electronics

MDPI

Article

Edge Container for Speech Recognition

Lukáš Beňo *, Rudolf Pribiš and Peter Drahoš *

Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, 812 19 Bratislava, Slovakia; rudolf.pribis@stuba.sk

* Correspondence: lukas.beno@stuba.sk (L.B.); peter.drahos@stuba.sk (P.D.)

Abstract: Containerization has been mainly used in pure software solutions, but it is gradually finding its way into the industrial systems. This paper introduces the edge container with artificial intelligence for speech recognition, which performs the voice control function of the actuator as a part of the Human Machine Interface (HMI). This work proposes a procedure for creating voice-controlled applications with modern hardware and software resources. The created architecture integrates well-known digital technologies such as containerization, cloud, edge computing and a commercial voice processing tool. This methodology and architecture enable the actual speech recognition and the voice control on the edge device in the local network, rather than in the cloud, like the majority of recent solutions. The Linux containers are designed to run without any additional configuration and setup by the end user. A simple adaptation of voice commands via configuration file may be considered as an additional contribution of the work. The architecture was verified by experiments with running containers on different devices, such as PC, Tinker Board 2, Raspberry Pi 3 and 4. The proposed solution and the practical experiment show how a voice-controlled system can be created, easily managed and distributed to many devices around the world in a few seconds. All this can be achieved by simple downloading and running two types of ready-made containers without any complex installations. The result of this work is a proven stable (network-independent) solution with data protection and low latency.

Keywords: edge computing; containerization; speech recognition; Azure cloud; ARM64; Docker; data privacy

Citation: Beňo, L.; Pribiš, R.; Drahoš, P. Edge Container for Speech Recognition. *Electronics* **2021**, *10*, 2420. https://doi.org/10.3390/electronics10192420

Academic Editor: Prasan Kumar Sahoo

Received: 30 August 2021
Accepted: 28 September 2021
Published: 4 October 2021

1. Introduction

Alexa [1], Siri [2], Cortana [3] and Google Assistant [4] have shown the human–machine interface of the future. Voice-activated queries have become more widespread across smart devices, and they have made lives easier, as they are offering convenience and simplicity. Along with industrial applications and smart homes, voice assistants are rising in vehicle driver assistance systems (According to Voicebot.ai, 73% of drivers will use an in-car voice assistant by 2022 [5]). Furthermore, voice assistants have also found their usage in healthcare and scientific laboratories, and they could also help seniors in their daily lives at home. The market for voice assistants is growing constantly, and according to a research report by Market Research Future (MRFR), "Voice Assistant Market–Information by Technology, Hardware and Application—Forecast till 2025" the market valuation stood at USD 1.68 billion in 2019 and is projected to reach USD 7.30 billion by 2025 [6].

Most of the solutions transform the human voice to specific commands in the cloud data centers. Therefore, every time there is a request for voice control, information must travel through a network to a remote place. Cloud computing can be described as a client-server architecture. Edge computing can be considered as an "extension" to cloud computing, as it brings the "computation" closer to the source of data generation, at the edge. [7]. Phones, cars, factories, smart devices and cities generate a big volume of data that consume the network, and in the near future, this load on the network could cause problems. Big latencies cost Amazon, Google and their customers millions. Amazon found

that every 100 ms of latency cost them 1% in sales [8]. Statista estimates that over 75 billion Internet of Things (IoT) devices will be connected to the network by 2025 [9].

An important aspect of every solution is also its delivery to the end user. As there are many devices with different kinds of operating systems, hardware and accessibility, it is necessary to provide a reliable delivery of the solution, despite the mentioned challenges. One of the possible options is the usage of Docker and containerization technology, which deals with these problems. Docker and containerization is probably the most talked-about infrastructure technology of the past few years, as it is heavily used by most of the big tech companies such as Amazon, Google, Red Hat, Microsoft and IBM, meaning it is finding its place in the industrial systems too. Docker and containerization were mainly used in the cloud infrastructures but now it is trying to build its position in the IoT edge devices. For this reason, it is essential to show how containerization can work in the local network on the Edge devices, with a focus on easy usage of containers. As containers can be set up in a way that an end user does not have to undertake any additional configuration and installation of dependencies on their own, everything is handled by a specific container.

Data privacy is an aspect that must be considered too, because in the wrong hands, personal information can be wielded as a powerful tool. The "Cambridge Analytica scandal" is a perfect example of that kind of misuse [10].

For these reasons, the work proposes a system that combines the advantages of voice control as a user interface and edge computing focused on data privacy, low latency, easy management and deployment by using containerization. Containers for speech recognition are not running in the cloud as usual but directly in the local network of the user on the edge devices. Containers are designed in a way that a user can download and run them with minimal effort spent on configuration, installation of dependencies and containers themselves. Containers can recognize each other in the local network, communicate and use a connected actuator and microphone, without any interaction from an end user. The work also brings a methodology for the creation of a speech recognition application and experimental verification of the proposed architecture.

The main goals and innovations can be formulated to the following points:

- Architecture of speech-controlled solution using edge computing and containerization;
- Solution running in Linux containers with a focus on automatic setup of all required dependencies and libraries;
- Simple adaptation of voice commands using a configuration file;
- Easy deployment and management of the solution;
- Portability, low latency and data privacy.

The article is organized as follows: Section 2 describes the current state-of-the-art regarding speech recognition, edge computing vs. cloud computing and a summary of the containerization concept. Section 3 presents the proposed architecture and designed methodology of the voice-controlled actuator. Section 4 presents the experimental verification of the proposed architecture. In Section 5, the results of the system implemented according to the proposed methodology and the expected benefits of the proposed architecture are described. Finally, Section 6 consists of the conclusions of this paper.

2. Background of Speech Recognition, Edge Computing and Containerization

Many recent implementations of voice control are focused on functionality, which can be offered by the proposed architectures, but they are not focused on specific topics, such as the processing of data near its source, data privacy, easy deployment and management of the solution. In a scientific article [11], authors use a conversation agent in the cloud, so every voice record must be sent remotely to a third-party provider. Furthermore, they are not dealing with the deployment question. It can be assumed that deployment and management of the edge devices are undertaken in the old fashion way—manually. Similarly, in a scientific article [12], it cannot be seen that the work is dealing with the question of deployment and the processing of data in a local network. Therefore, the study deals with topics such as speech-to-text libraries, edge computing, containerization, Docker

and necessity of data privacy. Even while using Alexa from Amazon or Siri from Apple on an iPhone, an internet connection is still needed, otherwise they cannot be used.

2.1. Speech-to-Text Libraries

Speech-to-text libraries create the core of the voice controlling, as they are needed for a transformation of human speech to text. For these libraries, it is important to have the capability to support more languages other than just English. They should also have a certification for data privacies or an option to run them as the container to achieve quick and easy deployment; in the best case, these containers could run locally on the network.

In Tables 1 and 2, the comparison of well-known speech-to-text libraries is shown, as they can be used for the transformation of the human voice to plain text, which can be later used for controlling of mechatronic devices or communication with them. Based on the comparison, Microsoft Cognitive Speech Service [13] was chosen, as it supports many programming languages in general. Microsoft Cognitive Speech Service was also chosen for the reason that it can run locally without a special setup and the container is officially released by Microsoft.

Table 1. In the following table, the capabilities of well-known speech-text-libraries are described with the focus on the certifications and language.

Name	Multilanguage Support	Certifications	Own Dictionary-Slang	Removal of Background Noises
Google Cloud Speech	Yes	Yes *	Yes	Yes
Microsoft Cognitive Services Speech	Yes	Yes *	Yes	Yes
Amazon Transcribe	Yes	Yes *	Yes	Yes
System Speech	Yes	No	Yes	No
DeepSpeech	No	No	Yes	No
Kaldi	No	No	Yes	No
Annyang	No	No	Yes	No
Voice-commands.js	No	No	Yes	No

* Types of certifications: System and organization controls (SOC), The Federal Risk and Authorization Management Program (FedRAMP), The Payment Card Industry Data Security Standard (PCI DSS), Health Insurance Portability and Accountability Act (HIPAA), Health Information Technology for Economic and Clinical Health Act (HITECH) and International Organization for Standardization (ISO).

Table 2. In the table, capabilities of well-known speech-to-text libraries are described with the focus on the programming languages and containerization.

Name	Owner	Programming Languages	Running in the Cloud	Docker Image-Container *
Google Cloud Speech	Google	C# (Core), Go, Node.js, PHP, Ruby **	Yes	No (Necessity to run the Container in K8s infrastructure)
Microsoft Cognitive Speech Service	Microsoft	C# (Core), C++, JavaScript, Objective-C/Swift **	Yes	Yes
Amazon Transcribe	Amazon	Java SDK, Ruby SDK, and C++ SDK	Yes	No
System Speech	Microsoft	C# (.NET Framework)	No	No
DeepSpeech	Mozilla	C, JavaScript and **	No	No
Kaldi	Kaldi	C++	No	No
Annyang	Annyang	JavaScript	No	No
Voice-commands.js	Jimmy Byrum	JavaScript	No	No

* Docker Image-Container is officially provided by the owner of the library, and it has the possibility to run on the edge device with the basic installation of a Docker engine. Notice ** Supported programming languages: Java, Python, C# (.NET Framework).

