Софийски университет "Св. Климент Охридски"

Факултет по математика и информатика

Учебна дисциплина „Компютърни Системи за управление на роботи“

# Курсов проект

на

Анастас Стефанов Дафов, фак. № 81220

Мартин Руменов Терзийски, фак. № 81155

Тема: „Проектиране и разработване на Система за измерване на качеството на въздуха„

Научен ръководител

доц. д-р Симеон Цветанов

София 2018

Съдържание

[Съдържание 2](#_Toc515373263)

[Увод 3](#_Toc515373264)

[1 Въведение 4](#_Toc515373265)

[2 Проектиране 6](#_Toc515373266)

[3 Разработване 10](#_Toc515373267)

[3.1 NodeMCU 10](#_Toc515373268)

[3.1.1 Използвани класове и методи 11](#_Toc515373269)

[3.1.2 Инсталация на необходимия софтуер 12](#_Toc515373270)

[3.2 Сървър 13](#_Toc515373271)

[3.2.1 Подробно описание на базата с данни 14](#_Toc515373272)

[3.2.2 Подробно описание на сървъра 14](#_Toc515373273)

[3.2.3 Инсталация на необходимия софтуер 15](#_Toc515373274)

[3.3 Клиент 16](#_Toc515373275)

[3.3.1 Подробно описание на различните пътища 16](#_Toc515373276)

[3.3.2 Инсталация на необходимия софтуер. 17](#_Toc515373277)

[4 Експерименти и тестове 17](#_Toc515373278)

[Заключение 22](#_Toc515373279)

[Източници 23](#_Toc515373280)

# Увод

Целта на проекта е да се проектира и разработи система за измерване на качеството на въздуха. Системата трябва да включва подходящ Уеб интерфейс, на който да се визуализират данните събирани от системата под формата на графики за различните категории данни. Чрез тази информация може да се следи качеството на въздуха освен в реално време и в продължителни периоди като последните ден, час, седмица, месец и година.

За реализирането на тези цели са необходими сензори за изчисляването на различни характеристики на качеството на въздуха, като температура, атмосферно налягане, количество на фини прахови частици, относителна влажност на въздуха и приблизителна надморска височина. Необходим е и процесор с връзка към Wi-Fi мрежа, който освен да управлява сензорите, да праща данните към Уеб сървър, който се грижи за запазването и предоставянето им за уеб интерфейса. Самите графики са достъпни за потребителя от уеб интерфейса.

# Въведение

В наши дни проблемът със замърсяването на околната среда става с все по-значим за човека. Тъй като човекът взима всички необходими му ресурси от околната, нейното замърсяване оказва пряко влияние върху всеки от нас, дори и да не го забелязваме. Освен това проблемът със замърсяването пряко влияе и върху бъдещето на човечеството и решаването на този проблем трябва да бъде от първостепенна важност за нас.

Атмосферата е само част от заобикалящата ни среда, но замърсяването на атмосферата вече е довела до сериозни проблеми и те само ще стават по-сериозни, ако не се вземат адекватни мерки. Замърсения въздух пренася всякакви невидими частици, продукт на човешката дейност, които освен да замърсяват въздуха който дишаме, могат да бъдат поети и с храната и водата които ядем и пием. Пренатоварването на нашите тела с такива вредни частици води до всевъзможни здравословни проблеми, което е едно пряко въздействие на атмосферното замърсяване върху човека. Други последствия са глобалното затопляне и парниковия ефект.

За момента няма универсално решение на този проблем, което да премахне всички вреди от замърсяването на атмосферата и да предотврати нейното замърсяване, но можем да вземем някакви мерки като започнем да събираме информация за състоянието на въздуха в околността ни. Тази информация може да бъде получена от специални станции за следене на качеството на въздуха. Една такава станция може да бъде проста система от подходящи атмосферни сензори и микроконтролер, който да обработва информацията от тях. За по-голяма ефикастност на станцията, тя може да бъде свързана към Интернет и чрез подходящ интерфейс да се показват данните от нейните измервания на всякакви машини на които може да бъде инсталиран Уеб браузър. По този начиин се прави достъпът до данните глобален. Интерфейсът е възможно да предоставя и средства за анализиране на същите тези данни – от графики показващи измерените стойности от различните сензори в различни периоди от време (час, ден, месец и т.н) до по-сложни методи включващи намиране на зависимости с други събития.

Предложената реализация, на такъв тип станция за измерване на качеството на въздуха, в този проект е много проста. Тя включва сензори за измерване на фини прахови частици, температура на въздуха, надморска височина, относителна влажност на въздуха и атмосферно налягане. В нея се включва и микроконтролер с вграден модул за връзка с Wi-Fi мрежа и Интернет. Задачата на микроконтролера е да приема данните, прочетени от сензора и да ги изпраща през Интернет към сървър. Сървърът е специално проектиран за целите на този проект и неговата работа включва обработка на данните и изпращането им в подходящ вид към клиентса част, достъпна през бразуър. Клиента ще може да види графики, предоставящи му средните получени стойности от всеки сензор за последния час, ден, седмица, месец и година, както и последните получени данни. Към сървъра е свързана и база от данни, служеща за запомняне на сензорните данни.

Тази станция, както вече беше споменато, е много проста за направа. Подходяща е за употреба и в домашни условия и за по-големи площи – от няколко пресечеки до потенциално цял жилищен квартал. Интегрирането на няколко такива станции и подходящото им разпределение по ключови точки, може да създаде мрежа покриваща цял град. С данните получени от мрежата, освен да се види състоянието на въздуха, може да се изгради и система за ранно предупреждение при внезапни критични ситуации или да се отрият неочаквани причини и източници на замърсяване. Данните от тези станции могат също така да послужат за различни проучвания свързани пряко или косвено с качеството на въздуха.

