ISSN 1846-6168 (Print), ISSN 1848-5588 (Online)
https://doi.org/10.31803/tg-20230226194209

Preliminary communication
Received: 2023-02-26, Accepted: 2023-03-24

Air Quality Supervision System using the IoT Platform

Marinko Stojkov, Goran Delija, Ivan Đuračić*, Tomislav Alinjak

Abstract: This paper presents the idea of the Internet of Things (IoT) platform, which is demonstrated by creating a device that measures air quality, collects and processes real-time data, and presents it in a web application. The device comprises a microcontroller board called Wemos D1 Mini, which is powered by an ESP8266 microchip. It also consists of two sensors, CCS811, which measures air particle quantity, and BME280, which measures air temperature, pressure, and humidity. The device is wirelessly connected to an MQTT broker using a Wi-Fi network and the MQTT protocol. The paper also provides detailed information on the functioning and usage of the MQTT protocol along with guidelines for programming and configuring the MQTT broker for both Wemos D1 and Raspberry Pi.

Keywords: air quality control; IoT; Raspberry Pi; web application; Wemos D1 Mini

1 INTRODUCTION

Over the past few years, the term *Internet of Things* (IoT) can often be heard in the media. Devices for which until recently it was unthinkable to be connected to the Internet, such as household appliances and cars, are now advertised by their manufacturers as *smart*, and it is difficult to find a product that does not have an IoT version [1, 2]. Significant coverage by high-speed mobile networks, as well as the price of electronic components, which has decreased to a very low level, have enabled enthusiasts and hobbyists to make their own versions of such smart devices at a very low cost. Low prices make it possible to turn almost any device into a part of the IoT, giving it a kind of digital intelligence and the possibility of automatic data exchange between two devices, the so-called M2M (*Machine to Machine*) communication, without the need for human intervention [3-6].

The *Internet of Things* was initially most interesting and useful to large companies and manufacturers, where it was primarily used for communication between devices (M2M) and for tracking expensive equipment and tools in production using RFID (*Radio Frequency identification*) tags. Recently, the focus has increasingly shifted to transforming our homes and offices into "*smart spaces*" [7- 8]. A device that performs trivial things such as turning off the lights and ordering groceries, gives the user more time to deal with more important things which contribute to stress reduction [9].

Industrial IoT architectures, which in recent years began to use the best data centers for data processing, have consequently received improvements in the form of increased production productivity, reduction of unplanned downtime in production as a result of failures, and reduction of electricity consumption [10-14].

In March 2019, the *Global mobile Suppliers Association* (GSA) announced that more than 100 mobile operators have NB-IoT (*Narrowband Internet of Things*) or LTE-M networks. By September 2019, this number had risen to 142 developed/started networks [10]. NB-IoT is best suited for IoT applications that require more frequent communication and data transmission because it operates on a licensed frequency spectrum and has no work cycle limitations [15, 16].

The basic idea for this work was to create an IoT sensor that will be able to run on batteries, so it will not have restrictions on the place of installation, and that will wirelessly send the measured values to a device connected to the Internet. Users are then able to check the sensor values at any time using a web application, for example using their smartphone.

The making of a sensor for air quality control, which also measures the amount of various particles in the air, was chosen. Such a sensor could have a wide application in mechanical engineering: companies engaged in, for example, metal processing could have insight into the air quality in their production halls at any time, in order to be sure of the safety of their employees. In the event that a process pollutes the air above a recommended level, the system could notify the responsible person, who could then take some action, such as ventilating the hall or removing workers from them.

As the main sensor unit, a development board based on the ESP8266 microcontroller, Wemos D1 Mini, was selected, to which the CCS811 air quality sensor and the BME280 atmospheric sensor were connected via the I^2C (*Inter – Integrated Circuit*) bus. A Raspberry Pi 3B minicomputer was chosen to receive data from the development board. A few years ago, the Raspberry Pi made a real revolution because despite its small dimensions (it is the size of a credit card) it contains all the parts like an ordinary personal computer, of course much weaker, but still with enough power to start and perform many tasks related to the use microcontrollers and sensors.

In the second chapter of this work, there is a description of the components of the air quality monitoring system, the setting of the MQTT protocol and the setting of the Python script to display the results. In the third chapter, the results of air quality measurements in real conditions are presented. The conclusion of this paper is given in chapter four.

2 DESCRIPTION

With the development of the IoT industry over the past few years, some new forms of wireless communication between devices have emerged. Some of the protocols for wireless communication were created on the basis of wireless technologies that were used many years before IoT as a

concept was even created, for example BLE (*Bluetooth Low Energy*) which arose from *Bluetooth Classic technology* that was primarily used to connect audio devices with computers or mobile phones.

The choice of protocol depends on the purpose of the IoT project. The basic selection criteria are consumption, range and bandwidth. Since IoT devices often do not have the ability to be powered by the power grid, they use batteries. In this case, it is important that the consumption of the device is as low as possible, so that its autonomy and service life are as long as possible. An autonomy of several years is often stated as a requirement for many IoT devices. As for the range, it can be from a few meters inside the building (e.g. Wi-Fi, BLE) to several kilometers in open space (e.g. LoRa – Long Range). Most often, as the range increases, the bandwidth, i.e. the amount of data sent by the IoT device, decreases, in order to keep the consumption as low as possible.