Testing was performed with Microsoft Cognitive Speech Service in addition to System Speech, DeepSpeech and Voice commands.js. Compared to Microsoft Cognitive Speech Service, the mentioned libraries were difficult to implement, and it was necessary to use

their specific programing language. Furthermore, they do not support the phrase trigger for an activation of the voice control.

2.2. Edge Computing vs. Cloud Computing

Instead of sending data back and forth, edge computing processes data on the edge of the networks (Figure 1), which allows faster response times and better connectivity. It eliminates the need to send data remotely to the cloud and reduces the volumes of data that must be transferred through the network [14]. Furthermore, edge devices are more secure than traditional cloud computing, as the processing is performed locally in the private network [15]. Thanks to the processing of data on the device of their source, data privacy can be more easily implemented and ensured, as there is no need to send data remotely to a third-party storage or solution. Even though big tech companies claim that they store data safely, there is still a possibility that user data can be stolen and misused [10]. An important aspect of edge computing is also Quality of Service (QoS) in Wireless Local Area Network (WLAN), why and how QoS can be achieved in WLAN is described in the scientific article "A survey on 802.11 MAC industrial standards, architecture, security and supporting emergency traffic: Future directions" [16].

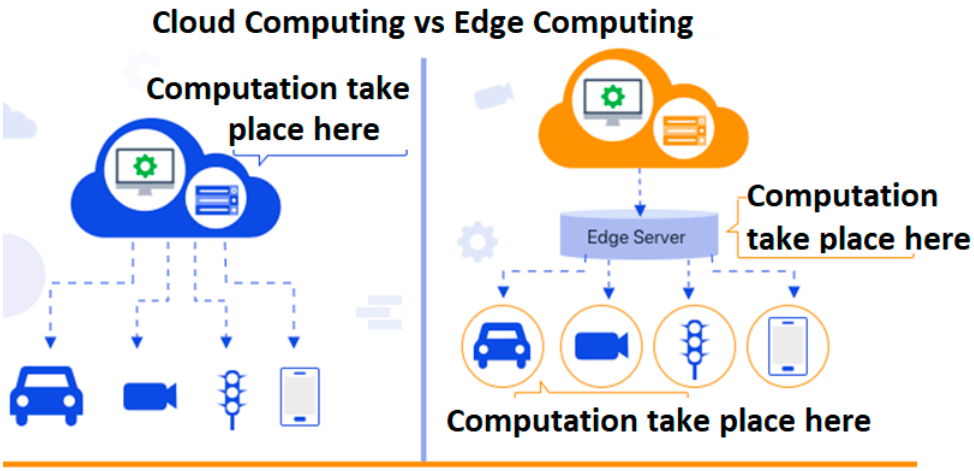

Figure 1. Difference between the cloud and edge computing [17].

2.3. Containerization and Docker

In containerization, the software is packaged together with all its necessary components such as libraries, frameworks, configuration and other dependencies so that they are isolated in their own "*container*" (Figure 2). Thanks to containerization, the applications have the ability to be run and moved in any environment and on any infrastructure. Dependency on an underlying operation system and infrastructure is eliminated. This makes the container an ideal tool for transporting applications to various platforms and infrastructures. The container is a type of bubble that acts as "a secure environment", which keeps the application independent [18].

The difference between a virtual machine (VM) and the container is that the VM is a virtual computer with its own central processing unit (CPU), memory, network interface, storage and created physical hardware. Both virtual machines and containers allow the isolation of applications, allowing them to run in various environments. The main differences are in size, portability, deployment and scalability [19,20]. The containers are so much smaller, typically megabytes and they do not require to pack anything other than the app itself and its running environment. Containers are compatible with various newer technologies such as Continues Integration/Continues Delivery (CI/CD) and DevOps, which help in reducing complexity and increasing distribution and maintenance efficiency [19]. VMs are intended for traditional and monolithic IT architectures.

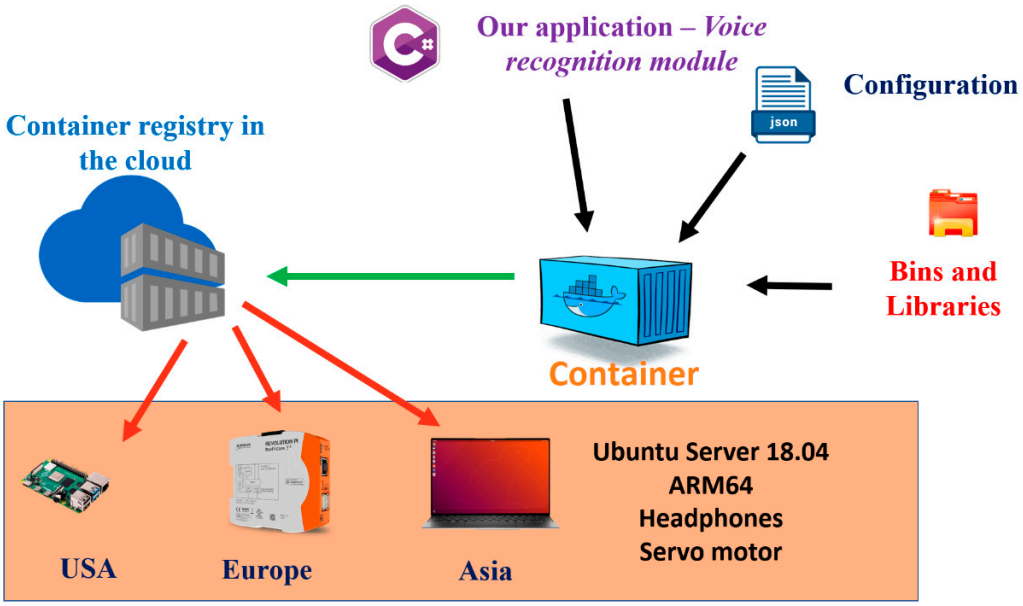

Figure 2. Packing of all dependencies to the container and the following distribution of the container to different machines around the world.

Docker is the most known open-source containerization platform that enables the packaging of applications into containers in a standardized way [21]. There also exist different providers for containerization, such as Kubernates or Amazon Elastic Container Service. These solutions provide more functionality, which is not always necessary, and their setup is more complex and resource-consuming. Docker is used in the proposed architecture, as it is heavily used by the developers. It has a straightforward setup, and it does not require a lot of resources to run.

3. Designed Architecture and Methodology of the Voice-Controlled Actuator

The following designed architecture describes the controlling of the actuator by using voice commands and the edge device (Figure 3). The transformation of the human voice into controlling commands for the actuator is performed directly on the IoT edge device or remotely in the cloud Azure. The IoT edge device is connected and managed via the cloud Azure. The concept in Figure 3 consists of the following main parts:

1. IoT Hub and the cloud Azure—contains information about used services from the cloud Azure and the IoT Hub running in the cloud Azure. These are used for management and deployment.
2. IoT edge devices using the runtime Azure IoT edge—describes the two categories of edge devices and the runtime Azure IoT edge used in the proposed architecture.
3. Containers for the speech control of the actuator running on the IoT edge devices—describes the functionality of the developed and used Linux containers for the control of the actuator by speech generated commands and an integrated development environment, Visual Studio Code. This architecture was created by two types of Linux containers:
    A. Configured container for Azure Speech Recognition service—transformation of voice to text.
    B. Created container for Voice recognition module—controlling of mechatronic device and processing of voice.

