# DOCUMENTATION MEDITEMP

Guillaume Lefrant gtvl2

## THE SCENARIO

In medical facilities, the products have to be kept in refrigerated places. This ensure that they don't get wasted away. It's important that most of them are kept at a specific and regular temperature.

MediTemp is the perfect solution for these facilities. It is a tool that will allow an administrator to control the products kept in his sector. By using this system, he will be able to see the temperature of each refrigerator and the products kept in it. And if there is a problem, he will be notified so he can adjust as soon as possible the system or dispatch a technician.

## PRELIMINARY REMARKS

This project is a just prototype and not a full scale system. It acts like a demonstration of what could be done. Thus, it can be extended pretty easily. There are comments in the code where it could be extended to make a real product.

A video of the prototype is available on Youtube.

In the video, I do not use the MBED device but the system has been tested with it too.

The code for MBED is available on the MBED website.

# The Components

## IoT Device

This device is similar to the K64F but with just the required components on it.

1. A permanent power source. The best would be to have the devices plugged to a power source or to the refrigerator.
2. Ethernet Interface to connect the device to the network.
3. A smaller CPU to limit heat impact.
4. Temperature Sensor. I think there is no need for an explanation on this one.

And that's all. The goal of this device is to regularly measure the temperature of the refrigerator and to send it to the server using MQTT protocol. Each device should be installed inside a given refrigerator.

## Server

It is a server written in NodeJS using ExpressJS framework. It includes a MQTT Client that listens to the publications of the devices and store them inside a MongoDB database. This server also includes a web interface which will be detailed below. Obviously, there is a MQTT Broker somewhere on the network which can be internal or external (online) as Mosquitto Server is an open-source solution. The server also includes few API endpoints which could be extended and used for mobile application or any type of raw access to the data.

## Web Interface

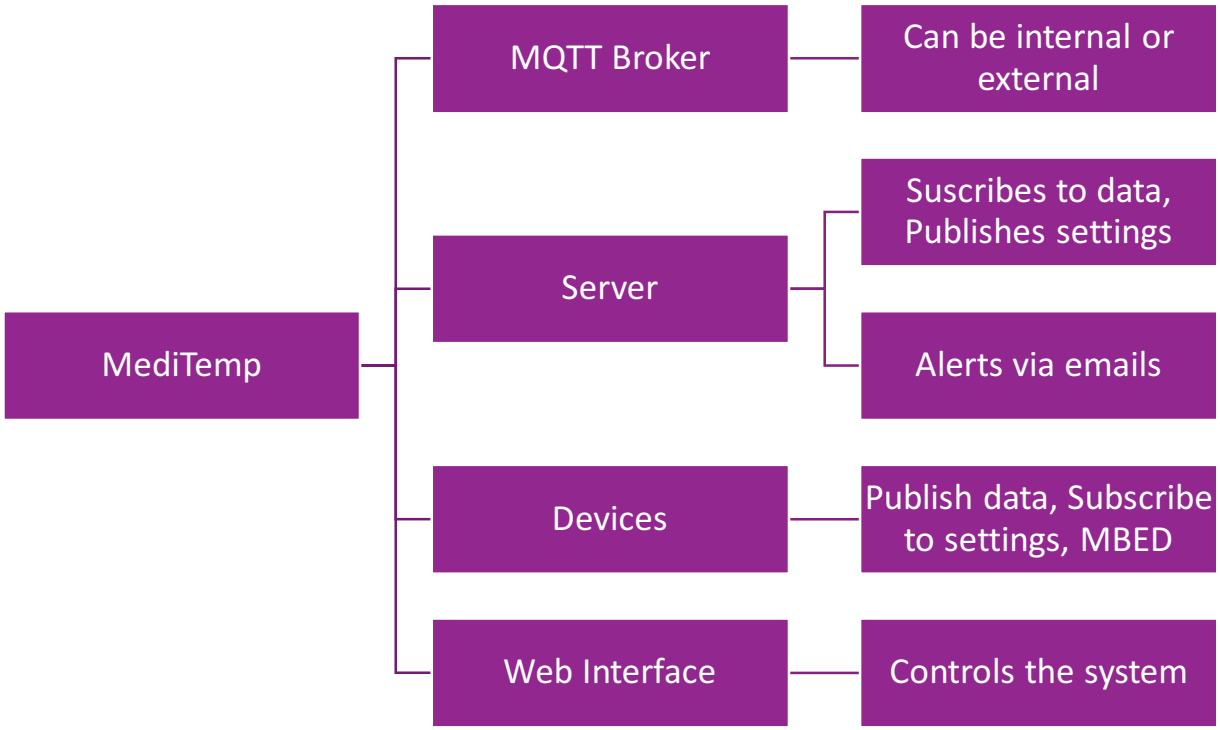
The Web Interface is built on top of the server and uses the data from the devices. It allows the administrator to monitor the refrigerators, assign products to refrigerator, keep track of inventory, change settings and to be alerted of any danger.

