Open-source multi-protocol gateway for Internet of Things

Bogdan Oniga
Department of Computer Sciences
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
Bogdan@Oniga.me

Adrian Munteanu
ETRO Department
Vrije Universiteit Brussel
Brussels, Belgium
acmuntea@etrovub.be

Vasile Dadarlat
Department of Computer Sciences
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
Vasile.Dadarlat@cs.utcluj.ro

***Abstract*—Internet of Things (IoT) is becoming increasingly ubiquitous with global applicability in a very wide spectrum of commercial sectors, making use of diverse technologies for custom usage in certain environments. The diversity of applications and protocols and the custom needs for each environment introduce a big challenge for IoT applications, demanding the support of communication and message exchanges in implementations that aggregate diverse protocols. The paper addresses the design of an open-source multi-protocol gateway which supports two communication protocols, namely, Bluetooth Low Energy (BLE) and LoRa. The proposed gateway represents the bridge between the devices and the network server and transports the received messages via a standard format readable by the network server. The messages between the gateway and the network server are transported over the Message Queuing Telemetry Transport (MQTT) protocol. The paper details the implementation of the proposed multi-protocol gateway and highlights its benefits. In particular, the proposed gateway demonstrates support for multiple protocols on the same hardware base and eliminates the need of using multiple hardware components for each type of protocol.**

Keywords—multi-protocol gateway, LoRa, Bluetooth Low Energy, web application, MQTT, Internet of Things.

I. Introduction

Internet of Things involves extending the communication to a new range of physical devices and everyday things, going beyond the classical devices, such as laptops and smartphones. The new devices are collecting environmental data and exchange messages over the internet being remotely monitored and controlled by a central network server. Various types of technologies are available on the global market built to support IoT characteristics and demands, being widely used in IoT architectures and custom environments. Among the existing protocols, Bluetooth Low Energy (BLE) and LoRa technologies are widely used in IoT deployments, representing two of the most powerful wireless technologies in terms of transmission capabilities and power consumption. One of the biggest challenges in such implementations is to enforce interoperability and adaptability of message exchanges between wireless protocols and to cope with the constraints introduced by the existence of protocol differences, architectural designs, transmission rates, inconsistencies between communication layers and specific circumstantial differences. Interoperability between wireless protocols can take advantage of many functions to maximize the efficiency and usefulness of various implementations.

Protocol interoperability has been addressed in the past in custom environments [1] by connecting several devices which operate using different protocols. To obtain such property, multiple infrastructure components need to support various protocols, including the gateways which represent the bridges between devices and network servers. Supporting multiple protocols and normalizing the exchange messages at gateway level is an important aspect when building interoperability between protocols. Research has been conducted in the past into developing multi-protocol gateways for IoT infrastructures, such as IoT gateways capable of interacting with diverse protocols (RF, ZigBee, BLE) [2], an automatic configurable IoT Gateway [3], as well as proprietary products on the global market (e.g. [4]).

In this work, we propose an open-source multi-protocol gateway which supports the aforementioned protocols, namely, Bluetooth Low Energy and LoRa, transporting the messages between the devices and the network server over TCP/IP by making use of MQTT protocol. In addition, we have followed a modular design which permits to easily integrate other software components implementing other protocols.

The main contributions of our research are to (*i*) provide an open-source solution for a prototype of an easily operated multi-protocol gateway which integrates BLE and LoRa, and (*ii*) enable bi-directional communication between the devices and the network server over the Internet through MQTT. In addition, such implementation eliminates the need of multiple hardware components used for each type of protocol, embedding support for multiple protocols on the same hardware base.

The paper is structured as follows. Section 2 presents the open-source gateways built for the BLE and LoRa protocols. Details about the proposed multi-protocol gateway integration are given in section 3. Section 4 summarizes the possible future extensions of the proposed open-source gateway. Finally, section 5 draws the conclusions of this work.

II. Open-source gateways

The main purpose of this research is to develop an open-source multi-protocol gateway capable to handle bi-directional signals from two different wireless protocols - BLE and LoRa, and to create the bridge between devices and the network server when operating over these two protocols. The typical communication distance profiles covered by IoT devices operating over LoRa and BLE are completely different, LoRa claiming much larger distances between devices and the gateway than BLE, thanks to its underlying spread-spectrum communication technology. A multi-protocol gateway brings the advantage of covering both close-by IoT devices operating over BLE as well as IoT devices operating over LoRa and covering a broader radius around the gateway.