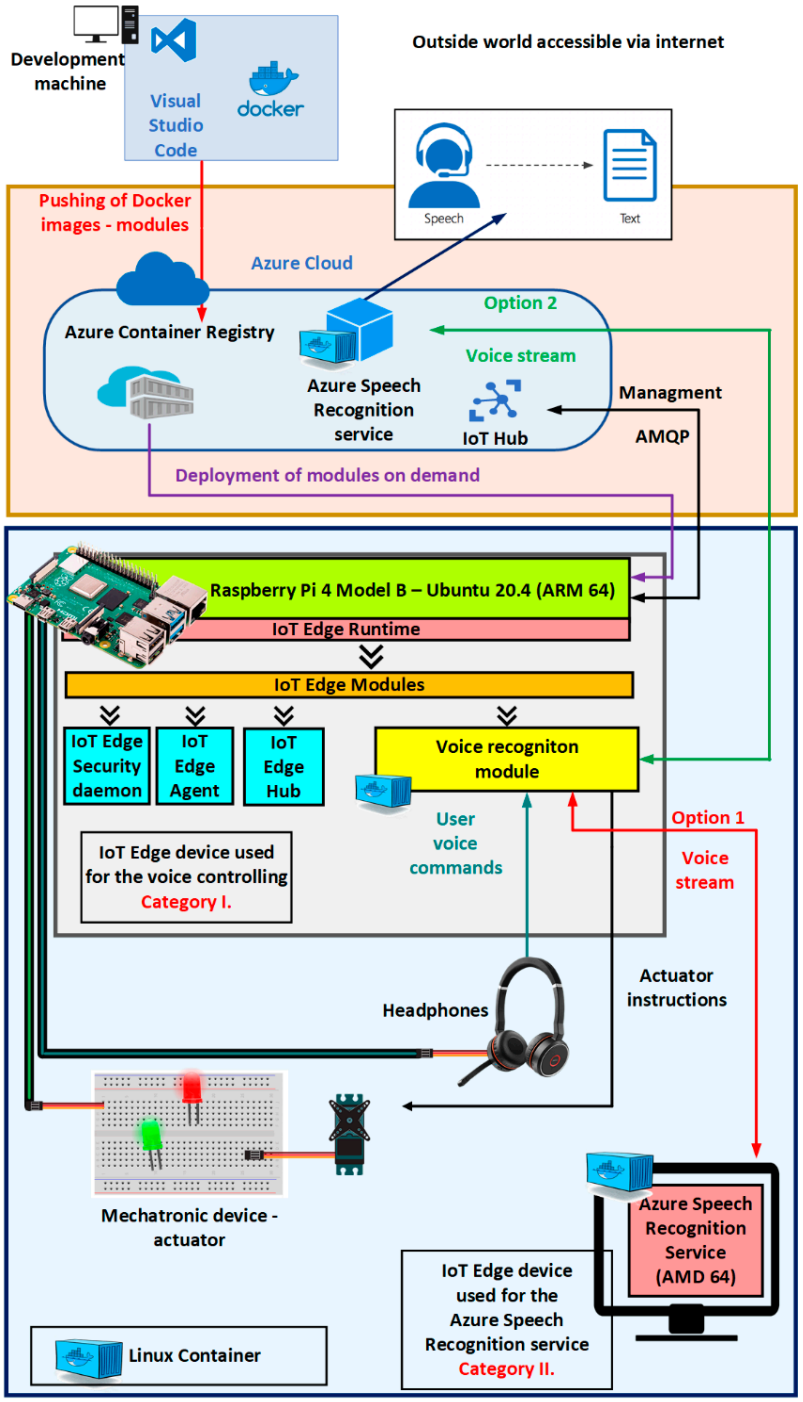

Figure 3. The entire architecture of the controlling of the actuator using voice commands.

In Figure 4, it is possible to see the sequence diagram of the initialization of the speech-controlled solution. The initialization of the voice-controlled actuator and the microphone for speech input is completed, the settings are loaded and the connection to the local edge or cloud service for the speech recognition (Azure Speech Recognition service) is established.

3.1. IoT Edge Devices Using the Runtime Azure IoT Edge

This chapter deals with the runtime Azure IoT edge [22] and the categorization of IoT edge devices used in the proposed architecture.

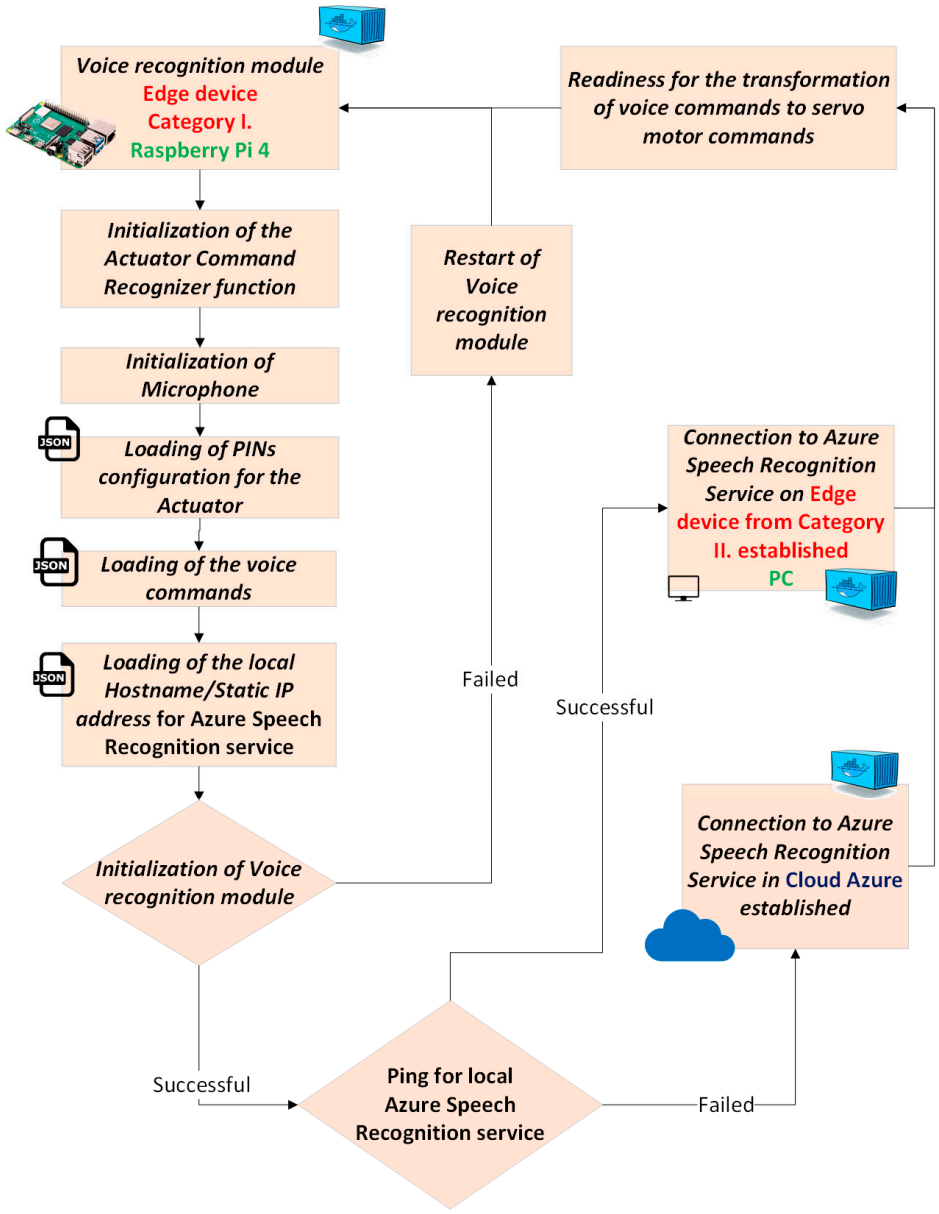

Figure 4. Sequential steps in the proposed architecture of voice control.

3.1.1. The Runtime Azure IoT Edge

To be able to control an actuator by using voice commands, it is necessary to have a device with specialized software that can transform the human voice into text and then perform subsequent analysis to create a meaningful command for the actuator. The specialized software Azure IoT edge runtime can allow the edge device to perform the mentioned transformation directly, and it was chosen because of the usage of Microsoft Cognitive Speech Service and it fulfills all requirements for edge computing.

The Azure IoT edge runtime is a set of programs that must be installed on the device to be considered an Azure IoT edge device. The runtime can run on small devices such as Raspberry Pi 3 [23] and 4 [24], Tinker Board 2 [25], a laptop, or even something bigger, such as an industrial server. The Azure IoT edge runtime provides [26]:

- Installation, updating of modules and securing that edge modules are always running, and the latest security updates;
- Communication between the edge modules—acting as a local message broker;
- Reporting the status of edge modules to the cloud for remote monitoring.

Azure IoT edge runtime has three main tasks, and they are handled by three components—modules running on the IoT edge device [26]:

1. IoT edge agent is one of the modules that creates the Azure IoT edge runtime. The agent is responsible for instantiating, starting, and running the modules. In the same way, it is responsible for reporting the status of individual modules to the IoT Hub. The Azure IoT edge module (in further text abbreviated as *module*) is the smallest computing unit deployed and controlled by the IoT edge. The module can contain Azure services (such as Azure Cognitive Speech Services) or its own project-specific code. For modules, it is necessary to mention two points [27]:
   - Module image—A package of the software that defines a module. Module images exist as container images and they are stored in a container repository in the cloud, and module instances are containers on devices (Figure 5).
   - Module instance—Specific running module image on the IoT edge device (Figure 5). Module instances are independent. Both modules have their own identity in the cloud Azure.

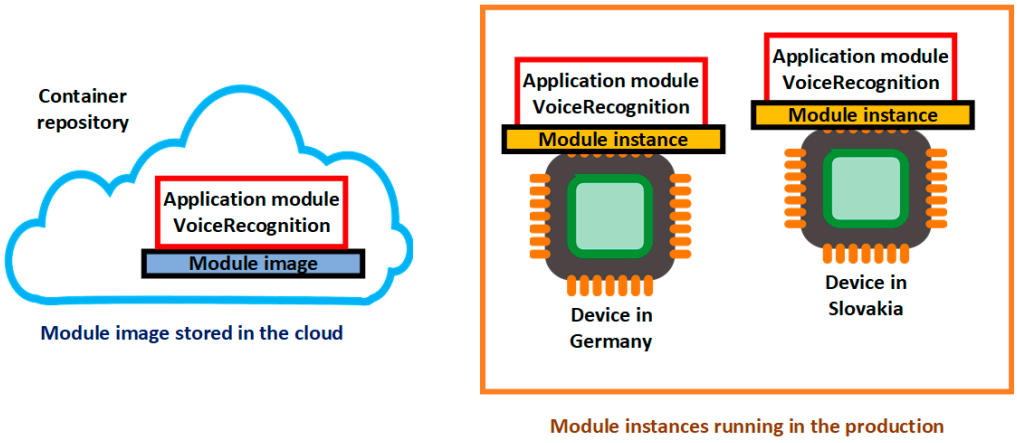

Figure 5. Module's image and instances.

