Internet of Things Laboratory

João Gonçalo Vieira Saramago
joaosaramago@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

November 2021

Abstract

With the growing use of the Internet of Things in the most diverse areas of work and leisure, it is important to offer IST students the possibility of receiving education in this area. The universities most interested in this field already have labs with microcontrollers, small computers, and wireless networks to interconnect devices. Besides networks, several have IoT platforms to manage communications, devices, and access to their data. Thus, to create an IoT lab at IST it is important to implement one of these platforms. After analyzing the open-source options available, ThingsBoard was the one that proved to be superior and was chosen to be implemented in the lab. The options of microcontrollers to be used to communicate with the platform were analyzed, and the ESP32 was the selected one. For students with little to no experience with microcontrollers, a library was created to simplify their communication with ThingsBoard. In addition, sets of sensors, actuators, and one microcontroller, called SensorBoxes, were created for installation and demonstration at the IST campus. Performance tests showed that the platform is adequate for the intended use, although the server provided by IST, where it was installed, had insufficient resources to use the platform with some intensity. Finally, it was possible to adapt existing lab work where two students would have to establish wired communications between microcontrollers. Using the platform these communications could be established over the internet, being an added value in situations like the one experienced during the COVID-19 pandemic.
Keywords: Internet of Things, Laboratory, ThingsBoard, ESP32

1. Introduction

The Internet of Things is the concept where different types of objects connect to the Internet, can communicate with each other and be controlled remotely. It aims to facilitate and optimize people's lives by automating different types of scenarios, both personal and professional. It is best known by its English name, Internet of Things, or its English abbreviation, IoT.

To connect devices to the internet we can use an IoT platform. This allows to connect and manage devices such as sensors and actuators, store, visualize and process the collected data. It can also control access to this data and other logic related to the application area. One of the many use cases can be managing large fleets of vehicles through the platform, to which GPS-enabled devices will be sending their location data. A user-driven mobile IoT application, for example, to control a smart home, fetches real-time or previously saved sensor data from the IoT platform and controls the home's actuators. Automations can also be created on the platform so that it triggers actuators autonomously depending on sensor data. It is an important tool in several areas of engineering, both for process optimization, using sensor data to understand what happens over time, and for creating new ideas and products to put on the market facilitating or adding something to consumers lives. It is something that students, especially in the area of computer science, should learn about during their higher education.

The main motivation of this work is to start creating the infrastructure for an IoT Lab at Instituto Superior Técnico (IST). Bring the possibility of managing sensors and actuators connected to the Internet through an IoT platform. Allow students to access sensors located around the campus and use them for assignments and projects in various disciplines. This access can be made inside or outside IST, so students anywhere in the world can access this network of sensors and thus continue their work. Besides access to the existing devices, students will also be able to create their own and integrate them into the network. An infrastructure like this brings new possibilities to several disciplines and courses, mainly to the Master of Computer Science and Engineering (MEIC). However, many courses at IST already have some programming component in their curriculum due to the influence of computing in all fields of work nowadays. Thus, other courses can also take advantage of this infrastructure. A Civil Engineering course may want to mount several sensors in buildings to perform studies. An Environmental Engineering student may conduct studies on environmental impacts by collecting data 24/7 and later analyzing it. Industrial Engineering and Management students may develop projects to monitor production lines, working according to Industry 4.0. Yet another example can be Chemical Engineering students to study laboratory environments, the behavior of solutions over time or work they are doing, for example monitoring water purification. Many possibilities exist in all areas and the availability of such a platform at IST would bring the opportunity for students from all courses to experience what is not only the future but also the present. Thus placing IST students at the digital forefront as they enter the professional world.

2. State of Art

This section presents the current state of IoT labs in higher education, the main technologies used in this work, and why they were needed.

2.1. IoT Labs in Higher Education

With the growing use of the Internet of Things in various areas of work and leisure, more and more universities are trying to introduce their students to this area. To that end, they create IoT labs focused on the technological exploration of the area, where students explore the technology behind the communications for IoT, and IoT labs focused on applicational exploration, where students seek to improve or use existing technology to create new IoT-related products for consumer use, thus bringing greater comfort and security to their lives. By creating these labs, universities can bring a more suitable environment for their students to carry out their semester projects and for thesis development. In addition to the labs, universities also allow other parts of the buildings to be automated, such as lights [1], so students can not only explore the area but also put it into practice in the real world, close to them.

One of the main things that the labs from other universities have in common is the availability of a wide range of hardware for students to use. This includes various microcontrollers, such as Arduinos and ESP32, and small development computers [2]. There are also various types of sensors and actuators for students to put their projects into practice.