A brief description of the most commonly used wireless IoT protocols:

- **Wi-Fi** – the most common standard used in homes and businesses is 802.11n. It works on frequencies of 2.4 GHz and 5 GHz with a range of up to 50 meters. Data transfer speeds are a maximum of 600 Mbps depending on the channel frequency and the number of antennas (the latest 802.11-ac standard should offer 500 Mbps to 1 Gbps) [17],
- **NB-IoT** - *Narrowband Internet of Things* is a standard of radio technology, *Low Power Wide Area Network* (LPWAN), developed by 3GPP, the consortium in charge of network standards, for use in mobile devices and services [15, 16].
- **BLE** (*Bluetooth Low Energy*) – it is primarily intended as an upgrade of *Bluetooth Classic technology* for use in IoT devices. It is mostly used in devices in the health sector (constant control and measurements, for example: heart rate, glucose level, blood pressure, etc.), fitness, transmitters (beacon), security and home entertainment industry [18]. The bandwidth of BLE is usually in the range from 150 kbps to 1 Mbps [19],
- **LoRa** (*Long Range*) – it has the possibility of geolocation with a theoretical range greater than 10 km under ideal conditions [20].

2.1 Air Quality Supervising System Components

The device, the construction of which is explained in this paper, reads the quality of the air in the room, i.e. the proportion of carbon dioxide and the level of metal oxides, and at the same time measures the temperature, pressure and humidity of the air. CCS811 and BME280 sensors are used for the above and they represent the first layer of the IoT architecture. Wi-Fi was chosen as the communication technology. Data collection from sensors is done by Wemos D1 Mini, a development board based on the ESP8266 module, which sends data via Wi-Fi to a server built on a Raspberry Pi computer. Raspberry Pi contains a database in which sensor data is saved, as well as a *Web application* for displaying measurement results in text and graphic form.

The CCS811 (Fig. 1 and Fig. 2) was selected as the main sensor of the air quality control system. It is a very low-consumption digital gas sensor that detects a wide range of *Total Volatile Organic Compounds* (TVOC), including carbon dioxide equivalent (eCO_2) and metal oxides (MOX) levels. Volatile organic gases are often categorized as air pollutants and/or compounds that irritate the senses, and can appear as a result of evaporation from various building materials such as paints, as well as the result of the operation of photocopiers and the presence of people (e.g. breathing, smoking).

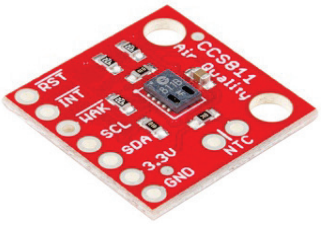

Figure 1 CCS811 sensor [21]

This sensor is intended for use in closed spaces, and is used in smart devices such as smartphones and smartwatches, or in home automation systems. This paper will use the sensor design on a separate printed circuit board, which uses a standard I^2C bus for communication. The unique I^2C address is $0{\times}5A_{HEX}$.

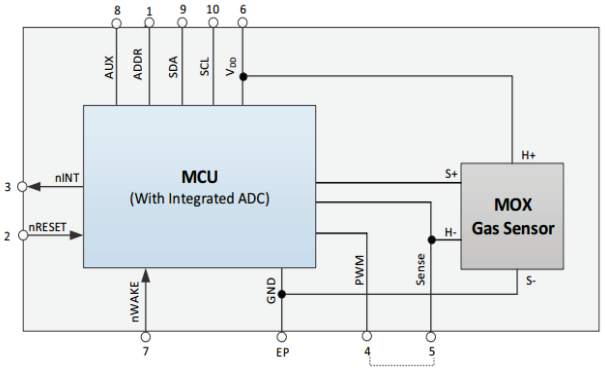

Figure 2 CCS811 Air quality sensor Block Diagram [22]

Tab. 1 shows the characteristics of the CCS811 sensor according to [23].

Table 1 Characteristics of the CCS811 sensor

Power supply:	1,8 V – 3,6 V
Current strength:	30 mA
Power Consumption:	60 mW (max)
Logic Level Voltage:	3,3 V – 5 V

I^2C is a type of communication bus that enables the connection of *master* (control) and *slave* (peripheral) devices. As already explained, it uses two lines for communication: *Serial Data Line* (SDA) for sending and receiving data and *Serial Clock Line* (SCL) for sending *clock*. Connecting multiple *slave* devices using the same lines is possible because each device has a unique address.

All peripherals listen to the SDA line and monitor whether its address has been sent. The *master* sends the first data packet containing the start bit and the corresponding address of the peripheral device with which it wishes to communicate. The peripheral device responds to the sent request for communication, after which the *master* starts sending data. In order to achieve communication with several *slave* devices, a *pull-up* resistor is used. *Pull-up* is the name for a resistor that has the task of "pulling" both SCL and SDA lines to 5 V, i.e. into the logic unit [24].

In order to simplify the programming process and avoid writing complex algorithms, the *Adafruit_CCS811* library is used for programming the CCS811 sensor, according to [25].

Figure 3 BME280 sensor [26]

In addition to the CCS811, a BME280 sensor is also used (Fig. 3 and Fig. 4). It is a combined sensor for measuring temperature, relative humidity and barometric air pressure. It has very small dimensions, low power consumption and is often used for home automation, in portable GPS modules or in smartwatches. It has high precision, so based on the pressure reading, it is possible to calculate the approximate altitude. As with the CCS811 sensor, a sensor design on a separate circuit board will be used. The standard I^{2}C bus is also used for communication. The unique I^{2}C address is $0{\times}76_{HEX}$.

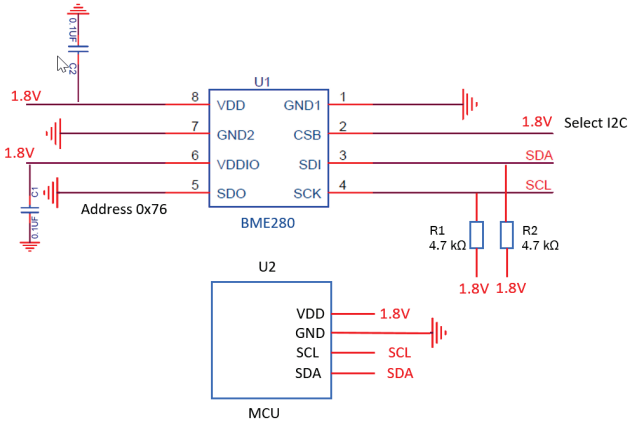

Figure 4 BME280 sensor electronics scheme [27]

The sensor characteristics are listed in Tab. 2 according to [28].

