Bild

[Software 3](#_Toc189035448)

[Website 3](#_Toc189035449)

[Konzept 3](#_Toc189035450)

[Recherche zu Version 1 3](#_Toc189035451)

[Version 1 3](#_Toc189035452)

[HTML 3](#_Toc189035453)

[JS 5](#_Toc189035454)

[Global variables 5](#_Toc189035455)

[Create\_graph\_xy 5](#_Toc189035456)

[canvas\_setting 5](#_Toc189035457)

[setTime 5](#_Toc189035458)

[time\_setting 6](#_Toc189035459)

[sensor\_setting 11](#_Toc189035460)

[measure\_text\_x\_axe\_and\_delete 14](#_Toc189035461)

[measure\_text\_y\_axe\_and\_delete 14](#_Toc189035462)

[Leere Funktionen 16](#_Toc189035463)

[CSS 16](#_Toc189035464)

[Endergebnis 16](#_Toc189035465)

[Hardware 17](#_Toc189035466)

[Allgemein 17](#_Toc189035467)

[Version 1 17](#_Toc189035468)

[Version 2 17](#_Toc189035469)

[Version 3 17](#_Toc189035470)

# Software

## Website

### Konzept

Als erstes musste ein Konzept entwickelt werden, wie die Website aussehen sollte. Es gab ein paar Anfangs-Bedingungen, dass die Website in der Lage sein sollte, Daten mit Filter Option darzustellen.

### Recherche zu Version 1

Zu aller erst wurde untersucht, wie wir nochmal ein Canvas in unsere Website initialisieren. Unsere Website besteht hauptsächlich (für den Anfang), aus einem JS, HTML, CSS -File. Das CSS-File ist für den Anfang irrelevant und ist erst bei der Verschönerung gegen Ende wichtig.

Jetzt kommen zuerst die Hauptelemente: 2x Comboboxen, 1x Button und 1x Canvas.

* Die Comboboxen sind für Filteroptionen, z.B. was für ein Sensor angezeigt werden soll und ab welcher Zeitangabe dies passieren soll.
* Der Button sollte dann diese Filteroptionen ausführen, indem man auf den raufklickt.
* Das Canvas stellt die Daten per Graph dar und gleichzeitig sollte es die Zeitangabe und den Sensortyp beim seinem gebildeten Graph anzeigen.

So das waren eines der wichtigsten Anfangsschritte, jetzt wird versucht diese Dinge umzusetzen.

### Version 1

Bei Version 1 waren mehrere Schritte notwendig, diese werde ich erklären:

#### HTML

Zu aller erst wurde natürlich mit dem HTML begonnen. Das HTML wurde größtenteils ohne die Hilfe vom Internet gemacht worden.

* Es gab gewisse Kleinigkeiten, wie „**StrokeThickness**“, wo gedacht war dies für ein paar Elemente einzubauen, dies hat jedoch nie funktioniert, auch mit mehrfacher Recherche, habe ich es nicht geschafft dies zu erreichen. Genauer gesagt, hat „**StrokeThickness**“ keinen Effekt gehabt, obwohl der Syntax gepasst hat und der Property Name richtig geschrieben worden ist. Vielleicht wird dieses Problem bei den nächsten Versionen behandelt.
* Genau, abgesehen davon gab es keine Probleme, wo recherchiert werden musste.

