TÉCNICO LISBOA

WIRELESS MOBILE NETWORKS

2nd Semester | 2018/2019 | MEEC

# WIRELESS MOBILE NETWORKS

## *Controlling Temperature around temperature sensitive electronic devices*


José Velez | Nº 81358

Vasco Amaral | Nº 81532


Lecturer:

Prof. António Manuel Raminhos Cordeiro Grilo

June 2019

## 1. Introduction

The main goal of this project is to implement and develop an IoT ecosystem to monitor and maintain air temperature around temperature-sensitive electronic devices, while displaying a User Friendly Interface in an Android Application, built with Google's Flutter Framework. Here, the users can control to which temperature is the electronic device sensitive to. They can also see the several data evolution, in beautifully displayed charts, like the temperature and the times the fan was turned on/off. The final implementation will be a system composed of 4 interconnected modules to address every necessity of this ecosystem. A database which is the memory, where the records are kept. A server which is the heart of the system, it is the connection between the other 3 modules and is responsible for all the data processing. A mobile App which is the user only connection to the system, from here the user can see and control everything that is part of the ecosystem. Last, is the only physical part of the ecosystem which is the Arduino, which actually measures the temperature and makes the fan work in order to reduce it, if necessary.

## 2. Architecture Specification

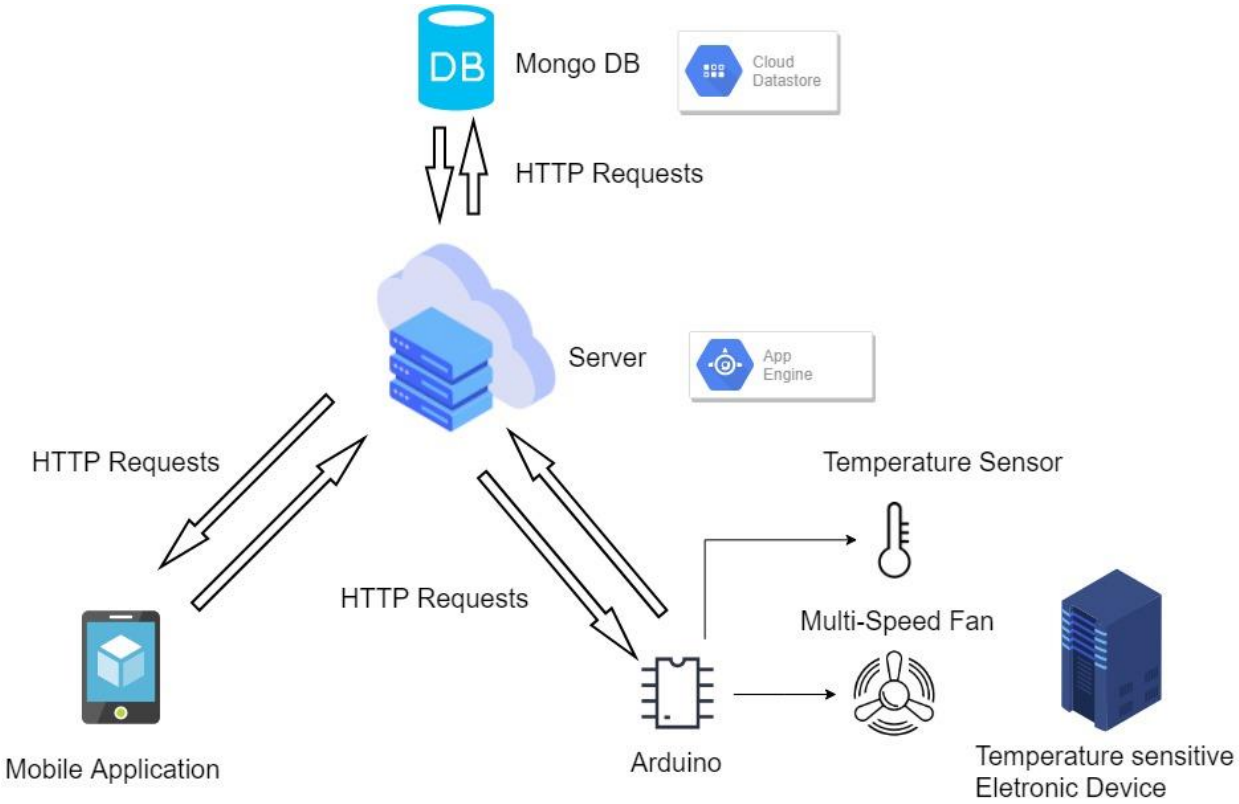


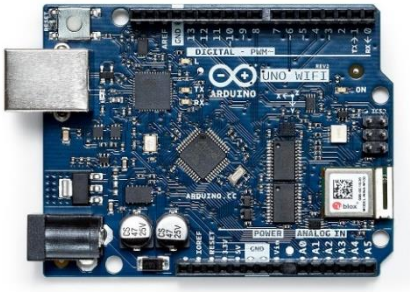

*Image 1 - Scheme of the Architecture of the system.*

The architecture of this project has 4 main components. A sensor, from which the temperature measurements will be taken from through an Arduino device, equipped with the necessary Wi-Fi module to send all the data to a web server. The main server, which can be considered to be the brain of the system, has the purpose of managing the communications between the different components while storing the data from the sensors in a database (a MongoDB database hosted in Cloud Datastore of Google Cloud Platform). The latter will also serve as a Web Service, providing several Rest APIs. One to serve as an Auth manager, where the client will send its authentication information, and the server will query the Database in order to verify if that user is already stored there. Two APIs to provide the client with the necessary data to display a temperature variation chart and a line chart with the status of the fan over time, both in the last 60 seconds, querying the Database to get the necessary information. Another one to allow the client to change the threshold temperature, from which the Arduino will turn on the fan. The server receives the request with the value from the client, and then sends it through another http request to the Arduino. At the same time, the server is constantly receiving the data collected by the temperature sensor, every 2 seconds, and storing it in the Database, in order to display that information later in the client, if requested.
The actuator will be a multi-level fan, also controlled by the Arduino, used to lower the temperature if required. The client will be implemented as an Android application, where the authenticated user will be able to observe the variations of temperature around the monitored device, the periods of time when the fan had to be used and its effect. The user will also be able to configure the threshold temperature, that will be set to turn on the fan as well as visualize the temperature variation and the periods the fan was on, in two neat looking animated charts.