In order to simplify the programming process and avoid writing complex algorithms, the *Adafruit BME280* library is used to program the CCS811 sensor, according to [29].

Table 2 Characteristics of the BME280 sensor

Power supply:	1,71 V – 3,6 V
Current strength:	< 1 mA
Logic Level Voltage:	3,3 V – 5 V
Temperature range:	–40 °C - 85 °C (± 1 °C)
Humidity range:	0 - 100 % (± 3 %)
Pressure range:	300 hPa- 1100 hPa (± 1 hPa)
Altitude range:	0 m - 9000 m (± 1 m)

The ESP8266 (Fig. 5 and Fig. 6) is a very low-cost 32-bit Wi-Fi microcontroller from *Espressif Systems* that fully supports the TCP/IP protocol. It contains a microprocessor running at 80 MHz or 160 MHz clock. It has a memory of 32 KiB for management commands and 80 KiB for user data. The operating voltage and the logic level voltage are 3.3 V and have 16 input/output pins. It can be used as a microcontroller or as a separate Wi-Fi module to supplement another microcontroller. It supports the IEEE 802.11 b/g/n Wi-Fi protocol [30].

Figure 5 ESP8266 – 12E [31]

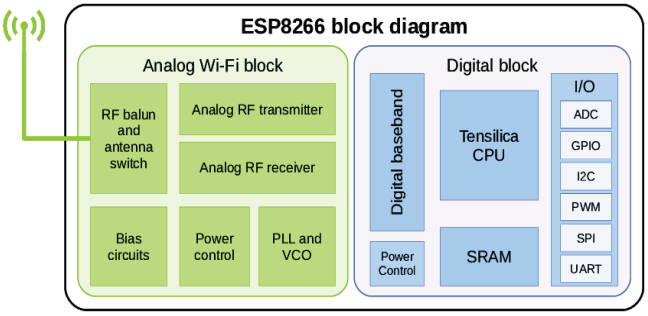

Figure 6 ESP8266 block diagram [32]

Since the ESP8266 microcontroller is intended for SMD (*Surface Mount Device*) soldering and operates at 3.3 V, in order to facilitate connection with a computer and simplify programming, in this work we use the Wemos D1 Mini development board (in the latest revision named Lolin). This board is based on the ESP-12 version of the ESP8266 chip and enables easy connection to a computer via a USB port and programming in the *Arduino IDE* development environment. Due to the combination of microcontroller and Wi-Fi connectivity, it is an excellent choice for demonstrating the capabilities of the IoT platform.

The processor with a frequency of 80 MHz on the Wemos board has at its disposal an increased memory capacity for saving program code, with a capacity of 1 MB, of which 0.8 MB is available, and 0.2 MB is reserved for the bootloader, and working memory with a capacity of 82 kB, of which 49 kB available to the user. Wemos D1 Mini has support for I^{2}C, *Serial Peripheral interface* (SPI) and serial

communication, contains nine digital input/output pins, all of which support *Pulse with modulation* (PWM) control and one analog input pin. In addition to all of the above, the board uses CH340 as a USB 2.0 interface, contains a 3.3 V regulator and four LED indicators. Before saving the program code, Wemos performs an automatic hardware reset [33].

In this work, the Wemos chip will collect data from the sensors and send them wirelessly to the Raspberry Pi via Wi-Fi, using the MQTT (*Message Queuing Telemtry Transport*) protocol. The Wemos board and his Pins are shown in Fig. 7 and Fig. 8.

Figure 7 Wemos (Lolin) D1 Mini development board [34]

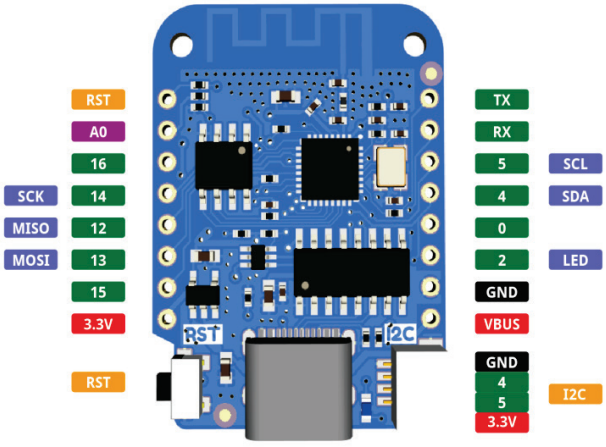

Figure 8 Wemos (Lolin) D1 Mini Pins [34]

Raspberry Pi is a minicomputer developed by "The Raspberry Pi Foundation", and it is designed as a cheap and accessible tool for hobbyists, enthusiasts, and for the needs of learning programming in schools, especially in less developed countries. It has very small dimensions, the size of a credit card, and all components are located on one circuit board. It contains ports and connectors for connecting an external power supply, mouse/keyboard, and video outputs for connecting a monitor. The operating system is stored on a microSD card, and in this work, the Raspbian operating system, based on Linux, is used.

Fig. 9 shows the connection scheme of sensors CCS811 and BME280 with Wemos D1 Mini using an experimental board.

