IBIMA Publishing
Communications of International Proceedings
https://ibimapublishing.com/p-articles/46ICT/2025/4626625/
Vol. 2025 (35), Article ID 4626625

VoIP Network and Protocols in IoT Applications*

Remigiusz RAJEWSKI and Marek WOJTASIK

Institute of Communication and Computer Networks, Faculty of Computing and Telecommunications,
Poznan University of Technology, ul. Polanka 3, 61-131 Poznan, POLAND

Correspondence should be addressed to: Remigiusz RAJEWSKI, remigiusz.rajewski@put.poznan.pl

* Presented at the 46th IBIMA International Conference, 26-27 November 2025, Ronda, Spain

Abstract

This paper presents the application of VoIP technology in the Internet of Things (IoT) environment, with particular emphasis on the integration possibilities of both technologies. The potential of using Raspberry Pi as a VoIP server and as a control hub for smart offices and homes is discussed, highlighting its low cost, efficiency, and flexibility. Key protocols such as SIP, RTP, and SDP, which enable effective voice and signaling communication in IP networks, were used. Various server environments are compared, identifying Asterisk as the most optimal solution for small and medium VoIP installations integrated with IoT, as opposed to alternatives such as FreeSwitch or 3CX, which may require more resources. In addition, the implementation of passive devices based on ESP32 microcontrollers is described, which enable functions such as alerting, remote device control, and collecting and processing sensor data. Practical applications in smart home and office environments, such as process optimization, increased security, and the ability to remotely manage systems, are highlighted. The results of the study confirm that the combination of VoIP and IoT technologies can significantly improve automation and communication in smart systems, while minimizing implementation costs and allowing further development toward more advanced solutions.

Keywords: IoT sensors, VoIP, microcontroller ESP32,

Introduction

In recent years, communication technologies have experienced rapid development, largely due to advances in digitalization and the global growth in Internet usage. Two particularly important technologies that have gained prominence are Voice over Internet Protocol (VoIP) and Internet of Things (IoT). VoIP enables real-time audio transmission via IP (Internet Protocol) networks, while IoT enables the integration of everyday devices with the Internet, enabling their interconnection and process automation.

VoIP technology enables voice transmission across IP-based networks, providing the foundation for modern digital communications. Its effectiveness depends on a set of protocols and standards that ensure both signaling and connection establishment, as well as transport and quality control of transmitted data. The challenges associated with real-time voice transmission, such as latency, jitter, and packet loss, require the use of appropriate mechanisms and technologies that minimize their impact on communication quality. In the context of the Internet of Things, additional requirements arise for performance, energy efficiency, and transmission reliability, further emphasizing the importance of appropriately selected protocols and standards.

Example protocols used in VoIP are as follows:

Cite this Article as: Remigiusz RAJEWSKI and Marek WOJTASIK, Vol. 2025 (35) “VoIP Network and Protocols in IoT Applications" Communications of International Proceedings, Vol. 2025 (35), Article ID 4626625, https://doi.org/10.5171/2025.4626625

- **SIP** – The Session Initiation Protocol is used to initiate, maintain, and terminate communication sessions, including voice, video, and messaging applications. SIP is used in Internet phone, private IP phone systems, and phone calls over LTE (VoLTE) networks – see Alan B. Johnston (2015).

- **SDP** – The Session Description Protocol is a network communication protocol that is used to describe the parameters of multimedia sessions, such as video, audio, and real-time data transmissions – described in details in RFC4566 (2006). SDP is used to negotiate and establish sessions in protocols such as SIP or Real Time Streaming Protocol (RTSP). This protocol defines key information, including media types, codec formats, IP addresses, ports, and session duration, to ensure a proper connection for all participants.

- **RTP** – The Real-time Transport Protocol is a network protocol that is used to transmit multimedia data in real time, such as audio, video, and sensory data – described by H. Schulzrinne, S. Casner, R. Frederick, and Van Jacobson (2003). It is a key element of transmission in applications such as video conferencing, VoIP, and live streaming. RTP provides mechanisms for synchronization, data segmentation, and media type identification, enabling the efficient transmission of multimedia streams over IP networks. Combined with RTCP (RTP Control Protocol), it also enables real-time monitoring of transmission quality and problem diagnosis, ensuring smooth and reliable communication.

The use of VoIP in the IoT opens new perspectives for efficient communication between devices and their integration with existing networked telecommunications systems. These solutions are used in areas such as smart homes and offices. The combination of these technologies allows for remote device control and device status indication, but also for the transmission of voice data in an optimized and secure manner.

The primary goal of this paper is to demonstrate that VoIP can significantly enhance IoT systems by enabling secure, real-time communication and automation in smart homes and offices at minimal cost. It aims to show how voice signaling can be used not just for calls, but also as a reliable trigger for physical actions (e.g., opening gates, sounding alarms) and for environmental monitoring (e.g., temperature and humidity alerts). The research validates a low-cost, scalable architecture that enables everyday devices to interact intelligently via standard IP networks, improving security, convenience, and operational efficiency. Ultimately, it proves that integrating VoIP with IoT opens new possibilities for remote control, instant notifications, and data-driven automation, paving the way for more advanced, interconnected smart environments.

The rest of the paper is as follows. Section 2 presents all the devices used and their role in our experiment. In turn, Section 3 present the software implemented on microcontrollers for incorporation into the VoIP system. Finally, Section 4 presents a proposal for integrating VoIP with the Internet of Things, presenting various applications and ways to use intelligent sensors, along with the development of their software.

Testbed

To practically implement and verify assumptions regarding the use of VoIP technology in signaling within the IoT, it was necessary to prepare an appropriate test environment. This consisted of a fully functioning small VoIP server, ESP32 microcontrollers, VoIP phones, and the required peripherals for testing. The selected solutions were also evaluated and analyzed to present an optimal design configuration that could be replicated under less demanding conditions.

Hardware peripherals are essential for testing the proposed solutions, such as VoIP telephones to check the possibility of making calls or Multipoint Control Units (MCU) to verify the correct operation of the created scripts integrating devices with the VoIP system.

VoIP Phones

The Tiptel IP280 (see Fig. 1) features a minimalist and compact design, making it a simple and reliable IP phone ideal for small and medium-sized office environments where ease of use and basic VoIP functionality are key. It supports essential office features such as speed dial, caller ID, and a basic address book, giving users easy access to key contacts.

