GymApp with IoT sensors and controller hub, cloud repository and control from web app

Ferran Palma Comas

February 2023

Abstract– Nowadays there are many systems that allow us to register several parameters of a training session: calories burned, time spent, etc. However, there is no solution that allows us to know the state of the facilities in which these training session is done in terms of temperature, humidity, capacity... This is therefore a need that must be met: giving the user a way of knowing the conditions of his training center. To this end, it is proposed to create a complete IoT system that is supported by cloud functions for data processing and storage, as well as their display to the user. To do this, there are sensors that capture information from the environment, a cloud that analyzes, processes and stores it, and a platform where the user can access the data in an attractive and intuitive way.
Keywords– Z-Wave, sensors, Home Assistant, IoT hub, Cloud, Cloud Repository, Azure, Firebase, Grafana

1 Introduction

Technology has advanced by leaps and bounds in all aspects of society. It has made a strong entry into the workplace, especially since the pandemic, a time that is hard to imagine without telework and the possibilities it has brought. Around 99 % of companies with more than 10 employees have computers and internet connections by 2020[17].

This development has also reached homes. Nowadays, it is hard to imagine a house without a WiFi network and, at least, one computer. In fact, in Spain, more than 95% of homes have broadband and more than 83% have at least some kind of computer[16].

In fact, technology has drastically changed the way we humans understand the world. Every day, more than 200 million users check twitter at least once a day[19]. Moreover, in 2020, 21.5% of all transactions were made using the phone wallet[4].

However, this incredible advance does not seem to have made much headway in the world of gyms, despite the fact that this world is becoming increasingly popular among Spaniards (it is estimated that in 2019 more than 5 million Spaniards were enrolled in a gym [9], an increase of 26% compared to the previous year).

• Contact: 1528193@uab.cat
• Specialization in: Computer Engineering
• Work tutored by: Daniel Franco Puntes (Departament d'Arquitectura de Computadors i Sistemes Operatius)
• 2022/23

While it is true that there are many applications that serve to provide information to the user about their training (calories burned in a session, the average heart rate of the session...), users have no way of knowing how are the facilities in which they practice their activity in the sense of occupation, availability of the machines, etc. This information is becoming increasingly relevant given the high occupancy figures in gyms. While it used to be unthinkable to go to the gym and not find any empty machine, today it is more than usual to arrive at the gym on a Monday afternoon and find that it is absolutely impossible to train given the large number of people there.

It is in this context that this project makes sense. The market niche to be targeted is very specific and there is currently no real solution to the problem posed in this work: the aim is to be able to give to the user a way of consulting real-time and historical data on his or her training centre through a complete IoT system.

2 Objectives

Having deduced the main objective: to give the user a way of consulting real-time and historical data on his or her training centre through a complete IoT system, it is now time to break this down into more specific objectives that allow a deeper understanding of the system's features.

1. Create a network of Z-Wave sensors that are capable of capturing useful information from the environment.

2. Capture all this information using open software options (Home Assistant [13]) installed on a microcon-

troller (Raspberry Pi 3 [10]).

3. Send the data remotely to some cloud service, such as Azure.
4. Analyse and process the data in order to generate valuable information.
5. Display the cloud data through a low-code user-friendly platform.

3 State of the art

This project uses several technologies that need to be studied. However, a brief study of these technologies will be made in order to justify their use, as an exhaustive study is beyond the scope of this project.

The first of the technologies used is Z-Wave. Z-Wave is a low-power communication protocol that operates, in Europe, in the 868.42 MHz band. This allows the communications of this protocol to have no interference with technologies such as WiFi or Bluetooth, which operate in the 2.4MHz band.

There are other similar protocols such as Zigbee. However, Z-Wave has one great strength: it is a closed standard. This means that no device using this protocol will have compatibility problems with other devices of the same type. This is not guaranteed when using Zigbee, because it is an open standard.

Z-Wave is designed to provide secure, low-latency transmission of small data packets at data rates of up to 100Kbps, and its networks can currently consist of up to 4.000 devices.

As far as the microcontroller is concerned, the RaspberryPi 3 Model B+ is a board with 1GB of RAM, an Ethernet port capable of working at speeds of up to 300 Mbps, SD card support... which are more than enough specifications for the system in question. There are other options a little more powerful even within the same brand, however, for the specifications of this project this device will be more than enough and will allow easy scalability if necessary.

Home automation software has come a long way in recent years. HomeAssistant is one such software. It first appeared in 2013. It is open source, multi-platform and since 2017 it allows the integration of many technologies. As of today, HomeAssistant is integrated with more than 2.300 technologies (including, for example, Z-Wave).

Another possible option would be to use OpenHab, which is also an open source software that serves the same purpose as Home Assistant. However, OpenHab requires a more complicated installation as well as a larger amount of resources to run.

Since the function of such software in this project is simply to bridge between the sensor network and the cloud, Home Assistant seems to be a better option.

Just as home automation software has come a long way in recent years, cloud-based technologies have not been left behind.