For students who want to create IoT projects without any knowledge of electronics, just programming skills, there is the more applicational side of these labs. One of the options usually made available to students in this situation is a sensor network, which once up and running will always be sending its sensor data to a server, the latter then being able to share the stored or real-time data through its API. IoT platforms are used for this type of functionality. Among the labs that use this technology to connect devices, some opt for paid and renowned versions such as the Amazon, Microsoft, or Google IoT platforms, or for platforms from their labs' sponsors, which influences the choice, but some also go for open-source options, which for the work to be implemented is the most appropriate. Labs that don't use IoT platforms to manage communications usually only provide a communications network, for example, LoRaWAN, where devices can communicate with each other, but for messages to be managed and stored you have to create that functionality in your own application.

2.2. IoT Platform

In order to choose the IoT platform that will be used, several important factors were established. Being an open-source and free project is the main point, it should have a large community, active development, good documentation, SDKs for several platforms, control and administration panel, and finally, support for local installation.

From the beginning, paid platforms or platforms with limited free plans were excluded, such as AWS IoT, Google Cloud IoT, Arduino IoT Cloud, among others. An open-source software has several advantages such as no costs to use it, more privacy, more flexibility in terms of adding features, not being dependent on the existence and interests of a company, among others. Considering that open-source software is created by its community it is very important that the chosen IoT platform has a large community behind it.

Among the several options available for IoT platform, the ones that stood out the most, considering the defined criteria, were ThingsBoard, Kaa IoT, OpenRemote, and SiteWhere. Taking a closer look at these options, it was found that ThingsBoard, even though it is the newest IoT platform it is by far the most popular one, having the highest number of Stars in its GitHub repository. In second place is the Kaa platform with approximately 14% of ThingsBoard's number of Stars, third is SiteWhere and last is OpenRemote. ThingsBoard's growth trend has also been much higher compared to the other options.

Looking at the activity of the repositories in the 31 days prior to September 14, 2021, we see that, as in the previous comparison, ThingsBoard stands out quite a bit from the others with 158 commits, while Kaa has 8, OpenRemote 55, and SiteWhere 0. All the platforms in question have dashboards to manage devices, users, view data, among other features.

In the past, others have compared the existing open-source IoT platforms. Leonidas Otalora compared in May 2019 the various options in terms of available features and integration of the platforms with other devices, and already at that time concluded that ThingsBoard was the best platform to use, excelling in all points of comparison, including usability and available documentation[3]. Eduard Bitencourt and Willian Anjos, in August 2018, also developed a project based on ThingsBoard, having compared it with other platforms and coming to the conclusion that it was indeed the best open-source platform to use[4].

The fact that the ThingsBoard[5] website indicates that several well-known companies are using its platform shows its maturity and that it is the best choice, respecting the criteria, for bringing this work into a professional environment. Among the various companies, we find T-Mobile, Bosch, Prosegur, and INSYS.

2.2.1 ThingsBoard

ThingsBoard[5] is an open-source IoT platform. It allows you to interconnect and manage devices such as sensors and actuators, store, visualize and process the collected data, making it possible to create new applications based on that data. It is designed to be scalable, fault-tolerant, robust, efficient, customizable, and durable. Devices communicate directly with the platform using MQTT, HTTP, or CoAP, however, gateways can be used to translate other protocols to MQTT in order to be interpreted by the system, such as Modbus, OPC-UA, and LoRa.

There is a free version (Community Edition) and a paid version (Paid Edition), however, the free one has all the features necessary for this work and one of the goals is to use a free open-source platform.

2.3. Microcontrollers in IoT

With the need to make many objects smart and put them in many different scenarios, three important points were considered when choosing microcontrollers for IoT in an academic environment. Being low cost, since it is intended to be used by people with little experience it is more likely for errors to occur and damage the device, so to be viable for IST the cost of each one has to be low. Its versatility, in order to decrease the learning curve the microcontroller SDK should be simple to use but also include more advanced features for those students who want them. The more functionalities the device has, the better the investment by IST and the wider the range of possible projects for students to create. Finally, its popularity, to facilitate learning it is important to have as much documentation as possible available on the Internet for students to access in order to clarify doubts or learn how to use the desired features, therefore, in addition to the official documentation, the larger the community using the microcontroller the greater the availability of tutorials and projects in which it is used.

Given the points mentioned and after exploring the available IoT tutorials and projects on the internet, the microcontrollers that stood out were the Arduino Uno[6] with Wi-Fi board based on an ESP8266[7], the ESP8266 itself,[8] and the ESP32[9].