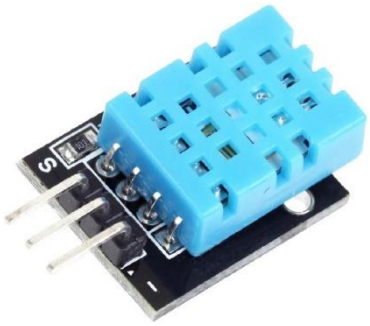
## 3. Sensor/Actuator Node

### 3.1 Hardware

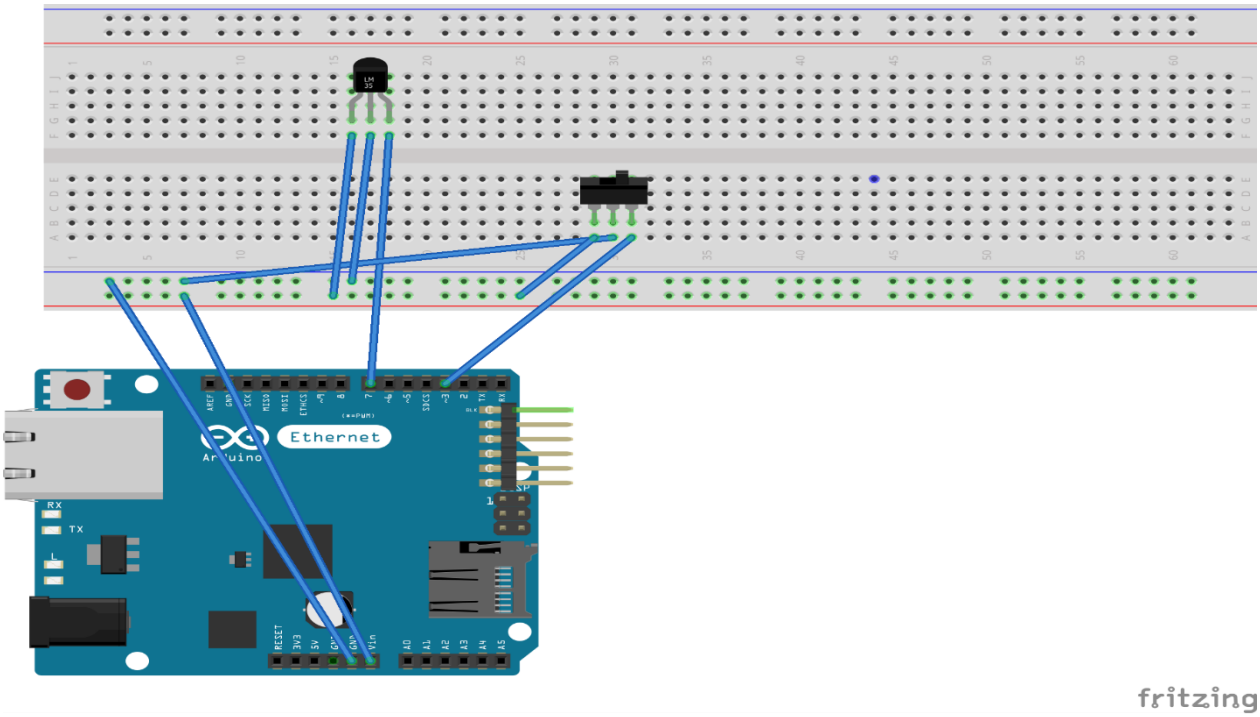
The physical system is composed of 4 modules, being 2 of them integrated, the Arduino Uno Wifi Rev2, which comes with an integrated WiFi module provided by the ECC608 crypto chip accelerator. The other 2 modules are the DHT11 temperature sensor and the DFR0332 fan module.



The DHT11 temperature and humidity sensor is connected to the Arduino through 3 pins, 2 for power (ground and VCC) and a last one to pass the data to the microcontroller, the pin 7, in this case. The library DHT11 made for