There are now multiple of cloud options, including Azure. This platform offers several of cloud services, including real-time data ingestion systems that are capable of receiving millions of events per second, others that are capable of analyzing this data virtually instantaneously, and others that can store data on the scale of terabytes. Moreover, all these applications tend to be highly scalable and flexible, adapting to the needs of each use case.

Finally, as far as user interface is concerned, there are also multiple options: programming a web page to visualise the data, use tools like Microsoft PowerApps to develop a mobile application... There are many possible ways to tackle this problem.

However, in our case the best option is Grafana. Grafana is a service that allows you to create customised panels on which to display data through a web application that can be consulted both from a computer and from a smartphone. In addition to requiring very little code to operate and having several integrations that facilitate the collection of data.

4 Methodology and Planning

The methodology followed in this project is the Kanban [18] methodology. This methodology is based on the creation of cards corresponding to tasks. For each card, a completion status can be assigned (in addition to other items such as a deadline or a short description of the task): pending, in process or completed. This allows, in a very graphic way, to be able to understand the status of the tasks that make up the project. In order to implement this methodology, Notion boards have been used.

The tasks that make up the project have been grouped into groups and the precedence and origin of each of these has been defined, so that a cascade planning of the project has been defined. According to the planning, for a phase to start, the previous one must be finished. This planning will allow to obtain, at the end of the last phase, a MVP (Minimum Viable Product). Furthermore, the result of this planning is a Gantt chart that clearly shows the time division of each of these groups of tasks (this chart can be consulted in the Github respository[5]). In addition, for each of the groups, it can be seen which tasks make up the group and what is the precedence relationship between each one of them.

5 System architecture

Figure 1 shows a schematic diagram describing the architecture of the system:

Basically the system works as follows: a sensors network captures data from the environment (temperature, brightness...) and sends it to the microcontroller. Once the information is in the microcontroller, it is sent to the cloud. It is in the cloud where this data is processed and analysed and it is this service that is responsible of communicating with the client to provide the information requested.

5.1 Sensors Network

The sensor network that we will use will communicate using Z-Wave, which, as already mentioned, is a low-frequency, secure, and low-consumption communication protocol. In addition, the market has many sensors that use this protocol, which allows us to create a system with a wide variety of devices that can measure an infinite number of different metrics.

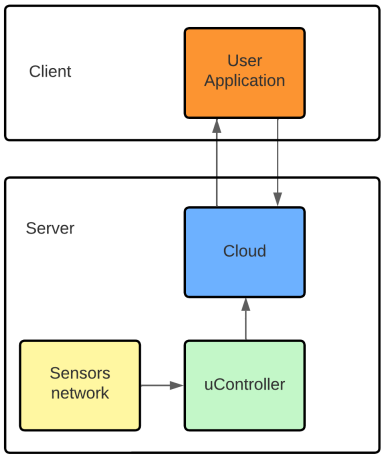

Fig. 1: System architecture

Regardless of the communication protocol we use, we must have a hub or central node, to which all the sensors communicate and which communicates with another device (usually a microcontroller).

5.2 Microcontroller

At has been said, in order to be able to deal with the data of the sensors, a microcontroller is usually needed. And the microcontoller (in our case, as mentioned above, a Raspberry Pi 3), needs some kind of software running on it to be able to handle the data. The kinds of software that can be used are varied and each one offers it's own characteristics.

One of them is to use Home Assistant. Home Assistant is a free software project aimed at home automation that puts local control and privacy first.

This software is perfectly adapted to the needs of our application. In addition, it has a number of features that make it a very good choice:

- It is secure.
- It does not depend on any external company but on it's community, as it is a free software option.
- As a consequence of this last feature, it offers a high level of data privacy.
- It has a large number of native functions as well as a large number of Ad-Ons created by the community.
- It is certainly a simple tool to install and intuitive to use.
- It has a large community behind it.
- It's prepared to be supported by a Raspberry.

5.3 Cloud

As mentioned above, Home Assistant provides a simple way to collect information from the sensors that are part of the IoT network.

For this information to acquire value, an important step is to be able to store and process it in the cloud.

There are many options on the market that allow this functionality to be implemented. One of them is Azure, a Microsoft service that offers multiple products and services in the cloud, some of which are perfectly suited to the needs of this project.

In addition, HomeAssistant offers an integration that allows it to communicate with Azure in order to send the information it holds.

It is for all these reasons that Azure has been chosen as the cloud service for this project.

5.4 Information Display

Being able to display information to the user in order for the application to be of real use is as important, if not more important, as being able to process this information properly.

Therefore, it is of vital importance to choose a platform that allows the user to consult the information of interest in an easy, simple and intuitive way.

Furthermore, this platform must be simple to develop, as this allows working with low-code tools (one of the objectives of the project) and also to focus more on the flow and processing of the data than on its presentation.

Azure offers a wide variety of options for displaying data, both using the platform itself or using third-party tools.

One of these options is Grafana, which is a free software tool that allows data to be displayed graphically. It can be installed both locally and on a server and allows anyone with a connection to the server to visualise the data available to them in a very graphical, intuitive and attractive way.

