+ and its low-power modes are exploited in order to design a low-power end device. Two different approaches, bare-metal and freeRTOS, for implementing the end device firmware are compared. Additionally, performance tests prove that increasing the clock frequency of the processor does not bring any advantage to this kind of applications.**

***Keywords*—IoT, low-power design, NB-IoT, power consumption**

I. Introduction

Nowadays Internet of Things (IoT) is a well known concept that is even being associated with specific application domains such as Industrial IoT (IIoT), Cellular IoT (CIoT), Internet of Medical Things (IoMT), among others. IoT applications are emerging at a fast pace, reflecting on the 26 billion of connected devices in 2019, a number that is predicted to grow by 178% in the next 5 years [1].

Some of these applications, such as remote utility metering and agriculture monitoring, due to safety restrictions or limited physical access, require a battery-powered end device. Thus, it is necessary to have special attention to power consumption, since it influences the device's autonomy and may have high impact on maintenance costs.

For long range coverage and low-power transmissions, the Low Power Wide Area Network (LPWAN) technologies are the right choice. There are two classes from two different spectrum: unlicensed spectrum technologies and licensed spectrum technologies. The former can operate within a Industrial, Scientific and Medical (ISM) frequency band and each operator must assure the infrastructure necessary to transmit the information (e.g. LoRa, Sigfox). The latter is supported by standards such the Long Term Evolution (LTE) and Global System for Mobile Communications (GSM), and by the current cellular infrastructure. There is even an effort to reduce hardware costs and to simplify network access mechanisms, targeting the embedding of the SIM card into the module hardware itself [2].

Still, the communication link is one of the most power consuming tasks of a wireless end device. On the licensed spectrum technologies, specially with NarrowBand Internet of Things (NB-IoT), the reduction of power consumption is achieved by reducing the physical layer of the communication link (when compared to the cellular physical layer), while maintaining the security, scalability and link reliability provided by the LTE network.

NB-IoT was introduced by the Third Generation Partnership Project (3GPP), being a LPWAN technology conceived to use the existent cellular network for low data rate, low-cost applications. It has three operation modes: in-band, guard-band, and standalone, regarding the transmission carrier and resource blocks. The mode used is determined by the service provider based on the cell site and base station that supports the end device. NB-IoT's advantages are [3]:

- Power optimization features such as Power Saving Mode (PSM) and extended idle-mode discontinuous reception (eDRX);
- Deep indoor coverage mainly because of its 180 kHz bandwidth;
- Low-cost NB-IoT radio modules (modem);
- LTE network security and scalability.

There are several research works on NB-IoT applications in various application domains, taking advantage of its low-power consumption and LTE network benefits [4]–[10]. However, the end devices designed used either Arm® Cortex®-M3 or M4 microcontrollers, or Arduino or Raspberry Pi boards as control units.

In this paper the design of an end device based on a Cortex®-M0+ for low-power, low-cost, and low-performance NB-IoT applications is presented, focusing on the impact of the different firmware coding approaches and exposing the various design choices towards a low-power end device.

978-1-7281-5414-5/20/$31.00 ©2020 IEEE

The remaining of this paper is structured as follows. §II focus on design principles for low-power NB-IoT applications. In §III the end device structure and its functionalities are described generally. Then there is an hardware description and on the software the bare-metal firmware is opposed to the RTOS-based one.

This paper concludes by presenting results from initial tests to the end device (in §IV) and identifying future research topics in §V.

II. System Design Principle

Currently there are two ways of integrating NB-IoT technology into an end device:

- using only the NB-IoT modem, programming its embedded application processor to establish the sensor's interface;
- using the NB-IoT modem exclusively for communication and add a low-power Microcontroller Unit (MCU) to run the application and establish interface with the end device sensors.

The first option for NB-IoT integration is still very limited and suitable solutions for custom low-power applications are difficult to find. In [11], the Qualcomm solution using this method was presented (MDM9206), which is supported by two different chip manufacturers (Quectel [12] and SIMCom [13]) and has integrated cloud device agents. The end device is based on a Cortex®-A7 application processor and runs ThreadX RTOS. Nordic Semiconductor also introduced a System-in-Package (nRF9160) as the integrated modem solution. The SiP has a Cortex®-M33 application processor and has its own development kit aiming to allow an easy and fast development [14]. Although this approach has clear advantages for fast prototyping, it has some disadvantages related to power consumption and data link costs, since more processing power is required from the module (one hardware solution fits all) and property cloud agents and web micro-services are required that leads to more maintenance and communication data link costs. For custom, low-power and minimal costs, neither of the above solutions seems flexible enough, so in this paper the second option was chosen. There are a lot more offers on the market for standalone NB-IoT modem solutions. This market is mainly dominated by Quectel. Other manufacturers include u-blox, Qualcomm, Nordic Semiconductors and SIMCom. Although the use of a standalone modem requires an additional microcontroller unit (MCU), it allows for more flexibility in terms of peripherals and MCU selection, allowing a faster custom design for development teams with a particular MCU skills. The communication with the NB-IoT modem is done via UART using standard AT commands, given by the modem manufacturers.

