CZECH TECHNICAL
UNIVERSITY
IN PRAGUE

F3

Faculty of Electrical Engineering
Department of Conrol Engineering

Bachelor's Thesis

Displaying the status of industrial devices and their supporting data in augmented reality

Zobrazení stavu průmyslových zařízení a podpůrných dat v rozšířené realitě

Jan Andrys
Cybernetics and Robotics

19.5.2021
Supervisor: Ing. Vojtěch Janů, Ing. Pavel Burget, Ph.D.

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Andrys Jan** Personal ID number: **474437**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Control Engineering**

Study program: **Cybernetics and Robotics**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Displaying the status of industrial devices and their supporting data in augmented reality

Bachelor’s thesis title in Czech:

Zobrazení stavu průmyslových zařízení a podpůrných dat v rozšířené realitě

Guidelines:

1. Get acquainted with the Unity development platform and its utilization of the development of augmented reality (AR) applications for Microsoft HoloLens2.
2. Choose a suitable library for the detection and localization of industrial devices in the augmented reality.
3. Design an architecture, which allow online transfer of data from different sources to the server, whereas the sources may be industrial device, cloud applications etc. The architecture must also allow to transfer data to the HoloLens headset. These data will be prepared according to the pre-defined rules.
4. Implement the designed architecture as a functional prototype.

Bibliography / sources:

[1] Mixed Reality Toolkit manual:
https://docs.microsoft.com/en-us/windows/mixed-reality/develop/development?tabs=
unit
[2] Unity manual: https://docs.unity3d.com/2018.4/Documentation/Manual/index.html
[3] HoloLens manual: https://docs.microsoft.com/en-us/hololens/hololens2-options?tabs=device

Name and workplace of bachelor’s thesis supervisor:

Ing. Vojtěch Janů, Testbed - CIIRK

Name and workplace of second bachelor’s thesis supervisor or consultant:

Ing. Pavel Burget, Ph.D., Testbed, CIIRC

Date of bachelor’s thesis assignment: **21.12.2020** Deadline for bachelor thesis submission: **21.05.2021**

Assignment valid until:
by the end of summer semester 2021/2022

Ing. Vojtěch Janů	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

. | Date of assignment receipt

Student’s signature

Acknowledgement / Declaration

I would like to thank my family for their intensive support during my studies. Furthermore, I appreciate my supervisor Ing. Vojtech Janu, for willingness, help, and given consultations. Lastly, I thank Ing. Pavel Burget Ph.D the head of Testbed in CIIRC, for an opportunity to work with the best available technology.

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague, 19.5.2021

Abstrakt / Abstract

U průmyslových zařízení je diagnostika jednou z nejdůležitějších částí, díky její správné funkčnosti je možné zabránit ztrátám ve výrobě. Nový pracovníci si často musí projít několik školení, aby pochopil, jak funguje použitý typ diagnostického systému v továrně a uměli ho správně využít. Nejen s urychlením procesu školení pracovníka může pomoci aplikace popsaná v této práci.

Práce má za cíl vytvořit funkční prototyp aplikace pro přenos dat z průmyslových zařízení do zobrazovacího zařízení HoloLens 2 a přijatá data zobrazit na konkrétních místech v prostoru, podle toho, kde se odesílající průmyslové zařízení nachází. Může jít jak o pevně umístěná, tak o pohybující se průmyslová zařízení, jejichž detekci zajišťuje software Vuforia. U zobrazování je možné vybrat typ zobrazení datových zpráv podle nastavených vzorů. Samotnou komunikaci zajišťuje protokol MQTT a zasílané zprávy jsou rozděleny na datové a kontrolní.

Na závěr bylo provedeno několik zátěžových testů, které ověřili kvalitu přijímaných a odesílaných zpráv.

Pro každý druh zařízení jsou jednotlivé programy napsány v jazyku C# a vývoj pro HoloLens 2 byl možný díky Unity Editoru a knihovně MRTK.

Klíčová slova: Rozšířená realita, Vuforia, Unity, C#, HoloLens 2, MRTK, MQTT, Průmysl 4.0

For industrial devices, the diagnostic system is one of the most important parts of the industrial facility. If it works well, it can prevent loss in production. The new worker must often complete several training courses to understand how the diagnostic system works and how to use it. The system described in this thesis can help with the worker's training.

This thesis aims to create a functional prototype application for data transmission from the industrial devices to the visualization device HoloLens 2. It shows the data on the specific places in the facility according to the place of the transmitted industrial devices. The industrial devices can be both fixed or mobile. Their detection ensures Vuforia software. For the visualization, the user can choose the type of data visualization according to offered patterns. The communication ensures MQTT protocol, and the messages are divided into groups of control and data.

In the evaluation part, several verifying tests that validated the quality of transmission were done.

The individual programs are written for every kind of device in C#, and the HoloLens 2 development is possible thanks to Unity editor and the MRTK library.

Keywords: Augmented reality, Vuforia, Unity, C#, Hololens 2, MRTK, MQTT, Industry 4.0

Contents /

Tables /

Chapter 1
Introduction

During the last ten years, Industry 4.0 research publications and facility implementations are growing exponentially [1]. Many manufacturing companies still have a big problem with the implementation of Industry 4.0, but they will have to change their production architecture for more effectiveness of their facilities [2]. To decrease the cost of products, manufacturing companies have to consider all production expenses and decrease them [3]. Virtual and augmented reality can be one of the opportunities how to decrease these expenses.

Nowadays, Virtual reality is one of the fastest growing fields of study. As a consequence, many developers try to integrate it into everybody's life. Examples of such attempts are integration into virtual shopping [4], teaching methods [6], control and display production system [5] or remote maintenance [7].

This thesis focuses on displaying and controlling production processes in virtual or augmented reality. Due to this work, it should be possible to come to the factory of the future, look at the product, and see all the information in augmented reality. For example, the displayed information can be the product's final destination, the number of parts that have to be delivered, or when the product will be finished.

At the same time, it is possible to use this technology for servicing different kinds of production lines simultaneously. Even in small production facilities, it is hard for a serviceman to know all the devices and to be able to repair anything in time. Augmented reality allows arranging calls with the producer of the malfunctioning device to navigate the serviceman by showing parts of the equipment that have to be repaired [8].

None of these technologies could work without reliable communication among all parts of the production system, such as visualization devices, manufacturing devices, and control systems. In this thesis, those devices are HoloLens 2, communication server, and production line devices. The role of the server is to provide all data of the concrete device from the production line, such as manipulators or a control system to the visualization devices [9].

The chapters of this thesis are organized as follows. First, the available devices for augmented reality will be described 2.1. Next HoloLens 2 glasses will be described in detail 2.1.2. Then will follow a description of the development frameworks for augmented reality 2.3, such as Vuforia Engine 2.3.3 with a mention about developer platforms 2.4 and Unity development 2.5. Finally, the summary of communication protocols 2.6 and a broker 2.7 suitable for augmented reality will be defined.

In the next chapter, the proposed architecture of the whole system is described 3; what key tasks individual components have 3.1 and how communication works between them 3.2.

The implementation chapter describes how the key features are implemented with a particular focus on the visualization device 4.

In the next chapter the performed experiments are described in the following order; the different image recognition according to the use of different QR codes 5.1, the permeability tests with changing size of messages determining the correct delay between

messages 5.2. Furthermore, the last experiment shows the best message size for the most desirable delivery time of individual messages 5.3.

Chapter 2
Theory

2.1 AR Devices

The first mention of virtual reality devices appeared in 1962 when Morton Heilig introduced his 4D cinema called Sensorama. It enabled watching scenes on a screen, hear a sound and feel a smell. In 1979 a Leep company came up with the concept of virtual reality where there was a screen in the field of vision. NASA used this type of system for training their astronauts. Nowadays, several different technologies of virtual reality are developed, which are much more advanced.

First of all, there are two different categories of `reality` devices. The virtual reality devices and augmented reality devices. In virtual reality devices or glasses, only the screen is in the user's field of vision. In a case when the screen is turned off, the user loses any eye contact with reality.

In case of the augmented reality, the devices use a unique system for the projection of scenes on a piece of glass or plastic in front of the eyes. The technology of projection differs for every producer. When the screen turns off, this device works like standard goggles, and the user can see reality without any limitation.

Both augmented and virtual reality devices have some common control parts. An example of such can be a gyroscope, accelerometer, or built-in compass that can be used to fast detect turning and roll the user's head. On the other hand, each type of device has its way how to interact with the user and the real world.

Every AR/VR platform establishes its position in space differently, and it greatly depends on the type of use. For some devices focused on the game industry, the user must connect to the computer by a cable. The device serves as a plane screen with the possibility of perfect positioning. External IR sensors are the ones that give the accurate position of the device in space.

For other devices, the connection is wireless. These devices without wired connection to a computer need to calculate the position in space by themself. The main source of position for those devices is 3D cameras. Magic leap One and HoloLens 2 are devices that are using the previously mentioned cameras.

2.1.1 Magic Leap One

The device developed in 2017 by the company of the same name, it is excellent primarily for good performance. The device is compounded of the headset, small but powerful computer, and remote control. Unfortunately, this device has a considerable disadvantage. The user spends a long time putting on all components before using the device.

The device should work on battery for 3 hours.

2.1.2 HoloLens 2

HoloLens 2 device inherits a lot from the old model HoloLens 1, which does not offer a wide range of HoloLens 2 abilities [10], are mentioned in the list below.

- Head tracking
- Eye tracking
- Microphone array
- Voice commands
- Spatial mapping
- WiFi, Bluetooth

These technologies for augmented reality determine our position in the world thanks to cameras that create a 3D model of space accurately enough to locate itself. Because of that, these devices are not suitable for use outside in open space. The created 3D model is simple with a precision of around five centimeters and approximately resolution of 10 square centimeters.

These two generations of the device are different in the manner of finding a position in space. HoloLens 2 itself can detect where it is in previously mapped space, or it can create a new one. In the previous generation, the HoloLens 1, the user had to calibrate the position using special QR codes that provide the right origin of coordinates.

The HoloLens 2 can detect the user's arms and hands and knows in real-time where in space are all joints of each user's fingers. However, users have to keep in mind that hands have to be in the camera range for the proper detection.

Thanks to the knowledge of each finger's joint position, it is possible to detect the different gestures [11]. Their brief list is stated below.

- rotation of the hand
- touching fingers
- near control - normal grasp of an object
- distant control - use rays for manipulation of an object

HoloLens 2 uses the recognition of four low-resolution cameras, one IR distance sensor, and unique IR cameras to detect an eye's move. Furthermore, as stated above, it is equipped with an accelerometer, gyroscope, and compass. The goggles contain a high-resolution camera for streaming an augmented reality.

Experiments in this thesis are done with the use of HoloLens 2 glasses.

2.2 Revolution Pi

The project of this thesis is developed for use in industrial facilities. It is designed for sending data from industrial devices to the Server and HoloLens. Revolution Pi is one of the industrial devices used for testing the project.

Revolution Pi is a modular and inexpensive industrial PC based on the well-known Raspberry Pi. It is produced in many variants for a specific type of use and satisfies European regulation EN 61131-2 or IEC 61131-2 for industrial stability. In the end, it consists of the Raspberry Pi Compute Module and one of the Revolution Pi components. The Revolution Pi has a specially adapted Raspbian operating system, which is equipped with a real-time patch. The Raspbian ensures that most of the applications developed for the Raspberry Pi can also run on the Revolution Pi.

2.3 AR framework

Augmented reality frameworks are packages of software tools for image recognition. Thanks to them, it is possible to develop augmented reality easily. The frameworks are used mainly for the detection of absolute coordinates in space.

The frameworks stated below can work with one camera, but some can utilize more cameras simultaneously or use a depth sensor. Examples of common devices with depth sensor are IPad, Samsung Galaxy S20 Ultra and other.

2.3.1 AR Kit

AR Kit is a development kit for augmented reality developed by Apple for devices with iOS operating systems only. It is possible to add other 2D or 3D objects to a scene and interact with them in the real world.

2.3.2 AR Core

It is a set of development tools developed by Google, which is primarily developed for Android devices, but it also allows to be used on iOS devices through ARkit. Mobile phones and tablets with Android versions higher than 7.0 and iOS versions higher than 11.0 can use this type of tool.

As is stated before, it is possible to detect relative coordinates in world space. Thanks to the collected point cloud, the framework enables the detection of surfaces, areas, planes and determines their size. The framework can assume the volume and direction of light in space and change the shade of additional objects in the scene accordingly.

The AR Ruler App is an example of an application utilizing AR core, which allows the measurement of real objects only by moving a mobile phone above the measured object [29].

2.3.3 Vuforia

This complex development kit for augmented reality is built on AR Kit and AR Core. Vuforia developers added extra functions and tools for interaction with reality. Vuforia has several applications for use in the manufacturing industry [12].

It is good to know that developers can use these products gratis for non-commercial use only, but they can use a limited number of Vuforia products. For example, there is a limited number of models for the model target.

The most important feature of the Vuforia engine is recognition and following of object according to its image or some pattern of the object. This type of detection is called image detection, and Vuforia works very well even with rotation and move of the object. The implementation of this function is not very hard.

Object detection is the next Vuforia engine feature. However, this type of object detection is quite obsolete. The feature Model target has better results, which is the detection of an object according to the object's CAD or 3D model. Using this function is not so easy as Image target. Application Model Target Generator provided by Vuforia creates a unique file from the CAD model for correct detection. This app enables to set an initial position of an object for detection. Whenever users want to start detecting real objects, they have to see them in this initial position.

Another important function of Vuforia Engine is the Area Target. After making a 3D scan of some space and creation a space model, using Area Target Generator provided by Vuforia, the device detects its position in space with massive precision. This feature's main advantage is a precisely defined coordinate system, and for a developer, the possibility to add an object to an exact place in space.

These features, the opportunity to easily add it to the Unity, which supports development for HoloLens, and using Vuforia for image recognition in different research articles, for example, [6] are the reasons for using Vuforia in this thesis. The comparison of AR framework and reasons for using it are stated in the section below 3.6.

2.4 Developer platforms

There are many robust languages suitable for production use. One of the most widely used languages in the manufacturing environment is C#. This programming language is based on C++. Its development started around 2000 by Microsoft. It is approved as ISO (ISO/IEC 2327:2012 and ISO/IEC 23270:2006) from 2003. All code written in this thesis is written in C#, and all developer platforms stated below support C#. First, two of them use .NET Ecosystem, which is based on .NET standard library. .NET is open-source.

