An IoT-oriented Fast Prototyping Platform for BLE-based Star Topology Networks

Lorenzo Invidia, Silvio Lucio Oliva, Andrea Palmieri, Luigi Patrono, and Piercosimo Rametta

***Abstract*—The Internet of Things (IoT) is characterized by many technologies, standards, tools and devices for a wide range of application fields and often, for the end-users (makers and developers), is hard to orientate in an equally wide range of offers from various manufacturers. In recent years, the Bluetooth Low Energy (BLE) communication protocol is achieving a large portion of the market, thanks to its low-power and low-cost orientation and its pervasiveness in mobile devices, like smartphones. For these reasons, BLE is increasingly used in IoT-oriented Wireless Personal Area Networks (WPAN), where a small set of devices arranged in star topology network and connected to a smartphone and a Wi-Fi gateway, can cover a large number of monitoring and controlling use case scenarios. This work presents the ST's STM32 Open Development Environment (ODE), a complete suite of hardware and software tools representing a reference point for end-users willing to create BLE-based star topology networks for a wide range of applications. Through a simple use case in a smart home context, it is shown how all provided tools can be used to fast prototype applications addressing all user requirements.**

***Index Terms*—BLE, Embedded System, Internet of Things, MQTT, Rapid Prototyping, STM32 ODE.**

I. INTRODUCTION

FROM a high-level conceptual perspective, the Internet of Things (IoT) refers to billions of physical devices around the world that are connected to the Internet, collecting and sharing data. This adds a level of digital intelligence to devices that would be otherwise dumb, enabling them to communicate without a human being involved, and merging the digital and physical worlds. From a low-level perspective, this concept refers to several devices capable to connect to one another, usually through wireless connectivity, and to the Internet to share data across the networks.

IoT [1] is characterized by common devices that become "smart" and collect data from sensors or other inputs. They communicate with other IoT devices and with mobile or cloud-based applications so that these data can be stored, shared, aggregated, analyzed as usable information. It can be easily understood that, in this context, one of the main critical aspects for smart devices is connectivity, both from one device to another in the same network and towards the Internet. Short-range wireless networks are playing an increasingly important role in the ecosystem of Internet-connected devices, offering quality in terms of reliability, speed and security comparable to wired links, but with the significant advantages of flexibility, mobility, and ease of connection.

In the last years, several protocols have been defined for the communication within Wireless Sensor Networks (WSN) [2] and toward the Internet, each characterized by different application scenario and performance, but no one managed to become a general solution to satisfy the incoming requirements of new IoT systems. Z-Wave [3], ZigBee, Thread 802.15.4 [4][5], 6LoWPAN [6], based on the IEEE 802.15.4 [7] standard, Wi-Fi 802.11 b/g [8], RFID [9-12], are just a few of the popular protocols and technologies designed for WSN applications.

As the use of WSN applications is increasing, the need to choose a particular protocol with respect to the other is determined by certain factors such as the requirements of the target application, size, consumption [13] [14] and duration of the network. In the last decade, the standard that is gaining increasing importance in this context is Bluetooth Low Energy (BLE) [15].

Bluetooth Low Energy is the intelligent, power-friendly version of Bluetooth wireless technology. Being able to successfully address such requirements, its rapid adoption is being guaranteed by the phenomenal growth in smartphones, tablets, and mobile computing. BLE is particularly suitable for IoT applications that do not require a continuous connection. In fact, it allows short interruptions of the radio connection, optimizing the use of the battery of the devices. Interference is greatly reduced using frequency-hopping and data integrity approaches and this makes Bluetooth connections much more powerful than 2.4 GHz Wi-Fi, with data transmitted that is rarely lost. BLE offers a secure and reliable connection for smart devices, which will extend even further with the arrival of the Bluetooth mesh network profile and changes to core specifications that will allow a longer range.

BLE provides a clear advantage for low-cost, low-power, and secure connectivity, with a single global standard enabling a wide range of devices from different manufacturers to

Manuscript received on January 9, 2019; revised March 5, 2019. Date of publication April 24, 2019. Date of current version June 3, 2019.

Luigi Patrono, Piercosimo Rametta and Lorenzo Invidia are with the Innovation Engineering Department, University of Salento, Via per Monteroni, 73100, Lecce, Italy.

Silvio Lucio Oliva and Andrea Palmieri work in the Advanced System Technology division of the STMicroelectronics, Lecce.

Luigi Patrono is the corresponding author (e-mail: luigi.patrono@unisalento.it).

Digital Object Identifier (DOI): 10.24138/jcomss.v15i2.682

communicate, no matter where they are installed. For IoT developers that eases many connectivity concerns, as consumers can use their smartphone as a host for multiple Bluetooth devices anywhere and anytime, offering services and data to other Bluetooth devices and connection points, including the cloud.

In developing WSN applications, various network topologies are possible. The topology of a wireless network is simply the way network components are arranged: it describes the physical layout of devices, routers, and gateways, as well as the data flow paths among them. Three of the most common wireless topologies for applications are: (i) star, (ii) mesh, and (iii) cluster-tree or hybrid [16].

In a star topology, all individual wireless nodes communicate directly with a central node. It is sometimes described as a "point-to-point" or "line of sight" architecture because each device communicates directly with the gateway. The gateway establishes a point-to-point communication with the peripheral nodes and transmits the received data to a local server or a cloud server, directly or by connecting to another network. This topology uses the least amount of power compared to other architectures thanks to simple direct wireless connections. However, the distance that data can be transmitted from the wireless device to the gateway is limited to a range of 30-100 meters. The devices in a mesh topology, instead, can also communicate with other nodes in the network (point-to-multipoint) using a capability called multi-hopping. A message can “hop” from node to node to node until it reaches the assigned gateway. The advantages of mesh over star topology includes a longer-range distance and a decrease in loss of data or transmission, at the expense of greater complexity of each node. Finally, a cluster-tree topology is a hybrid approach where wireless devices in a star topology are clustered around routers or repeaters that communicate with each other and the gateway in a mesh topology.

Driven by new trends, reduced costs, miniaturization and increased functionality, low-power wireless networks are increasingly being used in new areas of application. However, the efficient use of new intelligent systems is hampered by the lack of tools that allow the creation of vertical applications, from the implementation of the firmware to the distribution of the front-end application for the end user, whether it be a mobile application or web dashboard [17].

Almost all products have a series of features that can become "intelligent" and be introduced into the IoT with the addition of sensors, connectivity and an online application for control, management and generation of useful information.

Once a hardware device or a prototype is built, it is necessary to develop several layers of software: a built-in firmware, which defines the operation and allows the communication of the device, an application (often mobile) that becomes the user interface and a cloud platform for data collection and processing. This work overcomes these limitations and describes a complete suite of hardware and software tools, proposed by the ST’s STM32 Open Development Environment (ODE) [18], which allows the fast prototyping of IoT applications based on the BLE as low-level communication protocol, with nodes arranged in a star topology, since its ease in the considered use case. We have extended our previous work [19] with the addition of a cloud communication built within the same ODE. The adopted suite includes a vast set of hardware components, ranging from several MCU chips with increasing performance, to different versions of BLE and Wi-Fi communication interfaces, passing through a wide set of sensor and actuator expansion boards. All these blocks can be composed in modular prototypal development boards. Once the hardware layer is defined, different Integrated Development Environments (IDEs) can be used to develop the application’s business logic, also by exploiting examples and code snippets provided by the supporting community of developers. A companion Cloud Platform provides a remote endpoint to send collected data in order to be stored, visualized and accessed by other clients through Internet. This suite of tools provides a great flexibility for end-users in defining their final products. In fact, once the business logic has been implemented and all functional requirements have been satisfied, the hardware prototype can be finely optimized by composing one or more building blocks to upgrade performances or downgrade costs, according to application’s business core.

