Proposal for a weather monitoring system utilizing IoT technologies

Roman Budjač, Gabriel Gašpar
University of Žilina
Research Centre
Univerzitna 8215/1, 010 26 Žilina, Slovakia
{roman.budjac, gabriel.gaspar}@uniza.sk

Maroš Valášek
University of Žilina, Faculty of Electrical Engineering and Information Technology
Department of control and information systems
Univerzitna 8215/1, 010 26 Žilina, Slovakia
valasek.maros@gmail.com

Martin Bartoň
Slovak University of Technology, Faculty of Materials Science and Technology in Trnava
Institute of Applied Informatics, Automation and Mechatronics
Jana Bottu 25, 917 24 Trnava
martin.barton@stuba.sk

Abstract. *The rapid integration of modern technologies in everyday life, embodied by the Internet of Things (IoT), has significantly enhanced the efficiency and convenience of various sectors, including industrial systems. This article explores the design and implementation of a meteorological data collection, processing, and visualization system, focusing on the development and implementation of an IoT device capable of providing and transmitting these data. Additionally, we have developed a comprehensive dashboard to store, visualize, and analyze all incoming data in real-time. This system provides the capability for real-time data analysis and environmental monitoring.*

Keywords. data acquisition, IoT, microcontrollers, esp32, MQTT, Smart City

1 Introduction

In the current digital age, the integration of modern technologies into the daily functioning of society has become inevitable. This phenomenon extends to a wider range of devices that are categorized as the Internet of Things (IoT). These devices, found in homes, workplaces, or public spaces, for example, significantly increase the efficiency and convenience of our daily activities. IoT devices find applications across a wide range of sectors, from industrial production systems to the provision of data within cities and transport. The spread of IoT devices creates vast new opportunities for data collection, ultimately leading to improved efficiency, simplified tasks, and better decision-making.

Currently, the concept of Smart Cities could be implemented more effectively, and cities and regions worldwide are facing similar challenges and problems. Rapid urbanization and rapid population growth increasing demands on city infrastructure. This causes increasing demands on housing, rising energy prices, and also the need to reduce the impact of various crises, whether climatic, economic, or health, increasing the burden on the budget of local governments and the state. As we mentioned, rapid urbanization is demanding and putting pressure on all aspects of living in the cities(Ministry of Investments, 2023).

The Smart City concept not only offers solutions for so-called Smart Cities, but it is a concept that can be applied to many sectors. These can include, for example, smart housing and infrastructure, smart public transport, smart industry, smart energy grids, smart nature and more. IoT devices are used in the concept. The Smart City framework is part of the Industry 4.0 concept (Bauer et al., 2021).

In the Industry 4.0 concept, IoT can be used to develop so-called smart products. Things like the Internet of Things, the Internet of Services, the Internet of Energy, and others can be considered elements that create a link between the concepts of Smart City and Industry 4.0. Industry 4.0 can be considered as part of the Smart City concept (Lom et al., 2016). The development of

Industry 4.0 concepts and Smart Cities can solve problems related to electrical efficiency, urban production, demographic changes in large cities, and more.

This paper focuses on designing and implementing a system for collecting, processing, and visualizing meteorological data, as well as developing and implementing an IoT device that will provide and send this data. The solution assumes that using server containers will allow for efficient scalability and adaptability of the system when performance increases are needed. The server solution will also include functionality for monitoring and managing the status of the weather station and its sensors, including an automatic notification system for administrators when a malfunction of the device or sensors is detected. This project is a first attempt at creating a system to collect meteorological data. In the future, we plan to include a neural network for predictive capabilities.

The design phase will primarily focus on the weather station connectivity and ability to process and send data to server as well as communication architecture of the whole system will be created. The selected software and hardware solutions will reflect current technology standards and free open source technologies in the field.