In terms of costs, the official Arduino Uno has an RRP of 20€, being possible to find it at discount in some stores, the clone versions can be found for around 6€ from China with shipping to Portugal included. The Wi-Fi shields based on the ESP8266 are on sale for about 4.5€ from China with shipping included. The ESP8266 is currently on sale for about 3€ from China and shipped to Portugal. As for the ESP32, it is on sale for approximately 5 € from China and with shipping included to Portugal.

The Arduino Uno is a very popular device among microcontroller beginners for its simplicity and tolerance to abuse, however, it does not have access to Wi-Fi, so people who already have it and want to gain that functionality sometimes opt to add a Wi-Fi shield to it. The ESP8266 and ESP32 are also very popular for their extra functionality, extra power, and the ability to also program them through the Arduino IDE, with many people switching from Arduinos to these devices.

The advantage of the Arduino Uno with the Wi-Fi shield, in this work, is IST already having these devices. However, the ESP32 is the best option because, compared to the Arduino Uno, it is much faster, already integrates Wi-Fi in the controller itself, and it is also cheaper, around half the price when compared to the Arduino and the Wi-Fi shield. Compared to the ESP8266, the ESP32 is more expensive but faster and newer, being the successor the company's focus is now on it. In addition, the ESP32 already comes with Bluetooth, thus bringing a wider range of possibilities to students. For this reason, the ESP32 was the microcontroller chosen for this work, using the others for comparison and compatibility.

3. Architecture

This work intends to create an Internet of Things environment at IST and provide a foundation for students to explore this area in the different disciplines and existing courses. For this to be possible an IoT platform was implemented, in this case ThingsBoard, in a server accessible to everyone through the internet.