The rest of the paper is organized as follows: section II describes the hardware and software tools provided by the proposed platform, while section III describes how the above components provide functionalities to fast prototype IoT applications on star topology BLE networks. Our use case scenario is presented in section IV to functionally validate the whole platform. Then, a brief comparison with other competitors’ solutions is discussed in section V. Finally, conclusions and future works are summarized in section VI.

II. Technologies Overview

This Section presents a brief overview of the full suite of tools provided by the ST’s STM32 Open Development Environment (ODE). A short introduction of the Bluetooth Low Energy and the MQTT protocols, which are at the base of the communication within the wireless sensor network and from the central node towards the Internet, is reported. Then, the offer of hardware prototype boards is introduced, along with the main Integrated Development Environments (IDEs) used to rapidly prototype applications based on the described hardware and also interact with a Cloud server (provided by the suite) for storing and visualizing data.

A. Protocols Introduction

The Bluetooth Low Energy communication is built on two actors: a central and a peripheral device. Just like a client-server model, the peripheral - usually an IoT device - owns some information needed by clients. In most cases the central node has the resources needed to scan and start a connection with any peripheral that is advertising information that it’s interested in.

Peripheral data is structured by services and characteristics. Services are used to break data up into logic entities and

contain specific chunks of data called characteristics.
A characteristic provides further details about a service and encapsulates a single data point.

After a connection has been established, the central device starts to discover all the peripheral services and characteristics. Then it can interact with a service by reading, writing or subscribing to the value of its characteristics.

The BLE protocol stack is lightened compared to the Bluetooth classic:

- The Physical Layer features three advertising channels giving the chance to speed up the discovery of slave devices and leaving the others for data exchange.
- The link layer defines the procedures needed for scanning, advertising, creating and maintaining connections with devices.
- The Attribute Protocol (ATT) creates a communication between an Attribute Server and an Attribute Client. Data is stored in an attribute server in the form of "attributes" which can be discovered, read and written by an attribute client.
- The Generic Attribute Profile (GATT) defines the way that two devices transfer data back and forth, provides methods to discover services/characteristics and to read or write characteristics' value.
- Lastly, the Generic Access Profile (GAP) defines the API for modes (discoverable, connectable, bondable) and roles (peripheral, central, broadcaster, observer).

A cloud connection to manage data fetched from IoT nodes can be realized easily via the MQ Telemetry Transport (MQTT) [20]. Based on top of the TCP/IP suite, MQTT is a lightweight protocol specially designed for constrained environments, low-bandwidth networks and devices with limited performance. These features make it ideal to be used for most IoT nodes implementation.

MQTT defers from the classic Client-Server structure, where a client owns a direct communication to its endpoint. The MQTT's Publish-Subscribe model builds up a client that broadcast some information (publisher) and one or more clients that receive the same information (subscribers). Both endpoints do not know about the each other existence.

The prototype of client starts from a device with an embedded micro controller up to a server running a proper library, making MQTT a high scalable protocol, usually leaving a very low complexity at the client side. Furthermore, MQTT libraries are available for a huge variety of programming languages.

A fundamental role with MQTT is played by brokers. They are primarily responsible for receiving, filtering and dispatching messages, clients' authentication and authorization, so any client is always connected to a broker.

After a MQTT client has started a connection with a broker, it can publish messages. Each message basically consists of a packet id, a topic, used by the broker to forward the message to clients who are interested in, a Quality of Service level (QoS), which determines the delivery priority and the actual payload. MQTT payload data is completely free of any established format: it is up to the sender if it would to send textual data, binary data or JSON.

In order to receive messages, a client needs to subscribe to a broker. The subscribe message basically consists of a packet id and a set of pairs of a topic and QoS level.

B. Prototype Boards

The ST's STM32 ODE provides a modular hardware and software system to easy develop low/high complex prototypes, which can be placed rapidly in large production. It offers a rich catalogue of products that allows a complete IoT prototyping, from the choice of the board to other expansion shields needed. This brings great benefits to a user or a company who has a unique vendor for all hardware components and the same vendor for support or technical issues.
The STM32 ODE catalog (Table I) includes a wide range of evaluation boards for rapid prototyping of devices for different applications, from low power to high performance.

The STM32 Nucleo boards are available with different ARM Cortex®-Mx cores, memory cuts (flash and RAM), peripherals (SPI, U(S)ART, I2C, etc.) in order to meet different needs of customers, for different scenarios, with a competitive price.

Starting from the choice of a STM32 Nucleo board the user can extend the features by stacking one or more ST Expansion Boards, - e.g. environmental, motion or touch sensors, RF links, motor controls, security components, audio management etc. - with a scalable approach with unlimited possibilities for an easy application development or product evaluations.

Each STM32 Nucleo is characterized by a very simple hardware, an in-circuit debugger (ST-Link/V2-1) and several connectors (both the Arduino™ Uno V3 and the ST morpho) that allow quick connection with other hardware components, a comprehensive software HAL library and examples.

The NUCLEO-L476RG [21] (part of the STM32-L476xx series) can be taken into account to show an example of a typical prototype board. It is based on the high-performance ARM® Cortex®-M4 32-bit RISC core, operating at a frequency up to 80 MHz The STM32-L476xx series embed high-speed memories (Flash memory up to 1 Mbyte, up to 128 Kbyte of SRAM), a flexible external memory controller (FSMC) for static memories (for devices with packages of 100

TABLE I.
STM32 MCUS PORTFOLIO

STM32 Family	ARM Core	Freq. (MHz)	Flash (kB)	RAM (kB)
F0	Cortex-M0	48	16-256	4-32
F1	Cortex-M3	24-72	16-1024	4-96
F2	Cortex-M3	120	128-1024	64-128
F3	Cortex-M4	72	16-512	16-80
F4	Cortex-M4	84-180	64-2048	32-384
F7	Cortex-M7	216	256-2048	256-512
L0	Cortex-M0+	32	8-192	2-20
L1	Cortex-M3	32	32-512	4-80
L4	Cortex-M4	80	128-1024	64-320
L4+	Cortex-M4	120	1024-2048	640
H7	Cortex-M7	400	1024-2048	1024-1056

pins and more), a Quad SPI flash memories interface (available on all packages) and an extensive range of enhanced I/Os and peripherals connected to two APB buses, two AHB buses and a 32-bit multi-AHB bus matrix. The STM32L476xx devices feature an ultra-low-power with FlexPowerControl guaranteeing a consume of 30 nA in Shutdown mode and 120 nA in Standby mode.