Дизайнът на предложената от нас станция е базиран на дизайна за метеорологична станция предложен от Проект AirBG.info на Фондация Код: България ([https://airbg.info](https://airbg.info/)), които са част от глобалния проект Luftdaten.info( [https://luftdaten.info](https://luftdaten.info/)). Докато хардуерът в двете станции е почти идентичен, но кодът управляващ микроконтролера, сървъра и клиента е независим от този на Luftdafen.info.

# Проектиране

Системата се състои от три части – апаратна част, сървър и уеб интерфейс.

Апаратната част е съставена от контролен процесор и два сензора. Процесорът има и чип за връзка с Wi-Fi, за да може да се изпращат данните събрани от сензорите към сървъра. Първият от сензорите измерва количеството на праховите частици във въздуха. Тези частици могат да бъдат два вида :

* PM2.5 – това са прахови частици с диаметър до 2,5 микрометъра;
* PM10 – това са по-едри частици от тези в горната категория и техния диаметър е до 10 микрометъра.

Сензорът, измерващ тези частици, показва количеството им в кубичен метър.

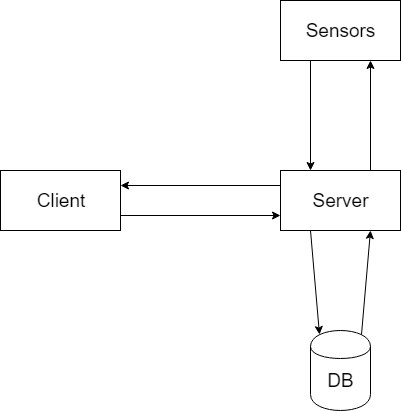
Вторият сензор, част от системата, измерва температурата, относителната влажност на въздуха, атмосферното налягане и приблизителна надморска височина. Температурата е измерена в градуси по Целзий, относителната влажност в проценти, атмосферното налягане в хектопаскали и надморската височина в метри.

**Необходимия хардуер е следния:**

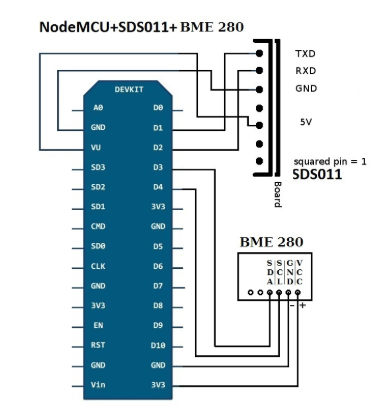
* **Един брой микроконтролер NodeMCU ESP8266;**
* **Един брой сензор CJMCU-280 BME;**
* **Един брой сензор nova PM SDS011;**
* **Осем броя кабели за свързване на крачетата(pins) на сензорите с микроконтролера;**
* **USB към Micro USB кабел, плосък с дължина три метра;**
* *Препоръчително е станцията да се постави в някаква защита, за справки вижте* [*https://airbg.info/how-to/.*](https://airbg.info/how-to/)
* *Трябва да се отбележи, че е необходимо спояване на* ***pins*** *към интерфейсите на сензора* ***CJMCU-280 BME****.*
* *Необходимо е захранване от поне 1А и вход за USB към него.*

Уеб сървъра е връзката между клиента (интерфейса) и данните. Данните са предоставени от сензорите и се запазват в база от данни. Сървъра запазва и чете тези данни.

Клиента (уеб интерфейса) е това, което предоставя данните на потребителя под формата на графики за час, ден, седмица, месец, година. Също така интерфейса предоставя и възможност да се видят последните данни получени от сензорите.



***Фиг. 1:*** *На картинката са показани различните компоненти на системата, означени с правоъгълници и потокът на данните между тях, означен със стрелки и базата от данни. Стрелката започва от източника на информация и завършва в приемника.*



***Фиг. 2****: На картинката е показан начинът на свързване на сензорите с микроконтролера, т.е кои крачета(pins) се свързват между сензорите и мокроконтролера с кабелите. Схемата е взета от* [*https://airbg.info/how-to/*](https://airbg.info/how-to/)*. Трябва да се вземе впредвид, че в реазлизираната в този проект станция, сензорът BME 280 е заменен със CJMCU-280 BME и при него има разлика в някой от означенията на интерфейсите. Конкретно SCL e заменено със SDI, а SDA със SCK.*

Функционално станцията е проектирана да събира данните от сензора и на фиксиран интервал от време да ги изпраща през Интернет към сървъра, след като е установена връзка към Wi-Fi мрежа. Сървърът се грижи за получаването на данните, обработването им с целта да се подготвят за изобразяването им в графики след изпращане към клиента и използването на базата от данни по начин подпомагащ му тази обработка и подготовка на данните. Клиентът предоставя удобен и изчистен интерфейс позволяващ на потребителя да види данните от сензорите и графиките, на прочетените данни от различните сензори, за различните периоди от време.

# Разработване

В този раздел ще бъдат разгледани различните софтуерни продукти използвани за създаването на кода стоящ зад отделните компоненти на станцията. Също така ще бъдат предоставени и инструкции за инсталацията на всички тези продукти, тяхното допълнително необходимо настройване и функцията които те изпълняват.