2. *IoT Edge Hub* acts as the local message broker, so it keeps the modules independent. Communication between IoT Edge Hub and IoT Hub in the cloud Azure is performed via protocols Message Queuing Telemetry Transport (MQTT) and Advanced Message Queuing Protocol (AMQP). Modules need to specify the inputs where they listen for the messages and the outputs where they write the messages. Inputs and outputs are defined as "routes":
   - Connection between the VoiceRecognition and Insight module (Figure 6) can be defined as:

   From/modules/VoiceRecognition/outputs/VoiceRecognitionOutput
   To BrokenEndpoint(\"/modules/insight/insightInput")
   - Connection between the VoiceRecognition module and IoT Hub in the cloud Azure (Figure 6) can be defined as:

   From/modules/VoiceRecognition/outputs/VoiceRecognitionOutput TO $upstream
3. IoT edge security manager (Figure 7) is the security core that protects the IoT edge device, and it is responsible for performing the logical operations such as encryption, decryption, hashing, generation of digital signatures and signature verification. The IoT edge security daemon starts when the IoT edge device is turned on, and its responsibility is the initialization of the IoT edge agent. The edge IoT security daemon provides access to application programming interfaces (APIs) such as [28] (Figure 7):
   - Container API—offers an interface for interacting with container systems, such as Docker and Moby.

- Management API—only the IoT edge agent can access it. It is used to create, start, stop and remove the modules.

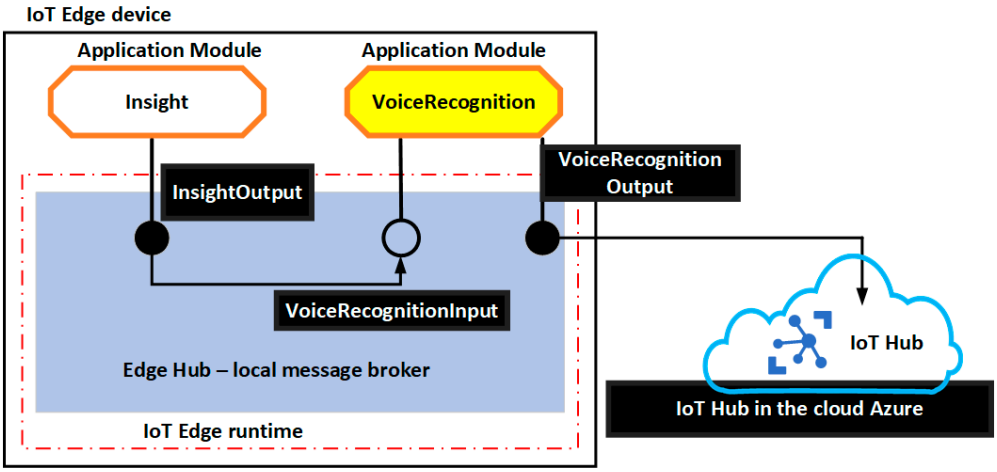

Figure 6. Routes in IoT Edge Hub deployed on the device.

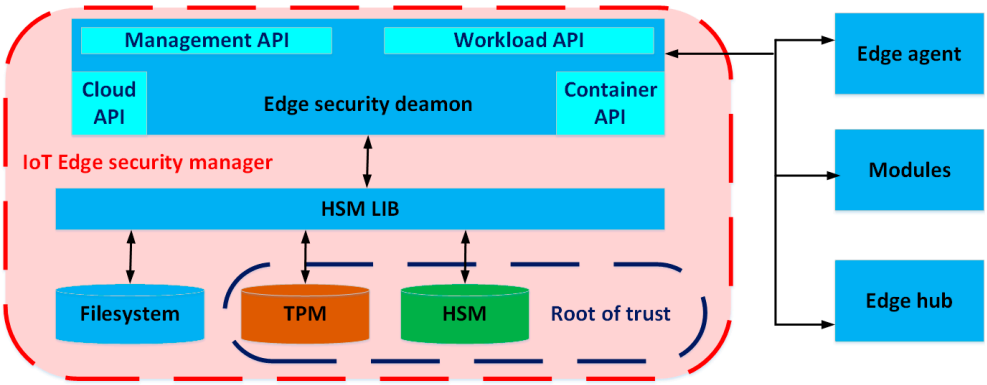

Figure 7. IoT edge security daemon structure and communication.

3.1.2. Supported Systems for IoT Edge Runtime

The IoT edge runtime can run on the following systems [29]:

- Tier 1 systems—Operation systems officially supported by Microsoft. Microsoft has automated tests and provides installation packages for them. The latest version of Azure IoT edge supports only the Linux containers. These operating systems include Raspberry Pi OS Stretch (ARM32v7) and Ubuntu Server 18.04 (AMD64 and ARM64 in the Preview version).
- Tier 2 systems—can be considered compatible with Azure IoT edge. Testing on these platforms has been performed at least once but continuous testing is not performed. Tier 2 systems are CentOS 7.5, Debian 8, 9 and 10, RHEL 7.5, Ubuntu 16.04 and 18.04, Wind River 8, Yocto and Raspbian Buster 1.

In the implementation of the created architecture, Ubuntu Server 18.04 for ARM64 (In the preview version) architecture from system Tier 1 is used.

3.1.3. IoT Edge Devices

IoT edge devices in the created architecture were divided into two categories and can be classified as (Figure 8):

- Category I—Edge device where the actuator and the Bluetooth headphones are connected and it is used for running the Voice recognition module—it can be a device such as Tinker Board 2, PC, Raspberry Pi 3 or 4.

- Category II—Edge device where Azure Speech Recognition service is running—this device must be more powerful, as it is used for the transformation of voice to text, and many devices from Category I can be connected to this device.

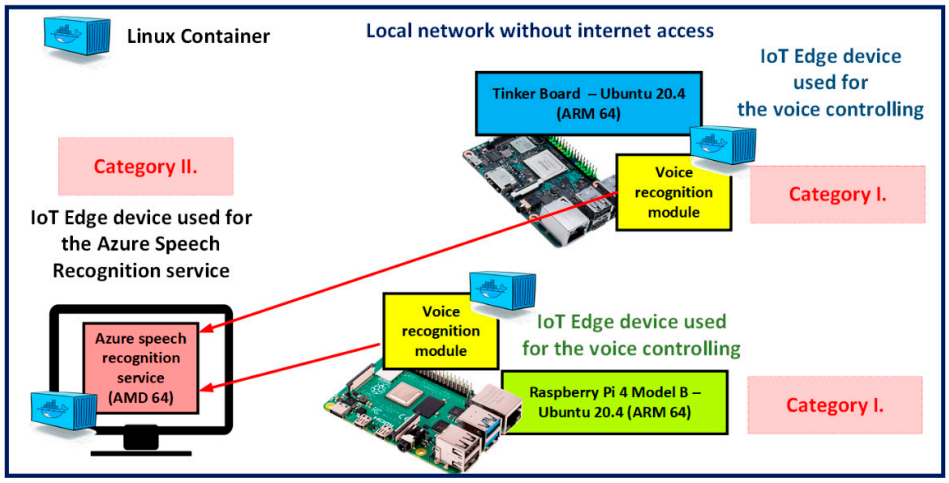

Figure 8. Categories for edge devices.

3.2. IoT Hub and the Cloud Azure

The cloud Azure is used for the deployment and the configuration of the IoT edge device. In the cloud Azure, the IoT Hub (Figure 9) is created, which provides a cloud-hosted backend solution for the connection of IoT edge devices. In addition, the cloud Azure is used for storing Linux container images in Azure Container Registry.

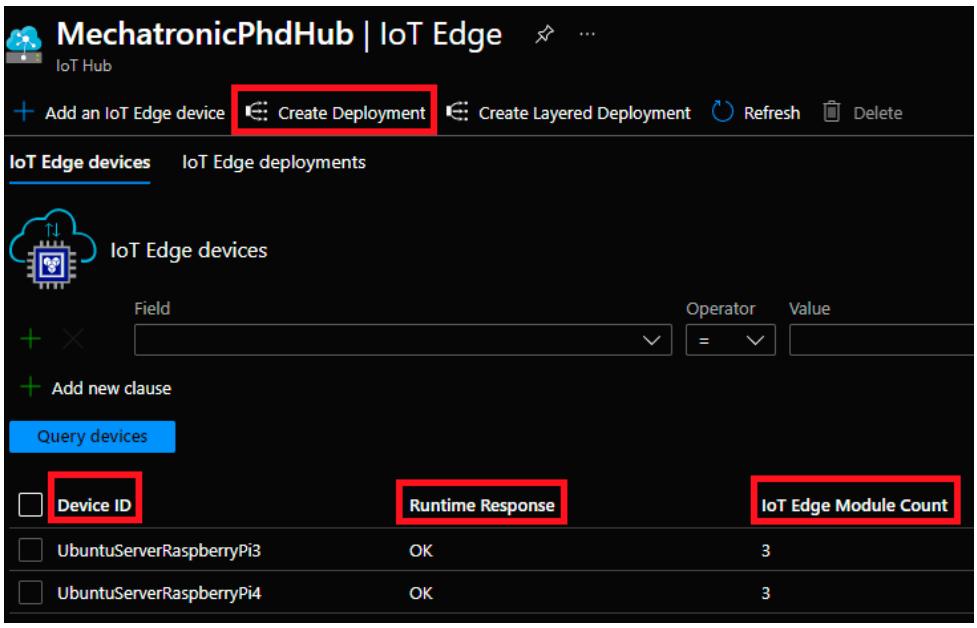

Figure 9. IoT Hub in the cloud Azure for management and deployment of IoT edge devices.