2.4.1 .Net Framework

It is .NET based platform that has final version 4.8, and its development process has already ended. It primarily supports Microsoft Windows operating systems development. Its development started in the late 1990s as part of the .Net strategy. It consists of two main parts: a Framework Class Library (FCL), which provides language interoperability, and Common Language Runtime, an application virtual machine that provides security, memory management, and exception handling.

2.4.2 .Net Core

It is an excellent choice for cross-platform development in C#. .NET Core apps are supported on Windows, Linux, and iOS. It is the successor to .NET Framework released under the MIT License. The first version of .NET Core was released in 2016, but .NET Core 3 adds support for Windows desktop application development and enables simple desktop applications from 2019. According to Microsoft [20], all new projects should be developed in .NET Core because it will have hardware support in the future.

2.4.3 UWP

Universal Windows Platform is a digital platform for Microsoft's operating systems. This platform helps develop a universal application for Windows 10, Xbox One, and Hololens without rewriting for each. In the past, using Windows application programming interfaces (APIs) was the only way how to develop an application for individual Microsoft's operating systems. This platform contains a universal API that enables the running of applications on mentioned devices. When a developer wants to use Unity for development, UWP is the standard platform for building prototypes for HoloLens.

2.5 Unity development

Unity is a game engine that supports desktop, mobile, console, and virtual reality platforms. Thanks to Unity is possible to create games and experiences in both 2D and 3D, and the engine offers primary scripting API in C#. It is the simplest way how to develop an application for HoloLens.

2.5.1 Object creation in Unity

The structure of objects in Unity is essential for understanding the techniques for object creation. The main building blocks of Unity are objects which have some components. Each object has a transform component by default, which sets the position, rotation, and scale of the object in the world. For each object, it is possible to create child objects, which cause a tree structure of the objects. Besides the transform component, scripts

are other components that ensure the properties of the object. The scripts provide interaction with users, textures, and so on.

Unity has many options how to create new objects, but there are two main ways. The first way creates a new object and adds components according to developer choice. The second way creates a new object based on a different object called prefab. The second way of object creation is more popular for Unity development than the first one.

It can be used by the command `Instantiate`, which load prefab from memory and creates a new object based on a prefab with prefab's components and prefab's children for the second way of object creation.

Among the main commands used to determine an object's properties belong `Addcomponent` and `Getcomponent`. Addcomponent command is relatively straightforward. It is the method that adds a new component to an object. Thanks to the Getcomponent command, it is possible to set the properties of the found component.

2.5.2 MRTK library methods

This library contains lots of basic scripts, objects, and prefabs for proper HoloLens functionality. The button prefab is mainly used in this project. The prefab contains several scripts. These scripts enable the user to interact with the program and run different methods. The interaction has two forms, near interactions and far interactions. Pointers ensure the far interactions. The whole MRTK input system is shown in figure 2.1.

Data visualization objects have scripts for rotation and scaling. These scripts are called Bounds control, which has several ways to visualize the object's bounds, but a unique collider script determines the size and position of the bounds. The second way for interaction with objects is the Object manipulator script, which allows the user to interact with objects more realistically. When this script is implemented, the user can grasp the object and move and rotate with it as well as with a real object.

An important feature of the MRTK library is a unique system keyboard used for writing user names and passwords.

The last valuable scripts are Solvers, which help the developers with simple tasks. It processes raw input data from the hand tracking, for example, allowed using palm up menu. Or a following solvers' features which determine the object position according to different object position or position of the user. For example, script `InBetween` ensures an object a position between two different objects.

The whole MRTK library architecture is shown on the image 2.2 taken from Microsoft docs [21].

2.6 Communication protocols

Communication protocols are formal descriptions of digital message formats and rules. In short, it is a language for computers and other devices to communicate the exchange of messages between each other. It can cover authentication, error detection, and correction.

This thesis works with IoT communication protocols, which are based on IP Communication protocols. Among the most used IP protocols belong TCP - Transmission Control Protocol and UDP - User Datagram Protocol. TCP is a connection-oriented protocol. It uses a three-way handshake and is slower than UDP [13]. UDP is a connection-less protocol that does not use the three-way handshake.

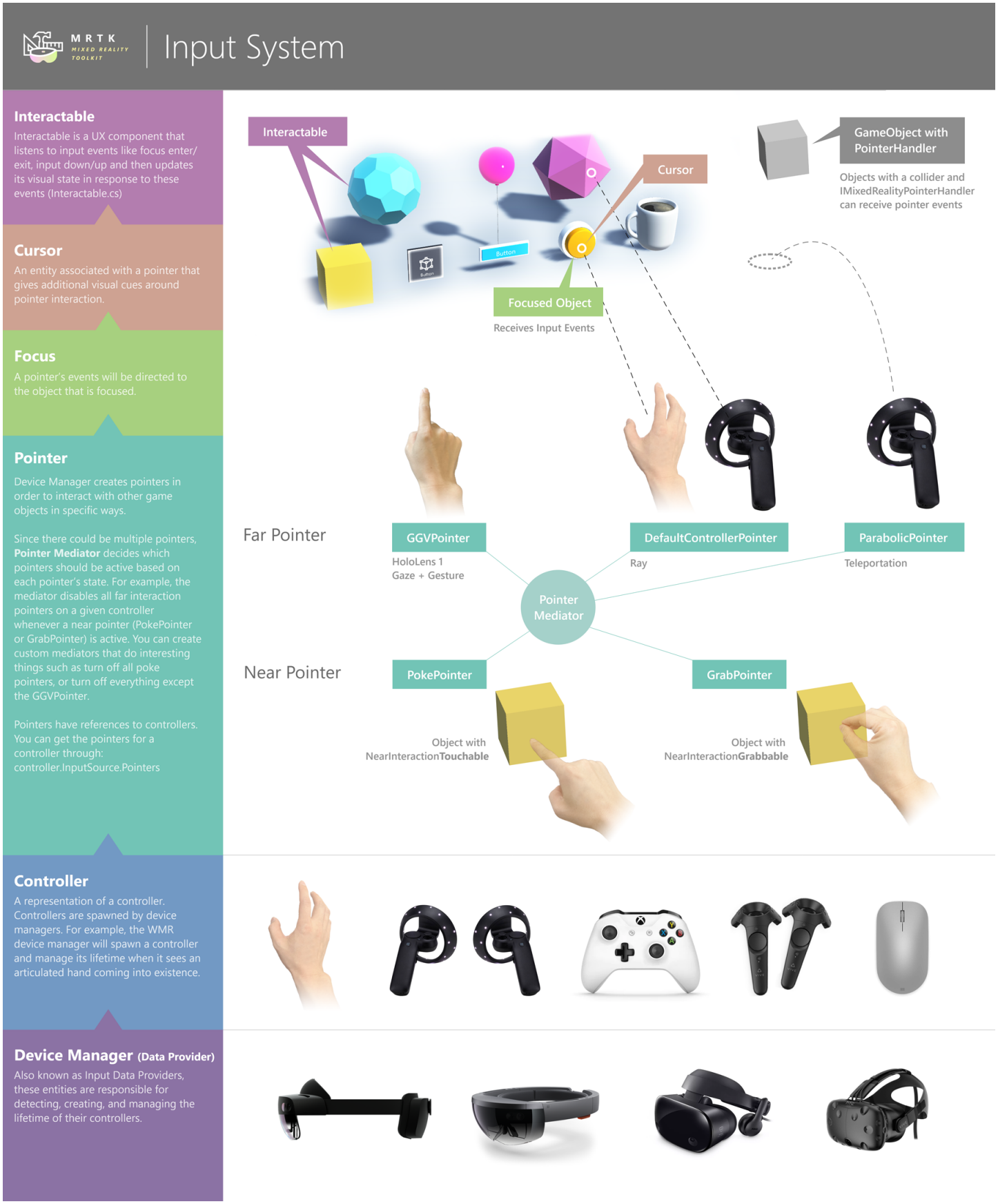

Figure 2.1. MRTK input system [22]

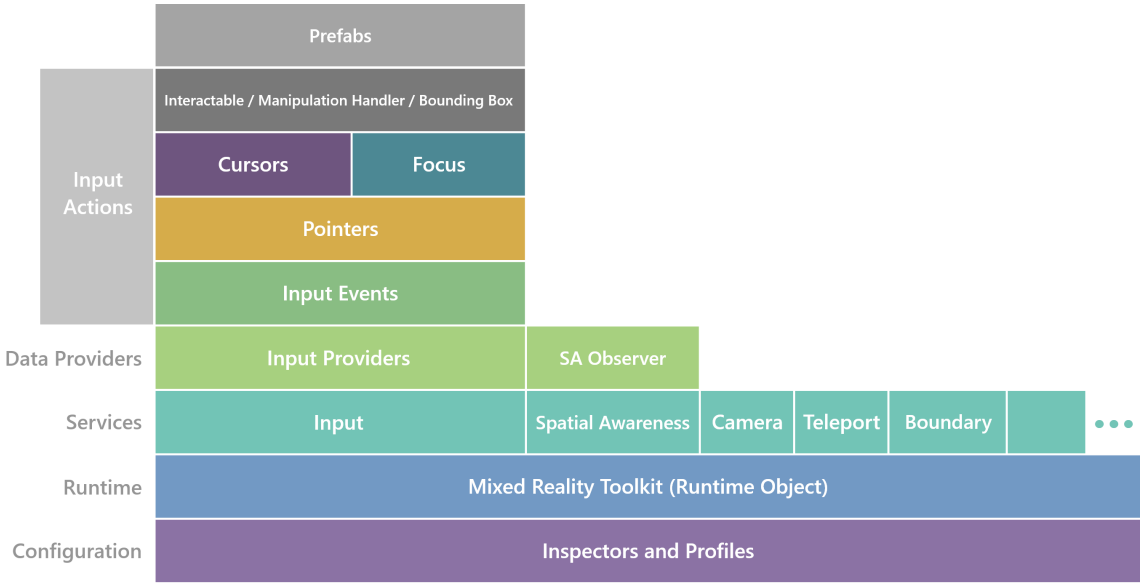

Figure 2.2. MRTK architecture [21]

2.6.1 MQTT

MQTT is a standard messaging protocol for the Internet of Things or device-to-device communication. It is lightweight and therefore great for use on low-performance devices. The protocol is an open OASIS standard and an ISO recommendation [14]. It allows the use of three quality of service levels for agreement between the sender and receiver.

MQTT client and MQTT broker have to be both parts of the communication. MQTT Brokers receive published messages and dispatch the messages to the subscribing MQTT clients. An MQTT message contains a message topic that clients subscribe. Mosquitto is one of the most widely used MQTT brokers.

MQTT protocol allows to publish and subscribe data under three different levels of quality of service (QoS). It is an agreement between the sender of the message and the message receiver that defines the guarantee of delivery for the specific message. Quality of service is the ability to provide different priorities to different applications, users, or data flows or guarantee a certain performance level to the data flow. There are three different QoS levels under MQTT:

- At most once (0)
- At least once (1)
- Exactly once (2)

It allows managing message flow thanks to a few commands. The following examples are composed of two clients A and B, and a broker, shown in figure 2.3. When client A used command Subscribe for a topic called `example`, the broker takes the command down and saves memory for client A. After this, client B can use the command publish with the same topic, which client A subscribes to, and send messages. The Publish command is executed in the broker, and the broker sends the message to client A. This example is for Quality of Service Exactly once (2). The figure below 2.4 shows the broker log for subscribing and publish sequence.

2.6.2 AMQP

AMQP is a binary application layer protocol. It provides flow-controlled and message-oriented communication with message-delivery guarantees such as at-most-once, at-

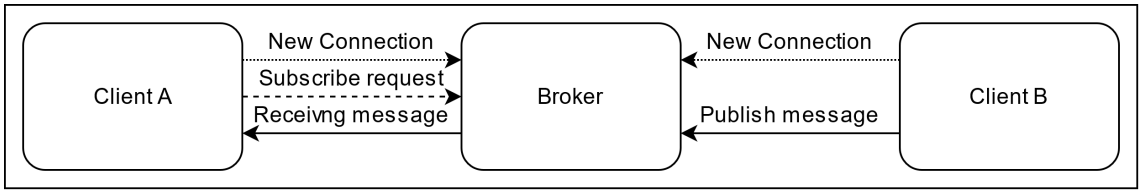

Figure 2.3. Setup for MQTT protocol example

```
1619170756: New connection from 192.168.1.162 on port 1883.
1619170756: New client connected from 192.168.1.162 as ClientA (p2, c1, k60).
1619170756: No will message specified.
1619170756: Sending CONNACK to ClientA (0, 0)
1619170759: New connection from 192.168.1.162 on port 1883.
1619170759: New client connected from 192.168.1.162 as ClientB (p2, c1, k60).
1619170759: No will message specified.
1619170759: Sending CONNACK to ClientB (0, 0)
1619170761: Received SUBSCRIBE from ClientA
1619170761:     example (QoS 2)
1619170761: ClientA 2 example
1619170761: Sending SUBACK to ClientA
1619170765: Received PUBLISH from ClientB (d0, q2, r0, m1, 'example', ... (20 bytes))
1619170765: Sending PUBREC to ClientB (m1, rc0)
1619170765: Received PUBREL from ClientB (Mid: 1)
1619170765: Sending PUBCOMP to ClientB (m1)
1619170765: Sending PUBLISH to ClientA (d0, q2, r0, m1, 'example', ... (20 bytes))
1619170765: Received PUBREC from ClientA (Mid: 1)
1619170765: Sending PUBREL to ClientA (m1)
1619170765: Received PUBCOMP from ClientA (Mid: 1, RC:0)
```

New connections established.

Client A subscribe request.

Client B publish request.

Client A receiving message.

Figure 2.4. Example of MQTT protocol

least-once, and exactly-once. Authentication and encryption are based on SASL or SSL/TSL.

Same as MQTT, it is composed of one broker and clients. The lower layer consists of publisher, exchange, queue, and consumer [15]. The use of exchange has a significant advantage against MQTT because it can determine communication threads according to use. There are a few exchange types available direct, topic, headers, and fanout.

