Minor Smart Things

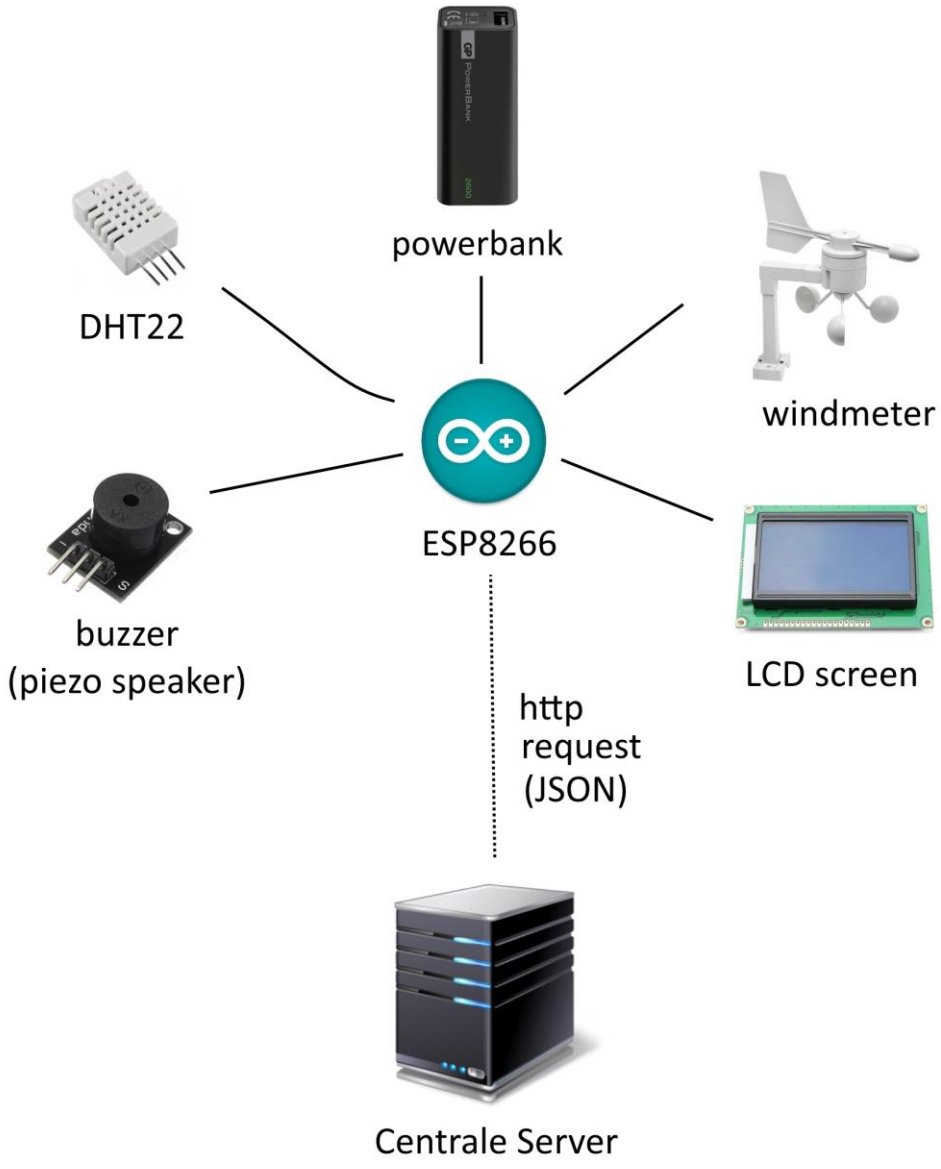
Documentatie Weerstation

Versie 1.0
