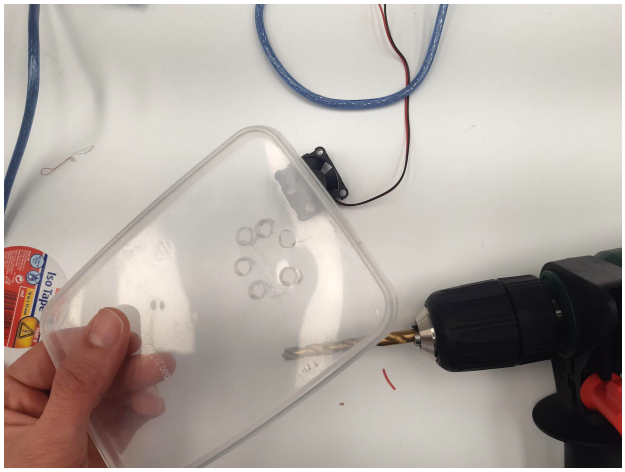
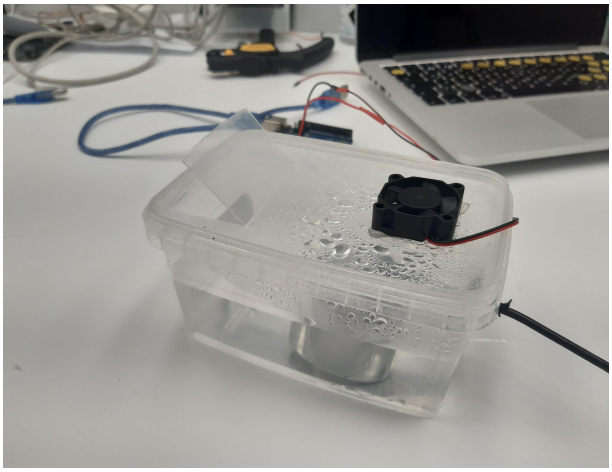
Part 2

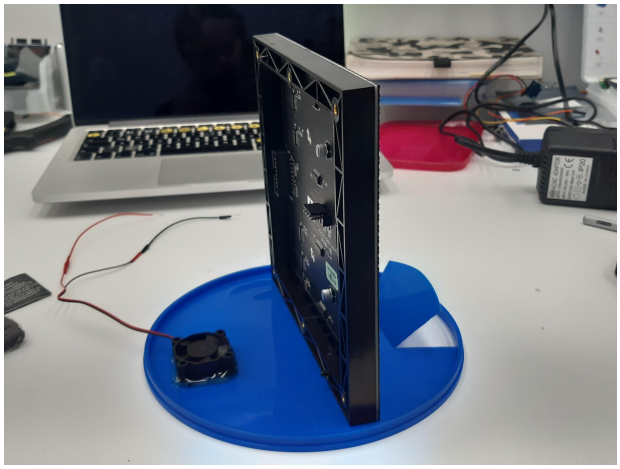
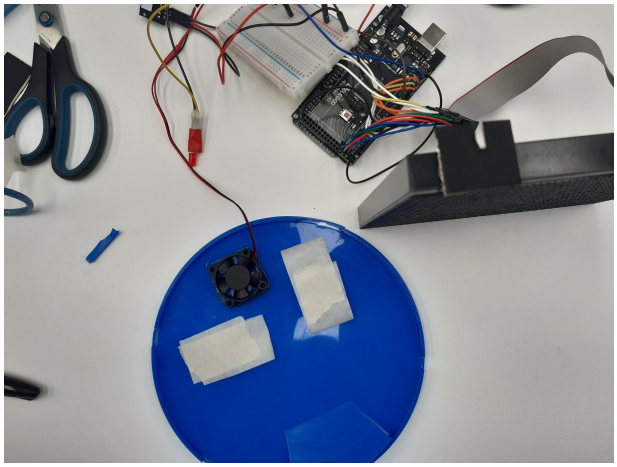
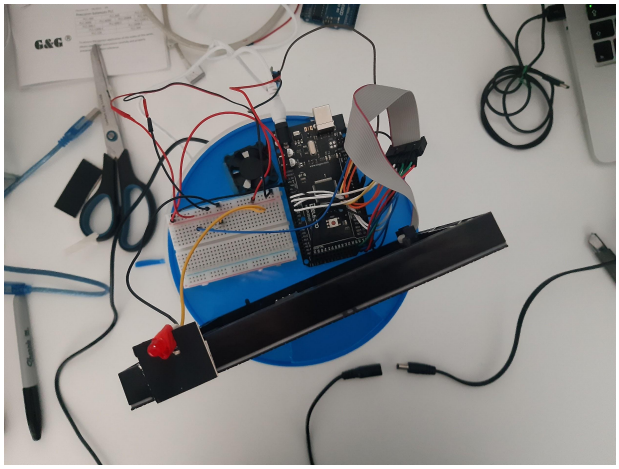
Ultrasonic fogger/atomizer with 12 LEDs

Using this ready-made atomiser seemed simple enough - plus had build-in added lights, because, why not?! It worked on a slightly different (less impressive) principle, but it basically saved the whole project at this point.