RabbitMQ is one of the most widely used AMQP brokers.

2.6.3 Zigbee

Zigbee is a low-cost, low-power, wireless mesh network standard targeted at battery-powered devices in wireless control and monitoring applications, based on IEEE 802.15.4 specification. It operates in the industrial, scientific, and medical radio bands of 2.4 GHz. The specification includes four fundamental components network layer, application layer, Zigbee Device Objects, and manufacturer-defined application objects.

It is slower but much more low-power against WiFi. It is an excellent choice for industry or smart homes.

2.6.4 WiFi

WiFi is a family of wireless network protocols based on the IEEE 802.11 family of standards. It is commonly used for local area networking of devices and Internet access. It can achieve speeds of over 1 Gbit/s. WiFi allows only star network layout. Furthermore, it has a significantly larger range than Zigbee. It is the most widened wireless protocol used for connecting people to the internet. That is the reason why it is implemented almost everywhere, including production facilities that is the reason why it should be a good idea to use it for monitoring systems [16].

2.7 Brokers

They are programs for connecting devices and allowing communication among themselves. Both brokers mentioned below use a star topology, and the broker is situated in

the center of the star. It ensures connectivity with devices by using particular messages, and when some device goes offline, it can send a message to all connected devices.

When the device wants to subscribe to some topic and send the correct message, the broker guarantees that it will receive a message under that topic when another device publishes it. It is possible to say that command `subscribe`, sending by device, is just for the broker. The broker saves the deices credentials, and when some device publishes some new message by using the command `publish`, the broker sends it to the subscriber (saved) device.

The brokers offer additional functions stated in the given protocol.

2.7.1 Mosquitto

Mosquitto is an MQTT broker offered in open source, commercial implementations, and managed cloud services. It provides the main job of the broker, as is stated above. It allows devices use commands `Connect`, `Disconnect Publish`, `Subscribe`, and `Unsubscribe`.

Command `Connect` allows the device to connect to the Mosquitto. And obviously, the command `Unconnect` allows the device to disconnect from Mosquitto.

Mosquitto has some internal parameters to which the device can subscribe and know the information about the broker. For example, how many clients/devices are connected, how many messages were sent. This information is available under the topic `\$SYS/broker/[name of internal parameter]`

Mosquitto manages to send messages under three different quality of service (QoS) as stated above.

Because of these advantages as commonly used, simple for use, and easy to manage, it is used in this project.

2.7.2 RabbitMQ

RabbitMQ is one of the most popular open-source message brokers. It is lightweight and easy to deploy on-premises and in the cloud. It supports multiple messaging protocols such as AMQP 0-9-1, AMQP 1.0, MQTT 3.1.1, and STOMP 1.0. In the basic version, it supports just AMQP 0-9-1.

It supports many programming languages such as Python, C#, and others. However, at the time of writing, the RabbitMQ client does not support the Universal Windows Platform for Unity, the platform for HoloLens 2. It should be possible to use the MQTT protocol for HoloLens communication, but this step should cause a much more complicated program architecture. On the other hand, the implementation is written with respect to future broker change.

RabbitMQ provides a management plugin for the maintenance of RabbitMQ over a web interface.

The complete control of messages is going over exchanges and queues.

Exchanges are parts of the broker which are connected to the publisher. Queues are parts of brokers connected to the subscriber. A developer can change the binds between exchange and queue and add specific properties to them. Exchange can be of several types:

- direct - send message to the queues whose binding key exactly matches the routing key of the message
- topic - send messages to queues according to the name of the topic and uses special biding keys `*`, and `#`
- headers - send messages to queues according to the first name of the topic

- fanout - send all messages to bind queues

It provides an opportunity to divide communication to control messages and data messages simply. These types of messages are used in this project and are described below 3.2. The images below show how communication should work for both types. In figure 2.5 the Message provider is called P, and the message receivers are called C`X`. In the figure 2.6 the broker is called the Server. The images are taken over from RabbitMQ tutorials [17].

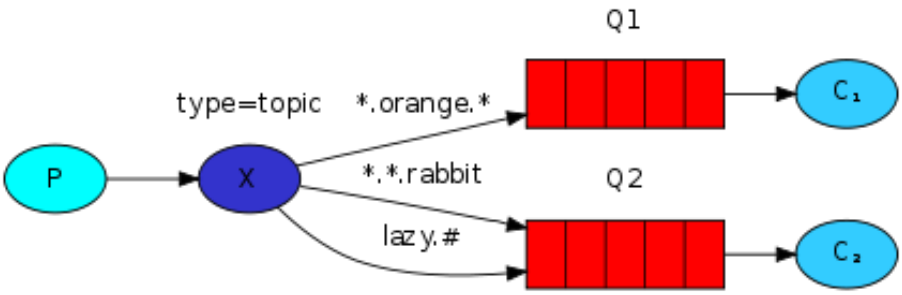

Figure 2.5. RabbitMQ example of data messages communication [18]

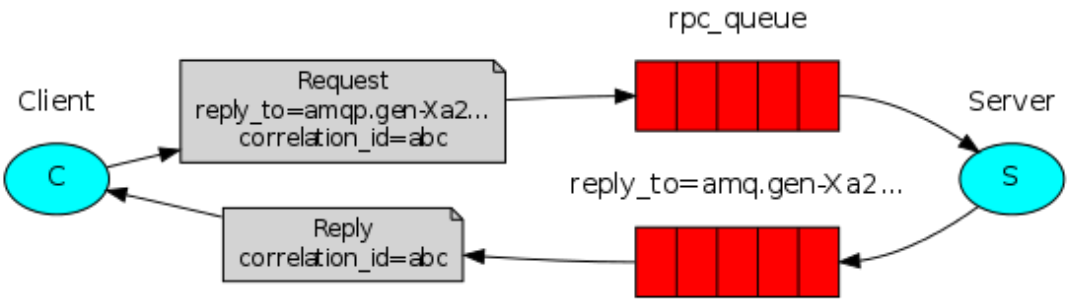

Figure 2.6. RabbitMQ example of control messages communication [19]

Chapter 3
Proposed Architecture

The main purpose of the practical part of this thesis is to create reliable and straightforward applications for industrial devices, control Server, and display devices. These applications have to communicate according to particular control messages. The communication is designed for near real-time performance. Data displaying devices can replace standard HMI and, in some situations, be more suitable for data visualization with features like showing the information on a particular location.

There are multiple types of users who can utilize created applications. The product managers have to know how many products are made in near real-time, and what caused production failure. The apps can be used exactly for that. The application can also help a service worker to maintain correctly industrial device. The application can navigate to the specific place of failure and provide maintenance steps.

The project is developed with respect to several key properties. All these properties are described in the sections below.

- Device variability
- Different platform visualization
- Easily add new devices
- Reliable communication
- Correct visualization for HoloLens
- Visualization customization by the user

First, important project properties are device variability, for example, the opportunity to use a Visualization device on HoloLens 2 and Android mobile phone, which is dependent on developer platforms described in section 4.1. With device variability is connected the opportunity to visualize data differently on the different platforms. This demand is fulfilled by the patterns described in section 4.2, which are linked for each subscribed topic.

This chapter shows proposed communication and visualization architecture. In the first part, programs for each device with a given purpose are mentioned 3.1. The next part describes how these programs communicate with each other 3.2. And the last part is dedicated to displaying device and its visualization architecture 3.5.

3.1 Components of the platform

The architecture of the whole platform consists of different components with different purposes. This part describes what individual components do, and figure 3.2 shows the individual components and connections among them.

3.1.1 Broker

It is the central node, which ensures whole communication. It is based on Mosquitto and uses MQTT for communication. For correct deployment of the whole platform,

the administrator has to install Broker on the computer TCP connected to other devices and launch it. The Broker runs in a console and from the console can be tested and debugged. The Broker allows setting multiple parameters, for example, number of connected clients, port number, SSL/TLS certificate, and many more. All these parameters can be easily set in mosquitto.conf file.

3.1.2 Server

It is the MQTT client connected to the Broker. The Server, as well as all components in the system, uses just one Broker. The Server saves information about message providers, visualization devices and stores available topics and patterns.

To fulfill industrial requirements of security the devices connect to the Server with a username and password. For security reasons, the clients can log in with the correct combination of the user name and the password exactly once production application, but this function is disabled for testing in developer mode.

The Server is written in C# utilizing official existing implementation of MQTT client written in .NetFramework. The official implementation is not the only existing MQTT client implementation. An example of such is the implementation used for Message providers, purposely chosen to show flexibility of the MQTT ecosystem.

3.1.3 Client - Message provider

Message provider is used by industrial devices, clouds, or production maintenance devices. It provides available data topics to Server and can publish data to the existing topics. For every datastream going from an industrial encoder, industrial robot, or next devices, there is exactly one topic provided. The client enables to add new topics during running and connect it with a specific input signal. For development are used mainly artificial data and topics.

Message providers support two developer platforms. Windows bound used for .NET-Framework and .NETCore, which supports different operating systems.

3.1.4 Client - Visualization

The visualization client can subscribe to some topics and display those data provided by a specific topic. One of the main ideas of the whole system is to be able to show data in chosen format on all visualization devices. The format has to be bound to a device with a particular position in space, and for each device displays a different format such as text or scope. The pattern section 4.2 provides more information about this topic.

The client is divided into three parts connected to each other showed in figure 3.1. The first part is responsible for receiving new data and sending request messages according to specific message protocol. The second part is responsible for creating a new object in a scene and updating it. And the last part handles user control.

Visualization client supports Universal Windows Platform, which is a platform for HoloLens 2. It supports development in Unity, and it is a reason why it would not be a problem to add support for other platforms, such as Android or iOS.

3.2 Low-level communication

As is stated before, the whole project contains several components which run on different devices. It is possible to describe this system in two ways. One way is to show how messages go through systems and which devices or components are passed through. The other way describes which message types are used for communication from the

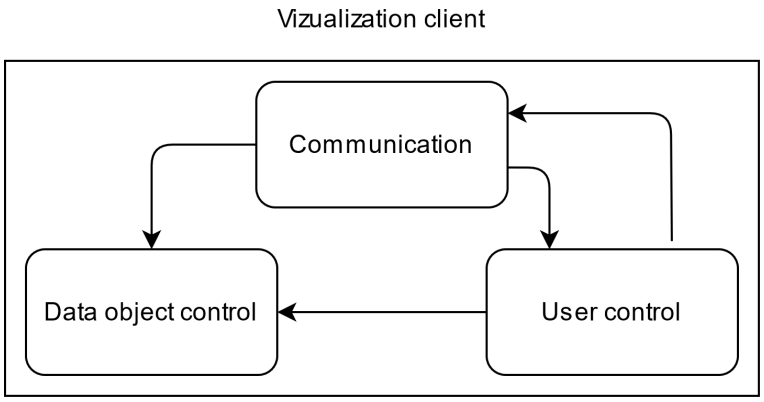

Figure 3.1. Visualization client with parts and their connections

application point of view and do not care about the real path of the message. This section describes the first of them, and other message flow explanatory images are in the sections shown below.

The core of the communication is the Broker, which can be described as a server, and other devices are just clients communicating with the Server. When components have the correct set of subscribing and publishing topics, it takes care of proper message delivery. Figure 3.2 shows a low-level architecture of the project with connected Message Providers, Visualization clients, and the Server.

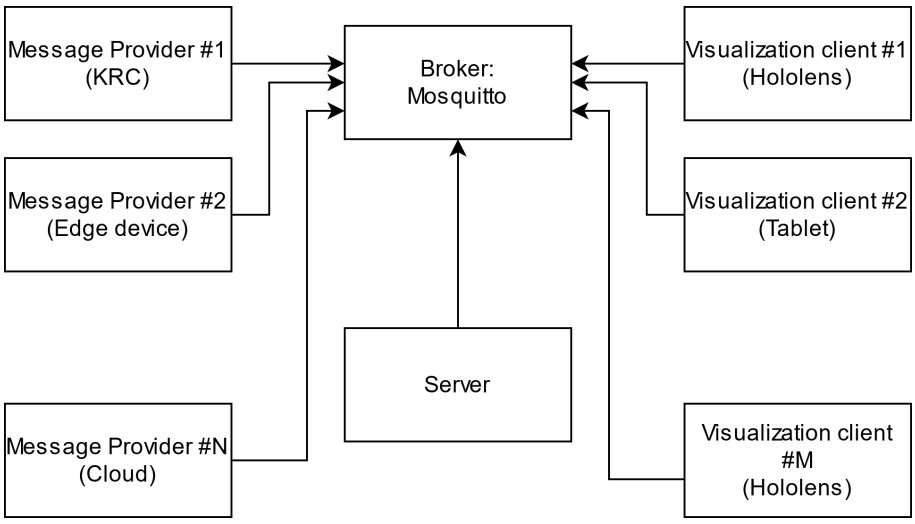

Figure 3.2. Low-level communication architecture

3.3 Control messages

The whole system uses two message types, data messages and control messages. This section describes the control plane. That means a process of connecting devices to the Server, provision available topics, and sending data messages. The state diagrams below show all described methods from the Message provider and the Visualization device point of view.

The Broker is ignored in this section for simplicity.

All control message commands are processed and executed by the communication Server. The reason for this is to minimize computation overhead on both ends, for data

stream devices as well as for Visualization devices. These control messages are used to establish a new connection to clients, to establish connection termination, and to get information about existing data streams. The current available control commands are the following:

- Connect [s, mp, mc]
- Disconnect [s, mp, mc]
- Subscribe [s, mc]
- Unsubscribe [s, mc]
- Available Topics [s, mc]
- Pattern Demand [s, mc]
- Provide Available Topics [s, mp]

In the brackets are stated devices that can use this command.

- [s] - The Server
- [mp] - The Message provider
- [mc] - The Message consumer - visualization device

Because of the use of the MQTT protocol, the control communication is going on the same communication channel as data. That is the reason for creating special topics just for this type of communication. The Server subscribes to the topic `/Manage` for receiving control messages from clients. The clients are configured to send all control commands to this topic. The clients are creating their topics for receiving control messages from the Server. The topics have a unique structure `/Manage/` + Device name of the client. All these control topics are created when the program is starting.

The demonstration of one control message in low-level architecture is showed in the figure below 3.4.