BLE and LoRa are key enabling technologies to provide IoT in highly inaccessible areas. The combination of short-range, inter-device communications provided by BLE and the long-range communication provided by LoRa, allows for the implementation of IoT networks in highly inaccessible geographical areas where no telecom infrastructure is anywhere close enough to be of practical use. This aspect is applicable in non-urban areas where the networks of sensors and devices need remote access [5]. One notes that effectively combining BLE and LoRa goes beyond the scope of this paper, the proposed multi-protocol gateway being a first step in this direction.

The gateway presents a modular design which permits to easily integrate other protocols and eliminates the need of multiple hardware components used for each type of protocol, embedding support for multiple protocols on the same hardware base. The management of the implemented protocols can be handled without affecting the functionality of the system.

The proposed multi-protocol gateway architecture is depicted in Fig. 1. The protocols run on Raspberry Pi 3 (RPi3) [16] used as the hardware base component for the gateway. Each protocol module and its devices are managed through the web administration panel deployed on the gateway.

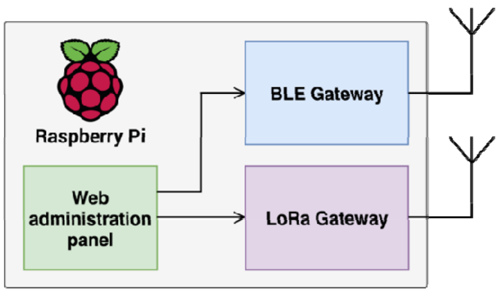


Figure 1. Multi-protocol gateway implemented on a Raspberry Pi base.

RPi3 is the hardware base component for the gateway utilized to manage and sustain bi-directional communication for BLE and LoRa protocols. RPi3 is one of the best options because it is one of the most popular mini-computers, it is actively supported by the open-source communities, it offers powerful hardware characteristics, and is widely used in IoT implementations.

A. *Open-source BLE Gateway*

The hardware component used by the BLE Gateway to communicate with BLE devices is the built-in Bluetooth module supported by the RPi3. As illustrated in Fig. 2, on top of the Bluetooth module we have built from scratch a web administration panel [10] which permits to manage the Bluetooth devices (discover, connect, disconnect devices) and to setup the gateway configuration to allow bi-directional communication with the network server through the MQTT protocol. The web application is built in the *Node.js* [7] programming language and uses the *noble* library [9] as an abstraction layer for the Bluetooth module in the application context.

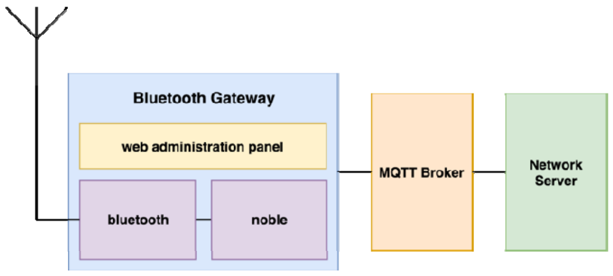


Figure 2. BLE Gateway components and services.

The web administration panel provides two important characteristics:

- It is an easily operated management platform for the BLE devices and gateway connections.
- It supports bi-directional connections between devices and the network server through MQTT.

The application implements two contexts used to manage the BLE gateway configuration and connections, namely:

- BLE Gateway configuration panel (see Fig. 3).
- BLE Gateway devices panel (see Fig. 4).

The configuration panel (Fig. 3) implements a web interface which allows the user to define the BLE gateway parameters such as *MQTT broker address* and *MQTT topics* utilized by the gateway to communicate with the network server. The devices panel (Fig. 4) represents the panel utilized to perform actions such as: *discover* near Bluetooth devices and list their *transmission parameters*, *connect/disconnect discovered devices*, *auto discover* and *auto connect* states utilized to automatically discover near Bluetooth devices and automatically connect the known devices (devices connected in the past) at start-up without any manual interaction.