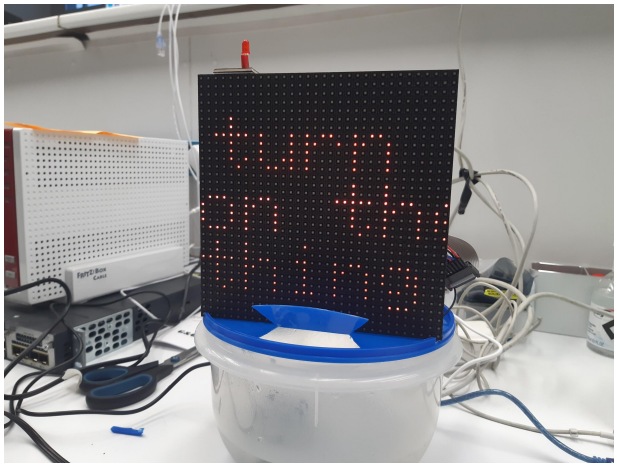
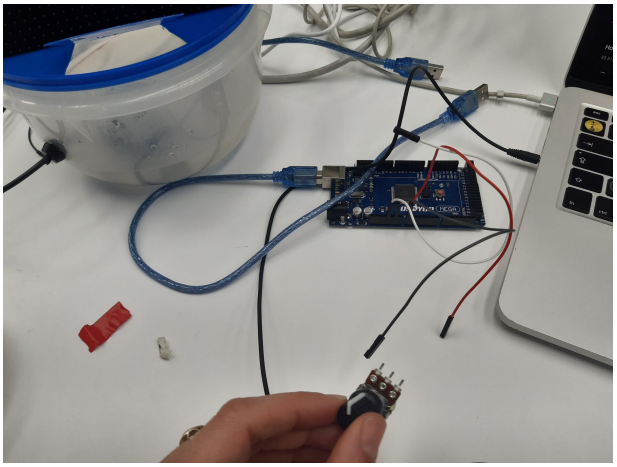
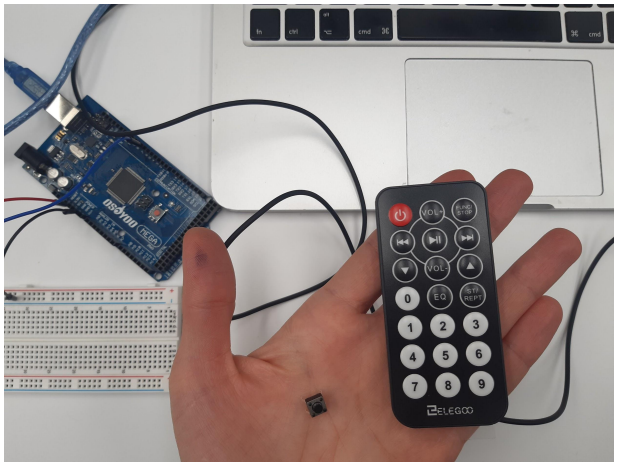
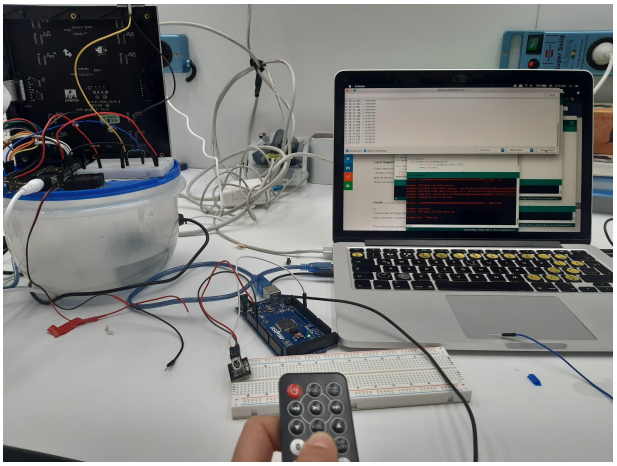
Figuring out how to minimise spillage was harder than I thought though. It seemed like nothing really changes depending on the water level, so I was just hoping that my actual water tank is wide and tall enough to separate the droplets from the fan above them.

I think I need a bigger water tank...

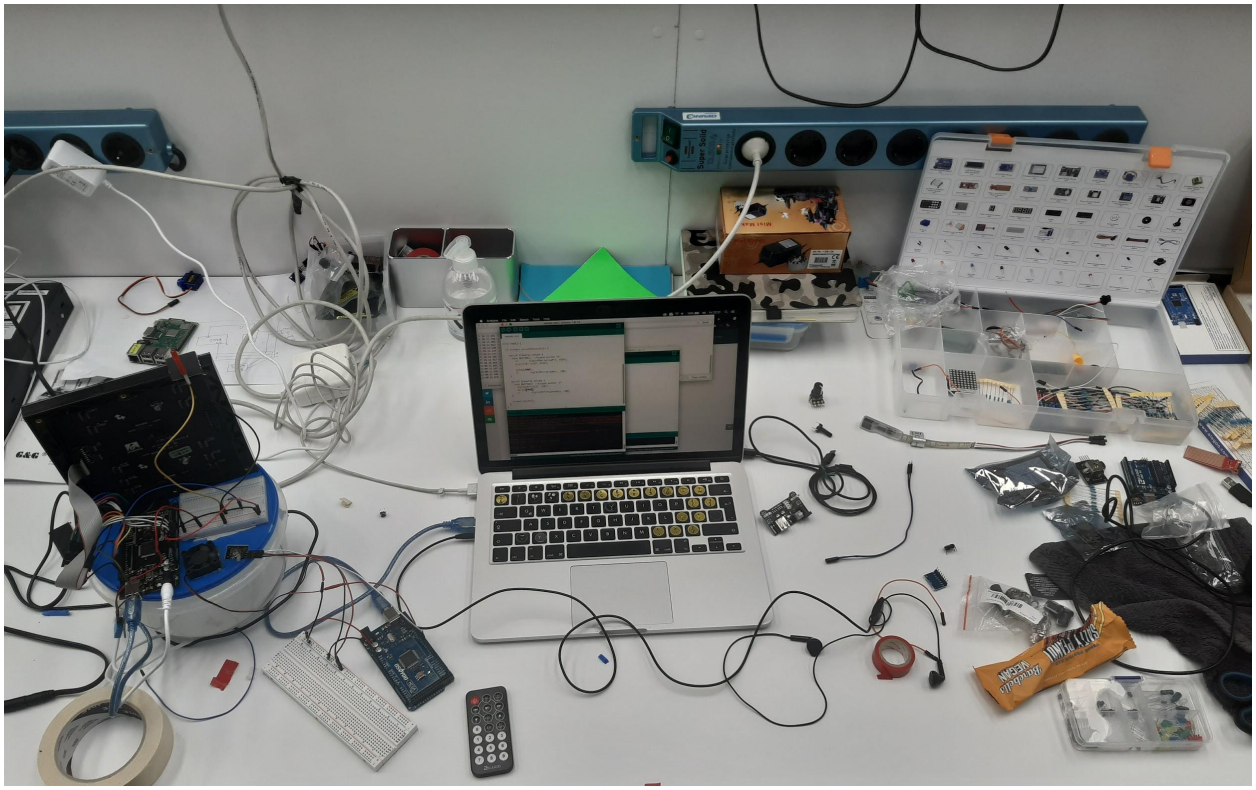





Have you tried turning it off and on again?