Sogleich wird der fertige HTML „Code“ gezeigt:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!---Showcasing our pieces of information of our sensors-->

    <title>Sensor-Data</title>

    <link rel="stylesheet" href="design.css">

    <script src="web\_code.js"></script>

</head>

<body style="background-color:darkslategrey;" onload="canvas\_setting()">

    <!--Vertical alignment for the objects-->

    <div class="stack\_ver">

        <!--Filter system-->

        <div class="stack\_hor"> <!--objects will be aligned in a horizontal way-->

            <!--select the sensors-->

            <select onchange="sensor\_setting()" id="box\_sens" style="background-color:dimgray;" name="Sensors">

                <option id="sens\_temp">Temperature</option>

                <option id="sens\_air\_moisure">Airmoisure</option>

                <option id="sens\_air\_pressure">Airpressure</option>

                <option id="sens\_gas">Gas</option>

            </select>

            <!--select the time-->

            <select onchange="time\_setting()" id="box\_time" style="background-color:dimgray;" name="Time">

                <option id="time\_today">Today</option>

                <option id="time\_yesterday">Yesterday</option>

                <option id="time\_this\_week">This Week</option>

                <option id="time\_last\_week">Last Week</option>

                <option id="time\_this\_month">This Month</option>

                <option id="time\_last\_month">Last Month</option>

                <option id="time\_this\_year">This Year</option>

                <option id="time\_last\_year">Last Year</option>

            </select>

            <!--apply button-->

            <button onclick="apply\_button()" style="background-color:darkslateblue;" id="butt\_apply">Apply</button>

        </div>

        <!--Canvas-->

        <!--xy box .... time x ... data y ---- doing it with a canvas-->

        <canvas  id="graph" height="800" width="1500" style="background-color:white;" style="border: 5px black"></canvas>

    </div>

</body>

</html>

#### JS

##### Allgemein

Der JS Code besteht aus mehreren Teilen, diese werden, Teil für Teil erklärt. Zusätzlich sind die Kommentare der Codezeilen auf Englisch, da ich dazu tendiere, in Englisch zu kommentieren.

##### Globale Variablen

*//for axe naming*

let x\_whole\_axe\_length = 0;

let y\_whole\_axe\_length = 0;

*//important for some naming and scaling the graph*

let x\_axe\_length = 0;

let y\_axe\_length = 0;

*//important for the scaling naming of each axes*

let y\_point\_length = 0;

let x\_point\_length = 0;

##### Create\_graph\_xy

function **create\_graph\_xy**(){

    const canvas = document.**getElementById**('graph');

    const ctx = canvas.**getContext**("2d"); *//getting the the features of the canvas, so i can promptly edit it AKA the context ... ctx*

*//getting the height and width*

    const width = canvas.width;

    const height = canvas.height;

*//fixing the center point, so i can draw lines with negative coordinates*

    ctx.**translate**(width\*0.2, height\*0.9); *//center point for all objects, where i will start to draw*

*//this point is ABSOlUTE, i do not have to change it or add it again!*

*//giving the lines a color and stroke width*

    ctx.lineWidth = 2;

    ctx.strokeStyle = '#000000';

*//beginnig to draw main lines*

*//y line*

    ctx.**beginPath**();

    ctx.**moveTo**(0, 0); *//0, 0 because we have a starting point ... translate(...)*

    ctx.**lineTo**(0, -height\*0.8);

    ctx.**stroke**(); *//drawing the elements*

*//x line*

    ctx.**beginPath**();

    ctx.**moveTo**(0, 0);

    ctx.**lineTo**(width\*0.7, 0);

    ctx.**stroke**(); *//drawing the elements*

*//beginning to draw arrows of the main lines*

*//arrow y*

    ctx.**beginPath**();

    ctx.**moveTo**(0, -height\*0.8);

    ctx.**lineTo**(width\*0.01, -height\*0.75);

    ctx.**lineTo**(-width\*0.01, -height\*0.75);

    ctx.**lineTo**(0, -height\*0.8);

    ctx.**stroke**();

    y\_whole\_axe\_length = height\*0.8; *//saving the whole length of the y axe*

    y\_axe\_length = height\*0.75; *//saving the length for later*

    y\_point\_length = y\_axe\_length\*0.93; *//important time data-showing*

*//arrow x*

    ctx.**beginPath**();

    ctx.**moveTo**(width\*0.7, 0);

    ctx.**lineTo**(width\*0.65, -height\*0.02);

    ctx.**lineTo**(width\*0.65, height\*0.02);

    ctx.**lineTo**(width\*0.7, 0);

    ctx.**stroke**();

    x\_whole\_axe\_length = width\*0.7; *//saving the whole length of the x axe*

    x\_axe\_length = width\*0.65; *//saving the length for later*

*//In addition, it depends on the arrow, because the lines of axe (x,y),*

*//would cross over the arrow, this would look ugly*

    x\_point\_length = x\_axe\_length\*0.98; *//important for the sensor data-showing*