For the design here presented, the Quectel BC66 module was selected based on power consumption, accessibility and documentation. Targeting low-power, the MCU selected was the STM32L071K8 which uses a Cortex®-M0+. It is well suited for small monitoring applications that do not require intensive processing or high data rates.

Regarding the MCU's peripherals, there are some design considerations to have in mind when targeting development time, reduced Bill-of-Materials (BoM), and low-power, namely avoiding analog sensors. The ADC peripheral is one of the most power consuming ones and can jeopardize the battery lifetime if left continuously powered-on [15]. Even though low-power MCUs such as the STM32L0 series are prepared to reduce this power consumption to a minimum, it might still be preferred to choose a sensor with digital output like I2C or SPI, since it can help reduce the sensor's energy consumption as well as offering other functionalities, such as an interrupt signal when the sensing value crosses a given threshold. In order to respect this design choice, all sensors chosen for the demonstration system have I2C interface, allowing the use of a single I2C peripheral to interface all the end device integrated sensors.

On MCU configuration there is also a relevant aspect to consider: the MCU system clock. It is well known that higher clock frequencies result in higher power consumption, but faster execution. On the contrary, by slowing the clock frequency, power consumption is reduced at the cost of slower execution. With the system design here exposed, it is the authors intention to test if, for a *wake-up – execute – sleep – repeat* application, it is better to execute fast, sleep more or execute slow, sleep less in terms of overall power consumption.

Another design constraint to have in mind is the approach to develop the end device firmware. There are two approaches for developing the code in the end device MCU (abstractly speaking):

- Bare-metal approach – where the code runs directly on top of the MCU, respecting the coding flow;
- RTOS-based approach – where there is a scheduling mechanism that allows tasks to run in pseudo-parallelism.

The first allows for a more controlled application flow and optimization, at the cost of being inflexible when it comes to design changes. The second allows for a more flexible application and theoretical better use of resources, by introducing pseudo-parallelism into the execution. However, it adds a considerable layer of overhead and may not be advantageous for an application as simple as the one considered.

In order to get to the bottom of this question, two firmware applications were developed for the end device design, one as a bare-metal application and the other as a RTOS-based application.

Lastly, it is equally important to choose the communication protocol. Since the NB-IoT standard supports either Internet Protocol (IP) or non-IP protocols, the choice of a communication protocol impacts both power consumption and data security/reliability.

Non-IP protocols can be SMS or other NB-IoT Non-IP Data Delivery (NIDD). User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) are the supported communication protocols over IP communication. For an IoT application, the UDP protocol assures more power savings in comparison to TCP, but it can not guarantee the message delivery.

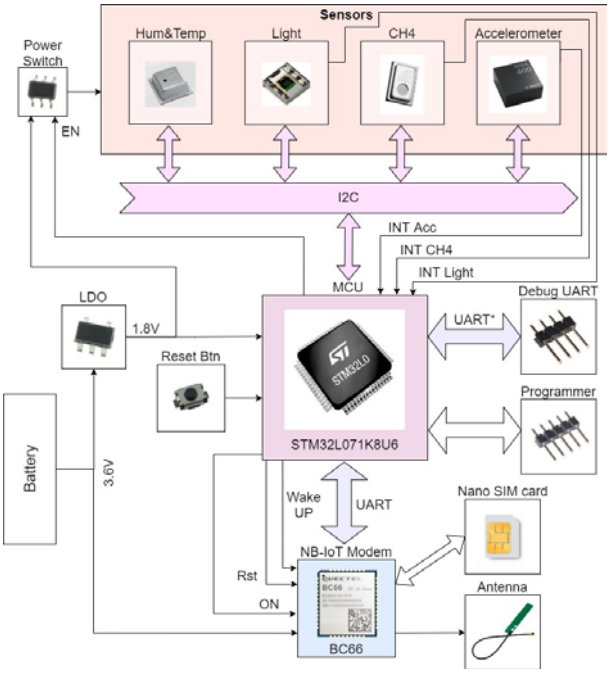

Fig. 1. System Diagram

On the application level, there is also compatibility with the Constrained Application Protocol (CoAP) and Message Queuing Telemetry Transport (MQTT) standards for Machine-to-Machine (M2M) communications. The former can be deployed seamlessly either with UDP transport layer or NIDD, and is being recommended as the preferable standard to use with NB-IoT [16]–[18].