2 Materials and Methods

The use of meteorological data in the Smart City concept is indispensable. Mainly because the environmental conditions in and around the city directly impact the comfort and quality of life not only of the inhabitants but also of the large number of working people who commute to the cities every day for work. Environmental conditions are also attractive for working people who commute to cities for work, perhaps even more interesting. According to the European Commission (EC), 15 percent of workers commute to cities for work (European Commission, 2019). This shows the importance of building information systems and making measured data available on the Internet so that it is also accessible to people who do not live in the city but work there. Measuring, collecting, and evaluating meteorological data in different parts of the city is even more important as cities and their inhabitants will have to face the challenges of extreme weather events, climate change, and their impact on urban infrastructure and inhabitants. High temperatures and poor air quality can significantly affect the health of urban residents. Smart City concepts use advanced technological solutions to monitor and predict air quality, enabling cities to implement effective adaptation measures (Camarillo-Escobedo et al., 2022).

Measuring and monitoring air quality and other environmental data in Smart Cities is an essential part of mapping the environment in which residents live, work, and spend their leisure time. This data then needs to be stored and analyzed appropriately. For example, high temperatures and poor air quality can significantly affect the health of urban residents. The Smart City concepts use advanced technological solutions such as deep learning to monitor and predict air quality based on historical data, enabling cities to implement effective adaptation measures (Tahirkheli et al., 2021). Intelligent air quality monitoring systems in Smart Cities often use a combination of low-cost sensors and advanced algorithms to create accurate pollution maps. These systems allow cities not only to monitor current pollution levels but also to predict future pollution levels.(Bibri and Krogstie, 2020).

This allows cities to introduce targeted measures to reduce residents' exposure to air pollution, for example, at times of high concentrations of dust particles and high temperatures. With the detailed mapping of parts of the city, such alerts and warnings can be specified for individual mapped areas, and thus, specific measures can be developed for individual areas and zones within the city (Neo et al., 2023). Using meteorological data in urban traffic management is essential, especially during severe weather conditions. For example, research shows how deep learning-based models, Bekkar et al. use meteorological data to predict air pollution, which has a direct impact on traffic management and public health. These data allow cities to take precautions to minimize the impacts of pollution (Neo et al., 2023).

Meteorological stations are essential in the Smart Cities concept because they not only provide the necessary meteorological data but also support decision-making processes in managing environmental objectives and planning infrastructure development. They collect important weather information critical for traffic management, energy management and crisis response to extreme weather conditions. A fundamental need for Smart City development is to collect data on what is happening in the city (Bibri and Krogstie, 2020). In Smart Cities, IoT technologies integrate this data into city information systems, enabling more accurate forecasts and adequate responses to weather. Advanced models based on deep learning, studies show, effectively predict air pollutants by leveraging data from these stations, allowing cities to take preventive action to protect citizens' health. Thus, integrating these stations into Smart Cities is crucial for developing Smart City services that respond to citizens' needs and improve safety and the environment in urbanized areas (Cheon and Mun, 2023).

2.1 Proposal of weather station based on IoT concept

The proposed system will be designed to collect sensor data from IoT devices. The submitted manuscript consists of several parts. The design of the weather station hardware, the design of the sensors, and the physical connection to the server. The next part deals with

the server's design and the system's implementation to collect data from the microcontrollers and transmit it to the server using MQTT protocol (Message Queuing Telemetry Transport) and Node-RED. The last part of the paper discusses in detail the implementation of the InfluxDB database system for storing real-time sensor data in the database and visualizing it using Grafana. The paper aims to design a weather station data acquisition system for collecting, storing, and visualizing the measured data. Also, the system provides capabilities for subsequent data analysis of historical data that has been measured. Above, the server will have alarm values set when some environmental data values are exceeded and will alert the users by email. All software is stored in a containerized docker system for easy future modifications and to ensure consistency across multiple development, release cycles, and production environments.

For these purposes, we have made our own weather station prototype. This is the first prototype of the weather station and also of the proposed data collection system on the server. This is the first step in our effort to create a network of measurement points and to create a detailed map between weather stations based on environmental data in order to obtain detailed environmental data from the campus environment in order to implement our devices concerning the Smart City concept.

Once the data is received into the system, processing, and computation will take place, such as calculating other data, sending alerts based on the value of the given data, or modifying the data into the correct data structure before storing it. After the data has been processed, all the retrieved and calculated data will be stored in the database system. The system will also be used to monitor the logs from the system to ensure that they can be accessible if needed. The proposed data storage system will monitor whether there is communication between the device and the server. The last part of the system will be data visualization through dashboards. In this proposal, the microcontroller will be used in the role of an IoT device will be used to collect data for the server.