**setTime**(); *//here i am sending httrequest, so our esp gets the current time*

}

##### canvas\_setting

function **canvas\_setting**(){

*//here i will draw a xy Graph with legends*

    const canvas = document.**getElementById**('graph');

    const ctx = canvas.**getContext**("2d"); *//getting the the features of the canvas, so i can promptly edit it AKA the context ... ctx*

*//getting the height and width*

    const width = canvas.width;

    const height = canvas.height;

*//clearing the canvas*

    ctx.**clearRect**(0, 0, width, height);

**create\_graph\_xy**();

*//why, you ask. Very simple, because these methods will update the graph x and y axe lines, the comboxes are sometimes bugged, and dont update at the beginning if you dont change the current item. I had several problems with that, maybe i will change this line of code in the future, but now*

**time\_setting**();

**sensor\_setting**();

}

##### setTime

function **setTime**() { *//from phillip, some parts of it is from me, but it was kinda changed*

    var check = false;

    try{

*// Get the current time from the client's device*

        var currentTime = new **Date**();

        var hour = currentTime.**getHours**();

        var minute = currentTime.**getMinutes**();

        var second = currentTime.**getSeconds**();

*// Send the time to the ESP32*

        var xhr = new **XMLHttpRequest**();

        xhr.**open**(

            "GET",

            "/set-time?hour=" +

              hour +

              "&minute=" +

              minute +

              "&second=" +

              second,

            true

        );

        xhr.**onload** = function () {

**alert**("Time set successfully: " + xhr.responseText);

        };

        xhr.**onerror** = function(){ *//on error function, cause those errors are only detectable through these methods*

            console.**warn**("A error happened, duo to: " + xhr.status);

        };

        if (xhr.status == 200){ *//200 meaning its okay and successfull*

            xhr.**send**();

        }

    }

    catch(err){ *//this catch wont necessary happen, because http request are kinda weird*

**alert**("It occurs a errors when sending the httprequest, please make sure, you are connected to your device!\n" + err);

    };

}

##### time\_setting

function **time\_setting**(){

*//reading zone ... getting the element and reading the value, which currently used e.g. sellected*

    const zone = document.**getElementById**("box\_time").value;

    let lapse = 0; *//whole line portions for the x axe, e.g. a day has 24 --> 24 hours, meaning 24 lines*

    let unit = 0;  *//for each time lapse*

*//here i will draw a xy Graph with legends*

    const canvas = document.**getElementById**('graph');

    const ctx = canvas.**getContext**("2d"); *//getting the the features of the canvas, so i can promptly edit it AKA the context ... ctx*

*//getting the height and width*

    const width = canvas.width;

    const height = canvas.height;

    const x\_axe\_name\_diff = width \* 0.005; *//dif ... difference*

    const height\_for\_x\_lapses = height\*0.01;

    switch (zone) { *//changes --> before it was x\_axe\_lenght - 1 in clearRect etc. --> But i found out that it is too much trouble, to implement it in that way all the time. So i gave the time and sensor axes each unique length, so that i have no trouble with line conflictions*

        case "Today":

            lapse = 24; *//1, because of 0 time*

            unit = x\_point\_length / lapse;