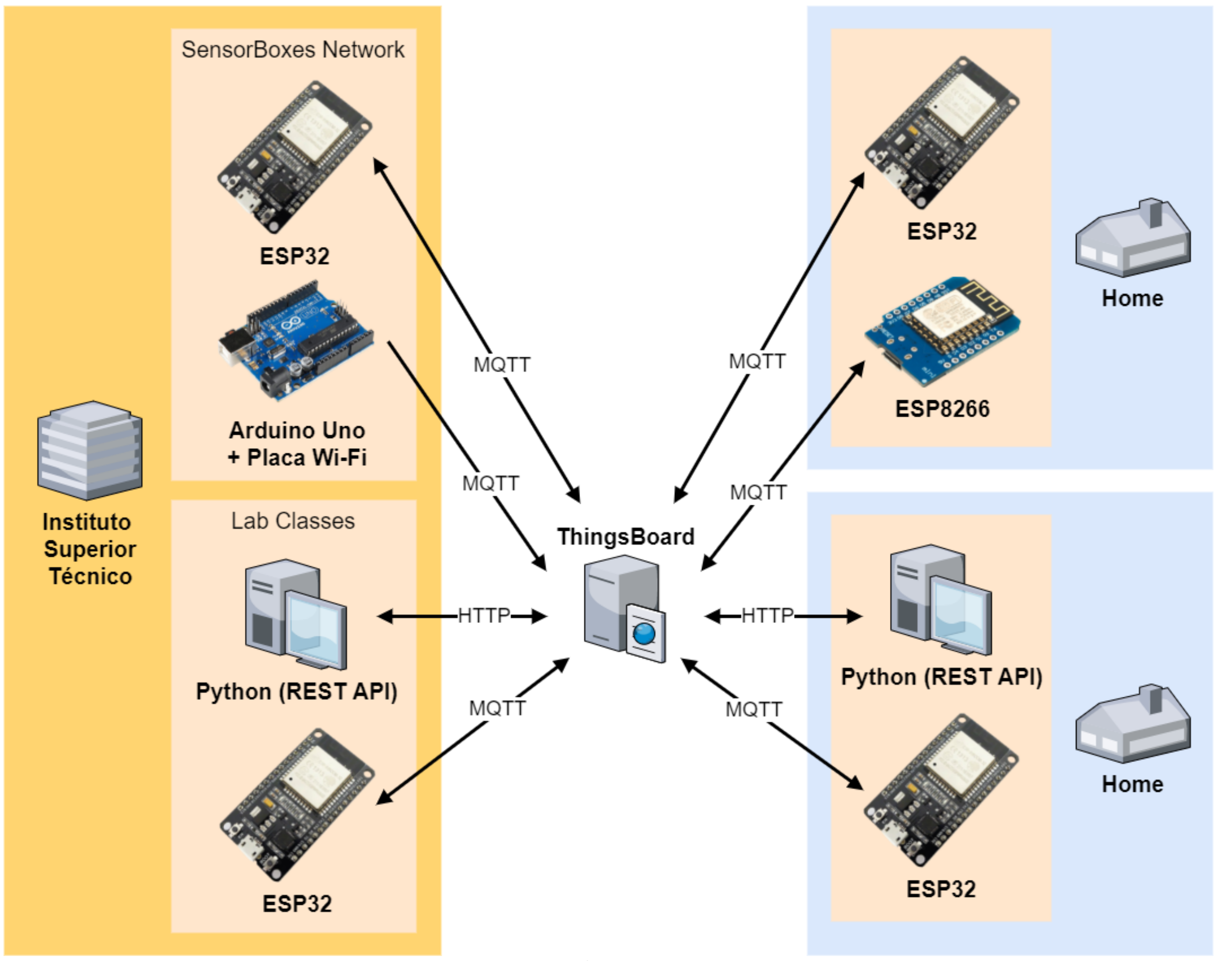

Figure 1: Architecture of the work to be implemented.

The platform serves as a connection point between all microcontrollers and the user, either the latter through the administration page or the REST API, for example, the user can via the REST API receive sensor data from certain microcontrollers and according to his logic trigger the actuators of other ones. On the administration page, the user can manage his devices, analyze the data they transmit in real-time or check their history. This analysis can be done using dashboards where he can create graphs and tables with data, and also buttons to control actuators. The user can define, through rule chains and by device types, what happens to the messages arriving at the server, if they are discarded, if they should be treated before saving, if they activate alarms, among many other options. Devices can also be associated with assets, for example, relating devices to the buildings or objects where they are installed. There are three levels of privileges in the platform, the system administrator who manages tenants and system settings, the tenant administrators who manage their tenant's devices, assets, who their clients are and what devices they have access to, and finally, the client level who has access to the allowed devices and their dashboards. A student who only wants to access existing sensors on campus will have to request a client account from the tenant administrator who manages those devices. It is also possible to make the devices public, however, to avoid malicious use or use that goes against the expected one, it is preferable that when a student or group wants to use them they have to ask for permission. If the student wants to create his own devices, he can do so by requesting a tenant account, so he acts as the owner of his business and provider of device access to his clients. If the student wants to install his device on the campus, available to all students, he will have to ask an administrator of the account that manages IST devices to add his there, or else one of the provisioning techniques will have to be activated so that the student alone can add his device to the sensor network at IST. The sensor data sent to ThingsBoard is called telemetry. Attributes are key-value pairs about device information, it can be whatever the user wants, like firmware version, parameters, or settings. The device can both publish and read its attributes. Users can also access the functionalities of the administration pages through the REST API using HTTP requests, these can be abstracted using SDKs in various programming languages already provided by ThingsBoard.

Considering that the system is to be used mostly by students that have just been introduced to microcontrollers its use must be as simple as possible, so a library must be created for the microcontrollers to simplify their connection and communication with the platform. ThingsBoard has an official library for Arduino that communicates with its IoT platform through the MQTT API. To further simplify the connection and communication with the server for inexperienced students, the library is created as an extension of the previous one, simplifying the use of the following functionalities:

- Wi-Fi network connection, including Eduroam network (WPA2 Enterprise)
- Server connection
- Connections maintenance
- Sending SensorBox details
- Subscribing to commands for the actuators
- Device provisioning
- OTA updates

The library is compatible with the Arduino Uno and its Wi-Fi shield, the ESP8266, and the ESP32. The protocol used to communicate with the platform is MQTT, however, the user will not need to know the message structure used by the various features since he can call functions that already handle that kind of work. With this library, the same code file can be used on all three microcontrollers mentioned.

To demonstrate the use of the platform to students and so that those not interested in building circuits with sensors and actuators can still take advantage of it, devices will be created to be installed around the IST, so it will be possible to obtain measurements of the campus environment through them. These devices are called SensorBoxes. The SensorBox is the name given to the set of sensors, actuators, and one microcontroller ready to be placed anywhere as long as it has access to the internet and electricity. In this assembly, there are also LEDs to show the status of the code and even a way to connect a power supply. The SensorBoxes will be for indoor use and come in two versions, a larger demo version where there is a LED for each sensor and actuator so you can see when they send or receive messages, and a smaller version just to be placed on a wall or table to collect information. The ESP32, ESP8266 and Arduino Uno with the Wi-Fi shield are compatible with the SensorBox construction. The choice of actuators should be thought about according to the target users of the device, whether it is to be used by one person or by students in general. Actuators influence the environment where they are, for example, a buzzer when triggered will produce noise and this may be unwanted in certain locations.

4. Implementation

This section explains the implementation of the work developed, the difficulties and limitations encountered.

