International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 11, Issue 06, June 2021)
Manuscript Received: 06 May 2021, Received in Revised form: 01 June 2021, Accepted: 06 June 2021 DOI: 10.46338/ijetae0621_06

Development of a Mobile Edge Device and an Information System to Analyze Particulate Matter Distribution as a mHealth Service

Franz Frederik Walter Viktor Walter-Tscharf
University of Trento Department Industrial Innovation, Trento, Italy

***Abstract*— Particulate matter is an air pollutant consistent of very small particles that are suspended in the air. Short-term exposure may result in respiratory symptoms such as shortness of breath, throat and nose irritation, chest tightness, coughing, in addition to eye irritation. The research aimes at creating a prototype of a mobile sensor system that can be used to analyze the particulate matter pollution on a location and on a time scale to measure the degree of pollution in the city. The engineering requirement to construct the edge device includes temperature (humidity) sensor, particulate matter sensor, GPS module, and an LCD for displaying the current sensor values. The health data of the mobile edge device can be analyzed through a developed analytics system, which allows the user to identify and avoid pollution sources. For the implementation of the web service the framework ReactJS, NodeJS with Express.js, and the database MongoDB are being used. The mHelath service is evaluated through field trials: New Year's Eve, various source identifications, and a demonstration through a journey from the subway station to the university. The paper outlines a mHealth service, which can collect data records of the surroundings and analyze the particulate matter in an information system to visualize risk locations of a user.**

***Keywords*— Mobile Health Service, Particulate Matter, Distribution, Raspberry Pi, ReactJS, NodeJS, Nova PM sensor SDS011, Neo-6M GPS**

I. Introduction

Particulate matter are extremely small particles in the air which, when inhaled, they can cause serious health effects. From 2007 to 2015, around 45.000 premature death cases have been traced back to this issue. Among others, the sources of particulate matter are waste burning, open fires, cigarettes and cars. In this paper, we want to investigate the latter. Due to the amount of cars in big capitals such as Berlin, urban citizens are exposed to remarkable amounts of particulate matter pollution.

In an effort to measure the degree of pollution in the city, we want to create a prototype of a mobile sensor system that can be used to analyze the particulate matter pollution in both a location and on a time scale in Berlin.

II. Outline

In the following paper, the research is structured into the following parts: The sections section III briefly introduces the current research literature, In section IV the setup and configuration of the hardware requirements of the device is described. The section V (Spatial Analysis and Source Identification) focuses on testing the prototype in various different conditions in Berlin. Among other reasons, the field trials were crucially carried out to distinguish how well the device works in general and whether the sensors are accurate enough for our application. The successful field trials justified further development of an Information System to analyze the health data. The implementation is documented in section VI detail in including a walk-through analytics procedure. The findings are summarized in section VIII and section VII.

III. Theoretical Background

For the thematic classification of the present work, the state of the art research, related methods, and concepts of relevant scientific papers are briefly introduced. For the selection of the particulate matter sensor the research is based on the performance assessment of [1]. The paper "Air pollution and particulate matter detector using raspberry Pi with IoT based notification" [2] aims to outline the design to create a prototype to detect the air pollution of particulate matter using a Raspberry Pi. The hardware setup and an approach with notifying the user by e-mail is introduces. The device is large and does not provide offline or mobile capabilities.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 11, Issue 06, June 2021)

Leaking a sensor for identifying the clients location. The paper on "Low Cost, Multi-Pollutant Sensing System Using Raspberry Pi for Indoor Air Quality Monitoring" [3] demonstrates the construction of a monitoring system including a detailed building block diagram, see Figure 2, which is from great advantage while build the system.

Conducting first experiments and building a prototype of a portable device to measure the particulate matter is content of the paper "Hotspot identification with portable low-cost particulate matter sensor" [4]. From advantage is the indication of the measurement route and the proposed way of showing the recorded data. However, the solution does not provide real-time visualization of the PM values. A new approach is an interactive system to analyze the risk areas based on an interchangeable edge device.

To categorize air quality, governments are issuing Air Quality Index tables, categorizing PM-value ranges into different indices. Recommendations for citizens are made, as seen in TABLE I based on [5, 6]

TABLE I
AIR QUALITY INDEX (AQI)

Index	$PM_{2.5}$	PM_{10}	Health messages for General population
1	0-11	0-16	Enjoy your usual outdoor activities.
2	12-23	17-33	
3	24-35	34-50	
4	36-41	51-58	Enjoy your usual outdoor activities.
5	42-47	59-66	
6	48-53	67-75	
7	54-58	76-83	Anyone experiencing discomfort such as sore eyes, cough or sore throat should consider reducing activity, particularly outdoors.
8	59-64	84-91	
9	65-70	92-100	
10	≥71	≥101	Reduce physical exertion, particularly outdoors, especially if you experience symptoms such as cough or sore throat.

