**Technical design Plant Patrol**

****

**Luuk Vlug**

**500807974**

[**luuk.vlug@hva.nl**](mailto:luuk.vlug@hva.nl)

**Technical Design Plant Patrol**

**Index**

[**Overview of the webpage**](#_rt4wrg41ug6) **3**

[**Sensor result**](#_ut2audszw11v) **4**

[**The graph with sensor results**](#_wf24p9zeu8j4) **5**

[**Status of the plant**](#_7ogem1xplpet) **6**

[**Forecast API**](#_ktkpx2g7x189) **6**

**Sensor control from the website**

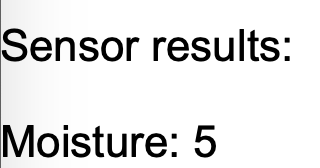
With an On/Off button, users can turn the moisture sensor on and off from the website. With the use of php, the status of the button (0 == off and 1 == on) is posted in a txt file. If the button is pressed, the status gets updated. In Arduino, the status is read and the sensor gets turned off or on depending on what the current status is.

If the sensor is turned on, the built-in led of the d1 mini will be on, else the led will turn off. I would have like to do this with an actual led, but due to circumstances, I was not able to obtain one. Following, a picture of the mentioned built-in led:



# Sensor result

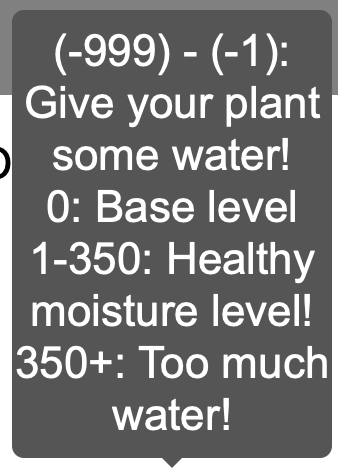
The next thing on the webpage is the sensor result. With the use of a Grove moisture sensor and some Arduino code, the sensor can measure moisture levels. In this case used for measuring the moisture of potting soil. The moisture value in Arduino is both shown in the serial monitor and sent to a Firebase real-time database. From this database, with the use of some Javascript code, the website is able to constantly display the latest value pushed to the database from Arduino. The result looks like this:



These results will keep updating as long as the sensor is connected and turned on. When looking at this value you might be confused as to what it means. Luckily below the moisture value, you will see this sentence:



This sentence is a popup. By clicking on it, you will be greeted with some text explaining the values. The popup looks like this (Note: these values may differ):



The moisture values are split up in categories, to give users a clear overview so they can know in what category their plant currently is. With this information users have an easier time deciding whether the plant needs water, has too much water or is just fine. These values are after calibration.

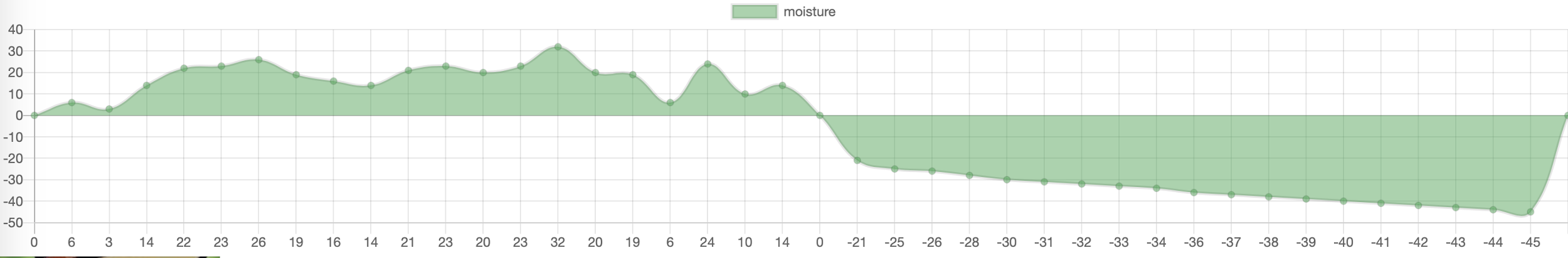
# The graph with sensor results

Below the written sensor result, you will find a button. This button is called ‘Clear graph’. When pressing this button, all values get cleared from the graph and the page gets reloaded. Without reloading the page, all values will still be visible until the user manually reloads the page. This is a workaround.



Everytime a new moisture result is picked up by the sensor, it is both displayed on the website, in the form of ‘Moisture: xx’ and the value is added to the graph. This chart is made with Javascript and basically stores the new values from the Firebase Real-Time database in the chart.

The graph where all these moisture values are added into looks like this:



As you can see in the picture, the graph can both display positive and negative values. This is for the calibration, explained in the user guide. Because sometimes the sensor can fluctuate a bit, trying to find the definitive value, this graph is a useful tool. If for example the sensor is still fluctuating, you don’t have to constantly look at the moisture value to see if it’s displaying equal values. You can instead just look at the graph and see if the values are still dropping. This graph is more of a quality of life feature.

# Status of the plant

Another feature are the pictures of the happy and sad plant you see below. Based on the moisture values (see popup for the ranges for happy/sad plants) the happy or sad plant picture is displayed to indicate the status of the plant. If you give the plant too much water, the plant will be sad, same with too little water. Some php code can display the right picture based on the moisture value.



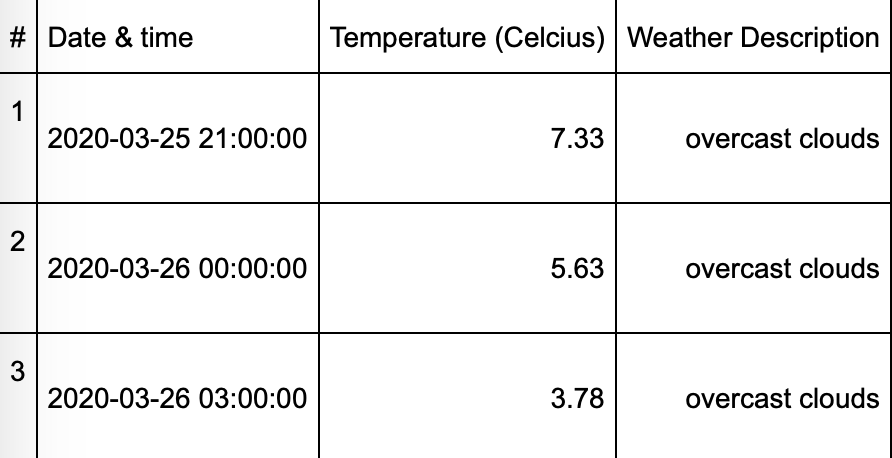
# Forecast API

The API used for Plant Patrol is the 3-hour forecast API from Openweathermap. The API is capable of giving the forecast of a city for 5 days, for 3-hour intervals each day. Because this was too much data, I limited it to 3 forecasts, so a range of 9 hours minimum with the interval. Openweathermap also has current weather API’s, but I found this one to be more relevant for my project.



The reason I chose to let users submit a location, is because there are no other limits to the project. Everything on the website is in English and others can buy these sensors themselves. For that specific reason, I did not want to limit the API to Amsterdam for example, but make this project relevant for most locations.

After submitting, the submitted city will stay in the textbox above, which acts as a ‘title’. Once you’ve submitted a city, it will be displayed in a table like this: (Amsterdam in this case)



These values are stored in a var in Javascript. These vars are received from a json format. By using the var xx = data [path] [path] [path] for example, I managed to get the right values stored and was able to display these values in the table as shown above.