4.1. ThingsBoard

In this work, the IoT platform was installed on a personal computer for development purposes and then on three different servers to experience a production environment and work around the problems encountered. It is also explained how to use the administration page and the REST API.

Thanks to the good documentation of ThingsBoard it is easy to run its server and it can be installed in many ways on several platforms. For the development of most of this work was used Docker running the ThingsBoard server on a personal computer, using the PostgresSQL database and the ”In Memory” message queue service. Using these chosen configurations for the server is the simplest way to get ThingsBoard running and ready to develop the rest of the work.

To bring this work closer to real use, we tried installing ThingsBoard directly on servers. All of them ran Ubuntu Server 20.04 LTS, so the installation process was mostly the same for all. Started by installing it in a local network server, with 4 cores and 8 GB of RAM, but without being able to access it from the outside. Using PostgresSQL and the Kafka message queue service, there were no difficulties in using the platform.

In order to make a ThingsBoard installation accessible through the internet, we tried it on an AWS EC2 t2.micro server, which has 1 vCPU and 1GB of RAM. These resources proved insufficient, resulting in only a few minutes of operation between each server restart.

Finally, to be able to replicate the installation done on AWS t2.micro but in a server with more resources and without spending money, a virtual machine with 2 vCPU and 2GB of RAM was requested to RNL, a support network to the computer science courses of IST. Once the machine was available, ThingsBoard was installed with PostgresSQL and the ”In Memory” message queue service, this way it was possible to run the server without problems. The server can be accessed through the address `thingsboard.rnl.tecnico.ulisboa.pt`.

To manage the system there is the system administrator, who can create new tenants and administrators for each tenant. Below this we have the tenant administrators, we can have in this case a tenant called Instituto Superior Técnico and as administrators the people responsible for managing the installed devices, for example, the IST SensorBoxes network. They will be able, among many other things, to create the devices and manage who has access to them. Students who want access to IST's SensorBoxes will have to request a client account from the IST tenant. This way it is possible to control who has access to the devices' data. If a student wants to create his own devices and integrate them with the platform he will have to request an account as an administrator of his own tenant, this way he will act as a company that has its devices and can provide access to their data to its clients.

4.2. SensorBox-ThingsBoard Library

This library is designed to make communications with ThingsBoard as simple as possible. Both this and the official ThingsBoard library, on which this one is based, use external libraries. To ensure that the library works properly and as described in this work, these must be also installed, with the Arduino Uno requiring extra libraries compared to the other microcontrollers.

The Arduino Uno with the Wi-Fi shield has its functionality limited due to its reduced processing power and little available memory, so it is not able to use the features that require subscribing to messages, it can only send information to the server.

The library was created in a development environment with a Wi-Fi network that has the WPA2-PSK authentication mode, the traditional home Wi-Fi network with an SSID and a pre-shared password. At IST the Wi-Fi network used is Eduroam, whose authentication mode is WPA2 Enterprise, so it requires the library to use that authentication method to connect to the Internet. Although it was not possible to test, the functionality of being able to connect to networks with WPA2 Enterprise was added experimentally to the ESP32.

Thanks to this library, the three microcontrollers mentioned in this work can run the same code to connect to and communicate with the Wi-Fi and ThingsBoard.

The library code can be viewed in its GitHub repository [10]. The explanation of the created code is commented in the code files. Usage examples can be found in the same repository within the ”examples” folder.

4.3. SensorBox

The SensorBox is the name given in this work to the assembly of sensors, actuators, and a microcontroller that communicates with the ThingsBoard.

It was designed, at this stage, for indoor spaces. It aims to have its entire interior visible so that students can better understand what they are working with. It should include LEDs that indicate its operating state, being that the difference between the normal and the demonstration version is that the former has only one LED representing all sensors and actuators while the latter has a LED dedicated to each sensor. Finally, it should also include a plug to connect a power supply. The construction should be universal in order to be compatible with as many sensors and actuators as possible. To be able to install sensors that make measurements related to air, it must be able to circulate inside, so it was decided to create an open sandwich construction, with a wooden base and an acrylic top. It is compatible with the assembly of two normal 400-pin breadboards side by side or with the assembly board from the official Arduino Uno kit. Besides microcontrollers and breadboards, it should also be possible to mount sensors that already have their own circuit on a PCB.

The SensorBoxes created, following the criteria described above, can be seen in the images 2 and 3.