*//clearRect(...,...,unit/2 + x\_axe\_length-1,...) --> +unit/2, because of the x-axe line shift, thats why, okay?*

            ctx.**clearRect**(-unit/2, 1, unit/2 + x\_axe\_length-1, height\_for\_x\_lapses + 0.02\*height + 12); *//clearing the axe, so it is cleared*

*//we are beginning at 1, because line with*

**measure\_text\_x\_axe\_and\_delete**(); *//deleting the text*

            for (let i = 0; i <= lapse; i++) {

                ctx.**beginPath**();

                ctx.**moveTo**(i \* unit, 0);

                ctx.**lineTo**(i\*unit, height\_for\_x\_lapses);

                ctx.lineWidth = 2; *//this one*

                ctx.strokeStyle = '#000000';

                ctx.**stroke**();

*//lapses naming*

                ctx.font = "12px serif";

                if (i < 10){

                    ctx.**fillText**( "0" + [i] + ":00", [-unit/3 + i\*unit], height\_for\_x\_lapses + 0.02\*height);

                }

                else{

                    ctx.**fillText**( [i] + ":00", [-unit/3 + i\*unit], height\_for\_x\_lapses + 0.02\*height);

                }

            }

            ctx.font = "20px serif";

            ctx.**fillText**("x in hours", x\_whole\_axe\_length + x\_axe\_name\_diff,1);

        break;

        case "Yesterday":

            lapse = 24; *//1, because of 0 time*

            unit = x\_point\_length / lapse; *//for each time lapse*

            ctx.**clearRect**(-unit/2, 1, unit/2 + x\_axe\_length-1, height\_for\_x\_lapses + 0.02\*height + 12); *//clearing the axe, so it is cleared --> 12, is the font size of the text*

*//we are beginning at 1, because line with*

**measure\_text\_x\_axe\_and\_delete**(); *//deleting the text*

            for (let i = 0; i <= lapse; i++) {

                ctx.**beginPath**();

                ctx.**moveTo**(i \* unit, 0);

                ctx.**lineTo**(i\*unit, height\_for\_x\_lapses);

                ctx.lineWidth = 2;

                ctx.strokeStyle = '#000000';

                ctx.**stroke**();

*//lapses naming*

                ctx.font = "12px serif";

                if (i < 10){

                    ctx.**fillText**( "0" + [i] + ":00", [-unit/3 + i\*unit], height\_for\_x\_lapses + 0.02\*height);

                }

                else{

                    ctx.**fillText**( [i] + ":00", [-unit/3 + i\*unit], height\_for\_x\_lapses + 0.02\*height);

                }

            }

            ctx.font = "20px serif";

            ctx.**fillText**("x in hours", x\_whole\_axe\_length + x\_axe\_name\_diff,1);

        break;

        case "This Week":

            lapse = 7; *//1, because of 0 time*

            unit = x\_point\_length / lapse; *//for each time lapse*

            ctx.**clearRect**(-unit/2, 1, unit/2 + x\_axe\_length-1, height\_for\_x\_lapses + 0.02\*height + 12); *//clearing the axe, so it is cleared*

*//we are beginning at 1, because line with*

**measure\_text\_x\_axe\_and\_delete**(); *//deleting the text*

            for (let i = 0; i <= lapse; i++) {

                ctx.**beginPath**();

                ctx.**moveTo**(i \* unit, 0);

                ctx.**lineTo**(i\*unit, height\_for\_x\_lapses);

                ctx.lineWidth = 2;

                ctx.strokeStyle = '#000000';

                ctx.**stroke**();

*//lapses naming*

                ctx.font = "12px serif";

                ctx.**fillText**([i], [i\*unit - unit/45], height\_for\_x\_lapses + 0.02\*height);

            }

            ctx.font = "20px serif";

            ctx.**fillText**("x in days", x\_whole\_axe\_length + x\_axe\_name\_diff,1);

        break;

        case "Last Week":

            lapse = 7; *//1, because of 0 time*

            unit = x\_point\_length / lapse; *//for each time lapse*

            ctx.**clearRect**(-unit/2, 1, unit/2 + x\_axe\_length-1, height\_for\_x\_lapses + 0.02\*height + 12); *//clearing the axe, so it is cleared*

