**AIR QUALITY MONITORING**

This paper is organised around the problem of environmental monitoring and ambient environmental monitoring, as parts of social computing and Internet of Things areas of research. In recent times, the use of sensing devices and technologies has had increasing significant interest in many fields, some remarkable results being achieved, particularly with evaluating, monitoring and analysing the air quality. The main purpose of the system developed in our project is to advance a way to monitor the air quality in a given room, providing the possibility of viewing history or even real-time data via a web service. Although the level of hardware is abstracted, the focus is on the problem that it addresses and on solving it, as well.

Monitoring air quality is crucial for public health and environmental protection. Setting up an air quality monitoring sensor involves several steps:

**1.Research and Planning:** Understand the specific pollutants you want to monitor, the regulatory standards, and the local environmental conditions.

**2.Selecting the Sensor:** Choose a reliable and accurate air quality sensor that can detect pollutants like particulate matter (PM), volatile organic compounds (VOCs), carbon monoxide (CO), nitrogen dioxide (NO2), and ozone (O3).

**3.Calibration:** Calibrate the sensor according to the manufacturer’s instructions to ensure accurate readings. This step is essential for maintaining the sensor’s precision.

**4.Installation:** Place the sensor in an area representative of the air quality you want to monitor, considering factors such as airflow, pollution sources, and accessibility for maintenance.

**5.Connectivity and Power Supply:** Ensure the sensor is connected to a stable power source and a data collection system, such as a microcontroller or a computer, to record and analyze the data.

**6.Data Management and Analysis:** Implement a data management system to store, process, and analyze the collected data. Utilize software or programming skills to interpret the sensor’s readings.

**7.Regular Maintenance:** Regularly inspect and clean the sensor to avoid contamination or malfunctions. Follow the manufacturer’s guidelines for maintenance and troubleshoot any issues promp Ensure.

**8. Data Reporting and Communication:** Make the collected data accessible to stakeholders, policymakers, and the public. Utilize visualizations and clear communication to ensure that the data is easily understandable and actionable.

**9.Compliance and Regulations:** Ensure that the data collected complies with local regulations and standards. If discrepancies arise, take necessary actions to rectify them and maintain data accuracy.

**10.Continuous Improvement:** Stay updated with the latest advancements in air quality monitoring technology and continuously improve the monitoring system to enhance its accuracy and reliability.

***System Development:***

Considering the speciﬁc development topics, our main interests focused on,

(i)Implementing a web-socket server ready to provide the platform structure and real-time data for all its clients.

(ii) creating an HTML pattern which has to encapsulate the web platform structure.

(iii) assuring a real-time communication with the acquisition component, both for data receiving and for sending those data to each of the users’ accounts.

(iv) real-time viewing and processing data for each customer. A sum of speciﬁc details about the modern software technologies and programming languages involved in these system components development is mentioned below.

Both the main server and the device-attaching server are written in C++ with Qtopia as a modern inter-platform framework for software development. A controls-component library for creating both graphical and non-graphical user interfaces is widely used in this project. Other important Qt-based applications are the KDE desktop environment, the web browser Opera or Google Earth, Skype and Qtopia as well. For the speciﬁc interest of the servers and the acquisition system in ad-dressing the Raspberry Pi board pins using the GPIO support, the C-based library Wiring Pi is used.

***System Resources of communication*:**

Ours system’s main server aims to achieve communication between the device and the web client. Speciﬁcally, all data interchanged between the device and its assigned applications must ﬁrst pass through this server. The presence of this server is particularly justiﬁed by the necessity of monitoring the data displayed by the device modules, both inside and outside of a network without any delay between responses due to limited resources. Another reason could be further expansion possibilities, as we will mention as future work.

Considering the software aspects, in making this component we used the following support: C++ as a programming language, as a framework for modern applications development, C Make to generate the solution valuing the advantages of a build system, Visual Studio for editing the code in a modern and up to date integrated development environment. For the remaining part of this section, we will describe the contributions of these technologies to developing our software components as servers, interfaces or managers. As the system workﬂow, when the main server starts, the ﬁrst step is to have all manager-applications successfully initialized.

