LoRaWAN applications - “Leonardo tasting LoRaWINE”

Joint International Meeting
PRiME 2020
October 4-9, 2020
Attendees register at NO COST!
PRIME
PACIFIC RIM MEETING
ON ELECTROCHEMICAL AND SOLID STATE SCIENCE
2020
REGISTER NOW

LoRaWAN applications - "Leonardo tasting LoRaWINE"

R Maksimova[1] and K Kolev[1]

[1] University of Food Technologies, Technical Faculty, Plovdiv, Bulgaria

E-mails: rossirm@abv.bg, kr_kolev@abv.bg

Abstract. This paper presents main and specific points of application of LoRaWAN IoT technology as a new idea for the realization of projects in various smart technological systems. In the course of implementing such a task, information technologies like - LoRaWAN, TTN and Python with the kind assistance of a framework for building a real-time wine monitoring web application were used. An example dashboard for monitoring of Food Technology processes was developed and provided using a specially assembled end node with appropriate sensors for the subject's needs. A block diagram of an Internet of Things (IoT) network including The Things Network is presented. A diagram of the end node is synthesized. A flowchart diagram of the dashboard's code is suggested. Different program structures for LoRaWAN applications are built in contrast of ready integrations.

1. Introduction

LoRa (Long Range), as a part of modern low-power wide-area networks (LPWAN), is a proprietary spread spectrum modulation scheme developed for radar applications in the 1940's as well as a radiofrequency (RF) carrier signal technology based in the physical layer (PHY – layer 1 in the OSI model) of a telecommunications network [1, 2]. Therefore, via LoRa modem the data can be converted to this signal. Although there are other, more commonly known radio frequency signal technologies (like Bluetooth and WiFi), LoRa is known for its improved transceiver sensitivity and range - covering up to kilometers under good conditions. This makes it suitable for large-scale networking solutions.

LoRaWAN is a media access control (MAC) protocol developed by the LoRa Alliance for large-scale communication acting as a bridge between signals and an application, allowing transmission of data to devices connected to the application [3]. It is a technology based in the data layer of a telecommunications network. Combined with LoRa RF signal technologies, LoRaWAN has given the possibility for creating low-power, cost-effective, long-term and bidirectional telecommunications solutions used in different situations. These benefits and the ability to reduce the number of Gateway devices needed (compared to other methods of communication such as mobile networks and WLAN) make LoRaWAN suitable for Internet of Things (IoT) networks. Its principles of operation allow battery powered devices to communicate with Internet based applications over long-range wireless connectivity.

Application can be every software package that LoRaWAN devices can communicate with on the Internet. For the purposes of this paper The Things Network (TTN) which is a decentralized infrastructure for the IoT can be used for adding applications. There are different types of applications from custom ones to pre-set integrations provided by TTN that could be used without any need for writing a programming code. Such integrations include functionalities like: syncing and managing LoRaWAN devices in Web Services IoT; designing, prototyping and real-time or historical data visualization IoT solutions; collection of location-ready application programming interfaces (APIs) all allowing free prototyping with instant sign-up; integrations for sending uplink data to an endpoint and receiving downlink data over HTTP; publishing uplink data to brokers for a real-time data exchange for the IoT; storage integrations for persisting data in a database for several days, retrieving the data, using API available over HTTPS and serving it in JSON format (for querying data and listing of devices for which data is stored); uploading coverage information to servers directly from the TTN backend side; aggregating and analyzing live data streams etc. Ready integrations are a good option for beginning from somewhere but for a more flexible implementation we could dive into the world of programming, for example some custom web application. A framework suitable in such a case can be the Python's web application framework - Dash by Plotly.

2. Architecture of IoT Networks with TTN

For the implementation of a basic IoT network general building components are required. Among the main hardware components are LoRaWAN gateways and LoRaWAN end devices. Gateways are the bridge between the end devices and TTN, connecting to the latter via a router. The Things Network backend systems are responsible for routing data from the Internet of Things between devices and applications, i.e. TTN is the bridge between LoRaWAN gateways and applications to which devices are connected.