How the whole process of new subscribing works is showed in the figure 3.6 at the initial state, the client does not know any topic to subscribe and any pattern.

3.3.1 Connect command

A client has two different connect commands. All clients and the Server has the first of them. This command serves for connection to the Broker and uses only the name of the device. The second one allows the clients to connect to the Server.

The only indicator for a connected device is Client ID. If the Client ID is equal to zero, the client is not connected to the Server. When a user uses the command Connect, the unconnected client sends the request to the Server with the login name and password stored in the Notes attribute. The Server compares the login information in the received message with the saved login information. When it is correct, the server checks if there is a connected device with the same login information. In case it is not, the Server chooses the first available client ID that replaces the zero client ID and sends it back to the client with a response that confirms the connection. Finally, if the client has not already connected, the Server will add this client to the Server's connected devices database. The whole process of establishing a new connection is shown in figure 3.5. The check of unique login information is turned off in developer mode.

When the server evaluation process detects an error, the Server sends the client's request back with zero client ID and error information placed in the response attribute.

3.3.2 Disconnect command

The disconnect command has two variants too. The first one is for the devices which want to disconnect from the Broker. And the second one serves the clients to disconnect from the Server.

The client's disconnection works quite simply. The server checks if the device is connected and sends the request back with zero client ID and confirmation message. The Server then deletes the client from the connected devices database. In a case when the Server detects some error, it works the same as the connect command. It just places the error message into the response attribute and sent it back to the client.

3.3.3 Provide all available topics command

It is a control message for a Message provider that has stored and published some data, but no Visualization device can subscribe to them because they do not know the topics. This command provides all available topics with number of visualization types (pattern 4.2) from the message provider to the Server. If the Server knows the attached patterns, it stores the topics with correct patterns to the Server's available topics database.

3.3.4 Available topics command

This control command provides the information about available topics for the Visualization device and stores the topics in its topics database. This information contains the name of topics and the number of visualization types saved in the Server's available topics database. This command allows the Visualization device to show all available topics for subscription.

Visualization of the topics move is shown in the figure 3.3.

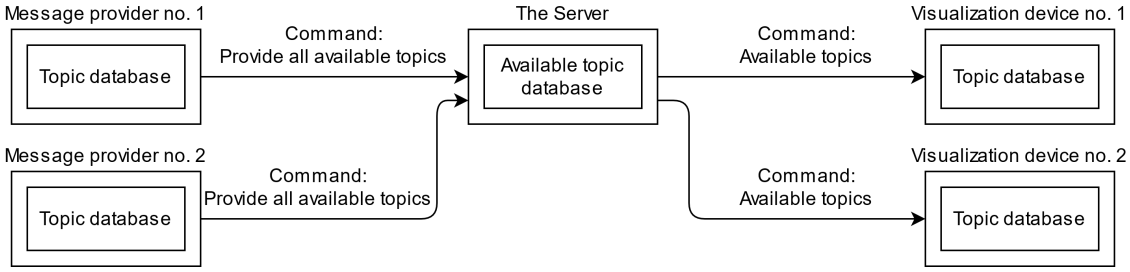

Figure 3.3. How the topics are stored and distributed

3.3.5 Pattern demand command

Each topic is visualized according to a defined pattern, which can be varied for different devices and data types. Exhaustive pattern explanation is stated in section 4.2. This request is used only when the visualization client does not know the pattern for a topic that it wants to subscribe.

3.3.6 Subscribe command

It is the essential command for the whole system. It ensures several important functions, for example creating visualization components described in section 3.5 and calling the pattern demand command.

When the user wants to show some information, they have to choose a topic for subscribing from the visualization's topics database, the client checks if it has the correct pattern. In a case when the client does not have this pattern, it sends a pattern

demand command. After this check, the client sends subscribe command, and the server checks if the topic is still available and sends back the result. If the result is positive, the client creates a visualization component according to the pattern.

It uses the subscribe broker command, which sends the information about the client's subscription, and the client then receives all data published under this topic.

The whole process of the new subscription described above is shown in figure 3.6.

3.3.7 Unsubscribe command

It is a straightforward command which sends the topic for unsubscribing. The client uses a unsubscribe order for the Broker. It is an informative command which is not used, but it could find some utilization in the future.

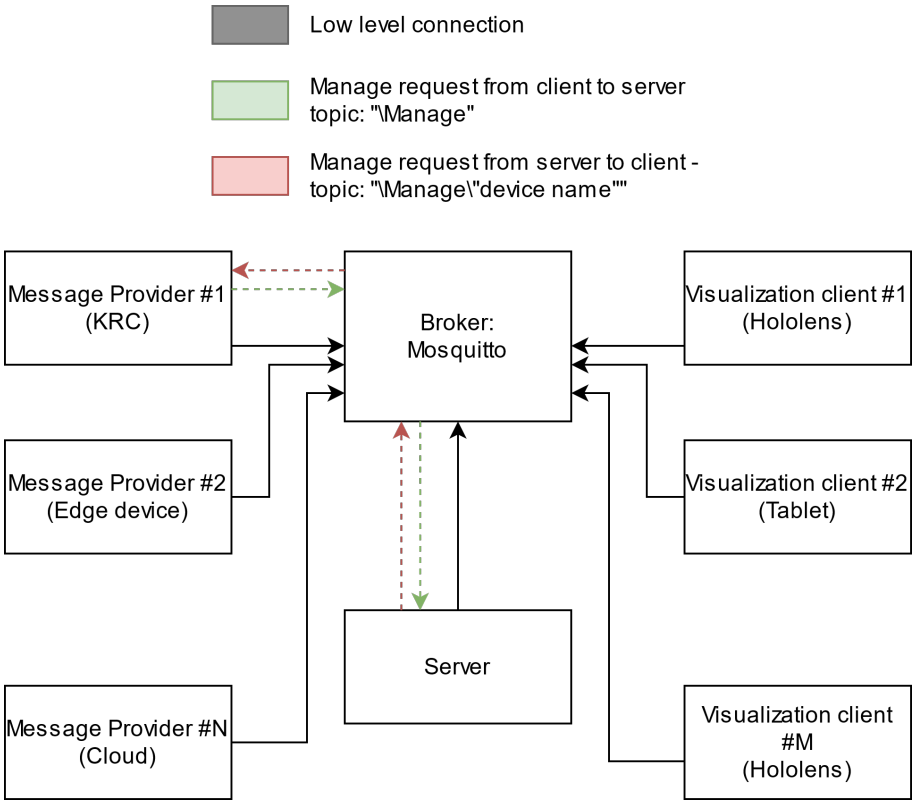

Figure 3.4. One control message in low-level communication architecture

3.4 Data message types

The whole system includes two different types of messages. The first type is used for control messages. Its description is stated in the section 3.3. The second message type is data message, which has several options for how it can be delivered.

Delivery methods with demonstrative images are described in following subsections.

3.4.1 P-B-V deliver

The message goes from the Message provider via the Broker to the Visualization device. In the current version, the delivery option is supported. From an implementation point of view, it is more accessible, and the real data journey is shorter, which means delivery

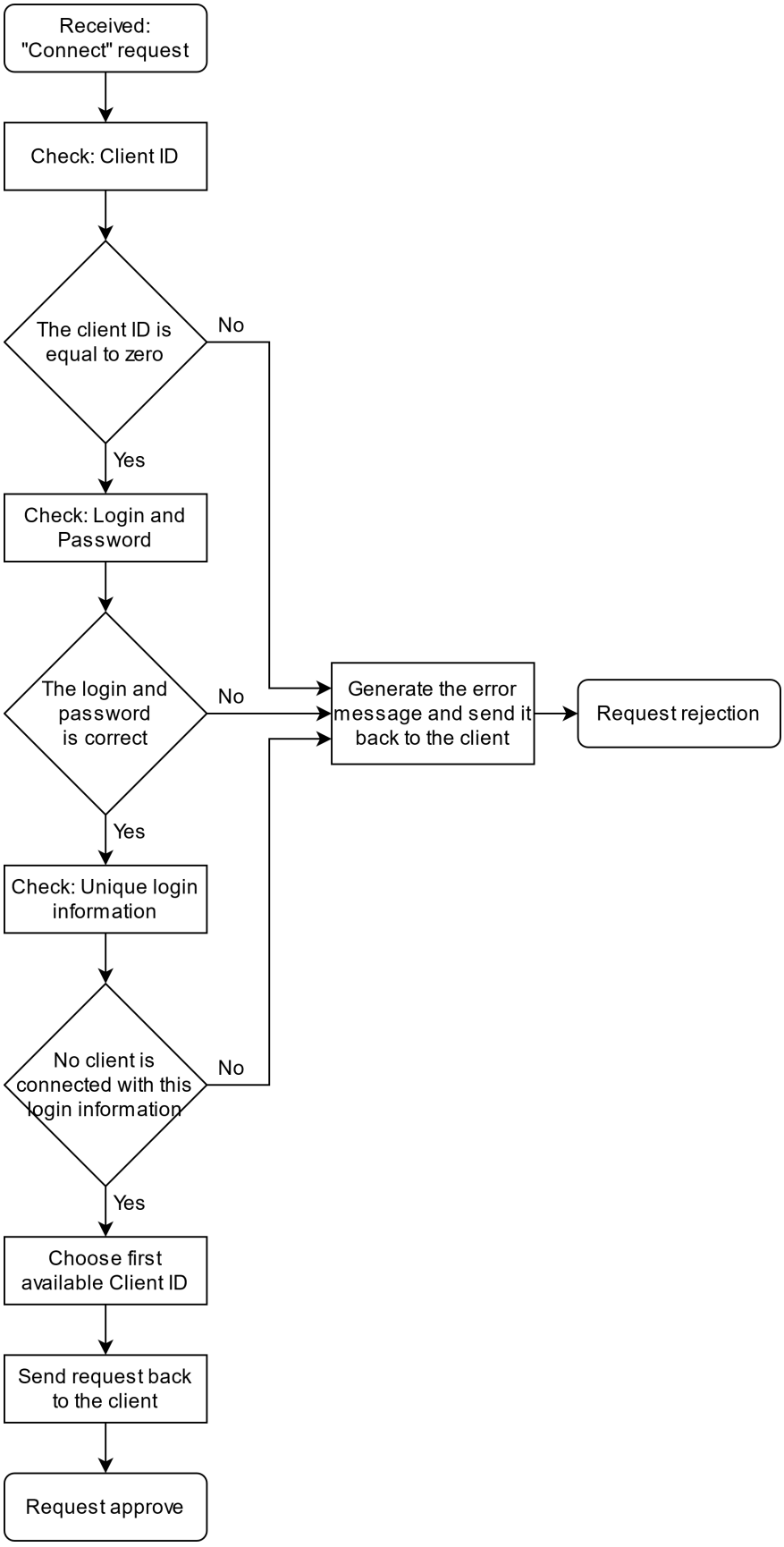

Figure 3.5. State diagram of the server's condition for new connection

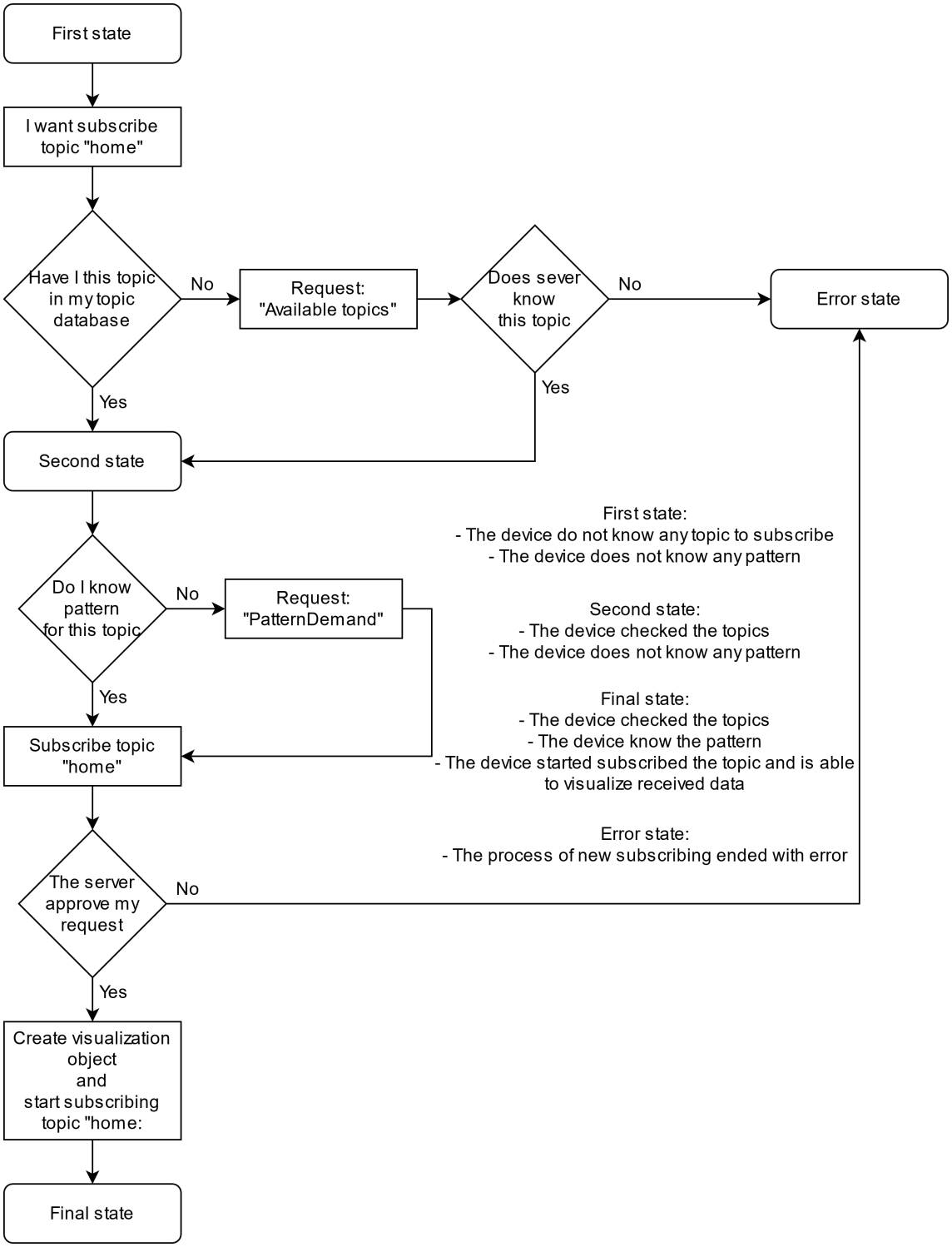

Figure 3.6. One control message in low-level communication architecture

time is faster, and the system has a better echo when it is compared with the second delivery method. On the other hand, there is no opportunity to process and transform data for the required format, and the Message provider has to do it.