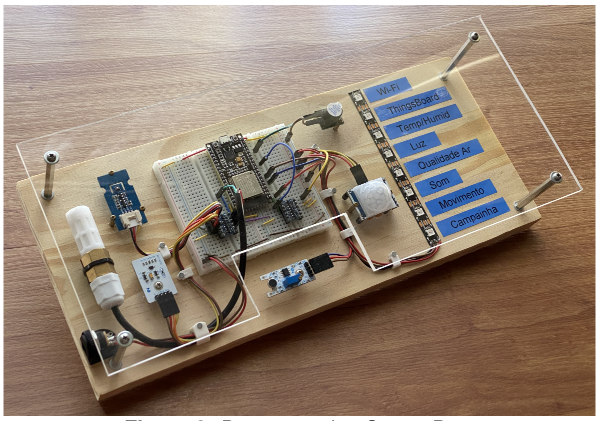

Figure 2: Demonstration SensorBox.

A variety of microcontrollers were used to show their compatibility with the library. The ESP32 is the one recommended for this work and used in the demo SensorBox. The ESP8266 although older and worse than the ESP32, is still a very viable option for this work and is the microcontroller used in the normal SensorBox. The Arduino Uno with the Wi-Fi shield is used because IST has quite a few kits and it is easier to spare those for this use. So the Arduino can be integrated into the system despite its limited functionality due to its low memory and weak processing power.

A list of interesting sensors and actuators was created for the IoT lab. The criteria were to be low-cost components, to prioritize digital sensors, sensors mainly focused on measuring the environ-

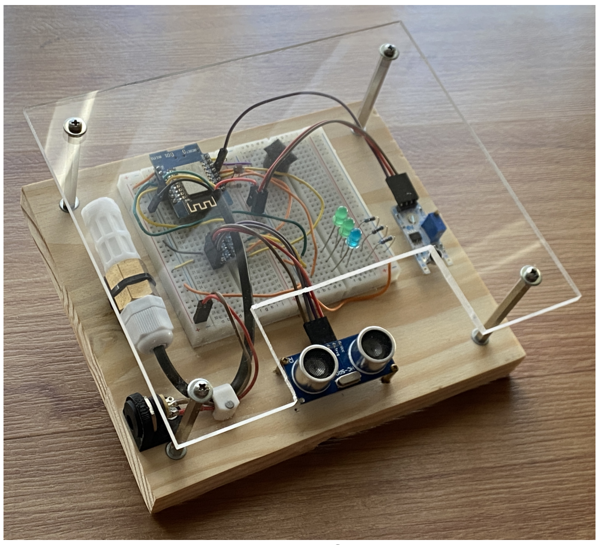
Figure 3: Normal SensorBox.

ment, not to repeat sensors already existing in the official Arduino Uno kit, and finally to be sensors popular in the community.

To demonstrate different ways of using the library created, the SensorBox Normal, with the ESP8266, was programmed without an operating system or similar, and the demo version, with the ESP32, uses FreeRTOS in its program. The Arduino Uno was programmed in the same way as the ESP8266 only adapting the code to its sensors. The code can be found in the library repository as examples of use and contains comments explaining what happens in it.

5. Evaluation

In this section, the capabilities of the created work are tested to see if it fulfills what is intended and if it is indeed beneficial to IST students.

5.1. Performance

The performance of the system depends on the conditions in which it is used, for example, if the server and the devices are on the same network or communicate over the internet, or the available resources of the server on which ThingsBoard is installed. The tests performed used for local communication the production environment on the local server and for communication over the internet the final production environment, which was installed on the RNL server.

5.1.1 Telemetry Latency

To measure the latency of the messages we used a Python program that sends telemetry representing a device and at the same time is subscribed via WebSocket to its own device's data. Right after the message is sent a timer is started that ends when the message is received back. This way we are able to correctly measure the time the message spends from the time it leaves one device until it is received by another, passing through the server. In this test, we are not only measuring the time it takes for the message to reach the server but also, to arrive at the server, be processed, sent to those who are subscribed, and received by them. The test code can be found in the library repository for SensorBoxes, in the ”python_test” folder with the name ”latency_test.py”.

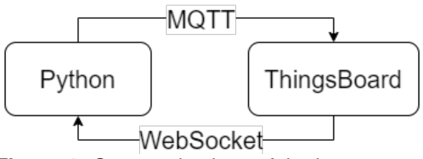

Figure 4: Communications of the latency test.

After getting the results of sending 500 messages to each of the servers, as expected, we see that the version where the server is on the same local network as the device has lower latency because the messages travel a much shorter path, averaging 9ms versus 206ms for the server on the Internet. Despite having different message queuing services it should have practically no influence because the server was only handling the few messages from this device, no one else was communicating with it. Note that the results of the test with the server communicating over the internet were more unstable over the several times the test was run, however, the values used for comparison are within the range of the most typical results. The result of the test with the local server was virtually always the same. In both situations, we got values well below the 1 second that was intended. The tests were performed in ideal load situations for the server, only one device communicated with it, however, this demonstrates that a server correctly sized for its use case can achieve good results.