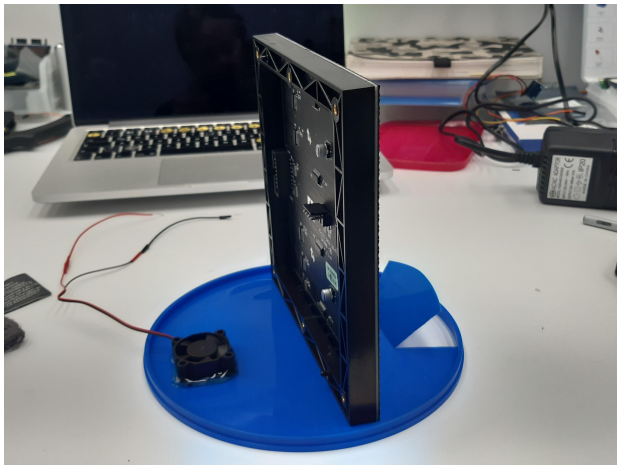
Potentiometer, remote control or just a push button?

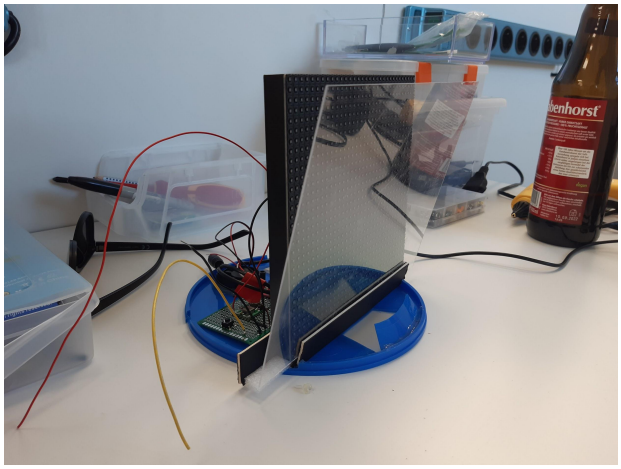
So my initial plan was just to use a simple push button. But where do I place it? The quickest was just simply on the breadboard, and it was fine for testing purposes, but to bring it to the next level, I was thinking about trying out the remote control... And it worked! At first. Turning the fan on and off was working fine, until I included the remote control code into the main app. I struggled for a while until I figured, the Arduino just can not handle random input while running so many different parts in the same loop (basically, anytime the LED matrix was connected, too, the push of the remote would just give unpredictable values). I had to drop the idea. One of the few things that actually arrived from my order were a handful of potentiometers, as my guiding tutorial suggested, too. However, maybe because it was quite late that night already, I just could not get it working properly. It started to smoke more than once, so I decided to return to the push button version.



Water + Electricity?

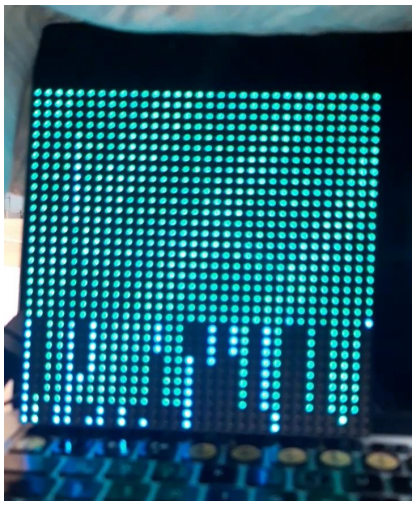
One of my biggest fears starting this project was the potentially dangerous combination of water and electricity. The least I could do is to turn this cheap sheet of plastic from an IKEA picture frame into a shield that protects the matrix. Especially since for enhanced visual effect of the outcoming mist, I had to invert the diagonal plastic part at the opening, to drive the mist more upwards than away.





Coding the waterfall

Coming this far, I was excited to start with some coding for the missing visual puzzle, the waterfall. It was rather far from what I had imagined, so a Freudian typo, “waterfail” seemed to be working title..

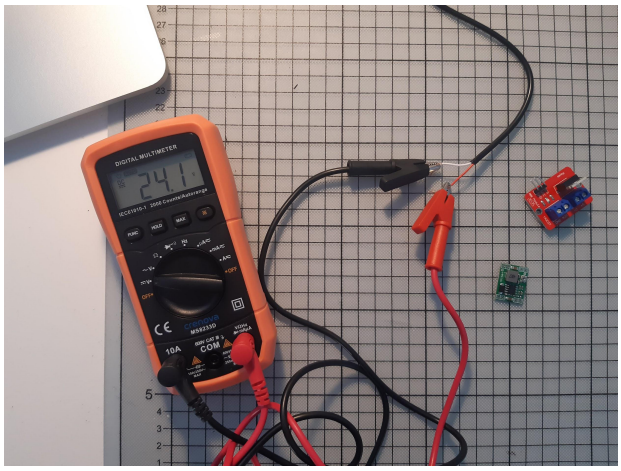


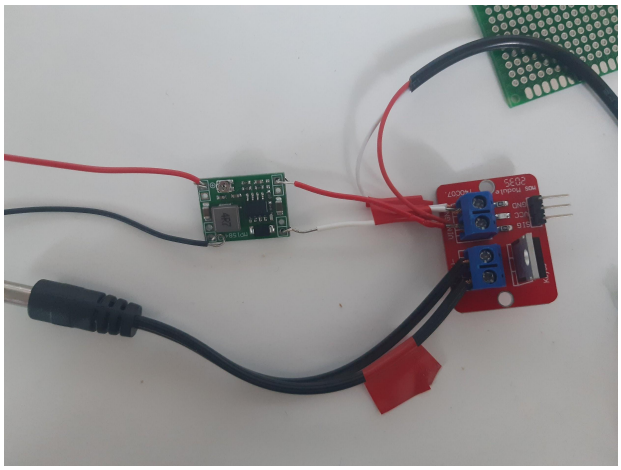
The (almost) last steps

The humidifier was coming together slowly, although some hardware compromises were made and the housing was far from what I had imagined, it was fully functioning at this point. I could have just focused on polishing the code - and possibly hiding all the parts and wiring, but then I received something in the mail. To be exact, I finally received all the missing parts that were needed to simplify the entire circuit.