The devices offer up to three fast 12-bit ADCs (5 Mbps), two comparators, two operational amplifiers, two DAC channels, an internal voltage reference buffer, a low-power RTC, two general-purpose 32-bit timer, two 16-bit PWM timers dedicated to motor control, seven general-purpose 16-bit timers, and two 16-bit low-power timers. The devices support four digital filters for external sigma delta modulators (DFSDM). They also feature standard and advanced communication interfaces (e.g. I2C, SPI, UART).

C. Integrated Development Environment (IDE)

Most STM32 Nucleo boards are supported and integrated into the ARM® Mbed™ online resources [22].

ARM Mbed OS is a software solution for the development of prototypes based on the 32-bit ARM Cortex-Mx microcontroller. Mbed makes the whole development process much simpler, significantly reducing time and costs. In fact, thanks to its operating system, it guarantees an abstraction from the hardware configuration. This means that code written on top of Mbed OS works on any board: the user can develop and test his/her code with different boards without rewriting it. To further help prototyping stages, Mbed provides not only the OS, but also Mbed Enabled™ services, a debugging interface and testing tools.

The Mbed OS developer site is a starting point where the user can browse an extensive catalogue of Mbed-enabled hardware for prototyping, including boards, modules or expansion boards. From the Hardware section, the user can browse the catalogue by Modules, Components or Boards and filter the hardware by vendor, connectivity, interface firmware etc. (Fig. 1). Therefore, the user is ready to implement the application logic simply choosing the board that he/she wishes to use and optionally selecting one or more expansion board to add functionalities. Alternatively, the user can import sample programs or libraries for the selected hardware directly into the online compiler simply clicking on the import button.

The applications for the selected platform can be developed using the ARM Mbed online IDE or an external IDE - e.g. Keil µVision [23], IAR [24], System Workbench for STM32 [25].

The SDK includes a C/C++ software platform that consists of high-level core libraries that provide microcontroller networking, drivers, RTOS and runtime environment, build tools and test and debug scripts for creating a firmware that runs on smart devices. Moreover, Mbed provides a components' database that includes lot of libraries for components and services that, connected to the microcontrollers, build the final prototype.

D. Cloud Server Platform

IBM Watson IoT Platform [26] is the cloud solution adopted to upload network data. By using Watson, users can collect connected device data and perform analytics on real-time data from the network.

The connection process is pretty simple and meets a few requirements for communicating with Watson IoT Platform: a communication by using HTTP or MQTT protocols and the devices messages must conform to the Watson message payload format.

The data management component of Watson IoT Platform

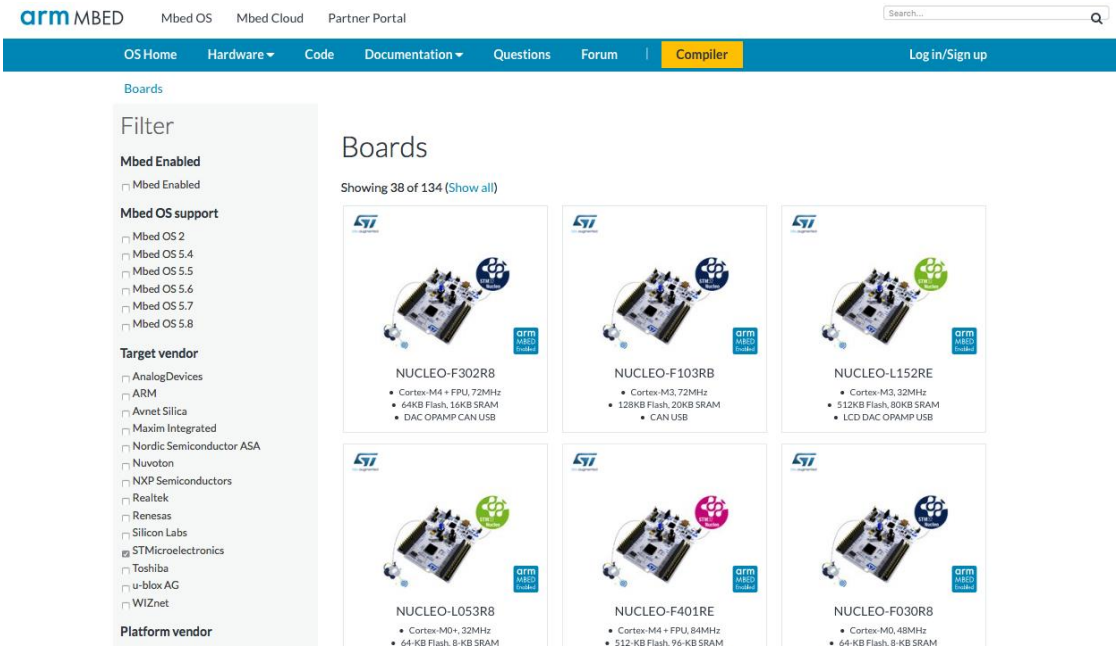


Fig. 1. The Mbed OS developer site with a catalogue of Mbed-enabled hardware

includes features to enable users to take advantage of the collection, transformation and normalization of different formats of device data into a single logical model. This enables them to group different devices together to create a Thing, which is a higher value asset-based data structure, so an application can interact with the logical model, regardless of the data format that is used by the individual devices or Things.

A registered mode allows the user to pair a device with the cloud. This solution enables end-users to control and send commands to the registered device via web interface.

E. Examples and Source Code Repository

The Mbed OS developer site makes several sections available to help developers start coding. A quick start example about Bluetooth Low Energy and a simple implementation of the BLE stack by a set of APIs can be found in conjunction with the expansion board documentation [27]. Moreover, a MQTT setup via mbed libraries or a TCP/IP one is available in the developer webpage [28].

The sample firmware, developed for a central node of a BLE star-topology network and described in the following sections, offers a rich overview of the implementations of these technologies and an effective example of the dual mode operation of the Bluetooth (master and slave).

The ST SensNet app (available on the app stores with the new name ST BLE StarNet) provides the possibility to manage and control the BLE star network through a user-friendly graphical interface. It's available for both Android [29] and iOS [30] devices, and the source code is available on GitHub along with the branch where the MEMS view and functionality have been developed.

III. System Architecture

By exploiting the whole set of tools described above, it is possible to rapidly create prototypes of applications in the IoT context.

In order to narrow the vastity of application field, in this work we focus on applications based on a star topology network, applied, just as an example, in a smart home scenario. This is not a technical limitation, since the described hardware and software components can manage different topology networks in several fields, but it is a discussion choice for the sake of readability.

Fig. 2 depicts the overall architecture of our application in such scenarios. The Wireless Personal Area Network (WPAN) is composed of a certain number of nodes - up to 7 peripheral nodes - arranged in a star topology. Devices with sensing and/or actuating capabilities read or modify some physical quantities of the surrounding environment, exchanging data only with the Central Node. This one, has a twofold role: on the one hand it acts as a BLE master/slave to monitor and control the peripheral nodes or by interacting with another BLE masters - like a smartphone -, and, on the other hand, it acts as a Wi-Fi gateway to exchange network data with the Cloud platform.