At the administration level, Grafana also offers a plugin that allows it to connect to Azure, it is very easy to use and hardly requires any code, so it adapts perfectly to the use case in which we are.

6 System development

This section details technical aspects of the infrastructure that has been built: from the installation of HomeAssistant and the set up of the sensors network, through its connection to Azure and the processing of data on this platform to the installation and configuration of Grafana.

6.1 Z-Wave Network

Figure 2 will help us to understand in detail how the Z-Wave sensor network works:

Fig. 2: Z-Wave Network scheme

The network consists of one sensor and a hub: the Multisensor 6 and the Z-Stick Gen5, all from Aeotec.

The sensor is paired with the Z-Stick, which acts as a central node and communicates with the microcontroller via USB, transmitting the data it receives from the sensor to the microcontroller.

The power supply for the sensor is flexible. It can be powered directly from the power supply via USB (using, for example, the voltage provided by the board) or using external batteries.

To pair the sensor with the hub, we have to follow the user guide of the sensor [1] as well as the Z-Stick guide to get the Z-Stick into pairing mode [2].

Once this is done, the Z-Wave network is set up and operational, so the next step is to process the data from the controller.

6.2 HomeAssistant

Installing Home Assistant requires a microSD memory card, a board on which to install it (Raspberry Pi3) and a software that can write image files (Balena Etcher [3]) to a memory device.

However, before proceeding with the installation of Home Assistant [15], it is extremely important to tell the Raspberry that it must boot to the image contained on the memory card.

Once done, we have to download the Home Assistant image via the GitHub repository [14] and write it to the memory card using Balena Etcher. The next step is to insert the memory card into the slot on the board and connect it to the power supply and to the internet via an Ethernet cable.

After a few minutes, Home Assistant will be installed and can be accessed via the url *http://homeassistant.local:8123* as long as we are connected to the same network as the microcontroller (which acts as a local server).

Once Home Assistant is fully installed, a page will appear where some basic system parameters can be configured and then the software is prepared to be used.

The first thing we need to do is to be able to detect the sensors in Home Assistant. To do this, Home Assistant has an integration that allows set up a Z-Wave network. Simply install it and it will automatically detect the Z-Stick and, consequently, all the devices that form part of its network.

By default, the Multisensor 6 [1] will send information every hour. Since the intention of this application is to be able to display information in real time, it will be good to change this parameter so that the sending of information has a lower periodicity.

To do this we have to go to the devices section of the Home Assistant menu and configure the sensor data sending time to the one desired. In our case, 10 minutes.

At this point, we now have a fully operational Z-Wave network and we can see the information through the dashboard that Home Assistant automatically generates.

Now it will be necessary to connect our Home Assistant server to the Azure cloud. In order to understand in detail how to configure this functionality, it is also important to understand what an Event Hub is and how it works (this information can be found later in the document). Basically, what we need to do to connect HomeAssistant to the EventHub is provide a key called ConnectionString; this key can be obtained from the EventHub permissions. So, to establish the connection, we must install the Azure integration in HomeAssistant (as we have done with the Z-Wave one) and provide the ConnectionString during installation. If this is correct, HomeAssistant and Azure will be connected and the data will start being sent to the cloud.

6.3 Azure

To understand the cloud application that has been implemented, it is important to understand which components make it up and how they are interconnected with each other.

As a main idea, it is necessary to understand that Azure offers multiple applications and these are executed in clusters, so, each time an application is created (i.e: a database) it will be necessary to create a cluster, which, by definition, may contain several instances of the same type of application.

In other words, we could have a database cluster for one application and within this cluster we could have several independent databases.

In addition, it is also possible to have different clusters that can be grouped into resource groups.

The figure 3 shows an overview of the components of our cloud application (if you want to consult this or any other image with bigger size, they are all available in the GitHub repository [5] of the references):

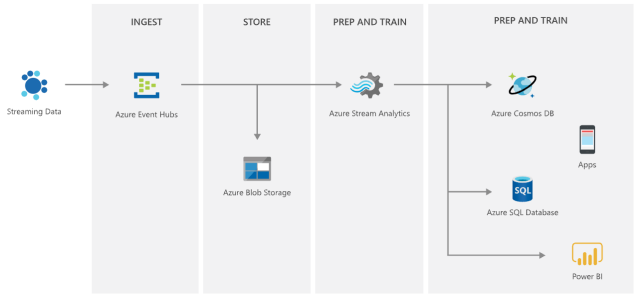

Fig. 3: Cloud application scheme

The data coming from the sensor network is dumped on an EventHub (let's understand the EventHub as a raw data container for the moment). These are processed and finally sent to a database. In the picture, there is an additional component called BlobStorage. This is basically a container the same as EventHub except that BlobStorage is meant to store raw data, while EventHub is meant to hold it for a retention period and then delete it. For the system that we are implementing, the BlobStorage is not necessary and therefore our scheme is the one shown in the image without taking this component into account.

6.3.1 Resource groups

As described in the official Azure documentation [8]: a resource group is a container that holds resources related to a certain solution.