B. *Open-source LoRa Gateway*

The LoRa gateway is built based on a new component, iC880A - LoRaWAN Concentrator 868MHz [15], which is a proprietary hardware component developed by IMST. This component is certified by the LoRa Alliance and is supported by multiple development boards such as Raspberry Pi, Beagle Bone, and Banana Pi. iC880A is able to receive packets from LoRa devices on up to 8 channels in parallel. As shown in Fig. 5, a complete LoRaWAN gateway can be easily setup by making use of a supported development board and the HAL software [13] [14]. In addition, we are using a LoRa Gateway Bridge [12], which abstracts the *packet_forwarder* UDP protocol into JSON over MQTT and enables the use of MQTT for sending/receiving data to/from the gateway.

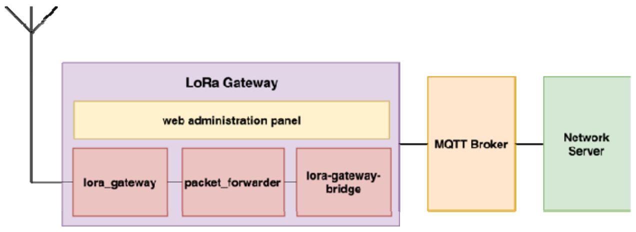
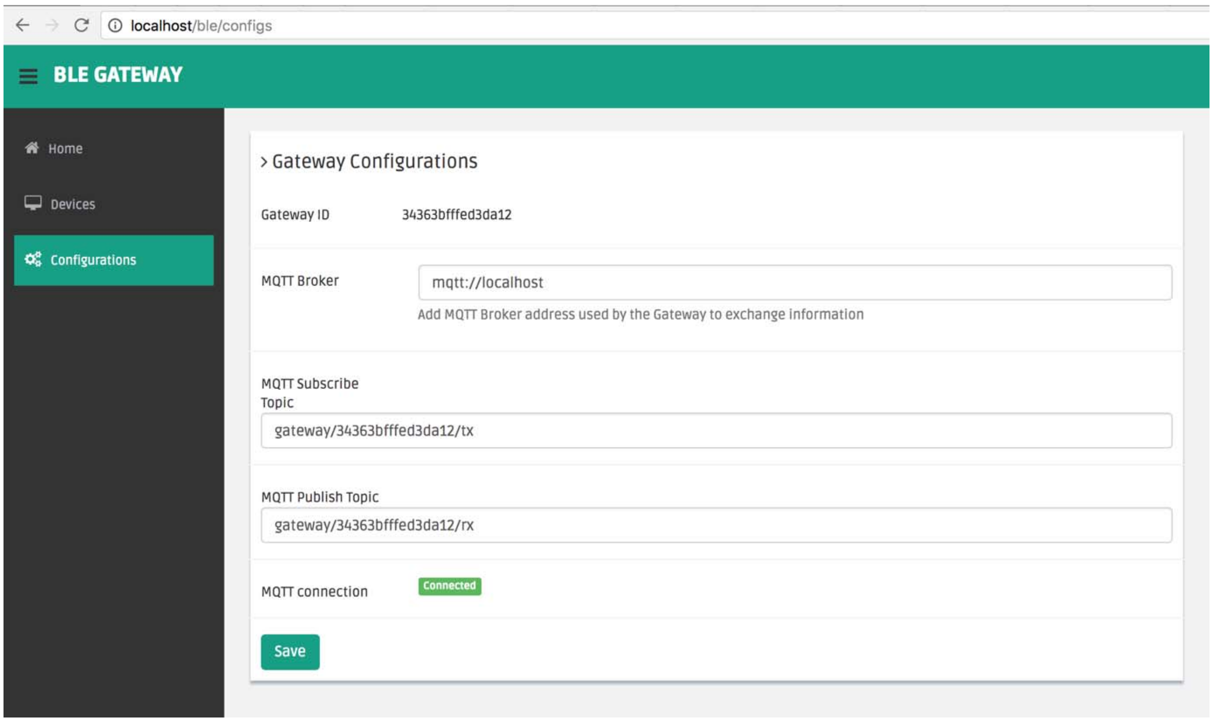


Figure 5. LoRa Gateway components and services.


