# DEVELOPING A DATA MONITORING SYSTEM PROTOTYPE IN THE INTERNET OF THINGS APPLICATIONS

**Alexey Andrushevich[1,2], Dzmitry Protasenia[1], Iosif Vojteshenko[1]**
[1]Belarusian State University, [2]Lucerne University of Applied Sciences and Arts

***https://doi.org/10.5281/zenodo.7853502***

***Abstract.*** *This paper presents development process for a hardware and software prototype of Internet of Things application. As a take-away of presented work it can be noted, that the configuration features and technical characteristics of the used devices and technologies should be carefully taken into account while shaping the design of IoT applications.*

***Keywords.*** *Internet of Things, edge computing, distributed applications, rapid prototyping*.

## INTRODUCTION

The Internet of Things (IoT) is an ecosystem of interconnected objects, people, information systems and resources that are combined together with services processing and responding to information from both the physical and the virtual world [1].

In the modern common understanding, the IoT ecosystem includes "things" – objects equipped with RFID tags or sensors; data exchange protocols between sensors; distributed wired and wireless networks for data exchange; routers and gateways; cloud services; big data collection, analysis systems as well as information protection systems [2]. Thus, "things" can be both physical or virtual objects that can be identified and networked.

Due to the logistics limitations of free export / import of technical devices, in some cases software and hardware complexes in the field of Internet of Things have to be based on devices from different vendors. Moreover, sometimes the hardware does not have the most recent firmware versions and years of production. Also, with current restrictions on the use of cloud storage and services from the world's leading manufacturers, the role of edge nodes for data accumulation and preliminary processing in their own server storages increases significantly during the development of components and systems in the Internet of Things.

The main objective of this work was to develop and to demonstrate a prototype for distributed IoT application consisting of several edge nodes that collect data from temperature sensors and send them to the server for further processing. IoT edge computing is a new computing paradigm "in the IoT domain" for performing calculations and processing at the edge of the network, closer to the user and the source of the data [3]. The edge nodes of the system are single board computers from different vendors. The software of the prototype is based on a set of software tools with different functionalities.

## DATA MONITORING PROTOTYPE DESCRIPTION

A general scheme of the developed IoT monitoring application is shown below (Fig. 1).

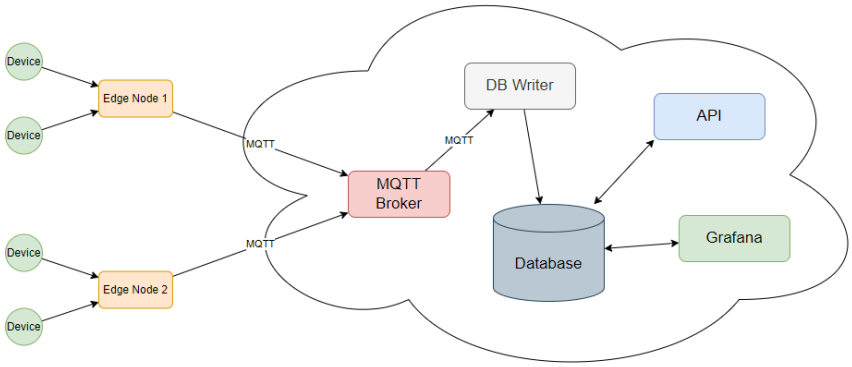


**Figure 1:** General scheme of data transfer and software interaction in the monitoring system

Each component of this system is described in more detail below.

**1. Edge nodes hardware description**

As edge computing nodes the authors have used two Raspberry Pi 1 Model B+ and one BeagleBone Rev A6 single-board computers (Fig. 2).

**1.1 The Raspberry Pi** single-board computer is often used in the development of prototype systems for the IoT [4, 5]. Our system uses Raspberry Pi 1 Model B+. It is the latest version of the first generation of Raspberry Pi single-board computer released in 2014. This computer has:

- 40 GPIO (General-Purpose Input/Output) pins allowing connection of various devices and expansion boards
- 4x USB 2.0
- 3.5mm audio output
- HDMI for connection to a monitor
- Ethernet socket
- microUSB socket for power supply

Raspberry Pi requires 5V power supply and about 1-2A current (power consumption depends on the computing and periphery load).

The data storage is provided by microSD card.

The operating system is installed on the microSD card using special Raspberry Pi imager software which can be found on the manufacturer's website (https://www.raspberrypi.com). By default, it is suggested to install the Raspberry Pi OS, which is based on Debian Bullseye.

If the board is used without a connected monitor, then the microSD card with operating system has to be inserted into the computer as a normal external storage. After that, it is possible to enter a special folder /boot (so called boot folder). Adding certain files to this folder will activate some functions during the initial boot of the Raspberry board.

For basic setup, a new user needs to be created. To do this, a text file userconf.txt is placed in the boot folder, which must contain one line of text in the format username:encrypted_password. The encrypted password must be obtained using the openssl passwd -6 command.

**Figure 2.** Raspberry Pi 1 Model B+ and BeagleBone Rev A6 single board computers

In order to have access to the board, SSH Server must also be enabled (it is disabled by default). To do this, an empty file named ssh must be created in the boot folder.

**1.2 The BeagleBone** Original Rev A6 single-board computer (fig.2) was manufactured in 2012 (https://beagleboard.org/). The processor used here is the AM3358 from Texas Instruments with a frequency of 720 MHz.

Device specification:

- 2 slots with 46 GPIO pins each
- 1x USB 2.0
- 1x mini-USB
- Ethernet
- Barrel jack for power.

One may notice that the hardware is a little different from the Raspberry. There are much more GPIO pins, which in theory allows to connect more devices.