IV. MOBILE HEALTH EDGE DEVICE

The aim is to create a device that utilizes a range of sensors and components. The measuring device should record sensor data and combine it into a timestamp before writing the timestamps into a csv-logfile. Some of the libraries utilized to run multiple sensors simultaneously had the impact of remarkably slowing down, therefore the Raspberry Pi, multithreading was implemented to make use of the four cores of the processor.

A. Engineering Requirements

A clear concept and design is essential when creating an edge device for an mHealth service. shows an overview of the necessary hardware components.

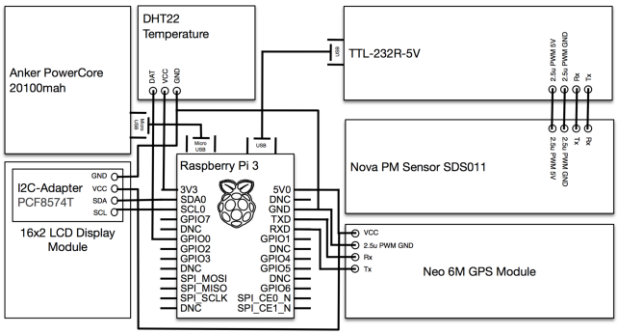

Figure 1. Hardware architecture

B. Temperature sensor

To measure temperature and humidity, we are using the "Adafruit DHT22" sensor. In our prototype setup, it is affixed outside the enclosure next to the GPS antenna to be able to record the environment. The sensor is connected to pin 11 (GPIO 17) in addition to 3.3 volts and ground. The "Adafruit _DHT" library is used and in the code responsible for reading the temperature and humidity values is shown.

```
def temperatureWork():
global currentTemp
global currentHum
while True:
    sensor = Adafruit_DHT.DHT22
    pin = '17'
    humidity, temperature =
     Adafruit_DHT.read_retry(sensor, pin)
    if humidity is not None and
    temperature is not None:
      currentTemp = "%.2f" % temperature
      currentHum = "%.2f" % humidity
  else:
      print('Failed reading. Try again!')
```

Listing 1. Temperature and humidity python code

The engineering of the code is based on [7] and [8]. Modifcation are made in regards of functional extensions and threading possibilities to increase performance.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 11, Issue 06, June 2021)

C. Particulate Matter Sensor

For measuring the particulate matter values the "Nova PM Sensor SDS011" is used. It comes with an adaptor converting its 7-pin interface into a USB-interface that can be connected directly to the Raspberry Pi. A predefned set of functions uses the given commands to instruct the sensor. In line 14 of **Listing 4** the concatenation of the PM-values to the csv-row can be seen. The sensor returns PM2.5 and PM10 values, which represent measured particulate matter distribution per cubic meter. PM2.5 particles are particles smaller than 2.5 micrometers in diameter, whereas PM10 particles are smaller than 10 micrometers [9]. Therefore a measured PM2.5 value of 50 means that within one cubic meter, approximately 50 particles smaller than 2.5 micrometers can be found.

D. GPS Component

The identification of the current location with longitude, latitude and altitude is implemented through the NEO- M GPS module. This component is connected as shown in Figure 1 and allows us to access the geographical information by reading the serial interface on RX and TX. For this communication to work the dependencies pynmea2, gpsd, gpsd-clients, python-gps and minicom need to be installed using the python pip package manager. The configuration /boot/cmdline.txt of the Raspberry Pi also must be changed. The line console=ttyAMA0,115200 will not be used anymore and it needs to be commented out. With the default configuration of the module demonstrated in **Listing 2**, which must be added to the /boot/confg.txt file of the Raspberry Pi. The necessary baud rate and frequency is automatically included during the boot up process.

```
1  dtparam = spi = on
2  dtoverlay = pi3-disable-bt
3  core_freq = 250
4  enable_uart = 1
5  force_turbo = 1
6  init_uart_baud = 9600
```

Listing 2. The boot confg file

The connection to the NEO-6M GPS [10] component can be tested via the Terminal prompt cgps -s or with the visualization of the data flow through cat /dev/ttyAMA0. With this setup the communication of the module can be implemented using only a few lines of python code, see [11] and [12, p.33].