## NodeMCU

Програмирането на процесора NodeMCU ESP8266 и сензорите nova PM SDS011 и CJMCU-280 BME се извършва чрез средата [Arduino](https://www.arduino.cc/). Използвани са следните библиотеки :

* [ESP8266WiFi](https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi) - С тази библиотека се осъществява свързването на процесора към безжична мрежа. Необходими са име и парола на мрежата за да се осъществи връзката. //начин на инсталация
* [SDS011](https://github.com/ricki-z/SDS011) – Библиотека с лиценз за използване GNU GENERAL PUBLIC LICENSE v3.0. Чрез нея се управлява връзката със сензорът nova PM SDS011 и обработката на данните от сензорът.
* [ESP8266HTTPClient](https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266HTTPClient) – Създава се HTTP клиент чрез който се изпращат събраните данни от двата сензора към сървъра.
* [Wire](https://github.com/esp8266/Arduino/blob/master/libraries/Wire/Wire.h) – Библиотеката предоставя инструменти за конфигуриране на pins на процесора. С нейна помощ се настройва процесорът за работа със сензора CJMCU-280 BME по протоколът I2C.
* [SPI](https://github.com/esp8266/Arduino/tree/master/libraries/SPI) – Библиотеката позволява да се конфигурира комуникацията със сензорът CJMCU-280 BME да бъде с протоколът SPI.
* [Adafruit\_Sensor](https://github.com/adafruit/Adafruit_Sensor) и [Adafruit\_BME280](https://github.com/adafruit/Adafruit_BME280_Library) – Библиотеките са необходими за разчитането на данните от CJMCU-280 BME и работата със сензора.

Средата Arduino е настроена да компилира кода със следните настройки:

* Board : NodeMCU 1.0(ESP – 12E Module);
* CPU Frequency : 160 MHz;
* Flash Size : 4M(3M SPIFFS);
* IwIp Variant : v2 Lower Memory;
* Debug Port : Disabled;
* Debug Level : None;
* Erase Flash : All Flash Contents;
* Upload Speed : 115200.

### Използвани класове и методи

Основните използвани класове създават обекти за работа с двата сензора. Това са класовете ***SDS011*** и ***Adafruit\_BME280*** предоставени от съответните библиотеки. За връзка с уеб сървъра се създава и обект на класа ***HTTPClient***.

Използвани са следните методи на класа ***SDS011*** :

* begin(D1,D2); - С този метод се задават pins с които е свързан сензорът с контролера. Необходимо е да бъде извикан, защото не може да се каже на контролера, че сензорът е свързан към него по друг начин с използваната библиотека.
* read(&p25,&p10); - Този метод ще запази прочетените от сензора стойности за двата вида прахови частици в двата му аргумента. Функцията връща стойност, която показва дали прочитането на данни от сензора е минало без грешка.

Използвани са следните методи на класа ***Adafruit\_BME280*** :

* begin(); - Този метод подготвя екземпляра на класа за бъдещата му работа.
* readTemperature(); - Методът връща прочетената стойност за температурата в градуси по Целзий.
* readPressure()/100.0F – Методът прочита и връща стойността за атмосферното налягане.
* readAltitude(SEALEVELPRESSURE\_HPA); - Методът връща относителната надморска височина на която се намира сензорът. Трябва да се има в предвид, че данните не са много прецизни, защото последователни четения могат да имат до 0,5 метра разлика. Аргументът SEALEVELPRESSURE\_HPA е зададен по-рано от извикването на този метод в програмата и трябва да показва относителното атмосферно налягане на мястото, където ще бъде поставена системета, в хектопаскали.
* readHumidity(); - Методът връща прочетената стойност за относителанта влажност на въздуха в проценти.
* Стойностите пазещи информацията за прочетените сензорни данни от всяка категория се пазят в променливи от тип **float**.

Използвани са следните методи на класа ***HTTPClient:***

* begin(server\_adress); - След създаване на обект от горния клас, се извиква този метод за да се започне връзка със сървъра. Адреса на сървъра се задава като символен низ и се препоръчва адреса да започва с http. При https не може да се установи връзка.
* POST(message); - И извикването на този метод се изпраща http заявка към сървъра. Аргумента е символен низ и в него са данните от сензорите.
* end(); - Прекратява връзката със сървъра.

Допълнителни методи :

* Wire.begin(D3,D4); - Методът позволява комуникацията по протоколът I2C чрез съответните pins. Към тях е свързан сензорът CJMCU-280 BME.
* Wifi.begin(ssid,pass); – Свързва контролера към мрежата отговарящата на аргументите на метода.
* Wifi.disconnect(); - Прекратява връзката към мрежата.

### Инсталация на необходимия софтуер

1. Инсталацията започва с изтегляне и инсталиране на Ардуино средата от <https://www.arduino.cc/en/Main/Software>. Свалете Arduino IDE от този адрес.
2. В Arduino, меню „File -> Preferences ->Setting, намерете Additional Board Manager URLs“ полето и вмъкнете този URL: <http://arduino.esp8266.com/stable/package_esp8266com_index.json>.
3. След това от менюто „Tools -> Board-> Boardmanager“, намерете „esp8266“ и инсталирайте „esp8266 by ESP8266 Community“
4. Необходимо е да се инсталира и CH340G USB Driver, за да може да се осъществи връзката между средата Arduino и процесорът NodeMCU.
5. Възможно е да имате вече инсталирани някой от библиотеките споменати по-горе от инсталирането на Ардуино. За тези които липсват, в Arduino използвайте менюто „Sketch -> Include Library -> Manage Libraries…” за да достъпите библиотечния мениджър и в него да потърсите липсващите библиотеки или влезте в менюто „Sketch -> Include Library -> Add .ZIP Library…” за да добавите ръчно липсваща библиотека. Ако използвате втория метод, трябва да свалите библиотеките в ZIP архив предварително. Използвайте линковете към самите библиотеки.