localhost/ble/configs

> Gateway Configurations

Gateway ID 34363bfffed3da12

MQTT Broker: mqtt://localhost

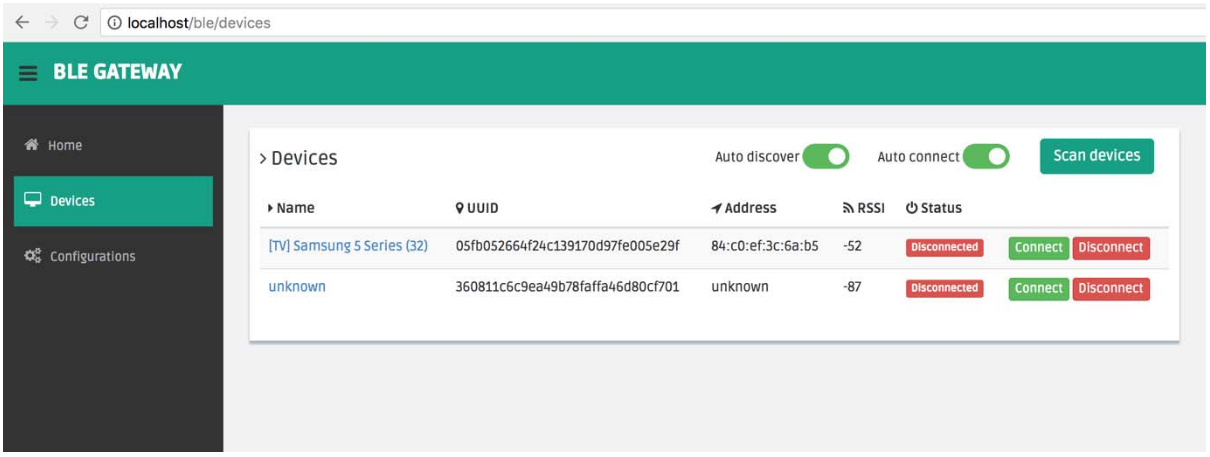
Add MQTT Broker address used by the Gateway to exchange information

MQTT Subscribe Topic: gateway/34363bfffed3da12/tx

MQTT Publish Topic: gateway/34363bfffed3da12/rx

MQTT connection: Connected

Figure 3. Define gateway parameters: MQTT broker address and MQTT topics used by the gateway to communicate with the network server.


localhost/ble/devices

> Devices

Name	UUID	Address	RSSI	Status	
[TV] Samsung 5 Series (32)	05fb052664f24c139170d97fe005e29f	84:c0:ef:3c:6a:b5	-52	Disconnected	Connect Disconnect
unknown	360811c6c9ea49b78faffa46d80cf701	unknown	-87	Disconnected	Connect Disconnect

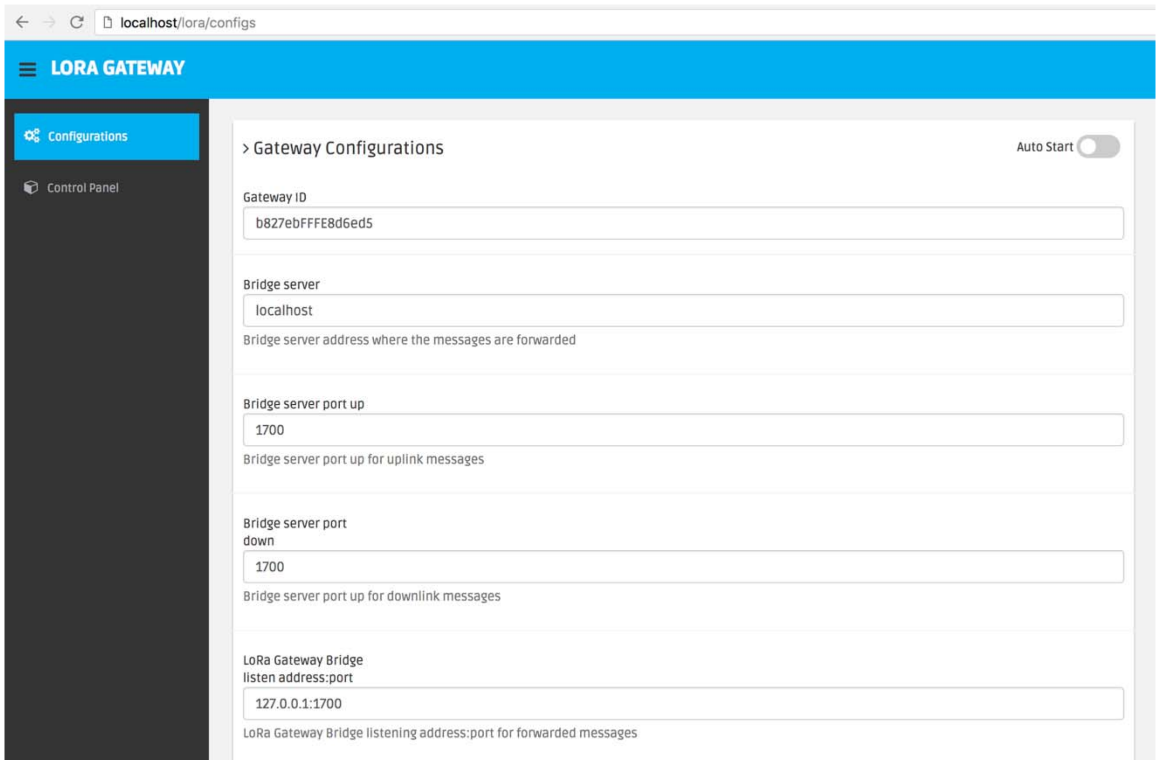
Figure 4. Actions implemented: discover near Bluetooth devices, connect/disconnect discovered devices, automatically discover and automatically connect known devices (devices connected in the past).