Arduino, is used to acquire the temperature data, from the sensor, in Celsius degrees, with a resolution of 1ºC.



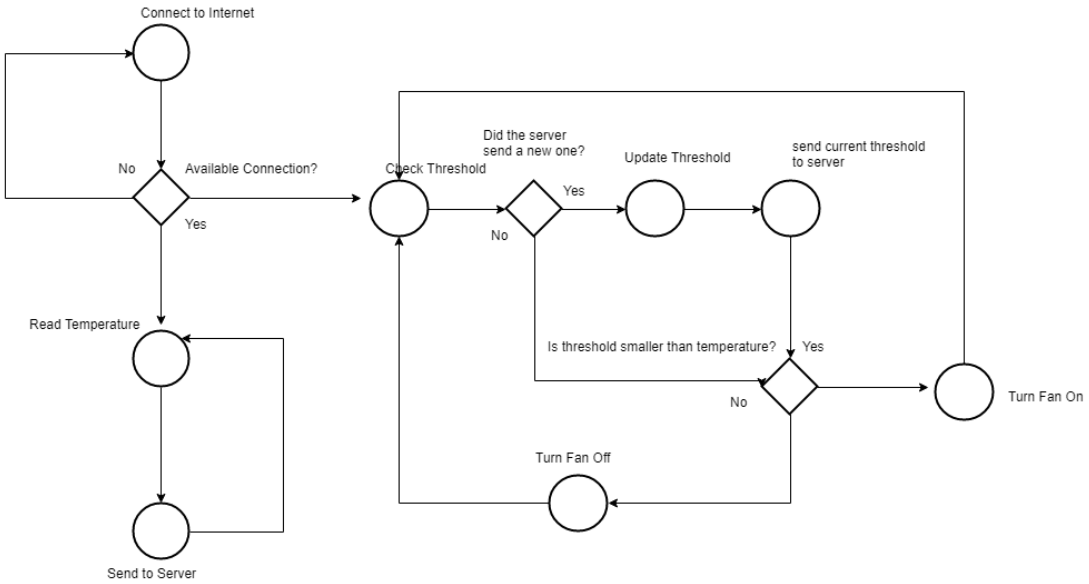
In order to cool the system, the DFR0332 complex actuator is used. The speed of the fan can be controlled by the entry voltage, making it possible to control the cooling of the system according to the data measured by the temperature sensor. This module is also connected to the Arduino through 3 pins, 2 for power (ground and VCC) and a last one to receive the values, that control the fan speed, from the microcontroller, the pin 3, which needs to be a PWM pin.