* След инсталацията на библиотеката Adafruit\_BME280 е необходимо да се направи промяна в нея, защото използвания сензор CJMCU-280 BME използва различен адрес за връзка. Влезте в папката където Ардуино е инсталирал библиотеката и намерете файла Adafruit\_BME280.h.(По подразбиране папката се намира в “C:\Users\%username%\Documents\Arduino\libraries\) Отворете файла и намерете следния ред : #define BME280\_ADDRESS (0x77) и променете (0x77) на (0x76) и запазете файла.

## Сървър

Сървърът всъщност представлява express REST сървър. Комуникацията с него става чрез HTTP Get и Post заявки на различни пътища. Всички части от приложението комуникират с него. NodeMCU-то изпраща данните от сензорите на сървъра. Клиента чете данните от сървъра и ги изобразява чрез графики.

Мястото, където се запазват данните е нерелационна база с данни **mongoDB**. Там се запазват данните предоставени от сензорите. От там се и взимат данните, когато клиента(браузъра) ги изисква.

Използвани са следните библиотеки:

* [expressJS](https://expressjs.com/) – Изключително популярна библиотека, предоставяща възможност за REST API.
* [cors](https://www.npmjs.com/package/cors) – Библиотека, позволяваща ни да заобиколим same-origin-policy, която по принцип ограничава комуникацията между frontend и backend.
* [body-parser](https://www.npmjs.com/package/body-parser) – Тази библиотека ни позволява лесно да обработваме данните, които са ни подадени в тялото на POST заявка.
* [moment](https://momentjs.com/) – Отново изключително популярна библиотека, що се отнася до обработка на времена и timestams. Това и прави тя на сървъра. Обработва времената на данните подадени от сензорите, за да бъдат запазени в базата от данни по лесен за обработка начин.
* [Mongodb](https://www.mongodb.com/) – нашата база с данни.
* [mongoose](http://mongoosejs.com/docs/api.html#Document) – помощна библиотека, позволяваща ни по-лесна работа с базата данни. Също така и създаване на модели.

### Подробно описание на базата с данни

Базата с данни е нерелационна mongoDB. Имаме една колекция (аналогия на таблица в релационните бази от данни) SensorsData. Що се отнася до документите(записи) в нея, всеки документ има следните полета:

* type – Тип на данните. Едно от: „pm25“, „pm10“ , „temperature“ , „pressure“ , „humidity“ , „altitude“
* timeSpan - За коя точно графика се отнасят данните. Данните за час, за ден, за седмица, за месец или за година
* data.x – Точното време за конкретната графика. Например, ако е таблицата за седмица и записа е от Вторник, тогава тази данна на записа ще е „Tuesday“
* data.y – Точната стойност за конкретното време показана от сензора
* count – Броят данни получени в конкретното време. Това ни е нужно за да смятаме средно аритметично за времето.

Базата с данни не пази архив, защото новите данни се презаписват върху стари данни, които вече не са ни нужни за графиките. Това позволява „безкрайна“ работа на приложението поне що се отнася до базата с данни. Максималният размер, който заемат данните е около 80 КБ.

За хостване на базата данни са използвани услуги, предоставени от Amazon и mlab.

### Подробно описание на сървъра

За да се постигне продуктивност на приложението повечето изчисления се правят на сървъра, но не всички. Приложението все пак има frontend част, която чертае графиките. Сървърът представлява всъщност express REST API. Пътищата са:

* GET / - Тук взимаме последните данни, които са подадени от сензорите.
* GET /fineParticles - Тук взимаме данните от сензора за фини прахови частици, които взимаме от базата с данни. Изпращаме само данните, които имат значение за графиките, тоест не изпращаме стари данни, ако има такива в базата. Върху тези данни по принцип ще се запишат нови, но е възможно все още да няма такива.
* GET /atmosphericPressure - Тук взимаме данните от сензора за атмосферно налягане, които взимаме от базата с данни. Изпращаме само данните, които имат значение за графиките, тоест не изпращаме стари данни, ако има такива в базата. Върху тези данни по принцип ще се запишат нови, но е възможно все още да няма такива.
* POST / - Така самият сензор предава данните към сървъра. В тялото на заявката се подават данните във формата ”pass|pm25|pm10|pressure|temperature|humidity|altitude”. Паролата е там, за да имаме поне някаква защита от пращане на произволни данни. Та тези данни се преработват на сървъра, за да се стигне исканият формат от базата данни, а имено данните да са подредени по удобен начин за изобразяване на графиките. Също така и се проверява дали може да се презапише върху стари данни и се презаписва, ако има такива подходящи стари ненужни данни.

Сървъра се хоства от Amazon, чрез Heroku на адрес <https://fathomless-peak-36706.herokuapp.com/>

### Инсталация на необходимия софтуер

След като клонирате репозиторито на проекта, отивате чрез терминал в папката **server** и изпълнявате командата **npm install.** За целта трябва да имате инсталиран nodejs: <https://nodejs.org/en/>. По принцип можете да свалите всеки един от модулите чрез npm по отделно.

ПП. Ако искате да стартирате проекта на вашия компютър с ваше NodeMCU със сензори ще Ви се наложи да промените няколко неща.

1. Ще трябва да махнете паролата, която се изисква от сървъра при POST заявка от NodeMCU –то. В момента е сложена, за да не може „лесно“ 3-ти лица да пращат данни до сървъра.
2. Ще трябва да създадете и свържете собствена база с данни, тъй като потребителя и паролата записани в адреса.