Thanks to SIP support and compatibility with popular IP Private Branch Exchang (PBX) systems, the Tiptel IP280 is suitable for most standard VoIP deployments. It offers good sound quality thanks to the built-in noise reduction, significantly improving call comfort. It also has a basic set of conferencing features that enable fast three-way calls, which can be invaluable in a business environment where interdepartmental collaboration is crucial – see Tiptel (2025) for details.

Fig. 1. Used device – the Tiptel IP280 phone

Compared to more advanced IP phone models, the Tiptel IP280 stands out for its simplicity and affordability, making it a good choice for organizations seeking a cost-effective, easy-to-use device that meets basic telecommunications needs. Although it does not offer the advanced features of the more expensive models, its reliability and intuitive operation make it an effective tool for everyday office communications.

This model features a clear and easy-to-use web interface that allows for real-time monitoring of the phone's performance and customization for VoIP telephony applications.

It should be mentioned that the choice of VoIP phone used in the system is purely subjective, as each VoIP phone will perform a similar function in the same or similar way.

ESP32 microcontrollers

The ESP32 (see Fig. 2) is a System-on-a-Chip (SoC) manufactured by the Chinese company Espressif Systems. The ESP32 is based on the Xtensa LX6 dual-core processor with a clock speed of up to 240 MHz, has up to 520 KB of SRAM and supports Flash memory, allowing for the storage of larger programs and data. Provides significant computing power, enabling the simultaneous execution of multiple complex tasks. This allows it to support applications requiring fast data processing, such as voice recognition, real-time data analysis, and complex control algorithms. It also offers advanced connectivity options. The built-in Wi-Fi (802.11 b/g/n) and Bluetooth (classic 4.2 and BLE) modules enable seamless communication with other devices and easy integration with IoT networks – see FORBOT (2025) for more details. This is particularly useful in home and industrial automation environments, where devices need to connect to a variety of networks or controllers without the need for additional connectivity modules. With numerous communication interfaces, such as UART, SPI, I2C, and CAN, the ESP32 is extremely flexible and can communicate with a variety of peripherals, from sensors to displays, which makes it suitable for both simple projects and more complex systems – see ESP32 (2025) for more details. In terms of energy efficiency, the ESP32 offers multiple sleep modes, making it a

suitable choice for battery-powered projects requiring extended operation in the field or in locations with limited access to electricity sources. Thanks to this combination of high performance, versatile communication options, and energy-efficient modes, the ESP32 is an ideal tool for a wide range of projects, from simple hobby applications to more advanced commercial solutions. Because of these features, we used these microcontrollers in our project.

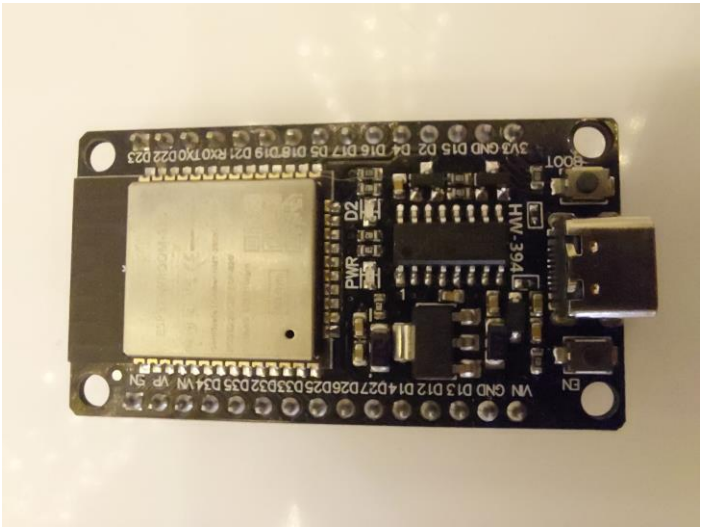

Fig. 2. Used device – the ESP32 microcontroller

The ESP32 microcontrollers in the project act as passive clients in the system. These devices are designed to register with the VoIP server and, upon establishing a connection, execute a previously written script that performs certain defined tasks. Primarily, the modules send feedback data, such as readings from various sensors in the form of HTTP page updates and previously implemented add-ons in the Grafana environment. Other tasks are the opposite of the previously mentioned functionalities, such as establishing a connection to a given active (or passive) client after a given value is read by a sensor, such as a motion sensor, occupancy sensor, IR camera, or microphone in listening mode.

PC computer

A key role in the project was played by a PC, which served as the central unit to run softphone software, in this case the Linphone – see Linphone (2025). The computer used in the project had a standard hardware configuration, sufficient to run the Linphone application and ensure adequate performance for the planned tasks.

As one of the most popular softphone solutions, the Linphone was chosen for its compatibility with the SIP protocol, its clear interface (see Fig. 3), and the ability to customize the configuration to meet the needs of our project. This software ran on a PC, enabling voice calls within the system and testing the integration functionality with IoT devices. The support of SIP signaling was crucial for ensuring communication between devices and their integration with the ICT system.

VoIP Server Hardware

The Raspberry Pi 4 is a miniature single-board computer created by the Raspberry Pi Foundation, offering a full-fledged computing environment in a small form factor and at a low price. This device offers sufficiently powerful components, including: quad-core 64-bit ARM Cortex-A72 processor running at 1.5 GHz, 4 GB of LPDDR4 RAM, two micro-HDMI ports (up to 4K), four USB ports (2x USB 3.0, 2x USB 2.0), an Ethernet port (up to 1 Gb/s), Wi-Fi, and Bluetooth modules. The Raspberry Pi 4 can be effectively used as a VoIP server for making voice calls over the Internet for several key reasons:

- **Performance** – Compared to previous models, the Raspberry Pi 4 has significantly more processing power, allowing it to run VoIP server software (such as Asterisk or FreePBX) and easily handle voice call processing, call management, and other IP telephony functions.

- **Low Cost** – The Raspberry Pi 4 is an extremely cost-effective solution, making it an attractive choice for small businesses or home users looking to build a low-cost, yet functional VoIP system.

- **Integrated Network Modules** – Thanks to the Wi-Fi module and 1Gb/s Ethernet port, the Raspberry Pi 4 can be easily integrated with a local network and the Internet, which is crucial for a stable VoIP call service.

