**Smart Home IoT Project**

# Borna Grgurić, Mario Vegh, Dorothy Phiri, Filip Medaković, Dominik Vidaković

# Introduction

In this document we will go through the process of developing a Smart Home platform from the idea at the beginning to the product at the end.

The original project name is Smart Home Gateway and Application. The project has the following specifications:

* a gateway that represents the central hub of the system
* IoT devices that send information to the gateway
* a server that receives sensor data from the gateway
* a client-side web application that:
  + shows sensor data received from the server
  + allows the user to control the IoT devices

We will explain the project architecture and development steps in the following order:

1. Raspberry Pi Gateway
2. Smart Sense Air sensor
3. Smart Plug
4. Server application
5. Web application
6. Android application

# The gateway

All communication from and to the gateway is made using the MQTT protocol. We recommend doing some research on how the protocol works before continuing. The only thing the gateway should be used for is running the MQTT Broker, but for simplicities sake, we added a python application to monitor and manage the MQTT message traffic.

# Smart Sense air sensor device

The sensors full name is Smart Sense AirQ indoor. The device has 3 parts that are relevant to this project:

* STM32 Cortex M3 microcontroller
* ESP8266 WiFi network processor
* The CO2, temperature and relative humidity sensors

In our case gateway communicates with this device via the ESP8266 WiFi network processor. We programmed the device using C++, the Arduino IDE and a CP2104 programmator. Fetching of the sensor data is done via serial port through the STM32 Cortex M3 microcontroller. The challenge in developing this part of the project is programming the communication from the ESP8266 WiFi network processor to the STM32 Cortex M3 microcontroller, and the other way around. The next step is programming the communication between the Smart Sense device and the gateway.

## Fetching the sensor data

The STM32 Cortex M3 microcontroller comes with a built in interface that is used by sending commands via serial port. To test some commands out, you can connect to the device by serial port directly and send commands for fetching the sensor data and receive the proper results. But for our project, this entire process needs to be automated, and that is done by deploying a C++ application to the device using the CP2104 programmator and Arduino IDE (+ some plugins). The commands are sent via serial port using the serial.write() function (dont forget to add a return (\r) input at the end of the command for it to be sent to the microcontroller). The response containing the requested data is then read from the serial port.

## Forwarding the sensor data to the gateway

The Smart Sense device should be in the same LAN as the gateway. Now we continue with programming in C++. When the ESP8266 WiFi network processor is connected to the WiFi, it should connect to the MQTT Broker running on the gateway. The air sensor data is then published via MQTT message to the broker. In our project, we are sending new data every 5 seconds.

## Controlling the Smart Sense device with MQTT commands

To control the Smart Sense device in any way, the device must first be able to recieve messages from the gateway. MQTT makes this easy for us. We just needed to add a bit more C++ code. This code subscribed us to a topic on the MQTT Broker running on the gateway. The subscribed topic will forward commands for the air sensor device, all that is left to do is program their functionality.

# Smart Plug

The smart plug needs no programming. When plugged into a electrical plug, it periodicaly sends data to multiple topics on the MQTT broker. The topic were the current electricity consuption is being sent is :

smarthome/node/83F91607004B1200/sensor/power/1027/value/power

The topic to turn the plug on/off is:

smarthome/node/83F91607004B1200/sensor/power/1027/switch/status

This topic accepts „true“ and „false“ as the commands to turn the plug on/off.

# Server

At the start of this project we were given a virtual machine with the specifications of our choice to use as we please. We used it as an accessible repository of all our work and a development enviroment in which we made the server application and the web application. Later we used the machine as the location where we deployed the server application as a cloud service, to make it accessible from any location. Our PostgreSQL database is also located on the virtual machine.