A custom smartphone application, acting as a BLE master, can locally interact with the WPAN by exploiting the BLE channel and the capability of the Central Node to act as a master and slave at the same time. This allows the smartphone application to pair with the Central Node - which acts as a peripheral with respect to the smartphone -, to access the data generated by the network or to control the nodes, while the

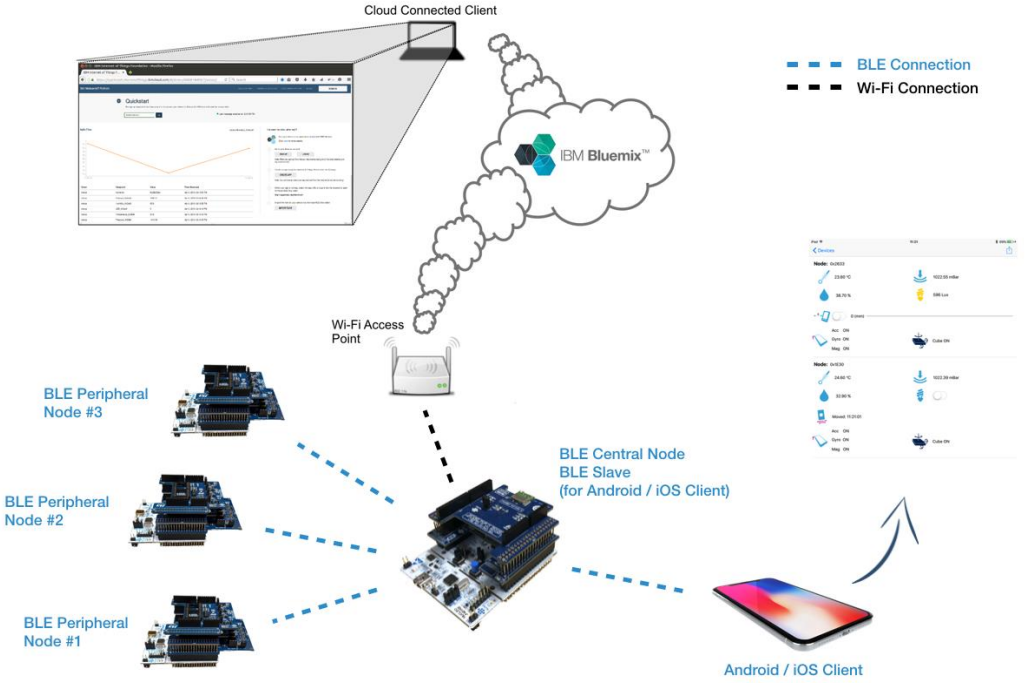


Fig. 2. Overall architecture of a typical application based on a BLE star topology network connected to the Cloud

Central Node keeps acting as a BLE master with respect to the peripheral nodes of the WPAN.

On the other side, once the collected data get the Cloud server through the Wi-Fi interface of the Central Node, they can be stored on the platform and accessed by any connected web client. This allows to remotely visualize data and eventually send command to any peripheral node.

A. Technical Details

In order to realize the hardware and software architecture illustrated above, some setup steps need to be implemented, especially to properly configure the Central Node and the peripheral nodes according to the application requirements.

The configuration stage starts selecting in the Mbed online user space the hardware components for the IoT prototype and generating the project, with the required libraries. First steps are navigating to *Hardware/Boards*, filtering the results by Target vendor and selecting the NUCLEO-L476RG board: this will add the board's firmware files in the compiler, simply by clicking at the import button. At the same way, by navigating to *Hardware/Components*, we select the *X-NUCLEO-IDB05A1* shield - it embeds the ST's BLE chip -, importing its own library [31] into the compiler. This library implements a set of APIs that guarantee an abstraction of the Bluetooth LE stack, making all setups and events managing an easy stuff (Fig. 3).

Other libraries required are the *MQTT* [32], that gives methods to easily connect IBM Cloud, and *easy-connect* library [33]; this library gives the users the possibility to switch between connectivity methods - e.g. Wi-Fi, Ethernet, Cellular - simply adding a few lines to the mbed_app.json file.

The developed firmware [34] configures an STM32 Nucleo board as the central node of the network. The central node can act as a BLE Master, connecting up to seven peripheral nodes, and receiving different kind of data from them, e.g. environmental data (temperature, humidity, pressure, lux), proximity or LED status. Moreover, the central node can act as a BLE Slave, sending the data information, received from the connected peripheral nodes, to an Android/iOS client (running the ST SensNet app) via a Bluetooth Low Energy connection. Then the master connects to the cloud via MQTT and starts to publish network data, making nodes information accessible via web.

```
1.  /* Callback when a local Attribute is written */
2.  ble.gattServer().onDataWritten(AttributeModified_CB);
3.  /* Callback when a peripheral node Characteristic is read */
4.  ble.gattClient().onDataRead(readCharacteristicCallback);
5.  /* Callback when a peripheral node Characteristic change */
6.  ble.gattClient().onHVX(onNotificationCallback);
7.  /* Callback when a peripheral node Descriptor is written */
8.  ble.gattClient().onDataWritten(perDescriptorWrittenCallback);
9.
10.
11. /* Callback when a node disconnects to the network */
12. ble.gap().onDisconnection(disconnectionCallback);
13. /* Callback when a node connects to the network */
14. ble.gap().onConnection(connectionCallback);
```

Fig. 3. Events managing setup. When a callback occurs the API pass control at the lower levels of the BLE

IV. System Validation

As a system validation scenario, let us suppose to create an application in a Smart Home/Ambient Assisted Living (AAL) context. In such scenario, an environment sensor node monitors humidity, temperature and pressure values of the domestic environment - the room where the sensor is placed in -; a proximity sensor can be used to check if a person comes near or moves away from a certain place, for example a corner in the room where a physical exercise tool is located - i.e. an exercise bike -. Finally, a wearable sensor equipped with inertial MEMS can be used to monitor the body motility state of the user. This sensing infrastructure can be used to monitor user's lifestyle in relation to ambient parameters: for example, it can be monitored how long the user is in a still position during the day, how many times the user interacts with the exercise bike to follow his/her slimming plan and how room temperature, humidity and pressure influence (or are affected by) user behavior. All collected data can be accessed both locally, through a smartphone app, and remotely through a web dashboard.

The platform described so far provides all the tools to create and prototype such smart home environment, leaving creativity and imagination to developers about the amount of other possible implementations.

A. Implementation Details

Different ST development boards have been adopted to setup the BLE star network. The Central Node hardware is composed of an STM32 Nucleo development board (ST Part Number: *NUCLEO-L476RG*), a BLE expansion board (*X-NUCLEO-IDB05A1*), and a Wi-Fi expansion board (*X-NUCLEO-IDW01M1*).