Datum laatst gewijzigd: 16-11-2018

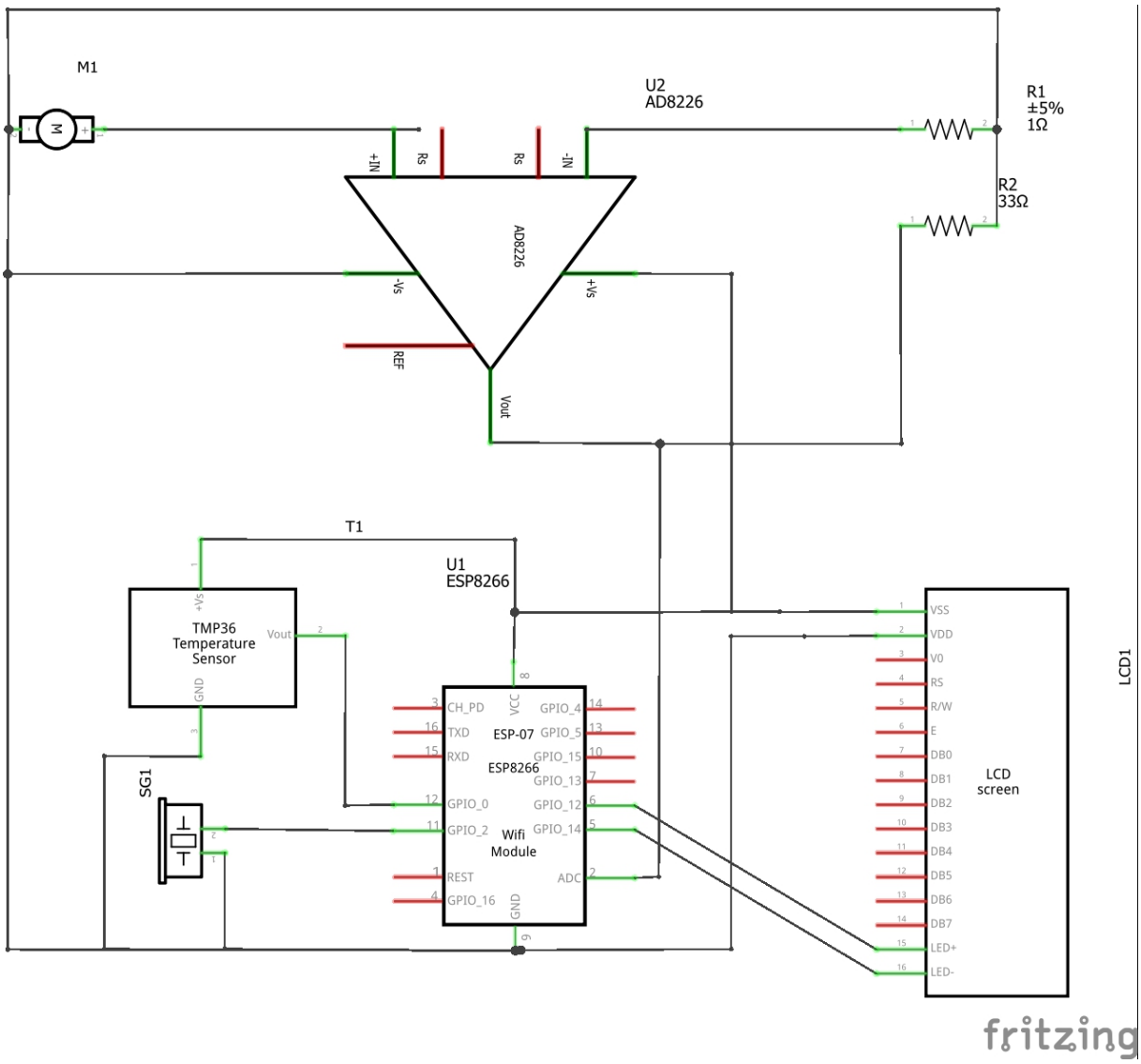
Oguzhan Cihan
0911315

Inhoudsopgave