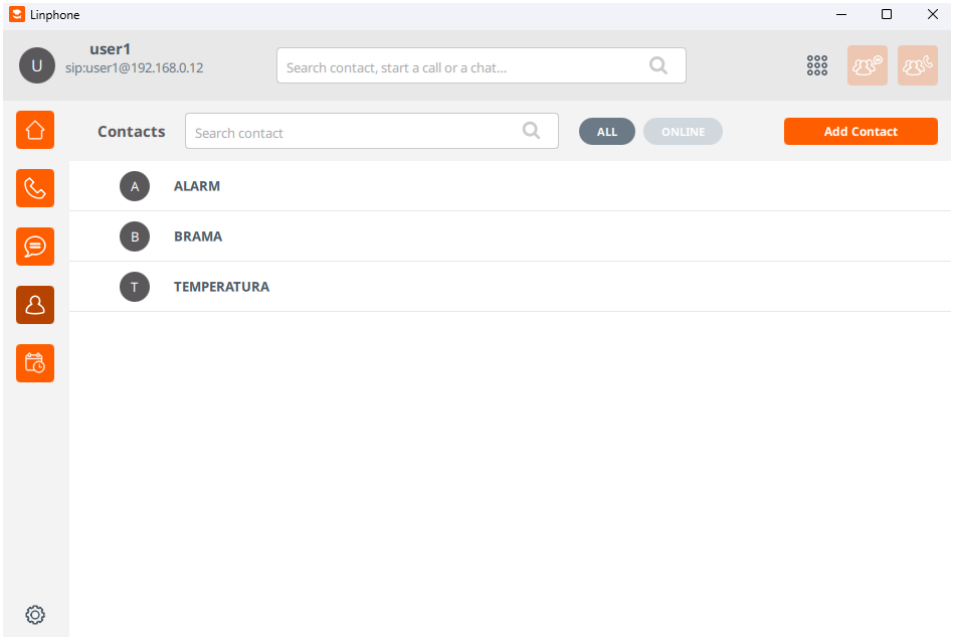

Fig. 3. Linphone program interface

- **Software Stability and Availability** – The availability of free open source software (such as the Asterisk) allows for easy configuration of a functional VoIP server that will support multiple users and features such as call forwarding, voicemail, call recording, etc.

The Raspberry Pi 4 can also act as a central management hub for a smart office, an environment where IoT technologies are integrated to automate and optimize various processes, such as lighting, air conditioning, security, and communication systems. Raspberry Pi 4 meets these requirements due to the following:

- **Low power and high efficiency** – The Raspberry Pi 4 consumes very little power, making it ideal for continuous operation, which is important for office automation systems that must be available 24/7.

- **Ease of programming** – The Raspberry Pi 4 supports multiple programming languages (e.g. Python), allowing us to easily write scripts to manage various IoT devices: from smart temperature sensors, through lighting, to security systems.

- **Extensive networking support –** Thanks to Wi-Fi and Ethernet modules, the Pi 4 can communicate with various wireless devices, which is crucial for integration with IoT devices. The Ethernet port ensures a stable wired connection to the local office network.

The Raspberry Pi 4 is one of the most popular hardware solutions for small SIP servers, and its flexibility and low price attract users looking to implement compact and cost-effective VoIP communication systems. Alternatives to official Raspberry Pi models include various clones, such as Orange P (see OrangePi 2025)), Banana Pi (see BananaPi (2025)) or Odroid C4 (see Odroid (2025)). These often offer similar architecture, but with improved components, such as better cooling, faster memory support, or additional interfaces, or offer

lower prices consistent with the guaranteed quality of the device. Such solutions can be an interesting option for users seeking greater performance or lower price while maintaining compatibility with the Raspberry Pi ecosystem.

VoIP Server Software

Asterisk is an open source software platform for building internet telephony systems. It is a highly flexible and versatile platform that can be configured as a telephone switchboard (Private Branch Exchange – PBX), conference server, VoIP gateway, or call center server. The software was created by Mark Spencer and is developed by Sangoma Technologies and the open source community – see Asterisk (2025) for more details.

Asterisk is one of the most popular open source solutions for building VoIP systems, but it is not the only option available. With the development of technology and the growing demands of users, many alternative platforms have emerged, offering different approaches to voice communication and VoIP signaling, such as FreeSwitch (see FreeSwitch (2025)), 3CX software (see 3CX (2025)), and Kamailio (see Kamailio (2025)).

Experiments

In our experiments, we used a few IoT devices (mostly sensors) that communicated with each other using modified VoIP protocols tailored to our needs. The used infrastructure is a typical telecommunication or computer network that can support VoIP. Therefore, there is no problem in extending the local test solution into a large network as well. Generally, the experiment network can be illustrated as shown in Fig. 4. In turn, the diagram describing how our experiment works step by step is shown in Fig. 5. We use VoIP messages; however, we modify them for our purposes to support communication with IoT sensors. Each device was responsible for other functionality as described in this section.

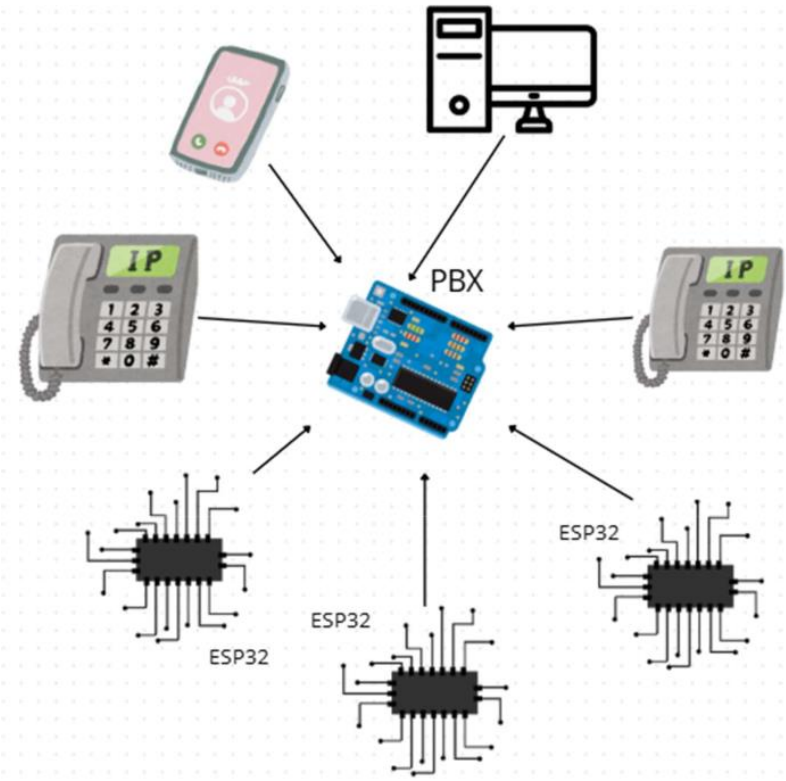

Fig. 4. Devices used in experiments

First Passive Device – Alarm Function

When it comes to building security, alarming, and information collection are crucial to improving operational efficiency by reducing costs and utilizing resources more efficiently. When unexpected actions occur, such as unexpected guests entering the building, we expect to be notified in some way.