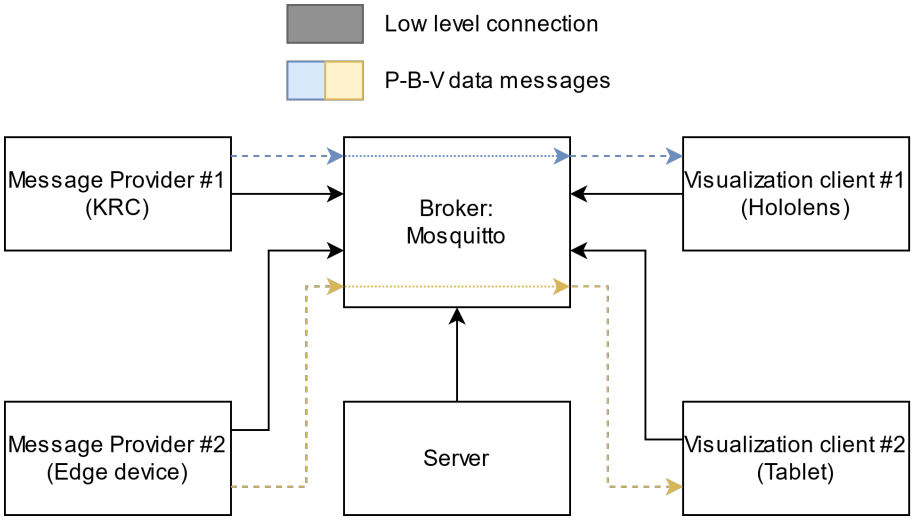

Figure 3.7. P-B-V data message delivery type

3.4.2 P-B-S-B-V deliver

The message goes from the Message provider to the Broker after that to the Server. The server process the message. Then it goes back to the Broker and to the Visualization device. It is a more complicated journey, but it can provide some benefits as it moves the exhaustive calculation operations to the Server. This is in many cases necessary because industrial devices such as PLCs, Robots, or HoloLens are lightweight.

This type of message delivery is great for graphs and figures, which are computation expensive.

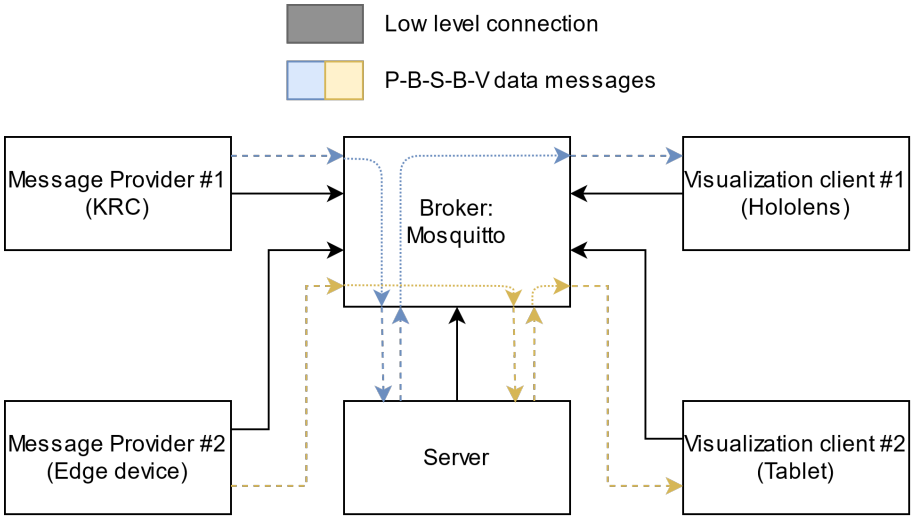

Figure 3.8. P-B-S-B-V data message delivery type

3.5 Architecture - visualization

The current project version has implementation only for HoloLens visualization devices. This device type for visualization uses its methods for creating physical objects, interaction, and control scripts. These methods are saved in the MRTK library. The methods for data visualization used in this project are saved in the Object visualization component.

The process of subscription to new topics was described before in section 3.3, but the section describes only the process of establishing the new subscriptions. After confirmation of the new subscribed topic, the HoloLens creates a new object for the visualization data and adds this object to the subscribed topics database. When the prefab of a visualization object is created, it is necessary to add the parts of the object according to its pattern 3.9. The functions 2.5.1 are used for the object creation.

The last important method of visualization part is the print message method, which is called for specific visualization objects. This method has to evaluate the incoming data and ensure correct visualization. The process of evaluating incoming data is described in the section below 4.4.

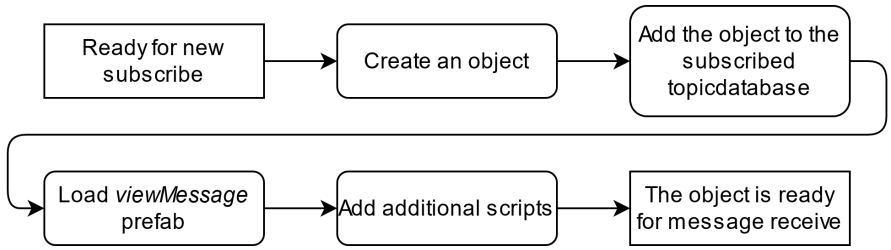

Figure 3.9. Process of new visuazation object creation

3.6 Recognition

The recognition serves to determine the position of a visualization object. There are two kinds of recognition. The first one ensures the position of an object itself. Among this type of recognition belongs image recognition and model target. The second type of recognition determines the correct origin of coordinates. According to this initial position, the Message providers can send only X, Y, Z coordinates for the correct position of an object. There are two examples of this recognition type, area target and default HoloLens position.

Table 3.1 shows how the message visualization object is attached to the real object using different detection types.

Detection types	Type of attachment
HoloLens position	coordinates
Image recognition	Image ID
Model target	Model ID
Area target	Objects ID and coordinates

Table 3.1. How real object attached to the visualization object

The HoloLens creates its 3D map of surroundings and knows the position in word space, but the origin of coordinates is dependent on the position where the program starts running. However, this position is changing, and it is tough and uncomfortable to ensure it every time. There are several ways for object detection, and the developer does not have to care about the user's position in space.

This project uses Vuforia for all recognition. With some limitations, the Vuforia offers all these recognition types available for free. The reason for Vuforia is based on information stated in bachelor thesis [28] which provides deep analysis of AR frameworks for augmented reality and the scientific papers which use Vuforia [30] and [31]. For non-commercial use, Vuforia offers advanced recognition features for free. When the Vuforia

is compared with the AR Kit or AR Core, only the Vuforia supports UWP, which is necessary for the project. There is a table 3.2 of AR frameworks that compare their available features, and the Vuforia is the best one.

Feature	ARKit	ARCore	Vuforia
Image recognition	+	+	+
3D recognition	+	+	+
Plane detection	+	+	+
Area target	-	-	+
Android support	-	+	+
UWP support	-	-	+
Unity	-	+	+
Basic free	+	+	+

Table 3.2. AR framework function comparison

3.6.1 Image recognition

It is the most common way how to detect an object. It works very easily and reliably. Nevertheless, each detected device has to have its image label. The labels can be several types, but the most effective is QR code, which is easy to create. This recognition type is quite uncomfortable because, for detection, it is necessary to label each device. The evaluation section shows which QR codes are good for detection 5.1.

3.6.2 Model target

It enables to detection of objects according to their CAD model. It does not require any additional labels, but it is much more computational demanding than image recognition.

Each object has to be detected according to its default rotation which is called guide view. The guide view disappears when the Vuforia finds the object. An example of such model target detection is shown in figure 3.10.

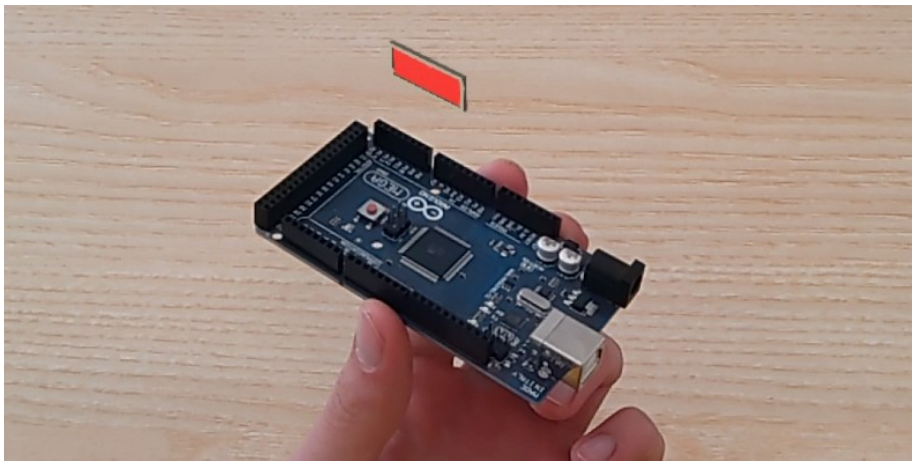

Figure 3.10. Model Target detection

3.6.3 Area target

Area target can find an environment scanned before as a scanner is possible to use an iPad with LiDAR. After successful detection, the Message provider can attach a visualization object to the position according to the room's origin coordinates. Alternatively,

there is another opportunity how to ensure the correct object position. For example, it is possible to create points in the room with a specific name, and the visualization object attaches to these points.

For correct detection of a room, it is necessary to stand in place used for scanning. Outside of these places is not very good quality of detection.

Chapter 4
Implementation

This chapter describes an implementation of the proposed architecture. First are described developer platforms used for individual components of this project, then is described implementation of the patterns. The chapter continues with an essential description of the communication message design. Furthermore, the following section of this chapter mentions the vital properties of the Visualization device.

4.1 Platforms

Overview of the used platform is stated in section 2.4. This section describes why each component of the system uses a different platform and the benefits of this implementation. The used platform is connected with an operating system that the individual component uses. The individual components with specific platforms and with the supported operating system are stated below.

- Server - .NET Framework - Windows
- Message Provider - .Net Framework, .NETCore - Windows, Linux, iOS
- Visualisation - UWP - Windows

The Message Provider is written in two different platforms for more extensive variability of supported devices the .NETCore implementation enables the use of it on the Revolution Pi with Linux.

The project's goal was to support other communication protocols such as AMQP through RabbitMQ as well, but unfortunately, to this date, there is no working AMQP client for UWP. The table below shows all tested AMQP clients with supported platforms 4.1. The last one, the Holorabbit library is a thin C wrapper around RabbitMQ meant to provide simple functions that can be accessed via a C# DLL Import command in Unity [25]. Because it was created for the first version of Hololens, it could be the reason it is not working on Hololens 2.

Name of the client library	Supported platform
RabbitMQ Client - C# [23]	.NETCore
Unity3D.Amqp - Unity [24]	Windows 10, macOS, Android, iOS
Holorabbit - UWP [25]	Windows 10, NO UWP

Table 4.1. Tested AMQP clients

The server and Message provider as well are console applications. Furthermore, after finishing the development process, they can be rewritten, for example, to the Windows Form application. However, for the developers, console applications are better because of their focus on functionality. The Visualization device is a UWP application, and it was tested only on HoloLens.

4.2 Patterns

Patterns are descriptions describing how to visualize incoming data. Patterns allow many types, such as plain texts, scopes, images, and more. Thanks to patterns, it is also possible to set default position and other features, for example, always turning a text to the user, ability to scale and move the text, and so on. According to the correct pattern, the evaluation of the message is described in section 4.4.

This type of solution offers an ability to visualize one message differently on diverse device types. This feature is in the current version limited because the data messages are sent directly to the visualization device with the determined pattern. Using the P-B-S-B-V delivery type, the pattern sent from the Message provider will work just as a recommendation, and the Server can assign the pattern to the topic according to subscribed device type.

Implementation of patterns adds complexity to the solution but has multiple advantages.

- Different types of visualization for each message
- Opportunity to visualize differently on different devices
- Simple setting the visualization type

All pattern content is saved only in the Server's database. Thanks to this, the administrator can quickly add new patterns when needed.

The code is defined as a particular class with some attributes that determine the visualization behavior. When the client's program starts running, it does not know any pattern. When the client wants to subscribe to a topic and does not know the pattern content, the Pattern demand command provides the pattern for the client. The pattern attributes are stated and described below.

- Pattern ID
- Type view
- Default position
- Description

4.2.1 Pattern ID

Pattern ID is a unique identification number for each pattern. Creating patterns with the same pattern ID is forbidden by the Server. This unique number is assigned to each topic and sent control with commands: Available Topics and Provide Available Topics to declare which pattern is used for each topic.

4.2.2 Type view

Type view is the string attribute that declares the main properties of the pattern. In the string can be saved lots of commands for visualization purposes. Semicolons separate the individual commands.

The structure of the string is specified. In first place is the name of the visualization style, for example, `text`, or `scope`. In second place is defined the position in space, for example, `fixed`, `free`, or `image`. The next places of type view string are placed additional properties and script's demands, for example, `headRotate`. More information about the visualization parameters can be found in section 4.4.

4.2.3 Default position

This pattern's attribute serves only for the `fixed` command in the type view attribute. In other cases, the visualization client omits this attribute. It could seem useless compared with the `free` command that declares each message's position in space. However, this attribute with the `fixed` command ensures the simplest way to send a message because it contains only the data without any additive control information.

Three numbers in meters separated by the semicolon are components of the default position attribute.

4.2.4 Description

For each pattern, a message provider has to send a message with additional information. The message often contains the information data and the control data for visualization, for example, the position in space and image ID. This attribute provides some additional information about the pattern and how to send a message for correct evaluation. In other words, the description says what component the data message must contain to be correctly evaluated.

4.2.5 Pattern examples

Pattern ID separates individual patterns, but the main distinguished property is the type view attribute. Their examples are stated below and described in section 4.4.