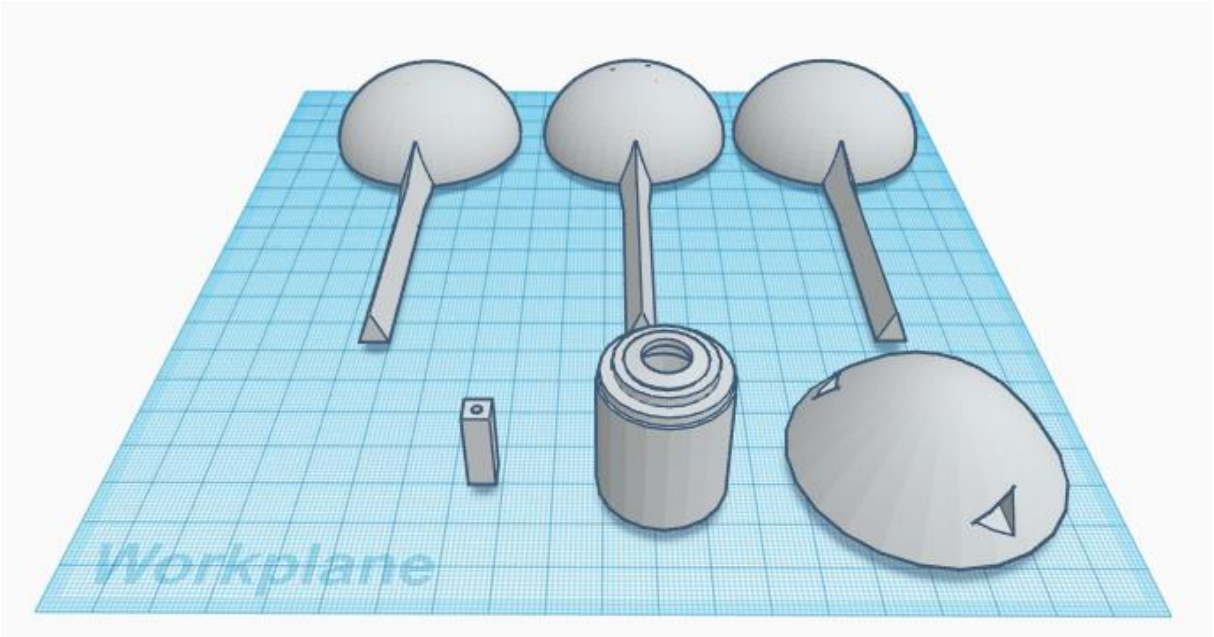
1 – Architectuurdiagram

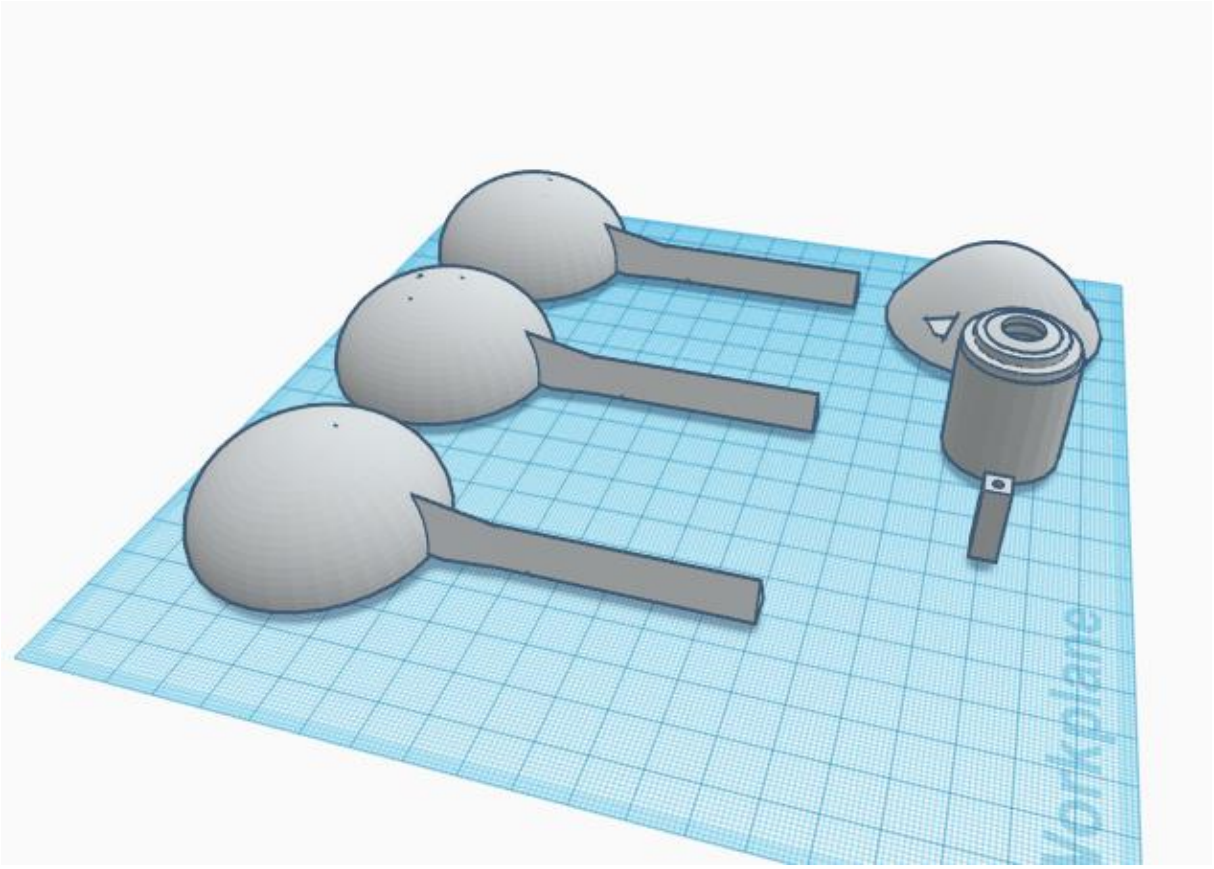


