THE CLOUD-BASED SYSTEM FOR MONITORING METEOROLOGICAL DATA BASED ON MICROCONTROLLER AND WEB APPLICATION

Željko Eremić*,
[0009-0003-3310-3081]
Dragan Halas
[0009-0005-1426-0326]

Technical College of Applied Sciences in Zrenjanin,
Zrenjanin, Serbia

Abstract:
The scope of this solution is an example of the implementation of a small and simple weather station that measures, stores, and displays data in proper format. C++, JavaScript, React.js, and Bootstrap were used for software development. For software development, an object-oriented methodology was used as well as appropriate design patterns by the needs. Many modern concepts are applied including programmable microcontrollers, cloud-hosted databases, and monitoring in real-time. Our findings and conclusions indicate that the cloud-based monitoring platform for tracking weather conditions has been successfully implemented. The research also contains a practical experiment that confirms the possibilities of the proposed IoT solution. The first chapter provides an insight into the problem, and lists related works in this area, as well as works that precede ours. The second chapter focuses on materials and methods and explains the methodology and hardware used, as well as the subsystems of the proposed system. Graphical interpretation was performed using React.js. The third chapter describes the practical experiment that was performed, as well as the presentation of the obtained results in several ways. At the end, a discussion is given. The last chapter gives concluding remarks related to the proposed solution and results.

Keywords:
Cloud computing, Internet of things, Object-oriented methodology, Sensor, Weather station.

INTRODUCTION

Cloud computing has been present for a long time in many areas of human creativity. One of these areas is meteorology. The advent of cheap and easy-to-use sensors and microcontrollers has opened the possibility for a wide range of experts to get involved in this area as well. On the other hand, there are significant improvements in the field of Information Technology. One of them is real-time databases, which enable easy storage but also reading of stored data. Another improvement is web applications that allow data to be read from these databases and interpreted in a convenient form.

The main goal of this paper is to present a monitoring system that can perform continuous measurements of several meteorological parameters, store these values over time, and finally show the movement of the values

Correspondence:
Željko Eremić

e-mail:
zeljko.eremic@vts-zr.edu.rs

DOI: 10.15308/Sinteza-2024-66-73

of these parameters in a certain period using graphs. Measurements are performed using appropriate sensors and microcontrollers, values are stored in a real-time database and a graphical display is provided by a web application.

With the advent of Arduino and similar platforms in recent years, there has been a large increase in interest in meteorology. Relatively cheap and simple sensors combined with programmable microcontrollers have attracted significant attention from both experts and amateurs. Related works in this area will be presented below.

One of the applications of meteorological data is in the field of beekeeping. The IoT Concept for Bee Colony Monitoring [1] provides a solution in this area. Values for temperatures and humidities inside and outside of beehives, and weight deviations are measured. The web-based interface displays the results in the form of graphs. "The offered autonomous beekeeping system applies sensors that provide useful data on hive status (internal and ambient temperature, humidity, weight of the hives). Such sensors provide important data to the users, so they can evaluate the hive and take further action." [1]

Paper [2] presents a cloud-based monitoring platform used in the field of agriculture. The parameters monitored are soil moisture (percentage volumetric water content), humidity, ambient temperature, dew point and soil temperature. This data is stored on a server and is graphically interpreted using the JavaScript library React.js and HTML script charts. "Chartjs is used for graphically visualizing the received data. It is based on JavaScript's HTML5 web syntax, thereby allowing for implementation of script tags for displaying graphs." [2]

Another NodeMCU-based system that allows data collection and recording is a WiFi-based portable weather station [3]. It uses a cloud server to store data, and data transmission is based on wireless communications.

When it comes to cheap solutions, the one presented in [4] is also interesting. The system is largely automated and measures and stores atmospheric parameters without human participation. Its work is based on PIC microcontroller, and GSM module and uses GPRS communication protocol. Unlike the solutions we propose in this paper, this system sends SMS messages.

The application for displaying Blynk results [5] is used to measure the values of temperature and relative humidity. Similar to our solution, it uses a NodeMCU microcontroller and DHT11 sensor.