- `text;fixed`
- `text;free`
- `text;image`
- `text;image;headRotate`
- `text;model`
- `text;area`

Figure 4.1 shows the difference between the head rotate script and the opportunity the visualization object rotates by a user.

4.2.6 Patterns visualization

Patterns are used for correct visualization object creation and specific print messages. In this project, a prefab is used for the creation of the new visualization object, but some patterns have specific demands for additional scripts. For example, a script that enables an object to follow a specific image and a script ensuring rotation of an object to the user's head.

When a visualization object is created, the constructor copies complete pattern information, especially the type view properties, to the new visualization object properties. Among them belong position, type view, number of used pattern, subscribed topics, and the game object itself.

4.3 Send request component

It is an essential part that helps to send control messages among devices. The clients store data for communication about themselves in this component. It contains several attributes which are sent from a client to the Server. Thanks to the particular function program can change these attributes according to demanding requests. The send request class attributes are as follows.

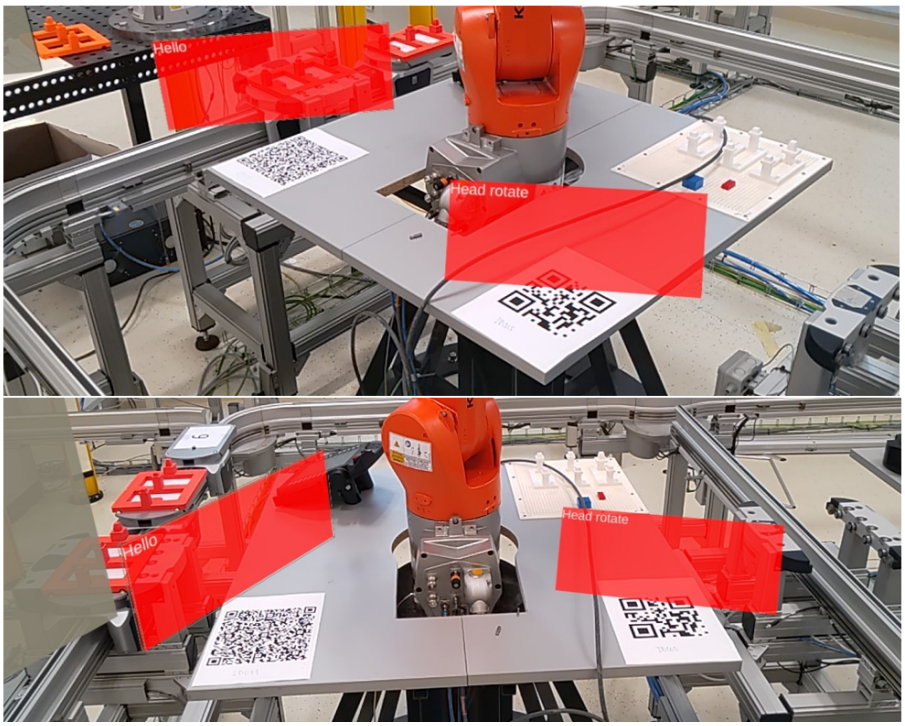

Figure 4.1. Use of head rotate script

- Device Name
- Client ID
- Request
- Notes
- Response

The clients define their device name according to their type with a random number. For example, `MessageProv-5852`, `HoloLens-7895` or `Tablet-2589`. Each client has an initial client ID equal to zero after connection to the Server. The Server assigns the unique client ID to this device. The request attribute is filled with one of the control requests stated in section 3.3. For requests that require additional information is served attribute Notes. For example, the request `Subscribe` requires the topic for subscribing.

The last attribute is called Response. It serves as a response from the Server to a client. When the Server denies the request, it sends the reason for rejection and changes the Notes parameter to zero, which caused the client's correct understanding.

It is essential to mention that the Server has its Send request part, but it only serves for the connection to the Broker. A client requests are just evaluated not stored there, and are sent back to the client. More information is available in the section 3.3.

4.4 Message Evaluation

Control and data messages are evaluated differently. A control request sends the client to the Server, which evaluates the request and sends the result back to the client. On the other hand, data messages are evaluated only by the Visualization device, and it uses the print message method.

4.4.1 Data evaluate

Patterns information has an impact on data message evaluation. However, it is not the pattern information precisely because they are copied to the visualization object component when the object is created. That is a reason why the change of pattern does not impact the new message evaluation. It is important to note that the changing pattern during the run is not a usual process. If the administrators need to adjust a pattern, they can create a new pattern.

When the Visualization device receives a new message, the evaluation process determines the type of message. It uses two data types, the control messages, and the data messages. Control message evaluation process is described in the section 3.3. However, it is straightforward. When the received message has the topic `/Manage/` + client device name, it is a control message. Other messages are label as data messages.

The first important evaluation step is to find the correct visualization object according to the message topic. This visualization object is created when the Server confirms the subscription of a specific topic. Nevertheless, the user can set if the visualization object is created with the Server's confirmation or if the client receives the first data by the topic. In the second case, the evaluation process has to create the physical object. When all conditions are fulfilled, the evaluation process calls for a specific object visualization print method.

4.4.2 Print message

The print method evaluates the type of visualization according to the patterns type view saved in the object. The type is evaluated according to their order. The first one is a type of message, for example, `text`, `figure`, and `image`. The second type is crucial for evaluation. It determines the position of the visualization object and the following properties. For the position, there are some examples: `fixed`, `free`, and `image`. Each of these position types has a different incoming data structure. For example, the `fixed` type is the easiest way how to visualize data. This type does not require any information and prints the whole incoming message.

Next position type is `free` which require this incoming data format: `data;posX;posY;posZ`. It prints only the `data` part, and other information is used for position determination. The position is received as an integer because it is much easier than float type send. However, Unity uses the metric system with the meter as a default unit. All position information is divided by one thousand for good precision (mm precision).

The last position type is `image` which ensures the image position to the visualization object. Vuforia determines the position of the image. This type requires following incoming data format: `data;imageName`. Same as the previous type, this type shows only the `data` part. The evaluation process tries to find the Vuforia image according to the given name. When it is found, the position of the visualization object is attached to the image with the specified offset. This solution enables changing the attached image for the specific message and raising the whole project's variability.

Among other position types belongs `model` and `area` which are described in the section 3.6. And there are next parameters as `Headrotate`, which are not important for data evaluation but for object creation at this moment.

4.5 Console implementation

As is mentioned before, the Server and a Message provider are console applications. These applications are elementary for debugging because the developer can let the

program write any variable. A similar debugging opportunity offers Unity, but the Unity debug console is available only in simulation mode. However, program simulation can behave differently than on the actual device, in this project, on HoloLens.

The developer's library for HoloLens does not offer any console for debugging. So the only way how to debug a program on HoloLens is to create it.

The created console 4.2 is composed of text object with background. The specific script ensures copying log messages to the text object. This method allows writing thirty log messages to the text object. It is possible to raise the number of message logs. However, a large number of log messages require bigger computational power.

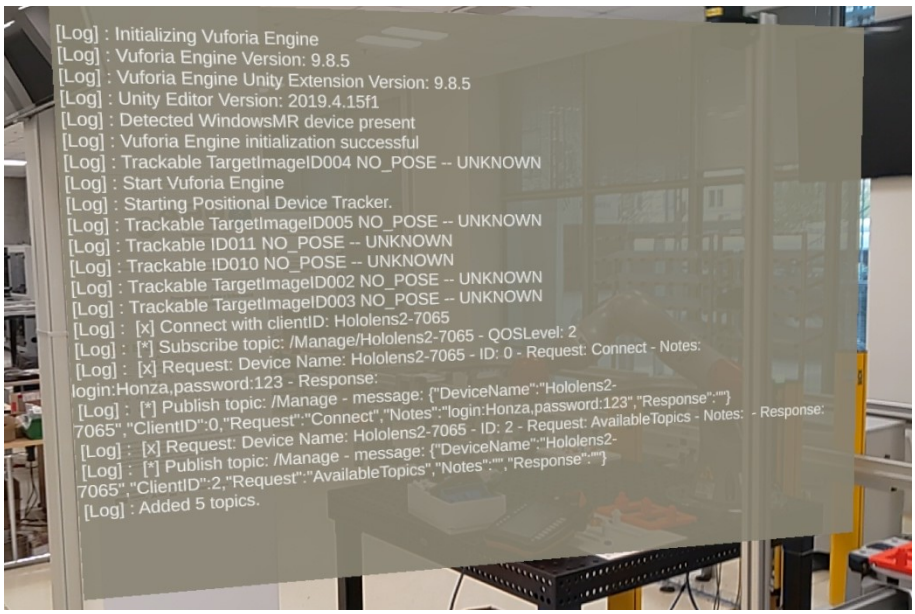

Figure 4.2. Visualization device console implementation

4.6 Menu

The control of the Visualization device is a big difference compared with the console applications. The console applications are controlled thanks to writing commands to the console window. For the Visualization device creating of the buttons is the easiest way to control the application.

The control buttons are divided into three groups. The first one ensures broker control. The second one is the most important for the correct functionality of the whole project, ensuring Server control. The last one is to print valuable data from the Visualization device's databases, the available topics, the subscribed topics, and the patterns that the client knows.

4.6.1 Broker control

This section of control buttons provides the opportunity to connect and disconnect the Broker, publish default messages, and subscribe to a topic. Although the two last control buttons are pretty obsolete, they were helpful for communication establishment. The last button in this section ensures the clearing of the console.

4.6.2 Server control

That is the essential buttons section. It offers an opportunity to connect and disconnect the Server. These two command uses the communication control request described 3.3. The next button is called `Subscribe Graphic` and when the user clicks on this button, the new buttons are created for each available and not subscribed topics 4.3. For subscribing, the user has to click on one of these topics, and the Visualization device sends to correct request to the Server.

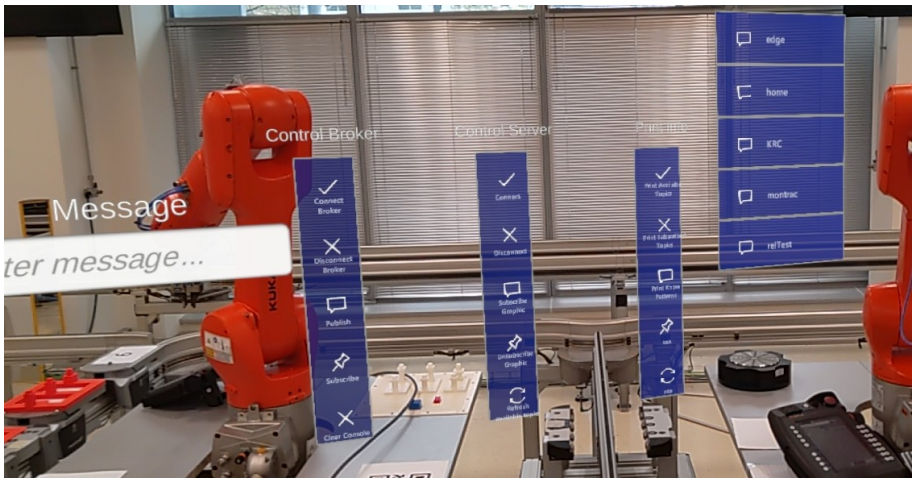

Figure 4.3. Visualization device, menu for new subscribe

The next button is called `Unsubscribe Graphic` works similarly, but it just shows subscribed topics after a click. After the user's choice, the Visualization device sends the `Unsubscribe` control request to the Server, and it stops the topic subscribe.

The last button is called `Refresh available topics` and it calls `Available topics` control request to the Server and saves all new available topics to the database. According to this database, the user can choose the desired topic.

4.7 Multi-model target

When a developer wants to detect an object without additional labels, he/she has to use a different detection feature. The best way for it is to use the model target feature. It offers to detect only one object, or it offers multi-model target detection. Both of these options have advantages and disadvantages. Both types of model target detection are saved in the dataset. In one moment, only one dataset can be activated.

The detection of a single model target is much easier thanks to the guide view, but it can detect only one object. The multi-model target detection offers detection of two and more objects, but Vuforia recommended a maximum of ten to twenty objects for detection in one dataset [27]. Moreover, multi-model targets require much more computing power, and it is much harder to detect the object than with the single model target.

The model target Vuforia features are not integrated into the main program at this moment. Nevertheless, it was developed a special switch for individual model target datasets, which enable switching among single model target datasets. First, the position of the detected object is saved, then a user is able to find individual model targets and use the multi-model target only for moving objects.

Chapter 5
Evaluation

5.1 Recognition evaluation

For the Visualization device, the determination of position is one of the most important parts. The device provides the position in a space according to the place where a program starts. The device itself does not enable detect specific objects in space. Better methods for determination of position and for object detection were mentioned in section 3.6 before.

During the testing phase, all available Vuforia recognition features were tried. The final application uses one of them. The final application uses image recognition. QR codes are used as images. The codes are generated only for the project purpose. It is interesting to see which codes are detected correctly and which are problematic for detection.

5.1.1 QR code compare

Vuforia image recognition is simple to use. The first step is to add an image for detection to the developer's database on the Vuforia developer portal. The image has several attributes. The first attribute is a type of target, except the single image type Vuforia for detection offers cuboid, cylinder, and 3D object types. The next and most important attribute is the width of the target image in scene units. The last attribute is the name of the target image. The second step is downloading the target's database and add it to Unity.

Three types of QR codes are used for the test. The images are shown in figure 5.1. Each image has a different target rating given by Vuforia, which is dependent, for example, on contrast, organic shapes, and repetitive patterns. These image properties are used for creating detecting feature points. The rating depends on the number of the feature points [26]. The image rating determines the expected quality of detection.

Figure 5.1. QR codes for detection with different rating

This test tries to verify correct image detection and proper image coordination. From the figure 5.2 the excellent quality of the detection can be seen. The `B` is the image coordination base, and the `X`, `Y`, `Z` are individual coordination axis. Nevertheless, images with a bad rating have a different origin of image coordination. The origin is moved down in the `Y` axis. The change can be better seen in the real HoloLens view.