Based on reason of consistency and the overview of the construction of the mHealth device the code is presented in **Listing 3**

```
1  import os, time, threading
2  from gps import *
3  from time import *
4  class GpsPoller(threading.Thread):
5    def __init__(self):
6      threading.Thread.__init__(self)
7      global gpsd #bring it in scope
8      gpsd = gps(mode=WATCH_ENABLE) #starting
   the stream of info
9      self.current_value = None
10     self.running = True #setting the thread
   running to true
11   def run(self):
12     global gpsd
13     while gpsp.running:
14       gpsd.next() #this will continue to
   loop   and grab EACH set of gpsd info to
   clear the buffer
15 if __name__ == '__main__':
16   gpsp = GpsPoller() # create the thread
17   gpsp.start() # start it up
18   print(gpsd.fix.latitude)
```

Listing 3. NEO-6M GPS python code

If these instructions for the configuration process of the NEO-6M GPS module are found not to be detailed enough for reproducing this setup, a more accurate documentation can be found on the GitHub repository [13] of the paper

E. LCD

An LCD is utilized to display the current sensor values. In this research, we are using a 2.6" 16x2 LCD equipped with an I2C adaptor. The I2C adaptor converts the LCD's 16-pin interface to the 4-pin I2C interface. It is connected to the Raspberry Pi's pin 3 (SDA) and pin5 (SCL) in addition to 5 volts and ground. The "I2C_LCD_driver" custom library is used. To integrate the library, one needs to follow these instructions:

1. Activate I2C on the Raspberry Pi based on [14, 15, 16]:
   - Run sudo apt-get install -y python-smbus
   - Run sudo apt-get install -y i2c-tools

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 11, Issue 06, June 2021)

- Run sudo raspi-confg
- Use the down arrow to select 5 Interfacing Options.
- Arrow down to P5 I2C.
- Select yes when it asks you to enable I2C.
- Also select yes if it asks about automatically loading
- the kernel module.
- Use the right arrow to select the Finish button.
- Select yes when it asks to reboot.

2. Download the I2C LCD driver [17]. Copy the script into the root folder with the sensorData.
3. Edit the I2C_LCD_driver.py file based on [18]:

- In line 18, change the I2CBUS value to 0, if your are using an original Pi, to 1 if your are using a newer version
- Run sudo i2cdetect -y 0 (respectively sudo i2cdetect -y 1 if using a newer version, again), to find the I2C address
- In line 21, change the ADDRESS value to the I2C address
- Save the file

The library can now be imported and used. In lines 24 to 29 of 4, the code sections responsible for displaying the sensor values on the LCD can be found. The visualization of the values is shown in **Figure 2** on the LCD.

Figure 2. Device for measuring particulate matter

F. Sensor Monitoring Agent

The script collecting information on particulate matter distribution consists of three steps. Firstly, the device must collect data from all the sensors. Secondly, this sensor data has to be composited into a well-formed csv-file entry. Finally, the most important values of the sensors are printed out on the LCD. These components and their interaction combines to create a well-functioning device, which reliably monitors the particulate mater distribution, see **Figure 2**. To fulfill the purpose of near real time recording it is necessary to collect all the sensor data simultaneously. This can be achieved by utilizing an agent which runs in the background, communicating with the connected sensors and and transferring the data to a cvs-report-file. This daemon is also responsible for displaying the different particulate matter values on the display. The procedure of the development is outlined in the subsection G. The focus of this section is the detailed description and explanation of the implementation. Also, in subsection H. an example is given to demonstrate the output of the sensors showing the format of the data recording service.

G. Collecting sensor data

The implementation of the Sensor Monitoring Agent combines the four components explained in Figure 1. The code presented in **Listing 4** shows a short and abstract preview of the implementation accessing all sensors simultaneously; it prints the sensor data to the LCD and creates a report csv-file. Lines 1 to 26 contain the thread responsible for accessing the sensors. In lines 27 to 31, the main method for the creation of the different threads can be found.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 11, Issue 06, June 2021)

```
1  def mainThreadWork():
2    rbStart = datetime.datetime.now()
3     .strftime("%Y-%m-%d %H:%M:%S")
4    fileName = "/home/pi/Recording" +
           rbStart + ".csv"
5    heading = ["PM2.5","PM10",
6    "Temperature","Humidity","Latitude",
7    "Longitude","Altitude","Time"]
8      with open(fileName, "w") as csvFile:
9        writer = csv.writer(csvFile)
10       writer.writerow(heading)
11       while True:
12         gpsdLatitude = gpsd.fix.latitude
13         gpsdLongitude = gpsd.fix.longitude
14         gpsdAltitude = gpsd.fix.altitude
15         row = [ ValuesOfPM[0],
16         ValuesOfPM[1], currentTemp,
17         currentHum, gpsdLatitude,
18         gpsdLongitude, gpsdAltitude,
19         datetime.datetime.now() ];
20         mylcd.lcd_display_string("2.5="
21         + str(ValuesOfPM[0])
22         + " 10="+str(ValuesOfPM[1]), 1)
23         mylcd.lcd_display_string("Lat="
24         + str(gpsdLatitude)
25         + " Log="+str(gpsdLongitude), 2)
26         writer.writerow(row)
27 if __name__ == "__main__":
28   GpsPoller().start()
29   thr.Thread(target=pmValues).start()
30   thr.Thread(target=tempWork).start()
31   thr.Thread(target=mainWork).start()
```