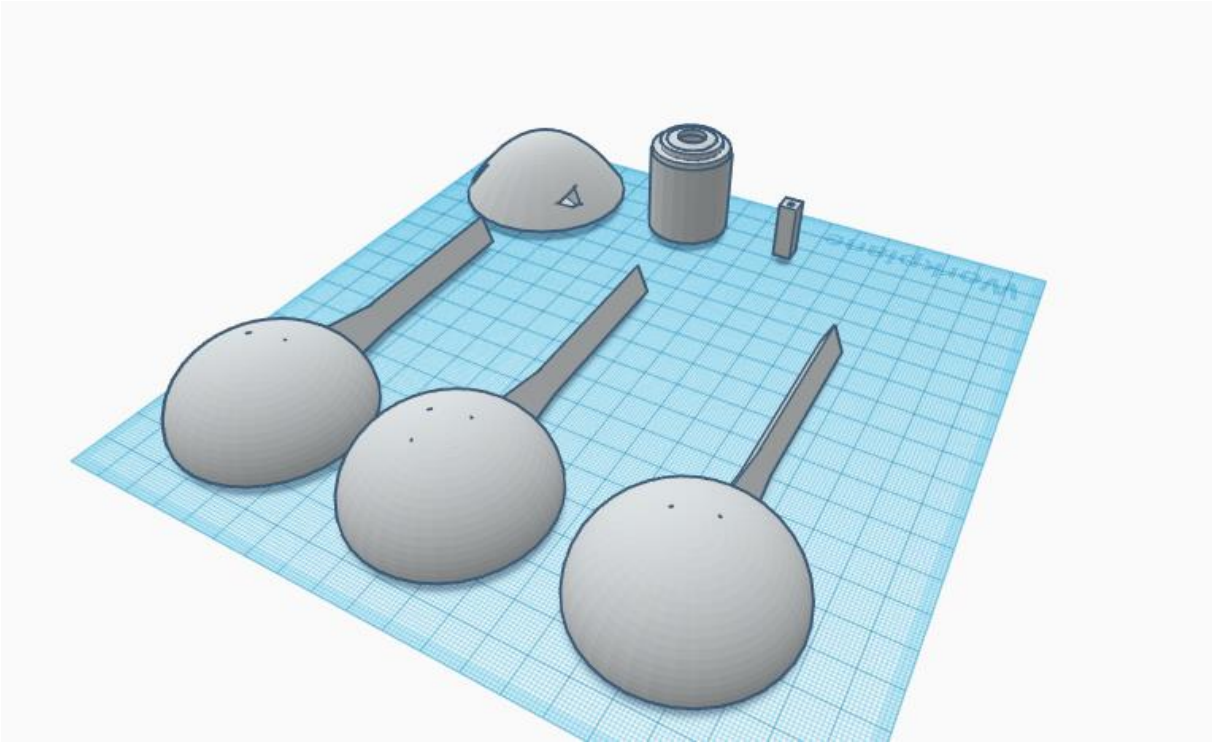
2 – Elektrisch schema met aansluiting van de sensoren en actuatoren