Thanks to the IoT Hub in the cloud Azure, IoT edge devices can be easily managed, and the following operations can be performed:

- Monitoring whether the IoT edge device is up and running;
- Deployment of modules;
- Checking the health of deployed modules;
- Launching an update of modules;
- Connection and authentication of the new devices with the same or modified configuration.

3.3. Containers for the Speech Control of the Actuator Running on the IoT Edge Devices

For the development of modules, a local development machine with Visual Studio Code for easy connection and management of the edge devices was used. The configuration of the IoT edge devices was achieved by using the management portal in the cloud Azure. As mentioned, the development machine handles the creation of modules and it was used for the development of the proposed Voice recognition module. Azure Speech Recognition service is provided officially by Microsoft. The developed Voice recognition module performs these tasks (see Figures 4 and 10):

1. Verification of the system requirements on the IoT edge device for the Azure Speech Recognition service. For the transformation of voice into text and better recognition, Azure Speech Recognition service with artificial intelligence was used. This service can run directly on the IoT edge device or remotely in the cloud Azure. If the Azure Speech Recognition service runs locally, it does not need a completely stable internet connection. Only every 15 min, the Azure Speech Recognition service must connect to the cloud Azure and provide information about the usage.

```
INITIALIZATION of the module
        OPEN connection to the IoT Edge runtime
        OPEN connection to the IoT Hub
        DO GPIO controller initialization
                LOAD PINs for the control of mechatronic device from the JSON file
                OPEN PINs for the control of mechatronic device
        LOAD commands for the control of mechatronic device from the JSON file
        LOAD trained phrase key word "Hey Bennie"
        DO system requirement check of memory and CPU
        LOAD endpoint for the Speech recognition service
        IF ping of a local Speech recognition service is successful
                OPEN connection to the Speech recognition service on the Edge device
ELSE
OPEN connection to the Speech recognition service in the cloud Azure
ELSE error occurs
PRINT error message and DO restart of the module
        LOAD audio config and try to access connected microphone
        IF loading of audio config is successful
                PRINT successful message
ELSE
PRINT error message
DO restart of the module
        CREATE instance of the Actuator Command Recognizer
OPERATION of the module
        USING the Actuator Command Recognizer in an infinite loop
        SEND voice command to the Speech recognition service for transformation to text
IF error occurs
PRINT error message and DO restart of the module
        ELSE
DO extraction of keywords from the recognized speech
DO matching of keywords to the command
        IF matching of the command is successful
                SEND signal to specific PINs based on the command
        ELSE
        PRINT message for repetition of the command
        IF any error occurs
        DO restart of the module
        ELSE
        CONTINUE with the speech control
```

Figure 10. Pseudocode of voice recognition module.

As the Azure Speech Recognition service requires significant computational power, it is necessary to have the IoT edge device in a local network that is capable of running it. That is the reason why two of the IoT edge devices are used in a local network. One IoT edge Device (Category II) is only for running the Azure Speech Recognition service as a Linux container and the other IoT edge device (Category I) is used for the voice control (Figure 8). Many devices (Category I) from a local network can be connected to the edge device (Category II) running Azure Speech Recognition service (Figure 11).

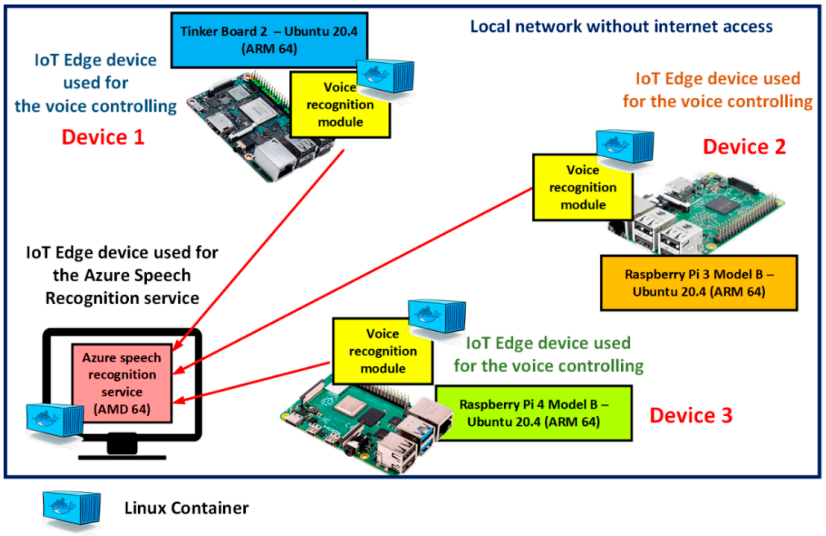

Figure 11. Three ways to connect many edge devices (Category I) on a local network to one single IoT edge device (Category II) running the Azure Speech Recognition service.

The developed Voice recognition module always checks if the endpoint for the Azure Speech Recognition service is available in a local network, and based on that, it will try to connect to the Azure Speech Recognition service running on the edge device in a local network (Option 1—Figure 12) or it will try to connect to the Azure Speech Recognition service, which runs remotely in the cloud Azure (Option 2—Figure 12). The endpoint for the Azure Speech Recognition service can be configurated in the JavaScript Object Notation (JSON) file through the hostname or the static local IP address.

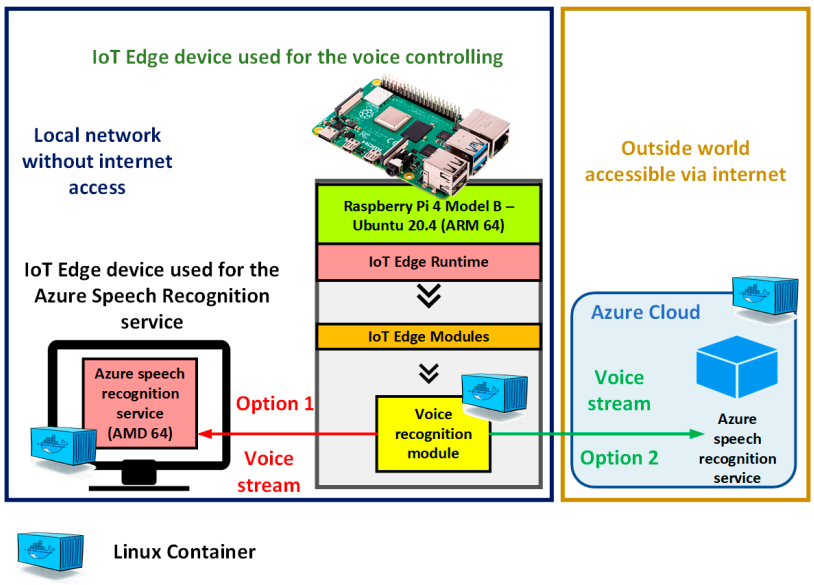

Figure 12. Possible location for running the Azure Speech Recognition service.

If the Azure Speech Recognition service runs as a Linux container on the IoT edge device, edge computing is performed. Edge computing leads to a better response time for the transformation of speech to text and no internet connection is required. The user is informed about every performed step through the console output from the Voice recognition module. In the console, information is displayed about checking the system requirements and which type of the Azure Speech Recognition service is used (cloud or edge option).

2. Configuration and setup of prerequisites for the Linux Voice recognition module—The created Linux container is prepared to automatically take the USB headset that is connected to the IoT edge device. It also contains all libraries and configuration, which are needed to access a microphone from the underlying system. No input is required from the end user to make the Voice recognition module functional.
3. Transformation of the human voice to actuator commands—Described in Figure 4, and it consists of three main components:
    I. Azure Speech Recognition service—responsible for the transformation of voice into text. The time for the speech recognition, which is performed remotely in the cloud Azure, is approximately about 5.9 ms when using an internet connection with the speed of 15 Mbps. When using the Azure Speech Recognition service on the edge device in a local network, the time of transformation of voice into text is approximately the same, as the cloud Azure option.
    II. Actuator Command Recognizer function—part of the Voice recognition module is responsible for the extraction of keywords from the transformed voice into text and the matching of keywords to specific actuator commands. To use voice commands, it is always necessary to trigger voice control by using the phrase "Hey Bennie"—similar to "Hey Alexa" for Amazon, "Hey Siri" for Apple or "Hey Mercedes" for Mercedes-Benz. The triggering phrase should always be unique in voice recognition systems. Actuator Command Recognizer function, meaning the proposed architecture currently supports four voice commands:
        - "Connect to the motor"—Used for connecting to the actuator, signalized by the Green Led turning on and the Red Led turning off.
        - "Disconnect from the motor"—Used for disconnecting from the actuator, signalized by the Red Led turning on and the Green Led turning off.
        - "Turn off everything"—Used to turn off the whole controlled system.
        - "Move the motor"—Actuator is going to be moved.

    If the command cannot be recognized, the end user is asked to repeat it.