Listing 4. Sensor monitoring agent code

The main working thread starts by creating a csv file in the home directory of the Raspberry Pi ("/home/pi/"). The agent is adding a new csv-file to this directory named after the current time stamp. Afterwards the table header is added as the first line of the file, followed by the continuous writing process of the dataflow from the sensors shown in lines 12 to 26. In lines 20 to 25, the particulate matter values and gps coordinates are written on the LCD. In order to reduce the complexity of the listing, this section did not deal with the already stated collection of sensor data.

H. Data recording format

The recorded collected sensor data is stored in the format presented in TABLE II. The file uses the MS-DOS comma separated values format (csv-file). This information is essential when importing the data into Microsoft Excel and visualizing it as a diagram

TABLE II
TWO ROWS EXAMPLE OF THE DATA RECORDING FORMAT

PM2.5	PM10	Temp.	Humidity	Lat.	Long.	Alt.	Time
9.7	15.2	7.90	80.50	52.511995	13.451651	106.3	2018-12-08 17:05:44.39
9.9	16.1	8.00	80.20	52.511995	13.451656	106.3	2018-12-08 17:05:44.92

Crucially, the time stamp in the last column of TABLE II depends on the time zone of the operation system in our case the time zone of the Raspberry Pi. The zone can be verified using the 'date' command. If the time zone does not match the expected time, the time zone can be changed with the command 'dpkg-reconfgure tzdata'.

V. SPATIAL ANALYSIS AND SOURCE IDENTIFICATION

The measuring device was built using a reasonably sized powerbank and a small general form factor, allowing the setup ft into a small enclosure. The scripts enable the edge device to record sensor data and to log health data as soon as it is connected to a power source. This was done with the reason to enable the user to portably manage the device in most conditions, without the need of a connected computer to record data.

The device was tested in various situations around the city of Berlin to disclose the particulate matter distribution. The test situations were chosen to represent a variety of circumstances including time of the day, weather conditions, location and special occasions (in this case NYE in Kreuzberg).

A. Straße des 17. Juni

At Tiergarten, close to TU Berlin, a field trial was made during a dry day. For this test setup, two recording devices were placed on opposite sides of the street. The camera filming the situation was placed in the middle of the street, in between the opposing lanes and overlooking the intersection. Two measuring stations are used in this experiment in an attempt to eliminate the possibility of external factors distorting the recorded particulate matter distributions.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 11, Issue 06, June 2021)

Figure 3. Straße des 17. Juni field trial (video screenshot)

In the video, a bus halting at the traffic light aside one of the measuring stations can be seen. A significant rise in PM distribution next to it can be observed, while on the other side of the street, the other device does not record such a drastic development. In this field trial, the max values for PM2.5 and PM10 where 3 and 15 particles per m^3.

B. Warschauerstraße

A field trial was conducted next to a high traffic street in Friedrichshain at nighttime and in rainy conditions, see **Figure 4**

Figure 4. Warschauerstraße field trial (video screenshot)

The video was taken from a distance overviewing the situation from a balcony of one of the team members. The measuring station is located next to the traffic light in the bottom right corner of the frame (pink umbrella). While watching the video, one can observe PM-values rising once a group of cars approaches and waits in front of the traffic light, before falling once the cars have left the intersection. In this field trial, the max values for PM2.5 and PM10 where 13.1 and 26 particles per m^3.

C. New year's Eve

During New Years Eve 2019/2020 a field trial was conducted to investigate the extent of particulate matter distribution caused by fireworks. For this experiment, the measuring device was placed on a balcony in Kreuzberg

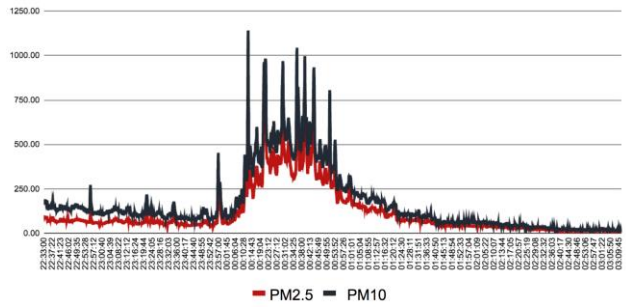

Figure 5. Kreuzberg New Year's Eve recording

The figure reveals tremendous levels of PM-distribution, which are explained in subsection C. At its peak, the PM10-levels are exceeding a value of 1100, while the government is classifying values over 100 into the most dangerous category, advising citizens to reduce outdoor activities (as seen in TABLE I).