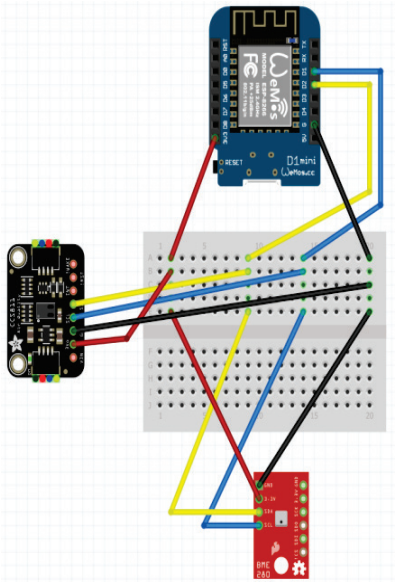

Figure 9 Scheme of connection of Wemos D1 and sensor in the *"Fritzing"* program

Fig. 10 shows the layout of the board with soldered sensors and Wemos D1.

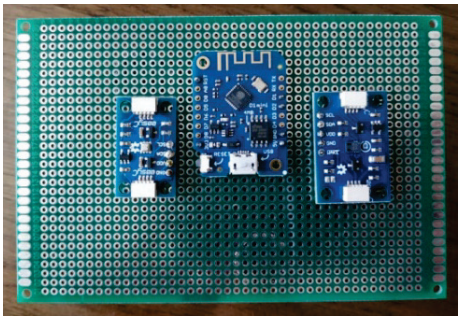

Figure 10 Wemos D1 sensors soldered on the board

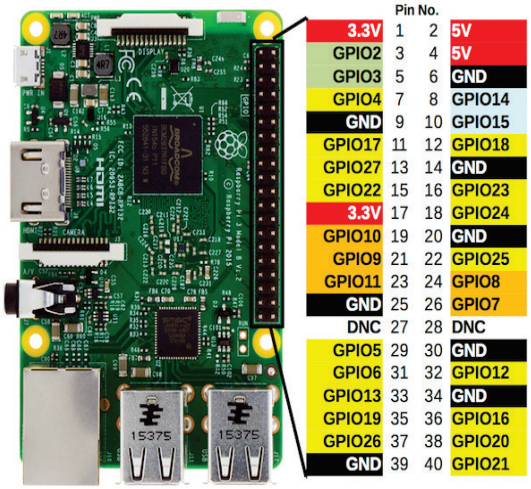

Figure 11 Layout of GPIO pins on the Raspberry Pi board

The Raspberry Pi 3B used in this work, contains forty General Input Output Pins (GPIO). GPIO pins are divided into pins with a constant voltage of 5.0 V, with a constant voltage of 3.3 V, pins for grounding (Ground - GND), pins

with an adjustable voltage and pins to which nothing is connected (Do Not Connect – DNC), according to [35].

Fig. 11 shows the arrangement of GPIO pins on the circuit board of the Raspberry Pi computer [36].

The Mosquitto MQTT broker and server will be configured on the Raspberry Pi minicomputer, which will enable local access to the Web application for displaying sensor readings.

2.2 MQTT Protocol Setup

MQTT is a communication protocol often used in IoT projects. It is responsible for data transfer management and is based on the *publish/subscribe* principle of operation. It is used when transmitting a small amount of data in remote locations, or in cases where a low bandwidth of wireless data transmission is available. It is simple and designed to be easily implemented [37]. It usually works over the existing TCP/IP protocol, but as a basis, it can also use other two-way protocols with loss protection (UDP protocol) [38].

Arlen Nipper and Andy Stanford-Clark created the first version of the MQTT protocol in 1999 [39, 40]. IBM released the next version (v3.1) in 2013, and then v3.1.1, which became the OASIS (*Organization for the Advancement of Structured Information Standards*) standard. The MQTT protocol started using the ISO standard in 2016 (ISO/IEC PRF 20922).

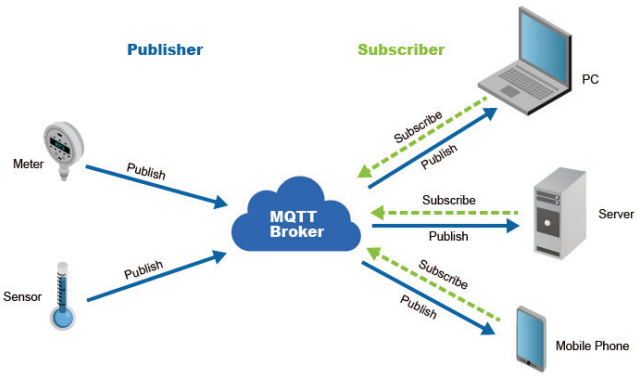

Figure 12 The working principle of the MQTT protocol [42]

Fig. 12 shows the basic principle of operation of the MQTT protocol. The MQTT *broker* communicates with all *clients* (devices). On the left side are clients that measure certain values and publish this data, and on the right side are clients that process the data to which they are subscribed. The MQTT protocol works on the publish/subscribe principle [41]. Sensors that measure some physical quantities publish measurement results on a certain topic. Clients to whom information needs to be delivered are subscribed to that same topic. When a message is published on a certain topic, it is received by all clients who have subscribed to that topic.

The MQTT broker performs the task of "intermediator" between clients and thus is the central part of the whole publish/subscribe mechanism. The main tasks of the MQTT broker are: receiving messages from all clients who publish on a certain topic, filtering messages, deciding whether to forward specific messages and sending messages to subscribed clients.

The job of an MQTT *broker* can be performed by any computer whose specifications are powerful enough for the chosen *broker*, whether it is based on a Windows or Linux operating system. Today there are a large number of such programs. Some of the most used free MQTT *brokers* are Mosquitto, Emqtt, Hive MQ, MOSCA. In this paper, Mosquitto is used as a *broker*, which is configured on a Raspberry Pi 3B computer.