Just one plug instead of two?!

This was by far the most complicated part, to figure out voltages without frying something again, but a day of careful wire cutting, measuring, adjusting and rewiring, I managed to build in the MOSFET Driver for enabling a switch pin on the Arduino for turning the fogger on and off and the Step-down Module connected to it for making 10V from the incoming 24V, as nothing else needed/could handle that high power (except the fan, but I figured by accident that it also runs on lower voltage, plus, it's much less loud than it would be on 24V or even 12V, as its specs suggested).



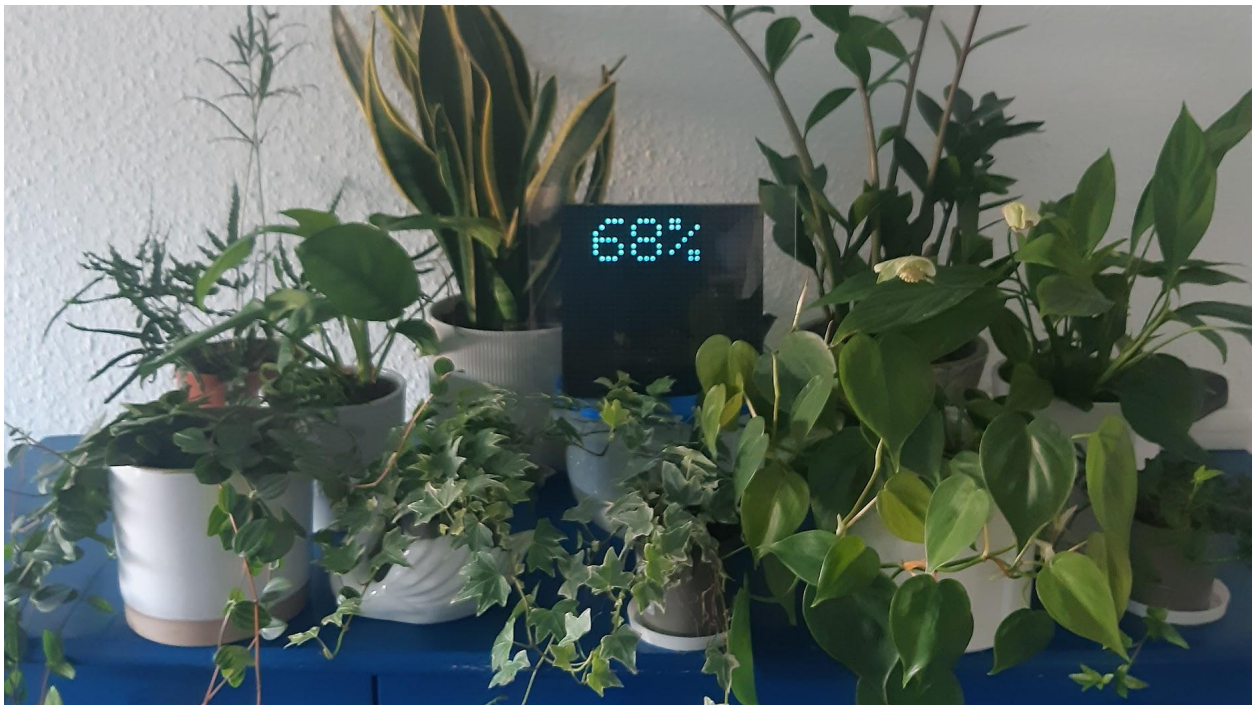


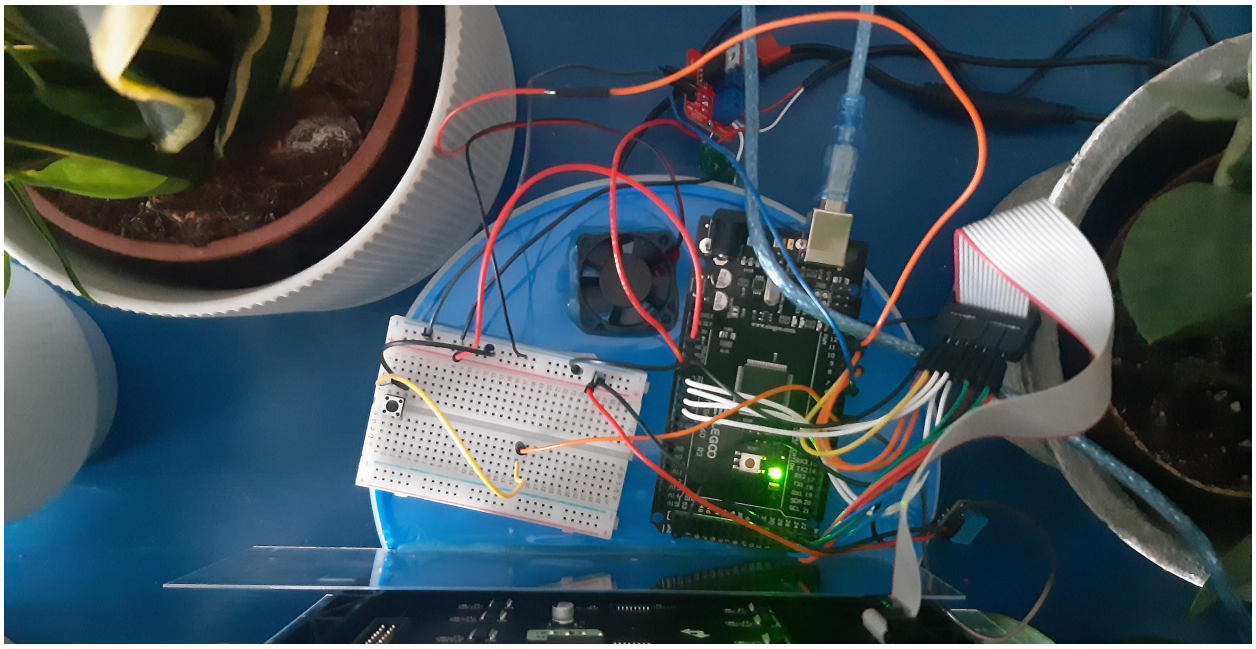
Ditch the LED

In the name of simplification, the indicator LED on the top had to go, too. Why complicate things, when all necessary information can be displayed on the matrix?