## Клиент

Клиентът (браузъра, уеб интерфейса или frontend) представлява Webpack VueJS vue-router приложение, чиято цел е да изобрази данните получени от сървъра по красив, лесно разбираем начин като информира потребителя, за данни в критични стойности.

Използвани са следните библиотеки:

* [Vue](https://vuejs.org/) – Библиотека за разработване на single page applications
* [Vue-router](https://router.vuejs.org/en/) – Библиотека, която ни позволява да не презареждаме на ново цялата странница като сменяме пътищата в браузъра.
* [axios](https://github.com/axios/axios) – Библиотека, която ни позволява лесно да се свържем със сървъра (или всякакви други REST APIs).
* [chartjs](https://www.chartjs.org/) – Библиотека, предоставяща ни възможност за чертаене на всякакви разнообразни графики.
* [moment](https://momentjs.com/) – Същата библиотека, която използваме и на сървъра. Тук я използваме, за да нагласим координатата за времето на графиките да съвпада с реалното време.

### Подробно описание на различните пътища

На различни пътища в браузъра достъпваме различни ресурси от сървъра. Преминаването от един път до друг е мигновено заради vue и vue-router. За да избегнем объркване имената на пътищата са подобни между сървъра и браузъра. На клиента са:

* / - Тук достъпваме последните данни от сензорите. Това се случва като през 5 секунди изпращаме GET request към сървъра на /.
* /fineParticles - Тук взимаме данните от сървъра, като използваме /fineParticles пътя на сървъра. Тези данни вече са разпределени според времето когато са получени. Единственото, което правим на клиента е че ги изобразяваме в графики. Отново пращаме такива заявки към сървъра на всеки 5 секунди, за да обновяваме новите средни данни за времевите периоди изобразени на графиките.
* /atmosphericPressure - Тук взимаме данните от сървъра, като използваме /atmosphericPressure пътя на сървъра. Тези данни също са вече разпределени.

Уеб интерфейса е качен в Google Firebase и се намира на адрес:

<https://fine-particles-web-interface.firebaseapp.com/>

### Инсталация на необходимия софтуер.

След като клонирате репозиторито на проекта, отивате чрез терминал в папката **client** и изпълнявате командата **npm install.** За целта трябва да имате инсталиран nodejs: <https://nodejs.org/en/>. По принцип можете да свалите всеки един от модулите чрез npm по отделно.

# Експерименти и тестове

Тестовете и тестовите резултати бяха разделени на две части. Първата част беше създаването на станцията по моделът предоставен от **airbg.info**. Втората част беше създаването на станцията по моделът представен в проекта, което се оказа по-лесна и удобна за употреба алтернатива.

Следвайки инструкциите предоставени в сайта **airbg.info**, успешно бе направена станцията по дадените стъпки. По време на тестването се сблъскахме със сериозни затруднения засягащи свързването към системата на външния проект. В последствие успешно беше уствановена връзка със системата и успешно беше изпозлван предоставения ни интерфейс, на който можехме да наблюдаваме графики на изпратените данни от сензорите. Не успяхме да интегрираме напълно нашата станция към глобалната мрежа от станции на проекта заради изключително сложния и бавен метод на регистрация на нашата станция към системата. Към трудността на цялото използване на системата на **luftdaten.info** допринесе и факта, че всичко в системата е написано на немски.

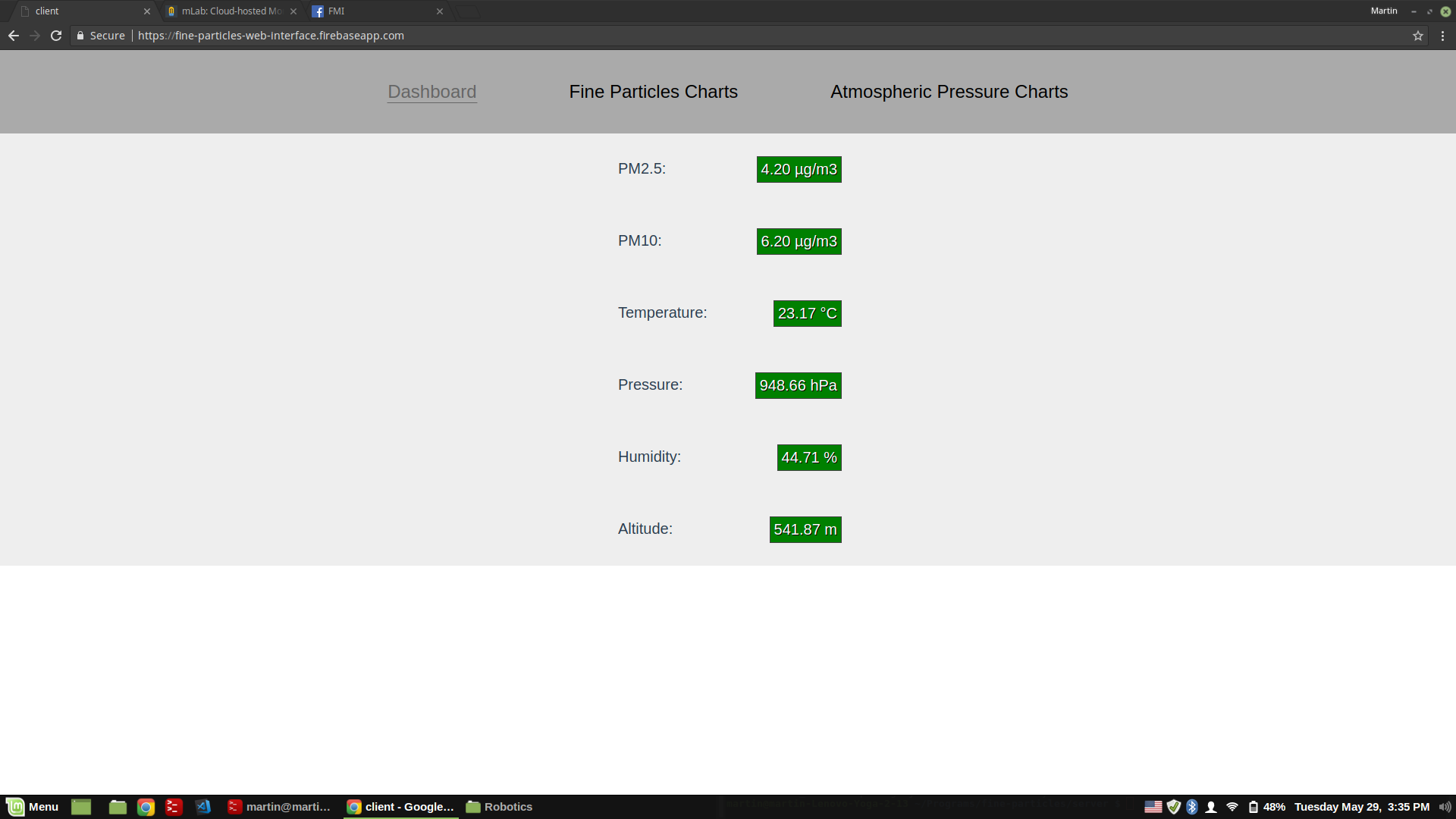
Преминавайки към собствената реализация на станция за измерване на качеството на въздуха, имахме за задачи да напишем код управляващ микроконтролера, както и собствен сървър и потребителски интерфейс (клиент). Успешно завършихме гореспоменатите задачи и преминахме към тестване на системата. Тестовете преминаха успешно и установихме, че цялостната система работи по желания начин.