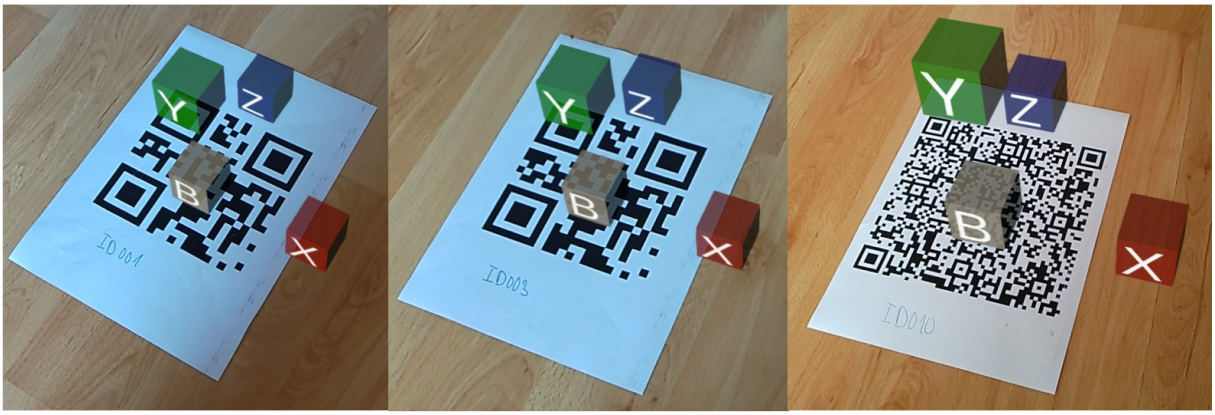

Figure 5.2. QR code target origin of coordination detection

Developers can avoid this inaccurate image origin detection by only using the targets with the best rating.

5.2 Permeability tests

These tests aim to find how many data messages per second the Visualization device can receive correctly. This receiving frequency is essential to be able to estimate the sending frequency of data messages by the Message providers.

The message permeability is tested for various messages delay and different sizes of messages. In section 2.6 before is mentioned the opportunity to choose one of the quality of services for MQTT protocol. Each subscribed topic has one of the available QoS, and published message also has one of QoS. All these tests were executed for each quality of services set for a subscribed topic. The quality of services of the published messages was set on the `Exactly once` for all tests.

The tests were executed in a test environment. All setup is shown in figure 5.3. Like the router, home router `TP-LINK Archer C20` was used, disconnected from the internet. The Raspberry Pi was used as the Visualization device. Distances between individual components were one up to two meters.

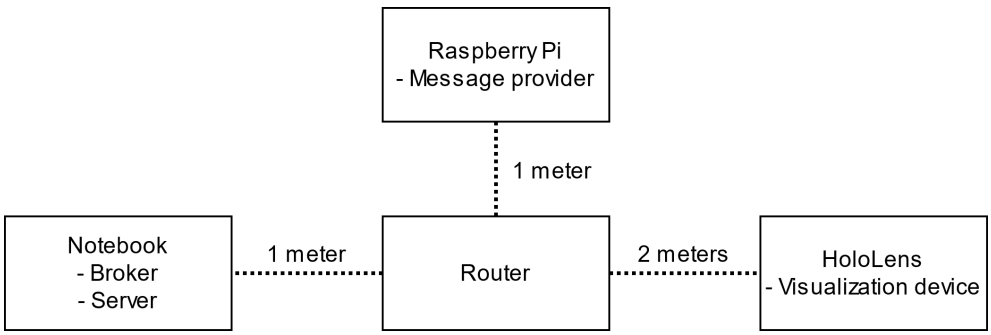

Figure 5.3. Permeability tests hardware setup

For the tests, scripts were developed, and a special pattern was created. When the Visualization device starts subscribing topic for tests, the tester writes how many bytes the Message provider will send in one message. The Message provider starts sending data messages with required delay between messages. It sends one thousand messages with one delay, and the Visualization device counts how many messages it received with which delay. The number of received messages for each message delay [ms], and for each message size [bytes] is shown in tables 5.1, 5.2, and 5.3. Each table shows the same measurement for different QoS of subscribed topic.

The message contains a little information, message ID, and information about a delay between two messages. The information about delay serves the Visualization device for correct counting the messages.

Size of messages [bytes]	10	100	10000	50000
200 ms	998	952	975	956
100 ms	958	933	888	879
75 ms	797	784	847	820
50 ms	502	543	746	748
25 ms	268	266	595	625
20 ms	204	229	644	694
15 ms	161	182	596	617
10 ms	96	116	511	472
5 ms	46	57	130	347

Table 5.1. Permeability test for at most once QoS

Size of messages [bytes]	10	100	10000	50000
200 ms	853	856	849	873
100 ms	891	903	886	923
75 ms	836	887	867	861
50 ms	793	820	830	808
25 ms	993	589	685	640
20 ms	657	590	581	771
15 ms	539	665	705	634
10 ms	527	418	407	558
5 ms	306	303	317	335

Table 5.2. Permeability test for at least once QoS

The tables show record data, but the figures provide a better data visualization for each QoS 5.4, 5.5, and 5.6. Figure 5.7 compare average values for individual quality of service.

For delay 200 ms, the quality of delivery is 75 % and higher. And for delay 100 ms, the quality of delivery is 85 % and higher, that is the reason why delay 100 ms was chosen for great delay gap between data messages sent by the Message provider, which means 10 Hz sending frequency. The last figure 5.7 shows that quality of service is not only a parameter that influences the quality of message delivery, and any restrictions cause the loss of data messages. It can be, for example, a small buffer of the Broker or a wrong router setting caused by the default setting.

Size of messages [bytes]	10	100	10000	50000
200 ms	822	766	851	858
100 ms	893	908	910	870
75 ms	841	871	864	814
50 ms	867	867	856	730
25 ms	534	516	593	722
20 ms	577	456	409	574
15 ms	601	554	483	529
10 ms	408	459	512	528
5 ms	309	298	275	64

Table 5.3. Permeability test for exactly once QoS

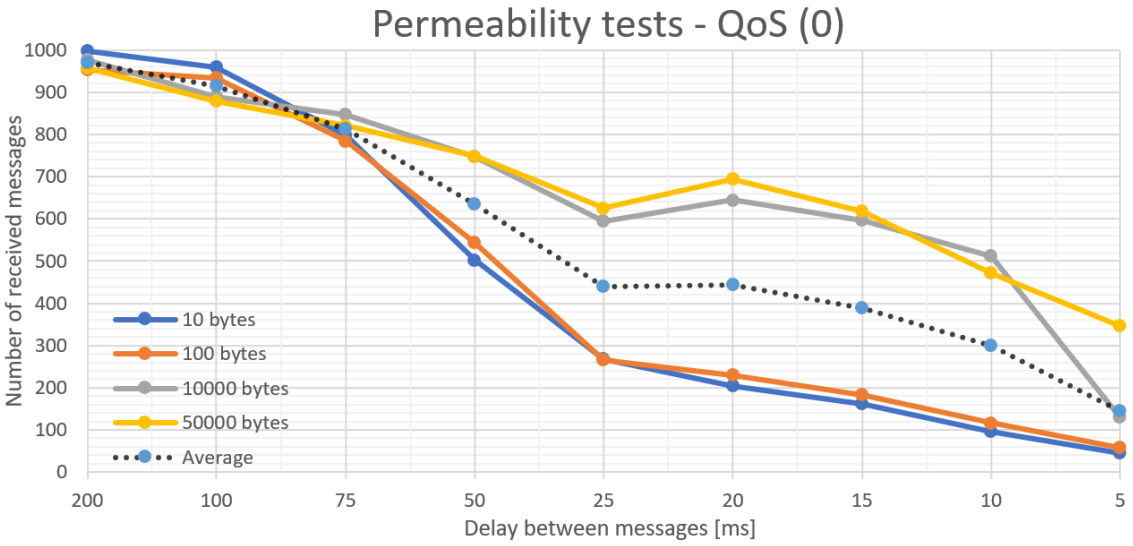

Figure 5.4. Permeability test QoS - At most once

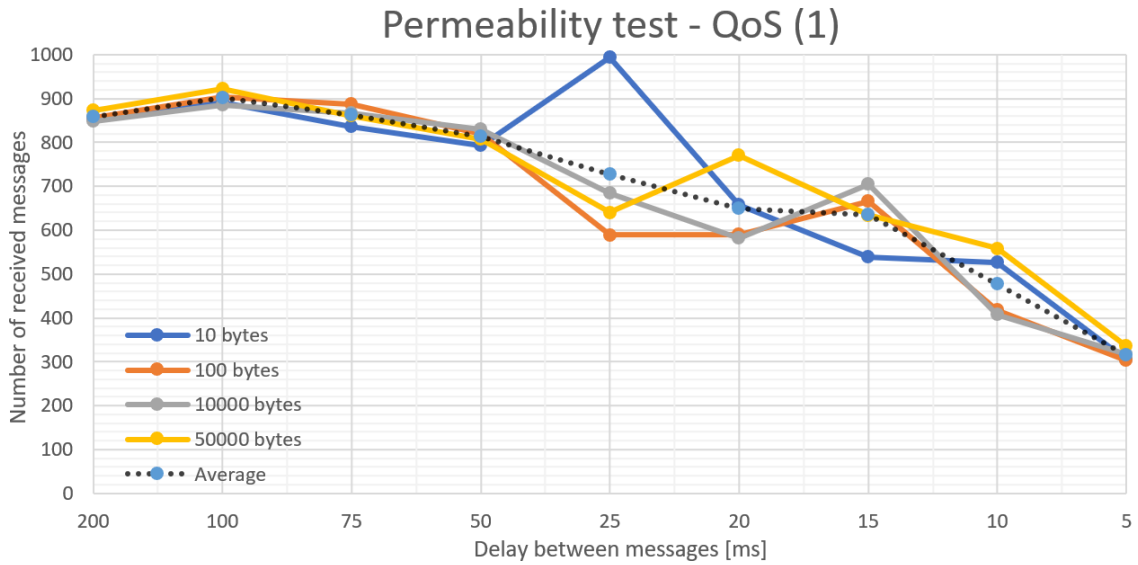

Figure 5.5. Permeability test QoS - At least once

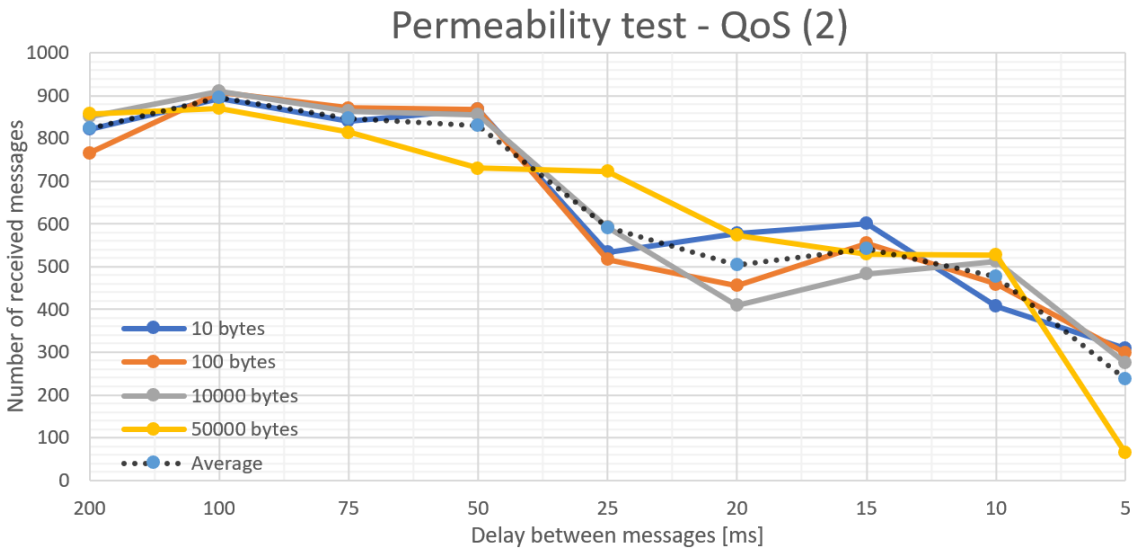

Figure 5.6. Permeability test QoS - Exactly once

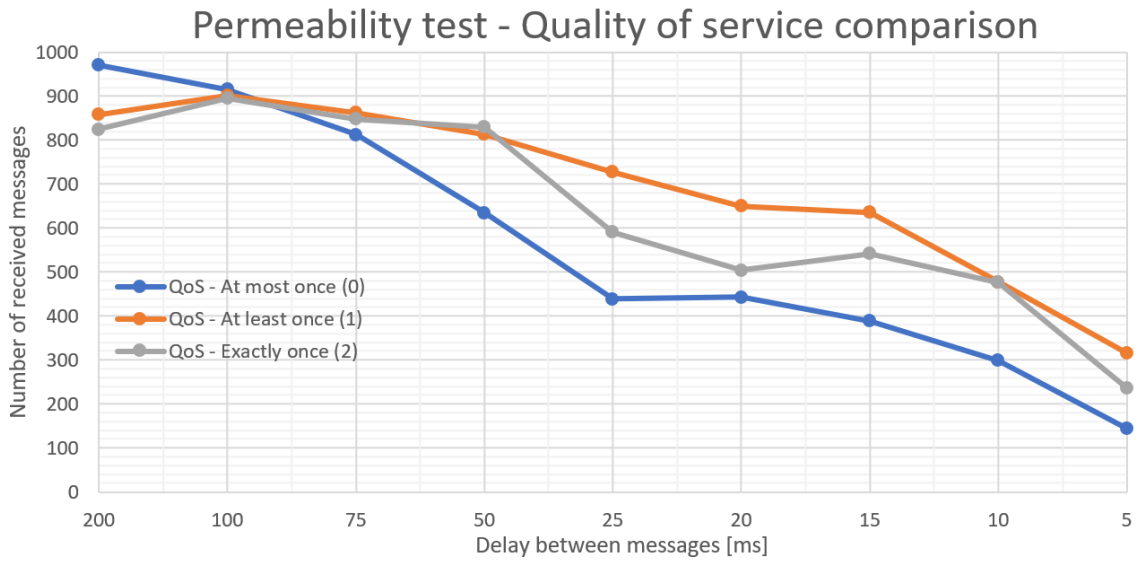

Figure 5.7. Permeability test QoS comparision

5.3 Echo test

This test tries to find the relation between message delivery time and the size of the message. The test uses the same setup as the previous tests and the same environment. The setup is showed in figure 5.3.