The work that largely coincides with this is [6] in terms of architecture. This solution allows you to view current values of temperature, relative humidity, and air pressure via the React web application. The storage of data from previous measurements is not supported, but only the last measured values are stored in the real-time database.

The system that precedes the one we present in the paper is described in the paper [7]. It describes the System for automatic measurement and storage of meteorological data. Continuous measurement of values for temperatures, relative humidity and air pressure and their storage is provided. The measurement results were finally converted into graphs using an Excel program.

Solution [6] uses simple gauge elements to display the last measured values but does not have archives of the values measured so far, nor related graphics. On the other hand, solution [7] provides the possibility of archiving multiple measurements but requires subsequent manual data processing. The work presented in this paper seeks to compensate for the shortcomings that exist in them, and also to offer much more conformal work about other similar approaches.

2. MATERIALS AND METHODS

The approach proposed here is a solution to the research problem of continuously measuring values of temperature, relative humidity, and air pressure, then keeping the values for a certain time and displaying the measurement results automatically. Appropriate architecture, approaches, tools, hardware, programming languages, and methods are used to achieve this solution, which will be presented below. The complete system consists of three subsystems which will be explained in more detail in the text that follows.

2.1. MEASUREMENT SUBSYSTEM

This subsystem consists of two sensors and a NodeMCU microcontroller. The sensors are the well-known and widespread DHT22 which measures temperature and relative humidity and the BMP180 sensor which measures air pressure. NodeMCU is a microcontroller that was chosen for this solution primarily because it has an integrated Wi-Fi card and easily establishes a connection with the router in whose range it must be.

NodeMCU is programmable, and for the needs of the solution presented in this paper, a program was written for it in the C ++ programming language. The source code of the software for the measurement subsystem is available on GitHub [8].

In addition to the measurement functions, this program also provides the necessary functionalities to send the measured values to the real-time database. The measurement is performed in the time specified in the program code. The entire subsystem is activated when it receives power. It is possible to power the system via a power bank device. An image of this subsystem (hardware) and its connection diagram can be seen in Figure 1.

2.2. STORAGE SUBSYSTEM

To be able to monitor the results of continuous measurement, it is necessary to keep them over time. The solution proposed in this paper uses a real-time database named Firebase. "The Firebase Realtime Database is cloud-hosted. Data is stored as JSON and synchronized in real-time to every connected client. When you build cross-platform apps with our iOS, Android, and JavaScript SDKs, all of your clients share one real-time database instance and automatically receive updates with the newest data." [9]

2.3. DISPLAY SUBSYSTEM

The display subsystem is purpose-built for this solution. It works in combination with a real-time database and is displayed in a web browser. Each time a measurement is made and the results are added to the real-time database, the web application is notified and automatically updated. Allows display in the form of a graph. To make all this possible, some specific React libraries are installed and used like Firebase, react-chartjs-2, react-svg-gaug, react-bootstrap, file-saver and xlsx.

There are two chart display modes, one summary that contains all three charts on the screen, and the other that is obtained by selecting a specific chart where only that chart is displayed in full screen. Also, through the main menu, it is possible to see the Reports section, where the measurement results can be obtained in JSON or CSV format. The CSV format allows you to export the results to a file, which Excel can open. Also, the results can be downloaded in Excel file format. The web application is available at a web location [10], and its source code is at GitHub [11].

2.4. COMPLETE SYSTEM

The complete system consists of three listed subsystems. The channel of communication between subsystems is the Internet, which has the consequence that subsystems can be located in physically different locations.

Measurement subsystem can be powered via a Power Bank which makes it mobile within the range of the router to which it is connected. Also, display subsystem can be performed on a device that is connected to the Internet and mobile. It could be said that our system is both distributed and to some extent mobile, which is certainly an advantage. The block diagram of the complete system can be seen in Figure 2.