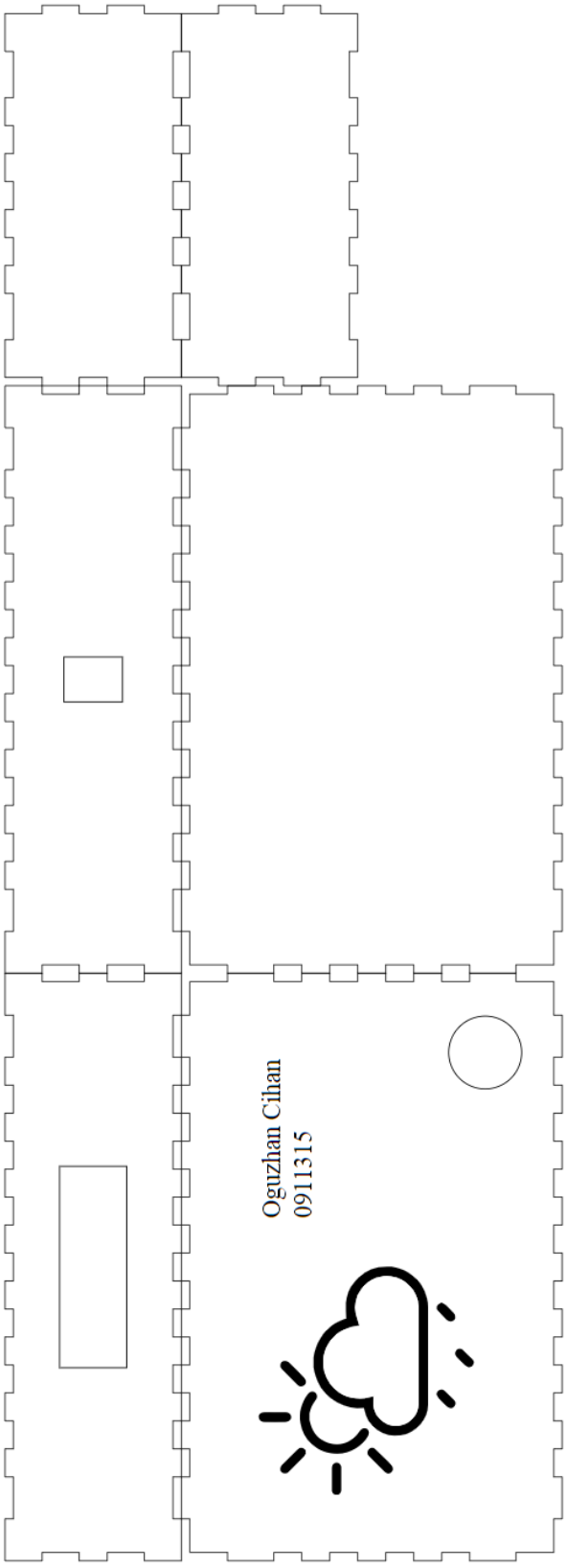
3 – Eenvoudige bouwtekening/ontwerp van windmeter