MQTT *clients* include subscribers and publishers. The MQTT client can simultaneously function as a subscriber (a client subscribed to a topic) and a publisher (a client publishes a message on a topic). The MQTT *client* task can be performed by a wide range of devices, from microcontrollers to personal computers, that is, any device that has implemented support for working with the MQTT protocol (MQTT libraries). MQTT libraries are available for a large number of programming languages: Java, JavaScript, C, C++, C#, as well as for the iOS and .NET platforms.

As an MQTT client, this paper uses a Web application made in NodeJS technology, which can be accessed in a local network, and is located in a server configured also on a Raspberry Pi computer.

Quality of Service (QoS) is an agreement between the publisher and subscriber of a message that defines the security of message delivery, depending on the level of the agreement.

In MQTT protocol, three levels of agreement are defined [43]:

- level 0: At most once – the message was sent only once, neither the client nor the broker perform any additional checks to see if the message was successfully received,
- level 1: At least once – the message is sent until the client confirms its receipt,
- level 2: Exactly once – the broker and the client exchange messages in two levels (two level handshake), in order to make sure that the message reaches the client exactly once.

Before installing Mosquitt, a software upgrade of the Raspberry Pi will be performed [44, 45].

sudo apt – get update
sudo apt – get upgrade

After that, Mosquitto and its associated client package are installed.

sudo apt-get install mosquitto -y
sudo apt-get install mosquitto-clients –y

After installing these two packages, it is necessary to configure the broker. The Mosquitto broker configuration file is located at /etc/mosquitto/mosquitto.conf.

sudonano /etc/mosquitto/mosquitto.conf

```
// A function to connect Wemos to an MQTT broker
voidconnect_MQTT(){

Serial.print("Connecting to ");
Serial.println(ssid);

  // Connect to Wi-Fi
WiFi.begin(ssid, wifi_password);

  // Waiting until a Wi-Fi connection is established before continue the
program

while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
  }

  // Debugging - printing the IP address of Wemos D1

Serial.println("WiFi connected ");
Serial.print("IP address: ");
Serial.println(WiFi.localIP());

  // Connection to MQTT broker

if (client.connect(clientID, mqtt_username, mqtt_password)) {
Serial.println("Successful connection to MQTT broker!");
  }
else {
Serial.println("Failed to connect to MQTT broker...");
  }
}
```

Figure 13 Web application opened on a smartphone

At the bottom of the file is the line:

include_dir /etc/mosquitto/conf.d

It will be deleted and these three lines will be added, telling Mosquitto that only clients with the correct username and password can subscribe to its topics, and that Mosquitto listens for messages on port 1883:

allow_anonymous false
password_file /etc/mosquitto/pwfile
listener 1883

If the user does not want the broker to require a username and password from the subscriber clients, he omits the first two lines. The changes to the file are saved and the file is closed. After that, it is necessary to specify a username and password. Enter the following command: - replacing username, where username is the desired username. Then the desired password is entered and the Raspberry Pi is restarted with the command:

sudo reboot

The Mosquitto MQTT broker is now configured and ready to receive messages from the Wemos D1 microcontroller board.

2.3 Python Script for Displaying the Results

To display the values from the sensors, which the Raspberry Pi receives from Wemos via the MQTT protocol, a Python script was created that displays the results in a Web application (Fig. 13), which can be accessed within the local network. The web application has a simple design, adapted for display on smartphones. The function responsible for generating the Web application is according to [46].

At the beginning of the program code, we enter the libraries for using the CCS811 and BME280 sensors, as well as the *PubSubClient.h* library for using the MQTT protocol on the Wemos D1 board. The *ESP8266WiFi.h* library allows Wemos to connect to a Wi-Fi network. The topics that will be sent via MQTT are defined, as well as the username and password, which must correspond to those configured in the Mosquitto broker on the Raspberry Pi.

Function for connecting Wemos D1 to MQTT broker and sending data. The previously defined username and password are used. In case of an inability to connect, it returns a warning to the user.

MQTT can only send data in string form (character). Therefore, it is first necessary to convert the measured values from the sensor from the real (float) form into character notation. Then the value is sent (Publish) to the MQTT broker. At the end of the code, disconnection from the MQTT topic is performed.

3 MEASUREMENT OF AIR QUALITY IN REAL CONDITIONS

Using the developed device, air measurements were made in three different environments:

- an office room,
- an underground garage,
- an industrial welding hall.

The sensor read values every five minutes for two hours, and before starting the measurement, it worked for one hour to make the measurement as accurate as possible (in the CCS811 sensor documentation, at least half an hour of "warm-up time" is recommended).

The values of the quantities measured by the CCS811 sensor and their impact on people are shown in Tab. 3.

Table 3 Characteristics of the BME280 sensor

Concentration CO_2(ppm)	Impact on people	Concentration TVOC-a (ppb)	Impact on people
< 500	normal	< 50	normal
500 - 1000	a little uncomfortable	50 -750	uncomfortable, anxiety
1000 - 2500	tiredness	750 - 6000	headache, depression
2500 - 5000	harmful to health	> 6000	headache, harmful to the nervous system

The first measurement was made in an office room with twenty employees. The CO_2 value ranged from 300 ppm to 550 ppm, while the TVOC value ranged from 10 ppb to 45 ppb. After 45 minutes from the beginning of the measurement, the window was opened and the room was ventilated, which contributed to the reduction of the concentration of both quantities. The measured concentrations are within normal limits and do not cause any disturbance or discomfort to people (Fig. 14).