*Fritzing Diagram of the setup (LDR11 on the left and an S1 representing a fan on the right, since there wasn't any in Fritizing)*

### 3.2 Software

The software that runs in the microcontroller has 2 main functions. The first one and simpler, is the direct connection with it's sensor and actuator, the sensor measures the temperature which is later compared against the current threshold value, if it is greater than it, the fan will increase it's speed, otherwise, the speed will be decreased.
At the same time the Arduino has a previously established connection with the server, every 2 seconds the Arduino sends the current measures through an HTTP request (described later). The Arduino is also expecting messages from the server to change the current value set for the temperature threshold that defines at which point the fan should be turned on.




*Flowchart to represent the Arduino's Software logic.*

### 4. Cloud Server

The cloud server, alongside with the independent database, was deployed on the Google Cloud Platform, managed by the App Engine. The server is the heart of the system, being the only module to communicate with all the others, every feature of the system is processed by it.
It offers 3 main services, each with several API's.

Google Cloud Platform

**Users:**

This service is meant to manage the users that have access to the system, it offers API's to create, read, update and delete (CRUD), as well as a login API, accessed by the mobile app to authenticate the user.

POST /users/login {username: “ “, password: “ “}

**Measures:**
This service is responsible for receiving the constant measurements and actions taken by the Arduino, process them and store it in the database. It is also responsible for providing the client with the all the processed data it stores, so it can show the manipulated data in different charts in the mobile app.
The measures service provides API’s for all the CRUD operations, as well as an API to store readings from the Arduino, and different API’s to provide the client with prepared data for several charts.

POST /measures/Arduino {temperature: “ “, curr_threshold: “ “}
GET /measures/last
GET /measures/tempChart
GET /measures/fanChart

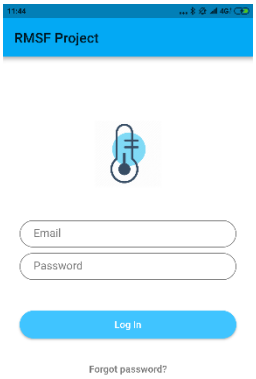
**Temperature control:**
This service is responsible to control the threshold temperature in the Arduino, it is ready to receive a new value from the client and send it to the Arduino so it can update it.

POST /tempcontrol/threshold {threshold: “ “}

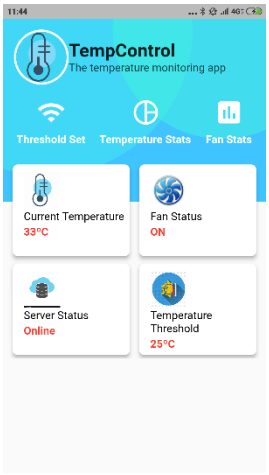
## 5. Mobile Application

A mobile application is needed to allow the user to monitor all the information, through real time dashboards with charts to display the relevant information, and to configure the temperature threshold. The client was developed using an open-source mobile application development framework created by Google called Flutter.
The App was built keeping the User Experience in mind, being very intuitive and user friendly, with some cool looking User Interfaces and animations, and it will communicate with the server specific endpoints through HTTP requests (using the Flutter’s http communications library along with a library to handle json objects and strings to create and parse the strings to be sent and received), in order to receive the information saved in the database, which the server has access to, as well as send information to it, for example new configurations to be passed to the Arduino.