The first design element of the system that integrates IoT devices with VoIP telephony is a device that alerts people when they enter the building. This was achieved by implementing simple SIP signaling traffic management on an ESP32 microcontroller. To this end, we wrote a C program in the Arduino IDE, connecting it to a contact door sensor.

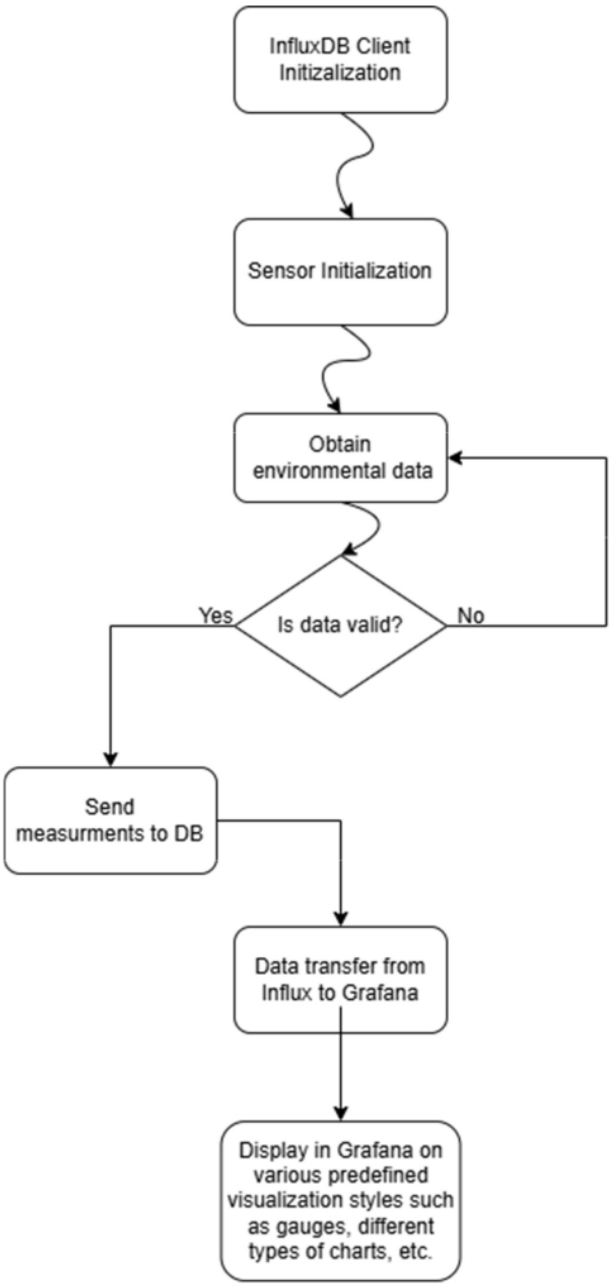

Fig. 5. Diagram describing how the experiment works step by step

Correctly reading whether a door or window is open is possible by connecting a digital pin set to the input state. When a state other than the low state is read on the connector, an open state occurs, which means the magnet and the actuator are actually separated, opening the door. This process is very similar to stopping to press a physical button. For this reason, in the diagram shown in Fig. 6, and the prototype version of the device itself is presented in Fig. 7, as well as in the program, the actuator will be represented as a button.

The received 401 Unauthorized response contains key information for user authentication, such as *realm* and *nonce*, which are saved in the microcontroller's temporary memory using the *extractRealmAndNonce* function.