So, all the applications that we use to make the data captured by the sensors capable of being displayed in Grafana will be grouped into a resource group.

Creating a resource group is a simple task and the process can be found also in the official Azure documentation cited above.

From now on, all the applications we use in the solution we are implementing will be grouped into the created resource group.

6.3.2 Event Hub

The first of the applications needed to implement the desired solution is an Event Hub, as seen in Fig. 3.

An Event Hub is a real-time data ingestion system that is capable of receiving millions of data per second.

The Event Hub is basically used as a container in which to dump all the data provided by the sensors. This data is not treated or processed, as this must be done in other phases of the application.

To create an Event Hub, apart from having a resource group in which to store it, it is necessary to create a namespace of Event Hubs. This can contain one or several Event Hubs. The namespace refers to the clusters mentioned above and their creation is necessary for each of the applications to be created, so from now on, it is understood that, to create each of the applications, it is necessary to first create a namespace that contains it.

The Event Hub creation interface asks us to enter several fields in order to create it: the subscription that will be used (this is necessary for anything that is created and will be omitted from here on), the resource group in which it will be located, the name of the namespace in which it will be and the server on which it will be hosted (this step refers to the geographic location of the Azure server where our application will be hosted. Again, is repeated in the same way as the subscription and the creation of the namespace and, therefore, will also be understood in subsequent explanations).

When we create the Event Hub, we get some information such as the resource group it is part of, the geographic location where it is hosted, the creation date...

Once the Event Hub is connected to Home Assistant, it will offer us metrics about data ingestion. Figures 4, 5 and 6 show some of the metrics provided:

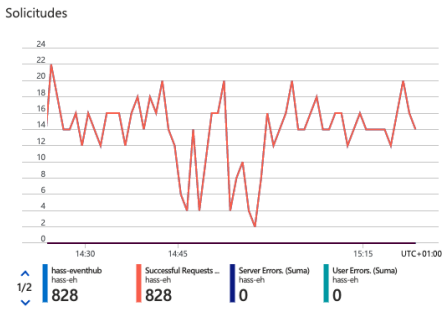

Fig. 4: Requests for data ingestion.

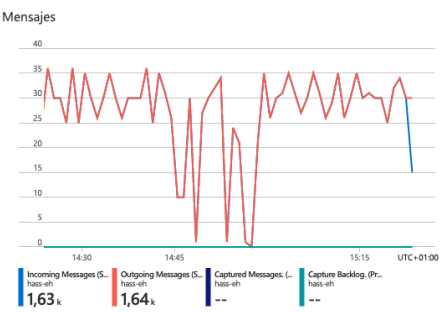

Fig. 5: Incoming and outgoing messages of the Event Hub.

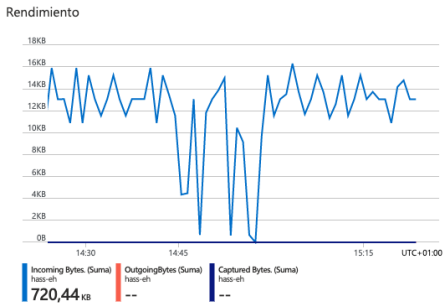

Fig. 6: Performance measured as input bytes.

However, at the moment the data is not being processed and, in fact, cannot be consulted as such.

6.3.3 Stream Analytics

In order to be able to process the raw data received and filter it to store in a database only those data that are of interest to us, Azure offers several tools. One of them is Azure Stream Analytics.

This tool offers a way to process data through SQL queries, so that, from a data source (in this case, the Event Hub), data is filtered through queries and spat out to an output. In this case, a database service offered by Azure: Azure Data Explorer.

Azure Stream Analytics works through jobs. Several jobs can be run at once and each job receives data from one or more inputs, performs one or more queries (functions can also be used, although this functionality has not been explored), and pushes the results to one or more outputs.

One of the jobs created runs the SQL query shown in figure 7:

Fig. 7: Query for the job

In which, basically, the data is filtered to only get the information provided by the sensor about the temperature and omit all the other information. The following images show the input data (figure 8) and the data obtained once the query has been executed (figure 9):

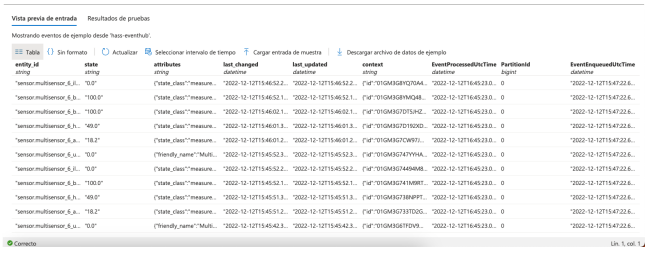

Fig. 8: Data before the query.

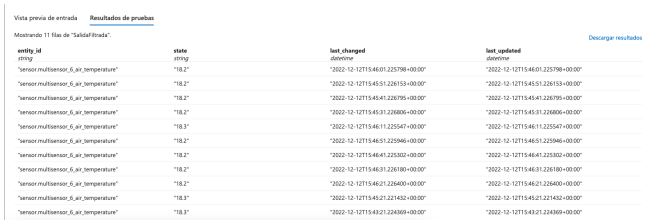

Fig. 9: Data after the query.