По време на тестовете бяха установени следните бъгове:

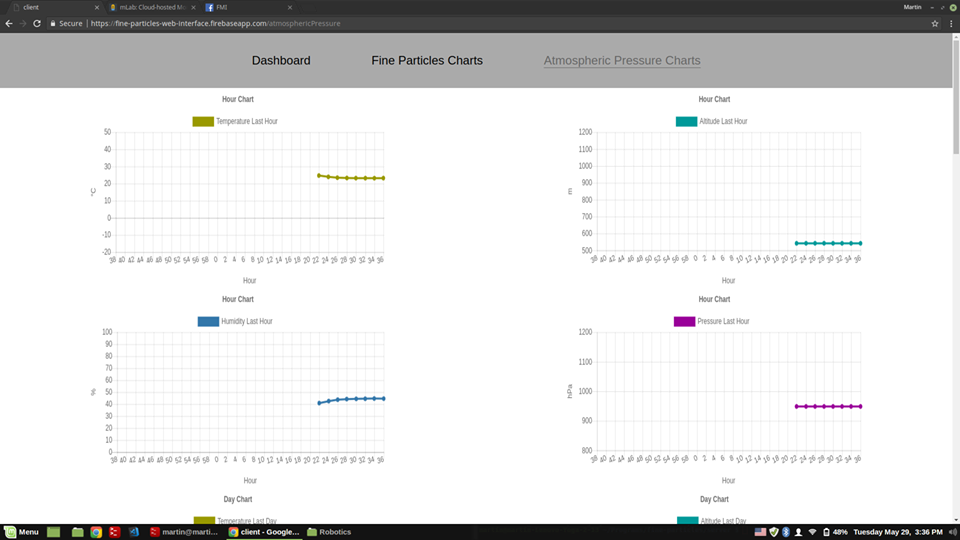
* Сензорът CJMCU-280 BME от време на време отчита очевидно грешни стойности, които в последствие правят и графиките към тези данни грешни. Този бъг може да е причинен от лошо спояване на pins към сензорът и/или не достатъчно добра връзка между сензора и микроконтролера.
* Библиотеката използвана за рисуване на графиките не винаги рисуваше коректно данните. Вместо по подадената линейна функция, показваща стойностите в различните времеви интервали на отделните графики, към няколкото типа сензорни данни, се рисуваше фигура по-близка до произволна начупена линия, а не графика на функция. Възможна причина за това е грешните сензорни данни получени от гореспоменатия сензор.
* Препоръчва се използването на сравнително къс захранващ кабел(по-къс от 1 метър).

Подробно описание на станцията и нейното сглобяване.