*//we are beginning at 1, because line with*

**measure\_text\_x\_axe\_and\_delete**(); *//deleting the text*

            for (let i = 0; i <= lapse; i++) {

                ctx.**beginPath**();

                ctx.**moveTo**(i \* unit, 0);

                ctx.**lineTo**(i\*unit, height\_for\_x\_lapses);

                ctx.lineWidth = 2;

                ctx.strokeStyle = '#000000';

                ctx.**stroke**();

*//lapses naming*

                ctx.font = "12px serif";

                ctx.**fillText**([i], [i\*unit - unit/45], height\_for\_x\_lapses + 0.02\*height);

            }

            ctx.font = "20px serif";

            ctx.**fillText**("x in days", x\_whole\_axe\_length + x\_axe\_name\_diff,1);

        break;

        case "This Month":

            lapse = 30; *//1, because of 0 time*

            unit = x\_point\_length / lapse; *//for each time lapse*

            ctx.**clearRect**(-unit/2, 1, unit/2 + x\_axe\_length-1, height\_for\_x\_lapses + 0.02\*height + 12); *//clearing the axe, so it is cleared*

*//we are beginning at 1, because line with*

**measure\_text\_x\_axe\_and\_delete**(); *//deleting the text*

            for (let i = 0; i <= lapse; i++) {

                ctx.**beginPath**();

                ctx.**moveTo**(i \* unit, 0);

                ctx.**lineTo**(i\*unit, height\_for\_x\_lapses);

                ctx.lineWidth = 2;

                ctx.strokeStyle = '#000000';

                ctx.**stroke**();

*//lapses naming*

                ctx.font = "12px serif";

                ctx.**fillText**([i], [i\*unit - unit/10], height\_for\_x\_lapses + 0.02\*height);

            }

            ctx.font = "20px serif";

            ctx.**fillText**("x in days", x\_whole\_axe\_length + x\_axe\_name\_diff,1);

        break;

        case "Last Month":

            lapse = 30; *//1, because of 0 time*

            unit = x\_point\_length / lapse; *//for each time lapse*

            ctx.**clearRect**(-unit/2, 1, unit/2 + x\_axe\_length-1, height\_for\_x\_lapses + 0.02\*height + 12); *//clearing the axe, so it is cleared*

*//we are beginning at 1, because line with*

**measure\_text\_x\_axe\_and\_delete**(); *//deleting the text*

            for (let i = 0; i <= lapse; i++) {

                ctx.**beginPath**();

                ctx.**moveTo**(i \* unit, 0);

                ctx.**lineTo**(i\*unit, height\_for\_x\_lapses);

                ctx.lineWidth = 2;

                ctx.strokeStyle = '#000000';

                ctx.**stroke**();

*//lapses naming*

                ctx.font = "12px serif";

                ctx.**fillText**([i], [i\*unit - unit/10], height\_for\_x\_lapses + 0.02\*height);

            }

            ctx.font = "20px serif";

            ctx.**fillText**("x in days", x\_whole\_axe\_length + x\_axe\_name\_diff,1);

        break;

        case "This Year":

            lapse = 12; *//--> it was 365, i changed it to 12, i want to display the months, the idea came from a friend, i found it quite genious*

            unit = x\_point\_length / lapse; *//for each time lapse*

            ctx.**clearRect**(-unit/2 , 1, unit/2 + x\_axe\_length-1, height\_for\_x\_lapses + 0.02\*height + 12); *//clearing the axe, so it is cleared*