III. System Structure and Function

The proposed end device is represented by the diagram in figure 1. The end device conceived for working with 1.8V sensors and MCU, has a NB-IoT modem that allows the direct connection to a 3.6V battery.

The network architecture presented in figure 2 illustrates the process of a simple application for monitoring some environmental conditions with a NB-IoT end device, pre-process the data and securely send the information to a Cloud server, so that it can be available on access platforms.

Due to the fact that the data collected from the nodes is being sent to a Cloud, it is essential to ensure that the communication protocol provides security & privacy measures. Moreover, communication protocols must create the minimum overhead possible to ensure a reduced transmission time.

The ArchNet [19] protocol uses a symmetric-asymmetric encryption process to ensure that the data is not readable by any third-party. The data serialization step ensures that the data is understandable by any programming language and program. Furthermore, the use of decentralized microservices and database (MongoDB), allows a more secure and efficient way of storing data and process it. The communication NB-IoT device – base station is guaranteed by the NB-IoT standard which can use features from LTE if not working in standalone.

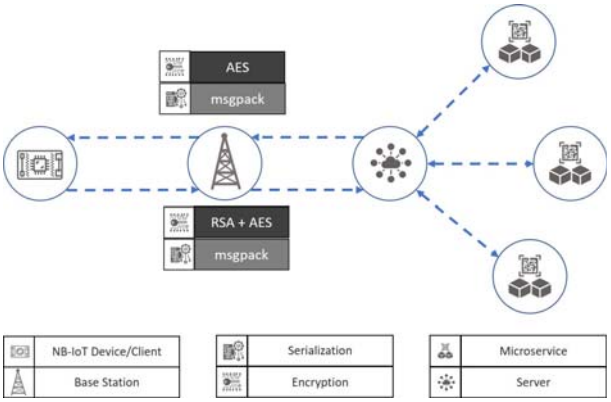

Fig. 2. NB-IoT Network Architecture

Then, from the base station to the server the communication is done through the internet, using a IPSEC Tunnel to assure data confidentiality, scalability and security.

A. Hardware Design

The end device has three main components: the MCU; the NB-IoT modem; and the sensors. The MCU is a STM32L071K8U6 (32 MHz Arm® Cortex®-M0+, 64 KB Flash, and 20 KB RAM), chosen by its small footprint and reduced number of peripherals, making it almost custom for the proposed use case. It controls the whole system, gathering the sensors' data, pre-processing it and communicating the monitored information through the NB-IoT modem.

The NB-IoT modem selected was the Quectel BC66 that has a power supply range between 2.1V and 3.63V, allowing direct connection to the battery. Additional hardware for the NB-IoT modem includes a nano SIM card (since eSIM is not yet available for deployment) and an antenna.

Regarding the sensors, four different sensors for humidity and temperature, light/luminosity, methane gas (CH_4), and acceleration were selected. All sensors have low-power consumption, low-power modes, and digital communication interface (I2C). Some also have programmable interrupt signals that allow to detect anomalies on monitored data without requiring constant reading requests and processing from the end device MCU.

For powering it all up, a 3.6V Lithium-Thionyl Chloride ($Li\text{-}SOC_2$) battery was chosen due to its pulse capability and low self discharge. A Low-Dropout (LDO) regulator with low quiescent current was included in order to regulate the voltage to 1.8 V.

The hardware components selected can be found in table I.

B. Software Design

As mentioned before, in order to make a comparative study of performance and consumption, two firmware coding approaches were used to execute the functionality identified in figure 3.

The purpose is to acquire data, pre-process it and send the information about the monitored parameters to the Cloud

TABLE I
HARDWARE SPECIFICATION

MCU	Core	Power Supply	Consumption
STM32 L071K8	Arm® Cortex® -M0+	1.8V - 3.6V	140 µA/MHz @ run 0.8 µA @ stop mode (w/ RTC) 0.65 µA @ standby (w/ RTC)
Modem	**Core**	**Power Supply**	**Consumption**
Quectel BC66	Mediatek MT2625	2.1V - 3.63V	3.5 µA @ PSM 240 µA @ Idle (eDRX = 81.92 s) 110 mA @ LTE Cat NB1
Sensor	**Interface**	**Power Supply**	**Low Power Modes**
Hum.Temp.	SPI; I2C	1.71V - 3.6V	yes
Light	I2C	1.6V - 3.6V	yes
CH4	I2C	1.75V - 3.6V	yes
Accel.	SPI; I2C	1.72V - 3.6V	yes
Battery	**Type**	**Nominal Voltage**	**Capacity**
LSH14	Li-SOC_2	3.6V	5.8 Ah

server periodically, while taking advantage of low-power modes to reduce average power consumption. If it is not possible to send the information at a given time, the end device shall try to reconnect to the base station at a latter time (minutes). If it can not connect at all, the end device enters in standby, a deep sleep state, waiting for some time (hours) before rebooting and retry to communicate again. This gives time to detect, on the Cloud server side, that this particular end device is inactive and possibly requires attention.