Non-IP protocols like LoRaWAN require some form of routing and processing before delivery of messages to an application. In figure 1 are presented the basic relations between the different components of an IoT network including The Things Network that performs those routing and processing functions in a decentralized and distributed way. For simplicity, the components are only used once, although there may be many-to-many relations. Messages coming from LoRaWAN end nodes are being received by gateways that are the part of the hardware which transmits the radio signal to the backend systems. They are connected to a router which is responsible for managing the gateway status and scheduling broadcasts. Each router is connected to one or more brokers. Brokers are the core of The Things Network. It is their responsibility to map a device to an application, forward uplink and downlink messages up to the correct application and down to the correct router (which forwards them to a gateway).

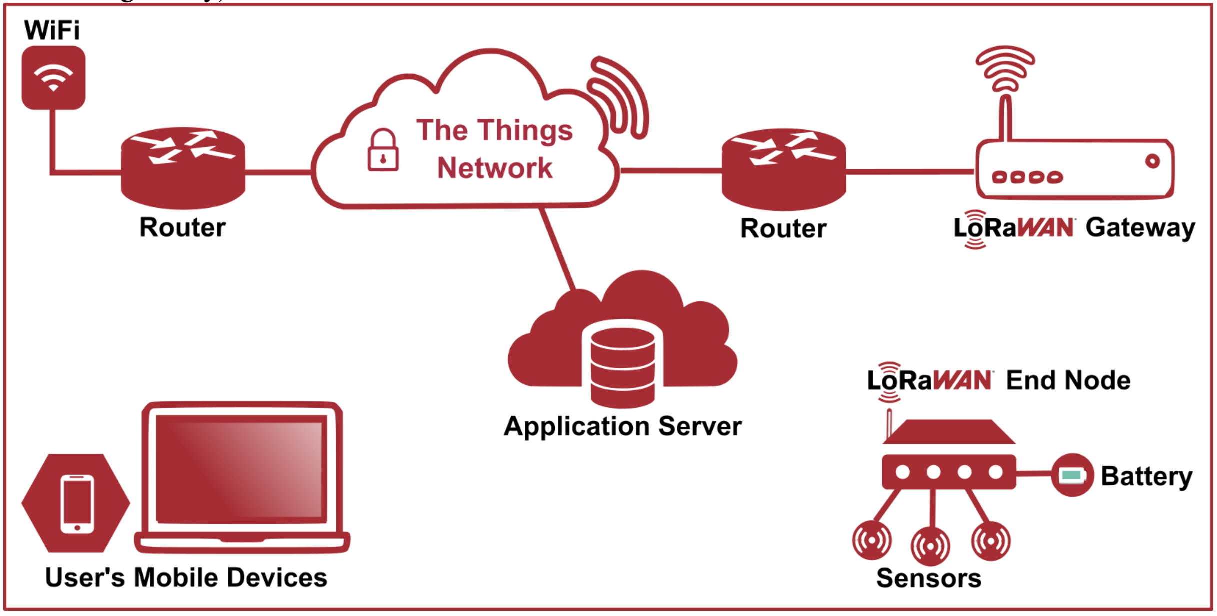


Figure 1. Block diagram of an IoT network architecture

The Things Network aims to be very flexible for deployment options. It prefers the option to contact the community network hosted by The Things Network Foundation or its partners. In this case, the Application connects to a Public Community Network Handler, typically using the Message

Queuing Telemetry Transport (MQTT) API [4]. But it is also possible to deploy private networks performing all of these functionalities in a private environment which keeps the data private, but still being able to use the hosted account on TTN for authentication and authorization.

Network security guarantees the authenticity of end nodes in the network, while the application security ensures that the network operator does not have access to the end-user application data [5]. The Things Network has strong security on the public network that supports end-to-end encryption, mitigations against various attacks and support for different 128-bit encryption keys for every single end device. Users need an account to use The Things Network. On the public community network, user accounts are stored in an account server. These accounts are protected by a password. In private networking setups, users may implement their own means of authentication and authorization. Users can create applications and they can authorize other users to access to applications. Applications are identified by a unique App ID. Each Application has one or more Access Keys to access application data stored on an Application server or to manage devices [6].