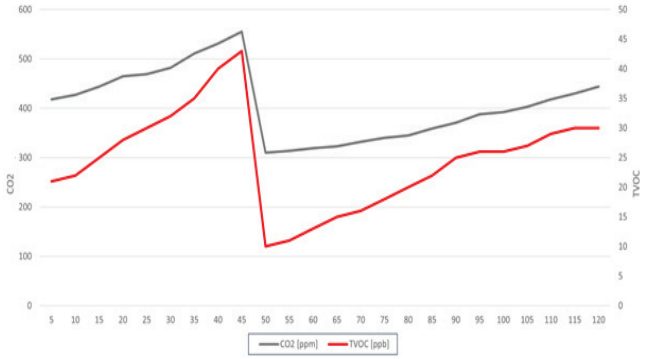

Figure 14 Measured air quality in the office room

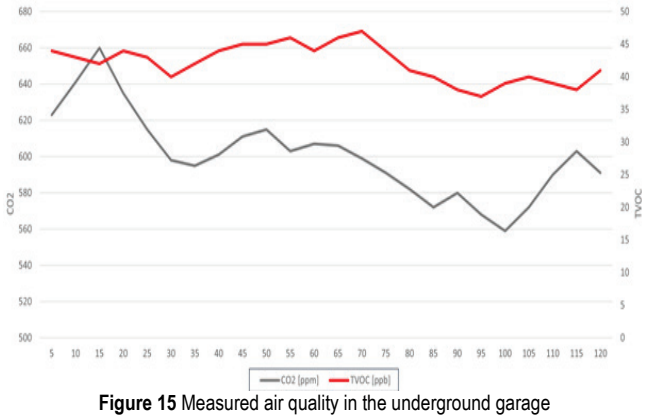

Figure 15 Measured air quality in the underground garage

The second measurement was made on the second level of an underground garage (Fig. 15). The CO_2 value ranged from 560 ppm to 660 ppm, while the TVOC concentration ranged from 40 ppb to 45 ppb all the time. The CO_2 value is slightly above normal, but still did not cause any disturbances, while the TVOC value is within normal limits. The concentrations were at a similar level all the time due to the constant forced air exchange in the underground garage.

The third measurement was carried out in an industrial hall where five welders and two other employees work in other jobs (e.g. drilling, cutting, grinding metal). CO_2 concentration ranged from 1500 ppm to 4750 ppm, while TVOC values were between 900 ppb and 2500 ppb. The measured concentrations are very high due to jobs that bring a large amount of metal particles into the air, as well as lubricants, paints and solvents (Fig. 16). This kind of working environment requires the use of protective breathing masks (respirators), because this high concentration of substances in the air is very harmful to the health of employees.

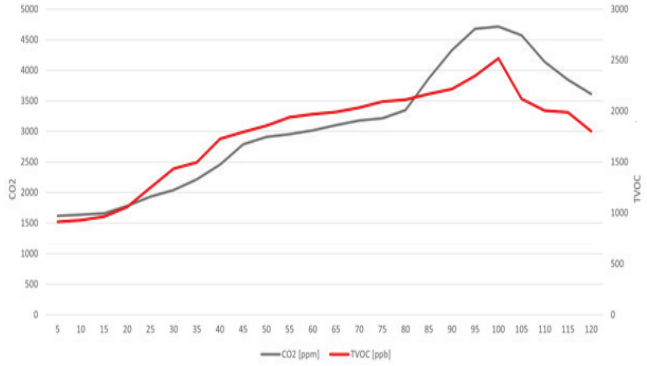

Figure 16 Measured air quality in the welding hall

By performing the measurements, an idea was obtained for a possible improvement of the device, which is the automatic sending of an e-mail message or SMS to the responsible person in the company when the concentrations of CO_2 and TVOC exceed a certain value, in order to ventilate the hall.

4 CONCLUSION

The concept of the *Internet of Things* is explained using the example of an air quality sensor. Since IoT is a relatively new concept and platform, it can be assumed that in the near future most products on the market will become part of this platform, and that they will have the ability to monitor or manage some of their parameters. IoT as an industry has a huge potential for development.

The processed air quality control device has a wide potential application, from industrial plants to home automation systems. Relatively cheap and easily available parts enable the control and processing of various sizes, and their storage in a database for the possibility of checking values in a past period of time and comparing them with today. Thus, for example, a comparison could be made as to whether a change in the production process of a company contributed to the improvement of indoor air quality, resulting in a more pleasant workplace for employees and a reduction in the harmful impact on their health.

The development of smart devices is facilitated by the use of development boards based on various microcontrollers, such as the used Wemos D1 Mini. Such development boards already contain many connectors and pins that facilitate experimentation and testing with various sensors and other electronics. Of course, mass-produced devices use only microcontrollers and those electronic components that are necessary, but for enthusiasts and those

who want to learn about electronics and programming, development boards are a great solution.

It was shown how cheap mini-computers like Raspberry Pi have enough power to run servers that control smaller IoT systems. Such computers come every year with better specifications and a lower price, and it is difficult to predict how far they will go in a few years and what they will be able to run.

Air measurements were made in three different environments: an office room, an underground garage and in industrial welding hall. Measured concentrations in an office room are within normal limits and do not cause any disturbance or discomfort to people. The concentrations in the underground garage were at a similar level all the time due to the constant forced air exchange. The measured concentrations in industrial welding hall are very high due to jobs that bring a large amount of metal particles into the air, as well as lubricants, paints and solvents.

Future research could explore the effectiveness of the processed air quality control device in different industrial and home settings and assess its impact on indoor air pollution levels, employee health, and overall productivity. Future research could also explore the economic feasibility of implementing the device in various contexts, and evaluate alternative designs and features to improve its performance and ease of use. In addition, further research may be needed to assess the potential benefits and limitations of integrating the device with other smart home technologies and environmental monitoring systems, and to explore new applications and markets for processed air quality control devices.

Acknowledgement

This work was funded by the European Union through the European Regional Development Fund, under the project "Smart Sticker for measuring and monitoring storage and transportation conditions of products" KK.01.1.1.04.0116.