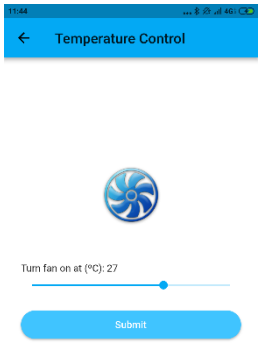
The main features of this mobile application will be the following:




- Log-in Screen, that will contain the app name and logo, as well as a form with username, a password field and a button to submit the form and send a POST request to the server, in order for the server to verify if those credentials are already stored in the Database, redirecting the user to the real time dashboard screen if that case is true. If the credentials are wrong, it will clear the form and send an informative message to the user. This way only authenticated users will be able to continue throughout the rest of the application.

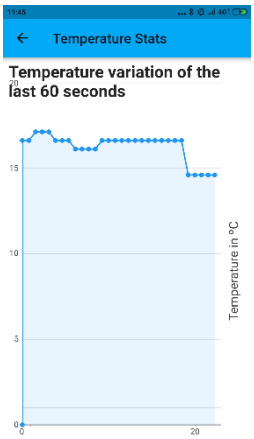



- The Real Time Dashboard Screen, while loading will send a GET request to the server, to receive all the necessary data to be displayed in that exact moment, such as the temperature (in ºC), the status of the fan (whether it is turned on or off), the threshold temperature (in ºC), and its connection status with the server. It also has 3 navigation buttons, which will redirect the user to three different pages, the Threshold Set Screen, the Temperature Stats Screen and the Fan Stats Screen.

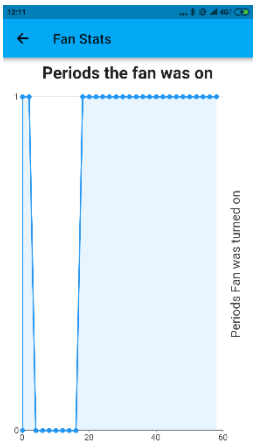



- The Threshold Set Screen will display a a sliding widget, to choose the threshold value and a submit button, that when pressed sends a POST request to the server with the new threshold value, so that the server sends that value to the Arduino, and from then on, until another change, the fan will be turned on if the temperature reaches that value.

- The Temperature Stats Screen displays an animated point line chart, with the information of the temperature variation of the last 60s. While loading the page, the app sends a GET request to the server in order to obtain the necessary data to display the chart.
  This chart was built using the “flutter charts” library.




- The Fan Stats Screen displays another animated point line chart, this time with the periods of time that the fan was turned on in the last 60 seconds. While loading the page the app sends a GET request to the server, which will then query the database for this information, and will show the value of 1 if the fan was on (every 2 seconds, which is the periodicity with which the Arduino sends the sensors information to the server), and 0 if the fan was turned off.




## 6. System Testing

While implementing the system, developing the mobile application, Arduino Software and the nodeJS server, we used several methods to test and debug it.
To test the server we ran it locally in order to print in the terminal several debug variables, to make sure it was handling all the communications and database accesses correctly.
To test the mobile app, we used an Android emulator to make sure the UI was always displayed correctly and in the most efficient way, and the Android Studio’s Debug Console to print several values sent and received and to make sure we were handling correctly the parsing and creation of JSON objects to strings and doubles, which was an interesting challenge, since it was the first time using the

Dart language (Flutter's development language) and due to it not having that many support online for being a recent Framework.
To test the sensors and actuators, we simulated the variations of temperature putting the sensor close to the heat vent of the computer in order to force the increase of temperature, since it replicates in a good manner the main goal of this project, that is to monitor the temperature around electronic devices that are easily heated and are sensible to high temperatures (like most ones). Along with this, by watching the current temperature variations, we would constantly change the threshold value so we could watch the fan reaction working accordingly.

## 7. Conclusion

Developing an IoT ecosystem was one of the most interesting projects we have developed so far, even with all the technical difficulties that came with it. Since we built everything from root (developing our own APIs in the server to attend several different types of requests), and used a very recent mobile application development framework, we faced several problems without the help of abundant online documentation and support, which slowed the development and debug in a considerate manner. Nevertheless, we believe to have been successful in delivering an interesting project, with good results and with a real world application for it.