This stage is highlighted by messages displayed in the console or through log ﬁles. After all the managers have successfully started, the server expects and is waiting for customers to connect. Once there is at least one connected customer, a message is received from him/her, and it is forwarded to the package manager, where the polymorphism and reﬂection principles will be used to sound the desired behaviour. These interpretations of incoming packets may result in database modiﬁcation, customer relationship creation, sending another package as a response to the sender, or sending another package to another customer.

***Server dedicated to attaching file:***

The speciﬁc server for involving the devices into the system is running entirelyon a Raspberry board, as compared to the central communication server whichhas to run on a specialized computer. In addition to its responsibilities alreadypresented in the previous section, this server can also handle for attaching a device to an aﬃliate application.

For the current state of our system, the web application Can be considered as being aﬃliated, meaning that once the user has logged in, (s)he needs to attach a device. For this to be feasible, the dedicated attachment Server must communicate directly with an operational button connected to the Raspberry hardware. In the step of attaching a device to a user account, the web application searches for and sends a package to the Raspberry board every second for one minute. The Answer from it, if there is a stable connection, varies depending on the status of the button. When the button is pressed, the board sends a packet of data to the main server, and this inserts into the database a speciﬁc link between the account And the device. The web application will also be notiﬁed via another data packet and the interface will change accordingly, thus exposing the information provided by the modules connected to the Raspberry device.

***The web application:***

The web application is the only component that can interact directly with a potential user. The three stages in which a user can be, following the web application scenarios, are: unauthenticated user, authenticated user without any attached device or authenticated user with an attached device. The unauthenticated user has access only to the registration and authentication functions. The authenticated user without an attached device has access only to the button that starts the process of attaching the device.

The web platform of this project represents the component that is accessible to any potential client. Users that do not own this system will only have access to registration and login functions, the remainder of the shares being blocked by a message which asks for attaching a device to the user account. Customers who own the system have full privileges, meaning access to viewing real-time data sent by the system as well as history. Moreover, this type of users have access to various settings, be they related to the system itself or customization. Concerning data management, our web platform provides the ability to view real-time data. This functionality is present in two forms Namely as,

(1).Quick data viewing, meaning that the latest data received from each category can be viewed on any page, or

(2) detailed data viewing, which is available only on the specially designed page for this functionality by giving access to many of the data received from each category as a chart.

***Implementing the data acquisition component:***

From the operational point of view, our data acquisition system has eﬃciently accomplished at least these three functions:

(1) converting the physical behaviour into a measurable signal, usually an electric one;

(2) measuring the sensors’ signals and managing the actions for an entire information overview;

(3) data analysing and viewing in a useful and user-friendly interface.

Data acquisition systems are continually increasing their popularity in modern industry and even research, based on the interest in accurately collecting data as a premise of processed-data reliability.

Despite this, the acquisition and maintenance costs to this type of systems are rather high. Our present project aims to ﬁnd a solution for this problem by addressing a low-cost and high-performance data acquisition and processing system. During the data acquisition time, the speciﬁc threshold for the input data is tested and, in case of exceeding values, the system launches an emergency notiﬁcation.

Hence, the connected customers receive SMS notiﬁcations in case of abnormal values purchased from modules. A third-party server has been used for the SMS notiﬁcations, that is Twilio Notiﬁcation System (Twilio Notify). This service allows the application to send notiﬁcation messages to multiple users, over diﬀerent communication channels, all from the same uniﬁed API. With a single API request, can connect with contacts using SMS, mobile apps, Facebook Messenger, and more. The user can even specify preferred channels for reaching certain users, and tag them for more granular sending capabilities.

***Entire coding:***

Here, We can setup a local server to demonstrate its working. But to monitor the air quality from anywhere in the world, you need to forward the port 80 (used for HTTP or internet) to your local or private IP address (192.168\*) of you device. After port forwarding all the incoming connections will be forwarded to this local address and you can open above shown webpage by just entering the public IP address of your internet from anywhere. You can forward the port by logging into your router (192.168.1.1) and find the option to setup the port forwarding.