The server is programmed using Java Spring Boot, and it does the following:

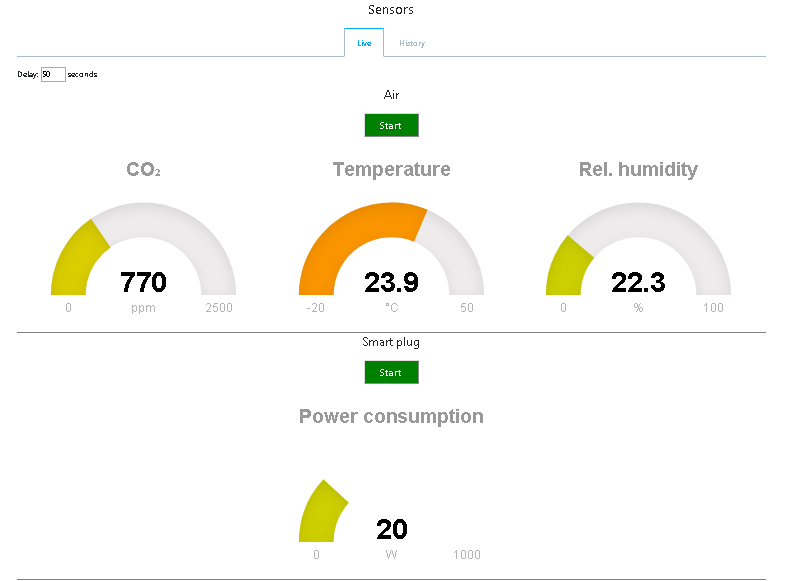
* Communicates with the database via Hibernate and JPA
* Communicates with the gateway via MQTT
* Communicates with the Web application via REST API

# Web application

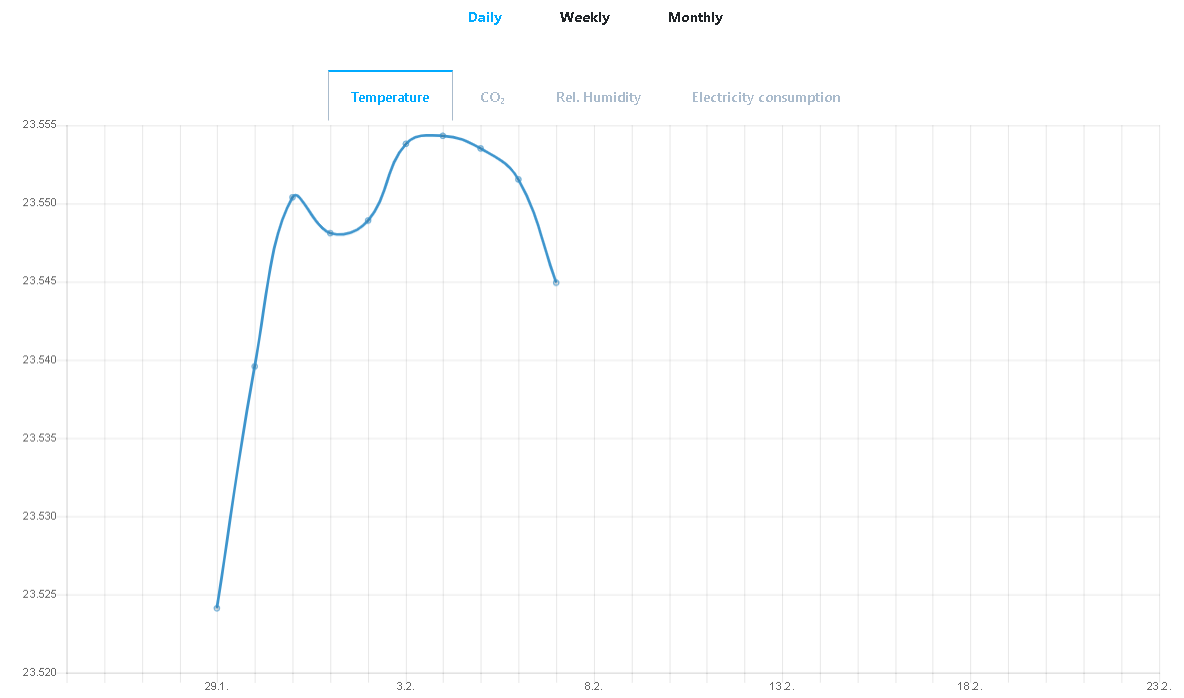
The web application is developed using Javascript. After a user registers and logs in, it asyncronicly fetches the latest sensor data and displays it to the user in a beautiful fashion. It also shows a history of recorded data using graphs. The graphs show the measurements of the :