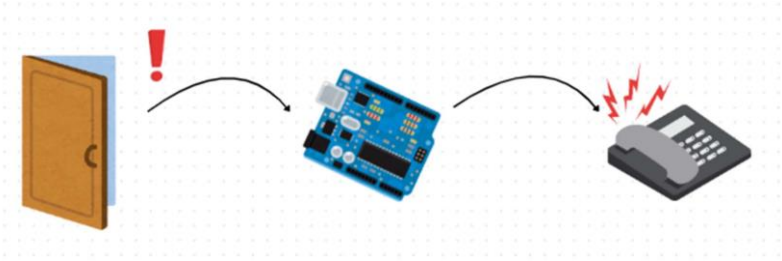

Fig. 6. Testbed of the first passive device

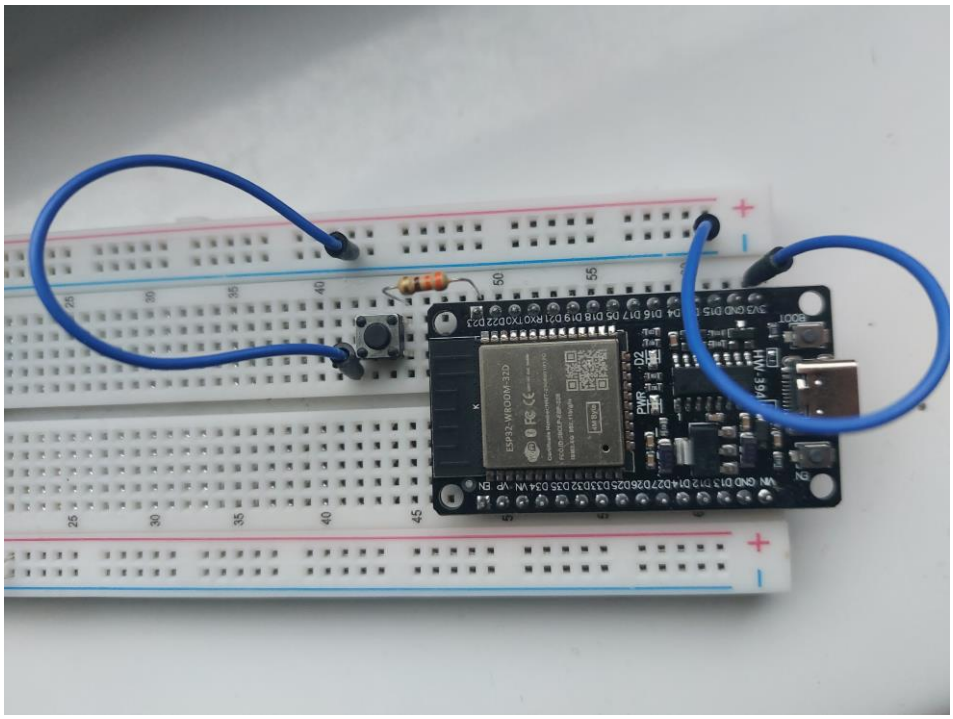

Fig. 7. Breadboard of the first passive device

The next function responsible for correctly sending the REGISTER authorization message is the *sendRegisterWithAuth* function. This function uses the simple *calculateMD5* function, also implemented by us. The *calculateMD5* function fully utilizes the previously implemented library MD5Builder.h and calculates a hash function compatible with the MD5 algorithm – described by R. Rivest (1992).

In the *sendRegisterWithAuth* function, the first step is to calculate the *Response* parameter of the body of the returned |REGISTER message using nested functions. Then, the function behaves similarly to the *sendRegister* function, but this time includes the elements responsible for authorization. Finally, it signals the programmer that the registration task has been completed. Thanks to the implementation method based on repeated authorization, if authorization is not successful, the *sendRegisterWithAuth* function is called again.

The *sendInvite* function, begins by calculating a unique *Call-ID* for this call to avoid parameter duplication, so that the server does not recognize too many requests with the same *Call-I* and return 500 Server Internal Error. This is followed by a similar sequence as during registration, but with the typical message content of the INVITE message body.

After sending the first INVITE request, excluding the authorization portion, the program waits 2 seconds to receive the 401 Unauthorized status message. After this, it calculates the values responsible for request authorization using the previously mentioned mechanisms. Afterwards, the response responsible for request authorization is calculated in the same way as during registration.

Then, the ACK message is sent, which is necessary for the proper flow of messages within the SIP protocol. This message is sent using the nested *sendACK* function, which contains the initial line and message body typical of a typical ACK message in the SIP protocol. The final step is sending the INVITE request, already populated with the calculated and substituted authorization parts. This is handled by the *sendInviteWithAuth* function, which does nothing other than populate the INVITE message and send it. At this point, the caller client receives a notification of an attempted call from the number assigned to the alarm device, signaling an intrusion into the building.

Second Passive Device – Remote Gate Opening Function

The second passive device implemented in the IoT integration system with Internet telephony is a device designed as a module located next to the electric gate controller. The opening of the gate conceptually occurs by receiving a high state on the gate controller input, which is also the module output: one of the digital pins of the ESP32 microcontroller.

As part of the concept presentation, a diagram was proposed for connecting a microcontroller with an LED signaling gate opening. This concept is shown in Fig. 8. In turn, Fig. 9 shows the physical implementation of this circuit on a breadboard with all the electronic components installed.

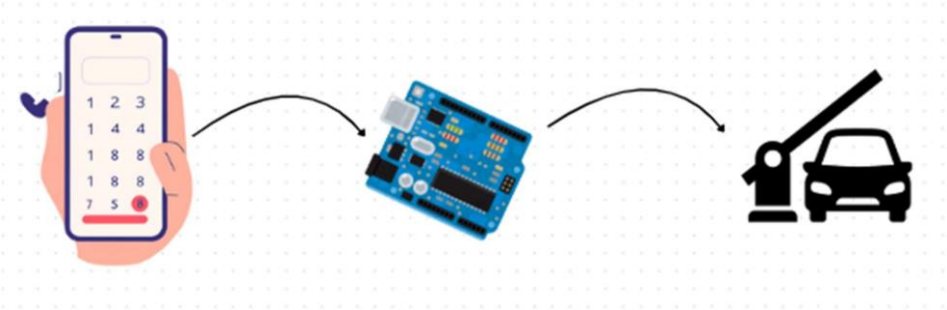

Fig. 8. Testbed for the second passive device

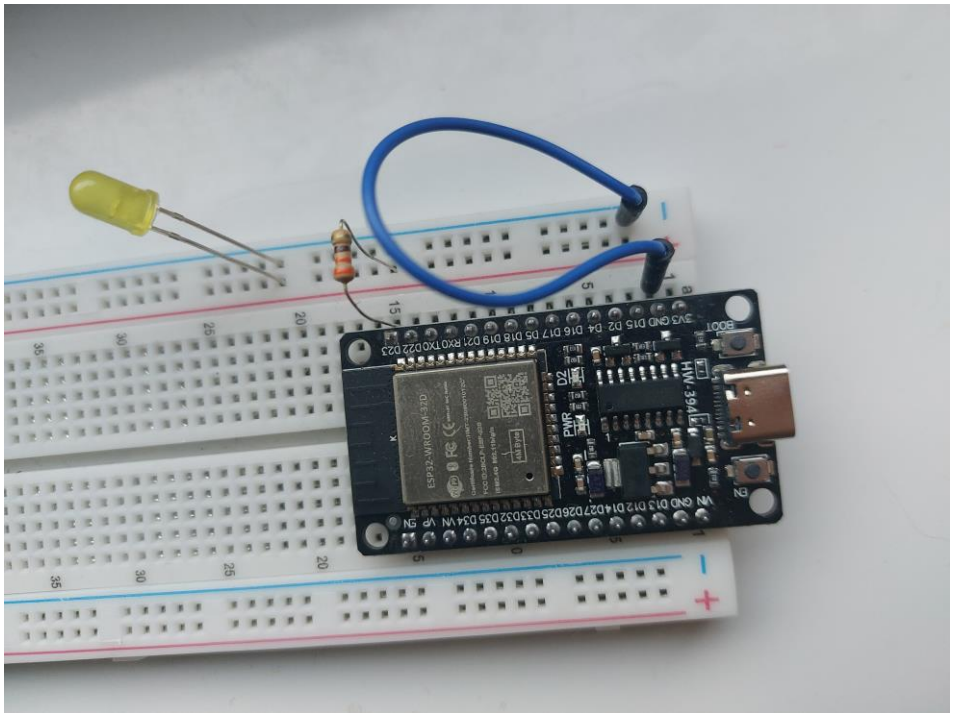

Fig. 9. Breadboard of the second passive device

The VoIP signaling program also needs to implement the functionality of a basic SIP client. However, this time the operation differs from the first passive client because of how the program is supposed to work. The gateway should open when a user registered with the module is called. The gateway opens upon receiving each INVITE request from each user of the system. The program still supports registration with authorization and re-registration after 50 minutes, although call initiation support is no longer required. However, support for receiving call initiation messages and performing an action upon receipt of such an INVITE message is required. The registration, authorization, and associated mechanisms remain the same as for the first device, so there is no need to repeat them.