The BLE expansion board is based on the BlueNRG-MS RF module [35], a very low power Bluetooth low energy master/slave network processor, compliant with the Bluetooth specification v4.1. It integrates a 2.4 GHz RF transceiver and a powerful Cortex-M0 microcontroller, on which a complete

power-optimized stack for Bluetooth single mode protocol runs, providing:
- Master, slave role support
- GAP: central, peripheral, observer or broadcaster roles
- ATT/GATT: client and server
- SM: privacy, authentication and authorization
- L2CAP
- Link Layer: AES-128 encryption and decryption

In addition, according to the Bluetooth specification v4.1 the BlueNRG-MS can support the following features through firmware updates:
- Multiple roles simultaneously support
- Support simultaneous advertising and scanning
- Support being Slave of up to two Masters simultaneously
- Privacy V1.1
- Low Duty Cycle Directed Advertising

TABLE II.
MIC-ENV NODE DESCRIPTION (FROM [19])

ST SW package	STM32 Nucleo	ST Expansion Boards	ST Expansion Board Function
FP-SNS-ALLMEMS1	Nucleo-L476RG	X-NUCLEO-CCA02M1	MEMS Microphone
	Nucleo-F446RE	X-NUCLEO-IKS01A2 / X-NUCLEO-IKS01A1	Motion MEMS and Environmental Sensors (temperature, pressure, humidity)
	Nucleo-F401RE	X-NUCLEO-IDB05A1	BLE connectivity

TABLE III.
PRX-ENV NODE DESCRIPTION (FROM [19])

ST SW package	STM32 Nucleo	ST Expansion Boards	ST Expansion Board Function
FP-SNS-FLIGHT1	Nucleo-L476RG	X-NUCLEO-IKS01A2 / X-NUCLEO-IKS01A1 X-NUCLEO-53L0A1 / X-NUCLEO-6180XA1	Motion MEMS and Environmental Sensors (temperature, pressure, humidity) Proximity and Ambient Light (LUX) Sensor
	Nucleo-F401RE	X-NUCLEO-IDB05A1	BLE connectivity

TABLE IV.
MOT-ENV NODE DESCRIPTION (FROM [19])

ST SW	STM32	ST Expansion	ST Expansion

Ultra-low-power sleep modes and very short transition time between operating modes result in very low average current consumption during real operating conditions, providing very long battery life.

In order to create the BLE network, the central node can connect three types of peripheral nodes, which embed different features and run three different software packages

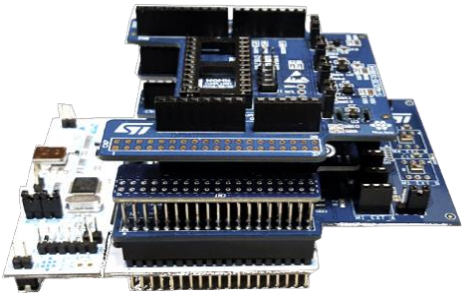


Fig. 4. MIC-ENV node (from [19])

Fig. 5. PRX-ENV node (from [19])

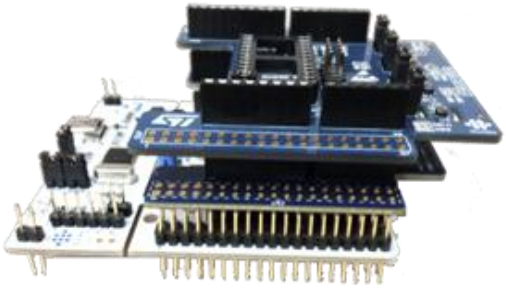


Fig. 6. MOT-ENV node (from [19])

[36-38]. To differentiate and describe the three nodes according to the different exported data, they have been named: *MIC-ENV*, *PRX-ENV and MOT-ENV*. Fig. 4-6 and Tables II-IV show for each peripheral node the part number of both the ST software package (freely available on st.com) and the STM32-Nucleo and Expansion boards that can be used, along with the indication of provided feature or the exported data types.

In order to start the sequence of operations, the system checks if the Wi-Fi is enabled and starts to connect the Access Point (Fig. 7). The system configures the MQTT network and prints on the screen the URL to manage and monitor network data. It's possible to encrypt these data uploaded enabling TLS protocol (Fig. 8). Lastly the initialization of the BLE module and some configurations are performed (Fig. 9).

The first stage sees the central node starting the discovering process for new peripherals. When a new node with a proper payload start advertising, a connection to it is created. Then, the master starts looking for services and characteristics of the connected peripherals showing their features to the user (Fig. 10).

Once at least one peripheral is connected to the central node, the master starts to print the nodes environmental data (Temperature, Pressure, Humidity, and Luminosity) on the serial terminal, and these data is uploaded to the IBM cloud every 1000 ms (Fig. 11). In the meanwhile, the central node performs a cycle of scanning of the network for new peripheral nodes and advertising to become visible to an Android/iOS device running the ST SensNet app (Fig. 12a), freely available on Google Play/App Store. When a connection to a peripheral node is established, notifications on that node are enabled and all data exported are shown.

The user can easily monitor the state of the network from the main view, which displays each peripheral node connected to the PAN (Fig. 12b). This view allows to interact with the peripherals, enabling a notification, sending commands and receiving data back. It is also possible to stop and restart the scanning for new peripheral nodes.

Basically, the central node behaves in a double way: as a master towards the peripheral nodes and as a slave with the Android/iOS device connected to.

Some devices embed some features that provide notifications. Examples are *Wake Up* (a notification that occurs when the card is moved) or *Proximity* notification (measuring the distance between an object and the node itself). These types of notifications can be activated or deactivated by the user and their values are displayed both in a serial terminal running on a host attached to the central node and in the SensNet ST application. Other notifications are provided by the MEMS sensors as shown in Fig. 13 - i.e. the Accelerometer, Gyroscope, Magnetometer and Sensor Fusion algorithms.

The system keeps track of the temporary disconnection of the nodes. The disconnection event is notified to the connected client which disables the cell related to the disconnected node and the user interaction (Figure 14). When the node returns connected, the respective cell returns active and user interaction is restored.

```
/******************************************************
*                                                     *
*          BLESTAR1 MBED Expansion Software           *
*                                                     *
******************************************************/


Wi-Fi Enabled!

To edit SSID and/or Password please refer to mbed_app.json file

Connecting to Access Point...

[EasyConnect] IPv4 mode
[EasyConnect] Using WiFi (IDW01M1)
[EasyConnect] Connecting to WiFi TP-LINK_2.4GHz
[EasyConnect] Connected to Network successfully
[EasyConnect] MAC address 00:80:E1:B7:D0:7A
[EasyConnect] IP address 192.168.1.111
```

Fig. 7. Attempt to connect to the Wi-Fi Access Point for the central node

```
Configuring MQTT network...

Connecting MQTT broker...

=====================================
Nucleo IP ADDRESS: 192.168.1.111

Nucleo MAC ADDRESS: 00:80:E1:B7:D0:7A

Server Hostname: quickstart.messaging.internetofthings.ibmcloud.com port: 1883

Client ID: d:quickstart:sensor:0080E1B7D07A

Topic: iot-2/evt/status/fmt/json

Subscription URL: quickstart.internetofthings.ibmcloud.com/#/device/0080E1B7D07A/sensor/
=====================================

--->TLS is OFF

--->TCP Connected

--->MQTT Connected
```