The device will be constructed as a small weather station that will provide all the necessary data about the current weather. The device will also monitor the voltage of the batteries from which it will be powered. A solar panel will also be connected to the device. In addition to the data from the weather station, the device will send the functional status of the individual sensors.

In the first part, we will discuss the theoretical knowledge about the hardware part of our system for both the server part and the microcontroller part. We will discuss the sensors used, the microcontroller and the device on which we will run the server part. We have chosen sensors that will allow us to measure air temperature, relative humidity, air pressure, wind speed and direction, and precipitation. In this section, we will discuss the different sensors that we use in our implementation.

2.1.1 Hardware components

ESP32 microcontroller selection
The ESP32 refers to a chip called ESP32. However, the term ESP32 is also used to refer to development boards that are built on ESP32 chips. Using pure ESP32 chips is neither easy nor practical, especially when learning, testing, or prototyping. In most cases, you will want to use an ESP32 development board. These boards already include all the necessary components to power and program the chip, connect the chip to the computer, connect pins to peripherals, the necessary antennas for Wi-Fi and Bluetooth, and other valuable conveniences. Also, some of these boards may already contain other useful devices such as sensors or modules, screens, or cameras (RTD, 2022).
Server based on Raspberry Pi 4
The Raspberry Pi is a single-board computer developed by the Raspberry Pi Foundation. It has a small footprint but enough power for classic uses such as web browsing or creating small home servers. Raspbian is recommended for classical device use up to operating systems designed for servers. There are also various other operating systems defined e.g. as emulators, systems designed for multimedia, but also for 3D printing and home automation. There are also several versions of the Raspberry Pi, ranging from small computers such as the Raspberry Pi 4B to even smaller microcontrollers called Raspberry Pi Pico, which are suitable for IoT devices whose size is negligible. The Raspberry Pi is an inexpensive computer running Linux, but it also provides a certain amount of pins that allow them to control electronic components for physical computing and IoT exploration (RaspberryPi Ltd., 2024). The Raspberry Pi is a low-cost computer running Linux, but it also provides a certain amount of pins that allow them to control electronic components for physical computation and IoT exploration. We chose Linux as the operating system, as most servers run on Linux, and it also presented an opportunity to learn something new. Among the Linux distributions, we chose Ubuntu Server, which can be considered as a distribution suitable for beginners as well as for professional use.
Temperature and humidity sensor DHT22
The DHT22 temperature and humidity sensor, known as the AM2302, has high accuracy in both temperature and humidity measurement. It consists of a capacitive part, which is used to measure relative humidity, and an NTC thermistor, which is used to measure temperature. Each sensor is calibrated at the time of manufacture. The small size of the sensor, the very low power consumption, and the possibility of data transmission via cable up to a distance of about 20 meters supports a large range of applications of this sensor, such as IoT devices (Sparkfun, 2024).
Temperature, humidity and barometric pressure sensor