*//we are beginning at 1, because line with*

**measure\_text\_x\_axe\_and\_delete**(); *//deleting the text*

            for (let i = 0; i <= lapse; i++) {

                ctx.**beginPath**();

                ctx.**moveTo**(i \* unit, 0);

                ctx.**lineTo**(i\*unit, height\_for\_x\_lapses);

                ctx.lineWidth = 2;

                ctx.strokeStyle = '#000000';

                ctx.**stroke**();

*//lapses naming*

                ctx.font = "12px serif";

                ctx.**fillText**([i], [i\*unit - unit/22], height\_for\_x\_lapses + 0.02\*height);

            }

            ctx.font = "20px serif";

            ctx.**fillText**("x in months", x\_whole\_axe\_length + x\_axe\_name\_diff,1);

        break;

        case "Last Year":

            lapse = 12; *//1, because of 0 time --> etc. the same works here*

            unit = x\_point\_length / lapse; *//for each time lapse*

            ctx.**clearRect**(-unit/2, 1, unit/2 + x\_axe\_length-1, height\_for\_x\_lapses + 0.02\*height + 12); *//clearing the axe, so it is cleared*

*//we are beginning at 1, because line with*

**measure\_text\_x\_axe\_and\_delete**(); *//deleting the text*

            for (let i = 0; i <= lapse; i++) {

                ctx.**beginPath**();

                ctx.**moveTo**(i \* unit, 0);

                ctx.**lineTo**(i\*unit, height\_for\_x\_lapses);

                ctx.lineWidth = 2;

                ctx.strokeStyle = '#000000';

                ctx.**stroke**();

*//lapses naming*

                ctx.font = "12px serif";

                ctx.**fillText**([i], [i\*unit - unit/22], height\_for\_x\_lapses + 0.02\*height);

            }

            ctx.font = "20px serif";

            ctx.**fillText**("x in months", x\_whole\_axe\_length + x\_axe\_name\_diff,1);

        break;

    }

}

##### sensor\_setting

function **sensor\_setting**(){

*//the same like setting\_time()*

    const sensor = document.**getElementById**("box\_sens").value;

    let lapse = 15;

    let unit = 0;  *//for each time lapse*

*//here i will draw a xy Graph with legends*

    const canvas = document.**getElementById**('graph');

    const ctx = canvas.**getContext**("2d"); *//getting the the features of the canvas, so i can promptly edit it AKA the context ... ctx*

*//getting the height and width*

    const width = canvas.width;

    const height = canvas.height;

    const y\_axe\_name\_diff = width \* 0.005; *//diff ... difference*

    switch (sensor) { *//the scaling depends on the sensors*

        case "Temperature":

            ctx.**clearRect**(-1, 2, -width\*0.1, -y\_axe\_length-1); *// clearrect(..,2,..,...), because if do less than that, we well we will delete the arrow. Why, you ask, very simple cause this a self drawn graph not like any other you find online*

**measure\_text\_y\_axe\_and\_delete**(); *//deleting the text aka arrow names, so conflicts wont happen*

*//-75°C...250°C*

            lapse = 10;

            unit = y\_point\_length / lapse; *//for each time lapse*

            for (let i = 0; i <= lapse; i++) {

*//line generating*

                ctx.**beginPath**();

                ctx.**moveTo**(0, -i \* unit);

                ctx.**lineTo**(-width\*0.01, -i\*unit); *// -i\*unit --> because we have a translate point, meaning everything under the translate point is positive and everything above it, is negative*

                ctx.lineWidth = 2;

                ctx.strokeStyle = '#000000';

                ctx.**stroke**();

*//giving numbers (names) to the lines with corosponding --> -75°C...250°C --> our complete range are 325°C --> meanig per lapse we have --> 32.5 degree difference*

                ctx.font = "20px serif";

                const unit\_temp = 325/lapse;

                ctx.**fillText**([-75+(i\*unit\_temp)] + "°C", -width\*0.06, -i\*unit);