1) Bare-Metal approach: The bare-metal application followed the flow exactly as exposed in figure 3, having three main layers: the hardware of the STM32L071K8U6, a hardware abstraction layer (HAL), and the application, as depicted in figure 4.

The hardware comprises the registers for interaction with peripherals and memory. Instead of writing directly to registers on the application layer, STMicroelectronics (ST) offers a HAL with functions that allow a more fluid manipulation of the hardware resources. In order to provide even more portability to the application, a low level module was developed so that the application doesn't get restricted to the HAL offered by ST.

The application layer has its own levels, depicted in figure 4 schematic. The bottom level comprises the system interaction, the modem and the specific sensors interaction. The system controls the Real-Time Clock (RTC) configuration and reading, and the power control. The modem module deals with the specificity of the NB-IoT modem selected (Quectel BC66), such as AT commands, and flow of execution for initialization and configuration. Each sensor specific module deals with the initialization, configuration and reading of the specified sensor, since their working mechanisms are different (proprietary for each sensor). The top level allows a certain level of abstraction, where the transmission and sensors modules are included. The former deals with transmission protocol, gathering the required information and requesting the modem module to send it in the correct format. The latter, centralizes the sensors information, allowing mass initialization and configuration, as well as unified organization of information from each sensor.

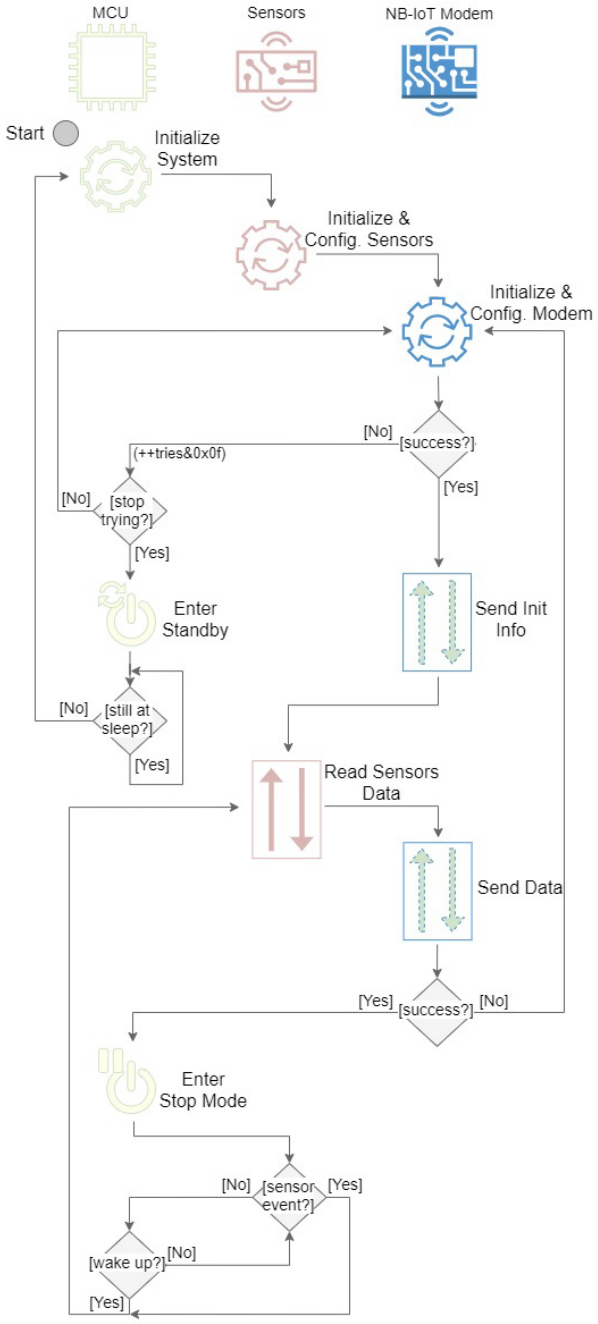

Fig. 3. System Flowchart

In this way, the main module can control the application flow in an abstracted way, so that its only concern is the correct execution of application procedures.

2) RTOS approach: There are several RTOSs available for IoT applications, both open-source as well as propriety. After analyzing some of the most known RTOSs, such as ThreadX and embOS (proprietary), and µC/OS-III, mbedOS and freeRTOS (open-source), the authors selected freeRTOS.