For the testing purpose, a pattern and special script were created. The Message provider sends data messages with a delay of 100 ms as a result of previous tests. The tester can choose the size of sending messages. The Message provider sent one hundred messages with time information and additional data to fulfill message size demand to the Visualization device. This test uses coordinated universal time (UTC) for correct echo evaluation. This time is the same for both devices. The Visualization device read the time label provided by the Message provider and compare it with the time of receiving the message. This delivery time is saved. The Visualization device also counts the number of received messages. After receiving all data, the tester calls the

Size of messages [byte]	10	100	10000	50000	100000	200000
Delivery time [ms]	52.667	41.181	43.949	82.089	108.899	194.3978
Number of received messages	90	88	94	90	89	93

Table 5.4. Echo test

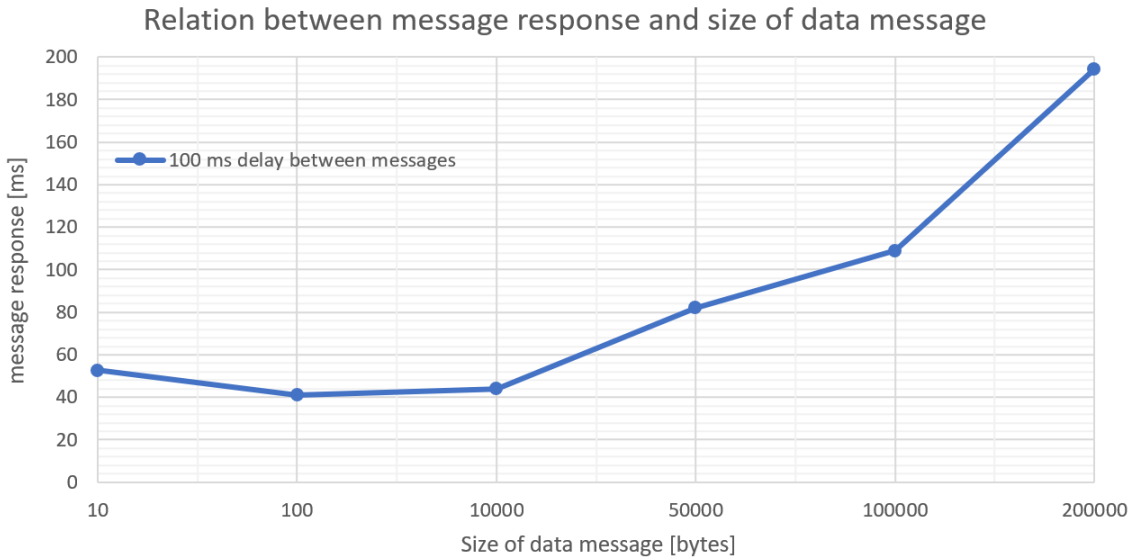

Figure 5.8. Echo test

method to visualize results that count average delivery time according to the number of received messages. These data are shown in table 5.4.

Figure 5.8 shows the relation between message delivery time and the size of the message. The shortest delivery time is between 100 B and 10 kB message sizes. That is the reason why the final application's messages are restricted to a maximum of 10 kB message size. It is interesting to observe that a message size 10 B has a higher delivery time than a bigger message. It could be caused by the broker architecture.

Chapter 6
Conclusion

This thesis was designed to display the status of industrial devices and their supporting data in augmented reality. The whole system depends on a reliable communication system used for control and data message transmission. This communication is based on the MQTT protocol and broker Mosquitto and uses the opportunity to send messages according to the report topic.

The whole system consists of the Broker and the Control server that manage all control communication. The next part of the system is the Message provider, which sends data from industrial devices. It supports several kinds of devices, including PLCs, Clouds, robots, and others. The only demand is the MQTT possibility of communication. The last part of the system is the Visualization device, which enables finding objects in the space and visualizing the data message on objects' position.

The tests in the evaluation chapter show that the proposed system works well. The tests also provide valuable data for future development. The first one, the images for image recognition, has to have enough features for detection. The second one, the system has the best transmission reliability when the delay between messages is around 100 ms, and it does not matter the size of the message. The last test verifies that the message size between 100 Bytes and 10 kBytes has the shortest delivery time.

The list below shows all reached goals.

- Reliable communication
- Image recognition
- Function at prototype
- Intuitive Visualization device control
- System variability
- Pattern's implementation
- Managing the whole system thanks to the Control server

Chapter 7
Future work

Currently, the system is still in the development process, and there are some tasks to improve the project. When these tasks are done, the system could enable work in the industry and help many workers with their job. The following improvements can further adjust the system.

It would be a good idea to move the Message provider program into Docker for collecting data from the Edge devices, which are one of the crucial devices for Industry 4.0. Thanks to this, it could be possible to collect more data.

The possibility of topics is not used at this moment. It should be great to create the tree topics' structure of all the industrial devices in a facility and choose subscribed topics according to this structure, not only from the available topic. After implementing this tree topics' structure, the user can easily choose the required topic, and the system's clarity will be much higher.

The patterns could also change in the future. It would be possible to replace the visualization pattern according to the user. For example, the user wants to view data on an industrial device's position and fix the position. For that, a visualization device has to know which pattern can be used for each message. There is a solution when the topic is provided with available visualization patterns for the topic, not solely with one like now, and the user can choose one pattern for visualization. That means a message provider has to supply control data for all available patterns and the visualization device choose the correct one.

Alternatively, there is another solution to the problem. It is possible to create a new pattern that provides the opportunity to choose the type of visualization according to the user's choice.

In addition, in the next part of this project, it would be suitable to implement a Visualization client for Android and iOS devices. The visualization client is written in Unity, which enables to change the platform easily.

References

[1] GHOBAKHLOO, Morteza, Masood FATHI, Mohammad IRANMANESH, Parisa MAROUFKHANI, and Manuel E. MORALES. Industry 4.0 ten years on: A bibliometric and systematic review of concepts, sustainability value drivers, and success determinants. Journal of Cleaner Production [online]. 2021, 302 [cit. 2021-04-22]. ISSN 09596526. Available from: doi:10.1016/j.jclepro.2021.127052

[2] NAKAGAWA, Elisa Yumi, Pablo Oliveira ANTONINO, Frank SCHNICKE, Rafael CAPILLA, Thomas KUHN, and Peter LIGGESMEYER. Industry 4.0 reference architectures: State of the art and future trends. Computers & Industrial Engineering [online]. 2021, 156 [cit. 2021-04-22]. ISSN 03608352. Available from: doi:10.1016/j.cie.2021.107241

[3] ENYOGHASI, Christian, and Fazleena BADURDEEN. Industry 4.0 for sustainable manufacturing: Opportunities at the product, process, and system levels. Resources, Conservation and Recycling [online]. 2021, 166 [cit. 2021-04-22]. ISSN 09213449. Available from: doi:10.1016/j.resconrec.2020.105362

[4] Ikea Place. Twnkls: a PTC company [online]. [cit. 2021-5-14]. Available from: https://twnkls.com/en/cases/ikea-place/

[5] Vuforia expert capture. PTC [online]. [cit. 2021-5-14]. Available from: https://www.ptc.com/en/products/vuforia/vuforia-expert-capture

[6] ROOPA, D., R. PRABHA, and G.A. SENTHIL. Revolutionizing education system with interactive augmented reality for quality education. Materials Today: Proceedings [online]. 2021 [cit. 2021-04-22]. ISSN 22147853. Available from: doi:10.1016/j.matpr.2021.02.294

[7] MASOOD, Tariq, and Johannes EGGER. Augmented reality in support of Industry 4.0—Implementation challenges and success factors. Robotics and Computer-Integrated Manufacturing [online]. 2019, 58, 181-195 [cit. 2021-04-22]. ISSN 07365845. Available from: doi:10.1016/j.rcim.2019.02.003

[8] MARINO, Emanuele, Loris BARBIERI, Biagio COLACINO, Anna Kum FLERI, and Fabio BRUNO. An Augmented Reality inspection tool to support workers in Industry 4.0 environments. Computers in Industry [online]. 2021, 127 [cit. 2021-04-22]. ISSN 01663615. Available from: doi:10.1016/j.compind.2021.103412

[9] NGUYEN-HOANG, Phuc, and Phuoc VO-TAN. Development An Open-Source Industrial IoT Gateway. In: *2019 19th International Symposium on Communications and Information Technologies (ISCIT)* [online]. IEEE, 2019, 2019, s. 201-204 [cit. 2021-04-18]. ISBN 978-1-7281-5009-3. Available from: doi:10.1109/ISCIT.2019.8905157

[10] HoloLens 2 technical specifications. MICROSOFT. HoloLens 2 - Overview [online]. [cit. 2021-04-22]. Available from: https://www.microsoft.com/en-us/hololens/hardware

[11] HoloLens 2 Gestures. MICROSOFT. HoloLens 2 - Documentation [online]. [cit. 2021-04-22]. Available from: https://docs.microsoft.com/cs-cz/windows/mixed-reality/mrtk-unity/features/input/gestures

[12] Vuforia: Developer portal [online]. PTC [cit. 2021-04-23]. Available from: https://library.vuforia.com/

[13] ČSN EN 60870-5-104 (334650): Systémy a zařízení pro dálkové ovládání - Část 5-104: Přenosové protokoly - Síťový přístup pro IEC 60870-5-101 používající normalizované transportní profily. Ed. 2. 2007.

[14] OASIS Message Queuing Telemetry Transport (MQTT) Technical Committee. OASIS [online]. 2021 [cit. 2021-04-23]. Available from: https://www.oasis-open.org/committees/mqtt/charter.php

[15] O'HARA, John. Toward, and Commodity Enterprise Middleware. Queue [online]. 2007, 5(4), 48-55 [cit. 2021-4-24]. ISSN 1542-7730. Available from: doi:10.1145/1255421.1255424

[16] ZHANG, Rui, Bo YAN, HongFei GUO, YongHeng ZHANG, Bin HU, HeXuan YANG, LuAn WANG, and Yan WANG. A New Environmental Monitoring System Based on WiFi Technology. Procedia CIRP [online]. 2019, 83, 394-397 [cit. 2021-4-24]. ISSN 22128271. Available from: doi:10.1016/j.procir.2019.04.088

[17] RabbitMQ: Tutorials. RabbitMQ [online]. [cit. 2021-4-24]. Available from: https://www.rabbitmq.com/getstarted.html

[18] RabbitMQ Tutorials: Topics. RabbitMQ [online]. [cit. 2021-4-24]. Available from: https://www.rabbitmq.com/tutorials/tutorial-five-dotnet.html

[19] RabbitMQ Tutorials: Remote procedure call (RPC). RabbitMQ [online]. [cit. 2021-4-24]. Available from: https://www.rabbitmq.com/tutorials/tutorial-six-dotnet.html

[20] .NET Core/5+ vs. .NET Framework for server apps. Microsoft Docs [online]. [cit. 2021-4-25]. Available from: https://docs.microsoft.com/cs-cz/dotnet/standard/choosing-core-framework-server

[21] MRTK library: Architecture overview. Microsoft Docs [online]. [cit. 2021-4-29]. Available from: https://docs.microsoft.com/cs-cz/windows/mixed-reality/mrtk-unity/architecture/overview

[22] MRTK library: Input system. Microsoft Docs [online]. [cit. 2021-4-29]. Available from: https://docs.microsoft.com/cs-cz/windows/mixed-reality/mrtk-unity/architecture/terminology

[23] RabbitMQ dotnet client. Github [online]. [cit. 2021-4-29]. Available from: https://github.com/rabbitmq/rabbitmq-dotnet-client

[24] Unity3D.Amqp: CymaticLab. Github [online]. [cit. 2021-4-29]. Available from: https://github.com/CymaticLabs/Unity3D.Amqp

[25] Holorabbit: fuadmefleh. Github [online]. [cit. 2021-4-29]. Available from: https://github.com/fuadmefleh/holorabbit

[26] Best Practices for Designing and Developing Image-Based Targets. Vuforia library [online]. [cit. 2021-5-10]. Available from: https://library.vuforia.com/features/images/image-targets/best-practices-for-designing-and-developing-image-based-targets.html

[27] Advanced Model Target Databases. Vuforia library [online]. [cit. 2021-5-10]. Available from: https://library.vuforia.com/articles/Solution/trained-model-target-datasets.html

[28] TROJAN, Ondrej. Multiplayer mobile game development using augmented reality. Prague, 2018. Bachelor's thesis. Czech Technical University in Prague. Supervisor Ing. David Sedlacek, Ph.D.

[29] AR ruler app. Google play [online]. [cit. 2021-5-15]. Available from: https://play.google.com/store/apps/details?id=com.grymala.aruler

[30] ADRIANTO, Dennise, Monica HIDAJAT, and Violitta YESMAYA. Augmented reality using Vuforia for marketing residence. In: 2016 1st International Conference on Game, Game Art, and Gamification (ICGGAG) [online]. IEEE, 2016, 2016, s. 1-5 [cit. 2021-5-15]. ISBN 978-1-5090-5479-4. Available from: doi:10.1109/ICGGAG.2016.8052642

[31] XIAO, Cheng, and Zhang LIFENG. Implementation of mobile augmented reality based on Vuforia and Rawajali. In: 2014 IEEE 5th International Conference on Software Engineering and Service Science [online]. IEEE, 2014, 2014, s. 912-915 [cit. 2021-5-20]. ISBN 978-1-4799-3279-5. Available from: doi:10.1109/ICSESS.2014.6933713

[32] SHI, Huichao, Li NIU, and Jinhao SUN. Construction of Industrial Internet of Things Based on MQTT and OPC UA Protocols. In: *2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)* [online]. IEEE, 2020, 2020, s. 1263-1267 [cit. 2021-04-18]. ISBN 978-1-7281-7005-3. Available from: doi:10.1109/ICAICA50127.2020.9182598

[33] SADIO, Ousmane, Ibrahima NGOM, and Claude LISHOU. Lightweight Security Scheme for MQTT/MQTT-SN Protocol. In: 2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS) [online]. IEEE, 2019, 2019, s. 119-123 [cit. 2021-04-22]. ISBN 978-1-7281-2949-5. Available from: doi:10.1109/IOTSMS48152.2019.8939177