Together with the built-in core of network operation support with the current VoIP server, the program code should be enriched with global variables assigning the operation of the diode imitating the gate controller input and ensuring multiple reactions to the same INVITE message.

To avoid leaving a session unfinished, waiting for the response timeout to be met, the *sendBusyHere* function was also proposed, which sends information to the server about the client's busy status. This function uses the same logic as the previously mentioned *send* functions, but differs in the initial status line and the message body typical of this type of message.

Third Passive Device – Data Collection and Alarming

In modern office spaces and smart homes, in addition to ensuring secure access to spaces inside buildings and performing remote actions, we also need to continuously monitor the operating parameters of devices or appliances coexisting within the building. According to this idea, another IoT module was designed to collect current data on ambient temperature and humidity and to trigger an alarm when the target range is exceeded. The diagram is shown in Fig. 10.

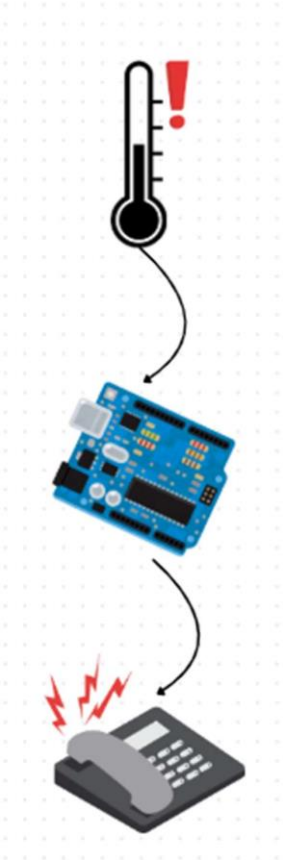

Fig. 10. Testbed for the third passive device

Principle of Operation

The module described here consists of an ESP32 microcontroller with a DHT11 digital temperature and humidity sensor connected to it. The structure of the sensor and the MCU assembly is shown in Fig. 11, where the temperature and humidity sensor is the blue component on the breadboard. The sensor is powered directly from the MCU board and requires only a single digital input to read the data. The sensor was chosen for its ease of use and very low price, which reduces the overall cost of the entire module. Despite all its advantages, the sensor has a low data resolution of 1°/1%, which is not a problem in the design assumptions. In practice, this module should be installed in situations requiring the monitoring of parameters, such as in a boiler room to monitor furnace temperature (the device was designed with this in mind).