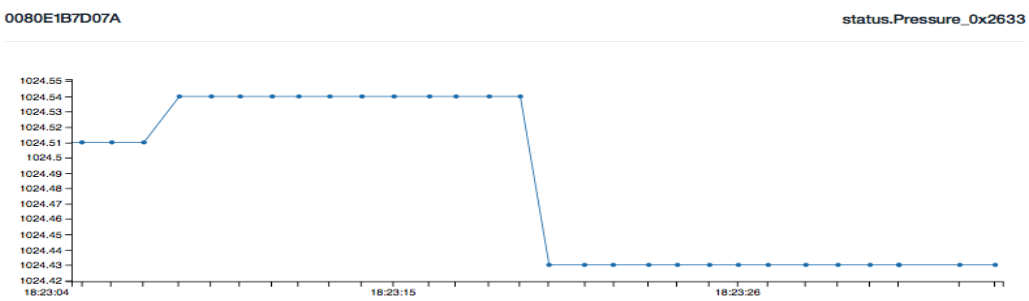
Fig. 8. Setup of the MQTT channel for the central node

```
Starting the BLE module...

Star HW Gatt Service added (handle 0x000c)

Star HW Config Char added (handle 0x000d)

Star HW Data Char added (handle 0x0010)

BLE CENTRAL MAC ADDRESS: f7:18:9c:c3:ff:3c

Start Device Discovery (1)

Device Discovery Complete (4)
```

Fig. 9. Initialization of the BLE channel for the central node

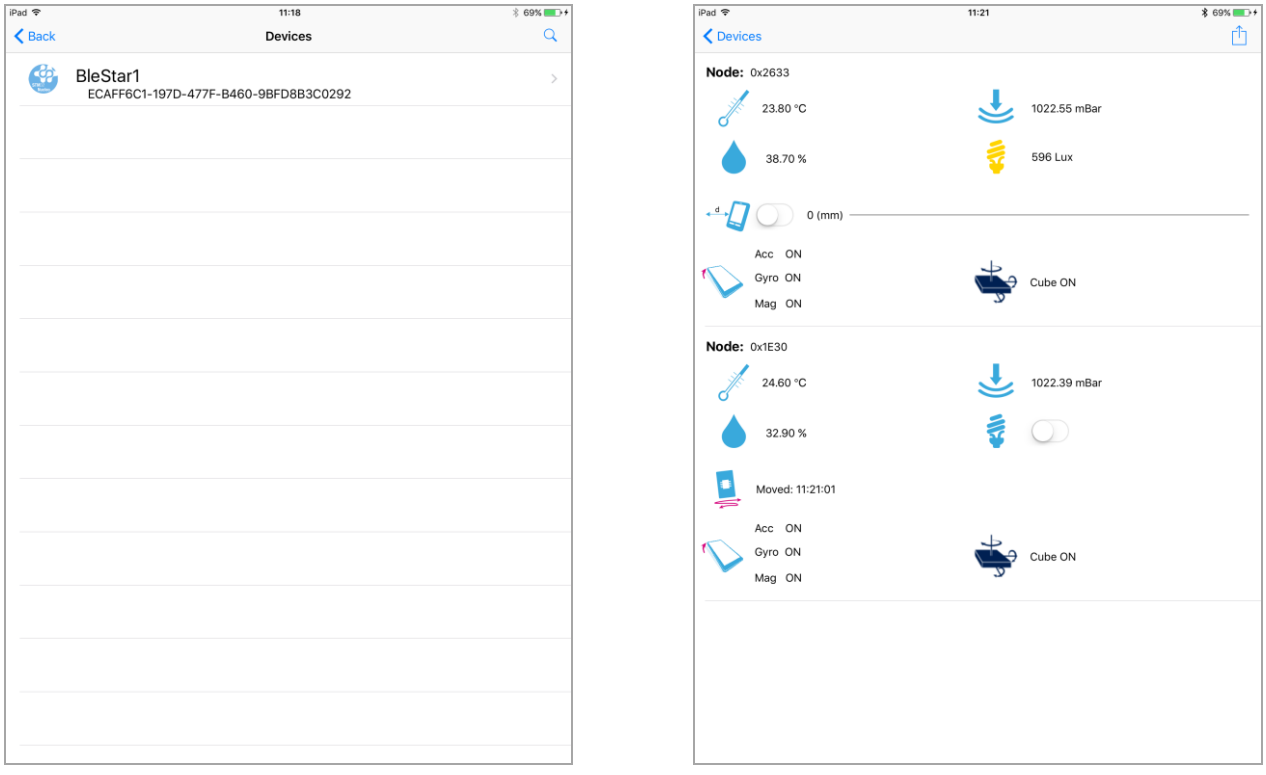
```
Start Device Discovery (1)

Peripheral (MOTENV) 1 inserted [c0 84 66 39 26 33]

Device Discovery Complete (4)

Client create connection with peripheral 1 at pos 1

Connected to peripheral: [c0 84 66 39 26 33] (1/1 - 0x0801) (role: master)

Discovering services for peripheral 1 (0x0801)
```

Fig. 10. Discovery of peripheral nodes of the network



Event	Datapoint	Value	Time Received
status	ST	BLEStar	Apr 3, 2018 6:23:34 PM
status	Pressure_0x2633	1024.43	Apr 3, 2018 6:23:34 PM
status	Humidity_0x2633	92	Apr 3, 2018 6:23:34 PM
status	Temperature1_0x2633	20.2	Apr 3, 2018 6:23:34 PM
status	LUX_0x1e30	11	Apr 3, 2018 6:23:32 PM
status	Pressure_0x1e30	1019.53	Apr 3, 2018 6:22:55 PM
status	Humidity_0x1e30	72	Apr 3, 2018 6:22:55 PM
status	Temperature1_0x1e30	20	Apr 3, 2018 6:22:55 PM

Fig. 11. Graphical user interface of the IBM Watson IoT Platform showing collected data



a) b)

Fig. 12. Screenshots of the ST SensNet smartphone app showing (a) the discovery phase and (b) data collected by each node (from [19])

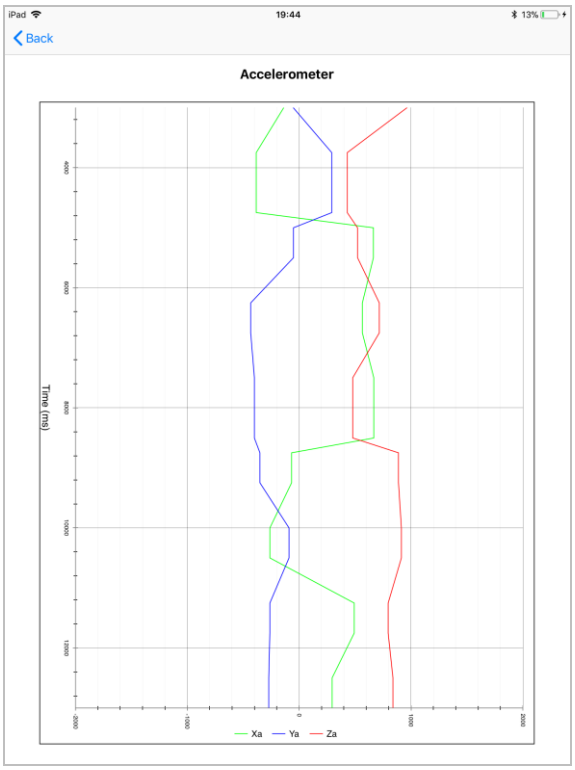


Fig. 13. Accelerometer data shown in the SenseNet app (from [19])