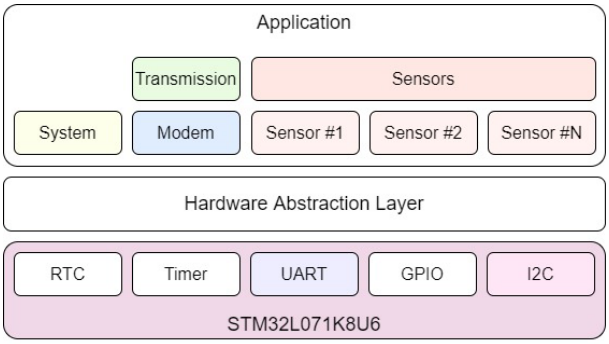

Fig. 4. System Architecture

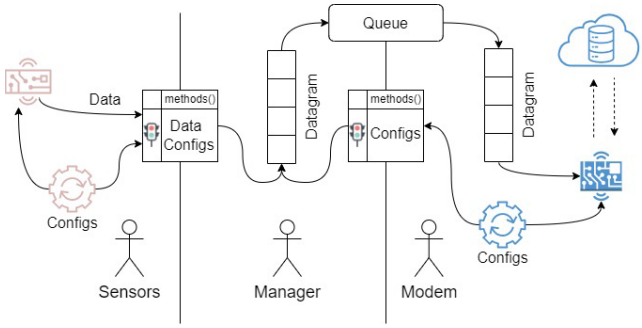

Fig. 5. Graphic Tasks Abstraction

According to [20], freeRTOS is not the fastest RTOS in most bench marked parameters, but its performance is comparable with the other open-source RTOSs analyzed. The choice of freeRTOS can be easily justified by its community support, previous development experience and seamless porting to the selected MCU using the ST Cube tool.

The freeRTOS application developed respects the same layers depicted in figure 4, except that it introduces a new layer, close to the hardware, representative of the freeRTOS kernel, that manages scheduling and context switching of tasks.

The functionalities offered by the end device firmware and its synchronization are depicted in figure 5, where a graphic tasks abstraction of the implementation in the RTOS context is provided. There are tasks and functions responsible for configuring and collect data from the sensors (left side of figure), which in turn are accessed by a manager using synchronization mechanisms such as semaphores. On the communication scope (right side of figure) there are also tasks responsible for initializing and configuring the NB-IoT connection to the base station and Cloud server, allowing the manager task to request configuration information and sending datagrams with respect to the appropriate synchronization mechanisms (semaphores and queues in this case).

The use of freeRTOS enables the mentioned pseudo-parallelism, making it possible to initialize the sensors "at the same time" the modem is being initialized, since there is a standstill between the transmission of an AT command and the modem response. In this time, modem tasks are blocked waiting for the response and other tasks can execute, or the MCU can be put into a shallow sleep mode for power consumption reduction.

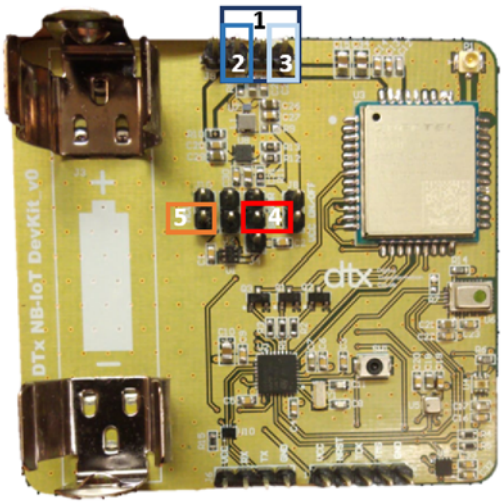

Fig. 6. NB-IoT Custom PCB

IV. Preliminary Results

Based on the diagram depicted in figure 1, a NB-IoT end device custom Printed Circuit Board (PCB) was designed and fabricated (figure 6). The design of the PCB allows to measure current consumption at five different circuitry test points:

1) All modules, from the point of view of the battery, overall end device power consumption (dark blue in figure 6);
2) MCU and sensors, from the point of view of the battery (including the LDO) (blue in figure 6);
3) NB-IoT modem only (communications power consumption) (light blue in figure 6);
4) Sensors only, from the point of view of the LDO (red in figure 6);
5) MCU only, from the point of view of the LDO (orange in figure 6).

The measurements were taken on subsystems 2 and 3 separately for better analysis. For this measurements, a Digital Multimeter (DMM) was used, the modem was supplied from an external power supply (3.5 V) when not being measured its consumption.