VI. Health Information Analytics System

The system is divided into two sections; the technology stack, which describes the theoretical background of each component, and then the implementation of the web-service with the chosen technologies.

A. Technology Architecture

The architecture visualization in **Figure 6** was designed to demonstrate the data flow between, and the requirements of, each component. This allows for a breakdown into distinguishable parts

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 11, Issue 06, June 2021)

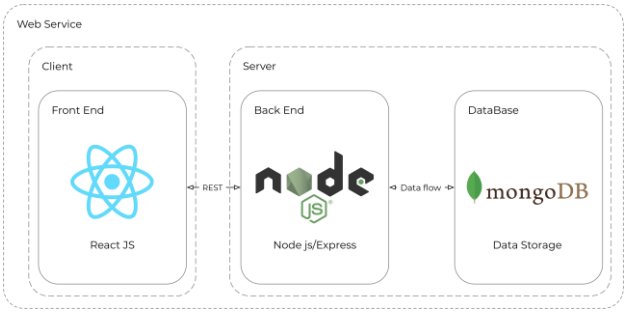

Figure 6. Technology stack architecture

The architecture consists of three main components; the client with the React front end, the server with the Node.js back end and the database MongoDB, which allows to store data for our application [19]. The following sections are outline each component, which is crucial as the components are mandatory for the implementation.

B. React Frontend

React is an open-source JavaScript library, founded and maintained by Facebook, which is used for building interactive user interfaces [20]. Single page services are a particularly popular application of React. It's main advantage is live compiling, without reloading the web page, and the creation of reusable UI components. React allows the developer to create large web applications that are fast, scalable, and simple. The system backing it uses the Model View Controller (MVC) pattern for organizing and structuring the code.

C. Node.js Backend

Node.js is an application runtime server-side JavaScript environment [21]. It is efficient, lightweight and by enabling the usage of JavaScript it is easy and fast for developers to write applications. In unity with the Express.js web application framework it is a great solution for building networking services and applications [22].

D. React Graphical User Interface Logic

When designing a front end, firstly certain guidelines have to be considered. Firstly, Adobe Colors Collection is used for the color hex codes and the icons are sourced from the icon database "The Noun Project" [23]. The second step is to evaluate a css framework. Among others, there are Bootstrap, Material Design Lite and Foundation. In our case, a lightweight and simple framework with an elegant design made sense, hence UiKit is used, a modular front end framework for developing fast and powerful web interfaces [24]. The focus in the research is on the more important parts; the upload functionality and the visualization of the measured values. The rendering of the uploaded csv and converted GeoJSON will be discussed in detail.

Firstly, for managing the upload functionality in React, we use the dependency FilePond. This resource allows us to include a simple drag and drop upload area into the web page within the React environment. The JavaScript Code presented in **Listing 5** shows the basic implementation of the required settings to allow csv file upload.

```
import ...
var styleUpload = {...};
class Upload extends Component {
 render() { return (
  <div style={styleUpload}>
  <FilePond
    onprocessfile={handlePondFile}
    allowMultiple maxFiles={300}
    name="images"
    server="{sH}:{sP}/api/image"/>);}
} export default Upload;
```

Listing 5. React upload code snippet

Another important part of the front end is the rendering of the map. For this purpose Google Maps is used and included into React. Every position and its corresponding measured PM-values for the recorded csv-file in the GeoJSON will be rendered onto the map. Therefore, the code demonstrated in **Listing 6** is utilized to fulfill this requirement.

```
import ...
class Map extends Component {
 componentDidMount() {
 const google = window.google
 var latestfile =
fetch("http://{sH}:{sP}/api/reports")
 .then(res => res.text())
 .then((result) =>
fetch("http://{sH}:{sP}/recordings/"+res
ult))
 .then(res => res.json())
 .then((result) => { initMap();
  for (var i = 0; i <
result.features.length; i++) {
   pm10 += result.features[i]
```