* Last 24 hours
* Current week
* Last month

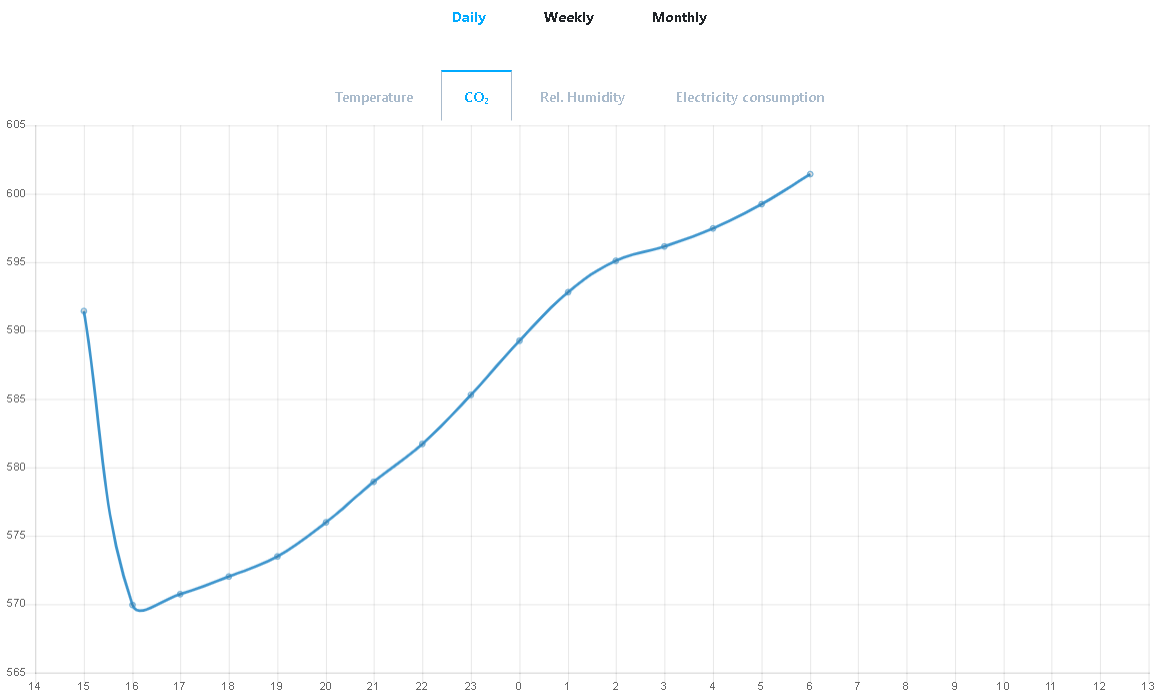
There are also buttons that turn the air sensor and the smart plug on/off.