Commands for the controlling of the actuator can be adjusted easily in the JSON file. There is no need to change them in the source code, and a new word can be easily added, deleted or modified (in case of an existing one). Voice recognition module is always trying to match the extracted text from the human voice to a specific word defined in the JSON commands configuration file. It must always match every word in the JSON file to a specific command. Imagine that someone would like to "connect to the motor". In this case, it is necessary to say something such as: "Hey Bennie, please connect me to the motor" and the connection to the motor/actuator is going to be established. If it is only "Hey Bennie, please connect me" the command will not be recognized, as the word "Motor" is missing in the command. "Hey Bennie" is always mandatory to trigger the communication with the proposed system. The structure of commands and matching words stored in the JSON file for the controlling of the actuator is described as:

- Command One
    - First matching word,
    - Second matching word,
    - ...
- Command Two

- First matching word,
- Second matching word,
- Third matching word,
- ...

III Controlling of the actuator connected to the IoT edge device—Voice recognition module is using predefined general-purpose input/output (GPIO) pins for the controlling of the actuator to make the solution more universal and flexible. If default PINs have already been taken, there is a possibility to overwrite the default values with their own specified values, as the configuration of the PINs is stored in the JSON file. Configuration in the JSON file can be easily changed by the end user. To apply the changes, it is enough to only modify the JSON file and restart the system. There is no need to create or rebuild the program inside the Voice recognition module.

Modules are developed by using the development framework for building cross-platform apps.NET Core. Modules developed using.NET Core can be deployed on Linux or Windows containers. The majority of the IoT edge devices use the operating system Linux.

Voice recognition module is a Linux container for the architecture ARM64, and the container is stored in the Azure Container Registry [30]. The development environment uses Visual Studio Code, which offers tools for module debugging on the IoT edge devices (Figure 13), as well as for pushing Docker images to the container registry in the cloud Azure. Visual Studio Code is also used for deploying and updating modules.

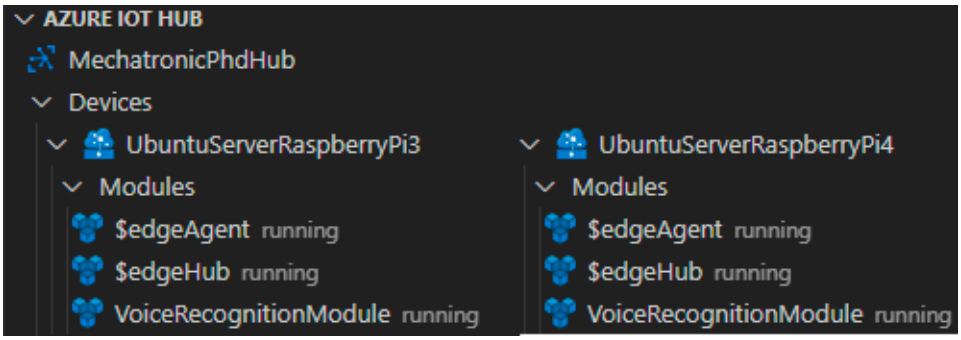

Figure 13. Visual Studio Code—debugging option. Possibility to see which modules are running on the specific devices. PC, Tinker Board 2, Raspberry Pi 3 and 4 are used in the experiment.

3.4. Methodology of the Implementation

The methodology for the setup of the proposed and implemented architecture can be formulated in the following steps:

1. Create Azure IoT Hub [31] in the cloud Azure for management and deployment of containers on the IoT edge devices.
2. Create the Azure Speech recognition service in the cloud Azure. The result of this action is the endpoint, which is later used for the cloud and edge computing. To perform this step, it is required to have a subscription, even though the Azure Speech recognition service offers free usage for limited requests for the speech recognition.
3. Install Ubuntu Server 18.04 [32] and IoT Edge runtime software [33] on the edge devices Category I and II.
4. Register the IoT edge devices in the Azure IoT Hub [31].
5. Configure the IoT edge devices and connect them to the IoT Edge Hub in the cloud Azure.
6. Connect the actuator and the microphone to the IoT edge device from Category I.
7. Define a deployment JSON file in the Azure IoT Hub portal with the containers Voice recognition module (Device from Category I) and Azure Speech recognition service (Device from Category II)
8. Trigger the deployment of mentioned containers, in other words, modules.

9. Wait for the initialization of the modules and use the system. During the initialization of the Voice recognition module, necessary configuration steps are performed automatically to allow connection and usage of the microphone.
10. Deployment of the mentioned modules can be verified through the Azure Iot Hub portal [31]. In the portal can be seen:
    - If the IoT edge devices are online,
    - If the deployment of mentioned modules were successful,
    - Error message in case an error occurs.

4. Experimental Verification

In terms of the proposed architecture in Figure 3 and the above methodology, the IoT edge experimental devices were implemented. A part of them can be seen in Figure 14. By using these devices, all modules were verified in cooperation with each other according to Figure 4. After establishing a connection to the cloud Azure and a simple installation of containers (Links to the containers in Section 5), the user can use this modern HMI and send voice commands to the connected actuator.

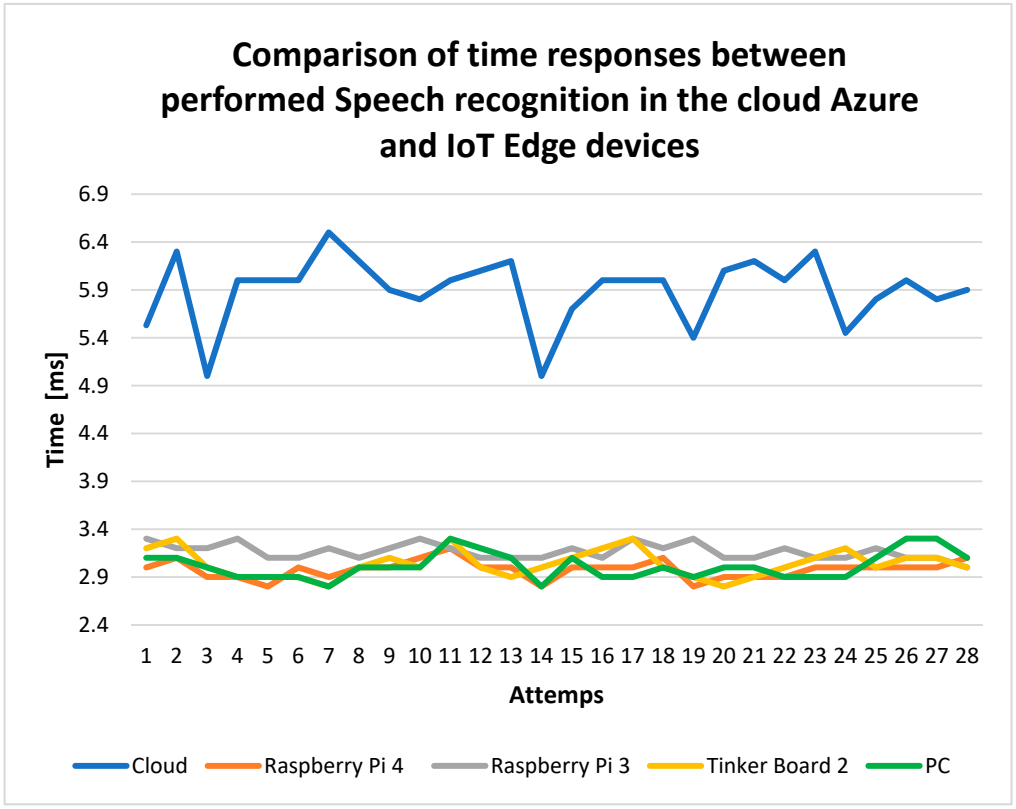

Figure 14. Comparison of time responses between performed speech recognition in the cloud Azure and IoT edge devices.

In Figure 15, the real implementation of the voice control actuator using Bluetooth headphones for better comfort and freedom is shown. For the demonstration, the IoT edge devices were used from:

- Category I—PC with 4 GB RAM and processor Intel Core i3-2330M, Thinker Board 2, Raspberry Pi 3 and 4;
- Category II—PC with 12 GB RAM and processor Intel Core i7-8850h.

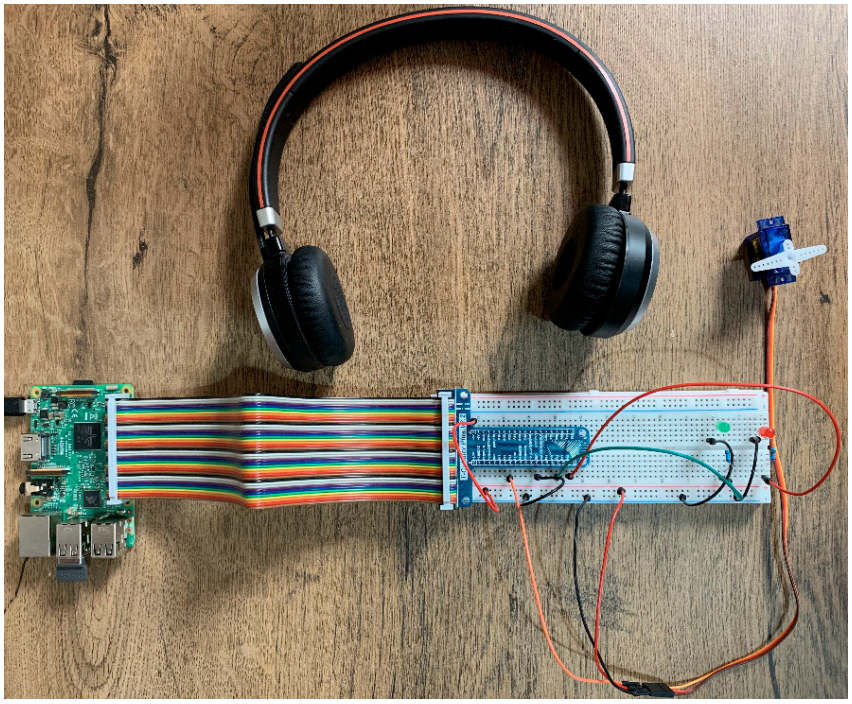

Figure 15. Real implementation of the voice control actuator using Bluetooth headphones for better comfort and freedom.