5 REFERENCES

[1] Delija, G. (2021). Kontrola kvalitete zraka korištenjem IoT platform. *Master's thesis*, University North, Croatia. https://urn.nsk.hr/urn:nbn:hr:122:241977, (in Croatian) (Accessed 1/18/2023)

[2] Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials, 17*(4), 2347-2376. https://doi.org/10.1109/COMST.2015.2444095

[3] Amodu, O. A. & Othman, M. (2018). Machine-to-Machine Comunication: An Overview of Opportunities. *Computer Networks, 145*, 255-276. https://doi.org/10.1016/j.comnet.2018.09.001

[4] Uzair, M., Al-Kafrawi, S. Y., Al-Janadi, K. M., & Al-Bulushi, I. A. (2022). A Low-Cost IoT Based Buildings Management System (BMS) Using Arduiono Mega 2560 and Raspberry Pi 4 for Smart Monitoring and Automation. *IJECES, 13*(3), 219-236. https://doi.org/10.32985/ijeces.13.3.7

[5] Yassein, M. B., Shatnawi, M. Q., Aljwarneh, S., & Al-Hatmi, R. (2017). Internet of Things: Survey and open No s of MQTT protocol. *Proceedings of the International Conference on Engineering & MIS (ICEMIS)*, Monastir, Tunisia, 1-6. https://doi.org/10.1109/ICEMIS.2017.8273112

[6] Hedi, I., Špeh, I., & Šarabok, A. (2017). IoT network protocols comparison for the purpose of IoT constrained networks.*Proceedings of the 40thInternational Convention on Information and Communication Technology, Electronics and Microelectronics*, Opatija, Croatia, 22-26 May 2017, 501-505. https://doi.org/10.23919/MIPRO.2017.7973477

[7] Shafiq, M., Gu, Z., Cheikhrouhou, O., Alhakami, W., & Hamam, H. (2022). The Rise of "Internet of Things": Review and Open Research Issues Related to Detection and Prevention of IoT-Based Security Attacks. *Wireless Communications and Mobile Computing*, e8669348. https://doi.org/10.1155/2022/8669348

[8] Crnjac-Milić, D., Dujmenović, I., & Peko, M. (2023). An Approach to the Application of the Internet of Things in Logistics. *Tehnički glasnik, 17*(1), 134-140. https://doi.org/10.31803/tg-20220609190233

[9] Medium.com. The internet of things (IoT). https://medium.com/zeux/the-internet-of-things-iot-5-reasons-why-the-world-needs-it-125fe71195cc (Accessed 1/18/2023)

[10] Ahire, B. A. & Sakhare, S. R. (2021). Sound Prohibited Zone for Smart Cities using IoT. *Tehnički glasnik, 15*(1), 92-97. https://doi.org/10.31803/tg-20210205091148

[11] Terroso-Saenz, F., González-Vidal, A., Ramallo-González, A. P., & Skarmeta, A. F. (2019). An open IoT platform for the management and analysis of energy data. *Future Generation Computer Systems, 92*(2), 1066-1079. https://doi.org/10.1016/j.future.2017.08.046

[12] Sadhu, V., Zhao, X., & Pompili, D. (2020). Energy-Efficient Analog Sensing for Large-Scale and High-Density Persistent Wireless Monitoring. *IEEE Internet of Things Journal, 7*(8), 544-559. https://doi.org/10.1109/JIOT.2020.2984484

[13] Casado-Vara, R., Martin-del Rey, A., Affes, S., Prieto, J., & Corchado, J. M. (2020). IoT network slicing on virtual layers of homogeneous data for improved algorithm operation in smart buildings. *Future Generation Computer Systems, 102*, 965-977.https://doi.org/10.1016/j.future.2019.09.042

[14] Yang, C. H., Lee, K. C., & Li, S. E. (2020). A mixed activity-based costing and resource constraint optimal decision model for IoT-oriented intelligent building management system portfolios. *Sustainable Cities and Society, 60*(3), 655-671. https://doi.org/10.1016/j.scs.2020.102142

[15] ShareTechnote. 4G/LTE – NB IoT Operation Mode. https://www.sharetechnote.com/html/Handbook_LTE_NB_LTE.html (Accessed 1/18/2023)

[16] Avsystem. What is Narrowband IoT? NB-IoT overview. https://www.avsystem.com/blog/narrowband-iot/ (Accessed 1/18/2023)

[17] IoT for all. The role of WiFi in IoT. https://www.iotforall.com/wifi-role-iot (Accessed 1/18/2023)

[18] Link Labs. Bluetooth Low Energy M"M and IoT Applications.https://www.link-labs.com/blog/bluetooth-vs-bluetooth-low-energy (Accessed 1/18/2023)

[19] Gomez, C., Oller, J., & Paradells, J. (2012). Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology. *Sensors, 12*(9), 11734-11753. https://doi.org/10.3390/s120911734

[20] Adelantado, F., Vilajosana, X., Tuset-Peiro, P., Martinez, B., Melia, J., & Watteyne, T. (2017). Understanding the limits of LoRaWAN. *IEEE Communications Magazine, 55*(9), 34-40. https://doi.org/10.1109/MCOM.2017.1600613

[21] SparkFun. Air Quality Breakout - CCS811. https://www.sparkfun.com/products/14193 (Accessed 1/19/2023)