Applications and devices can be managed via The Things Network Console, that is a closed-source interface over the open-source TTN command line (CLI) with the same APIs of the open-source server components that applications can use. In addition, TTN provides a User Interface (UI) to its closed-source Network Operations Center to track user's network usage, as well as click-and-run integrations for popular application platforms [7]. There are Brokers and Handlers inside The Things Network which map the traffic of a specific application to be forwarded to an Application server. The Handlers interpret binary data and act like a bridge to higher-layer protocols like MQTT. At the core of things there is a Discovery Server, which is TTN's key to a decentralized architecture. In this server routers, brokers and handlers announce themselves and there they can be looked up [4]. All of these functionalities are performed by The Things Network.

3. Hardware - "Leonardo the taster"

The Things Network supports LoRaWAN for long range (to kilometers, though the strength of the signal is decreased over increasing of the distance and obstacles), low power (months to years on battery) and low bandwidth (51 bytes/message) communication, supports any certified LoRaWAN devices, but also offers its own devices [8, 9]. To connect a device to TTN it needs to have a LoRaWAN module wired or as a shield, which often communicates via serial interface. Currently supported LoRaWAN modules by TTN are the Microchip RN2483 and the RN2903. To save users from the complexity of commands and responses, some modules come with an SDK.

The Things Gateway and The Things Uno board, based on Arduino Leonardo (not Arduino Uno), were purchased and used. The Things Uno (Leonardo in this paper) is fully compatible with the Arduino IDE and existing shields. It is a class A lowest power device (compared to Classes B and C) and supports two-way communication [10]. TTN has built an Arduino SDK which enables transceiving of messages providing an Arduino library which makes it possible devices to communicate with it. Leonardo is assembled with three sensors and acts like a LoraWAN end node. Figure 2 presents a diagram of the main components involved in assembling the end node:

- a DS18S20 Temperature digital sensor - providing 9-bit Celsius temperature measurements, communicating on a 1-Wire bus for data (and ground) with the main microcontroller of the board [11];
- a TCS34725 Color sensor - color light sensor with RGB and clear light sensing elements. An IR blocking filter, integrated on-chip and localized to the color sensing photodiodes, minimizes the IR spectral component of the incoming light and allows color measurements to be made accurately. It communicates via the Inter-Integrated Circuit (I^2C) protocol with the board [12];
- an MQ-3 Gas analog sensor - measuring alcohol. Sensitive material of the sensor is SnO_2 (stannic oxide) which has lower conductivity in clean air. When alcohol gas exists, the conductivity gets higher along with gas concentration rising. Measuring electrode and heater are fixed into a crust made by plastic and stainless steel net. The MQ-3 module has 6 pin, 4 of them are used to fetch signals, and other 2 are used for providing heating current [13].