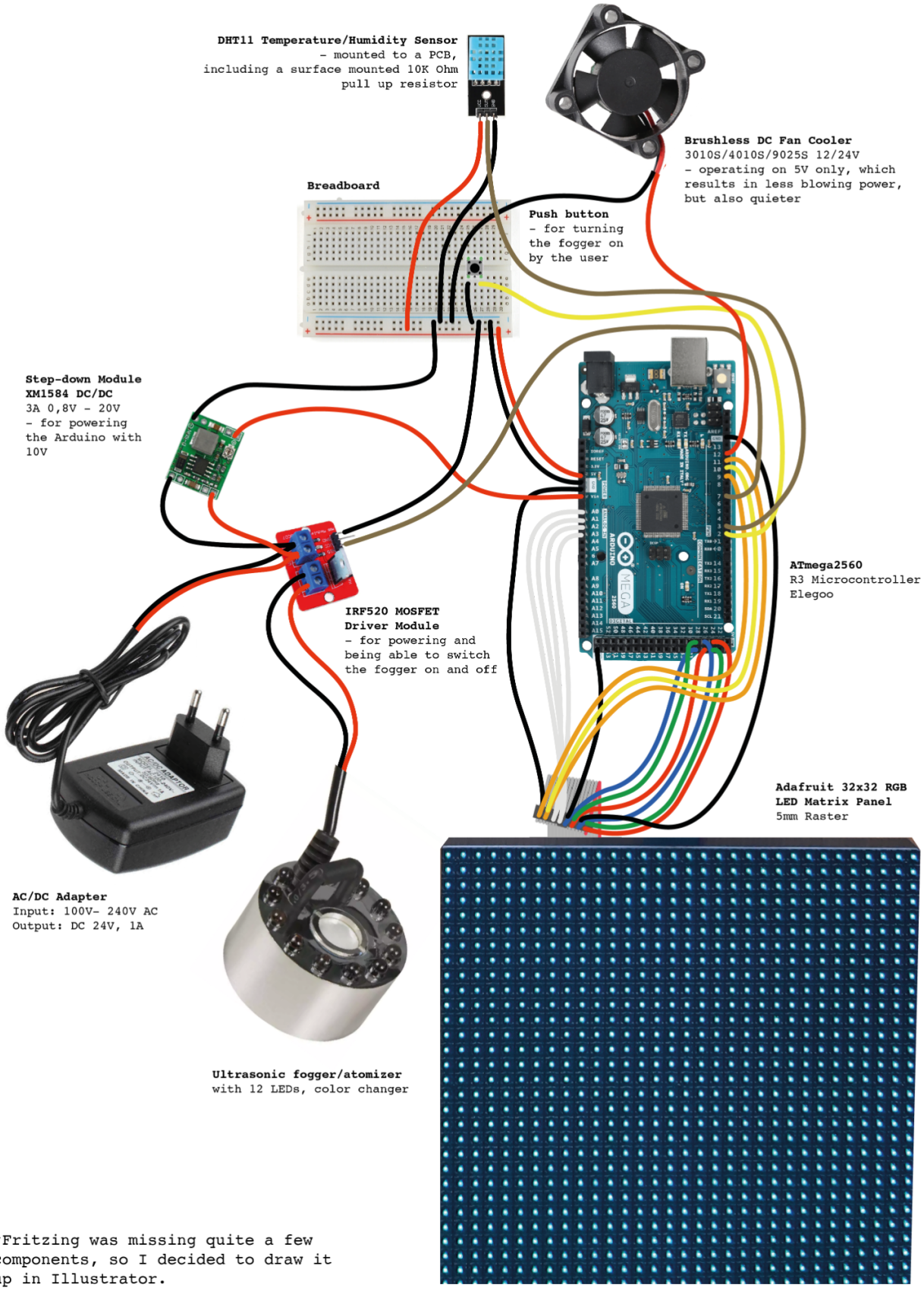
1. the final physical prototype

I placed the device into its envisioned environment. Of course there was a lot more iteration on the code needed, but it was finally looking ready!





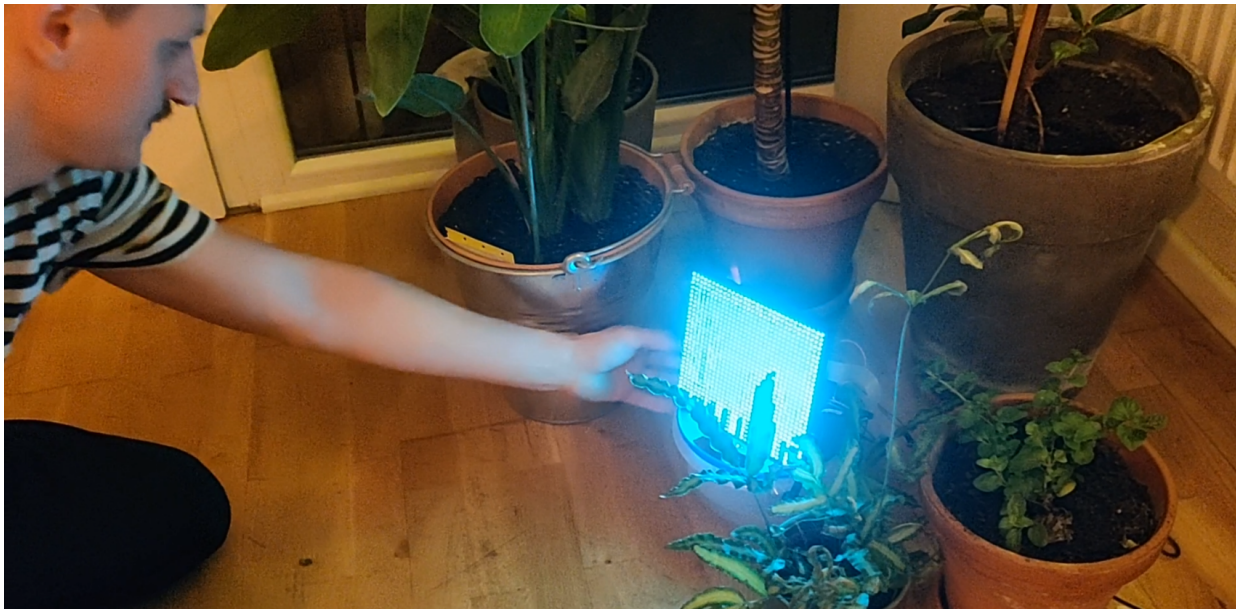
2. circuit plan*



*Fritzing was missing quite a few components, so I decided to draw it up in Illustrator.