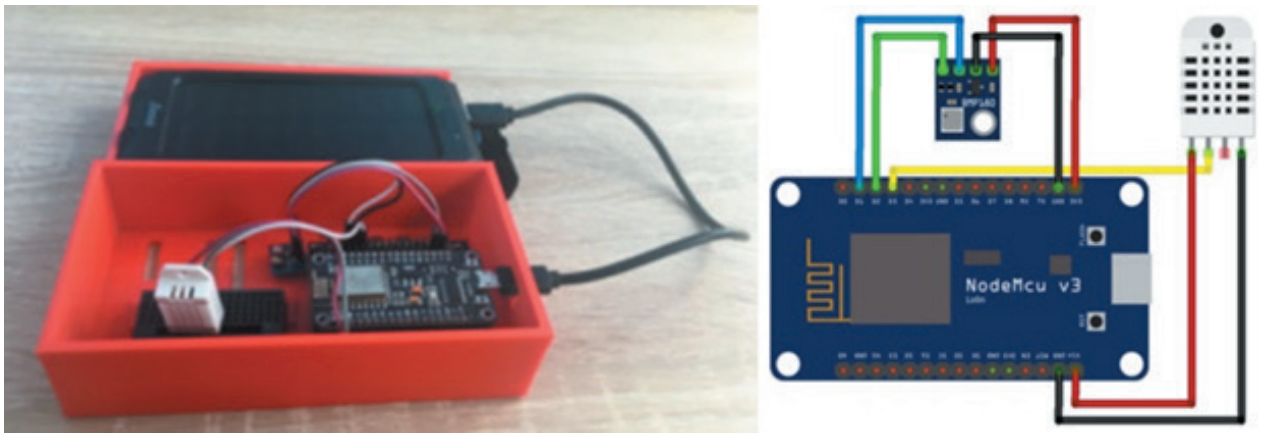

Figure 1. Measurement subsystem and its connection diagram.

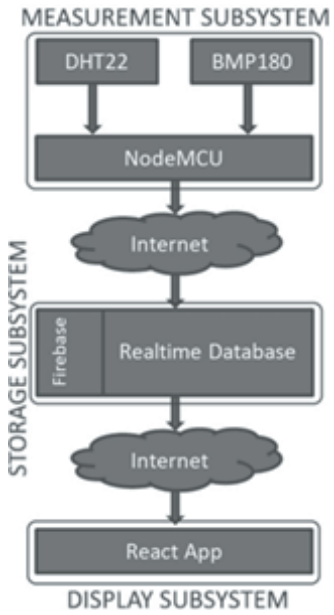

Figure 2. The connections of the subsystems are shown in the block diagram.

3. RESULTS AND DISCUSSION

To verify the operation of this system, an experiment was performed. The measurement was performed in the period from October 25, 2021, at 5 pm to October 26, 2021 at 4 p.m. The frequency of measurements was one measurement every hour so that a total of 24 measurements were made. The location of the measurement was in the city of Zrenjanin, Serbia. The measurement was performed using the measurement subsystem, which is shown in Figure 1. The following data were recorded for each measurement:

- Atmospheric pressure (mBar)
- Day of the week (0-6, where it is 0 Sunday and 6 Saturday)
- Relative humidity (%)
- Temperature (°C)
- Time (HH-MM-SS format)

The measurement results are stored in a real-time database. Two data are also recorded during each measurement and they are the current time and day of the week. This data is obtained by sending a request to the NTP server, which is described in [12]. "The idea here is to use NTP to set the computer clocks to UTC and then apply any local time zone offset or daylight saving time offset. This allows us to synchronize our computer clocks regardless of location or time zone differences." [12]

Two nodes in a real-time database were used in data storage:

- meteoarchive3
- meteoarchive3_current

The meteoarchive3 node contains the measurement history organized in such a way that the results of each measurement are placed in a separate subnode. To save the results of the last performed measurement, the node meteoarchive3_current is used. Data from meteoarchive3 in table format is given in Table 1. Part of the contents of the meteoarchive3 and meteoarchive3_current can be seen in Figure 3.