The comparison in figure 7 opposes the current consumption of the end device with a bare-metal firmware to the current consumption of the end device with a RTOS-based firmware. Both ran at the same clock frequency, in the same hardware conditions, the application has identical timings for execution, except between the two dashed lines showed in both graphs. This time is variable and depends on how fast the modem is able to connect to the network. When the energy consumption drops it means that the MCU has been put into stop mode and is waiting for a RTC wakeup. For tests purposes, this wakeup time was set to 10 seconds.

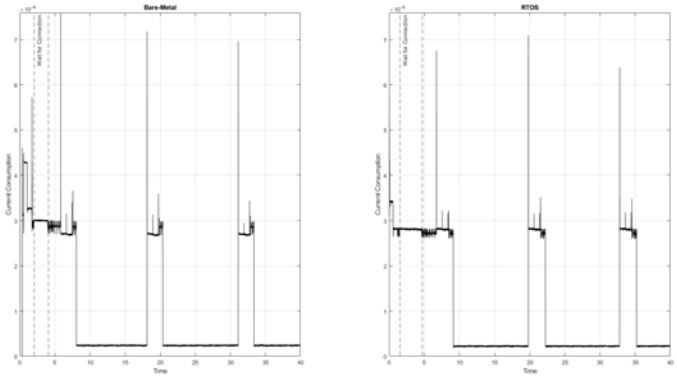

Fig. 7. Current Consumption of Bare-Metal (left) and RTOS-based (right) Firmware @ 1.048 MHz

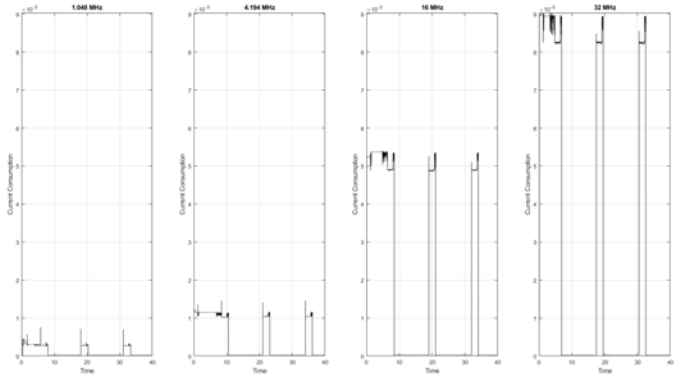

Fig. 8. Current Consumption at Different Clock Frequencies

From the results, there are two important phases to analyze: the use of the RTOS compensates at the initialization phase, having less 20 µA of average current consumption than the bare-metal implementation, but at the cyclic repetition, the bare-metal firmware achieves less 5 µA of average current consumption than the RTOS-based one. Measurements for other clock frequencies sustain these results, the higher the clock frequency, the larger the difference of current consumption between the two implementations at both phases.

The current consumption difference between the four clock frequencies selected for experimental measurements can be seen in figure 8. The average execution time and current consumption per peak (active time after wakeup) are listed in table II. It shows that using the MCU's maximum clock frequency (32 MHz) only decreases execution time by 10% while the power consumption is increased by 2900%, when compared to the values obtained at a lower clock frequency of 1.048 MHz.

One of the STM32L071K8's low-power features is the option to select the core supply voltage. For clock frequencies below 4.194 MHz it is possible to run the core with a 1.2V supply voltage instead of the 1.8V. The impact on current consumption of using this feature in lower clock frequencies can be observed in figure 9. It can represent savings of more than 345 µA on the average consumption, which for low-power

TABLE II
AVERAGE EXECUTION TIME AND CURRENT CONSUMPTION AT DIFFERENT CLOCK FREQUENCIES

	1.048 MHz	**4.194 MHz**	**16 MHz**	**32 MHz**
Execution Time (s)	2.2427	2.1110	2.0711	2.0691
Current Consumption (mA)	0.2751	1.0538	4.9571	8.3573

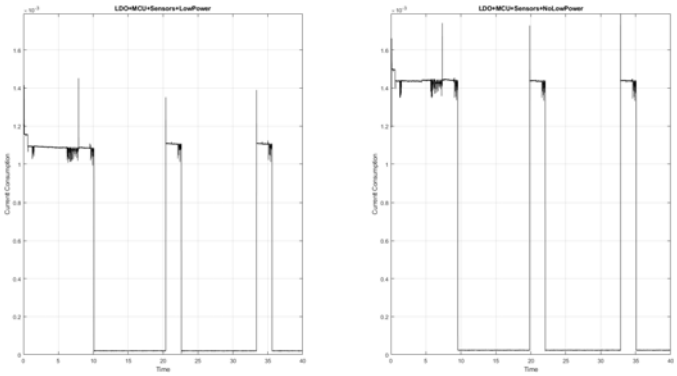

Fig. 9. Current Consumption with (left) and without (right) Low Power Configurations

sensitive applications is significant.