3. user testing

Of course the first version of the code left many open cases un-handled, so in retrospect I did ask my friends a bit too early to test the interaction, but they still gave some, mostly positive feedback, especially about the speed of changing the different screens.



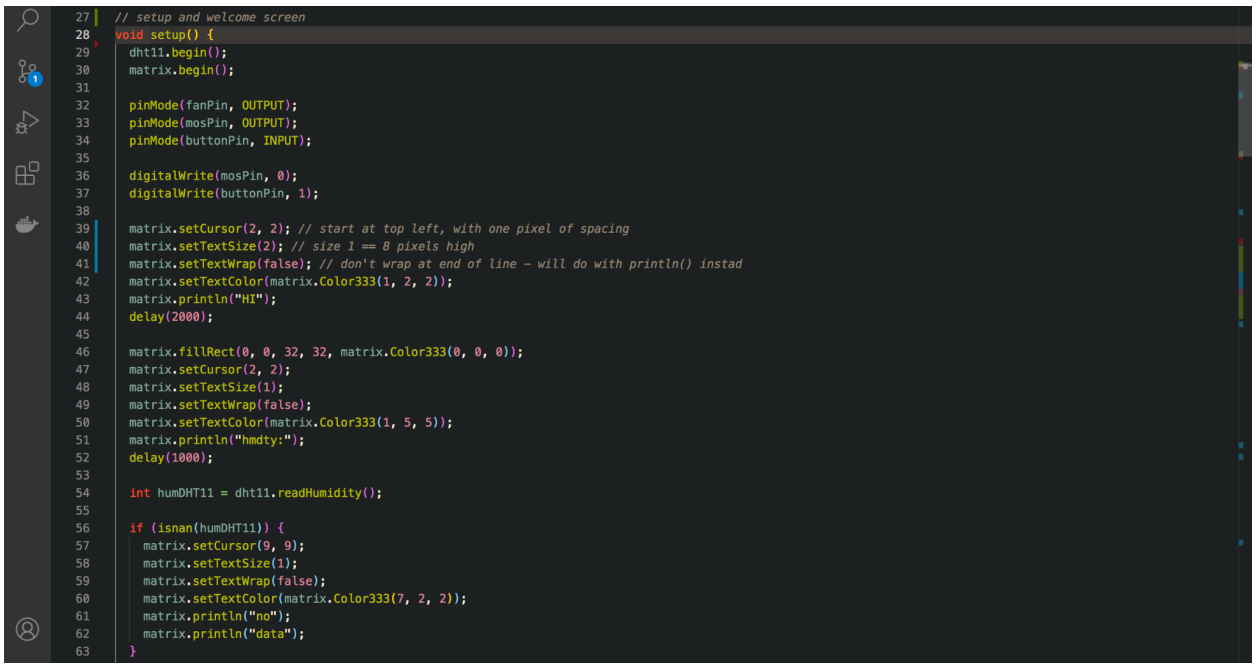
For the final logic of the screens and instructions, please watch this video.

4. source code

1. importing libraries, declaring pins and variables, defining set a value for optimal humidity

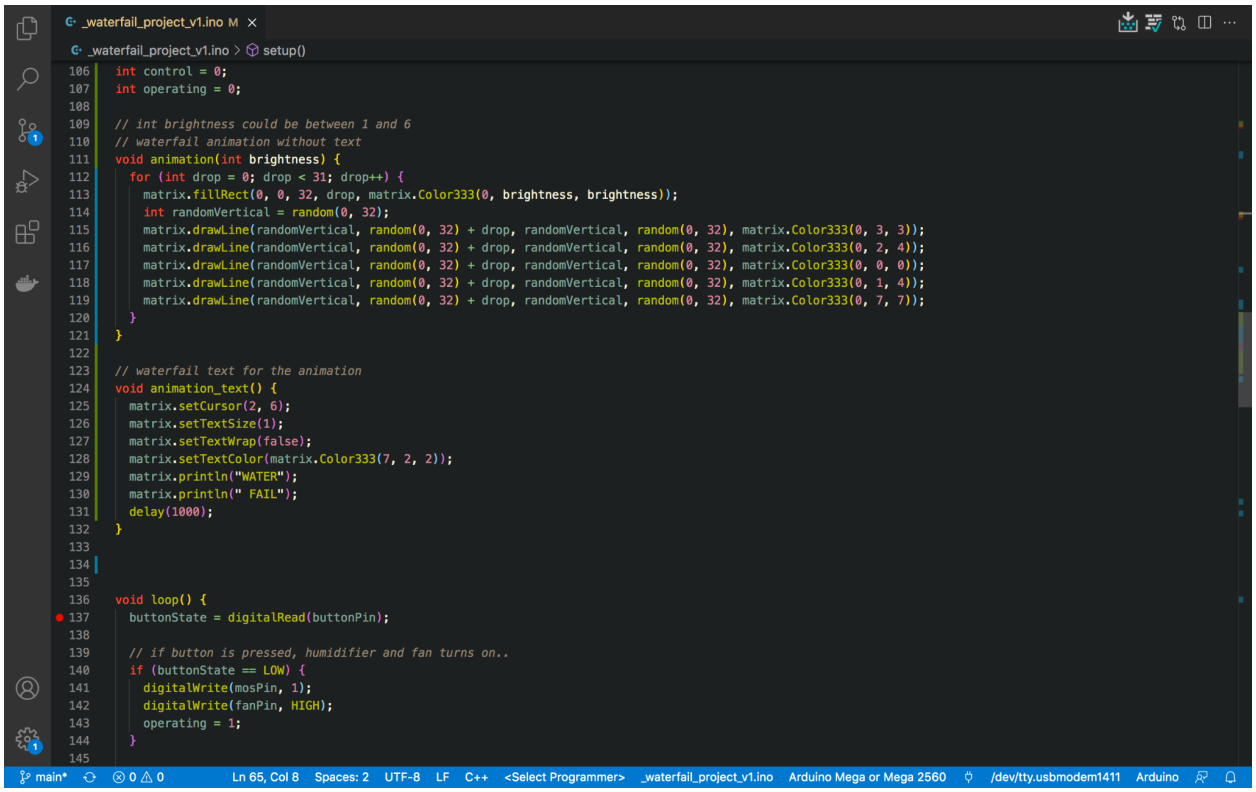
```
#include <RGBmatrixPanel.h>
#include <DHT.h>

#define CLK 11
#define OE   9
#define LAT 10
#define A   A0
#define B   A1
#define C   A2
#define D   A3

#define DHTPIN11 3
#define DHTTYPE11 DHT11

RGBmatrixPanel matrix(A, B, C, D, CLK, LAT, OE, false);
DHT dht11(DHTPIN11, DHTTYPE11);

const int fanPin = 12;
const int mosPin = 7;
const int buttonPin = 2;

int buttonState = 0;

const int optimalHumidity = 50;
const int bufferHumidity = optimalHumidity + 10;
```