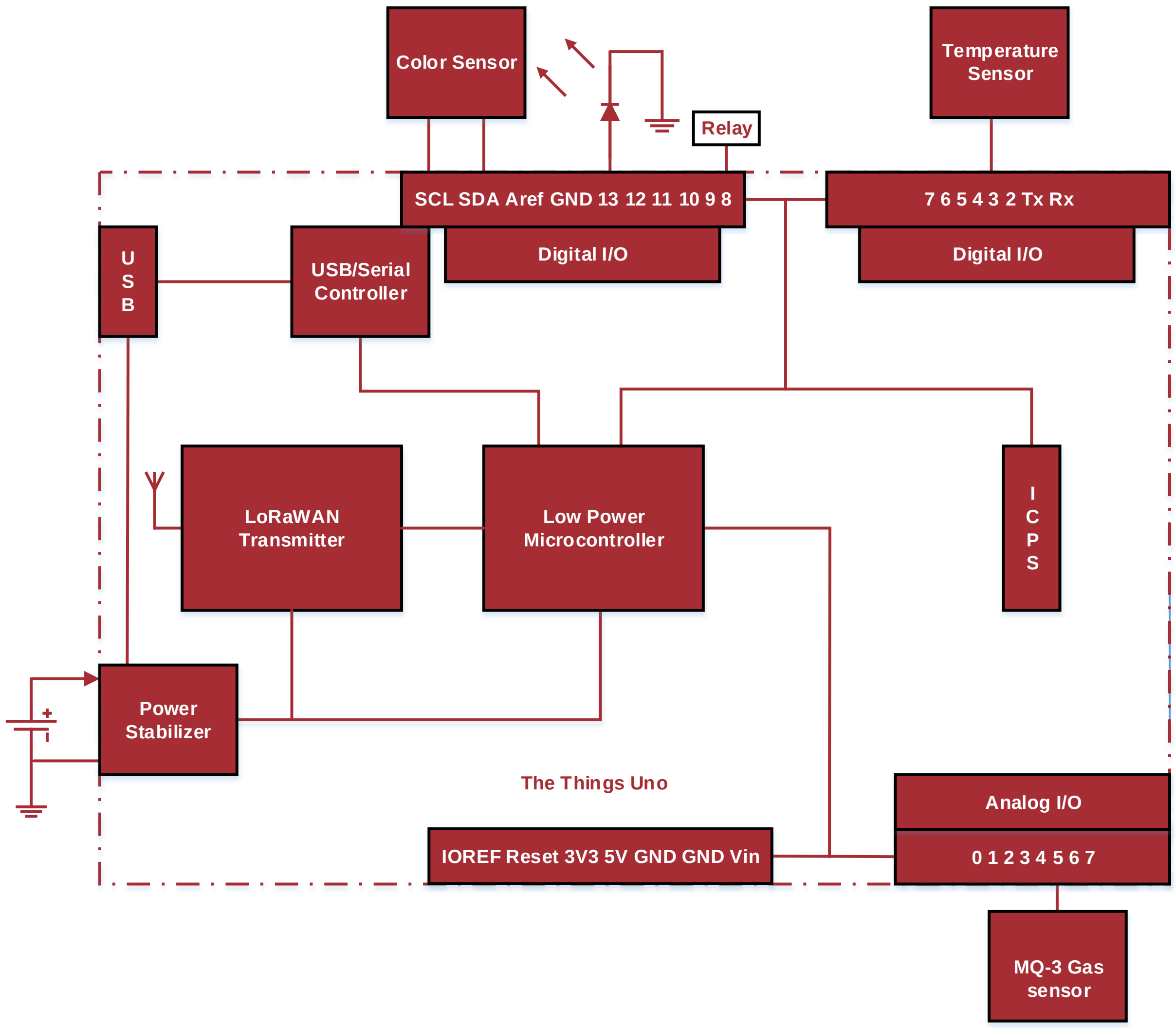


Figure 2. A diagram of our end node

4. Software - "Tasting LoRaWINE"

Before a device can communicate with an application, the latter has to be added to the network and devices to be registered to it. This can be achieved through the TTN Console which is available after creating an account from The Things Network's official website. There are multiple available manners for building applications with TTN:

- working directly with APIs - using standard protocol libraries for direct connection to TTN Handler APIs, sending uplink messages from devices to the application and receiving downlink messages to devices back via the MQTT protocol or managing applications and devices registered to it via the HyperText Transfer Protocol (HTTP);
- via friendly SDKs - providing a layer of abstraction for lighter interaction with the available APIs of TTN;
- via click-and-run integrations - using the same APIs or SDKs an application could use directly. Along with the platform's private or public APIs, they connect the platform-based application to The Things Network. Platform integrations eliminate the need to write code. There are various pre-set integrations available, such as Azure IoT Hub or Amazon Web Services IoT, through which users can manage their applications and devices, making synchronization with The Things Network [14].

Our way for communicating with a LoRaWAN end node via TTN passes through TTN Console:

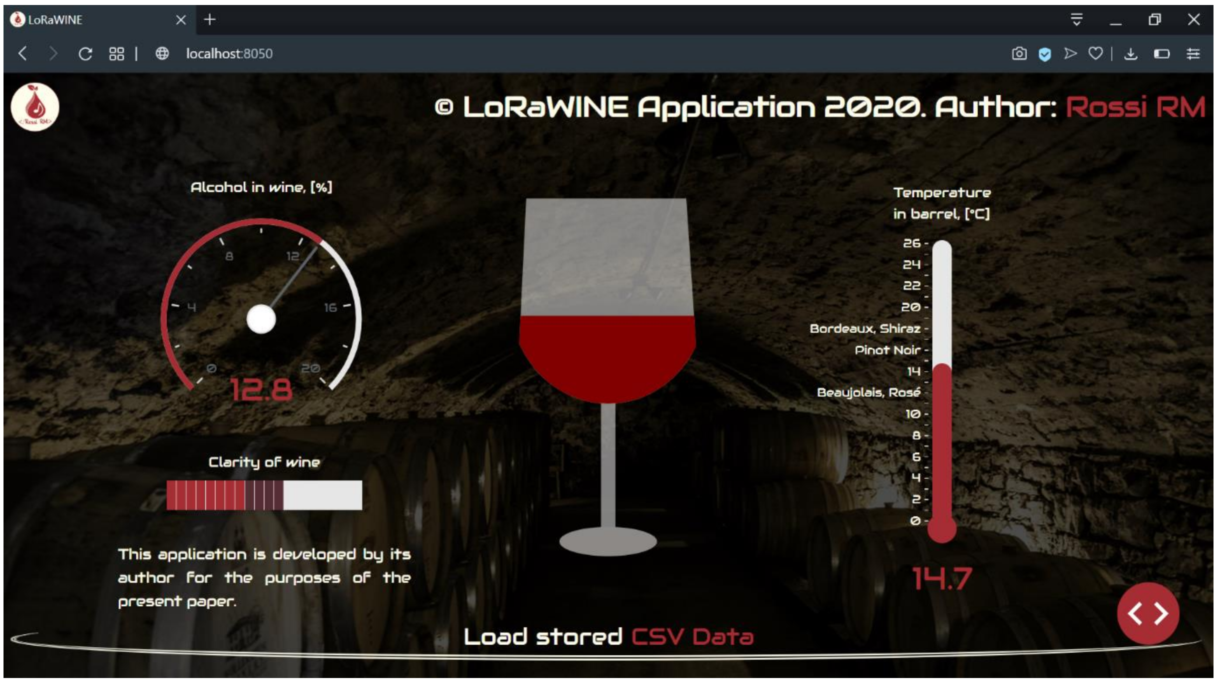
- Registering The Things Gateway;
- Adding an application;
- Registering devices to the application;
- Building the application.

After registering our end device to our application on TTN we can go on building our web application with the kind assistance of the Python's Dash framework, using TTN provided Python Application SDK which allows transceiving messages to and from IoT devices. According to TTN and not only "Python is powerful, and fast; plays well with others; runs everywhere; is friendly and easy to learn; is Open." Python is a challenge for today's computer engineer [15]. The package ttn for Python provides exchanging traffic with The Things Network via MQTT protocol as a client and a manager of applications (for registering, transceiving uplink and downlink messages, generating callback events etc.). This package can be installed by the command "$ pip install ttn" from a command line.

The MQTT protocol, which was invented in 1999, is a publish and subscribe lightweight messaging protocol for low-bandwidth devices, high-latency or unreliable networks. Such features make the protocol ideal for machine-to-machine (M2M) or IoT world of connected devices, as well as for mobile applications [16].

The web application framework for Python - Dash is based on Flask, React.js and Plotly.js. It is suitable for building data visualization web applications with a highly customized UI (User Interface) in pure Python. Applications of Dash are rendered in the web browser running locally on localhost:8050 port of the user's device. It can be deployed to servers and shared via URLs [17]. Dash is nearly platform independent and mobile ready. It is an open-source library with MIT license and can be installed for Python by the command "$ pip install dash" from a command line.

We have developed an exemplary web application and named it "LoRaWINE Application" that acts as a dashboard. A screenshot of the real-time panel of our application in working mode is shown in figure 3. It is intended for providing the ability to monitor different technological conditions in real time, for example conditions in a wine barrel during wine aging stage from the overall technology of wine making.



This application is developed by its author for the purposes of the present paper.

Figure 3. A screenshot of LoRaWINE web application rendered in a web browser

Our dashboard is updating on a predefined interval of time. It listens for callback events on the predefined interval. The data, visualizing on the dashboard, are being fetched by the sensors of our end device put on a suitable place inside the wine barrel. The app pulls live data from TTN via MQTT protocol through the provided Python SDK and parameters like temperature, color, clarity of wine and alcohol content can be updated on panel's display every set updating time. Except the possibility of a real-time monitoring of sensor data, the web application is capable of storing the data into CSV spreadsheets that can be uploaded and previewed any time the user needs some information for past recorded data. This function acts like a database and on the panel is available under the name "Load stored CSV Data" which has the functionality of a button opening a file explorer window for uploading and rendering the created spreadsheet file. Pre-set click-and-run ready integrations of TTN could provide such functionalities but the occasions where components of different nature are combined in a common panel are rare and may cost us a sum when we choose such an option. Figure 4 presents a flowchart diagram tracing the course of logic of our developed web application.