Table 1. Data from meteoarchive3 in table format.

atm pressure	day of week	rel humidity	temperature	time
1027	1	27.5	18.4	17:00:04
1027	1	31.4	16.1	18:00:04
1027	1	34.1	14.2	19:00:04
1027	1	36.2	13.4	20:00:04
1027	1	40.6	12.1	21:00:04
1026	1	44	10.7	22:00:04
1026	1	44	10.6	23:00:04
1026	2	48.4	9.4	00:00:04
1026	2	50	9	01:00:04
1025	2	50.7	9	02:00:04
1025	2	51.4	8.6	03:00:04
1024	2	54.7	7.9	04:00:04
1024	2	56.4	7.4	05:00:04
1024	2	56.7	7	06:00:04
1024	2	56.1	7.2	07:00:04
1025	2	56.3	7.2	08:00:04
1025	2	54	7.8	09:00:04
1025	2	52.1	9.4	10:00:04
1025	2	46.5	11.1	11:00:04
1025	2	34.3	16.8	12:00:04
1024	2	29.3	18.5	13:00:04
1023	2	28.4	19.5	14:00:04
1023	2	27.4	19.9	15:00:04
1023	2	27	20	16:00:04

Figure 3. Nodes meteoarchive3 and meteoarchive3_current.

The appropriate account and settings on Firebase have been previously created for this experiment. Whenever a change occurs in Firebase, the new state is loaded into the web application in the appropriate place. Firebase is in our case a storage subsystem. The display subsystem is based on React.js. "Reactjs is a JavaScript library used to facilitate interactive and reusable UI components. It uniquely performs operations on both the client side and server side." [2]. The measurement so far as well as the last modified results can be seen using the display subsystem. This subsystem is realized using a web application. It is placed on the site https://monitoring-meteo.web.app [10]. The X-axis contains values in format: HH:MM: SS [D] where HH are hours, MM minutes, SS seconds, and [D] is the day of the week that takes values from the range 0 to 6, where it is 0 Sunday and 6 Saturday.

The web application contains the following features:

- Display of the last changed values via the gauge and display of all three graphs (Figure 4)
- Overview of individual graphics (access via the main menu)
- Generate reports in the following formats: JSON, csv and Excel files (Reports in the main menu) shown in Figure 5.

The results of the conducted experiment will be presented and discussed below.

Atmospheric pressure is the pressure exerted by the air atmosphere on the Earth's soil. It mostly depends on the altitude, but it also depends on the temperature and humidity of the air.