The filtered data is sent to the specified database and saved in it's corresponding table. In this particular application, there are two jobs: the one corresponding to the images and filtering the data concerning temperature and a second with a very similar query but filtering the data concerning to the humidity.

Each of these jobs sends the data to a different table in the database (the details of the tables and the creation of the database are specified in the corresponding section).

6.3.4 Azure Data Explorer

In order to be able to persist valuable data that has been analysed in the cloud, a database is necessary.

Azure offers several applications that function as databases. One of them is Azure Data Explorer. This tool is very useful because the aplication that we will use to display the data to the user offers a plugin that allows to connect to such a database.

When we create the database we simply need to offer Azure two values: how many days of retention the data has and how many days this data is cached. By default, these metrics have values of 365 and 31 days respectively. For a first prototype of the system, these values are more than enough, but it is good to know that this option is offered and that, if necessary, the parameters can be modified.

Once the database is created, it is important to create the tables that will contain the information. For each table we must give it a name as well as define which fields it will contain. For each field it is necessary to specify the name and the type of data it will contain.

In our particular case, both tables had 4 fields: *entity id* (String), *state* (String), *last changed* (Datetime) and *last updated* (Datetime).

6.3.5 Aplication

For the user to be able to consult the data provided by our IoT sensor network, it is necessary that the platform used to display them (Grafana) is able to access the database we have in Azure.

This connection cannot be established directly and a tool provided by Azure must be used: the applications. These are based on the use of Azure Active Directory, which is a cloud-based identity and access management service.

Therefore, in order for Grafana to access the cluster in which the Azure Data Explorer database is located, it is necessary to create an application and connect it to Grafana.

To create an application, we must give it a name and, once created, connect it to Grafana (this part is described in detail in the Grafana section of the document).

However, this application by itself can't provide Grafana any information for the panels. It needs to have access to the database. To do this, from the database itself, some permissions have to be modified to give the application read permissions. The details of how this process is done are detailed in the official Azure documentation [7].

6.4 Information Display

Once we have obtained processed information, it is time to give it value through presenting it to the final user.

To do this, we have chosen to display the information in panels so that both users and gym administrators can consult it. These panels are grouped into dashboards, and these dashboards are the ones that the user can consult. This gives the project an extra point: it is useful not only for gym users but also for gym administrators.

6.4.1 Webpage

The information of interest to the user and the gym manager is not the same. So it will be necessary to differentiate between users and administrators in order to be able to provide the interest information for each one.

The user will simply want to know what the status of the room is at any given time, while the manager is likely to be interested in having historical as well as current data to make decisions based on it.

To be able to do this differentiation, when accessing the application, the first thing that will be displayed will be a web page in which, in the case of being an administrator, authentication will be requested to be able to view the data, and in the case of being a user, direct access to the information panel will be possible.

A first prototype of the website has been created, as shown in figure 10:

Fig. 10: Webpage prototype

As we can see, it consists of a form to authenticate and an extra button to go to the panel without any credentials. This form will be the one that the gym administrators will have to fill in in order to have access to their dashboard. In case of being a user, no authentication will be necessary and the data can be consulted by clicking on the corresponding button.

6.4.1.1 Flask Server

In order to host this website, a server is required.

For this prototype, two options have been implemented: a local server and one using a cloud server. To create the server that is hosted locally, it has been done by an application that uses Flask. Flask is a Python framework that allows you to create and host web pages with very little code.

To use Flask, it's necessary to install it in the machine that is going to act as the server using the command *pip3 install flask*. Once it is installed, just a basic script is needed, in which parameters such as the address and port on which the server will open can be defined. In this case, it will run locally on port 80.

However, to display a web page other than the default that comes with Flask, some HTML, CSS and JS have to be written. The files that contain the code have to be referenced in the Flask aplication in order to show to the user the webpage.

The details of this code are hosted in a GitHub repository that can be consulted in the appendix [5] or in the following link: *https://github.com/ferranpalma/GymApp-with-IoT-sensors-and-controller-hub-cloud-repository-and-control-from-web-app*

6.4.1.2 Azure App Service

As mentioned in the previous section, the website has been posted both locally and remotely.

To hang it remotely, we have chosen to use a service offered by Azure called WebApps. This allows you to host web applications and that these can be accessed from an address of the type: *http://webAppName.azurewebsites.net*.

In order to host the application there are various methods, as can be seen on the Azure website. One of them is through a VSCode extension, and just follow the tutorial as shown on its official website[6].