Table III presents the code and RAM size in bytes for each firmware. This is the necessary allocation of memory in order to implement the same application using different coding approaches. As expected, the RTOS-based firmware requires more memory, this is directly related to the initialization of tasks, queues and kernel.

During the experimental assessment, a problem was identified in the design related to the quiescent current consumption of the selected LDO. On an isolated measurement, it was found that the sensors have an average consumption of 2 µA and the MCU in stop mode has an average consumption below 1 µA, thus the verified 24 µA average current consumption when the system is in standby was attributed mainly to the quiescent current of the selected LDO.

As for the NB-IoT modem current consumption measurements, figure 10 presents one execution cycle from the start of the flowchart in figure 3. The modem was configured with no eDRX and PSM set to 2 hours with active-time of 10 seconds (time in which it is idle). However, for tests purposes, the MCU wakeup time was set to 5 minutes, so that a data package could be sent within the measurement time. The results show the drastic change in current consumption, dropping from an average of 17 mA in full functioning to an average of 560 µA in idle and mere 5 µA average when in PSM.

When testing the responsiveness of the modem, a problem with the previous selected battery was found, as it could not suffice the peak currents observed when establishing communication. This motivated to change from a LS17500 3.6 Ah battery to a LSH14 5.8 Ah battery, not because of the

TABLE III
CODE SIZE IN BYTES FOR EACH FIRMWARE

	Code	RO	RW	ZI
Bare-Metal	30184	324	24	4304
RTOS	39452	364	216	14816

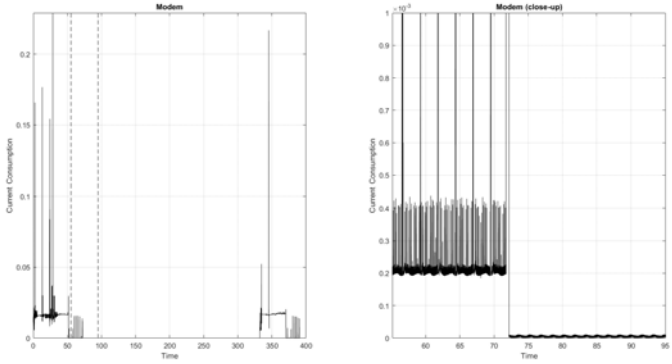

Fig. 10. NB-IoT Modem's Current Consumption with 20 Seconds Idle Time. Full measured cycle on the right and transition from idle to PSM close-up on the left

increased capacity, which theoretical allows for a longer life time, but because of its pulse capability, supporting up to 2 A peak of current for 0.1 seconds.

V. CONCLUSION

In this paper the design and development of a low power IoT end device with seamless wireless communications supported by NB-IoT was presented. The preliminary experimental results obtained allowed to conclude that selecting a lower clock frequency, for the end device MCU, is a better solution for reducing power consumption in the proposed IoT use case. The main reason for this to happen is because the application is much more dependent on the time taken by other hardware modules (sensors' measurement time and modem's communication link) than on exhaustive computations during a given task. So faster code throughput won't result in a significant decrease of the time required to return the end device to sleep mode.

Based on the results, it is also possible to conclude that the bare-metal application can offer a better energy consumption profile than the RTOS-based one. Besides allowing a better memory management, the bare-metal firmware has a coding flow more straightforward than the RTOS, thus making it better for troubleshooting when in development phase. However, code maintenance is more difficult, it is less flexible to change parts of the application and it is more difficult to include more power savings mechanisms, such as entering shallow sleep when waiting for external responses, which could translate on extending the end device battery lifetime even further.

Regarding battery lifetime, considering the LDO quiescent current of 24 µA previously explained, the end device reaches a 576 µAh per day, on a two-times-a-day update of the sensor values at 1.048 MHz clock frequency. When adding the communications contribution, based on the measured NB-IoT modem current consumption and with the same two-times-a-day update frequency, which gives a rough average of 483 µAh per day, the overall battery lifetime is 15 years.

Although these results are already promising for the perspective of a 10-year battery life target, they can be further optimized by means of software and hardware changes. A change to a nano-quiescent current LDO would significantly reduce the current consumption during the sleep phase of the system. On the software level, some changes to the application could also help reduce power consumption while increasing measurements resolution, by waking up just the MCU and the sensors more frequently, and wake up more sporadically just to send the data collected. Further analysis on current consumption between using the LSI or the LSE to drive the RTC clock need to be made, as well as using UDP or TCP for data communication.

So far, it is plausible for a NB-IoT design to achieve the announced 10-year battery life, provided that both software and hardware are power consumption aware, and identified high consumption modules within the system can be further improved in order to minimize their impact.