            }

            ctx.font = "20px serif";

            ctx.**fillText**("Temperature", -1, -[y\_whole\_axe\_length + y\_axe\_name\_diff]); *//arrow naming*

        break;

        case "Airmoisure":

            ctx.**clearRect**(-1, 2, -width\*0.1, -y\_axe\_length-1); *//clearing the axe, so it is cleared*

**measure\_text\_y\_axe\_and\_delete**(); *//deleting the text aka arrow names, so conflicts wont happen*

*//0...100% RH*

            lapse = 10;

            unit = y\_point\_length / lapse; *//for each time lapse*

            for (let i = 0; i <= lapse; i++) {

*//line generating*

                ctx.**beginPath**();

                ctx.**moveTo**(0, -i \* unit);

                ctx.**lineTo**(-width\*0.01, -i\*unit); *// -i\*unit --> because we have a translate point, meaning everything under the translate point is positive and everything above it, is negative*

                ctx.lineWidth = 2;

                ctx.strokeStyle = '#000000';

                ctx.**stroke**();

*//giving numbers (names) to the lines with corosponding --> -75°C...250°C --> our complete range are 325°C --> meanig per lapse we have --> 32.5 degree difference*

                ctx.font = "20px serif";

                const unit\_temp = 100/lapse;

                ctx.**fillText**([(i\*unit\_temp)] + "%", -width\*0.06, -i\*unit);

            }

            ctx.font = "20px serif";

            ctx.**fillText**("Airmoisure", -1, -[y\_whole\_axe\_length + y\_axe\_name\_diff]); *//arrow naming*

        break;

        case "Airpressure":

            ctx.**clearRect**(-1, 2, -width\*0.1, -y\_axe\_length-1); *//clearing the axe, so it is cleared*

**measure\_text\_y\_axe\_and\_delete**(); *//deleting the text aka arrow names, so conflicts wont happen*

*//300hPa...1100hPa*

            lapse = 10;

            unit = y\_point\_length / lapse; *//for each time lapse*

            for (let i = 0; i <= lapse; i++) {

*//line generating*

                ctx.**beginPath**();

                ctx.**moveTo**(0, -i \* unit);

                ctx.**lineTo**(-width\*0.01, -i\*unit); *// -i\*unit --> because we have a translate point, meaning everything under the translate point is positive and everything above it, is negative*

                ctx.lineWidth = 2;

                ctx.strokeStyle = '#000000';

                ctx.**stroke**();

*//giving numbers (names) to the lines with corosponding --> -75°C...250°C --> our complete range are 325°C --> meanig per lapse we have --> 32.5 degree difference*

                ctx.font = "20px serif";

                const unit\_temp = 800/lapse; *//1100hPa - 300hPa = 800hPa*

                ctx.**fillText**([300 + (i\*unit\_temp)] + "hPa", -width\*0.06, -i\*unit);

            }

            ctx.font = "20px serif";

            ctx.**fillText**("Airpressure", -1, -[y\_whole\_axe\_length + y\_axe\_name\_diff]); *//arrow naming*

        break;

        case "Gas":

*//i dont know we have to test it first*

        break;

    }

}

##### measure\_text\_x\_axe\_and\_delete

function **measure\_text\_x\_axe\_and\_delete**(){ *//for x-axe arrow naming*

    const canvas = document.**getElementById**('graph');

    const ctx = canvas.**getContext**("2d"); *//getting the the features of the canvas, so i can promptly edit it AKA the context ... ctx*

*//getting the width*

    const width = canvas.width;

    const x\_axe\_name\_diff = width \* 0.005; *//dif ... difference*

    ctx.lineWidth = 0.5;

    ctx.strokeStyle = '#000000'; *//if strokestyle was not set, it will now be set in the occupied function*

*//getting the measurments - the naming of the x-axe will be measured here*

    var x\_months\_length = ctx.**measureText**("x in months").width;

    var x\_days\_length = ctx.**measureText**("x in days").width;

    var x\_hours\_length = ctx.**measureText**("x in hours").width;