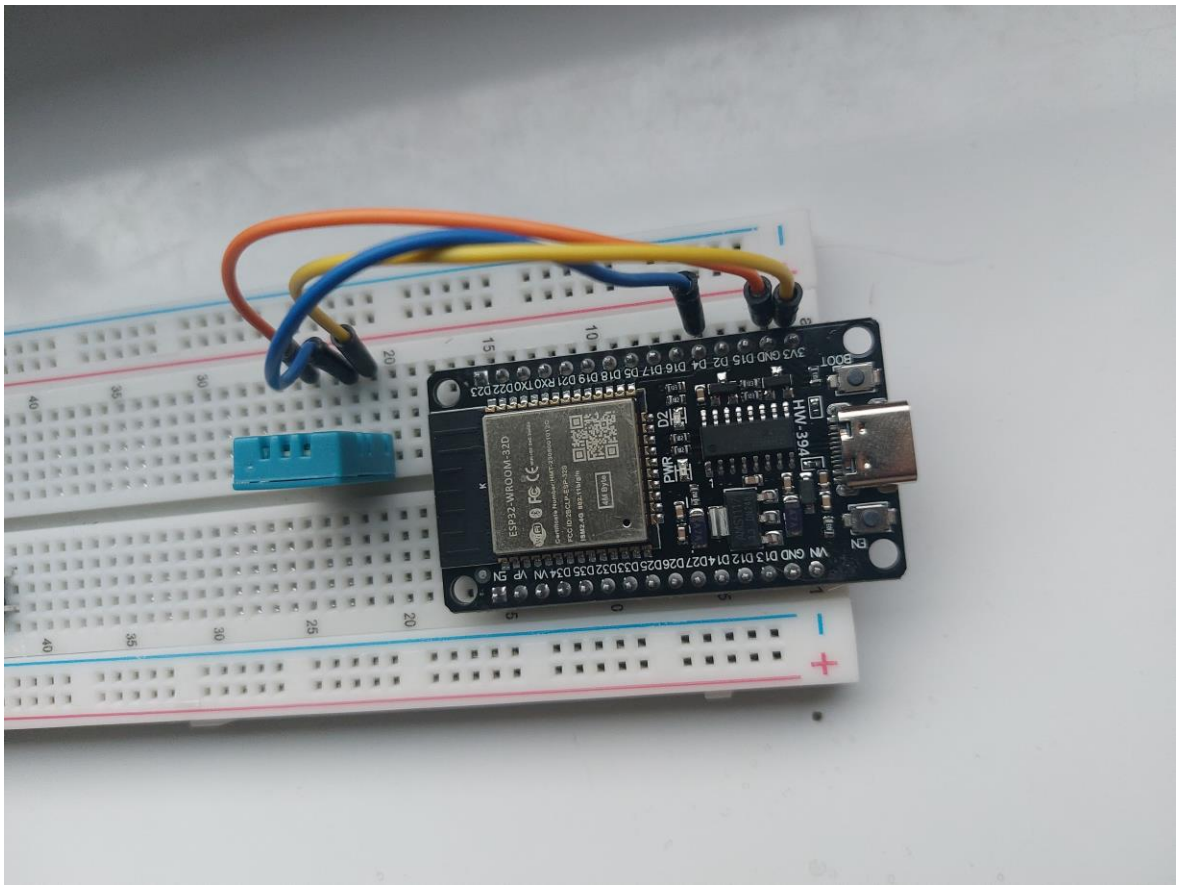

Fig. 11. Breadboard of the third passive device

To ensure data collection functionality, InfluxDB, a popular free cloud data processing and storage solution, was used. InfluxDB plays a key role in the project as a central repository for temporal data from temperature and humidity sensors. InfluxDB enables the storage, processing, and analysis of time-stamped data, allowing for precise and efficient tracking of changes over time. Thanks to integration with the ESP32 microcontroller, data from the DHT11 sensor is sent to InfluxDB, where it is collected in a defined bucket. This solution enables subsequent visualization of metrics in tools such as Grafana, which is important for environmental monitoring and real-time decision-making, such as sending an INVITE request to a SIP server when humidity exceeds a specified threshold. InfluxDB allows the project to easily scale and analyze historical data, making it more flexible and versatile.

Data Storage

InfluxDB is a modern time-series database designed for storing, processing, and analyzing time-stamped data. It is particularly useful for applications such as infrastructure monitoring and IoT. It is optimized for time-sensitive operations, such as aggregation, interpolation, and filtering. InfluxDB supports fast data insertion, flexible querying using the Flux language, and integration with popular visualization tools such as Grafana. InfluxDB also offers a simple interface for displaying the collected results, as shown in Fig. 12.

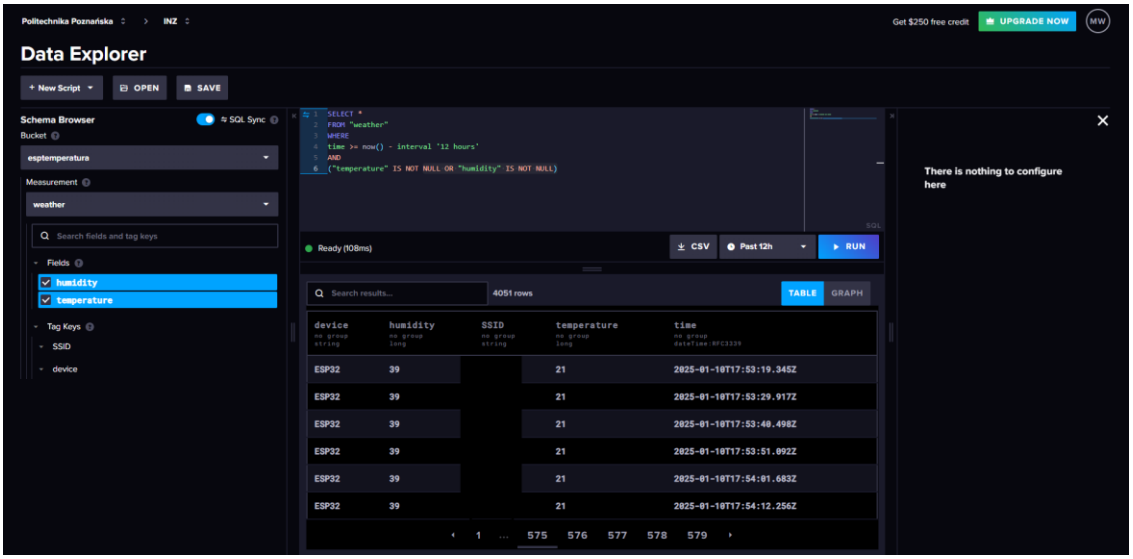

Fig. 12. Interface of API InfluxDB

To successfully begin working with InfluxDB databases, it was necessary to initially register a personal account and select the appropriate cloud server. For this project, the cloud provided by Amazon Web Services was chosen because of the servers' location in Europe.

Data Presentation

For convenient data presentation, the Grafana solution was chosen. Installed alongside the Asterisk SIP protocol server software, the program acts as an intermediary between the data collected in the InfluxDB cloud and the user, who expects a clear and simple data presentation in the form of charts or indicators (see Fig. 13).

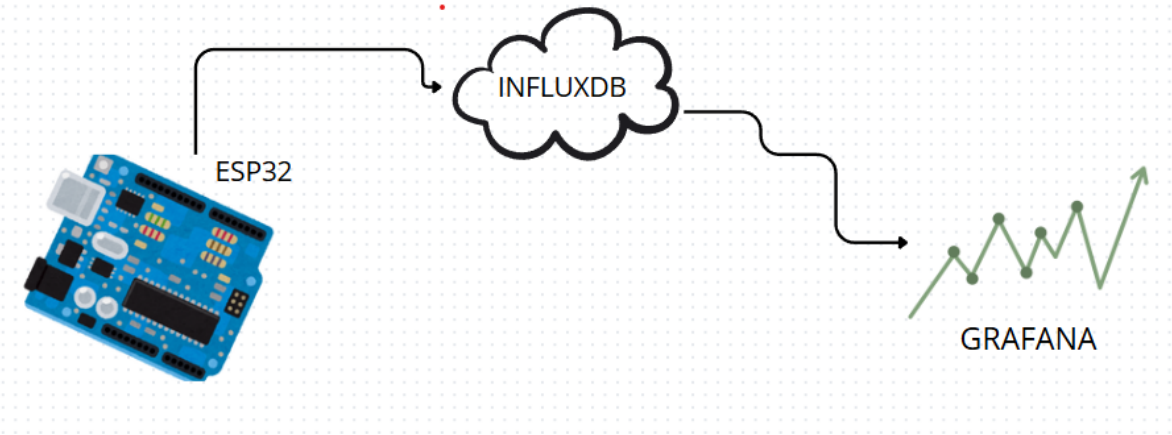

Fig. 13. Integration of InfluxDB and Grafana

For Grafana applications to function correctly, properly defining the data source is crucial. GrafanaLabs, a company developing open source data presentation software, simplified integration with various databases by creating numerous guides (see Grafana (2025) for details), which were used in our project. In short, to begin importing data into the presentation system, we must enter a unique user URL and provide our identity credentials. An example system presenting data graphically is shown in Fig. 14. In the top chart, the current humidity and the recorded humidity history are shown. In turn, the bottom part of the figure shows the current temperature and the history of the recorded temperature.