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 11, Issue 06, June 2021)

```
12      .properties.PM10
13    pm2 += result.features[i]
14      .properties.PM2[5]
15    coords =
result.features[i].geometry.coordinates;
16    latLng = new
google.maps.LatLng(coords[1],coords[0]);
17    if (typeof
result.features[i].properties.PM2[5] !=
undefined){
18      color =
getColorCodePM10(result.features[i].prop
erties.PM10);
19    } else {
20      color = getColorCodePM10(5);
21    }
22    var circle = new google.maps.Circle({
23      strokeColor: color, map: map,
24      center: latLng, radius: 2});
25    };})
26  function initMap() {
27   map = new google.maps.Map(
28    document.getElementById('map'),
29    {zoom: 13, styles: exampleMapStyles,
30      mapTypeId: 'terrain'}
31   );
32  }
33  function
getColorCodePM10(currentValue){...}
34  function
getColorCodePM25(currentValue){...}
35  const exampleMapStyles = [...]
36 }
37 render() {return(<div id="map"
style={styleMap}/>);}
38 };
39 export default Map;
```

Listing 6. React map rendering code

Lines 5 to 9 show the fetching of the last uploaded GeoJSON fle name and getting its content. Afterwards for each objects represented in the fle, the location and its values are added to the map (lines 11 to 25). The colorCodes, according to the TABLE I are fetched using the getColorCode functions in lines 17 and 21. These functions return the corresponding hex color code depending on the value of the particulate matter. After the coordinates are added to the map, the map component is returned, see lines 22 to 25.

E. Express.js Server Logic

For the processing of uploaded csv-files, a server side application is needed to store, convert and analyse the recordings. In the frond end section we implemented the component "FilePond", which lets the user drag and drop files to upload. This only provides us with the browser client interface. In fact the code presented in **Listing 7** shows the basic implementation of the required upload functionality.

This code not only stores the file, it also converts the csv-file to JSONobjects. When analysing and visualising geolocations in a map JSON formated objects are not the best approach. For this purpose GeoJSON objects are more efficient. Line 17 shows the conversion of json to GeoJSON with the modified parameters.

```
1  app.post("/api/image",
2    upload.array("images", 12),
3    function(req, res, next) {
4    res.send([req.files[0].filename]);
5    asyncCall(req.files[0].path);
6  });
7  // convert .csv -> .Json -> .GeoJSON
8  async function asyncCall(filepath) {
9    ...
10   const wt =
fs.createWriteStream(filepath +
'.geojson',
11     {flags: 'a'});
12   const pasedGeoJson = GeoJSON.parse(
13     await csv().fromFile(path),
14     {Point: ['Latitude', 'Longitude']});
15   wt.write(pasedGeoJson);
16   wt.close();
17 }
18 app.get('/api/reports', function(req,
res) {
19     var f = getLatestFile("{{path}}")
20     res.send(f);
21 });
22 function getLatestFile(dirpath) {...}
23 ...
```

Listing 7. Express.js server logic

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 11, Issue 06, June 2021)

The lines from 6 to 12 are showing the http post requests for handling the upload of files from the FilePond instance. The function (lines 14 to 24) for converting the csv to JSON and JSON to GeoJSON is being executed asynchronously. This function stores the csv-file on the server with line 15 and then converts this file to GeoJSON fle (in lines 16 to 23). Another important part is accessing the converted GeoJSON file on the server. This is done by sending a http get request on the '/api/reports' endpoint implemented in lines 27 to 30. This endpoint returns the last uploaded and converted GeoJSON file name.

F. Analytic Procedure

In this section the main focus is the user interaction with the front end. The starting point of the user interaction is the upload page. Here, one can chose or drag&drop a file into the browser window. The user can upload as many files as he likes. Keep in mind that for the analysation and visualisation, on the overview page, the last file uploaded by the user is used.

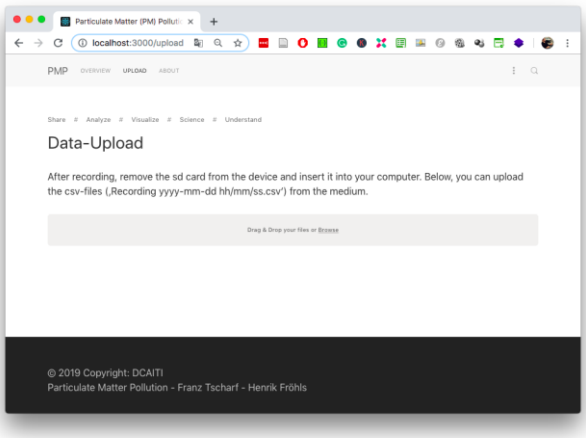

Figure 7. File upload page

If the user drag&drops a file into the browser and uploads it to the server, the overview page can be opened. Directly information about the journey (recorded csv-file) is shown (**Figure 8**). The time, start gps point, end gps point and the total consumption of particulate matter smaller than 2.5 and 10 micrometers are shown on the page.