VI. ACKNOWLEDGMENT

This work has been supported by NORTE-06-3559-FSE-000018, integrated in the invitation NORTE-59-2018-41, aiming the Hiring of Highly Qualified Human Resources, co-financed by the Regional Operational Programme of the North 2020, thematic area of Competitiveness and Employment, through the European Social Fund (ESF).

This work has been supported by FCT – Fundação para a Ciência e Tecnologia within the R&D Units Project Scope: UIDB/00319/2020.

The authors would like to thank the support team of NOS Comunicações for providing the NB-IoT connection and server service, and for technical support throughout the installation.

REFERENCES

[1] A. Bera, "80 Insightful Internet of Things Statistics - 2020 Edition," feb 2019. [Online]. Available: https://safeatlast.co/blog/iot-statistics/#gref

[2] Huawei, "Huawei demonstrates world-first nuSIM implementation," nov 2019. [Online]. Available: https://www.huawei.com/uk/press-events/news/uk/2019/huawei-demonstrates-world-first-nusim-implementation

[3] Deutsche Telekom AG, "NarrowBand IoT The Game Changer for The Internet of Things," Tech. Rep. October, 2017.

[4] K. Changyun, "Design of Safety System for Kitchen Based on NB-IOT," *2019 3rd International Conference on Robotics and Automation Sciences (ICRAS)*, pp. 74–78, 2019.

[5] Y. Cheng, "Design of Air Quality Monitoring System Based on," *2019 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS)*, pp. 385–388, 2019.

[6] W. Jianxin, S. Junpan, and H. Ruyuan, "Design of a Smart Independent Smoke Sense System Based on NB-IoT Technology," *2019 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS)*, pp. 397–400, 2019.

[7] D. Xiong, Y. Chen, X. Chen, M. Yang, and X. Liu, "Design of Power Failure Event Reporting System Based on NB-IoT Smart Meter," *2018 International Conference on Power System Technology, POWERCON 2018 - Proceedings*, no. 201804270000855, pp. 1770–1774, 2019.

[8] S. Duangsuwan, A. Takarn, and P. Jamjareegulgarn, "A Development on Air Pollution Detection Sensors based on NB-IoT Network for Smart Cities," *ISCIT 2018 - 18th International Symposium on Communication and Information Technology*, no. Iscit, pp. 313–317, 2018.

[9] S. Yang, S. Khan, X. Chuanxi, Z. Yifeng, and P. Shengchun, "Design and Realization of a Buoy for Ocean Acoustic Tomography in Coastal Sea based on NB-IoT Technology," *OCEANS 2019 - Marseille*, pp. 1–4, 2019.

[10] W. Manatarinat, S. Poomrittigul, and P. Tantatsanawong, "Narrowband-Internet of Things (NB-IoT) System for Elderly Healthcare Services," *2019 5th International Conference on Engineering, Applied Sciences and Technology (ICEAST)*, pp. 1–4, 2019.

[11] N. Naik, "Cellular IoT — MDM9206 Modem and New LTE for IoT SDK - Qualcomm Developer Network," jul 2018. [Online]. Available: https://developer.qualcomm.com/blog/cellular-iot-mdm9206-modem-and-new-lte-iot-sdk

[12] Quectel, "Quectel QuecOpen." [Online]. Available: https://www.quectel.com/technology/quecopen.htm

[13] SIMCom, "9206 IOT SDK." [Online]. Available: https://www.simcom.com/service-2.html

[14] Nordic Semiconductor, "nRF9160 - Nordic Semiconductor," 2018. [Online]. Available: https://www.nordicsemi.com/Products/Low-power-cellular-IoT/nRF9160#infotabs

[15] STMicroelectronics, "Ultra low power Application note STM32L0," STMicroelectronics, Tech. Rep., 2014.

[16] T-Mobile, "Narrowband IoT Solution Developer Protocols," Tech. Rep., 2019.

[17] M. Stusek, K. Zeman, P. Masek, J. Sedova, and J. Hosek, "IoT Protocols for Low-power Massive IoT: A Communication Perspective," *International Congress on Ultra Modern Telecommunications and Control Systems and Workshops*, vol. 2019-October, no. November, 2019.

[18] K. K. Nair, A. M. Abu-Mahfouz, and S. Lefophane, "Analysis of the narrow band internet of things (NB-IoT) technology," *2019 Conference on Information Communications Technology and Society, ICTAS 2019*, pp. 1–6, 2019.

[19] S. Branco, "Archnet," Apr. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3763813

[20] R. R. Belleza and E. P. De Freitas, "Performance study of real-time operating systems for internet of things devices," *IET Software*, vol. 12, no. 3, pp. 176–182, 2018.