2. in the `void setup()` pins are set, welcome screen appears and the initial humidity is measured



```
// setup and welcome screen
void setup() {
  dht11.begin();
  matrix.begin();

  pinMode(fanPin, OUTPUT);
  pinMode(mosPin, OUTPUT);
  pinMode(buttonPin, INPUT);

  digitalWrite(mosPin, 0);
  digitalWrite(buttonPin, 1);

  matrix.setCursor(2, 2); // start at top left, with one pixel of spacing
  matrix.setTextSize(2); // size 1 == 8 pixels high
  matrix.setTextWrap(false); // don't wrap at end of line - will do with println() instad
  matrix.setTextColor(matrix.Color333(1, 2, 2));
  matrix.println("HI");
  delay(2000);

  matrix.fillRect(0, 0, 32, 32, matrix.Color333(0, 0, 0));
  matrix.setCursor(2, 2);
  matrix.setTextSize(1);
  matrix.setTextWrap(false);
  matrix.setTextColor(matrix.Color333(1, 5, 5));
  matrix.println("hmdty:");
  delay(1000);

  int humDHT11 = dht11.readHumidity();

  if (isnan(humDHT11)) {
    matrix.setCursor(9, 9);
    matrix.setTextSize(1);
    matrix.setTextWrap(false);
    matrix.setTextColor(matrix.Color333(7, 2, 2));
    matrix.println("no");
    matrix.println("data");
  }
```

3. if it is too low, the user is asked to turn the fogger on, the waterfall animation runs - if it is optimal, it informs the user as well, no action required



```
int control = 0;
int operating = 0;

// int brightness could be between 1 and 6
// waterfail animation without text
void animation(int brightness) {
  for (int drop = 0; drop < 31; drop++) {
    matrix.fillRect(0, 0, 32, drop, matrix.Color333(0, brightness, brightness));
    int randomVertical = random(0, 32);
    matrix.drawLine(randomVertical, random(0, 32) + drop, randomVertical, random(0, 32), matrix.Color333(0, 3, 3));
    matrix.drawLine(randomVertical, random(0, 32) + drop, randomVertical, random(0, 32), matrix.Color333(0, 2, 4));
    matrix.drawLine(randomVertical, random(0, 32) + drop, randomVertical, random(0, 32), matrix.Color333(0, 0, 0));
    matrix.drawLine(randomVertical, random(0, 32) + drop, randomVertical, random(0, 32), matrix.Color333(0, 1, 4));
    matrix.drawLine(randomVertical, random(0, 32) + drop, randomVertical, random(0, 32), matrix.Color333(0, 7, 7));
  }
}

// waterfail text for the animation
void animation_text() {
  matrix.setCursor(2, 6);
  matrix.setTextSize(1);
  matrix.setTextWrap(false);
  matrix.setTextColor(matrix.Color333(7, 2, 2));
  matrix.println("WATER");
  matrix.println(" FAIL");
  delay(1000);
}



void loop() {
  buttonState = digitalRead(buttonPin);

  // if button is pressed, humidifier and fan turns on..
  if (buttonState == LOW) {
    digitalWrite(mosPin, 1);
    digitalWrite(fanPin, HIGH);
    operating = 1;
  }

```

4. in the `void loop()`, we start to check the humidity and temperature, and also if the button is pressed. According to different humidity readings, the code prompts the user to turn on the fogger, in case it is not yet turned on. Otherwise confirms that the levels are optimal.

```
void loop() {
  buttonState = digitalRead(buttonPin);

  // if button is pressed, humidifier and fan turns on..
  if (buttonState == LOW) {
    digitalWrite(mosPin, 1);
    digitalWrite(fanPin, HIGH);
    operating = 1;
  }

  int tempDHT11 = dht11.readTemperature();
  int humDHT11 = dht11.readHumidity();
  delay(500);

  // if humidity is below optimal
  if ( humDHT11 < optimalHumidity ) {
    control = 0;

    // when it is not running
    if (operating == 0) {
        matrix.setCursor(10, 2);
        matrix.setTextSize(1);
        matrix.setTextWrap(false);
        matrix.setTextColor(matrix.Color333(7, 2, 2));
        matrix.fillRect(0, 0, 32, 32, matrix.Color333(0, 0, 0));
        matrix.print(humDHT11);
        matrix.println("%");
        matrix.fillRect(0, 10, 32, 32, matrix.Color333(0, 0, 0));
        delay(1000);
        matrix.println("turn");
        matrix.println("it on!");
        matrix.println("<");
        delay(1000);