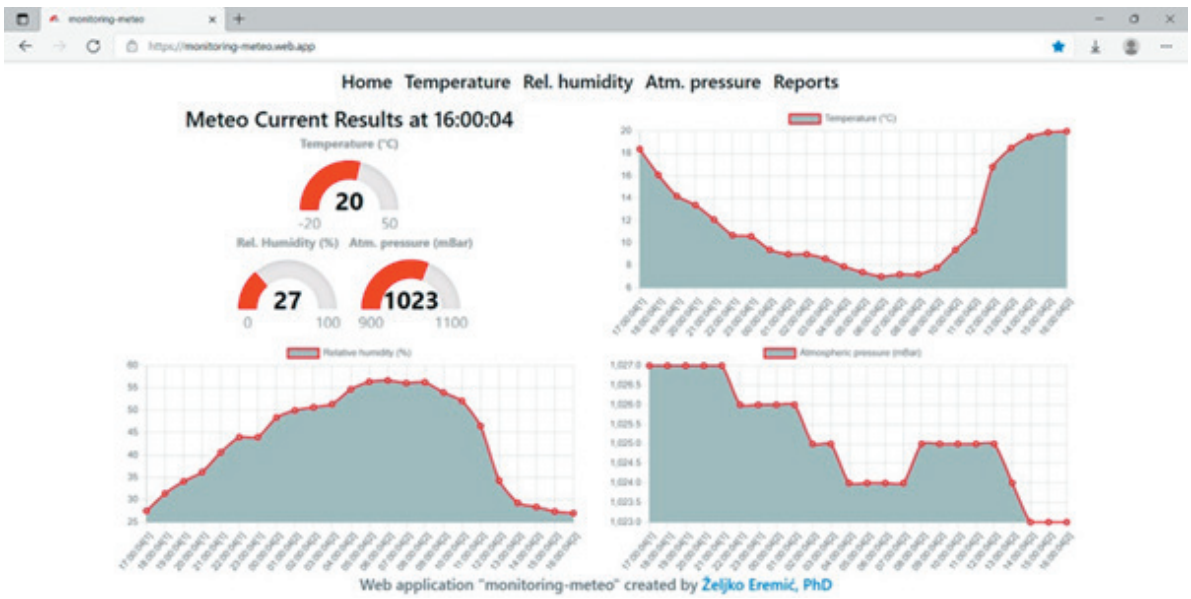

Figure 4. Display of the last changed values via the gauge and display of all three graphs.

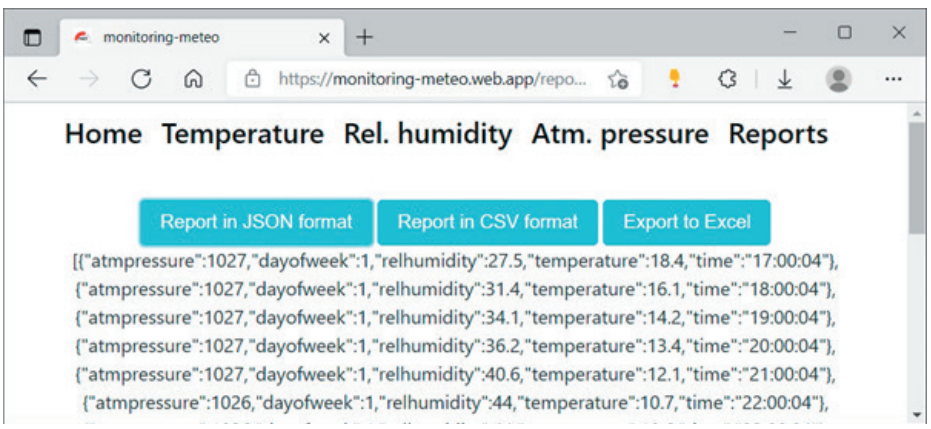

Figure 5. Reports.

Relative humidity is the ratio of the current amount of water vapor in the air and the maximum amount of water vapor that the air can contain at a given temperature. Relative humidity depends on the temperature and pressure of the observed system. The same amount of water vapor gives different values of relative humidity at different temperatures.

From the graphics in the figure, you can see that the temperature drop causes higher relative humidity, as well as the drop in atmospheric pressure. An increase in temperature causes a decrease in relative humidity. From this, it can be concluded that there was no rain in the observed period.

Our work shares many similarities with the works listed in the Introduction. Paper [1] measures temperature and humidity using graphics at its output. In another paper [2], similar values are measured as in our work and the cloud-based monitoring platform is also used, and the main difference is reflected in the fact that the field of application is agriculture. In the field of weather forecasting, there is a solution [3] that also uses NodeMCU, but also in [4] it includes different hardware in the form of a PIC microcontroller, GPRS communication protocol and sending SMS messages. The work presented is based on past work [6], because it is possible to view the results of live measurements, but without storing the results of previous measurements, and only by using a gauge, and in another [7] it is possible to archive a larger number of measurements but without the possibility for automatic graphical display of results.

The weakness of the proposed system is that it depends on the measurement subsystem being within range of the Wi-Fi router, and that the real-time Firebase database is owned by Google LLC. There is also a dependence on time data obtained from NTP servers over the Internet. The advantage of this system is that the hardware and software related to the subsystem for measurement and display are the work and property of the author, which gives the opportunity for further development. The software is available on the GitHub site and the hardware connection diagram is given in this article, which opens the possibility for other researchers to use our architecture as a basis for their research. Measurement and display subsystems are largely mobile in space. The software solutions and methods used also enabled the design of the display subsystem to be at a high level concerning similar works. Another advantage is the ability to download measurement results in as many as three popular formats, which is useful for further processing of this data. Automatically responding to new display subsystem measurements is certainly a great advantage.