Once the application is up, it can be consulted through the following link: *https://gymdatatfg.azurewebsites.net/*, as can be seen in figure 11:

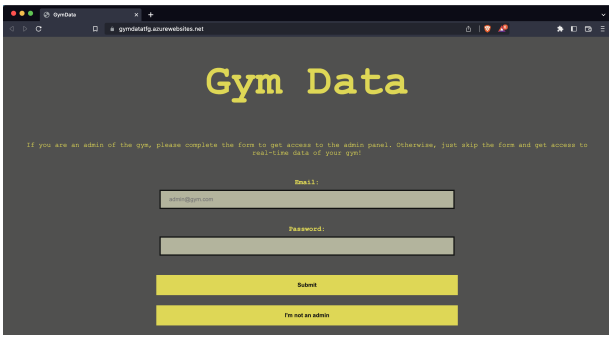

Fig. 11: Webpage hosted in Azure servers

6.4.1.3 Firebase

Firebase is a platform for developing web and mobile applications developed by Google. Among many other services, Firebase offers user authentication applications. This is the functionality of Firebase that has been used in this project to be able to create administrator accounts that have access to a different dashboards than users.

In order to use Firebase, we must register an application with this Google service. Once the application is registered, we will be given credentials that uniquely identify our application.

The application offers multiple services, as mentioned. One of these services is for authenticating and managing users. This functionality allows us to choose which identification methods we will use, create accounts... In this case, our application uses email authentication, so that the user accounts we manage are identified by an email and a password. The accounts can be created from the Firebase console and the credentials simply need to be provided to the gym administrators. By entering these credentials in the website form, we can access the administrator panel. In order to use Firebase authentication, it is necessary to use a JavaScript script that can be consulted in the Github repository of the appendix (note that the code is not functional as the keys are missing and, for security reasons, cannot be shared).

6.4.2 Grafana

Grafana is a multi-platform open source analytics and interactive visualization web application. It provides charts and graphs that are fullfiled with information when Grafana is connected to supported data sources. Those charts and graphs are grouped into dashboards, that can be consulted using a web browser.

Grafana has been used to host the dashboards for both users and administrators.

6.4.2.1 Instalation

To install Grafana and make it run on a local server on your machine, just install the necessary dependencies and execute the commands detailed on its official website [12].

Once the server is up, it is listening on port 3000, so, to be able to work with it, just go to localhost:3000.

6.4.2.2 Grafana Cloud

If instead of wanting to host the Grafana server locally, we want it to be hosted remotely so that the dashboards are accessible from anywhere and we are not limited to being able to consult them only if we are connected to the same network as the local server, we can use Grafana Cloud.

This is a tool that works the same as the local server with the only difference that it is hosted on the servers offered by Grafana. To have access to this tool, all you have to do is register on the Grafana website [11] and start creating the panels in the same way that we will do it locally.

This service is free in its basic version and has a subscription method depending on the needs of each application.

6.4.2.3 Conexion with Azure Data Explorer

Grafana, in a similar way to Home Assistant, has plugins developed by the community. Those plugins work with both versions: local Grafana server and Grafana Cloud.

One of these plugins allows to connect directly to the Azure application that has access to the database.

To do so, is mandatory to install the plugin and fill in the fields as shown in figure 12:

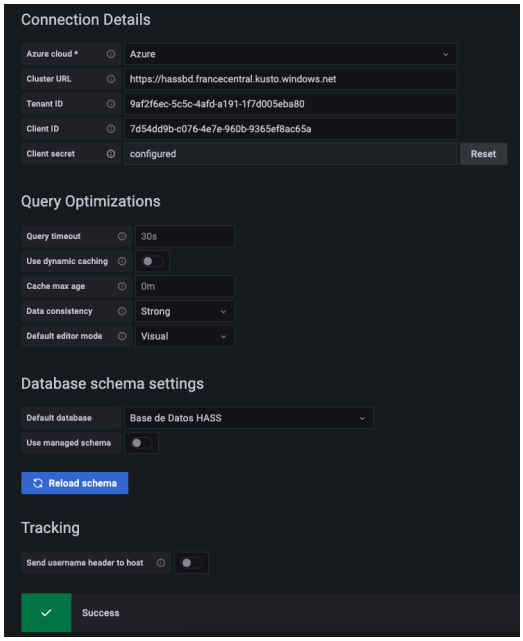

Fig. 12: Connection between Grafana and the Azure data base.

The Cluster URL parameter corresponds to the cluster where our database is running, while ClientID and TenantID correspond to two attributes of the application. Finally, the secret also corresponds to the application and can be created from the application through the Azure Active Directory platform.

Once this is done, it will be enough to choose which of the cluster's databases is the one we want to query (in our case, the only one there is) and test the connection. If it is correct, a confirmation message will be displayed as shown in the image.

6.4.2.4 Design of the panels

Once we have the data accessible in Grafana, it is time to design the panels in which to display the valuable information we have obtained with our system.

As said before, those panels are grouped into dashboards. In each dashboard it's possible to have multiple panels that are used to display information in different formats depending on convenience: bar charts, tables, pie charts...

Each of these panels executes queries to the database that you specify to obtain the information to be displayed. Figure 13 shows an example of a query that is made to display the temperature over time:

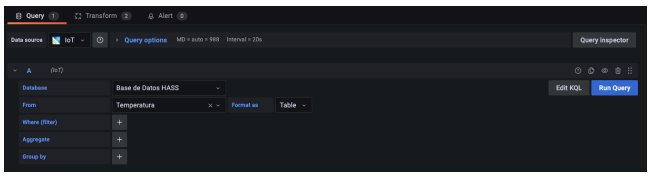

Fig. 13: Query to get the Temperature information in Grafana.