BME280
The BME280 is a sensor that measures the environment's temperature, barometric pressure, and humidity. The sensor is suitable for a large number of applications, such as the measurement of environmental variables in weather measurement. This sensor can be connected to your device using I2C protocol or SPI. We can use the I2C protocol for simple wiring, but care must be taken to ensure that the sensors do not collide between I2C addresses. The module has its own pull-up resistor, connected between the power supply and the SDA/SCL pins by default. When using multiple sensors, we can change the I2C address by soldering a jumper on the back of the PCB. The module has a total of 4 pins: two for communication and 2 for power. The power supply can range from 1.8 to 3.6VDC, but typically it is 3.3VDC. (Adafruit.com, 2024)
Wind speed sensor WH-SP-WS01
The sensor named WH-SP-WS01 is designed for wind speed measurement, which works on the mechanical principle. The sensor is designed in a non-contact way using a magnetic reed contact. The sensor has a cable that is terminated with an RJ11 connector. The sensor can be used alone, but it can also be used in conjunction with a sensor for measuring wind direction, which has a separate connector for connecting this sensor. Subsequently, these sensors are routed through one telephone quad line. The wind speed is calculated based on the switching frequency of the sensor, where 1Hz equals 2.4km/h.
Precipitation measuring sensor MS-WH-SP-RG
The rainfall sensor works on the principle of a self-emptying rainwater collector. Therefore, there are no other sensors or devices below the sensor into which water could flow, thus harming them. Also, the sensor must not be covered by drain holes and should have sufficient space to absorb or drain water. For the most accurate sensing, the sensor should be positioned as horizontally as possible, which is what the small spirit level on the sensor is designed to do. The signal from the sensor is routed through an RJ11 connector cable.
Sensor for measuring electrical quantities INA219
The INA219 sensor is an integrated circuit developed by Texas Instruments to measure current and voltage in electrical circuits. This sensor is often used to monitor and control the power consumption of electrical equipment. Its main purpose is to provide an accurate and reliable way to measure input voltage and current consumption for a variety of applications, such as powering embedded systems, solar panels, batteries, motors, LED lighting, and other devices. The main purpose of this device is to provide an accurate and reliable way to measure input voltage and current consumption for a variety of applications, such as powering embedded systems, solar panels, batteries, motors, LED lighting, and other devices (Texas Instrument, 2015).

The sensor supports measurement of shunt voltages ranging from 0 to 26VDC. To power the device, we connect to the 3.3VDC input pin, and the device uses a maximum current of 1mA. The device can operate at temperatures ranging from -40°C to 125°C.

The sensor contains a total of 4 pins, 2 for power and two for communication using the I2C protocol. The sensor also includes a contact interconnect option on the back, allowing the I2C address of the device to be changed. The sensor contains a total of 4 usable addresses. The sensor also has a pull-up resistor on the back, which is connected to the basic setup.

2.1.2 Software components

MQTT protocol
The MQTT is a lightweight publish/subscribe message transfer protocol that runs on top of the TCP/IP protocol. MQTT is designed for orderly, lossless, and bidirectional connections for embedded devices, microcontrollers, and networks. The publish-subscribe model of the MQTT protocol is based on the Client-Server model, but the server on which MQTT runs is more like a broker or gateway. An MQTT client can send a message to a specific "Topic", or subscribe to messages to a specific " Topic". The MQTT Broker will then be in charge of processing the messages received and sending them to the correct device. The Broker will keep a list of Topics and process publish/subscribe requests from clients (Cui, 2017). For the implementation, we will also use a program called Mosquitto, which will serve as an MQTT broker that will allow us to receive data from the device and send it to the server. In this section, we will explain the program and the MQTT protocol. The MQTT protocol provides a lightweight method for transmitting messages that use the Publish/Subscribe model. This makes MQTT suitable for use in IoT devices such as sensors with low power consumption, mobile devices, and others such as cell phones or microcontrollers (Eclipse Foundation , 2018). First, we need to define the communication between the device and the MQTT Broker, which is done using Publish/Subscribe. The MQTT Broker then communicates with Node-RED in the same way. After processing the data in Node-RED, we will send an array to InfluxDB containing two JSON objects. In the first object, we will have all the data we want to store in each measurement in the database. In the second object, there will be auxiliary variables, called tags, that express auxiliary values, such as sensor names, the device name that provided us with the data, and others. The communication between InfluxDB and Grafana will be bidirectional so that when the display period changes, the data provided by the database will be updated.
Data collection using Node-RED
Node-RED is an open-source web programming tool that uses a flow-based approach to application programming. This is based on nodes, each of which performs a specific task, according to which it processes incoming data and sends it to the next node in the sequence. The user then creates the independent be-

haviour of the application by simply placing and linking nodes in a so-called data flow. In addition to being user-friendly, this approach also allows for clearer programming and greatly facilitates troubleshooting of potential problems. The Node-RED user environment is accessible via a web browser and provides a wide variety of nodes that perform basic automation tasks. Users can also create their own nodes or use community nodes. Storage of measured data

To store the measured data, we chose a time series database named InfluxDB.