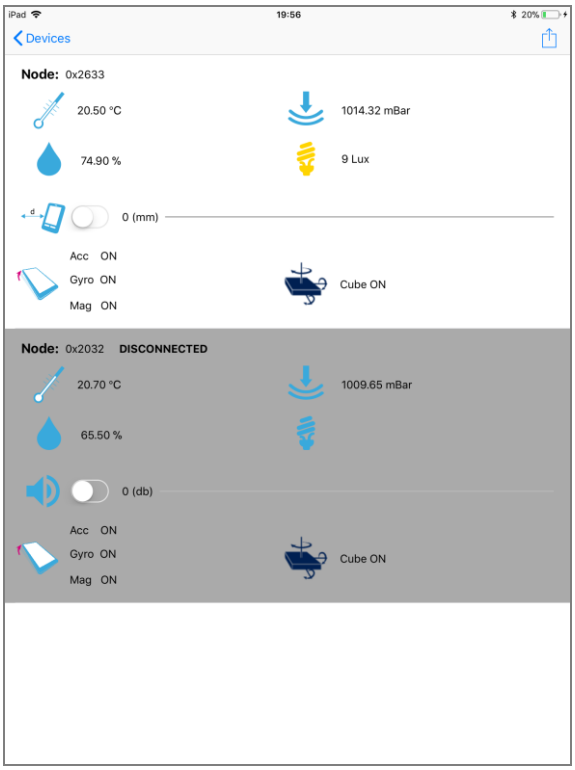


Fig. 14. A temporarily disconnected node (from [19])

V. Discussion

Compared to other solution on the market - e.g. Arduino, Espressif Systems, Adafruit -, STMicroelectronics offers a large product catalogue as much as competitors, guaranteeing a wide range of available components for prototyping, both hardware and software.

A wide range of microcontroller, in terms of different computing performances, memory cuts and/or power consumptions, allows users to keep the right one for their needs, therefore the right development board that matches the kind of prototype they aim to achieve.

Other competitor's hardware solutions provide only a few or a single board in order to satisfy a larger set of implementations, so the ST ones provide greater flexibility.

Concerning the expansions, ST arises as a unique vendor for all hardware components, in order to help users buy expansions, easily put together all the features they want to add to the prototype and guaranteeing them a unique reference for support and/or technical issues.

ST hardware also features a full compatibility with different development environments: it's possible to develop firmware using the STM32Cube software and tools [39], Mx, Arm Mbed OS or Arduino IDE [40], because of the third part hardware support - i.e. Arduino shield -.

All ST hardware is characterized by a simple design and it is production-oriented, thus guaranteeing a fast optimization of prototypes to make them easily and rapidly in large production, saving time to manufacturers or companies.

This work provides both makers and professional developers with all information about the hardware and software tools available on the market nowadays, identifies a fast and affordable solution for the prototyping and shows how to easy realize an IoT prototype.

The WPAN implementation, described in this paper, matches the perfect compromise between an ultra-low power consumption and a necessary computing capability because of the central node complexity and requirements. Here the NUCLEO-L476RG board, thanks to its ultra low power features, the core Cortex-M4 running @80MHz and the memory cuts (both in terms of flash and RAM), has been identified and adopted for realizing the prototype.

The wide variety of platforms in the ST portfolio provides the developers with the possibility to make the best choice according to the particular designed application or use case, having at the same time a ready for production device and so greatly decreasing the time to market.

VI. Conclusions

In this paper we have extended our prototyping of WSN, presented in the latest paper for Splitech, with the addition of a cloud connection that allows to interact with the WSN remotely. Also, this part is built on the ST's STM32 Open Development Environment (ODE). It is a complete suite of

hardware and software tools allowing end-users to fast prototype applications for a wide range of scenarios.

The suite is composed of (i) a wide portfolio of hardware components, ranging from CPU modules, RF interfaces, memory cuts, to pre-set development boards with associated sensor/actuator expansion boards, (ii) a set of IDEs to fast configure the firmware of the boards, including the business logic of the application (iii) a companion Cloud platform which allows to remotely store data collected by the WPAN, eventually controlling end devices via Internet, and (iv) a supporting community providing repositories and sample codes to speed up application development.

The described system architecture and the related validation proof-of-concept have been deliberately focused on a star topology network for the sake of readability. Nevertheless, this simple use case shows how such a simple kind of network can fulfill user requirements for a wide range of application scenarios.

The work aiming at enhancing the offer of each component of the suite is in an ongoing phase, with a particular focus on the implementation of the latest release of the Bluetooth Low Energy stack allowing mesh networking.

REFERENCES

[1] Mainetti, L., Patrono, L., Vilei, A., "Evolution of wireless sensor networks towards the Internet of Things: A survey" (2011), 2011 International Conference on Software, Telecommunications and Computer Networks, SoftCOM 2011, art. no. 6064380, pp. 16-21.

[2] Alessandrelli, D., Mainetti, L., Patrono, L., Pellerano, G., Petracca, M., Stefanizzi, M.L. "Implementation and validation of an energy-efficient MAC scheduler for WSNs by a test bed approach" (2012), 2012 20th International Conference on Software, Telecommunications and Computer Networks, SoftCOM 2012, art. no. 634761.

[3] Z-Wave website. http://www.z-wave.com/ (accessed December 2018)

[4] Zigbee Alliance website. http://www.zigbee.org/ (accessed December 2018).

[5] Thread Group web site. https://www.threadgroup.org/ (accessed December 2018).

[6] IPv6 over Low power WPAN (6lowpan), https://datatracker.ietf.org/wg/6lowpan/about/ (accessed December 2018).

[7] IEEE 802.15 WPAN™ Task Group 4 (TG4), http://www.ieee802.org/15/pub/TG4.html (accessed December 2018).

[8] IEEE 802.11 Wireless Local Area Networks, http://www.ieee802.org/11/ (accessed December 2018).

[9] Calcagnini, G., Censi, F., Maffia, M., Mainetti, L., Mattei, E., Patrono, L., Urso, E., "Evaluation of thermal and nonthermal effects of UHF RFID exposure on biological drugs" (2012), IEEE Transactions on Information Technology in Biomedicine, 16 (6), art. no. 6218185, pp. 1051-1057.

[10] Catarinucci, L., Colella, R., De Blasi, M., Mighali, V., Patrono, L., Tarricone, L., "High performance RFID tags for item-level tracing systems" (2010), SoftCOM 2010 - International Conference on Software, Telecommunications and Computer Networks, art. no. 5623656, pp. 21-26.

[11] Catarinucci, L., Colella, R., De Blasi, M., Patrono, L., Tarricone, L., "Experimental performance evaluation of passive UHF RFID tags in electromagnetically critical supply chains" (2011), Journal of Communications Software and Systems, 7 (2), pp. 59-70.