5.1.2 Telemetry Send Time

Besides the latency of sending messages, it is important to see if from the moment you decide to send a message until it is sent there is any significant time lag. For this purpose it was created code that sends a message and measures the time it takes for the sending function to execute, it can be seen in the library repository in the ”examples” folder with the name ”message_sent_test”. It does this 100 times and at the end returns statistics about the measurements. A comparison is made between the values of the three microcontrollers covered in this work, the ESP32 with the FreeRTOS based API, the ESP8266 with the NonOS API, and the Arduino Uno with Wi-Fi card. Thanks to the library created, the same code file is used on all of them.

Analyzing the results we see that the time taken by the ESP32 and the ESP8266 is very small, at most 2ms for the message to be sent, not being an issue. However, the Arduino Uno takes much longer to send the message, 114ms on average, this is due to its lower processing power and mainly to the fact that it has to communicate with the Wi-Fi shield to send the desired message, all this causes a considerable delay when we want the user to have a real-time experience. So it is not recommended to use the Arduino Uno when a minimal delay in measurements is important to the user.

5.1.3 Server Load

The purpose of this test is to compare the performance of the two server configurations mentioned in the latency test. The same code was used as in the latency test but now with more than 1 device sending a message per second. Various combinations of number of devices and number of messages per second were used. In this test, each device has a user subscribed to its messages. The duration of each test is 60 seconds. Three combinations of parameters were chosen for the test, representing different levels of expected usage of the server, from little to moderated usage. With Test 1, 10 devices send 1 message every 2 seconds second, this is a rate of 5 messages per second, 300 messages per minute. Test 2 has 50 devices sending 1 message per second, a rate of 50 messages per second, 3,000 messages per minute. Finally, Test 3 has 250 devices sending 1 message per second, a rate of 250 messages per second, 15,000 messages per minute.

As with the latency test, the values from the Internet server tests were more unstable, however, the selected values represent the majority of the results obtained. As expected the local server performed much better, staying almost always below the 1 second latency, the desired maximum. The web server due to the longer path the messages take and the server configuration, in test 2 already performed worse than the local server in test 3. In the third test, the latency is mostly always above one second and messages are already lost. So the server hosted in the RNL is only suitable for development and very light loads. For normal use, the server should be ordered with more RAM resources and consequently the possibility of installing the Kafka message queue on it. In both it would also be beneficial to use the hybrid database, where Cassandra would be used for receiving messages.

There are official performance tests of ThingsBoard on AWS servers. Those tests are not directly comparable to the ones performed here as the official ones only test the publishing of messages, not the subscription to them by users. However, both types of tests complement each other and give a general idea of the servers' performance in sending and subscribing messages and their delays.

5.2. SensorBox-ThingsBoard Library Size

It was measured the impact of the official library and the one created in this work on the microcontrollers' memory. For the ESP32 and ESP8266, the increase created by the SensorBox library compared to the official library is not enough to consider not using it since it will greatly simplify the user's code. Once the user begins to make his own code to access the features that this library provides, he would already be increasing the space occupied by his code, minimizing the difference compared to the use of the SensorBox library. For the Arduino Uno memory is already a scarcer resource, however, the SensorBox library already excludes non-compatible functionalities and thus most of the code it adds will already have to be manually created by the user if the library is not used, for example, the connection to the Wi-Fi network and the ThingsBoard server. So including the library created in this work is still an advantage even on the Arduino Uno.

5.3. Laboratory Work

Three laboratory guides were created based on those of the ACIC course to be tested with students and thus to prove the usability of the platform in a learning environment. Due to the pandemic situation in which this work took place and the fact that the lab work needs physical presence to provide the necessary material for it to be done, it was not possible to carry out the tests. However, the guides on which these are based have already been tested several years with students and are therefore adequate. Being able to keep the same idea as the ACIC guides, only adapting them to use the technologies related to this work without increasing the difficulty, shows that the system created here is suitable for use by students in classes and consequently in projects. The guides can be found in the repository of the SensorBoxes library [10] inside the ”guides” folder.

The difference of these labs compared to the original ones is the use of the ESP32 instead of the Arduino Uno and the communication over the Internet instead of I2C.

With these new guides, besides being possible for students to work in groups at a distance, they also gain experience with IoT platforms, important in the professional world where these are increasingly used to connect the various objects in our lives. In addition, they would also get to know the platform to use in other works if they want, such as to access the SensorBoxes or add new devices.