```

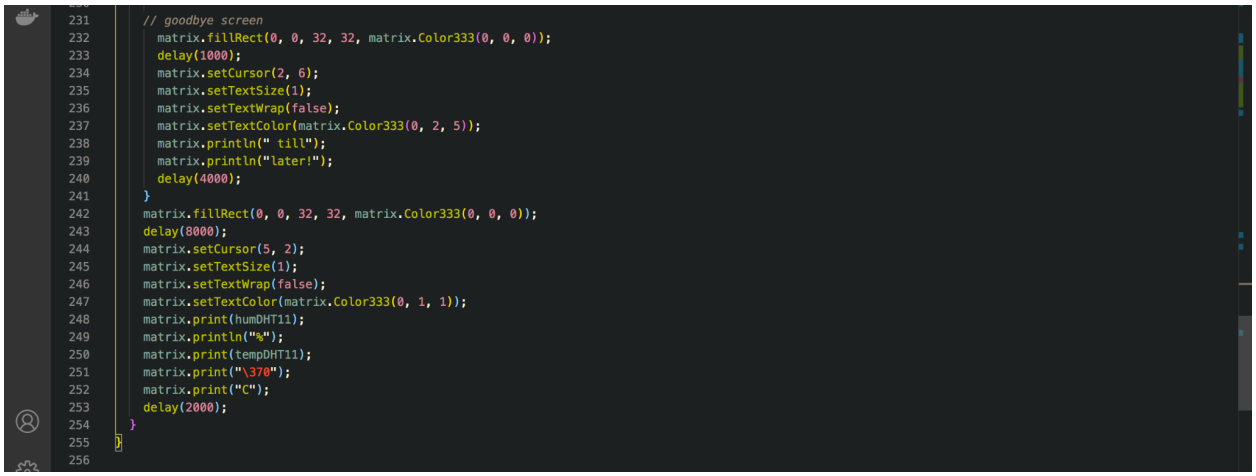
5. In case the fogger is already turned on, but not optimal yet, it keeps showing the user the rising values and confirms that it's "working on it". It does not turn off yet as soon as it reaches the optimal humidity, otherwise it would drop below optimal quickly again. Therefore a buffer value has to be reached for the fan and the fogger to automatically turn off.

```
  // when it is running
  } else if (operating == 1) {
      matrix.fillRect(0, 0, 32, 32, matrix.Color333(0, 0, 0));
      matrix.setCursor(5, 2);
      matrix.setTextSize(1);
      matrix.setTextWrap(false);
      matrix.setTextColor(matrix.Color333(7, 2, 2));
      matrix.print(humDHT11);
      matrix.println("%");
      delay(500);
      matrix.setTextColor(matrix.Color333(2, 1, 3));
      matrix.println("working");
      matrix.println("on it");
      delay(1000);

      animation(1);
      delay(1000);
  }
}

// if humidity is above optimal but below buffer -> keep showing current humidity
else if (bufferHumidity > humDHT11 && humDHT11 > optimalHumidity) {
  matrix.fillRect(0, 0, 32, 32, matrix.Color333(0, 0, 0));
  delay(3000);
  matrix.setCursor(5, 2);
  matrix.setTextSize(1);
  matrix.setTextWrap(false);
  matrix.setTextColor(matrix.Color333(0, 1, 1));
  matrix.print(humDHT11);
  matrix.println("%");
  matrix.print(tempDHT11);
  matrix.print("\370");
  matrix.print("C");
  delay(2000);
}
```

6. once the humidity is above the buffer value, the device informs the user that all is good, and turns off the fan and fogger

```
Search
// humdidity is above buffer, aka very optimal
else if (humDHT11 > bufferHumidity) {
  if (control < 1) {
    digitalWrite(mosPin, 0);
    operating = 0;

    matrix.fillRect(0, 0, 32, 32, matrix.Color333(0, 0, 1));
    matrix.setCursor(5, 2);
    matrix.setTextSize(1);
    matrix.setTextWrap(false);
    matrix.setTextColor(matrix.Color333(1, 7, 7));
    matrix.print(humDHT11);
    matrix.println("%");

    delay(1000);

    matrix.setCursor(2, 15);
    matrix.println("GOOD!");
    delay(3000);

    animation(1);
    delay(2000);
    digitalWrite(fanPin, LOW);
    control++;
```

7. a goodbye screen appears, and the loop continues to show the humidity and temperature without too much visual clutter, and with longer pauses. Once the humidity drops below optimal, it enters again at the top of the loop and asks the user to turn it on.



```
231     // goodbye screen
232       matrix.fillRect(0, 0, 32, 32, matrix.Color333(0, 0, 0));
233       delay(1000);
234       matrix.setCursor(2, 6);
235       matrix.setTextSize(1);
236       matrix.setTextWrap(false);
237       matrix.setTextColor(matrix.Color333(0, 2, 5));
238       matrix.println(" till");
239       matrix.println("later!");
240       delay(4000);
241     }
242     matrix.fillRect(0, 0, 32, 32, matrix.Color333(0, 0, 0));
243     delay(8000);
244     matrix.setCursor(5, 2);
245     matrix.setTextSize(1);
246     matrix.setTextWrap(false);
247     matrix.setTextColor(matrix.Color333(0, 1, 1));
248     matrix.print(humDHT11);
249     matrix.println("%");
250     matrix.print(tempDHT11);
251     matrix.print("\370");
252     matrix.print("C");
253     delay(2000);
254   }
255 }
256
```

For the complete code please check the repository on [github](#).

5. final thoughts

Without a doubt, this project was where I feel I learnt the most in the shortest amount of time. Neither design or coding is new to me, but hardware, electricity and prototyping kind of was. However the LUs were not directly helping my personal work, they pushed me and gave me the courage to jump into it, which was key at the beginning. I was surprised, motivated and sometimes devastated by the number of iterations needed, mostly because of incompatible hardware. I feel that even with those issues tackled, I still have a lot of room for improvement in this very project, and actually planning on refining and further simplifying it in the near future. First and foremost I wanted to build something usable, therefore my next steps will be rebuilding the hardware, replacing the matrix with an OLED screen and hopefully integrating the remote control. Then, a proper round of user testing will possibly become a greater part of the development process. This remains a work in progress, but I have already gained the confidence to tinker and experiment, which is invaluable to me.

Without a doubt, my favorite project so far.