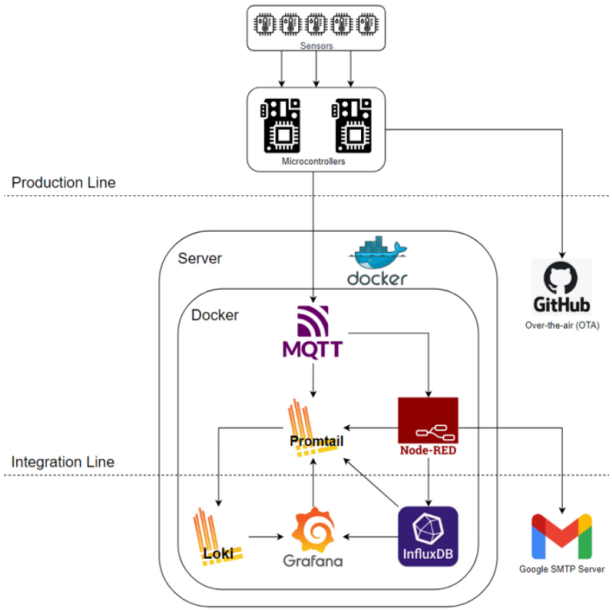

Figure 1: Proposed server software components

Data storage

InfluxDB was designed to store sensor data, calculated data, and logging logs from the application. InfluxDB is an open-source time series database written in the Go programming language and being developed by InfluxData. This database is optimized for faster, high-availability data retrieval and storage of time-dependent data in areas such as operations monitoring, IoT sensor data storage, and real-time analytics. SDBs are optimized to measure change over time. The characteristics that make time-dependent data different from other data are, e.g., data life-cycle and especially the large amount of different data (InfluxData, 2022).

To visualize the data for the user, we used the Grafana tool. Grafana is an open-source interactive data visualization platform created by Grafana Labs that allows users to visualize data (Redhat, 2024).

Storing and managing logs

The system logs are stored and managed by Grafana Loki and Grafana Promtail. Grafana Loki is a horizontally scalable log aggregation system built for log collection efficiency. It focuses on log streaming, storage, and indexing. Promtail is the agent responsible for collecting logs from various sources, such as system logs and Docker logs, and then sending them to Grafana Loki.

Promtail and Loki are usually deployed on each device running applications that need monitoring. Promtail and Loki are only responsible for collecting and storing logs. To use them, e.g., for visualization, we need to link Loki and Grafana together. Fig. 1 shows all software components running on the server together.

Hardware design and wiring

Fig. 4 shows the device installed. In total, there are five sensors that collect seven different quantities. The housing in the picture is then fitted over the BME280 and FHT22 sensors to protect them from the weather. The solar panel is mounted on a heavy concrete block, which protects it from blowing away and avoids the need to mount the panel directly on the roof. Up to this stage, the visualisation in Grafana has only been simple serial graphs. Once the latest version of the data collection device was installed, the creation of the main dashboard began, as described in the server-side implementation.

In the final version, the device's behaviour was changed when an error condition occurred or when sensors were not connected. The device's connection to Wi-Fi was redesigned, resulting in longer battery life, which in turn allowed more frequent data collection from every 15 minutes to 5 minutes. This was followed by incorporating rainfall amount collection from an approximate value to continuous rainfall amount data collection, even while the device was asleep.

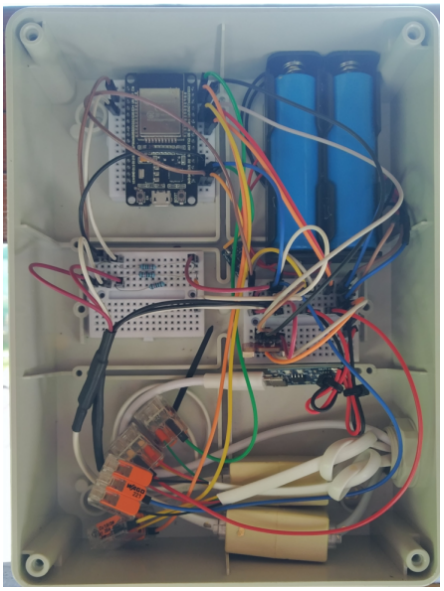

Figure 2: Hardware components housed in a box on a meteorological station.