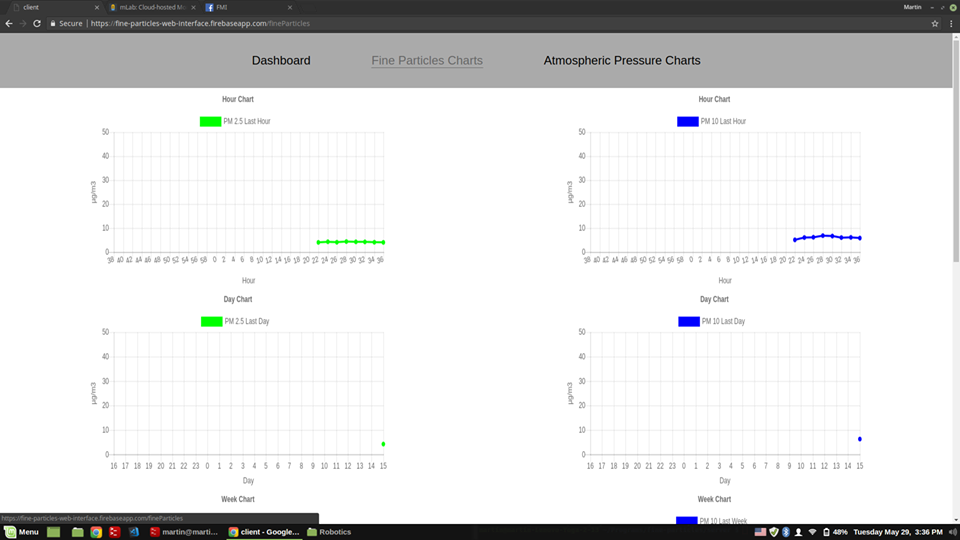
След успешното завършване на проекта направихме снимки илюстриращи работата на системата.



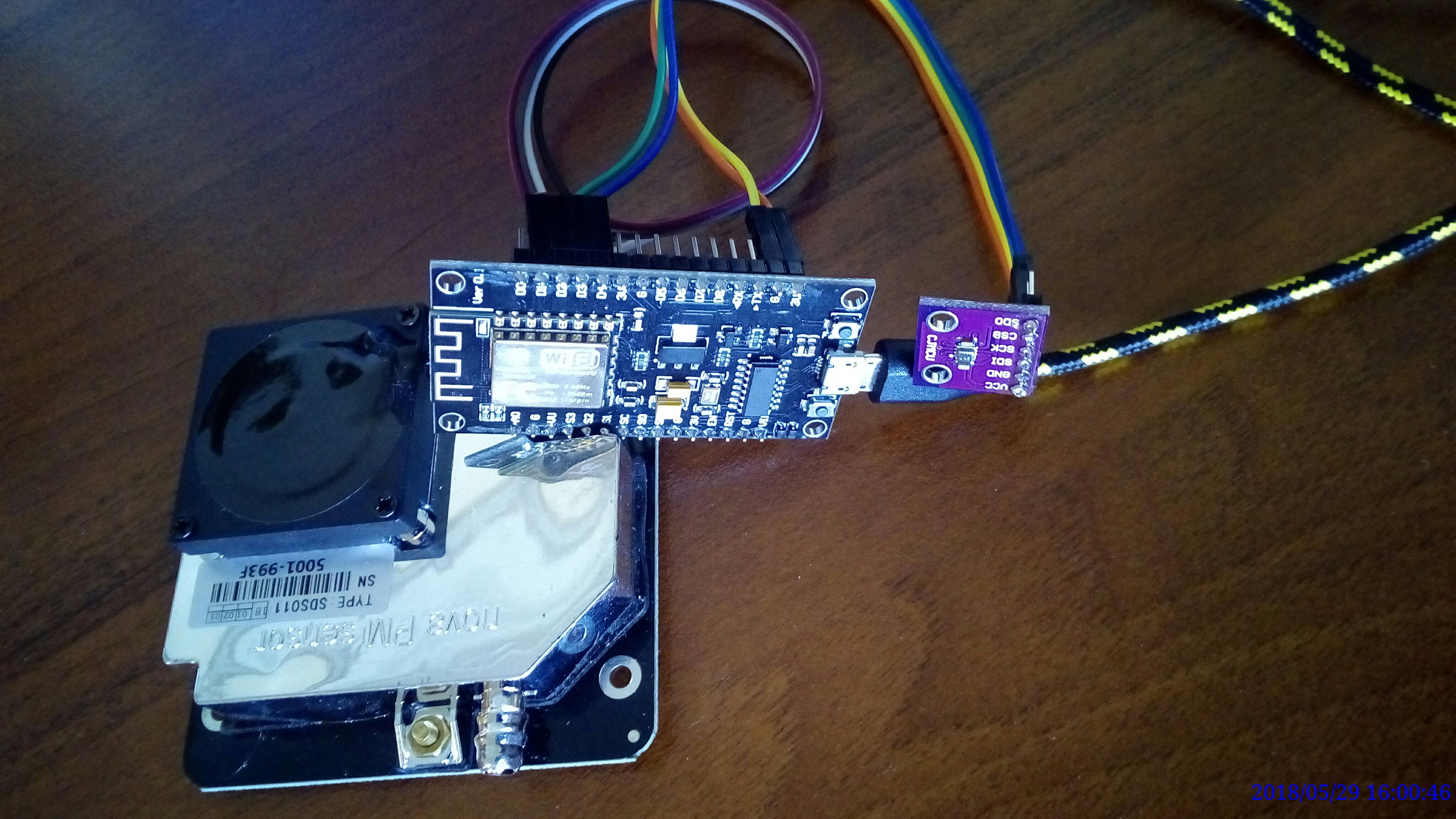
***Фиг. 3 :*** *На картинката* *е показана страница от клиентската част на която се виждат последно прочетените данни от сензорите.*