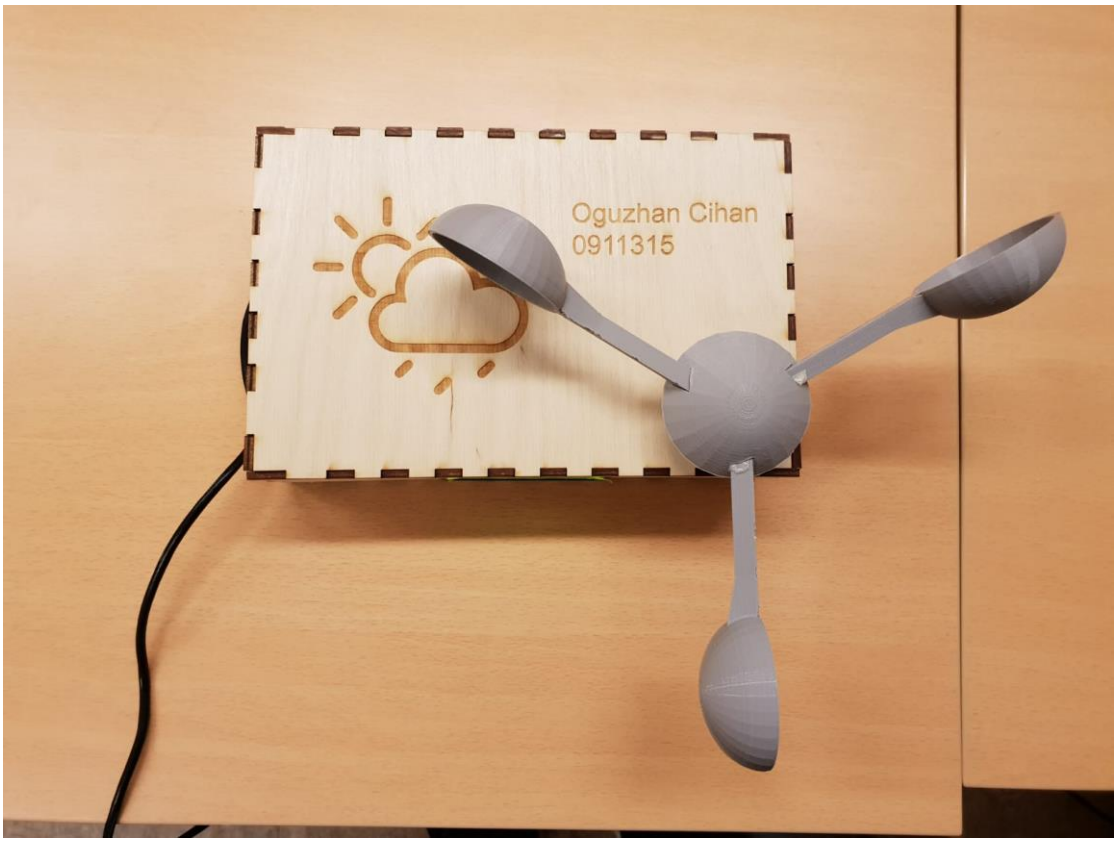




4 – Eenvoudige bouwtekening/ontwerp van behuizing



5 – Foto van het prototype



6 – Gebruikershandleiding

Stap 1

Voorzie het weerstation van stroom door deze aan te sluiten op het stopcontact of met een powerbank.

Stap 2

Zet een mobiele hotspot aan waarmee het weerstation kan verbinden en hierdoor internet connectie heeft.

Stap 3

Wacht en zie hoe de temperatuur, luchtvochtigheid en windsnelheid worden gemeten.

Stap 4

Na 20 seconden wachten hoor je één piep. Dit betekent dat er een 201 httpcode is ontvangen.

Er wordt om de vijf minuten een HTTP request (JSON) gedaan naar een centrale server. Deze request ziet er als volgt uit:

```
{
    "deviceId": "f36962f4-d379-4cce-9537-a897a2608570",
    "temperatureC": 24,
    "humidity": 52,
    "windspeed": 25
}
```

Na het doen van een request kan je de volgende response codes ontvangen.

HTTP 201: Succesvol aangemaakt	= één piep
HTTP 400: JSON data is incorrect	= twee piepjes
HTTP 429: Er is vaker dan 1x binnen 5 minuten data naar de server verstuurd.	= drie piepjes

7 – Programmacode

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <ArduinoJson.h>
#include <DHT_U.h>
#include <DHT.h>
#include <Wire.h>  // This library is already built in to the Arduino IDE
#include <LiquidCrystal_I2C.h> //This library you can add via Include Library >
Manage Library >
LiquidCrystal_I2C lcd(0x3F, 16, 2);

int frequency=1000; //Specified in Hz
int buzzPin=D2;
int timeOn=1000; //specified in milliseconds
int timeOff=250; //specified in millisecods

#include "RunningAverage.h"
RunningAverage sensorAverage(100);
int sampleCount = 0;
int curInputValue;
float h = 0;
float t = 0;
float w = 0;

#define DHTPIN D1     // what digital pin we're connected to
#define DHTTYPE DHT22   // DHT 11
DHT dht(DHTPIN, DHTTYPE);