Successfully performed experiments confirmed that using the proposed architecture, modern software and hardware, it is possible to successfully measure several parameters over a period of time. It is also possible to archive and display these values via a web application in a graphical format. The possibility of downloading the results in JSON and CSV format, as well as in the form of an Excel file, should not be neglected.

4. CONCLUSION

This paper presents a cloud-based monitoring platform. The system contains three subsystems: measurement subsystem, storage subsystem and display subsystem. The first subsystem was made using microcontrollers and appropriate sensors. The second subsystem was used for data storage. The development involved a cost-free real-time database Firebase. The third subsystem is a web application available on the Internet using React.js.

Similar comparable solutions were presented, as well as the solutions that were the basis for what was presented. The hardware components are purposely assembled for the purposes of this research. The software for both the first and the third subsystems was purposely written in the C++ or JavaScript programming languages, while the second subsystem was created using the existing software solution.

The essence of this paper was to develop a cloud-based system for monitoring meteorological data, storage and display, which was successfully realized. The obtained results were discussed. By analyzing the results obtained, it can be concluded that the results are logical and that there was no rain in the observed period. In addition to the display, it is also possible to export data in three different ways.

Although its purpose at the moment is related to meteorology, there are no obstacles to adapting to another area in the future. Further improvements to the representative system are planned, both in the field of meteorology and in other areas where it can be applied. There are possibilities for this system to be connected to other technical systems in the future following IoT principles. Adding some new sensors to this architecture is certainly an option that is much expected in future research.

5. REFERENCES

[1] A. Zabasta, N. Zhiravetska, N. Kunicina and K. Kondratjevs, "Technical Implementation of IoT Concept for Bee Colony Monitoring," in *In 2019 8th Mediterranean Conference on Embedded Computing (MECO)*, Budva, 2019.

[2] K. E. Adetunji and M. K. Joseph, "Development of a Cloud-based Monitoring System using 4duino: Applications in Agriculture," in *In 2018 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD)*, Durban, South Africa, 2018.

[3] I. Sarkar, B. Pal, A. Datta and S. Roy, "WiFi-based portable weather station for monitoring temperature, relative humidity, pressure, precipitation, wind speed, and direction," in *2020*, Singapore, In Information and Communication Technology for Sustainable Development.

[4] M. Đorđević and D. Danković, "A smart weather station based on sensor technology," *Facta universitatis-series: Electronics and Energetics*, vol. 32, no. 2, pp. 195-210, 2019.

[5] J. D. Irawan and R. P. Prasetya, "IoT Data Logger Using Blynk Framework," *International Journal of Latest Engineering and Management Research (IJLEMR)*, vol. 6, no. 1, pp. 17-23, 2021.

[6] Ž. Eremić, "Sistem za automatsko merenje meteoroloških podataka i prikaz putem web aplikacije," *DIT-Društvo-Istraživanje-Tehnologije*, vol. 34, pp. 57-61, 2020.

[7] Ž. Eremić, D. Halas and D. Nemet, "Sistem za automatsko merenje i čuvanje meteoroloških podataka," *DIT - Society, Research, Technology*, vol. 36, pp. 45-50, 2021.

[8] "System for automatic measurement and storage of meteorological data," [Online]. Available: https://github.com/ezeljko1981/System-for-automatic-measurement-and-storage-of-meteorological-data. [Accessed 1 3 2024].

[9] "Firebase Realtime Database," [Online]. Available: https://firebase.google.com/docs/database. [Accessed 1 3 2024].

[10] "Web application "monitoring-meteo"," [Online]. Available: https://monitoring-meteo.web.app/. [Accessed 1 3 2024].

[11] "Web application "monitoring-meteo" - source code," [Online]. Available: https://github.com/ezeljko1981/monitoring-meteo. [Accessed 1 3 2024].

[12] "Getting Date & Time From NTP Server With ESP8266 NodeMCU," [Online]. Available: https://lastminuteengineers.com/esp8266-ntp-server-date-time-tutorial/. [Accessed 1 3 2024].