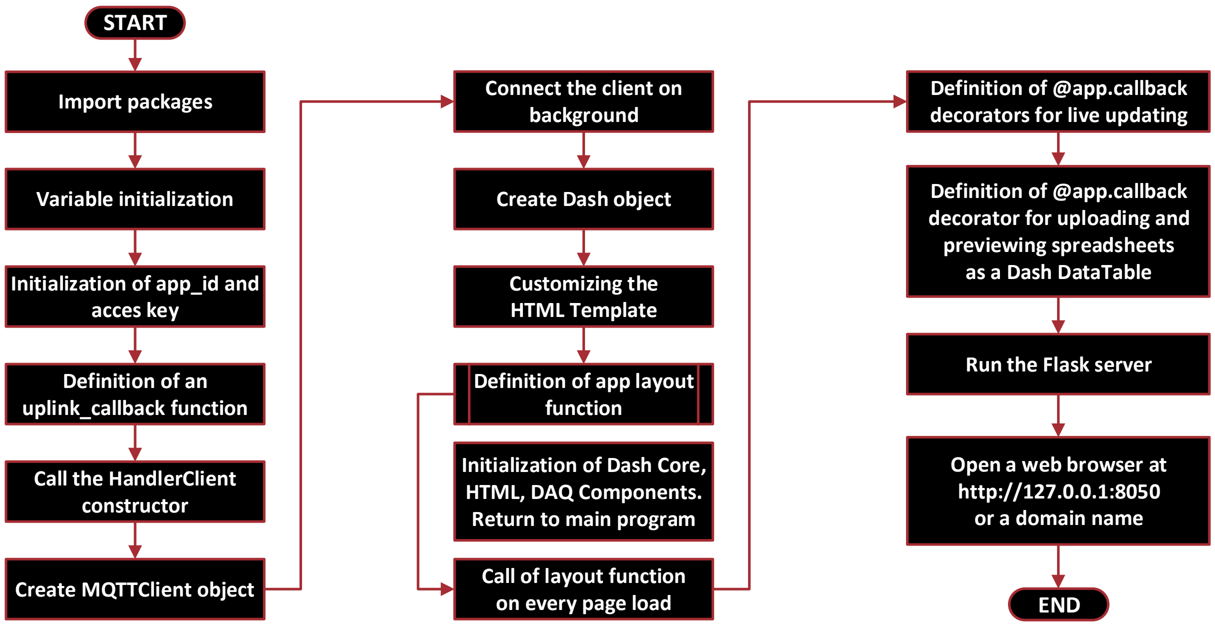

START
Import packages
Variable initialization
Initialization of app_id and acces key
Definition of an uplink_callback function
Call the HandlerClient constructor
Create MQTTClient object
Connect the client on background
Create Dash object
Customizing the HTML Template
Definition of app layout function
Initialization of Dash Core, HTML, DAQ Components. Return to main program
Call of layout function on every page load
Definition of @app.callback decorators for live updating
Definition of @app.callback decorator for uploading and previewing spreadsheets as a Dash DataTable
Run the Flask server
Open a web browser at http://127.0.0.1:8050 or a domain name
END

Figure 4. Flowchart diagram of LoRaWINE web application

The structure of a Dash web application's code can be divided to blocks of code, that are clearly distinguishable and act as separate functional units in the web application performing their own callback events. Such a structure looks like this:

```
@app.callback(Output(component_id='my-wine', component_property='style'),
        [Input(component_id='interval-component', component_property='n_intervals')])
def update_color(n):
  global color
  color = rgb_from_color_sensor
  return color

@app.callback(Output(component_id='my-gauge', component_property='value'),
        [Input(component_id='interval-component', component_property='n_intervals')])
def update_gauge(n):
  global alcohol
  alcohol = gas_percentage_from_mq3
  return alcohol
```

5. Conclusion

This subject's idea is a form of integrating LoRaWAN in the world of Food Technologies and not only. It demonstrates the power and the freedom of Dash web applications programmed in Python and is an example for building custom LoRaWAN applications in contrast of pre-set click-and-run integrations. The powerful available components provided by Dash in Python language are very useful in such an implementation with a scientific mark. Similar ideas as the one in this paper are borrowed from the process of developing larger projects by its authors that can be integrated in interdisciplinary fields from the world of science.

The fact that we use a well-prepared basic technology such as TTN would give rise to a lighter and friendlier usage of LoRaWAN networks. TTN is a good platform for developing novel ideas and projects, and in conjunction with Python and Dash, it becomes a powerful and flexible tool for implementing various scientific tasks. The possibility of assembling different customized end devices based on The Things Uno breathes a life to projects like ours giving its users the opportunity to be next-generation "oenologists". Such a dashboard can live not only within the boundaries of Local Area Networks but also on public communities in Internet accessible from every Internet connected node of the world. Such advantages of custom Dash web applications give the sense of the Internet of Things. Among the main advantages of Dash-based web applications, together with LoRaWAN, over ready pre-set dashboards, are the accessibility and flexibility of the source code.

References

[1] Semtech 2005 AN1200.22 LoRa™ Modulation Basics (Available online: http://wiki.lahoud.fr/lib/exe/fetch.php?media=an1200.22.pdf accessed 27 February 2020)

[2] Penkov S and Taneva A 2019 Algorithm development with LoRaWAN for monitoring system *IOP Conf. Ser.: Mater. Sci. Eng.* **618** (2019) 012007

[3] LoRaWAN Architecture The Things Network https://www.thethingsnetwork.org/docs/lorawan/architecture.html (accessed 29 February 2020)

[4] Network Architecture The Things Network https://www.thethingsnetwork.org/docs/network/architecture.html (accessed 27 February 2020)

[5] Kolev K and Maksimova R 2019 An Approach to Infrastructure for Environment Sensor Network *Proc.: IX International Conference Industrial Engineering and Environmental Protection (IIZS 2019 Zrenjanin, Serbia)* pp 428-434

[6] Network Security The Things Network https://www.thethingsnetwork.org/docs/network/security.html (accessed 28 February 2020)

[7] The Things Network Console The Things Network https://www.thethingsnetwork.org/docs/network/console/ (accessed 28 February 2020)

[8] Devices The Things Network, https://www.thethingsnetwork.org/docs/devices/ (accessed 26 February 2020)

[9] Hidayat M, Nugroho A, Sutiarso L and Okayasuet T 2019 Development of environmental monitoring systems based on LoRa with cloud integration for rural area *IOP Conf. Ser.: Earth Environ. Sci.* **355** (2019) 012010

[10] Tokmakov D, Asenov S and Dimitrov S 2019 Research and development of ultra-low power LoraWan sensor node *Proc.: XXVIII International Scientific Conference Electronics - ET2019 (September 12 - 14, 2019, Sozopol, Bulgaria)* pp 1-4

[11] Maxim 2015 DS18S20 High-Precision 1-Wire Digital Thermometer (Available online: https://datasheets.maximintegrated.com/en/ds/DS18S20.pdf accessed 2 March 2020)

[12] Ams AG 2018 TCS3472 Color Light-to-Digital Converter with IR Filter Datasheet (Available online: https://ams.com/documents/20143/36005/TCS3472_DS000390_2-00.pdf accessed 2 March 2020)

[13] Winsen 2014 Alcohol Gas Sensor MQ-3 manual (Available online: https://cdn.sparkfun.com/datasheets/Sensors/Biometric/MQ-3%20ver1.3%20-%20Manual.pdf accessed 2 March 2020)

[14] Build an Application The Things Network https://www.thethingsnetwork.org/docs/applications/options.html (accessed 25 February 2020)

[15] Maksimova R and Kolev K 2019 Modern Approaches with Python in Sensory Fusion *Proc.: IX International Conference Industrial Engineering and Environmental Protection (IIZS 2019 Zrenjanin, Serbia)* pp 58-66

[16] MQTT http://mqtt.org/ (accessed 26 February 2020)

[17] Introduction to Dash https://dash.plot.ly/introduction (accessed 2 March 2020)