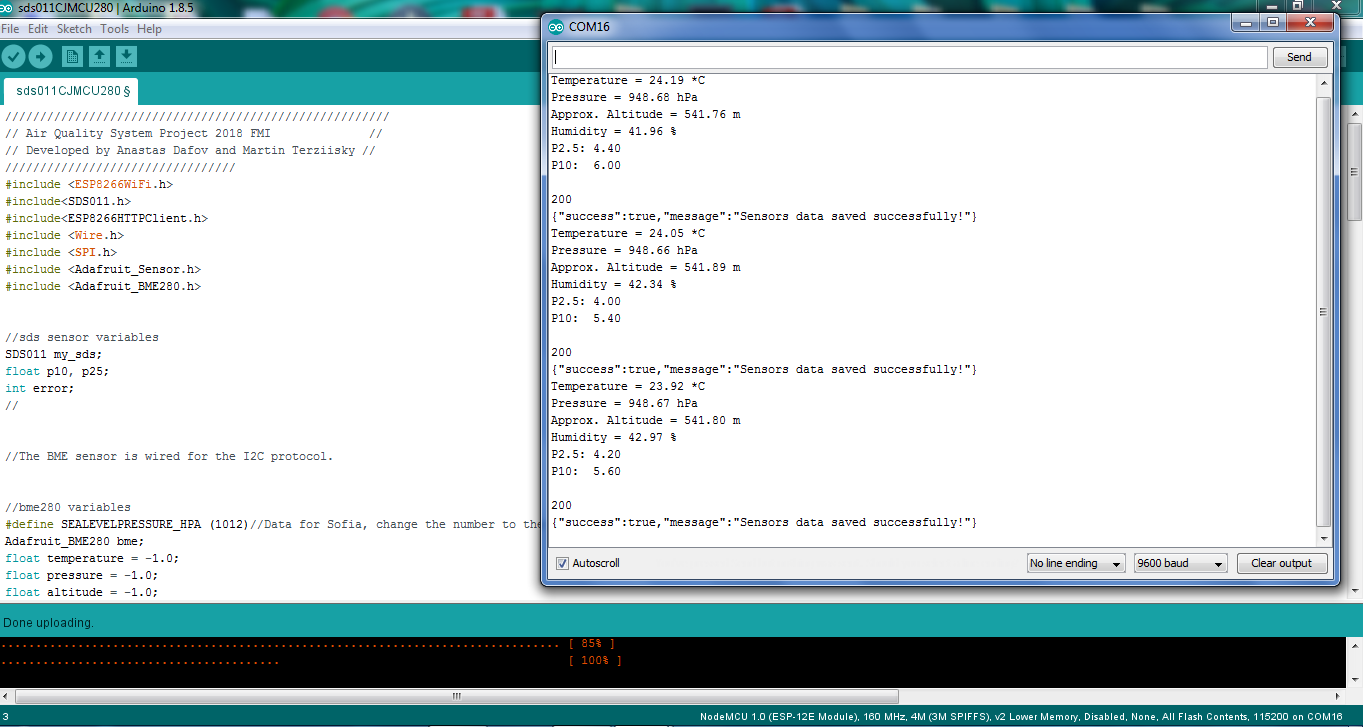
***Фиг. 4 :*** *На**картинката се виждат част от графиките, показващи данните събрани от сензора* ***CJMCU****, на клиентската част.*



***Фиг. 5 :*** *На**картинката се виждат част от графиките, показващи данните събрани от сензора* ***nova PM sds011****, на клиентската част.*



***Фиг. 6 :*** *На снимката се вижда физическата реализация на станцията.*



***Фиг. 7 :*** *На картинката се виждат данните изпратени от микроконтролера към Serial Monitor на съответния свързан порт. Те съдържат измерванията от всички сензори, както и отговори „200“ (ОК) и съобщение за успешно приета заявка към сървъра, от сървъра.*

# Заключение

Системата може да бъде подобрена по няколко начина.

В настоящия момент мрежата към която се свързва процесорът се задава ръчно в кода на управляващата го програма. Това значи, че при смяна на мрежата или дори промени по името и паролата на текушо използваната мрежа, се налага прекомпилиране и качване на новия код върху чипа при всяка такава промяна. За по-голямо удобство при ползване на системата, е добре да може да се задава мрежата динамично по-подходящ за това начин.

Сензорите не винаги измерват данни които са реалистични или направо грешни. За да се елиминират тези грешни данни и да се повиши точността на измерванията може да се прибегне до следните две решения на проблема. На първо място да се имплементира метод за изхвърлянето на нереалистичните данни. На второ място да се използват по-прецизни аналози на текущите сензори. Така с минимални промени по кода, ще се повиши качеството на системата.

При продължителна работа на системата се получават грешни данни. Това е така, тъй като има малки грешки в логиката на backend-а. Бъдещо подобрение на системата е да се отстранят тези грешки.

В заключение, станцията за измерване на качеството на въздуха е прост и евтин източник на информация за голяма част от заобикалящата ни околна среда, а именно атмосферата.

# Източници

Сходни проекти:

[https://airbg.info](https://airbg.info/)

[https://luftdaten.info](https://luftdaten.info/)

Arduino:

[https://www.arduino.cc](https://www.arduino.cc/)

Библиотеки към Arduino:

<https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi>

<https://github.com/ricki-z/SDS011>

<https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266HTTPClient>

<https://github.com/esp8266/Arduino/blob/master/libraries/Wire/Wire.h>

<https://github.com/esp8266/Arduino/tree/master/libraries/SPI>

<https://github.com/adafruit/Adafruit_Sensor>

<https://github.com/adafruit/Adafruit_BME280_Library>

Продукти и библиотеки към сървъра, БД и клиента:

[https://www.mongodb.com](https://www.mongodb.com/)

<http://mongoosejs.com/docs/api.html>

<https://expressjs.com/>

<https://www.npmjs.com/package/body-parser>

<https://www.npmjs.com/package/cors>

<https://momentjs.com/>

<https://vuejs.org/>

<https://router.vuejs.org/en/>

<https://www.chartjs.org/>

<https://github.com/axios/axios>

<https://nodejs.org/en/>