[12] P. Solic, Z. Blazevic, M. Skiljo, L. Patrono, R. Colella, J.J.P.C. Rodrigues (2017), "Gen2 RFID as IoT enabler: Characterization and performance improvement", IEEE Wireless Communications, 24 (3), June 2017, art. no 2402, pp. 33-39.

[13] Alessandrelli, D., Mainetti, L., Patrono, L., Pellerano, G., Petracca, M., Stefanizzi, M.L. "Performance evaluation of an energy-efficient MAC scheduler by using a test bed approach" (2013), Journal of Communications Software and Systems, 9 (1), pp. 84-96.

[14] Anchora, L., Capone, A., Mighali, V., Patrono, L., Simone, F. "A novel MAC scheduler to minimize the energy consumption in a Wireless Sensor Network" (2014), Ad Hoc Networks, 16, pp. 88-104.

[15] Bluetooth Special Interest Group. https://www.bluetooth.com/bluetooth-technology/radio-versions (accessed December 2018).

[16] Studytonight.com. Types of Network Topology, https://www.studytonight.com/computer-networks/network-topology-types (accessed December 2018).

[17] Mainetti, L., Mighali, V., Patrono, L., Rametta, P., Oliva, S.L., "A novel architecture enabling the visual implementation of web of Things applications" (2013), 2013 21st International Conference on Software, Telecommunications and Computer Networks, SoftCOM 2013, art. no. 6671847.

[18] St.com. STM32 Open Development Environment (ODE), http://www.st.com/content/st_com/en/products/ecosystems/stm32-open-development-environment.html (accessed December 2018).

[19] S. L. Oliva, A. Palmieri, L Invidia, L Patrono, P. Rametta, "Rapid Prototyping of a Star Topology Network based on Bluetooth Low Energy Technology" (2018), 3rd International Conference on Smart and Sustainable Technologies (SpliTech).

[20] Mqtt.org. The MQTT protocol. http://mqtt.org/ (accessed December 2018).

[21] St.com. STM32L476RG, http://www.st.com/content/st_com/en/products/microcontrollers/stm32-32-bit-arm-cortex-mcus/stm32-ultra-low-power-mcus/stm32l4-series/stm32l4x6/stm32l476rg.html (accessed December 2018).

[22] Arm Mbed OS developer site. https://os.mbed.com/ (accessed December 2018).

[23] armKEIL. Keil µVision IDE. http://www2.keil.com/mdk5/uvision/ (accessed December 2018).

[24] IAR Systems. IAR Embedded Workbench. https://www.iar.com/iar-embedded-workbench/ (accessed December 2018).

[25] OpenSTM32 Community. System Workbench for STM32. http://www.openstm32.org/HomePage (accessed December 2018).

[26] IBM Cloud. Internet of Things Platform. https://console.bluemix.net/catalog/services/internet-of-things-platform (accessed 27 April 2018).

[27] armMBED. Getting started with X-NUCLEO-IDB05A1. https://os.mbed.com/teams/ST/code/BLE_HeartRate_IDB0XA1/ (accessed December 2018).

[28] armMBED. Cloud_IBM_MbedOS. https://os.mbed.com/teams/ST/code/Cloud_IBM_MbedOS/ (accessed December 2018).

[29] Invidia L., ST SensNet Android mobile application, https://github.com/lorenzoinvidia/STSensNet_Android (accessed December 2018).

[30] Invidia L., ST SensNet iOS mobile application, https://github.com/lorenzoinvidia/STSensNet_iOS (accessed December 2018).

[31] armMBED. X_NUCLEO_IDB0XA1. https://os.mbed.com/teams/ST/code/X_NUCLEO_IDB0XA1/ (accessed 27 April 2018).

[32] Mapelli L., MQTT implementation. https://os.mbed.com/users/mapellil/code/MQTT/ (accessed December 2018).

[33] armMBED. EasyConnect. https://github.com/ARMmbed/easy-connect/ (accessed December 2018).

[34] Invidia L., Central node Firmware, https://os.mbed.com/users/lorevee/code/ble-star-mbed/ (accessed December 2018).

[35] st.com. Bluetooth Low Energy Network Processor supporting Bluetooth 4.1 core specification, http://www.st.com/content/st_com/en/products/wireless-connectivity/bluetooth-bluetooth-low-energy/bluenrg-ms.html (accessed December 2018).

[36] STM32 function pack for IoT node Prx-Env, https://www.st.com/en/embedded-software/fp-sns-flight1.html (accessed December 2018).

[37] STM32 function pack for IoT node Mic-Env, https://www.st.com/en/embedded-software/fp-sns-allmems1.html (accessed December 2018).

[38] STM32 function pack for IoT node Mot-Env, https://www.st.com/en/embedded-software/fp-sns-motenv1.html (accessed December 2018).

[39] st.com. STM32CubeMX. http://www.st.com/en/development-tools/stm32cubemx.html (accessed December 2018).

[40] Arduino. https://www.arduino.cc (accessed December 2018).

Luigi Patrono received his MS in Computer Engineering from University of Lecce, Lecce, Italy, in 1999 and PhD in Innovative Materials and Technologies for Satellite Networks from ISUFI-University of Lecce, Lecce, Italy, in 2003. He is an Assistant Professor of Computer Networks and Internet of Things at the University of Salento, Lecce, Italy. His research interests include RFID, EPCglobal, Internet of Things, Wireless Sensor Networks, and design and performance evaluation of protocols. He is Organizer Chair of the international Symposium on RFID Technologies and Internet of Things co-sponsorized by IEEE ComSoc. He is author of about 120 scientific papers published on international journals and conferences and four chapters of books with international diffusion.

Piercosimo Rametta received the Master's degree in Computer Engineering with honors at the University of Salento, Lecce, Italy, in 2013. His thesis concerned the definition and implementation of a novel mash-up tool for Wireless Sensor Networks' configuration. From November 2013 to October 2018 he collaborated with IDA Lab — IDentification Automation Laboratory at the Department of Innovation Engineering, University of Salento. His activity focused on the definition and implementation of new mash-up tools for managing smart environments based on Wireless Sensor Networks and Internet of Things.

Lorenzo Invidia received his BA in Information Engineering from University of Salento, Lecce, Italy, in 2018. His thesis work in the IoT field concerned the BLE technology, the development of iOS applications and the prototyping of embedded devices. He is currently attending a master degree course in Cybersecurity at Sapienza University of Rome.

Andrea Palmieri graduated in Informatics Engineering at the University of Salento. He has been working as Senior Application Development Engineer for STMicroelectronics in Lecce (Italy) since 2003. After an initial experience in the multimedia and networking fields, today he is involved in the STM32ODE (Open Development Environment) program for the development and the promotion of the platforms based on the STM32 MCUs. He is a developer of the software package for the Bluetooth Low Energy expansion board.

Silvio Lucio Oliva graduated in Electronic Engineering at the Polytechnic University of Milan. He has been working as Senior Design Architect Engineer for STMicroelectronics in Lecce (Italy) since 2002.
After an initial experience in the multimedia and networking fields, today he is involved in the STM32ODE (Open Development Environment) program for the development and the promotion of the platforms based on the STM32 MCUs. He is a developer of the software package for the Bluetooth Low Energy expansion board.