[22] InnovatorsGuru. AMS - Ultra_Low Power Digital Gas Sensor for Monitoring Indoor Air Quality, General Description, Block diagram from the manual, page 3. https://innovatorsguru.com/wp-content/uploads/2020/01/CCS811_Datasheet-DS000459.pdf (Accessed 1/20/2023)
[23] SparkFun. Characteristics of the CCS811 sensor. https://cdn.sparkfun.com/assets/learn_tutorials/1/4/3/CCS811_Datasheet-DS000459.pdf (Accessed 1/18/2023)
[24] NXP. I²C-bus specification and user manual.https://web.archive.org/web/20210426060837/https://www.nxp.com/docs/en/user-guide/UM10204.pdf (Accessed 1/18/2023)
[25] GitHub. Arduino library for the CCS811. https://github.com/maarten-pennings /CCS811 (Accessed 1/18/2023)
[26] Distrelecwebshop images. BME280 sensor picture. https://www.distrelec.biz/Web/WebShopImages/landscape_large/2-/01/Adafruit-2652-30091192-01.jpg (Accessed 1/18/2023)
[27] Bosch. Complete schematic of typical BME280 use-case. https://community.bosch-sensortec.com/t5/Knowledge-base/BME280-series-humidity-sensor-design-guide/ta-p/7385 (Accessed 1/18/2023)
[28] Bosch. Technical data of the sensor BME280. https://www.bosch-sensortec.com/products/environmental-sensors/humidity-sensors-bme280/ (Accessed 1/20/2023)
[29] Last minute engineers. Arduino library manager. https://lastminuteengineers.com/bme280-arduino-tutorial/ (Accessed 1/20/2023)
[30] Espressif. ESP8266 overview. https://www.espressif.com/en/products/socs/esp8266 (Accessed 1/20/2023)
[31] Auscomtech. ESP8266MOD picture. https://www.auscomtech.com.au/wp-content/uploads/2018/12/AD483-2.jpg (Accessed 1/20/2023)
[32] Orellana, C., Macías, M., González-Velasco, H., Manso, A., & Gallardo-Caballero, R. (2019). Low-Power and Low-Cost Environmental IoT Electronic Nose Using Initial Action Period Measurements. *Sensors, 19*(14), 3183. https://doi.org/10.3390/s19143183
[33] Asinelli, M. G., Serra, M. S., Marimon, J. M., & Espaulella, J. S. (2018). The smARTS_Museum_V1: An open Hardware device for remote monitoring of Cultural Heritage indoor environments. *HardwareX, 4*, e00028. https://doi.org/10.1016/j.ohx.2018.e00028
[34] Wemos.cc. Wemos D1 mini board. https://www.wemos.cc/en/latest/d1/d1_mini.html (Accessed 1/20/2023)
[35] RaspberryPi. Specification of the Raspberry Pi. https://www.raspberrypi.org/products/raspberry-pi-3-model-b/ (Accessed 1/20/2023)
[36] Bigmessowires. Raspberry GPIO. https://www.bigmessowires.com/wp-content/uploads/2018/05/Raspberry-GPIO.jpg (Accessed 1/20/2023) https://doi.org/10.1007/978-1-4842-3948-3_1
[37] Tareq, R. W. & Khaleel, T. A. (2021). Implementation of MQTT Protocol in Health Care Based on IoT Systems: A Study. *IJECES, 12*(4), 215-223. https://doi.org/10.32985/ijeces.12.4.5
[38] U-blox. MQTT explained. https://www.u-blox.com/en/blogs/insights/mqtt-beginners-guide (Accessed 1/22/2023)
[39] Sugumar, K. (2020). MQTT-aLightweight Communication Protocol Relative Study. *Author Preprints*, 1-5. https://doi.org/10.22541/au.159076954.48540044
[40] Mishra, B. (2018). TMCAS: An MQTT based collision avoidance system for railway networks. *Proceeding of the 18th International Conference on Computational Science and Applications*, Melbourne, VIC, Australia, 2-5 July 2018, 1-6. https://doi.org/10.1109/ICCSA.2018.8439562
[41] Jaloudi, S. (2019). Communication protocols of an industrial internet of things environment: A comparative study. *Future Internet, 11*(3), p. 66. https://doi.org/10.3390/fi11030066
[42] Oringnet. The working principle of the MQTT protocol. https://oringnet.com/upload/media/Technology/MQTT/MQTT_1091217-01.jpg (Accessed 1/25/2023)
[43] Toldinas, J., Lozinskis, B., Baranauskas, E., & Dobrovolskis, A. (2019). MQTT Quality of Service versus Energy Consumption. *Proceedings of the 23rd International Conference Electronics*, Palanga, Lithuania, 17-19 June 2019, 1-4. https://doi.org/10.1109/ELECTRONICS.2019.8765692
[44] Self Hosted Home. Setting up MQTT Broker for DIY Home Assistant Sensors. https://selfhostedhome.com/setting-up-mqtt-broker-for-diy-home-assistant-sensors/ (Accessed 1/25/2023)
[45] PiMyLifeUp. Installing the Mosquitto MQTT Server to the Raspberry Pi. https://pimylifeup.com/raspberry-pi-mosquitto-mqtt-server/ (Accessed 1/25/2023)
[46] Random nerd tutorials. MQTT Libraries. https://randomnerdtutorials.com/esp8266-nodemcu-mqtt-publish-dht11-dht22-arduino/ (Accessed 1/25/2023)

Authors' contacts:

Marinko Stojkov, PhD, Full Professor
University of Slavonski Brod,
Trg Ivane Brlic Mazuranic 2, 35000 Slavonski Brod, Croatia
mstojkov@unisb.hr

Goran Delija, student
University Center Varaždin, Mechanical Engineering Department,
Jurja Križanića 31b, 42000 Varaždin, Croatia
goran.delija1@gmail.com

Ivan Đuračić, PhD student
(Corresponding author)
University of Slavonski Brod,
Trg Ivane Brlic Mazuranic 2, 35000 Slavonski Brod, Croatia
idjuracic@gmail.com

Tomislav Alinjak, PhD
Elektra Slavonski Brod,
Petra Krešimira IV 11, 35000 Slavonski Brod, Croatia
tomislav.alinjak@hep.hr