On top of the LoRa gateway, we have built a web administration panel [11] which permits to configure, manage and debug parameters of the software and hardware components. The web application is built using the same programming language as for the BLE gateway, namely *Node.js*, sharing the same design interface to provide similar user experience in both implementations.

Two application contexts were built, used to monitor and manage the gateway parameters and connections:

- It is an easily operated management platform for the BLE devices and gateway connections.
- It supports bi-directional connections between devices and the network server through MQTT.

The configuration panel (Fig. 6) implements a web interface which allows the user to define the LoRa gateway parameters such as *gateway id*, *bridge server address*, *port up* and *port down* where the UDP messages are forwarded by the packet_forwarder (in this case the values are *localhost* for address and port *1700* for port up/down as we are using the LoRa Gateway Bridge on the same device), listening *address:port* of the LoRa Gateway Bridge, *MQTT broker address* and *MQTT topics* utilized by the gateway to communicate with the network server. The control panel (Fig. 7) is utilized to control and debug gateway's services and allows the user to *read the current hardware status* (SPI values, LoRa gateway registries) and *perform start/stop/restart actions* of gateway's services (packet_forwarder, LoRa Gateway Bridge).



› Gateway Configurations   Auto Start

Gateway ID
b827ebFFFE8d6ed5

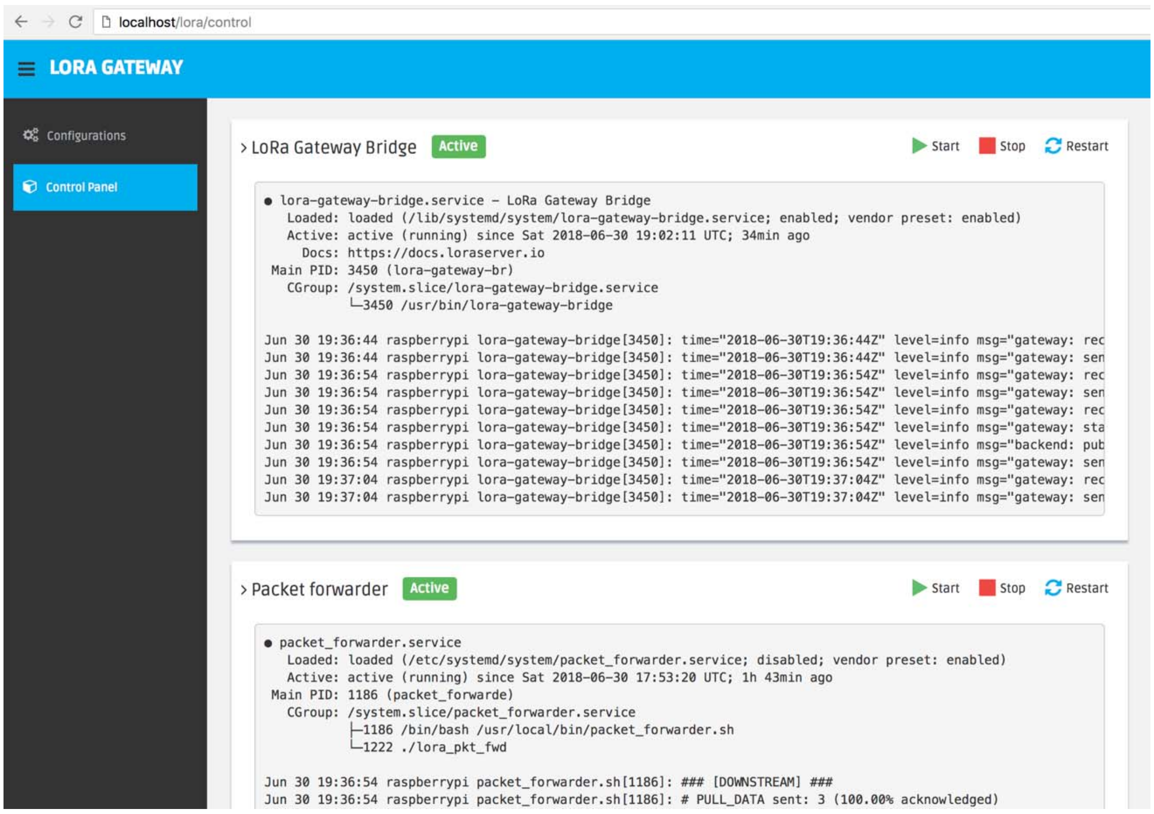
Bridge server
localhost
Bridge server address where the messages are forwarded