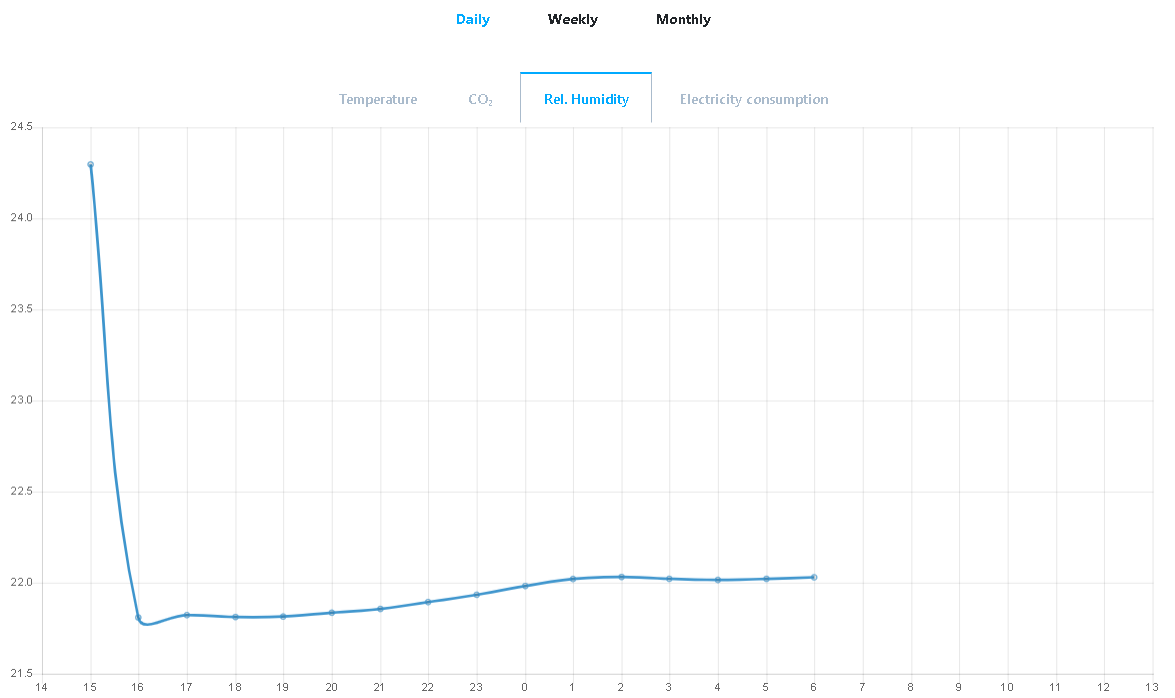
Picture 1. All sensors with data



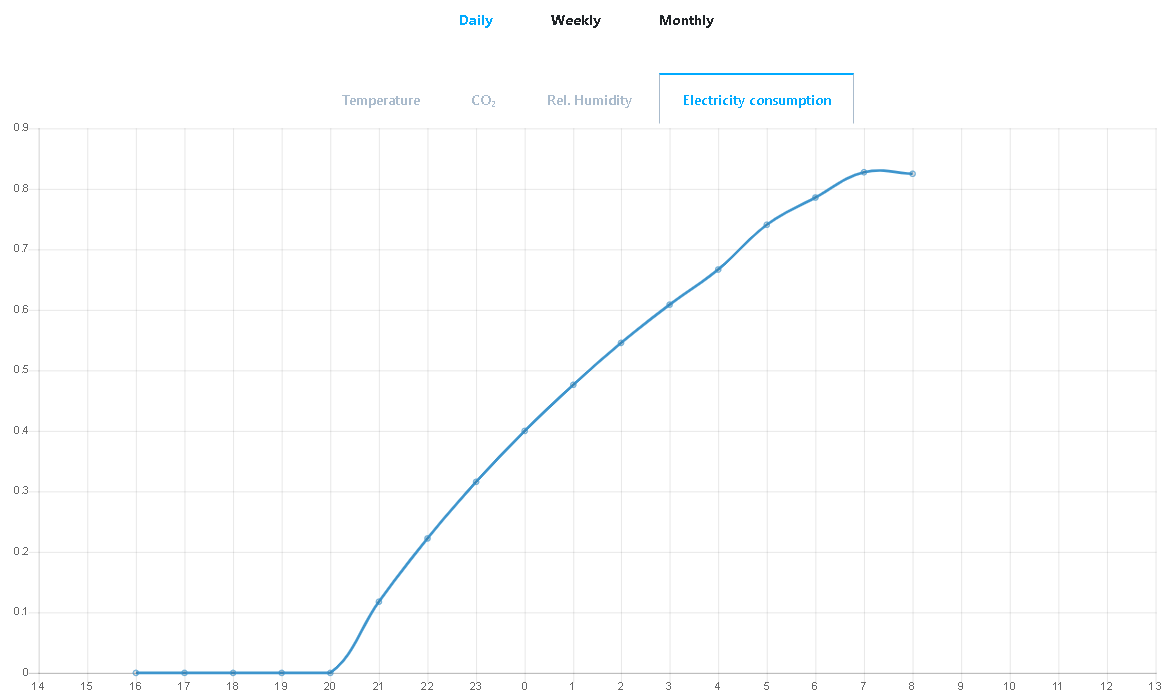
Picture 2. Temperature measurement



Picture 3. CO2 measurement



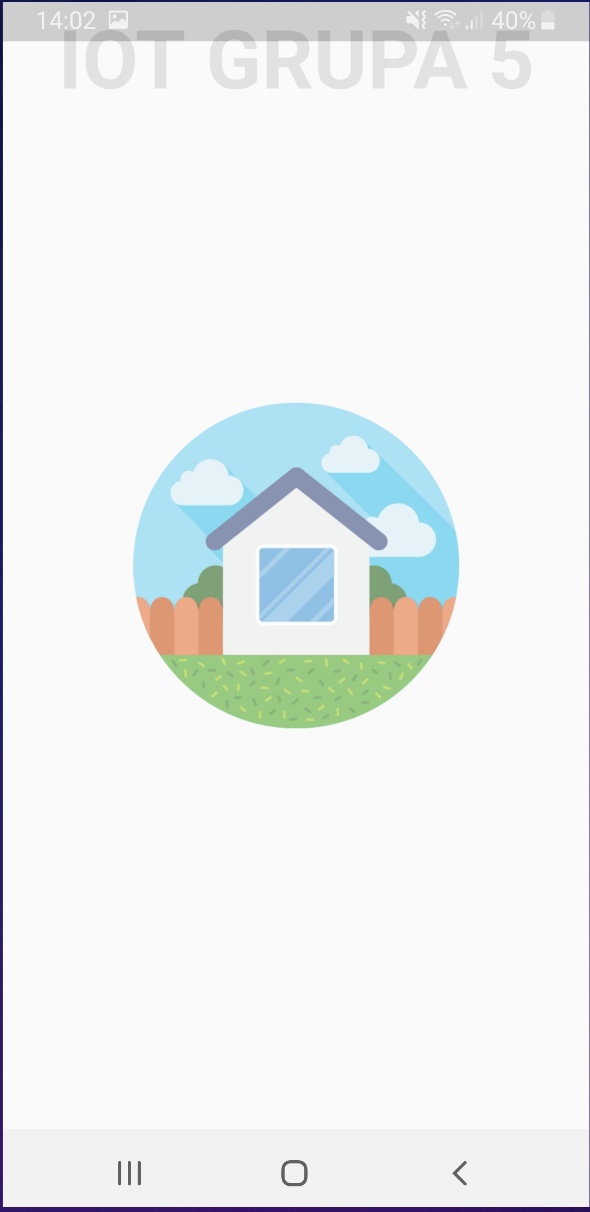
Picture 4. Humidity measurement



Picture 5. Electricity consumption

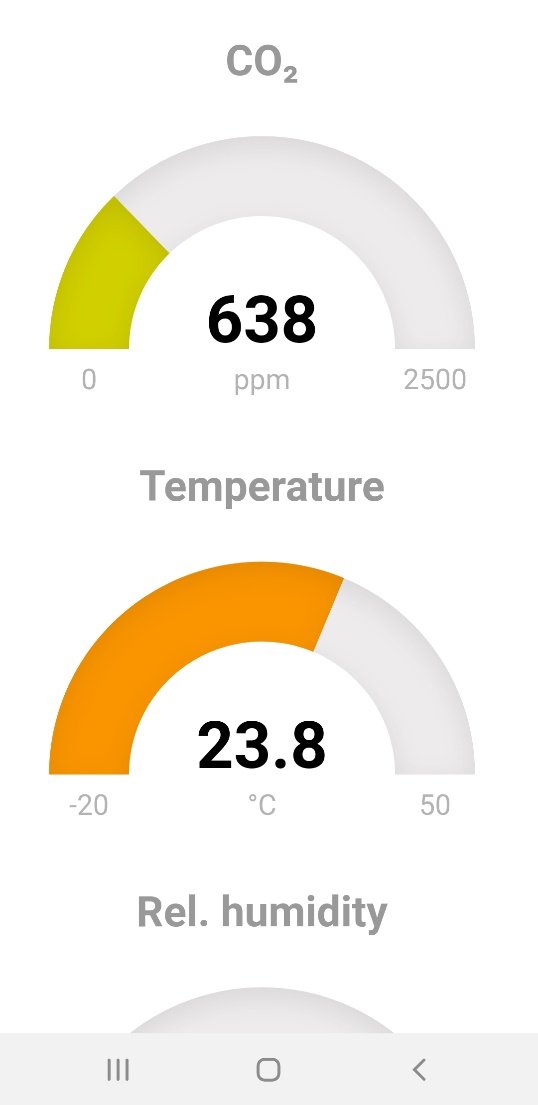
# Android application

The android application uses the webview andrioid component to connect to the webpage and show the temperature,CO2 ,humidity and electricity consumption on a mobile phone. This allows the users to see the state of their home and controll the sensors from any location. Our mobile application also uses broadcast receiver to check if the user is connected to the internet. if the user is not connected to internet the mobile application will show our users internet connection problem error as shown in Picture 9.