Fig. 14. The Grafana dashboard

The web application hosting program runs in parallel with the VoIP server adding additional load on the selected Raspberry Pi microcomputer. However, after performance testing, the programs do not interfere with each other's operation, and it is still possible to use both systems simultaneously.

Program Structure

The program loaded into the MCU is based on the program of the device that supports the door lock monitoring function. This software has been radically changed in some aspects due to the need to support a cloud solution based on time-distributed data series. It was necessary to include the appropriate libraries provided by InfluxData and declare the global constants necessary when creating the data storage container. Using the classes provided by the libraries previously included, the InfluxDB client should also be initialized. It is also necessary to initialize the digital temperature and humidity sensor using the DHT class. In the setup section of the program, in addition to initializing the execution of external library methods, the programmer is signaled by sending appropriate information about the microcontroller's workflow via serial communication. The loop section of the program ensures the correct reading of the data provided by the DHT11 sensor and its sending to the database. In addition, messages were added to the serial port to monitor the accuracy of the readings. For device performance testing, a condition was set to initiate a connection at a humidity reading higher than 50%, due to easier manipulation of humidity (due to the influence of wet human breath).

Security Aspects

A client sends a REGISTER request without credentials, receives a 401 response with a nonce and realm from the server, then computes HA1 as MD5 (username:realm:password), HA2 as MD5 (method:uri), and the final response as MD5 (HA1:nonce:nc:cnonce:qop:HA2). The client includes these in the Authorization header (with username, realm, nonce, uri, response, cnonce, nc, and qop) and resends the request. The server recalculates the response using stored credentials and grants 200 OK if the hashes match. This challenge-response flow prevents plaintext password transmission and resists replay attacks via nonces and counters. All of the above was implemented on microcontrollers, and there is no way that PBX could recognize that it is not a ‘real’ designated VoIP device, such as an IP Phone or a Personal Computer with VoIP handling software.

SIP Digest Authentication prevents several critical threats in VoIP systems. It stops password eavesdropping by transmitting only hashed credentials, not plaintext, making intercepted traffic useless to attackers. Replay attacks are blocked by using a server-generated nonce, a client nonce (cnonce), and a nonce counter (nc), ensuring that captured authentication responses cannot be reused. Unauthorized registration or call initiation is prevented, as only users with valid credentials can successfully authenticate, reducing risks like impersonation and toll fraud. While not a complete defense against man-in-the-middle attacks, the challenge-response mechanism prevents authentication without the correct password, even if messages are intercepted.

Conclusions

The purpose of this work was to investigate the feasibility of using SIP-based VoIP internet telephony in the Internet of Things environment and to implement a practical project using ESP32 microcontrollers as clients. An implementation analysis was conducted, which resulted in programming and connecting devices acting as sensors and an actuator controlling the gate opening mechanism after attempting to connect to the MCU.

The research and system testing demonstrated that VoIP technology can be effectively integrated with IoT devices, offering flexible and modern communication solutions. The use of the ESP32 microcontroller allowed the creation of an effective and cost-effective prototype characterized by high functionality and the possibility of further expansion.

The results of the work confirm that the use of VoIP in the IoT can significantly expand the functionality of networked devices, especially in the context of home and industrial automation. At the same time, the project revealed certain limitations, such as the need for a stable connection to a WLAN network and the requirement for appropriate communication security to protect against unauthorized access.

The project could serve as a starting point for further research and implementations in the integration of VoIP with IoT devices. Further steps could include implementing advanced security mechanisms, expanding the system to include integration with mobile applications, and supporting multi-node systems such as multibranch corporations or even VoLTE-based systems.

Acknowledgment

This paper was supported from the funds of the Ministry of Science and Higher Education, Poland for the year 2025 under Grant 0313/SBAD/1315.

References

- The 3CX website, Small Business Phone System – On-Premise or Cloud, [Accessed 29-01-2025], https://www.3cx.com/smb/.
- The Asterisk official website, Community Asterisk, [Accessed 29-01-2025], https://www.asterisk.org/community/.
- The official BananaPi website, Banana Pi IoT-Banana Pi open source hardware community, Single board computer, Router, IoT, STEM education, [Accessed 29-01-2025], https://www.banana-pi.org/en/banana-pi-ai-iot/.
- The ESP32 microcontroller documentation website, [Accessed 20-01-2025], https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf.
- FORBOT Damian Szymański, ESP32 co warto wiedzieć?, [Accessed 20-01-2025], https://forbot.pl/blog/leksykon/esp32.
- The FreeSwitch website, GitHub – signalwire/freeswitch: FreeSWITCH, [Accessed 29-01-2025], https://github.com/signalwire/freeswitch.
- The Grafana documentation website, Get started with Grafana and InfluxDB, [Accessed 20-01-2025], https://grafana.com/docs/grafana/latest/getting-started/get-started-grafana-influxdb/.
- Mark Handley, Van Jacobson, Colin Perkins, SDP: session description protocol, *Internet Engineering Task Force (IETF)*, 2006.
- Alan B. Johnston, SIP: understanding the session initiation protocol, *Artech House*, 2015.
- The Kamilio official website, The Kamailio SIP Server Project 2013; The Open Source SIP Server, [Accessed 29-01-2025], https://www.kamailio.org/w/.
- The Linphone official website, Getting Started – Linphone, [Accessed 29-01-2025], https://www.linphone.org/en/getting-started/.
- The Odroid official website, odroid-c4:odroid-c4 [ODROID Wiki], [Accessed 29-01-2025], https://wiki.odroid.com/odroid-c4/odroid-c4.
- The OrangePi official website, OrangePi 4A, [Accessed 29-01-2025], http://www.orangepi.org/html/hardWare/computerAndMicrocontrollers/details/Orange-Pi-4A.html.
- Ronald Rivest, The MD5 message-digest algorithm, *Internet Engineering Task Force (IETF)*, 1992.

- Henning Schulzrinne, Stephen Casner, Ron Frederick, Van Jacobson, RTP: A transport protocol for real-time applications, *Internet Engineering Task Force (IETF)*, 2003.
- The Tiptel phone official website, [Accessed 20-01-2025], https://www.tiptel.de/fileadmin/user_upload/tiptel_yealink/Dateien/Manuals/tiptel-IP-280_Yealink-SIP-T20P_Manual_V50-1_EN.pdf.