Bridge server port up
1700
Bridge server port up for uplink messages

Bridge server port down
1700
Bridge server port up for downlink messages

LoRa Gateway Bridge listen address:port
127.0.0.1:1700
LoRa Gateway Bridge listening address:port for forwarded messages

Figure 6. Define gateway parameters: gateway id, bridge server address, port up and port down where the UDP messages are forwarded by the packet_forwarder, listening address:port of the LoRa Gateway Bridge, MQTT broker address and MQTT topics utilized by the gateway to communicate with the network server.



› LoRa Gateway Bridge Active

```
● lora-gateway-bridge.service - LoRa Gateway Bridge
   Loaded: loaded (/lib/systemd/system/lora-gateway-bridge.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2018-06-30 19:02:11 UTC; 34min ago
     Docs: https://docs.loraserver.io
 Main PID: 3450 (lora-gateway-br)
   CGroup: /system.slice/lora-gateway-bridge.service
           └─3450 /usr/bin/lora-gateway-bridge

Jun 30 19:36:44 raspberrypi lora-gateway-bridge[3450]: time="2018-06-30T19:36:44Z" level=info msg="gateway: rec
Jun 30 19:36:44 raspberrypi lora-gateway-bridge[3450]: time="2018-06-30T19:36:44Z" level=info msg="gateway: sen
Jun 30 19:36:54 raspberrypi lora-gateway-bridge[3450]: time="2018-06-30T19:36:54Z" level=info msg="gateway: rec
Jun 30 19:36:54 raspberrypi lora-gateway-bridge[3450]: time="2018-06-30T19:36:54Z" level=info msg="gateway: sen
Jun 30 19:36:54 raspberrypi lora-gateway-bridge[3450]: time="2018-06-30T19:36:54Z" level=info msg="gateway: rec
Jun 30 19:36:54 raspberrypi lora-gateway-bridge[3450]: time="2018-06-30T19:36:54Z" level=info msg="gateway: sta
Jun 30 19:36:54 raspberrypi lora-gateway-bridge[3450]: time="2018-06-30T19:36:54Z" level=info msg="backend: pub
Jun 30 19:36:54 raspberrypi lora-gateway-bridge[3450]: time="2018-06-30T19:36:54Z" level=info msg="gateway: sen
Jun 30 19:37:04 raspberrypi lora-gateway-bridge[3450]: time="2018-06-30T19:37:04Z" level=info msg="gateway: rec
Jun 30 19:37:04 raspberrypi lora-gateway-bridge[3450]: time="2018-06-30T19:37:04Z" level=info msg="gateway: sen
```

› Packet forwarder Active

```
● packet_forwarder.service
   Loaded: loaded (/etc/systemd/system/packet_forwarder.service; disabled; vendor preset: enabled)
   Active: active (running) since Sat 2018-06-30 17:53:20 UTC; 1h 43min ago
 Main PID: 1186 (packet_forwarde)
   CGroup: /system.slice/packet_forwarder.service
           ├─1186 /bin/bash /usr/local/bin/packet_forwarder.sh
           └─1222 ./lora_pkt_fwd

Jun 30 19:36:54 raspberrypi packet_forwarder.sh[1186]: ### [DOWNSTREAM] ###
Jun 30 19:36:54 raspberrypi packet_forwarder.sh[1186]: # PULL_DATA sent: 3 (100.00% acknowledged)
```

Figure 7. Gateway control and debug features: read the current hardware status (SPI values, LoRa gateway registries), start/stop/restart actions of the gateways's services (packet_forwarder, LoRa Gateway Bridge).