The created architecture (Figure 3) was successfully tested on PC, Tinker Board 2, Raspberry Pi 3 and 4 as well. As an operation system for the mentioned IoT edge devices, Ubuntu server 18.04 was used. Testing of the experiment was performed in the time schedule between morning and evening at 24 °C (4 days, 7 h period). In total, 28 attempts were performed by each edge device and the cloud Azure. Based on Figure 14, it can be stated that speech control performed by the edge devices has a better latency in comparison to using the speech recognition in the cloud Azure. In the performed experiments, the response time for the speech recognition performed by the IoT edge devices was around 2.97 ms. That is approximately 58.37% better than the cloud Azure response time (Figure 14). However, even if the Azure Speech Recognition service runs in the cloud Azure, the response time for voice transformation into text is around 5.9 ms when using an internet connection with the speed of 15 Mbps.

During the verification and testing of the created architecture, an outage of internet connection was simulated. The result of this simulation was that speech recognition was running stable and without any issues using IoT edge devices. Edge computing was performed in this scenario. The time responses were almost similar to the responses that we obtained in Figure 14.

Distribution and maintenance were verified by remote updating of modules on the IoT edge devices by using the IoT Hub portal in the cloud Azure. Updating of modules were realized many times during the development phase, as it can be seen in a Docker image name of the voicerecognitionmodule:1.7.11. Deployment of modules always went smoothly, and it took approximately 1 min using an internet connection with the speed of 15 Mbps.

The phrase triggering of the speech control was also verified by using commands without the trigger phrase. In this case, commands were not recognized and performed. The speech control was triggered only when the phrase “Hey Bennie” was used.

5. Results

The result of the work are functional modules-containers, the system implemented according to the proposed methodology. The benefits and main goals of the proposed architecture in comparison to other implantations can be formulated as follows:

1. Stability and low latency—As described in Section 4, the response time is shorter when the processing is performed by the IoT edge device and not in the cloud Azure. Speech recognition performed by the IoT edge devices was around 2.97 ms, meaning an approximately 58.37% better response time than the cloud Azure response time (Figure 14). Even if the Azure Speech Recognition service runs in the cloud Azure, the voice transformation into text is around 5.9 ms when using an internet connection with the speed of 15 Mbps.

As it was shown in the experiment, the container for the Azure Speech Recognition service can run locally in the network. Many big companies are performing voice controlling only via their cloud services, or they require a special local infrastructure setup that fits their needs. The problem with the internet connection can be seen when using Siri on iPhone or Alexa from Amazon. If the internet connection is switched off, they cannot be used. By running the Azure Speech Recognition service locally, we are achieving increased stability and data privacy.

2. Portability—Developed Voice recognition module can be deployed on Windows or Linux operating systems thanks to the framework. NET Core. As the Voice recognition module is a Linux container, it can be reused in other IoT solutions such as AWS IoT Greengrass [34] or Siemens MindSphere (currently available only for developers) [35]. These solutions use Linux containers as well. The usage of the proposed modules in the solutions from Amazon or Siemens will require modification of communication routers, but the core functionality of modules will stay the same. Furthermore, the usage on the operation system Windows will require a small adaption, the Voice recognition module would have to be rebuilt as a Windows container.
3. Data privacy—Increased data privacy when running the Azure Speech Recognition service locally on the IoT edge devices.
4. Distribution and maintenance—The solution can be easily distributed to many other IoT edge devices in a short time. It is only necessary to have the IoT edge device that will fulfill the requirements and it is connected to the Azure IoT Hub in the cloud Azure. Updates of modules are performed remotely by the cloud Azure. There is no need to have physical access to the devices. Furthermore, migration to different devices with similar architecture is quick and smooth. The migration from old Raspberry Pi 3 to 4 was completed approximately in 15 min.

Created modules can be stored and made available on the Azure Marketplace or they can be stored in the public Docker registry. Thanks to the Azure Marketplace, applications can reach many developers and customers. IoT Edge modules can be published and run on many IoT edge devices that support containers.

5. The endpoint for the Azure Speech Recognition service can be easily adjusted in the JSON file in the Voice recognition module without a code change. The endpoint for the Azure Speech Recognition service can be a local Hostname, such as "LocalAzureSpeechRecognition" or a local static IP address. Furthermore, the commands for the voice recognition can be easily adjusted in the JSON file in the Voice recognition module, without code changes, and the same for the PINs configuration. All of the mentioned adjustments can be performed while the Voice recognition module is up and running. Containers can be simply pulled and run from the following repositories:
    1. Container for the Voice recognition module (designed and created module, the latest stable version is 1.7.11) for Category I:
    Commands: docker pull researchphd.azurecr.io/voicerecognitionmodule:1.7.11-arm64v8 *docker run researchphd.azurecr.io/voicerecognitionmodule:1.7.11-arm64v8*

2. Container for the Azure Speech Recognition service (official container from Microsoft, always the latest version is downloaded) [36]:

docker pull mcr.microsoft.com/azure-cognitive-services/speechservices/speech-to-text docker run –rm -it -p 5000:5000 –memory 8g –cpus 4 mcr.microsoft.com/azure-cognitive-services/speechservices/speech-to-text Eula=accept Billing="CreatedBillingEndpoint" ApiKey="CreatedApiKey"

6. Voice recognition module—The created Linux container is prepared to automatically take the USB headset that is connected to the IoT edge device. It contains all libraries and the configuration that is needed to access the microphone from the underlying system. No input is required from the end user to make Voice recognition module functional. By default, Linux container image does not support the usage of the microphone from the underlying system; therefore, it was necessary to set up this functionality in the Voice recognition module. The Voice recognition module is always running locally as it is responsible for the control of the actuator. As mentioned above, the module also supports the phrase triggering. The phrase triggering leads to bigger reliability of the architecture, as the voice control is only performed when desired.

In Table 3, the main differences between the created architecture and the architectures used in different implementations are described. The mentioned architectures are the closest to the proposed architecture from the article.

Table 3. Comparison between created architecture and existing architectures.

	Speech Recognition Performed on the Edge Device	Containerized Solution	Easy Update and Distribution of Solution to Many Devices	Multilanguage Support	Phrase Triggering
Created architecture from the article	Yes	Yes	Yes	Yes	Yes
Valera Román [11]	No	No	No	Yes	Yes
Yvanoff-Frenchin [12]	No	No	No	Yes	Not mentioned
Tahseen Ali [37]	Yes	No	No	No	Not mentioned
Martinek [38]	Yes	No	No	Not mentioned	Not mentioned
Benítez-Guijarro [39]	No	No	No	Not mentioned	Yes
Nasef [40]	No	No	No	Yes	Not mentioned
Kološvari [41]	No	No	No	Yes	Not mentioned

Even article [42] does not deal exactly with speech recognition, which can be taken as proof that edge computing is used and beneficial.

The limitations of the proposed architecture can be listed:

- Dependency on the operation system Linux with the architecture ARM64. Note: Linux container can be transformed to Windows container. This option requires recompilation of the Voice recognition module with a proper base image [43].
- Setup of the Azure Speech Recognition service and the IoT Hub in the cloud Azure. This setup requires knowledge about cognitive services [44] and IoT [31] in the cloud Azure. Furthermore, in other implementations, it was possible to see the usage of speech services from the different cloud providers, where it is necessary to set up these services in the cloud.

6. Conclusions

The popularity of voice control in IoT and Industrial IoT is growing. In the future, most companies will have to face this challenge. Voice control will be able to help in situations where physical contact with devices is difficult or even impossible. It also brings a certain level of comfort to the end users. Voice control has already found its place in smartphones, where it is already widely used, and it is being slowly adopted in the industry, healthcare

and research laboratories. This article presents a new architecture and methodology of design and implementation of a modern HMI for the voice control of the actuator. Along with the voice control of the actuator, the solution can also be reused for the setting and the confirmation of parameters, restrictions, limits, alarms of various filters and sensors. The proposed architecture includes a combination of new approaches and modern digital technologies, container–edge–cloud, artificial intelligence for speech recognition, and its own software components, which create a complete functional system. This system has been experimentally validated by running the created containers on four different IoT edge devices—Thinker Board 2, PC, Raspberry Pi 3 and 4. These new approaches bring benefits to both the end user and the developer, such as data privacy and good latency by practicing edge computing, easy deployment thanks to containerization and management by using the cloud Azure.