In this case, not only has the query become necessary but some modifications have had to be made, as shown in figure 14.

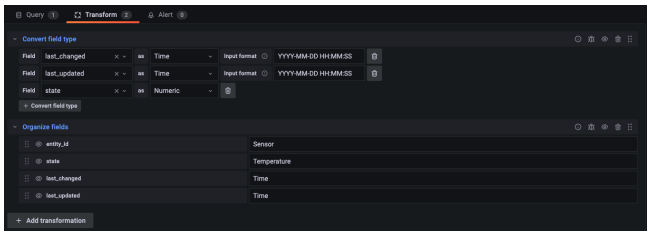

Fig. 14: Transformations applied to the query.

These modifications are called transformations in Grafana, and they allow us to work on the data obtained from the database to adapt it to the Grafana data types for the panel to display the information. For example, in the particular case of the query above, two transformations have been required: one that allows us to change the data type and format of the information (for example, transform a String to a Number or specify the format of the time data) so that Grafana is able to process it and another one that allows us to rename the fields returned by the query.

It is important to note that these transformations can be performed without fear of modifying or losing data from persistent storage. The change of data type is done internally in Grafana and does not affect the database stored in Azure; in fact, the application to which Grafana connects to obtain the data only has read permission, so, as said, transformations can be performed without fear of modifying or losing data from persistent storage.

These dashboards, in addition to queries, offer a great deal of customisation in the display of data. It is possible to adjust the scale of the axes, the colours in which the graph is displayed according to certain thresholds...

These customizations do not require code and are quite intuitive. Figure 15 an image of the panel to which the queries and transformations explained above correspond:

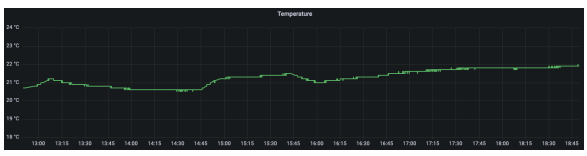

Fig. 15: An example of a simple Grafana panel.

This panel is very simple but it serves to give an idea of what a panel is. As will be seen later, the final panels are much more elaborate.

7 Final Result

Finally, we have achieved the goal of the work: be able create a proof-of-concept of a system than is able to give the user (and the administrator) a way to consult real-time and historical data on their training center through a complete IoT system.

In the previous pages, all the pieces that make up the system have been explained in detail. Since a lot of information has been provided, it is now a good time to reorganize ideas and make a general outline of the project in order to understand how the final project looks when all the pieces of the puzzle are together. Figure 16 shows a block diagram of the system:

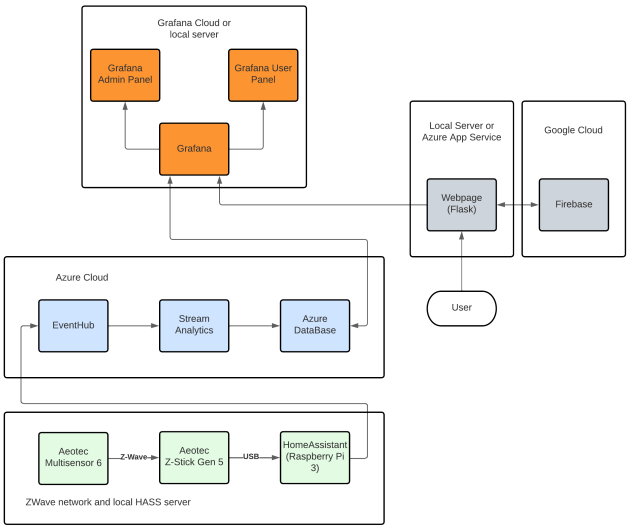

Fig. 16: Final block diagram

Basically, the sensors are capturing information and this is sent to the hub using the ZWave protocol. From the hub, the information is transmitted to HomeAssistant via USB. In turn, HomeAssistant spits out all this data to the Azure EventHub. This raw data is treated using Azure jobs and once the data of interest is obtained, it is stored in a database hosted by Azure. It is this database that Grafana accesses to put the information on the panels. This can be done whether Grafana is on-premises or using Grafana Cloud services. So, when the user or administrator wants to consult the data of the gym, they access a website that can be both in local (and then, to access the web page, the user must be on the same network as the server) or hosted by Azure (and then it can be accessed from anywhere) and, depending on their role (user or administrator), they are authenticated using Firebase if necessary, and are redirected to the corresponding Grafana panel.

Figures 17 and 18 show the panels, both for the administrator (Fig. 17) and the user (Fig. 18):

Fig. 17: Administrator panel

Fig. 18: User panel

As we can see, the information displayed is very different. For the plain user, only a history of temperatures in the last 12 hours as well as the current temperature and humidity is shown; the type of information that brings value to this type of user. On the other hand, the administrator panel shows much more information, such as temperature and humidity histograms. This information may be valuable to the gym administrator as, based on this and their requirements, they can make certain decisions (e.g.: automate the air conditioning based on the data). This is providing an extra value to the data for the managers of the entity.