III. Multi-protocol Gateway Integration

As described in Sections II-A, II-B, the BLE and the LoRa gateways are independent implementations in terms of software and hardware components. There are no dependencies between the implementations and they can run one without the other, exposing a modular design which permits to add or remove any other protocols. In our contextual integration, both web administration panels are integrated in the same context and the gateways can be managed and monitored using an uniquely and easily operated web interface. Each gateway exposes a web application running independently and, as part of the integration, a reverse proxy is used to forward the web traffic to the specific required resources (see Fig. 8). As shown in Fig. 9, the reverse proxy is configured to forward the web traffic matching the url *hostname/**lora*** to the LoRa gateway web application and the url *hostname/**ble*** to the BLE gateway web application.

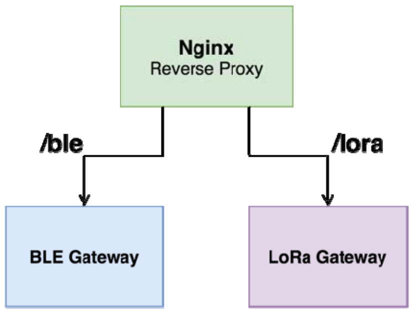


Figure 8. Reverse proxy - forward the web traffic based on required resources.

```
...

server {
    listen 80 default_server;

    server_name  localhost;

    location /lora/ {
        proxy_pass http://127.0.0.1:3000/;
    }
    location /ble/ {
        proxy_pass http://127.0.0.1:3001/;
    }
}

...
```

Figure 9. Reverse proxy - lightweight nginx config file.

Integrating a new protocol gateway implies hardware support for the new protocol and a web administration panel to easily debug and control the gateway services. In case of adding support of a new protocol gateway web administration panel (Fig. 10), the reverse proxy must implement a new path and forward the web traffic, when matching the path, to the running web panel.

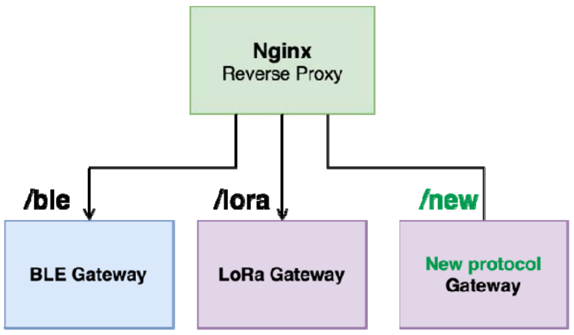


Figure 10. Reverse proxy - adding new protocol gateway web administration panel

Multi-protocol gateway resource consumption

To evaluate the resources required by the proposed multi-protocol gateway, in Table 1 we report the resource consumption when running the LoRa and BLE gateways separately and simultaneously. The ranges of values illustrated in Table 1 are measured in the following circumstances: all the services in place to support gateway management through the web administration panels, the hardware in place to send/receive messages to/from devices. The current consumption was measured using an USB power meter capable to read the voltage and the current absorbed by the hardware.

The ranges of values may vary based on the gateway usage in terms of connected devices, number of simultaneously exchanged messages and web administration panel usage. In terms of resources, there is enough room to integrate other protocols (hardware \& software) for a regular usage of the gateway.

TABLE I. RESOURCE CONSUMPTION VALUES WHEN RUNNING THE GATEWAY SERVICES INDEPENDENTLY AND SIMULTANEOUSLY

Protocol	Free Memory RAM	Current
Initial resource consumption	802.8 MB	0.26 A
BLE gateway running	721.8 MB	0.28-0.30A
LoRa gateway running	723.1 MB	0.32-0.36A
LoRa & BLE gateway running	675.8 MB	0.35-0.42A

* Memory RAM: 927 MB, Voltage measured: 5.08 V

IV. FUTURE WORK

At this stage, the multi-protocol gateway represents an open-source prototype which was devised and implemented motivated by the desire of bringing a cost-free solution and an alternative to the proprietary solutions existing on the global market. As this open-source solution is currently in a prototype stage, multiple important features have yet to be implemented to grow the project maturity.