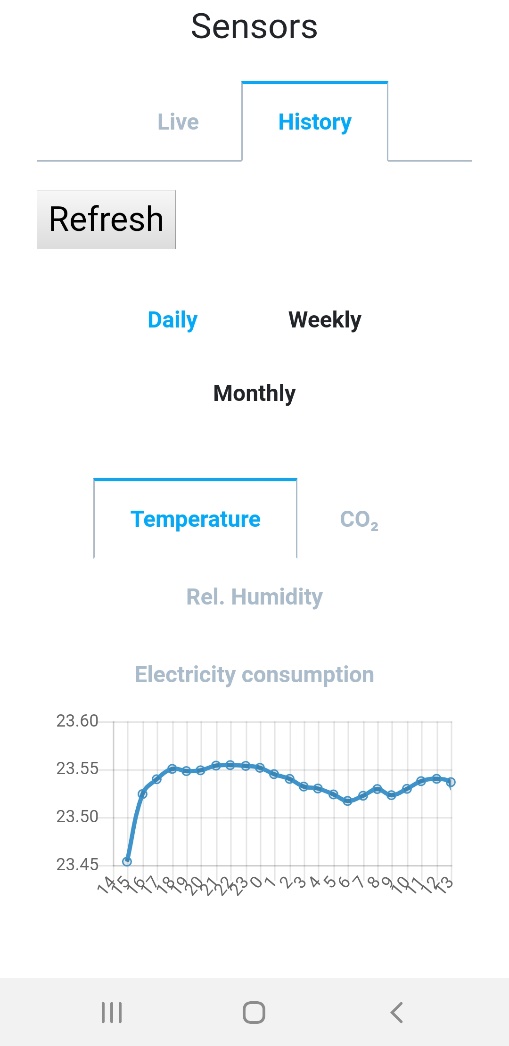
Picture 6. Splash screen

This is a splash screen that loads on every request to web application using the webview andriod component.



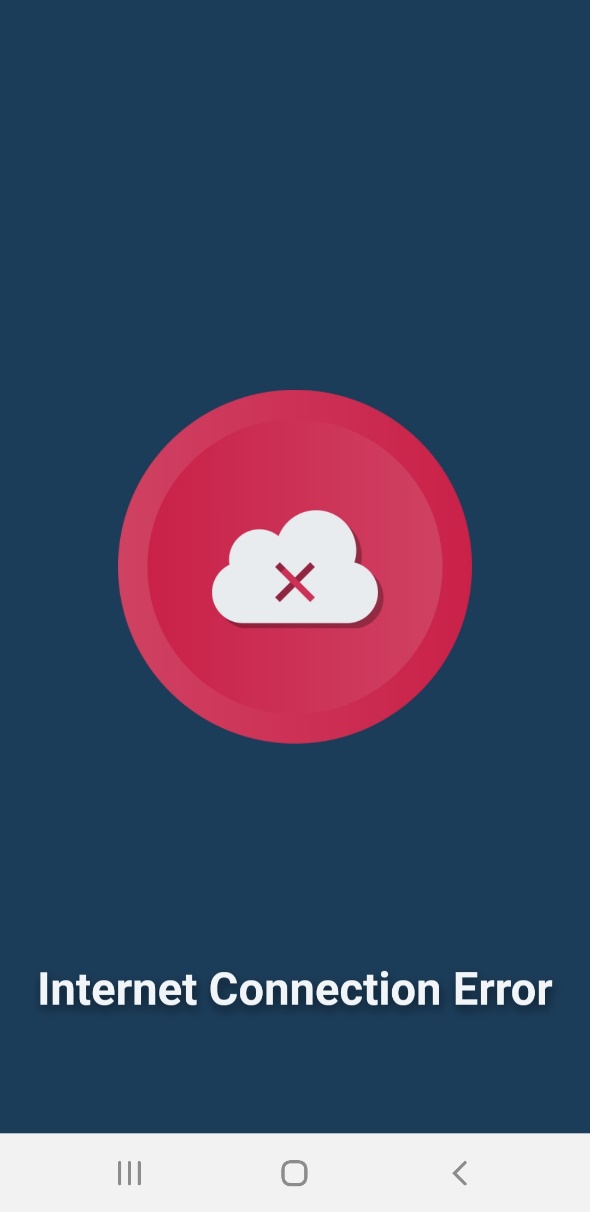
Picture 7. Mesurement on mobile view

These are live measurements from the wep page that loads every 5 seconds and are shown to the user on the mobile phone.



Picture 8. Temperature measurement

Apart from the live measurements the user can also see summary dailly, weekly and monthy of all measurements.



Picture 9. Internet connection error