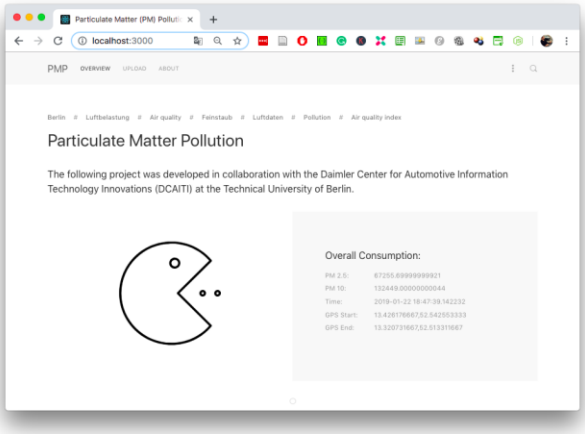

Figure 8. Overview page

The most important feature of this service is the visualisation of the geolocation on the map. When scrolling down on the overview page, a rendered map can be found (**Figure 9**). The displayed route consists of green, yellow and red parts. The colors are bases on the AQI indices depending on the recorded particulate matter distribution. The colors are explained in TABLE I.

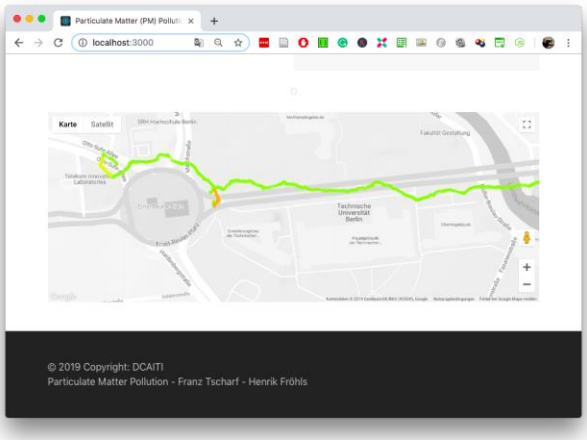

Figure 9. Map of the particulate matter distribution

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 11, Issue 06, June 2021)

In conclusion the service is very usable in the health care client domain to analyze potential risk areas in relation to particulate matter pollution. In **Figure 9** the red dots are visible indicating a hight pollution of smaller than 2.5 micrometer thick particular matter. Through these findings the user is able to avoid those location and increases his well being.

VII. Discussion

The use of cheap sensors was not the optimal solution (the GPS module could be updated). The implementation of map rendering GeoJSON objects is resource consuming and can take longer on the client side than expected, depending on setup, file size and network quality. This is extremely important for the user experience. Building a web service fully JavaScript based was a challenging approach. If work on this project is continued, there are a few additional features, which could be implemented. Firstly, more information could be displayed on the map, e.g. data of every recorded point could be analyzed further and displayed. Also, the possibility for the service to be accessible for more users with individual profiles and deployment on the web. Last the mHealth Service would be from great advantage in context of extending or replacing the current particular matter sensor with a thermal camera to detection covid-19 risks. The increased global awareness in our society when considering the hazards of particulate matter pollution could be enhanced by the service.

VIII. Conclusion

In conclusion this project was an interesting and challenging task; particularly the construction of a hardware device and collecting records of our surroundings was a joyful experience. The processing of the data and development of a mHealth service displaying the health data was a challenging yet interesting task. The developed edge device is a working prototype and we were able to use it in a productive environment. We are glad that all requirements were identified in the beginning could be met and even exceeded.

Acknowledgments

The research for this paper was financially supported by Exprivia S.p.A, University of Trento (UniTrento), Fondazione Bruno Kessler (FBK) and the European Institute of Innovation Technology (EIT).

Additionally, I like to thank Henrik Froels who contributed to the implementation of our proofof-concept prototype within the scope of the project

References

[1] Hai-Ying Liu, Philipp Schneider, Rolf Haugen, and Matthias Vogt. 2019. Performance assessment of a lowcost PM2.5 sensor for a near four-month period in oslo, norway. *Atmosphere*, 10, 2, (January 22, 2019), 41. issn: 2073-4433. doi: 10.3390/atmos10020041. Retrieved 05/04/2021 from https://www.mdpi.com/2073-4433/10/ 2/41 Ding, W. and Marchionini, G. 1997 A Study on Video Browsing Strategies. Technical Report. University of Maryland at College Park.

[2] Meo Vincent C. Caya, Angeline P. Babila, Alyssa Moya M. Bais, Seoi Jin V. Im, and Rafael Maramba. 2017. Air pollution and particulate matter detector using raspberry pi with IoT based notifcation. In *2017IEEE 9th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*. 2017IEEE 9th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM). (December 2017), 1–4. doi: 10.1109/HNICEM.2017. 8269490

[3] He Zhang, Ravi Srinivasan, and Vikram Ganesan. 2021. Low cost, multi-pollutant sensing system using raspberry pi for indoor air quality monitoring. *Sustainability*, 13, 1, (January 3, 2021), 370. issn: 2071-1050. doi: 10.3390/su13010370. Retrieved 05/04/2021 from https://www.mdpi.com/2071-050/13/1/370.