## Simulator

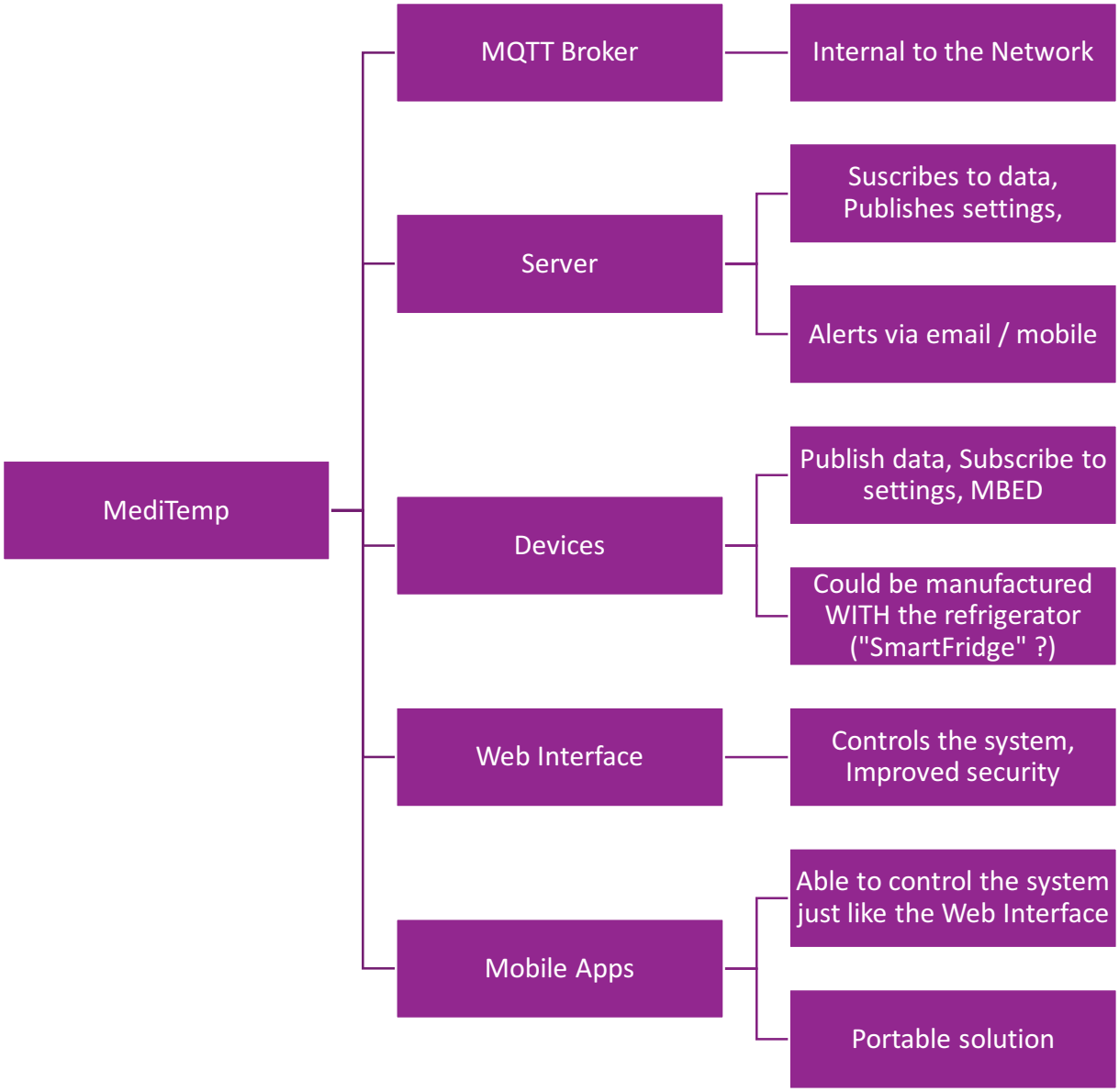
I've also included a Simulator written in NodeJS which reproduces the comportment of the MBED device because it's difficult to test it in real conditions. The Simulator is also written in NodeJS using the same MQTT package that the one for the Server.

# Architecture

## Current Prototyped Version



## Possible Full Scale Solution



## REMARKS

In the prototype, I do not provide authentication at any level but it must do present in a real system. The Server / Web Interface should be password protected (using PassportJS for instance) and not remotely accessible. To prevent any data corruption, the MQTT Broker should be accessible only from the company network and not from the outside as well.

The architecture I've created is highly extendable and doesn't require a lot of modifications to add new features.

## Installation & Usage

Everything is referenced in the README.md at the root of the project.

### MQTT Broker

A broker is required to make this system work. You can install your own from the Mosquitto Website, or use a provided one. In this prototype, I use the one provided by the University.

## Server

First, you need to install the packages:

*cd server && npm install*

Then, to launch the server:

*npm start*

If you need to enable the mailing alert system, you'll need to specify a SMTP server:

*MAILER_HOST="" MAILER_PORT="" MAILER_USERNAME="" MAILER_PASSWORD="" npm start*

### Web Interface

Add the UI components with:

*cd server/public && bower install*

The interface is launched at the same time as the server.

### Simulator

Again, the node packages first:

*cd simulator && npm install*

Then, launch it using:

*PRIVATE_ID="XXXX-XX-XXXX" npm start*

If you do not provide PRIVATE_ID, a new one will be generated.

### Device

Compile the software and flash the binary directly to the MBED device.