Currently, the web administration panels does not implement an authentication layer. One of the next steps is to develop a Web Authentication Method (user/password authentication and/or certificate based authentication) to authenticate the users. Also, to provide device authentication inside the network, the BLE gateway has to support MQTT secure communication over the TLS.

Debugging and control features are very important in order to easily handle the gateway functions. The LoRa gateway already implements such features and it is a nice feature to have for the BLE gateway as well. A particularity for the BLE gateway is the information transmitted by the connected Bluetooth devices about their functions, services & characteristics. Displaying such information helps the users to easily read the functions implemented at device level and to interact with the devices through their functions. The connected Bluetooth devices are not authenticated at this stage, hence, another important step is to develop Bluetooth device authentication at gateway level (e.g. by using the passkey method).

An automated solution to install the desired protocol specific gateways represents a major step to grow the project maturity. Such solution reduces the complexity of installing a multi-protocol gateway and abstracts the user interaction with the entire technology stack utilized by the gateway.

Implementing support for multiple protocols (other than BLE and LoRa) will boost gateway's maturity enabling integration with diverse technologies and networks.

V. Conclusions

We have developed an open-source multi-protocol gateway which integrates two important IoT protocols: Bluetooth Low Energy (BLE) and LoRa. The main contribution of our research is to provide an open-source solution for an easily operated multi-protocol gateway which enables bi-directional communication between devices and the network server over the Internet through the MQTT protocol. The proposed multi-protocol gateway represents an open-source prototype implemented from the desire of bringing a cost free solution and an alternative to the proprietary solutions existing on the global market. The proposed solution eliminates the need of multiple hardware components used for each type of protocol, embedding support for multiple protocols on the same hardware base. In addition, we have followed a modular design which permits to easily integrate other protocol implementations.

Acknowledgment

This research has been supported by Fonds Wetenschappelijk Onderzoek (FWO), Vlaanderen (projects G025615 and G084117) and is one of the research topics included in the PhD work of B. Oniga.

References

[1] B.P. Dave, G. Lakshminarayana and N.K. Jha, "COSYN: Hardware-software co-synthesis of heterogeneous distributed embedded systems," IEEE Transactions on Very Large Scale Integration Systems, vol. 7, no.1, pp. 92-104, March 1999.

[2] A. Amiruddin, A.A.P. Ratna, R. Harwahyu and R.F. Sari, "Secure multi-protocol gateway for Internet of Things," Wireless Telecommunications Symposium, WTS 2018, pp. 1-8, Phoenix, Arizona, USA, 2018.

[3] B. Kang, D. Kim and H. Choo, "Internet of Everything: A Large-Scale Autonomic IoT Gateway," IEEE Transactions on Multi-Scale Computing Systems, vol. 3, no. 3, pp. 206-214, July-Sept. 2017.

[4] Sentrius RG186 LoRa-Enabled Gateway, IoT Network Development Wireless Gateway with WiFi, LoRaWAN, and Bluetooth, produced by Laird Technologies.

[5] J. Kaye, "LoRa+BLE puts IoT everywhere on the map," Available online, http://www.mwrf.com/systems/lorable-puts-iot-everywhere-map.

[6] The NGINX Application Platform. Available online, https://www.nginx.com/.

[7] The Node.js JavaScript run-time environment. Available online, https://nodejs.org/en/.

[8] Express.js: minimalist web framework for Node.js. Available online, https://expressjs.com/.

[9] Noble: Bluetooth module library for Node.js. Available online, https://github.com/noble/noble/.

[10] Bluetooth Low Energy (BLE) Gateway. Available online, https://github.com/bogdanoniga/ble_gateway/.

[11] LoRa gateway web panel. Available online, https://github.com/bogdanoniga/lora_gateway/.

[12] LoRa gateway bridge. Available online, https://github.com/brocaar/lora-gateway-bridge/.

[13] LoRa gateway. Available online, https://github.com/Lora-net/lora_gateway/.

[14] LoRa packet forwarder. Available online, https://github.com/Lora-net/packet_forwarder/.

[15] iC880A - LoRaWAN® Concentrator 868MHz. Available online, https://wireless-solutions.de/products/long-range-radio/ic880a.html/.

[16] Raspberry Pi 3 Model B. Available online, https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.