[4] Csongor Báthory and Arpad Bence Palotas. 2019. Hotspot identifcation with portable low-cost particulate matter sensor, (March 2019), 6

[5] Sef van den Elshout, Bartelds Hans, Heich Hermann, and Léger Karine. 2008. Common Information to European Air, 38. Retrieved 03/20/2021 from https://www. airqualitynow.eu/download/CITEAIR-Comparing_Urban_Air_Qality_across_Borders.pdf.

[6] 2021. Air quality now - about US - indices defnition. Retrieved 05/04/2021 from https://www.airqualitynow. eu/about_indices_definition.php.

[7] 2019. DHT22 – xblog. (August 2019). Retrieved 05/31/2021 from https://www.3gcomet.com/dht

[8] lady ada, Tony DiCola, and Kattni Rembor. 2012. DHT11, DHT22 and AM2302 sensors. Adafruit Learning System. (July 2012). Retrieved 05/31/2021 from https:// learn.adafruit.com/dht/overview.

[9] OAR US EPA. 2016. Particulate matter (PM) basics. US EPA. (April 19, 2016). Retrieved 05/31/2021 from https ://www. epa . gov/pm- pollution/particulatemater-pm-basics.

[10] 2015. NEO-6 series. u-blox. (June 25, 2015). Retrieved 05/04/2021 from https://www.u-blox.com/en/product/neo-6-series.

[11] Dan Mandle. 2012. Getting GPSd to work with python and threading. Things I've Figured Out. (September 6, 2012). Retrieved 05/31/2021 from https://www.danmandle.com/blog/geting-gpsd-to-work-with-python/.

[12] Saad Benrouyne. 2015. CAR TRACKING ANTI-THEFT SYSTEM. (April 1, 2015). Retrieved 05/31/2021 from https://www.aui.ma/sse-capstone-repository/pdf/CAR-RACKING-ANTI-THEFT-SYSTEM.pdf.

[13] Viktor Walter-Tscharf. 2021. FranzTscharf/python-NEO-6m-GPS-raspberry-pi. original-date: 2018-11-16T16:58:33Z. (January 11, 2021). Retrieved 05/04/2021 from https://github.com/FranzTscharf/Python-NEO-6M-GPSRaspberry-Pi.

[14] Shawn Hymel. 2015. Raspberry pi SPI and i2c tutorial - learn.sparkfun.com. (October 29, 2015). Retrieved 05/31/2021 from https://learn.sparkfun.com/tutorials/raspberry-pi-spi-and-i2c-tutorial/i2c-on-pi.

[15] 2020. HuskyLens is an easy-to-use AI machine vision sensor. SEN0336. (June 15, 2020). https://wiki.dfrobot.com/HUSKYLENS_V1.0_SKU_SEN0305_SEN0336.

[16] 2015. Using i2c with SMBus and raspbian linux on the raspberry pi. AB Electronics UK. In collaboration with AB Electronics. (October 2, 2015). Retrieved 05/31/2021 from https://www.abelectronics.co.uk/kb/article/1/i2c-part-2---enabling-i-c-on-the-raspberry-pi.

[17] Denis Pleic. 2021. Raspberrypi i2c lcd python driver. Gist. Retrieved 05/05/2021 from https://gist.github.com/DenisFromHR/cc863375a6e19dce359d.

[18] Ada Lady. 2012. Adding a real time clock to raspberry pi. Adafruit Learning System. (August 31, 2012). Retrieved 05/31/2021 from https://learn.adafruit.com/adding-a-real-time-clock-to-raspberry-pi/set-upand-test-i2c.

[19] 2021. Die beliebteste datenbank für moderne apps. MongoDB. Retrieved 05/04/2021 from https://www.mongodb.com/de.

[20] 2021. React – a JavaScript library for building user interfaces. Retrieved 05/04/2021 from https://reactjs.org/

[21] Node.js. 2021. Node.js. Node.js. Retrieved 05/04/2021 from https://nodejs.org/en/.

[22] 2021. Express - node.js-framework von webanwendungen. Retrieved 05/04/2021 from https://expressjs.com/de/.

[23] 2021. Noun project: free icons & stock photos for everything. Retrieved 05/05/2021 from https://thenounproject.com/.

[24] 2021. UIkit Retrieved 05/05/2021 from https://getuikit.com/.