In addition, the administrator dashboard allows it to be edited while the user dashboard has this feature disabled, since users should only be able to consult them.

8 Conclusion and Future Lines

Finally, it has been possible to develop the system that had been proposed in the objectives with the added value that it is not only useful for the user but also for the room administrator.

In addition, two options have been offered: either that the entire system is hosted in the cloud or have a large part of it locally, thus achieving a product that can be adapted to the needs of each consumer.

This project has served to more than solidly justify the viability of a project of this kind: a sensor system that is easily expandable and that sends information that is captured and processed in the cloud with high scalability to end up being presented to the end user in different ways depending on their role.

Furthermore, the possible future lines of work are clearly defined: to increase the network of sensors and to implement more Stream Analytics jobs to be able to capture more information from the environment and thus provide the user and especially the room manager with quality information through which to make the appropriate decisions or automate certain tasks.

In a more user-driven approach, sensors could be used to detect which machines are free and which are occupied. In an administrator-driven approach, the Z-Wave network could be extended to be able to detect, for example, possible water leaks in the changing rooms and correct them instantly. In an approach for both, sensors could be used to monitor the capacity of the gym and individual rooms, thus providing valuable information to both parties.

In terms of analysis and automation, work could be implemented to establish correlations between variables and, for example, automatically switch the air conditioning on or off at a certain time.

The possibilities are almost limitless, and which lines of work to follow will depend largely on the needs of each of the application's users.

In any case, and as already mentioned, a solid base has been laid on which to continue building in a more efficient way.

REFERENCES

[1] Aeotec. *MultiSensor 6 user guide*. Aeotec. October 2022.

[2] Aeotec. *Z-Stick Gen5+ user guide*. Aeotec. October 2022.

[3] BalenaEtcher. *balenaEtcher*. September 2022. URL: https://www.balena.io/etcher/.

[4] CleverTap. *37 Mobile Payment Statistics Marketers Need*. October 2022. URL: https://clevertap.com/blog/mobile-payment-statistics/#infographic.

[5] Ferran Palma Comas. *GymApp with IoT sensors and controller hub, cloud repository and control from web app*. 2023. URL: https://github.com/ferranpalma/GymApp-with-IoT-sensors-and-controller-hub-cloud-repository-and-control-from-web-app.

[6] Azure Contributors. *Quickstart: Deploy a Python (Django or Flask) web app to Azure App Service*. January 2023. URL: https://learn.microsoft.com/en-us/azure/app-service/quickstart-python?tabs=flask.

[7] Azure Contributors. *Visualize data from Azure Data Explorer in Grafana*. December 2022. URL: https://learn.microsoft.com/en-us/azure/data-explorer/grafana.

[8] Azure Contributors. *What is a resource group*. October 2022. URL: https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/manage-resource-groups-portal#what-is-a-resource-group.

[9] Deloitte. *European Health Fitness Market Report 2019*. October 2022. URL: https://www2.deloitte.com/content/dam/Deloitte/es/Documents/tecnologia-media-telecomunicaciones/Deloitte-ES-TMT-European-Health-Fitness-Market-2019.pdf.

[10] Raspberry Pi Foundation. *TitRaspberry Pi 3 Model B+*. Raspberry Pi Foundation. September 2022.

[11] Grafana. *Grafana*. December 2022. URL: https://grafana.com/.

[12] Grafana. *Install Grafana*. December 2022. URL: https://grafana.com/docs/grafana/latest/setup-grafana/installation/.

[13] HomeAssistant. September 2022. URL: https://www.home-assistant.io/.

[14] HomeAssistant. *HASS installer*. September 2022. URL: https://github.com/home-assistant/operating-system/releases/download/9.3/haos_rpi3-64-9.3.img.xz.

[15] HomeAssistant. *Install HASS in Raspberry Pi*. September 2022. URL: https://www.home-assistant.io/installation/raspberrypi.

[16] INE. *Equipamiento y uso de TIC en los hogares - Año 2022*. October 2022. URL: https://www.ine.es/dyngs/INEbase/es/operacion.htm?c=Estadistica_C&cid=1254736176741&menu=ultiDatos&idp=1254735576692.

[17] INE. *Indicadores sobre uso TIC en las empresas - Años 2021-2022*. October 2022. URL: https://www.ine.es/dyngs/INEbase/es/operacion.htm?c=Estadistica_C&cid=1254736176743&menu=ultiDatos&idp=1254735576692.

[18] Kanbanize. *¿Qué es Kanban? Explicación para principiantes.* September 2022. URL: https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban.

[19] Twitter. *Twitter Metrics 2019-2020-2021*. October 2022. URL: https://s22.q4cdn.com/826641620/files/doc_financials/2021/q4/Final-Q4'21-Selected-Metrics-and-Financials.pdf.