Also, device error conditions, such as overflow of variables or division by zero, etc., were treated in the latest version. After fixing the flaws from the last version, the device was moved to a real environment where it collects weather data. The device is mounted on the roof along with the solar panel, and the box with

the microcontroller was placed under the roof.

3 Results

The weather data collection device has gone through several development versions to reach the final version that is currently implemented in the real environment. This final version includes several improvements and modifications based on experience and testing of previous versions.

The final device is equipped with several types of sensors to measure different meteorological variables. These sensors include the BME280 to measure temperature, humidity, and air pressure, the DHT22 to measure temperature and humidity, an anemometer and wind rosette to measure wind speed and direction, and a rain gauge to measure the amount of precipitation. These sensors are strategically placed to accurately and reliably collect data from the surrounding area.

Power for the device is provided by two 2500mAh 18650 batteries, which are wired in parallel to increase the overall capacity. The device is also equipped with a solar panel that provides battery charging, ensuring continuous operation of the device even in the absence of other power sources. The batteries are connected to a battery charger that regulates the charge from the solar panel to prevent overcharging or damage.

The device is housed in a SCABOX (I68) shown in Fig. 2 , which provides protection from external influences such as rain, dust, and extreme temperatures. Prior to installation, a hole was drilled into the box, and a grommet for the cabling was subsequently inserted into the box to ensure that the cables were protected from damage whilst allowing easy access for maintenance.

Figure 3: Final dashboard.

4 Conclusion

The entire equipment programme consists of four parts. The first part is the auth.py file, which contains the data for connecting to Wi-Fi and the MQTT Broker. The second part is the version.json file where the current firmware version of the device is stored. The third part contains all the libraries needed for the device to function. The fourth part contains the boot.py and main.py files, where the main code of the device is located. The

Figure 4: Meteo station

boot file connects to Wi-Fi and checks and updates the firmware. The main file initializes pins and variables, collects sensor data, processes it, and sends it to the MQTT Broker. Once connected to the MQTT Broker, the device sends the collected data, disconnects, and sleeps for 5 minutes. If an error occurs, such as an unconnected sensor or a failed connection to the MQTT Broker, the device also falls asleep for 5 minutes and then repeats the entire process.

Once all the improvements were implemented, the final version of the device was placed on the roof along with the solar panel, with the microcontroller box placed under the roof for protection from the weather. This placement allows optimal meteorological data collection, which is then processed, stored, and visualized on the server. The collected data include variables such as temperature, humidity, air pressure, wind speed, and direction, amount of precipitation, battery status, and the status of equipment and sensors. This ensures continuous and reliable collection and processing of meteorological data in real-time.

The latest version of the weather station is conceived as an IoT device that is powered by a solar panel and batteries that send data to a server every 5 minutes using the MQTT protocol. When collecting rainfall data, a time window of 5 minutes would represent a large inaccuracy, which is why this part was subsequently redesigned to allow for continuous rain data collection, even with the device in a sleep state and not measuring ambient variables. Furthermore, an OTA updates approach was used for the weather station, which allows a new firmware version to be uploaded to the weather station via Wi-Fi without having to access it directly.

In this study, we designed and built a specialized IoT-enabled weather station to advance monitoring environmental conditions within urban areas. We carefully selected and proposed hardware and software components that align with the IoT and Smart City

concepts. The weather station is equipped to gather, process, and visualize meteorological data effectively. It operates with a high degree of reliability, transmitting data every five minutes using MQTT protocol and maintaining functionality even during low-power conditions to ensure continuous rainfall data collection. Additionally, we implemented over-the-air firmware updates, enhancing the station's adaptability without needing physical adjustments.

Our proposal shows the integration of cutting-edge IoT technologies, utilizing MQTT, Node-RED, InfluxDB, and Grafana to enhance the functionality and efficiency of the weather station, thereby facilitating sophisticated data analysis and real-time environmental monitoring in urban settings.