#include "MQ135.h"

#include <SoftwareSerial.h>

#define DEBUG true

SoftwareSerial esp8266(9,10); // This makes pin 9 of Arduino as RX pin and pin 10 of Arduino as the TX pin

const int sensorPin= 0;

int air\_quality;

#include <LiquidCrystal.h> 

LiquidCrystal lcd(12,11, 5, 4, 3, 2);

void setup() {

pinMode(8, OUTPUT);

lcd.begin(16,2);

lcd.setCursor (0,0);

lcd.print ("circuitdigest ");

lcd.setCursor (0,1);

lcd.print ("Sensor Warming ");

delay(1000);

Serial.begin(115200);

esp8266.begin(115200); // your esp's baud rate might be different

sendData("AT+RST\r\n",2000,DEBUG); // reset module

sendData("AT+CWMODE=2\r\n",1000,DEBUG); // configure as access point

sendData("AT+CIFSR\r\n",1000,DEBUG); // get ip address

sendData("AT+CIPMUair\_quality=1\r\n",1000,DEBUG); // configure for multiple connections

sendData("AT+CIPSERVER=1,80\r\n",1000,DEBUG); // turn on server on port 80

pinMode(sensorPin, INPUT);        //Gas sensor will be an input to the arduino

lcd.clear();

}

void loop()

{

MQ135 gasSensor = MQ135(A0);

float air\_quality = gasSensor.getPPM();

if(esp8266.available()) // check if the esp is sending a message 

{

if(esp8266.find("+IPD,"))

{

delay(1000);

 int connectionId = esp8266.read()-48; /\* We are subtracting 48 from the output because the read() function returns the ASCII decimal value and the first decimal number which is 0 starts at 48\*/ 

 String webpage = "<h1>IOT Air Pollution Monitoring System</h1>";

 webpage += "<p><h2>";   

 webpage+= " Air Quality is ";

 webpage+= air\_quality;

 webpage+=" PPM";

 webpage += "<p>";

 if (air\_quality<=1000)

{

  webpage+= "Fresh Air";

}

else if(air\_quality<=2000 && air\_quality>=1000)

{

  webpage+= "Poor Air";

}

else if (air\_quality>=2000 )

{

webpage+= "Danger! Move to Fresh Air";

}

webpage += "</h2></p></body>"; 

 String cipSend = "AT+CIPSEND=";

 cipSend += connectionId;

 cipSend += ",";

 cipSend +=webpage.length();

 CipSend +="\r\n";

 sendData(cipSend,1000,DEBUG);

sendData(webpage,1000,DEBUG);

 cipSend = "AT+CIPSEND=";

 cipSend += connectionId;

 cipSend += ",";

 cipSend +=webpage.length();

cipSend +="\r\n";

String closeCommand = "AT+CIPCLOSE="; 

closeCommand+=connectionId; // append connection id

closeCommand+="\r\n";

sendData(closeCommand,3000,DEBUG);

}

}

lcd.setCursor (0, 0);

lcd.print ("Air Quality is ");

lcd.print (air\_quality);

lcd.print (" PPM ");

lcd.setCursor (0,1);

if (air\_quality<=1000)

{

lcd.print("Fresh Air");

digitalWrite(8, LOW);

}

else if( air\_quality>=1000 && air\_quality<=2000 )

{

lcd.print("Poor Air, Open Windows");

digitalWrite(8, HIGH );

}

else if (air\_quality>=2000 )

{

lcd.print("Danger! Move to Fresh Air");

digitalWrite(8, HIGH);   // turn the LED on

}

lcd.scrollDisplayLeft();

delay(1000);

}

String sendData(String command, const int timeout, boolean debug)

{

String response = “”;

Esp8266.print(command); // send the read character to the esp8266

Long int time = millis();

While( (time+timeout) > millis())

{

While(esp8266.available())

{

// The esp has data so display its output to the serial window

Char c = esp8266.read(); // read the next character.

Response+=c;

}

}

If(debug)

{

Serial.print(response);

}

Return response;

}

***Conclusion***:

This project-based article presents both hardware and software considerations for developing an original system for analysing the air quality and sending notiﬁcations in case of emergency. The main target of this system was to oﬀer a low expensive way to facilitate monitoring real-time information about the quality of the air from a certain room and also to provide a history of the collected data. As a necessary step forward from a home to a smart home, this project is also a real solution for an industry monitoring system to have a comfortable climate or even to prevent working accidents. The leading features of our system are presented above. The inner web approach is provided by a high-performance web system.