However, this single-board computer has only one USB and it lacks a HDMI connector, which makes it impossible to connect I/O devices or a monitor to it and to use it fully as a standalone computer.

The microSD card is also used for data storage in this case.

There are 2 ways of powering this computer: via mini-USB or via the barrel jack. The barrel jack supplies the standard voltage and current of 5V and 1-2A. For powering via mini-USB the current limit is 0.5A, so the barrel jack is recommended for more stable operation in the field.

A bootable microSD card was generated to install the operating system (this can be done using any auxiliary software utilities such as Rufus or even the Raspberry Pi utility).

**2. Software technologies and components description**

**2.1 The MQTT (MQ Telemetry Transport)** protocol [6] has been chosen to transmit sensor readings. It operates on a publisher / subscriber pattern. The sensors publish a message to the appropriate channel (topic) and the subscribed listening clients receive the message and process it.

In order to work with the MQTT protocol, a so-called broker is needed, which directly implements the communication between the clients. Eclipse Mosquitto, a lightweight open-source broker, was chosen as the broker [7].

**2.2 TimescaleDB (an extension of PostgreSQL DBMS)** [8] was used for data storage. TimescaleDB is a relational database optimized for the use of temporal sequences. It optimizes the

storage of temporal sequences by using hypertables, which are virtual representations of a set of individual tables (chunks).

**2.3 DBWriter description.** To write data from the broker to the database we need another object called DB Writer. It is a MQTT client that listens to a specific topic. When DBWriter receives a message, it processes a received message and writes the data into the database. In our prototype, this object is implemented as a Python script (Fig. 3).


**Figure 3**: Diagram of the DB Writer object

**2.4 A REST API** has been implemented to manage the system objects and obtain the necessary information from possible client applications.

The API component developed in this paper works over HTTP protocol and the main data transfer format is JSON.

When designing the API component according to the original database schema, its main functional tasks were to implement the following methods:

- creation, modification and deletion of "Station" objects
- creation, modification and deletion of "Device" objects
- obtaining information about all "Station" objects
- obtaining information about one "Station" and all its "Devices" via its unique identifier
- obtaining additional information about a "Station", such as the average temperature and humidity
- obtaining readings from a "Device" by its unique identifier over a specific period

Further it is possible to add other necessary methods in the future.

The REST API component was developed in Python programming language using FastAPI libraries for implementing client-server interaction and SQLAlchemy for interaction with TimescaleDB database. The ORM (Object-Relational Mapping) approach was selected when using SQLAlchemy library. It allows to interact with the database through the language objects.

**2.5 Grafana** [9] was used to visualise the collected data. With its help dashboards were created and panels were placed on them to display the information. Visualised data were extracted from the database using SQL queries that additionally used both special macros provided by Grafana and functions of the TimescaleDB DBMS.

**CONCLUSION**

In this work, we have designed and implemented a prototype of hardware and software system for one of the basic tasks in the field of Internet of Things, namely, taking readings from sensors, processing them and sending to the global storage for further operations with them (visualization, statistics, abnormality tracking, etc.). Different technologies have been used to solve this comprehensive task.

Using Raspberry Pi and BeagleBone single-board computers as examples, we have performed configuration and comparative analysis of these computers and revealed their usage features as edge nodes within one system.

The MQTT protocol was used to organize the data transfer between the edge nodes and the server. To store the transferred data, a special time series database (TimescaleDB) optimized for storage and manipulation of temporal data was used.

The Grafana tool was used to visualize the collected data. Queries were developed to display the necessary information (e.g. averages over a certain period) in the form of graphs in panels.

A REST API was developed to manage key objects and retrieve the necessary information from possible client applications.

Based on the work done and the data obtained, it can be concluded that it is possible to build an IoT application based on different devices. However, the configuration features and technical characteristics of the used devices should be taken into account while shaping the design of IoT applications.

**REFERENCES**

1. ISO/IEC 20924:2018 Information technology - Internet of Things (IoT) - Vocabulary [Electronic resource]. - Mode of access: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/06/94/69470.html . - Date of access: 18.03.2023.
2. Fundamentals of Internet of Things technologies / V.I. Dravitsa [et al]. -Minsk: RIBSH, 2022. - 108 c. (In Russ)/
3. Pérez, J., Díaz, J., Berrocal, J. et al. Edge computing. Computing 104, 2711–2747 (2022). https://doi.org/10.1007/s00607-022-01104-2.
4. T. Gizinski and X. Cao, "Design, Implementation and Performance of an Edge Computing Prototype Using Raspberry Pis," 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 2022, pp. 0592-0601, doi: 10.1109/CCWC54503.2022.9720848.
5. Alexey A. Andrushevich, Vardak Ahmad Samir, Iosif S. Vojteshenko. Prototype of hardware and software complex for control of “smart home” applications // Information Systems and Technologies: Proceedings of the International Scientific Congress on Informatics. In 3 parts. Part 2, Rep. of Belarus, Minsk, Oct. 27-28, 2022. - Minsk: Belarusian State University, 2022. - P. 231-237. (In Russ).
6. MQTT [Electronic resource] - Access mode: https://mqtt.org/.- Date of access: 25.03.2023.
7. Eclipse Mosquitto Documentation [Electronic resource] - Mode of access: https://mosquitto.org/documentation/- Date of access: 30.03.2023.
8. Timescale Documentation [Electronic resource] - Mode of access: https://docs.timescale.com/- Date of access: 28.03.2023.
9. Grafana Documentation [Electronic resource] - Mode of access: https://grafana.com/docs/grafana/latest/- Date of access: 29.03.2023.