// WiFi parameters to be configured
const char* ssid = "S9"; // Hoofdlettergevoelig
const char* password = "ictlab2018"; // Hoofdlettergevoelig

void setup(void) {
  Wire.begin(2,0);
  lcd.init();   // initializing the LCD
  lcd.backlight(); // Enable or Turn On the backlight


  Serial.begin(9600);
  dht.begin();
  Serial.print("Bezig met verbinden");
  WiFi.begin(ssid, password); // Connect to WiFi

  // while wifi not connected yet, print '.'
  // then after it connected, get out of the loop
  while (WiFi.status() != WL_CONNECTED) {
     delay(500);
     Serial.print(".");
  }

  // Verbonden.
  Serial.println("OK!");

  // Access Point (SSID).
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());
```

```
  // IP adres.
  Serial.print("IP: ");
  Serial.println(WiFi.localIP());

  // Signaalsterkte.
  long rssi = WiFi.RSSI();
  Serial.print("Signaal sterkte (RSSI): ");
  Serial.print(rssi);
  Serial.println(" dBm");
  Serial.println("");

  while (!Serial) continue;


  Serial.println();

}

void loop() {



  // Check if any reads failed and exit early (to try again).


  curInputValue = (analogRead(A0));             // The input value, from 0 to 1024
  sensorAverage.addValue(curInputValue);        // Add the value to the array to
calculate an average
  w = 0.0685327758 * sensorAverage.getAverage();    // wind average
  sampleCount++;
   lcd.clear();
   lcd.setCursor(0, 0);
   lcd.print("T:");
   lcd.print(t);
   lcd.setCursor(9, 0);
   lcd.print("V:");
   lcd.print(h);
   lcd.setCursor(0, 1); // bottom left
   lcd.print("W:");
   lcd.print(w);
  if (sampleCount < (20000 / 200)) // 60000 ms / 200 ms, so per minute
  {
    delay(200);
    return;
  }
  // What happens after this if, happens every 60000 ms
  h = dht.readHumidity();
  t = dht.readTemperature();
  sampleCount = 0;
  if (isnan(h) || isnan(t)){
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  lcd.clear();
   lcd.setCursor(0, 0);
   lcd.print("T:");
   lcd.print(t);
   lcd.setCursor(9, 0);
```

```
   lcd.print("V:");
   lcd.print(h);
   lcd.setCursor(0, 1); // bottom left
   lcd.print("W:");
   lcd.print(w);
  Serial.println("sensor average: " + String(w));
  sensorAverage.clear();                          // clear the array to
calculate a new average

 StaticJsonBuffer<200> jsonBuffer;
   JsonObject& root = jsonBuffer.createObject();

  root["deviceId"] = "5eedc70f-7441-4246-b6f5-faedc0f3847e";
  root["temperatureC"] = t;
  root["humidity"] = h;
  root["windspeed"] = w;

  root.printTo(Serial);

 if(WiFi.status()== WL_CONNECTED){ //Check WiFi connection status
   HTTPClient http; //Declare object of class HTTPClient

   http.begin("http://hro-sma.nl/api/weatherupdates"); //Specify request
destination
   http.addHeader("Content-Type", "application/json"); //Specify content-type
header
   http.addHeader("X-Device-Id", "5eedc70f-7441-4246-b6f5-faedc0f3847e");
//Specify content-type header
   //http.addHeader("Content-Length", "0");
   String jsonString;
   root.prettyPrintTo(jsonString);
   int httpCode = http.POST(jsonString); //Send the request
   String payload = http.getString();  //Get the response payload

   Serial.println(httpCode); //Print HTTP return code
   Serial.println(payload); //Print request response payload

   if(httpCode == 201)
   {
    Serial.println("correct");
    tone(buzzPin, frequency);
    delay(timeOn);
    noTone(buzzPin);
    delay(timeOff);
   } else{
    Serial.println("niet correct");
      tone(buzzPin, frequency);
      delay(timeOn);
      noTone(buzzPin);
      delay(timeOff);
      tone(buzzPin, frequency);
      delay(timeOn);
      noTone(buzzPin);
      delay(timeOff);
    }


   http.end();  //Close connection
  } else {
```

```
    Serial.println("Error in WiFi connection");
  }
  delay(300000);  //Send a request every 5 minutes
  //delay(20000);  //Send a request every 20 seconds
}
```