Author Contributions: Conceptualization, L.B. and P.D.; methodology, L.B.; software, L.B.; validation, L.B. and R.P.; writing—original draft preparation, L.B.; writing—review and editing L.B., R.P. and P.D.; funding acquisition, P.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Slovak Research and Development Agency, grant No. APVV-17-0190 and the Slovak Cultural Educational Grant Agency, grant No. 039STU-4/2021.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The paper was partially supported by the Slovak Research and Development Agency, grant No. APVV-17-0190 and the Slovak Cultural Educational Grant Agency, grant No. 039STU-4/2021.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Alexa. Available online: https://developer.amazon.com/en-US/alexa (accessed on 2 June 2021).
2. Siri. Available online: https://www.apple.com/siri/ (accessed on 2 June 2021).
3. Cortana. Available online: https://www.microsoft.com/en-us/cortana (accessed on 2 June 2021).
4. Google Assistant. Available online: https://assistant.google.com (accessed on 2 June 2021).
5. 73% of Drivers Will Use an In-Car Voice Assistant by 2022: Report. Available online: https://voicebot.ai/2019/11/17/73-of-drivers-will-use-an-in-car-voice-assistant-by-2022-report (accessed on 1 May 2021).
6. Voice Assistant Market Size USD 7.30 Billion by 2025, Registering a 24.32% CAGR—Report by Market Research Future (MRFR). Available online: https://www.globenewswire.com/en/news-release/2021/06/23/2252069/0/en/Voice-Assistant-Market-Size-USD-7-30-Billion-by-2025-Registering-a-24-32-CAGR-Report-by-Market-Research-Future-MRFR.html (accessed on 10 July 2021).
7. From the Cloud to the Edge. Available online: https://rtview.com/from-the-cloud-to-the-edge/ (accessed on 11 July 2021).
8. 100 ms of Latency Cost This Company 1% in Sales. Available online: https://www.presstitan.com/100ms-latency/ (accessed on 15 July 2021).
9. Internet of Things (IoT) Connected Devices Installed Base Worldwide from 2015 to 2025. Available online: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/ (accessed on 16 July 2021).
10. 10 Reasons Why Privacy Rights Are Important. Available online: https://www.humanrightscareers.com/issues/reasons-why-privacy-rights-are-important/ (accessed on 20 July 2021).
11. Valera Román, A.; Pato Martínez, D.; Lozano Murciego, Á.; Jiménez-Bravo, D.M.; de Paz, J.F. Voice Assistant Application for Avoiding Sedentarism in Elderly People Based on IoT Technologies. *Electronics* **2021**, *10*, 980. [CrossRef]
12. Yvanoff-Frenchin, C.; Ramos, V.; Belabed, T.; Valderrama, C. Edge Computing Robot Interface for Automatic Elderly Mental Health Care Based on Voice. *Electronics* **2020**, *9*, 419. [CrossRef]
13. Speech. Available online: https://azure.microsoft.com/en-us/services/cognitive-services/speech-services/ (accessed on 30 July 2021).
14. Cloud Computing vs. Edge Computing: Friends or Foes? Available online: https://www.forbes.com/sites/forbestechcouncil/2020/03/05/cloud-computing-vs-edge-computing-friends-or-foes/ (accessed on 2 August 2021).
15. Cloud vs. Edge. Available online: https://www.redhat.com/en/topics/cloud-computing/cloud-vs-edge (accessed on 2 August 2021).

16. Memon, K.S.; Nisar, K.; Hijazi, M.H.A.; Chowdhry, B.S.; Sodhro, A.H.; Pirbhulal, S.; Rodrigues, J.P.C.J. A survey on 802.11 MAC industrial standards, architecture, security & supporting emergency traffic: Future directions. *J. Ind. Inf. Integr.* **2021**, *24*, 100225. [CrossRef]
17. Difference between Edge Computing vs. Cloud Computing? Available online: https://www.akira.ai/blog/edge-computing-vs-cloud-computing/ (accessed on 2 July 2021).
18. What Is Containerization? Available online: https://www.redhat.com/en/topics/cloud-native-apps/what-is-containerization (accessed on 3 August 2021).
19. Zhang, Q.; Liu, L.; Pu, C.; Dou, Q.; Wu, L.; Zhou, W. A Comparative Study of Containers and Virtual Machines in Big Data Environment. In Proceedings of the 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), San Francisco, CA, USA, 2–7 July 2018. Available online: https://arxiv.org/pdf/1807.01842.pdf (accessed on 3 August 2021).
20. Yadav, A.K.; Garg, M.L.; Mehra, R. Docker Containers Versus Virtual Machine-Based Virtualization: Proceedings of IEMIS 2018. In *Emerging Technologies in Data Mining and Information Security*; Springer: Singapore, 2019; Volume 3. [CrossRef]
21. Docker. Available online: https://www.ibm.com/cloud/learn/docker (accessed on 3 August 2021).
22. Azure IoT Edge. Available online: https://azure.microsoft.com/en-us/services/iot-edge (accessed on 27 March 2021).
23. Raspberry Pi 3. Available online: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/ (accessed on 7 September 2021).
24. Raspberry Pi 4. Available online: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/ (accessed on 7 September 2021).
25. Tinker Board 2. Available online: https://tinker-board.asus.com/product/tinker-board-2.html (accessed on 8 September 2021).
26. Understand the Azure IoT Edge Runtime and Its Architecture. Available online: https://docs.microsoft.com/en-us/azure/iot-edge/iot-edge-runtime (accessed on 15 April 2021).
27. Understand Azure IoT Edge Modules. Available online: https://docs.microsoft.com/en-us/azure/iot-edge/iot-edge-modules (accessed on 15 April 2021).
28. Azure IoT Edge Security Manager. Available online: https://docs.microsoft.com/en-us/azure/iot-edge/iot-edge-security-manager (accessed on 25 April 2021).
29. Azure IoT Edge Supported Systems. Available online: https://docs.microsoft.com/en-us/azure/iot-edge/support?view=iotedge-2020-11 (accessed on 25 June 2021).
30. Container Registry. Available online: https://azure.microsoft.com/en-us/services/container-registry/ (accessed on 9 September 2021).
31. Create an IoT Hub Using the Azure Portal. Available online: https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-create-through-portal (accessed on 25 June 2021).
32. Ubuntu 18.04.5 LTS (Bionic Beaver). Available online: https://releases.ubuntu.com/18.04.5/ (accessed on 25 June 2021).
33. Install or uninstall Azure IoT Edge for Linux. Available online: https://docs.microsoft.com/en-us/azure/iot-edge/how-to-install-iot-edge?view=iotedge-2020-11 (accessed on 26 June 2021).
34. AWS IoT Greengrass 1.10 Provides Support for Docker Containers and Management of Data Streams. Available online: https://aws.amazon.com/about-aws/whats-new/2019/11/aws-iot-greengrass-supports-docker-containers-management-data-streams (accessed on 8 May 2021).
35. Cloud Foundry How Tos—Docker Container in Cloud Foundry. Available online: https://developer.mindsphere.io/paas/howtos/howtos-docker-in-cloudfoundry.html (accessed on 29 April 2021).
36. Install and Run Docker Containers for the Speech Service APIs. Available online: https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/speech-container-howto?tabs=stt%2Ccsharp%2Csimple-format (accessed on 29 July 2021).
37. Tahseen Ali, A.; Abdullah, H.S.; Fadhil, M.N. Voice recognition system using machine learning techniques. *Mater. Today Proc.* **2021**. [CrossRef]
38. Martinek, R.; Vanus, J.; Nedoma, J.; Fridrich, M.; Frnda, J.; Kawala-Sterniuk, A. Voice Communication in Noisy Environments in a Smart House Using Hybrid LMS+ICA Algorithm. *Sensors* **2020**, *20*, 6022. [CrossRef] [PubMed]
39. Benítez-Guijarro, A.; Callejas, Z.; Noguera, M.; Benghazi, K. Coordination of Speech Recognition Devices in Intelligent Environments with Multiple Responsive Devices. *Proceedings* **2019**, *31*, 54. [CrossRef]
40. Nasef, M.; Sauber, A.; Nabil, M. Voice gender recognition under unconstrained environments using self-attention. *Appl. Acoust.* **2021**, *175*, 107823. [CrossRef]
41. Kološvari, A.; Stojanović, R.; Zupan, A.; Semenkin, E.; Stanovov, V.; Kofjač, D.; Škraba, A. Speech-recognition cloud harvesting for improving the navigation of cyber-physical wheelchairs for disabled persons. *Microprocess. Microsyst.* **2019**, *69*, 179–187. [CrossRef]
42. Magsi, H.; Sodhro, A.H.; Al-Rakhami, M.S.; Zahid, N.; Pirbhulal, S.; Wang, L. A Novel Adaptive Battery-Aware Algorithm for Data Transmission in IoT-Based Healthcare Applications. *Electronics* **2021**, *10*, 367. [CrossRef]
43. Create a Base Image. Available online: https://docs.docker.com/develop/develop-images/baseimages/ (accessed on 10 September 2021).
44. Azure Cognitive Services. Available online: https://azure.microsoft.com/en-us/services/cognitive-services/ (accessed on 10 September 2021).