Compared to I2C, the limitation that must be considered is the communications delay as seen in the previous points of this section. However, in the real world and for the use indicated in the guide, the latency being added to communications when using the ESP32 is not a problem.

6. Conclusions

With this work began the development of an IoT laboratory for IST, having chosen to start with the implementation of an infrastructure to interconnect devices through the internet to which students can have access anywhere in the world. Considering only open-source IoT platforms as an option, we decided to use ThingsBoard due to its vastly superior popularity over competing platforms, very active development, and an ever-increasing range of features. Three popular devices in this area for those just starting out were compared, the ESP32, the ESP8266, and the Arduino Uno with a Wi-Fi shield. It was concluded that the ESP32 is the best option, being the one with the best performance, better Wi-Fi connection, more pins to use, and costs about the same as the other options, however, if the Arduino Uno is the official version it becomes much more expensive. The Arduino Uno is not ideal for use with the ThingsBoard due to its limited resources, plus the Wi-Fi shield provided added complexity to its use because of its problems with the logic level converter. Assemblies were created to serve as a basis for the installation of sensors spread around IST and to demonstrate their operation to students. A library for the mentioned devices was also created, on top of the official library, to further simplify their communication with the server to the point where the same code could be used in all of them, when the functionalities used are available in all of them. Thanks to the library it was possible to create an easy to follow lab guide to be used in class, without the students having any experience in communicating over the internet with these devices.

Through the performance tests conducted, it was observed that a properly sized ThingsBoard server is capable of serving the objective of the work, to capacitate IST with an IoT platform that can be used by students inside and outside of it, to explore and perform various types of work in the area, from the most relaxed to the most demanding in terms of response times, within what is expected in the IoT area.

Thanks to the infrastructure created, students from various courses can experience the Internet of Things in their areas of interest, thus gaining a good knowledge base for the professional world of today and tomorrow where more and more objects are connected to IoT platforms sending data about their surroundings or what they do, to be later analyzed for various purposes.

6.1. Future Work

Future developments of this work would include the installation of several SensorBoxes at IST, for example, with sensors measuring the state and quality of the environment, so that interested students can do projects about detecting the occupation of interior spaces. Other possibilities would be the creation of tutorials about the use of other existing functionalities in the ThingsBoard, the creation of a webpage for the IoT lab so that students can learn about what is available to them both in terms of software and hardware, and can consult information on how to use them. As for the SensorBoxes, the future goal would be to develop outdoor assemblies where the components would be protected from the weather, and solar-powered versions could even be created for flexibility of installation on campus without having to run electrical wires to them. Finally, in the lab environment, specialized workbenches could be created for IoT and microcontrollers, where there would be boards with various combinations of pre-assembled sensors and actuators together with the necessary breadboards and power supplies so that students could more easily and with greater organization experiment using various components without having to deal with mounting them individually and possibly damaging them, they would only have to connect the desired sensor wires into the assembly they were making.

References

[1] ISCTE IoT Lab. `https://istar.iscte-iul.pt/portfolio-posts/vr-lab/`. Last access on 21-Out-2021.

[2] Swami Keshvanand Institute of Technology IoT Lab. `https://www.skit.ac.in/research/iot-lab.html`. Last access on 21-Out-2021.

[3] Leonidas Andrade Otalora. Implementación de una plataforma colaborativa del internet de las coisas para la captura de variables ambientales para el municipio santiago de cali. pages 55–58, Mai 2019.

[4] Willian Pereira dos Anjos Eduardo Natan Bitencourt. Iot centralization and management applying thingsboard platform. pages 25–27, Aug 2018.

[5] Open-Source IoT Platform - ThingsBoard. `https://thingsboard.io`. Last access on 15-Set-2021.

[6] Arduino Uno official page. https://store.arduino.cc/arduino-uno-rev3. Last access on 9-Set-2021.

[7] Página do shield Wi-Fi. https://www.instructables.com/ESP8266-ESP-12E-UART-Wireless-WIFI-Shield-TTL-Conv/. Last access on 9-Set-2021.

[8] Microcontroller with Wi-Fi - ESP8266. https://www.espressif.com/en/products/socs/esp8266. Last access on 15-Set-2021.

[9] Microcontroller with Wi-Fi and Bluetooth - ESP32. https://www.espressif.com/en/products/socs/esp32. Last access on 15-Set-2021.

[10] Arduino library, SensorBox-ThingsBoard-SDK. https://github.com/JoaoSaramago/SensorBox-ThingsBoard-SDK. Last access on 28-Out-2021.