# BlueLab IoT Architecture

Vitor Vaz da Silva

Electronics Telecommunication and Computer Department
ISEL/IPL – Instituto Superior de Engenharia de Lisboa
Instituto Politécnico de Lisboa
Lisboa, Portugal
e-mail: vsilva@deetc.isel.ipl.pt

CTS-Centre of Technology and Systems,
UNL – Universidade Nova de Lisboa,
Caparica, Portugal.
e-mail: vvd.silva@campus.fct.unl.pt

***Abstract*—Connected objects that build up the general idea of Internet of Things (IoT) need hardware and or a software structure to which they attach. There are many IoT solutions that are provided by companies, academia or independent developers. The BlueLab IoT platform is a possible solution for a set of different sensor devices that provide realtime data entries, which are stored in a database. Data entries for each station are organized as raw data sets. Those data sets can be processed later by applications and stored as processed data (P1), which can be further processed and stored producing higher level data. Data sets (raw and processed) marked as shared can then be used to form a data library (datalib). A BlueLab Iot project is composed by several datalibs; thus, the same data can be used by several users and projects without being replicated. Some physical devices that produce data (temperature, humidity, pressure, air quality) have been built and data stored within the BlueLab system, almost continuously for two years, providing a useful data resource to be used as a ground for further BlueLab characteristics. The novel contribution of this paper is the work in progress of the BlueLab system by presenting its architecture and available resources to developers. A hands on example is also presented.**

***Keywords-IoT; CPS; Embedded Systems; BlueLab.***

## I. INTRODUCTION

Almost unstoppable are the things that can now be connected to the Internet as they are built with that intention, and older things that were present before the Internet of Things (IoT) concept can also be connected by a suitable interface; a Thing over Internet (ToI) that dilutes in the global IoT domain [1]. There are many different ideas of how to connect devices to each other, how to cooperate, synchronize, exchange, store and analyse data [2]. The BlueLab IoT system provides a platform for users to add their devices and store sensorial data. The data can also be visualized and processed to offer meaningful information. IoT systems need to consider security issues [3] and this awareness is also present on the BlueLab IoT system including user data and devices. The paper continues in section II with the system's architecture, both hardware and software, followed by section III where tested hardware has been used with different configuration sets in the stations, and then section IV with results and discussion from the whole system.

## II. ARCHITECTURE

The BlueLab architecture has two domains, the logical and physical. The logical domain is composed by the conceptual components that build up the BlueLab and allow software interfaces to be built. The physical domain is mainly the hardware part of the BlueLab's overall system, which includes the possible communication and interconnection configurations.

### A. Physical Architecture

The BlueLab IoT physical architecture is shown in Figure 1. The stations are fixed or mobile, have sensors and Wi-Fi connections. Data gathered is sent to the Database either directly through the Wi-Fi connection to a Gateway (or router) or by sending it through a Distributor (under development). A Distributor is a First In First Out (FIFO) data buffer system; it gathers data from specific stations and sends that data to the Database through a Gateway, whether through a Wi-Fi or cable connection to a Gateway. Stations are data sources and do not need data from other stations to provide their own data; they do not communicate with each other. In case of failure of communication with the server, it is the station (and or the Distributor) that has to cope with it; and data loss may eventually occur.

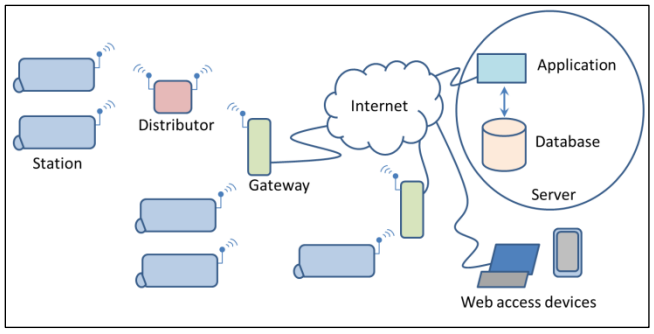


Figure 1. BlueLab's physical architecture.

The stations shown in Figure 1 can also be a smartphone with the BlueLab IoT system. Also shown in the figure are the user Web access devices, which use an Internet browser to communicate with the BlueLab Application where it is possible to manage the system by logging in successfully [4].

 ISBN: 978-1-61208-771-9 

### B. Logical Architecture

To get access to the BlueLab system, a user login interface is needed. The user login interface is modelled in a database and is shown in Figure 2.

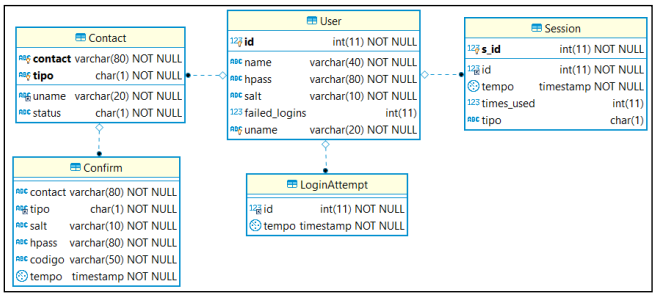


Figure 2. Database model for User access and authentication

A user account must have a contact which can be a phone number or an email and a name associated to it. Several strategies to recognize the authentic user are implemented, like password and associated hash procedures, email confirmation, phone confirmation by Short Message Service (sms), and failed login attempts within a time interval [5][6]. Those strategies are supported by the User, Contact, Confirm and LoginAttempt structures shown in Figure 2. For every valid login there is a random session number *sessionId* that has to be used in subsequent calls. The session is supported by the Session structure of Figure 2. A user may have more than one valid session. Each session may have a time limit, after which another login has to be issued to retrieve a different session number. A session may also be invalidated if a determined number of calls are exceeded.

Each user has a unique *uname* which is used as the database schema name which is associated to that user. After the creation of a new account the user has to build up the environment. The environment is composed by one or more stations; devices that are able to communicate with the BlueLab system and send values to be stored in the users' database schema. Each value is stored as a key value pair in the entry structure as shown in Figure 3.

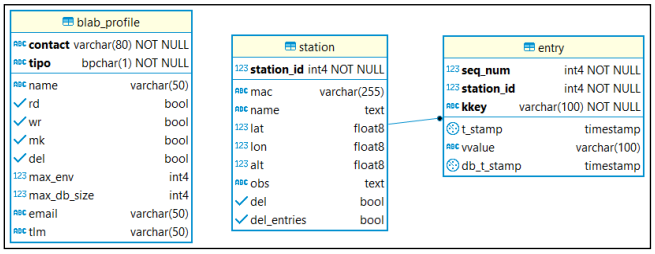


Figure 3. User profile and environment structures

A station has a structure where its identification is stored. The *station_id* is unique for each user and is system defined. The mac is user defined and it is not necessarily a mac address, it could be for example a phone's IMEI; and is unique for each user. Although latitude (*lat*), longitude (*lon*) and altitude (*alt*) are values that identify the position of a station, the station can still be mobile, and if needed the lat, lon and alt values can be sent and stored as an entry like any other sensor value. Any entry is identified by the station where it came from, the sequence number and a key, which can be the name of the physical reality that is being stored; e.g.: "temp". For that key the associated value is also stored; e.g.: "24.3", but the units are not, although the key can be used to hold the units; e.g.: "temp C". By using strings to store values the data type is not necessary because the context is in the key, and it allows storage to be uniform for all entries. Sequence numbers start at 1. Associated with an entry there are two timestamps, one belongs to the time at which the variable was sampled, and the other, database timestamp, the time at which the value was stored. A station that has different sensors and sends all values on the same frame will have all the entries with the same sequence number and same database timestamp. A station that uses the BlueLab system is not obliged to have a Real Time Clock (RTC) or any other time counting procedure, or it may or not have a buffer to hold samples prior to their sending to the database; which will in any of these situations have a significant difference between the station's timestamp and the database timestamp, besides the possibility of being in different timezones. Thus, it is easy to show or search for values belonging to keys in the time series using the sequence numbers and or the timestamps. The sequence number also ensures that the same value is not stored twice due to failures in the communication or the station, and also it is used for delivering values in order when the database is searched. A station can store entries without sequence, which means that there will not be a time series for those entries and they always have the last value. Those key value pairs are stored with sequence 0; values and their timestamps' are updated accordingly. This functionality can be used for the station to store values that it might use afterwards, like a memory. It can be used by the user to set parameters that are used by that station; and likewise values that a station communicates with the user: e.g., status or a sensor value. All these processes are asynchronous.

Within the database schema of a user there may exist one or more profiles supported by the *blab_profile* structure of Figure 3. These profiles help the stations to login into the BlueLab system. Logging in can be accomplished by the usual contact (email or phone number) and password, or only by a direct access code, which can be 80 characters long. By using the direct access code, a station does not need to store the user email or phone and password, which could jeopardize the users' account and entire database, should that station be captured for mischievous purposes. The user can change the direct access code at will. The same direct access code can be used by more than one station at a time; this kind of aggregation is a simple way to create a domain of stations. Each profile has flags that are set by the user when using the main profile and those flags indicate the privileges that that profile has for reading, writing or deleting data.

 ISBN: 978-1-61208-771-9 

BlueLab IoT users can also share their data, or organize it in order to build up the architecture and provide different layers of meaning. For that a project has to be created as shown by the structures in Figure 4.

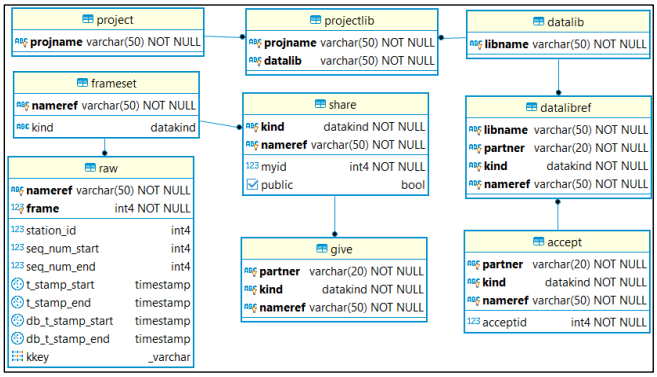


Figure 4. Project and data sharing structure

A user may have several projects, each one known by its unique name, and a project may use data libraries, as shown in the structures of Figure 4. Each data library has a unique name and references data which belongs to the user or to another user, known as partner. A partner is a BlueLab user that shares data; gives to or accepts from other users. A user does not share data that belongs to others; i.e. a user cannot give to a third user what was accepted from a second user. The partner identification is the same value as that of the database schema name, *uname*, referenced above in Figure 2. To share data acquired directly from the stations a user must build a raw set of frames. The raw is a set of selected frames from a station. A frame is the description of the data between two sequence numbers and or two timestamps (device and or database). All raw sets to be shared are on the share structure, and available to be given to partners and used by the user by adding them to the *datalibref* structure. The *datalibref* structure also holds the shared descriptions from the partners that were selected and stored on the accept structure. All this sharing strategy does not involve copying or storing data. When the data is needed, a request using the shared raw description is made to the original database schema.

Under development is the possibility of building processed data, which is data that results from processing raw data, from one or more stations, and eventually from partner's shared data. The reference to the processed data is also through the frameset and share structures (index *kind*). It is planned to add different layers of increasing meaning of processed data, P1, P2 and so on.

A station may issue an alarm which results on the sending of a sms by the BlueLab system. The sms alarm service is payed beforehand and credits or a number of available sms are stored in the operator structure shown in Figure 5. The structure allows support for several operators and offering them as payed services. Messages will be kept for the user to browse through and eventually delete the selected ones.

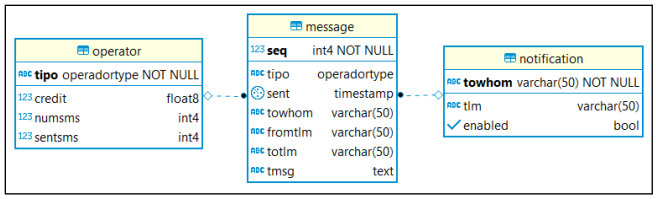


Figure 5. Alarm service structure

All destination sms numbers have to be previously entered and enabled in the notification structure of Figure 5 where they are identified by a token (*towhom*). Thus, a station can only send alarms by naming the destination token. A pre-existing token is "self", which means the users' phone (*tlm*) of the profile that was used for login.

## III. MATERIAL AND METHODS

Several stations have been built using ESP8266 and ESP32 developer modules with the Arduino SDK. The stations are portable, and their power supply input is of 5V. The fixed stations are continuously connected to the mains electrical power distribution. Other stations use power packs or solar energy that charges a li-ion battery. Shown in Figure 6 is a fixed station out of its containing box.

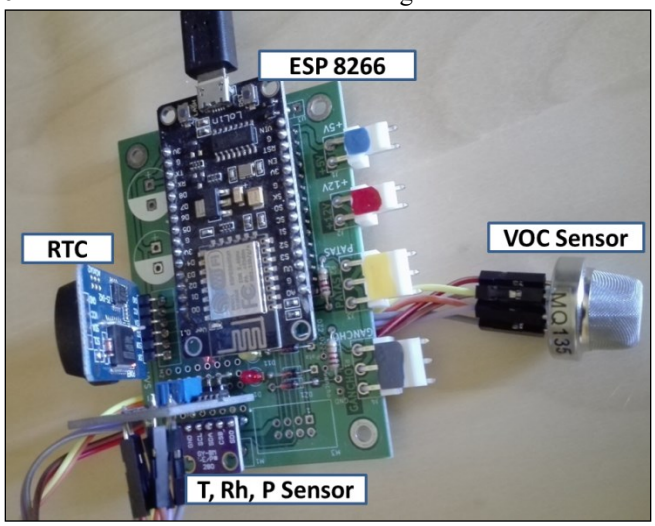


Figure 6. BlueLab IoT fixed station for ambient variable sensing

The device sensors are from a diverse set of sensor modules. The BME280 module provides temperature, relative humidity and air pressure values, and uses an I2C protocol for data retrieval [7]. The TLC555 module has a circuit that measures a capacitive moisture sensor for soil and translates its' value to an analogue output (0-3V), which has a response curve $moisture = A * response^{B}$ with constant values $A$ and $B$ found by calibration procedures [8]. An Hall effect current sensor with a 20 A range, ACS712, with analogue output (0-5 V) directly proportional to the sensed current [9]. Air quality sensor MQ135, that can detect ammonia, sulphide, benzene series steam, smoke and other toxic gases [10][11]. The circuit for the air quality sensor can

 ISBN: 978-1-61208-771-9

also be used for similar type of Volatile Organic Compound (VOC) sensors [12].

The fixed station of Figure 6 is composed by an ESP8266 microcontroller, a RTC with battery, a MQ135 sensor and a BME280 module. This fixed station is acquiring in-house data since July 2018. Note that the sensor's calibration *A* and *B* values can be stored with sequence number 0, defining them as constants with an appropriate identification string.

There is a hands on example at github [13]. The code can be downloaded into an ESP8266 or ESP-32 microcontroller and it uses as sensors a digital input, and an analogical input, which is floating if not connected, so that touching it will produce different readings. On the above mentioned user link there is an Android app that can use the phone's light sensor and the latitude, longitude and altitude values of the GPS to send to the BlueLab IoT system; everything can be safely deleted.

## IV. Results and Discussion

Two graphs are displayed in Figure 7 showing the values stored by the fixed station in a house from 25.08.2019 00:00 till 07.09.2019 23:59, totalling 12526 samples each.

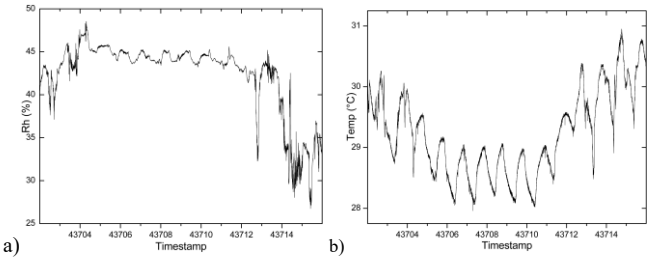


Figure 7. Graphs for a) relative humidity and b) temperature (ºC) values

Samples were obtained every 3 s smoothed by a 4-point moving average filter, for all sensors. The resulting waveforms were then sampled every 90 s and its values sent through the communication link to the BlueLab IoT database. This continuous process can then be searched, and displayed as shown in Figure 7. The data displayed is classified as raw. This raw data could then be processed (not yet done) on a first stage and be classified as P1. For example, from the figures, the circadian rhythm can be extracted, and characteristics like the min, max and mean values stored as P1. By interpretation of the circadian data, it would be easy to conclude with high probability that during 7 days the house had no human intervention; maybe occupants were on holidays.

## V. Conclusion

BlueLab Iot system is a possible solution for developers to add a station that gathers data in any format, stores and retrieves it from a database. It has been working continuously since July 2018. All improvements made to the system since then have not compromised the acquired data. Frequently, tests are made by adding different devices with several combinations of sensors, and power requirements, including battery packs and solar panels. Users can access the BlueLab IoT system using an Internet browser and configure their profile and options, and can also visualize and delete their data received from their stations. Data can be shared among users of the system; this allows a user to include on a project its own and shared data. Presently each user has a different schema on the same database, but in future, each user can define its own database on different servers. A station is a device with the BlueLab IoT system, for example a smartphone application, or an embedded system or a computer program that gathers data from several sources and sends them to the BlueLab Iot system. As future work, it is planned the processing of data to produce P1 framesets, real time use case scenarios, and improvement of the user interface.

## References

[1] T. Kramp, R. van Kranenburg, and S. Lange, "Introduction to the internet of things," in *Enabling Things to Talk: Designing IoT Solutions with the IoT Architectural Reference Model*, 2013, pp. 1–10.

[2] J. Mineraud, O. Mazhelis, X. Su, and S. Tarkoma, "A gap analysis of Internet-of-Things platforms," *Comput. Commun.*, vol. 89–90, pp. 5–16, 2016.

[3] S. Vashi, J. Ram, J. Modi, S. Verma, and C. Prakash, "Internet of Things (IoT): A vision, architectural elements, and security issues," in *Proceedings of the International Conference on IoT in Social, Mobile, Analytics and Cloud, I-SMAC 2017*, 2017, pp. 492–496.

[4] V. Vaz da Silva, "BlueLab IoT," 2020. [Online]. Available: https://bluelab.pt/iot. [Accessed: 21-Jan-2020].

[5] P.-H. Kamp, "LinkedIn Password Leak: Salt Their Hide," *Queue*, vol. 10, no. 6, p. 20, Jun. 2012.

[6] A. Conklin, G. Dietrich, and D. Walz, "Password-based authentication: a system perspective," in *37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the*, 2004, pp. 1–10.

[7] "BME280 - Combined humidity and pressure sensor (and temperature) datasheet," *Bosch*, 2015. [Online]. Available: https://www.digchip.com/datasheets/parts/datasheet/1727/BME280-pdf.php. [Accessed: 29-Sep-2019].

[8] R. Radi, M. Murtiningrum, N. Ngadisih, F. S. Muzdrikah, M. S. Nuha, and F. A. Rizqi, "Calibration of Capacitive Soil Moisture Sensor (SKU:SEN0193)," in *2018 4th International Conference on Science and Technology (ICST)*, 2018, pp. 1–6.

[9] V. Miron-Alexe, "Comparative study regarding measurements of different AC current sensors," in *2016 International Symposium on Fundamentals of Electrical Engineering (ISFEE)*, 2016, pp. 1–6.

[10] "MQ135 Semiconductor Sensor for Air Quality," *Winson*, 2015. [Online]. Available: https://www.winsen-sensor.com/d/files/PDF/Semiconductor Gas Sensor/MQ135 (Ver1.4) - Manual.pdf. [Accessed: 30-Sep-2019].

[11] K. Vandana, C. Baweja, Simmarpreet, and S. Chopra, "Influence of Temperature and Humidity on the Output Resistance Ratio of the MQ-135 Sensor," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 6, no. 4, pp. 423–429, 2016.

[12] C. Durán, J. Benjumea, and J. Carrillo, "Response Optimization of a Chemical Gas Sensor Array using Temperature Modulation," *Electronics*, vol. 7, no. 4/54, 2018.

[13] V. Vaz da Silva, "BlueLab IoT at GitHub," 2020. [Online]. Available: https://github.com/tektonia/bluelab_iot. [Accessed: 21-Jan-2020].

 ISBN: 978-1-61208-771-9 