The final deployment of the IoT meteorological station underscores a significant advancement in Smart City environmental monitoring. The system's ability to consistently transmit accurate data every five minutes via the MQTT protocol and its novel capability to collect continuous rainfall data even in low-power states confirm its operational efficacy and reliability. With over-the-air firmware updates facilitating effortless upgrades, the station remains at the forefront of technological innovation. This project proves the feasibility and critical importance of IoT systems in enhancing urban resilience and sustainability.

Acknowledgments

Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00562.

References

Adafruit.com (2024). Adafruit BME280 Humidity + Barometric Pressure + Temperature Sensor Breakout. Available online: https://learn.adafruit.com/adafruit-bme280-humidity-barometric-pressure-temperature-sensor-breakout/overview.

Bauer, M., Sanchez, L., and Song, J. (2021). IoT-Enabled Smart Cities: Evolution and Outlook. *Sensors*, 21(13):4511. Number: 13 Publisher: Multidisciplinary Digital Publishing Institute.

Bibri, S. E. and Krogstie, J. (2020). The emerging data–driven Smart City and its innovative applied solutions for sustainability: the cases of London and Barcelona. *Energy Informatics*, 3(1):5.

Camarillo-Escobedo, R., Flores, J. L., Marin-Montoya, P., García-Torales, G., and Camarillo-Escobedo, J. M. (2022). Smart multi-sensor system for remote air quality monitoring using unmanned aerial vehicle and lorawan. *Sensors*, 22(5).

Cheon, M. and Mun, C. (2023). The Climate of Innovation: AI's Growing Influence in Weather Prediction Patents and Its Future Prospects. *Sustainability*, 15(24):16681. Number: 24 Publisher: Multidisciplinary Digital Publishing Institute.

Cui, P. (2017). Comparison Of IoT Application Layer Protocols. Master's thesis, Auburn University, United States – Alabama. ISBN: 9798368493824.

Eclipse Foundation (2018). Eclipse Mosquitto. Available online: https://mosquitto.org/.

European Commission, u. (2019). Majority commuted less than 30 minutes in 2019. Available online: https://ec.europa.eu/eurostat/web/products-eurostat-news/-/DDN-20201021-2.

InfluxData (2022). Time Series Database (TSDB) Guide | InfluxDB. Available online: https://www.influxdata.com/time-series-database/.

Lom, M., Pribyl, O., and Svitek, M. (2016). Industry 4.0 as a part of smart cities. In *2016 Smart Cities Symposium Prague (SCSP)*, pages 1–6.

Ministry of Investments (2023). Slovakia's digital transformation strategy 2030 and digital action plan. Available online: https://www.smartcity.gov.sk/wp-content/uploads/2023/05/Ak%C4%8Dn%C3%BD-pl%C3%A1n-inteligentn%C3%BDch-miest-a-regi%C3%B3nov-na-roky-2023-2026.pdf.

Neo, E. X., Hasikin, K., Lai, K. W., Mokhtar, M. I., Azizan, M. M., Hizaddin, H. F., Razak, S. A., and Yanto (2023). Artificial intelligence-assisted air quality monitoring for smart city management. *PeerJ Computer Science*, 9:e1306. Publisher: PeerJ Inc.

RaspberryPi Ltd. (2024). Raspberry Pi 4 Model B specifications. Available online: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/.

Redhat (2024). What is Grafana? Available online: https://www.redhat.com/en/topics/data-services/what-is-grafana.

RTD (2022). Getting Started with the ESP32 Development Board | Random Nerd Tutorials. Available online: https://randomnerdtutorials.com/getting-started-with-esp32/.

Sparkfun (2024). RHT03 (DHT22) Humidity and Temperature Sensor Hookup Guide - SparkFun Learn. Available online: https://learn.sparkfun.com/tutorials/rht03-dht22-humidity-and-temperature-sensor-hookup-guide/all.

Tahirkheli, A. I., Shiraz, M., Hayat, B., Idrees, M. S., Sajid, A., Ullah, R., Ayub, N., and Kim, K.-I.

(2021). A survey on modern cloud computing security over smart city networks: Threats, vulnerabilities, consequences, countermeasures, and challenges. *Electronics*.

Texas Instrument (2015). Ina219 zero drift, bidirectional current power monitor with. Available online: https://www.ti.com/lit/ds/symlink/ina219.pdf.