*//now deleting*

    ctx.**clearRect**(x\_whole\_axe\_length + x\_axe\_name\_diff, 1, x\_days\_length, -20); *//20px, we have to do "ctx.font = 20px serif", because it's allignment is up .... (...,1,...,...) because of the line width*

    ctx.**clearRect**(x\_whole\_axe\_length + x\_axe\_name\_diff, 1, x\_days\_length, x\_axe\_name\_diff); *//double insurance, sometimes it deletes almost everything, but not all the text --> x\_axe\_name\_diff, because it's deletes only the neccessary lines*

*//x\_hours*

    ctx.**clearRect**(x\_whole\_axe\_length + x\_axe\_name\_diff, 1, x\_hours\_length, -20);

    ctx.**clearRect**(x\_whole\_axe\_length + x\_axe\_name\_diff, 1, x\_hours\_length, x\_axe\_name\_diff); *//same here*

*//x\_months*

    ctx.**clearRect**(x\_whole\_axe\_length + x\_axe\_name\_diff, 1, x\_months\_length, -20);

    ctx.**clearRect**(x\_whole\_axe\_length + x\_axe\_name\_diff, 1, x\_months\_length, x\_axe\_name\_diff);

}

##### measure\_text\_y\_axe\_and\_delete

function **measure\_text\_y\_axe\_and\_delete**(){ *//for y-axe arrow naming*

    const canvas = document.**getElementById**('graph');

    const ctx = canvas.**getContext**("2d"); *//getting the the features of the canvas, so i can promptly edit it AKA the context ... ctx*

*//getting the width*

    const width = canvas.width;

    const y\_axe\_name\_diff = width \* 0.005; *//diff ... difference*

    ctx.lineWidth = 0.5;

    ctx.strokeStyle = '#000000'; *//if strokestyle was not set, it will now be set in the occupied function*

*//getting the measurments - the naming of the x-axe will be measured here*

    var y\_temp\_length = ctx.**measureText**("Temperature").width;

    var y\_airmois\_length = ctx.**measureText**("Airmoisure").width;

    var y\_airpress\_length = ctx.**measureText**("Airpressure").width;

*//now deleting*

*//temperature*

    ctx.**clearRect**(-1, -[y\_whole\_axe\_length + y\_axe\_name\_diff], y\_temp\_length, -20); *//20px, we have to do "ctx.font = 20px serif", because it's allignment is up .... (...,1,...,...) because of the line width*

    ctx.**clearRect**(-1, -[y\_whole\_axe\_length + y\_axe\_name\_diff], y\_temp\_length, y\_axe\_name\_diff); *//double insurance, sometimes it deletes almost everything, but not all the text*

*//airmoisure*

    ctx.**clearRect**(-1, -[y\_whole\_axe\_length + y\_axe\_name\_diff], y\_airmois\_length, -20);

    ctx.**clearRect**(-1, -[y\_whole\_axe\_length + y\_axe\_name\_diff], y\_airmois\_length, y\_axe\_name\_diff); *//same here*

*//airpressure*

    ctx.**clearRect**(-1, -[y\_whole\_axe\_length + y\_axe\_name\_diff], y\_airpress\_length, -20);

    ctx.**clearRect**(-1, -[y\_whole\_axe\_length + y\_axe\_name\_diff], y\_airpress\_length, y\_axe\_name\_diff);

}

##### Leere Funktionen

function **apply\_button**(){ *//applying the filter options on the graph*

}

function **update**(){

}

function **HTTP\_SET**(){ *//here i will set and save the values for the sensor data*

*//bme should also measure temperature*

*//ozon in ppm, does it change*

}

#### CSS

#### Endergebnis

A white rectangular object with black text

Description automatically generated

# Hardware

## Allgemein

## Version 1

## Version 2

## Version 3

Hier: