**ZÁPADOČESKÁ UNIVERZITA V PLZNI**

**FAKULTA STROJNÍ**

**Studijní program:** **N0715A270012 – Průmyslové inženýrství a management**

**Studijní specializace:** **Bez specializace**

# DIPLOMOVÁ PRÁCE

**Možnosti zobrazování průmyslových metadat pomocí rozšířené reality**

Autor: Bc. Matěj JEDLIČKA

Vedoucí práce: Doc. Ing. Petr HOŘEJŠÍ, Ph.D.

Akademický rok 2023/2024

# Zadání DP

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta strojní
Akademický rok: 2023/2024

Studijní program: Průmyslové inženýrství a management
Forma studia: Prezenční

## Podklad pro zadání DIPLOMOVÉ práce studenta

| | |
|---|---|
| Jméno a příjmení: | **Bc. Matěj JEDLIČKA** |
| Osobní číslo: | **S22N0012P** |
| Adresa: | Sokolovská 335, Staré Sedlo, 35601 Sokolov 1, Česká republika |
| Téma práce: | Možnosti zobrazování průmyslových metadat pomocí rozšířené reality |
| Téma práce anglicky: | Augmented Reality Options for Displaying Industrial Metadata |
| Jazyk práce: | Čeština |
| Vedoucí práce: | Doc. Ing. Petr Hořejší, Ph.D.<br>Katedra průmyslového inženýrství a managementu |

Zásady pro vypracování:

1. Úvod
2. Současná obdobná řešení
3. Průmyslová data (dělení, senzory)
4. Možné variantní přístupy
5. Realizace vlastního demonstračního řešení
6. Závěr

Konzultant: Ing. Tomáš Macháč

Seznam doporučené literatury:


- Okita, A., Learning C# Programming with Unity 3D, second edition, Routledge 2019, ISBN-13: 978-1138336810
- Sung, K., Smith, G., Basic Math for Game Development with Unity 3D: A Beginner's Guide to Mathematical Foundations, Apress 2019, ISBN 978-1484254424
- Linowes, J., Unity 2020 Virtual Reality Projects: Learn VR development by building immersive applications and games with Unity 2019.4 and later versions, 3rd Edition, Packt Publishing 2020, ISBN 978-1839217333
- LaValle, S. M., Virtual Reality, Cambridge University Press 2023, dostupné online na http://lavalle.pl/vr/
- Oficiální Unity3D návody dostupné na https://learn.unity.com/


Podpis studenta: Datum:

Podpis vedoucího práce: Datum:

# Prohlášení o autorství

Předkládám tímto k posouzení a obhajobě diplomovou práci zpracovanou na závěr studia na Fakultě strojní Západočeské univerzity v Plzni.

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této diplomové práce.

V Plzni dne: …….  …….

podpis autora

# ANOTAČNÍ LIST DIPLOMOVÉ PRÁCE

| | | | |
|---|---|---|---|
| **AUTOR** | **Příjmení**<br>Jedlička | **Jméno**<br>Matěj | |
| **STUDIJNÍ PROGRAM** | N0715A270012 – Průmyslové inženýrství a management | | |
| **VEDOUCÍ PRÁCE** | **Příjmení (včetně titulů)**<br>Doc. Ing. Hořejší, Ph.D. | **Jméno**<br>Petr | |
| **PRACOVIŠTĚ** | ZČU – FST – KPV | | |
| **DRUH PRÁCE** | **DIPLOMOVÁ** | ~~**BAKALÁŘSKÁ**~~ | **Nehodící se škrtněte** |
| **NÁZEV PRÁCE** | Možnosti zobrazování průmyslových metadat pomocí rozšířené reality | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **FAKULTA** | strojní | | **KATEDRA** | KPV | | **ROK ODEVZD.** | 2024 |

**POČET STRAN (A4 a ekvivalentů A4)**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **CELKEM** | 98 | | **TEXTOVÁ ČÁST** | 67 | | **GRAFICKÁ ČÁST** | 31 |

| | |
|---|---|
| **STRUČNÝ POPIS (MAX 10 ŘÁDEK)**<br><br>**ZAMĚŘENÍ, TÉMA, CÍL POZNATKY A PŘÍNOSY** | Diplomová práce se zabývá prozkoumáním možností zobrazení průmyslových metadat v rámci rozšířené reality a vytvoření vlastního řešení. První část práce se zaměřuje teoretický popis pojmů jako jsou Průmysl 4.0, rozšířená reality, průmyslová data a další. V další části práce jsou prozkoumány softwarové možnosti pro tvorbu aplikace. Následuje popis tvorby GUI, databáze, komunikace a samotné aplikace. Na závěr je popsáno převedení výstupu práce do reálného prostředí. Výsledek práce bude sloužit pro vědecké účely. |
| **KLÍČOVÁ SLOVA**<br><br>**ZPRAVIDLA JEDNOSLOVNÉ POJMY, KTERÉ VYSTIHUJÍ PODSTATU PRÁCE** | Rozšířená realita, metadata, vizualizace, Vuforia, Unity3D, 3D model, internet věcí |

# SUMMARY OF DIPLOMA SHEET

| | | | |
|---|---|---|---|
| **AUTHOR** | **Surname**<br>Jedlička | **Name**<br>Matěj | |
| **STUDY PROGRAMME** | N0715A270012 – Průmyslové inženýrství a management | | |
| **SUPERVISOR** | **Surname (Inclusive of Degrees)**<br>Doc. Ing. Hořejší, Ph.D. | **Name**<br>Petr | |
| **INSTITUTION** | ZČU – FST – KPV | | |
| **TYPE OF WORK** | **DIPLOMA** | ~~**BACHELOR**~~ | **Delete when not applicable** |
| **TITLE OF THE WORK** | Augmented Reality Options for Displaying Industrial Metadata | | |

| | | | | | |
|---|---|---|---|---|---|
| **FACULTY** | Mechanical Engineering | **DEPARTMENT** | Industrial engineering and management | **SUBMITTED IN** | 2024 |

**NUMBER OF PAGES (A4 and eq. A4)**

| | | | | | |
|---|---|---|---|---|---|
| **TOTALLY** | 98 | **TEXT PART** | 67 | **GRAPHICAL PART** | 31 |

| | |
|---|---|
| **BRIEF DESCRIPTION**<br><br>**TOPIC, GOAL, RESULTS AND CONTRIBUTIONS** | This thesis explores the possibilities of displaying industrial metadata in augmented reality and creating a custom solution. The first part of the thesis focuses on the theoretical description of concepts such as Industry 4.0, augmented reality, industrial data, and others. The next part of the thesis explores the software options for creating the application. This is followed by a description of GUI creation, database, communication, and the application itself. Finally, the translation of the output of the thesis into a real environment is described. The result of the work will be used for scientific purposes. |
| **KEY WORDS** | Augmented reality, metadata, visualization, Vuforia, Unity3D, 3D model, internet of things |

# Obsah

# Seznam obrázků

# Seznam grafů

# Přehled použitých zkratek a symbolů

| | |
|---|---|
| FST | Fakulta strojní |
| KPV | Katedra průmyslové inženýrství a výzkumu |
| ZČU | Západočeská univerzita v Plzni |
| HW | Hardware |
| SW | Software |
| AR | Rozšířená realita (Augmented reality) |
| VR | Virtuální realita (Virtual reality) |
| BIM | Informační model budovy (Building Information Modeling) |
| CPS | Kyber fyzikální systém (Cyber physical systém) |
| AV | Rozšířená virtualita (Augmented virtuality) |
| MR | Smíšená realita (Mixed reality) |
| MAR | Mobilní rozšířená realita (Mobile mixed reality) |
| GPS | Globální navigační systém (Global positioning systém) |
| HMD | Náhlavní souprava (Head-mounted display) |
| FOV | Zorné pole (Field of Vision) |
| QR | Quick Response |
| CAD | Počítačem podporovaný design (Computer Aided Design) |
| SLAM | Simultaneous Localisation and Mapping |
| MEMS | Mikro elektromechanická součástka (Micro Electro Mechanical Systems) |
| IoT | Internet věcí (Internet of Things) |
| MTG | Model Target Generator |
| MQTT | Message Queuing Telemetry Transport |
| GUI | Grafické uživatelské rozhraní (Graphic user interface) |
| SQL | Strukturovaný dotazovací jazyk (Structured query Language) |
| APK | Aplikační balíček pro Android (Android Application Package) |
| API | Rozhraní pro programování aplikací (Application programming interface) |
| NR | Node-RED |

# Úvod

V současné éře, kde se technologický pokrok ubírá směrem k sofistikovanějším výrobním postupům a informačním technologiím, se přístup k výrobě stále více posouvá od digitálního směrem k inteligentnímu paradigmatu. Firmy se nyní nevyhnutelně musí zaměřit na zvyšování celkového stupně industrializace, automatizace a digitalizace, aby udržely krok s vyššími standardy efektivity, odbornosti a konkurenceschopnosti. V tomto nekonečném procesu inovací se stále častěji setkáváme s pojmem Průmysl 4.0. Tento koncept zavádí do výrobního prostředí propojené informační sítě, kde lidé a stroje komunikují jako nikdy předtím. V této interakci vyniká trend rozšířené reality, která spojuje virtuální a reálný svět, čímž umožňuje zobrazovat aktuální stav strojů pomocí dat získaných senzory. Tato forma komunikace mezi člověkem a strojem, prostřednictvím rozšířené reality, otevírá firmám nepřeberné možnosti pro rozvoj a údržbu. Od zlepšení procesů až po lepší prostředí pro školení zaměstnanců, rozšířená realita se stává klíčovým prvkem moderního průmyslu, přinášejícím nové dimenze efektivity a inovace.

Autor této práce měl na začátku základní programovací zkušenosti a žádné zkušenosti s vytvářením aplikací pro rozšířenou realitu.

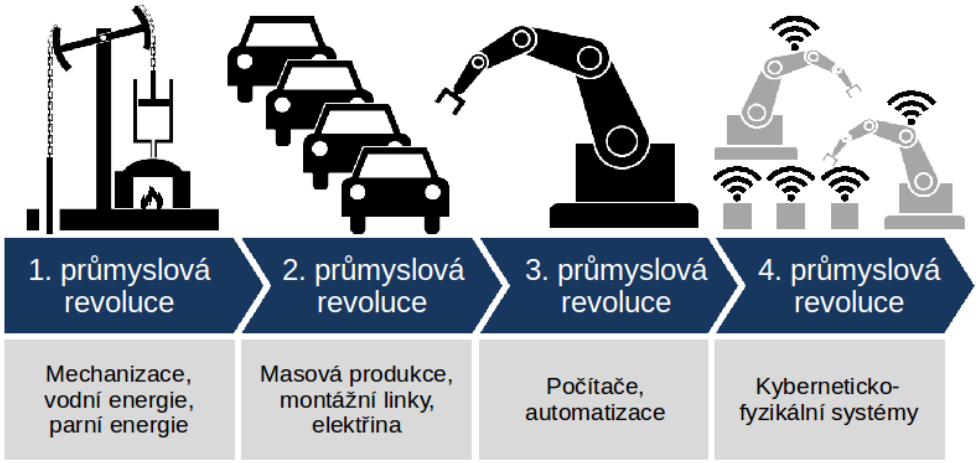
Hlavním cílem této práce je rešerše současných obdobných řešení, a vytvoření aplikace v rozšířené realitě, která má za úkol vizualizovat, skrze médium, kterým bude chytrý telefon, průmyslová data ze senzorů. Uživatel by s touto aplikací měl být schopen sledovat reálné dění na pracovištích a neměla by také chybět možnost interakce s jednotlivými 3D modely. V práci bude popsána instalace a nastavení veškerých nástrojů tak, aby byla aplikace snadno replikovatelná a mohla sloužit jako základ pro další výzkum. Aplikace byla naprogramována v jazyce C#, nativním programovacím jazyce softwarového prostředí Unity3D, ve kterém rovněž byla vytvořena grafické rozhraní. Komunikace byla zajištěna skrze software Mosquitto od společnosti Eclipse, který poskytuje možnost hostovat MQTT server. Veškerá vygenerovaná data jsou spravována v rámci databázového prostředí MySQL. Rovněž bylo vytvořeno vlastní API pomocí nástroje Node-RED, který využívá programovacího jazyka JavaScript.

První část této práce se zabývá teoretickým základem, tedy vysvětlením důležitých pojmů jako je rozšířená realita, metadata či senzory. V této části byla rovněž provedena rešerše současných obdobných řešení. Druhá část se zabývá samotným vytvořením aplikace a její zhodnocení.

# 1 Průmysl 4.0

Pojem průmysl 4.0 se objevil již v roce 2011, kde byl představen na veletrhu v Hannoveru. Od té doby se tomuto tématu věnovala řada vědeckých publikací, článků a konferencí. Termín "Průmysl 4.0" se používá pro průmyslovou revoluci, která probíhá v současnosti. Této průmyslové revoluci předcházely tři další průmyslové revoluce. Krédem první průmyslové revoluce, která započala v druhé polovině 18. století, byla mechanizace – začala vynálezem tkacího stavu a stroji poháněnými energií vody a páry. Následovalo intenzivní využívání elektrické energie v 70. letech 19. století, které vedlo k druhé průmyslové revoluci – její počátek je spojován se zavedením první pásové linky. Třetí „digitální“ průmyslová revoluce, je období rozsáhlé digitalizace a je spojována s uvedením první výrobní linky řízené počítačem v roce 1969[1].

Na rozdíl od těchto revolucí, které byli popsány ex post, byla 4. průmyslová revoluce predikována a priori. Její „náplň“ se tedy stále rozvíjí. Obecným konsensem je ale pojem propojení, a to ve formě CPS. Je důležité zmínit, že v dnešní době se již hovoří o páté průmyslové revoluci, která se zejména soustředí na globální dopad systémů[2]. Mnozí ale konkurují, že tento „Green“ trend však nelze explicitně spojit s žádnou novou technologií a pokud k tomuto připojíme fakt, že Průmysl 4.0 není plně definovaný, lze 5.0 chápat jako jakousi nadstavbu Průmyslu 4.0. Přehled průmyslových revolucí lze vidět na Obrázek 1-1 Průmyslová revoluce.




**Obrázek 1-1 Průmyslová revoluce**

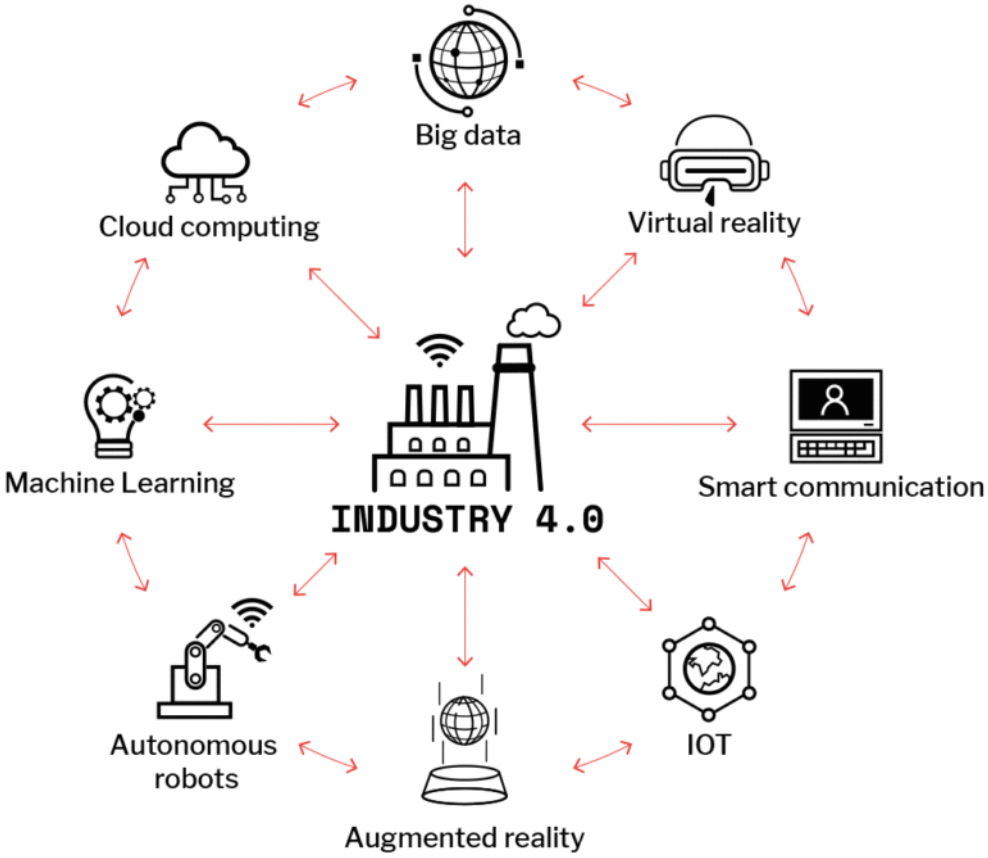
Základem Průmyslu 4.0 je zvýšit produktivitu a efektivitu a zároveň dosáhnout vyššího stupně automatizace.

V 21. století jsou životní cykly výrobků kratší a spotřebitelé požadují složitější, jedinečné a personalizované výrobky ve větším množství, což představuje výzvu pro výrobu. Cílem Průmyslu 4.0 je zajistit flexibilní, hospodárnou a efektivní výrobu. Všechny části výrobního procesu budou komunikovat se všemi ostatními částmi prostřednictvím centrálního systému řízení výroby, čímž se dosáhne vyššího stupně automatizace.

Ke zvýšení produktivity využívá Průmysl 4.0 technologií jako: AR, Big Data[1], Cloud computing, CPS, RFID čipy, internet věcí a služeb, Komunikace typu stroj-stroj, Automatizace, Smart X[2], kyber bezpečnost a další[1]. Tyto technologie ovlivňují způsob výroby, ale také

[1] Soubory dar příliš velké na to, aby je dokázaly zachytit, zpracovat, nebo spravovat běžně používané soubory
[2] Inteligentní funkce v jakýchkoli věcech, např. v produktech, sítích, továrnách atd.

vytváří dodatečnou hodnotu pro zákazníka. Propojení systému zajišťuje, že výrobek je „Smart“. Hlavní myšlenkou chytrého výrobku je rozšířit roli výrobku tak, aby se stal aktivní, nikoli pasivní součástí systému. Výrobky mají paměť, v níž jsou uloženy provozní údaje a požadavky, takže výrobek sám žádá o potřebné zdroje a organizuje výrobní procesy potřebné k jeho dokončení[3]. Díky tomu lze sledovat celý životní cyklus produktu. Podniky tak získávají detailní přehled o spotřebě jednotlivých výrobků a funkcí strojů, a mohou tak sledovat a řídit výkonnost, dodávky a logistické trasy.

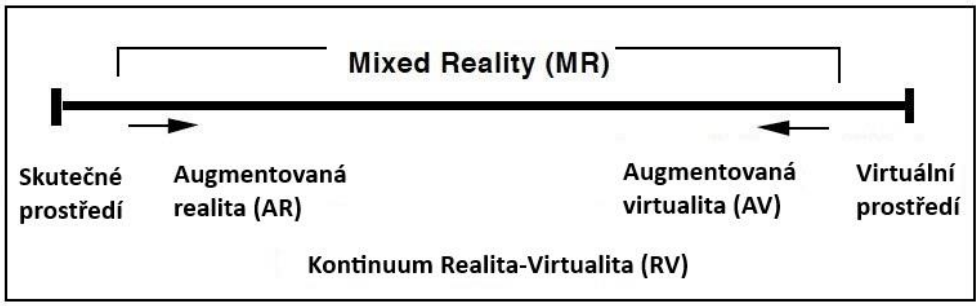



**Obrázek 1-2 Komponenty Průmyslu 4.0**

# 2 Rozšířená realita

V době, kdy se hranice mezi fyzickou a digitální sférou stále ztenčují, se rozšířená realita ukazuje jako průlomová technologická hranice, která propojuje virtuální a hmatatelný svět. Rozšířená realita představuje změnu paradigmatu ve způsobu, jakým vnímáme informace a jak komunikujeme s prostředím, a nabízí jedinečné spojení prostředí reálného světa s počítačově generovanými vylepšeními.

Pojem virtuální reality[4] dnes zná téměř každý, to samé se ale nedá říct od realitě rozšířené. Rozšířená realita se liší od virtuální reality tím, že nevytváří zcela nové světy, ale obohacuje stávající reálný svět o počítačem vytvořené objekty. Existuje kontinuum realita-virtualita, které popisuje přechod mezi reálným a virtuálním prostředím. Levá část kontinua zahrnuje prostředí tvořené pouze reálnými objekty, zatímco pravá část představuje prostředí složené pouze z virtuálních objektů. Smíšená realita je definována jako prostředí, ve kterém jsou reálné a virtuální objekty kombinovány. V rozšířené realitě převažuje reálná část, zatímco v rozšířené virtualitě převažuje virtuální část[5].




**Obrázek 2-1 Kontinuum Realita-Virtualita**

Rozdíl mezi AR a AV je, v případě AR, projekce digitálních artefaktů do reálného světa, a pro AV se jedná o projekci reálných objektů do virtuálního prostředí.

Jedna z prvních definic AR byla popsána Ronaldem T. Azumou v roce 1997[6], který AR popisuje jako systém který:

- Kombinuje virtuální elementy s reálným prostředím
- Je interaktivní a překrytí je v reálném čase
- Je registrována ve 3D

Tato definice jako taková, není spjata se žádnou konkrétní technologií a nedefinuje dostatečně pojem AR[7]. Huang et al.[8] dodává do této definice ještě čtvrtý bod, kterým je mobilita. Tím vzniká pojem MAR, který využívá schopnosti a funkce mobilních zařízení, jako jsou chytré telefony, tablety a nositelná zařízení, v kombinaci s všudypřítomným a cenově dostupným přístupem k internetu a pokrokem v oblasti kooperativních sítí, počítačového vidění a mobilního cloud computingu[9]

## 2.1 Oblasti použití

AR je technologie, která našla využití ve spoustě oblastí a její presence stále roste. Downey předkládá vyčerpávající přehled potenciálu AR v digitálním článku[10] v kontextu náhlavních souprav, který je však dostatečný pro objasnění použitelnosti této technologické oblasti a který

přesně identifikuje celý potenciál této oblasti, jako je rozpoznávání obličeje, rozpoznávání objektů a návrh personalizovaných rozšířených informací.

Následující příklady představují pouze část oblastí, které jsou zmíněné v článku:

- Stavebnictví a architektura – projektové řízení s kombinací reality s digitálními modely, stavebními pokyny nebo předchozí vizualizací konečného výsledku.
- Bezpečnostní složky – biometrický záznam a čtení na místech činu nebo detekce.
- Medicína – interakce s lékaři nebo databází s diagnostikou na dálku, rozšíření operací pomocí AR
- Psychologie – Studium a nacvičování sociálních interakcí s výrobky a lidmi

## 2.2 Typy rozšířené reality

Aby bylo možné používat aplikace rozšířené reality, je nutný sledovací software, který zachytí reálné prostředí a přidá do něj virtuální objekty. Přesnost tohoto začlenění závisí na oblasti použití, přičemž v chirurgii je vysoce přesná sledování nezbytné, zatímco např. v herním využití není rozhodující.

Sledovací zařízení by mělo co nejpřesněji a v reálném čase zachytit skutečné prostředí, všechny objekty v něm rozpoznat a sledovat úhel pohledu uživatele a/nebo polohu značky. Rozlišují se dva hlavní principy[11]:

- Inside-out tracking určuje informace o sledování sám pohybující se objekt. Údaje poskytuje prostředí, např. samotné značky.
- Outside-in je v případě kdy sledovaný objekt nemá žádné znalosti o své vlastní poloze a orientaci.

Sledovací zařízení používaná na principu inside-out jsou pasivní, a proto výrazně levnější a jsou stále více upřednostňována. Ke sledování se používají specifické senzory nebo kombinace různých senzorů. V zásadě lze rozlišit dvě různé metody: Nevizuální a vizuální sledování.

### 2.2.1 Nevizuální sledování

Mezi nevizuální metody sledování patří například[12]:

Kompas – Magnetické pole země se používá k určení orientace vzhledem k zemským osám.

GPS – Poloha přijímacího zařízení (např. chytrého telefonu) se vypočítá pomocí satelitního systému určování polohy.

Ultrazvukové snímače – Vzdálenost, a tedy i vzájemná poloha se určuje měřením doby šíření ultrazvukových vln mezi několika vysílači a přijímači.

Optoelektronické snímače – Vzdálenost mezi několika vysílači a přijímači se měří pomocí optoelektronických snímačů; jedná se o technologii snímání v neviditelném světle, např. prostřednictvím infračerveného záření.

Inerciální senzory – K měření náklonu zařízení se používají gyroskopy. Proměření pohybu podél přímé osy se využívá akcelerometr. Pro získání orientace pak slouží magnetometr[13].

### 2.2.2 Vizuální sledování

Vizuální sledování se obvykle provádí pomocí videokamery ve dvou krocích:

- Inicializace: V obraze kamery se vyhledá sledovaný vzor a vypočítá se jeho orientace. Značka nemusí být zarovnána ortogonálně ke kameře.
- Sledování nebo předvídání možného pohybu: V tomto kroku se sleduje obraz zkreslený orientací na dalších snímcích videozáznamu a omezuje se oblast, která má být analyzována.

Existují několik technologií vizuálního sledování, z nichž nejpoužívanější jsou:

1. Hand-held devices neboli „ručně držené zařízení“, je spojené se zařízeními jako chytré telefony a tablety. Smartphone má v dnešní době téměř každý, proto se jedná o nejrozšířenější typ. Velkou výhodou je tedy dostupnost, ale také fakt, že moderní telefony jsou vybavené technologiemi jako GPS, gyroskop, kompas, kamera, displej, připojení k internetu a výpočetní jednotku. Všechny tyto funkce jsou k dispozici uvnitř malého zařízení, které se vejde do ruky. Na druhou stranu je to i nevýhoda, jelikož zařízení je nutno držet a soustředit se na obrazovku a je možnost ztráty povědomí o okolí. Tento fakt jde proti základnímu požadavku na AR, kterým je obohacení reality.

2. Kamera je připevněna na hlavě uživatele, typ HMD. Sledovací zařízení pak vypočítává polohu hlavy uživatele. S ohledem na přesnou vizualizaci hraje stále důležitou roli konstantní úhel pohledu. Jedná se o využívání vyhrazených náhlavních souprav, často označované pouze jako brýle. Oproti mobilnímu zařízení poskytuje HMD podstatně větší imerzi a volnost pohybu, jelikož uživatel není závislý na umístění zařízení. I přesto, že HMD brýle poskytují větší imerzi, tak se stále potýkají s limitovaným FOV (field of view), neboli zorným polem. Vyřešení tohoto problému slibuje společnost Apple se svým produktem Apple Vision. Dalším negativem tohoto způsobu zobrazování je cena. Kvalitní HMD s pokročilými funkcemi jsou drahé (Apple vision stojí přibližně 80 000 Kč), a právě tyto náklady mohou být překážkou v přijetí, zejména na spotřebitelském trhu. Příkladem mohou být Google Glass, které mimo jiné neuspěli právě kvůli ceně.

3. Alternativně může být kamera, například webová kamera, také připevněna k počítači a tracker vypočítává polohu skutečných objektů pomocí rutin pro zpracování obrazu. Tímto způsobem lze vypočítat jak polohu kamery vzhledem ke scéně, tak polohu a orientaci objektů v ní umístěných[14].

Rozšířenou realitu lze na základě vyhledávání vzorů obecně rozdělit do dvou hlavních kategorií. Toto rozdělení vychází ze schopnosti technologie se orientovat v prostoru[15]:

1. Rozšířená realita založená na značkách:

   V AR založené na značkách se ke spuštění zobrazení digitálního obsahu používají vizuální značky, jako jsou QR kódy, obrázky nebo jiné identifikovatelné vzory. Systém AR rozpoznává tyto značky prostřednictvím kamery zařízení, a když je značka detekována, související digitální informace se v reálném čase překryjí na značku. Tento typ AR se často používá v aplikacích, kde je rozhodující přesné sledování a sladění digitálního obsahu s fyzickým světem.

   Umělé značky, nebo také markery, jsou široce používaným prostředkem pro označování objektů v systémech rozšířené reality. Umělá značka je opticky optimalizována tak, aby ji sledovací zařízení dokonale rozpoznalo. Umělé značky navíc obsahují kódy, které definují informace, jež se mají zobrazit v místě značky[16]. Marker je dvourozměrný nebo trojrozměrný objekt, který lze díky jeho typu a tvaru snadno identifikovat (sledovat) pomocí kamery. Použití markerů pomáhá optimalizovat inicializaci a sledování. Umělé značky usnadňují sledovacímu mechanismu, tj. příslušnému softwaru pro zpracování obrazu, určit polohu a díky jejich jasné geometrii

je zarovnat s kamerou. V závislosti na aplikaci je důležité přesně určit polohu kamery, pokud mají být například do prostředí integrovány trojrozměrné objekty[17].

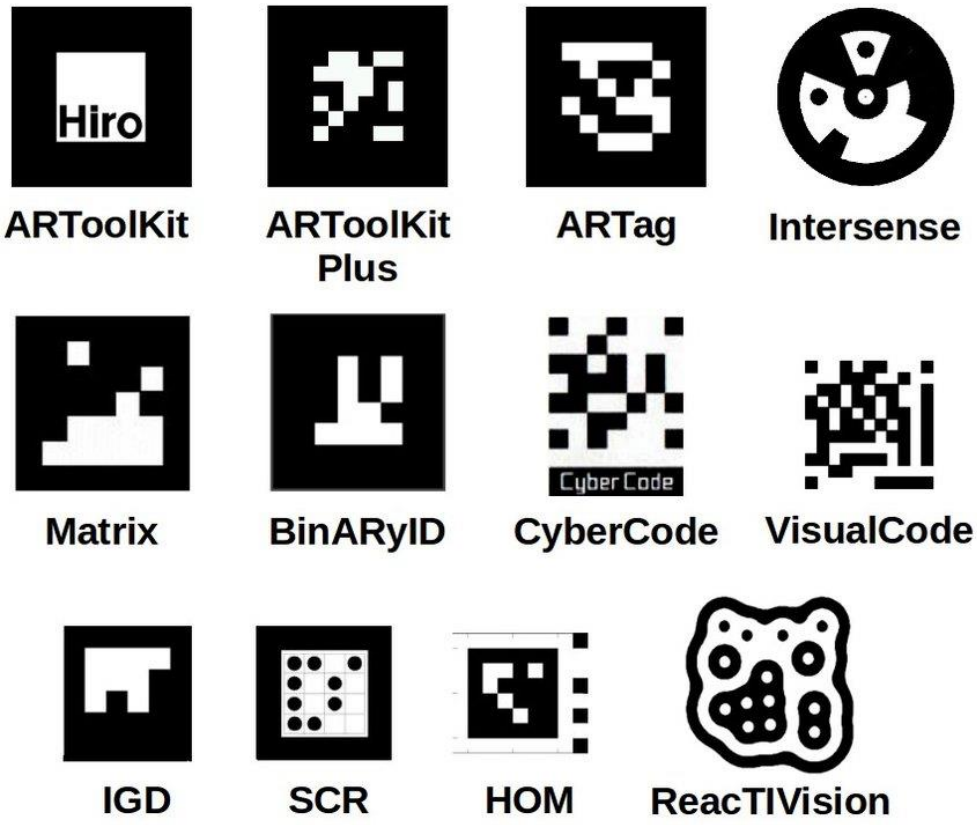
V ideálním případě splňuje značka všechna následující kritéria[16]:

- Pomáhá sledovacímu zařízení při určování polohy a orientace kamery.
- Je optimálně rozpoznatelná bez ohledu na její orientaci.
- Je součástí řady snadno rozlišitelných markerů, aby bylo možné označit velké množství objektů.
- Lze snadno rozpoznat a identifikovat bez velkého výpočetního úsilí.
- Rozpoznatelná kamerou i na větší vzdálenost.

Značka je dále určena následujícími vlastnostmi[16]:

A. Tvar: Aby bylo možné jednoznačně určit polohu objektu v trojrozměrném prostoru, jsou zapotřebí alespoň čtyři nelineární body. V ideálním případě by tyto body měly být rohy čtverce, aby je bylo možné stejně dobře rozpoznat v jakékoli orientaci. Neznamená to ale, že celá značka musí mít tvar čtverce.
B. Barvy: I když barevné značky umožňují větší rozmanitost než jednobarevné značky, existují technické důvody proti. Mnoho systémů digitálních fotoaparátů reprodukuje lidské vidění, které reaguje citlivěji na jas než na chrominanci (barvu). Na jednom snímku je proto více jasových informací než barevných. Monochromatická značka také umožňuje analyzovat čistě šedé obrazy, což na jedné straně snižuje spotřebu paměti a na druhé straně umožňuje použití efektivnějších algoritmů.
C. Umístění: Počínaje černým rámečkem, který ohraničuje značku, je ideální, pokud je umístěna na bílém pozadí. Tím se maximalizuje dosažený kontrast a usnadní se detekce okrajů značky. Tohoto kontrastu černé a bílé lze snadno dosáhnout vytištěním značky na bílé pozadí.
D. Identifikace: Aby bylo možné značky rozlišit, musí být uvnitř rámečku snadno rozlišitelné obrázky. Aby bylo k dispozici velké množství různých značek a aby se minimalizovala korelace mezi jednotlivými snímky, je nejlepším řešením použití dvourozměrného čárového kódu.

Příklady vyvinutých značek pro systémy rozšířené reality jsou k Obrázek 2-2 Příklady značek pro AR.[18]




**Obrázek 2-2 Příklady značek pro AR**

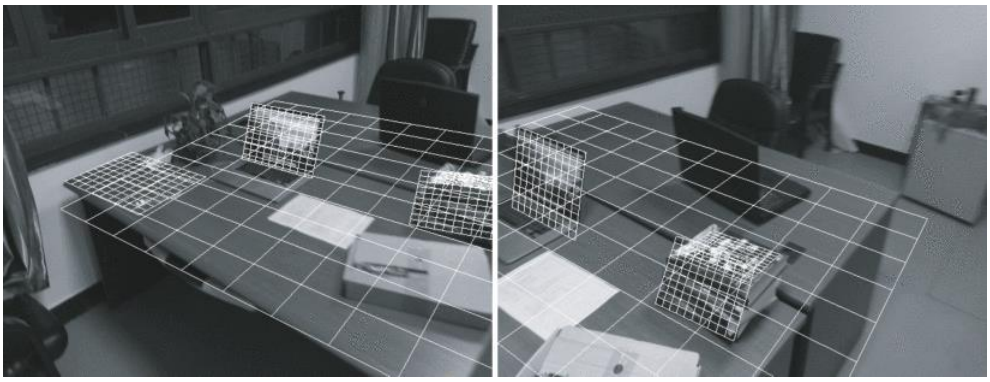
2. Rozšířená realita bez značek

Použití značek není vždy výhodné nebo je v některých aplikacích z estetických důvodů prostě nežádoucí. Například v rozsáhlých prostředích je třeba umístit velké množství umělých značek[14]. Tento problém se řeší „marker-less“ sledováním, tj. sledovacích metod, které se obejdou bez umělých značek. Jako referenční body se používají přirozené tvary vyskytující se ve scéně[19]. Mezi ně patří například čáry, body a kruhy.

Další možností vizuálních sledovacích systémů jsou metody založené na modelech. V nejjednodušším případě je k dispozici dvourozměrná šablona detekovaného objektu, tj. dvourozměrný obraz povrchu detekovaného objektu. Objekt je rozpoznán na základě známých rysů na povrchu a je zahájeno sledování[20]. Tento typ značky je znám jako 2D značka nebo texturní značka.

Sledovací zařízení rozpozná hrany známých objektů a přiřadí je k objektům v rámci modelu CAD. Znalost struktury a uspořádání scény zvyšuje spolehlivost a výkonnost trackeru. Tento přístup také umožňuje lépe předvídat pohyb skrytých objektů[21]. Místo nutnosti vytvořit kompletní prostředí v podobě modelu CAD však lze sledování inicializovat také pomocí modelu CAD. Pro tento účel je v prostředí k dispozici objekt, jehož CAD model je k dispozici. Tento model se pak použije k určení polohy a směru kamery. Systém poté vyhledá v prostředí prvky, pomocí kterých bude pokračovat ve sledování. Tento přístup funguje i v dynamicky se měnícím nebo z velké části neznámém prostředí[22].

Dalším vývojovým stupněm je tzv. metoda SLAM. Tato metoda vůbec nepoužívá značky. Je proto také výpočetně velmi náročná metoda, ale umožňuje rozpoznávat zcela neznámá prostředí[23] (viz obr. 2-3[24]).



**Obrázek 2-3 Metoda SLAM**

AR založená na markerech i bez markerů mají své vlastní silné stránky a aplikace a výběr mezi nimi závisí na konkrétních požadavcích a zkušenosti s AR. AR založená na značkách nabízí přesné sledování a zarovnání, ale vyžaduje předdefinované značky, zatímco AR bez značek poskytuje větší flexibilitu v různých prostředích, ale spoléhá na přesnost dat senzoru pro polohu a orientaci. Mnoho aplikací AR může obsahovat prvky obou typů a vytvářet tak všestranné uživatelské prostředí

# 3 Průmyslová data

V dnešním průmyslovém prostředí se data stala hnací silou pokroku. S příchodem Průmyslu 4.0 se tradiční průmyslové procesy spojují s digitálním světem, a vzniká tak revoluce poháněná průmyslovými daty. Tato data zahrnují širokou škálu informací generovaných ve výrobních závodech, dodavatelských řetězcích, energetických sítích a dalších. Zahrnuje datové toky vycházející ze senzorů zabudovaných ve strojích, telemetrii z výrobních linek, sledování provozních parametrů v reálném čase a složitost systémů, které řídí moderní podniky.

Pokud se podnik označí jako lidské tělo, tak data jsou krví, která pohání informované rozhodování, podporuje strategie prediktivní údržby a zajišťuje provozní efektivnost. Představují tak klíč k optimalizaci procesů, zmírnění rizik a dosáhnutí vyšší efektivity. Průmyslová data umožňují přejít od reaktivních k proaktivním modelům, kdy poznatky získané z historických dat a analýzy v reálném čase zajistí lepší chod všech procesů.

Samotný objem a složitost dat však představuje výzvu. Digitální „záplava“ vyžaduje strategie správy dat, robustní opatření kybernetické bezpečnosti a pokročilé analytické schopnosti pro získání užitečných poznatků. Cesta od surových dat k využitelným informacím vyžaduje nejen technologickou zdatnost, ale také myšlení, které využije transformační potenciál spojený s využitím těchto dat.

## 3.1 Senzory

Výše byla použita analogie k lidskému tělu, kterou lze použít i zde. K životu člověka nestačí mít pouze mozek. Aby mozek mohl zpracovávat informace, je nutné je nejdříve nějak získat. Všechny vjemy získává mozek prostřednictvím smyslů – hmat, čich, sluch, zrak a chuť. Podobně poskytují spínače a senzory informace o prostředí, které má řídicí jednotka zpracovat. Výsledkem tohoto zpracování je určení činností k provedení, které bude provádět aktuátor, nebo také akční člen.

Termín sensor, nebo také čidlo či snímač, se používá pro vstupní zařízení, které poskytuje použitelný výstup v reakci na stanovený fyzický vstup. Například termočlánek je senzor, který převádí teplotní rozdíl na elektrický výstup. [25]

Níže jsou uvedeny některé z běžnějších pojmů používaných k definování výkonu senzorů.

- Přesnost
  Rozsah, v jakém může být hodnota uvedená měřicím systémem nebo prvkem chybná. Například snímač teploty může mít přesnost ±0,1 °C. Chyba měření je rozdíl mezi výsledkem měření a skutečnou hodnotou měřeného množství.

- Rozsah proměnné systému

  Limit, mezi nimiž se může vstup lišit. Například snímač teploty odporu může být uveden jako snímač s rozmezím −200 až +800 °C.

- Citlivost
  Označuje změnu výstupu přístrojového systému nebo prvku systému v reakci na změnu množství měřeného daným množstvím, tj. poměrem výstupu/vstupu. Například termočlánek může mít citlivost 20 V/ °C, takže výstup 20 V pro každou změnu teploty 1 °C.

Na trhu jsou k dispozici doslova tisíce různých konstrukcí snímačů pro monitorování a řízení průmyslových procesů, takže není možné popsat všechny typy. Místo toho bude pro každý typ měření představeno několik různých technologií reprezentujících běžné procesní senzory.

### 3.1.1 Snímače teploty

Teplota je ze základních měřených veličin ve velkém množství procesů, a k jejímu měření lze použít různé sensory. Snímač teploty je zařízení, které shromažďuje informace o teplotě z prostředí a převádí je na konkrétní hodnoty.[25]

Termistory

Slovo termistor pochází ze zkráceného termického odporu. Odpor termistoru je funkcí okolní teploty. V omezeném rozsahu teplot je změna odporu $\Delta R$ termistoru úměrná změně teploty $\Delta T$: $\Delta R=\alpha\Delta T$, kde $\alpha$ je charakteristický teplotní koeficient termistoru. Teplotní součinitel, který má jednotky $\Omega/°C$, může být kladný nebo záporný v závislosti na složení konkrétního termistoru.

Kladné termistory, či pozistory, zkráceně PTC (Positive thermal coefficient) zvyšují svůj odpor s rostoucí teplotou. Při normálních teplotách mají velmi malý odpor, se zvyšující se teplotou se zvyšuje i vnitřní odpor, a snižuje se množství proudu, které je schopné protékat. Negativní rezistory, či negistor, zkráceně NTC (Negative thermal coefficient) svůj odpor se zvyšující se teplotou snižují.

Termočlánky

Funkce termočlánků je založena na objevu Thomase Seebecka z 1821, který ukázal že obvod vytvořený z vodičů dvou různorodých kovů generuje elektrický proud, když se jeden ze spojů zahřeje. Termočlánky jsou levné, robustní a malé a jsou ideální pro měření rychlých změň teplot. Je ale důležité mít na vědomí, že termočlánky měří teplotu mezi dvěma body, nikoli absolutní teplotu. Pro maximalizaci termoelektrického potenciálu byly vyvinuty různé slitiny kovů a pro zjednodušení kalibrace a měření byly definovány standardní termočlánkové spoje. Termočlánky typu J se skládají ze železných a měděno-niklových drátů a mají termoelektrický koeficient 51 ($\mu V/°K$) při pokojové teplotě; lze je použít až do 700 °C. Termočlánky typu K se skládají z nikl-chromových a nikl-hliníkových slitin a mají koeficient 41 ($\mu V/°K$) při pokojové teplotě; lze je použít při teplotách nad 1 000 °C.

Infračervené teploměry

Předměty, které mají vyšší teplotu než jejich okolí, vyzařují elektromagnetické záření; tuto skutečnost dokládá světlo vyzařované zářícími předměty, které jsou "červeně horké" nebo "bíle horké". Kvalitativní pozorování spočívá v tom, že objekty teplejší než přibližně 500 °C vyzařují matnou červenou záři; s rostoucí teplotou objektu se světlo stává jasnějším a také bělejším. Toto vyzařování se nazývá záření černého tělesa a je způsobeno tepelnou excitací elektronů, které vyzařují energii. Elektronická excitace a záření se zvyšují s teplotou objektu.

Dokonce i předměty, které mají teplotu mírně vyšší než pokojovou, vyzařují světlo; je infračervené, a proto ho nelze vidět okem, ale s vhodným detektorem lze toto světlo použít k měření teploty bez kontaktu s předmětem. Bezkontaktní měření je užitečné zejména tehdy, když teplota přesahuje užitečný rozsah termočlánků (například v pecích na výrobu oceli).

### 3.1.2 Snímače tlaku

Tlak je definován jako síla působící na jednotku plochy. U plynů a kapalin může tento tlak pocházet z hmotnosti kapaliny (na dně nádrže) nebo z kinetické energie spojené s tepelným pohybem molekul kapaliny. Jednotkou tlaku je pascal (Pa), který je definován jako 1 newton

na metr čtvereční. Měření tlaku v zařízeních se často vyjadřuje jako manometrický tlak, což je tlakový rozdíl převyšující okolní nebo atmosférický tlak.

Diafragma

Protože tlak je podle definice síla působící na jednotku plochy, lze ho měřit přímo měřením síly na kotouč o známé ploše. Tuhý kotouč lze použít, pokud je namontován na měchu, který umožňuje pohyb v závislosti na síle; v takovém případě lze sílu měřit přímo pomocí snímače síly. Běžnější konstrukce používá pružnou membránu nebo membránu, která se pod tlakem pružně deformuje. Po změření výchylky ji lze převést na sílu (a nakonec na tlak) pomocí mechanického modelu deformace membrány při rovnoměrném zatížení.

Tenzometry

Nejběžnější typ snímače tlaku využívá ke sledování deformace membrány tenzometry. Tenzometr je elektrické zařízení, jehož výstupní signál je úměrný velikosti tahové nebo tlakové deformace, která na něj působí vnějšími silami. Deformace je relativní nebo zlomková změna délky, takže například tyč o délce L, která se v tahu prodlouží o vzdálenost $\Delta L$, vykazuje deformaci $\sigma$, kde $\sigma=\Delta L/L$.

### 3.1.3 Snímače hladiny

Snímače hladiny se používají ke sledování množství kapaliny nebo pevných částic v nádrži nebo zásobníku. Výstupem snímače je tedy vzdálenost měřená buď od horní, nebo od spodní části nádrže. Jedním ze způsobů, jak odhadnout hladinu kapaliny v nádrži, je měření manometrického tlaku na dně nádrže. Mezi další možnosti patří:

Kapacitní snímače

Kapacitní snímač hladiny měří elektrickou kapacitu mezi elektrodou sondy a druhou elektrodou, kterou je často kovová stěna nádrže. Kapacita střední elektrody se mění v závislosti na výšce materiálu, přičemž začíná na nejnižší hodnotě, když je nádrž prázdná, a dosahuje maximální hodnoty, když je nádrž plná. Skutečná hodnota kapacity závisí na geometrii elektrody a nádrže a na relativní permitivitě materiálu.

Ultrazvukové a akustické senzory

Dalším přístupem je určit rozdíl mezi horní částí materiálu a plynem v prostoru nádrže. Protože se zvukové vlny odrážejí jinak v závislosti na hustotě nebo akustické impedanci, rozhraní mezi materiálem a prostorem v nádrži obvykle poskytuje dostatečný kontrast pro vytvoření odrazu (ozvěny). Frekvence zvuku může být v rozsahu lidského sluchu nebo může být ultrazvuková, ale v obou případech musí jít o puls relativně krátkého trvání.

### 3.1.4 Proximity sensory

Proximity sensory nebo také sensory blízkosti, detekují přítomnost či nepřítomnost objektů pomocí elektromagnetického pole, světla nebo zvuku. Existují různé typy, z nichž každý je vhodný pro určité prostředí. K dispozici jsou následující typy snímačů přiblížení:

Indukční senzory přiblížení

Jedná se o bezkontaktní senzor, který detekuje železné kovy, jako je uhlíková ocel, nerezová ocel a litina. Indukční snímač přiblížení se skládá z cívky, oscilátoru, detekčního obvodu a výstupu. Oscilátor vytváří oscilující magnetické pole, které vyzařuje kolem vinutí

cívky, jež je umístěno na povrchu snímače. Když se železný kov dostane do blízkosti magnetického pole, indukuje se na povrchu kovu malý proud (vířivé proudy). Tento malý proud mění vlastní frekvenci magnetického obvodu, což následně snižuje amplitudu kmitání. Detekční obvod sleduje amplitudu kmitání a spustí výstup z výstupních obvodů, když se kmitání sníží na dostatečnou úroveň.

Kapacitní senzory přiblížení

Jedná se o bezkontaktní senzor, který detekuje kovové i nekovové předměty ve formě prášku, granulátu, kapalin a pevných látek. Kapacitní senzory se chovají jako kondenzátory. Využívají elektrických vlastností kapacity a změny kapacity na základě změny elektrického pole v okolí aktivní plochy snímače. Kovová deska na snímací ploše snímače funguje jako první deska kondenzátoru a je elektricky propojena s vnitřním obvodem oscilátoru. Snímaný objekt funguje jako druhá deska kondenzátoru. Vnější kapacita mezi cílem a vnitřní deskou snímače tvoří část kapacity zpětné vazby v obvodu oscilátoru. Jak se cíl přibližuje k ploše snímače, oscilace se zvyšují, dokud nedosáhnou prahové úrovně a neaktivují výstup.

Fotoelektrické senzory přiblížení

Fotoelektrický senzor se skládá ze světelného zářiče (vysílače), foto přijímače a podpůrných obvodů. Slouží k detekci přítomnosti nebo nepřítomnosti objektů pomocí světelného zářiče a přijímače. Vysílač vysílá paprsek viditelného nebo neviditelného světla do detekčního přijímače. Existují tři typy fotoelektrických snímačů (1):

1. Průchozí paprsek: U tohoto snímače jsou vysílač a přijímač umístěny v odděleném krytu proti sobě. Vysílač vysílá konstantní světelný paprsek; k detekci dojde, když objekt procházející mezi vysílačem a přijímačem tento paprsek přeruší.

2. Retro reflexní: Zpětný reflexní snímač funguje podobně jako průchozí paprsek. Na rozdíl od průchozího paprsku jsou zde vysílač a přijímač umístěny ve stejném pouzdře a směřují stejným směrem. Vysílač vytváří světelný paprsek a vysílá jej směrem k reflektoru, který pak paprsek vychýlí zpět k přijímači. K detekci dochází, když je světelná dráha narušena nebo přerušena.

3. Rozptýlená odrazivost: U tohoto snímače jsou vysílač a přijímač umístěny ve stejném krytu a směřují stejným směrem. Vysílač vysílá konstantní paprsek světla, který se rozptyluje do všech směrů a vyplňuje detekční oblast. Když do této oblasti vstoupí cílový objekt, odkloní část paprsku zpět k přijímači. Cíl funguje jako reflektor. K detekci dojde, když se světlo odrazí od rušivého (nebo cílového) objektu.

### 3.1.5 MEMS snímač

Senzory MEMS jsou miniaturní zařízení, která integrují mechanické prvky, senzory, aktuátory a elektroniku na jediném čipu. Senzory MEMS jsou známé svými kompaktními rozměry, nízkou spotřebou energie a schopností měřit různé fyzikální parametry[26].

Mezi nejpoužívanější MEMs sensory patří:

1. Snímače tlaku: Snímače tlaku MEMS se běžně používají v průmyslových automatizačních systémech, automobilových systémech a lékařských přístrojích pro měření a monitorování tlaku v uzavřených systémech, jako jsou nádrže nebo potrubí.
2. Akcelerometry: Tyto akcelerometry se používají v letectví, automobilovém průmyslu a spotřební elektronice k měření zrychlení, náklonu, vibrací a pohybu, aby bylo možné detekovat pohyb a řídit stabilitu.

3. Gyroskopy: Používají se hojně v navigačních systémech, bezpilotních letadlech a zařízeních virtuální reality k měření úhlové rychlosti a orientace s velkou přesností.
4. Teplotní senzory: Tyto teplotní senzory se používají v systémech HVAC, průmyslových procesech a lékařských zařízeních ke sledování a regulaci teplot pro optimální výkon a bezpečnost.
5. Mikrofony: MEMS mikrofony lze nalézt v chytrých telefonech, nositelných zařízeních a audio systémech, které zachycují a převádějí zvukové vlny na elektrické signály pro čistou reprodukci zvuku.
6. Senzory plynu/průtoku: Tyto senzory se hojně využívají pro monitorování životního prostředí, vývoj lékařských přístrojů, systémy kontroly emisí v automobilech a průmyslové bezpečnostní systémy k detekci a měření různých plynů nebo průtoku kapalin, aby se zajistilo dodržování kvalitativních a bezpečnostních norem.

## 3.2 Metadata

Nejjednodušší definicí metadat je „data o datech“ nebo „informace o informaci. Pojem metadata se využívá v různých kontextech jako označení informací, které se týkají konkrétních věcí jako publikovaných materiálů, internetových stránek muzejních předmětů a další. Obecně řečeno, metadata obsahují informace, které popisují jakoukoli entitu nesoucí informaci. Tato definice ale plně nevystihuje pojem.

Sdružení pro katalogizaci[27] (CC: DA – Association Comitte on Cataloging: Description and Accses) definuje metadata jako strukturovaná, kódovaná data, která popisují charakteristiky entit nesoucích informace a jako taková umožnují funkce pro identifikaci, vyhledávání a správu entit.

S rozvojem výzkumu a aplikace metadat se jejich definice zpřesňovala: metadata se stala "strukturovanými informacemi, které popisují, vysvětlují, lokalizují nebo jinak usnadňují vyhledávání, používání nebo správu informačního zdroje"[28] a stala se "daty spojenými buď s informačním systémem, nebo s informačním objektem pro účely popisu, správy, právních požadavků, technické funkčnosti, používání a využívání a uchovávání"[29]. Obvykle se jako "metadata" věci často označují jednotlivé výpisy metadat a popisy nebo záznamy.

Velké množství standardů pro slovníky metadat bylo vytvořeno nebo navrženo komunitami v různých oblastech. Standardem je formální dokument, který stanovuje jednotná kritéria, metody, postupy a praktiky. Klíčovou složkou těchto standardů je soubor prvků, která určuje strukturu a sémantiku prvků. Například mezinárodní standard Dublin Core Metadata Element Set[30] (DCMES nebo "Dublin Core") definuje 15 základních prvků, které se používají k popisu informačních zdrojů. Metadatová komunita používá v různých kontextech řadu termínů, které označují sady metadatových prvků, kromě strojově zpracovatelných schémat, jejichž prostřednictvím jsou tyto prvky kódovány. Termíny "metadatové standardy", "sady prvků", "metadatové slovníky", "metadatová schémata", "slovníky vlastností", "datové slovníky" a "metadatové slovníky" mají malé rozdíly, ale v literatuře se používají zaměnitelně.

### 3.2.1 Historie

Vývoj metadat v éře internetu probíhal v první polovině 90. let 20. století v několika paralelních oblastech. Vědecká komunita se snažila najít způsob, jak organizovat rostoucí množství vědeckých dat, což vedlo k zavedení obsahových standardů pro digitální geoprostorová metadata a pokynů pro kódování a výměnu elektronických textů. Humanitní komunita zase vytvořila mezinárodní standardy pro kódování a popis uměleckých děl. Knihovnická komunita také vyvinula metadatové standardy pro popis a vyhledávání zdrojů. V

roce 1994 OCLC[31] (Online Computer Library Center) spustilo projekt pro katalogizaci webových zdrojů, který předznamenal vznik Dublin Core, jednoho z nejvýznamnějších metadatových standardů. Po finančním workshopu Dublin Core se metadatové hnutí rozšířilo do různých institucí, podniků a organizací. Od 90. let až do roku 2005 se vyvinulo mnoho dalších metadatových standardů v různých oblastech. Níže jsou uvedeny některé z mnoha standardů pro metadatové struktury:

- IPTC Photo Metadata Standards[32]
- Content Standards for Digital Geospatial Metadata (CSDGM)[33]
- Guidelines for Electronic Text Encoding and Interchange (TEI Guidelines)[34]
- Encoded Archival Description [35]
- Dublin Core Metadata Element Set (DCMES, nebo také DC)[30]
- Darwin Core [36]
- Online Information eXchange (ONIX) [37]
- Learning Object Metadata (LOM)[38]
- Friend of a Friend (FOAF)[39]

Na konci 90. let docházelo k rapidnímu nárůstu projektů zaměřených na metadata. Webové stránky DCMI (Dublin Core Metadata Initiative) shromažďovaly seznam experimentů a projektů z oblasti metadat z celého světa. Hlavním důvodem šíření těchto projektů je skutečnost, že nejsou žádná omezení pro typ nebo množství zdrojů, které mohou být popisovány metadaty, ani pro počet překrývajících se metadatových standardů. Metadatové standardy jsou používány pro datové struktury a jejich rozsah se pohybuje od 15 do více než 500 prvků. Tyto standardy ovlivnily vývoj metadatových slovníků a v současnosti existuje přibližně 470 takových slovníků.[40]

V roce 2011 byl zaveden rozsáhlý metadatový slovník Schema.org[41], který byl vytvořen vyhledávači jako Bing, Google nebo Yahoo!. Tento slovník zahrnuje mnoho schémat umožňujících popsání a zviditelnění webových stránek na vyhledávačích. Schema.org se neustále rozšiřuje a má vliv na stále větší množství webových stránek ve všech oblastech. V posledních dvaceti letech došlo ke značnému rozvoji metadat a výzkumu v této oblasti. Vznikly národní a mezinárodní digitální knihovny jako NSDL, Europeana nebo DPLA, které sbírají metadata od poskytovatelů a poskytují je skrze své platformy.

S rozvojem sémantického webu, Linked Open Data a Big Data se metadata stala ještě důležitější. Metadata jsou považována za "chytrá data", která reflektují procesy organizace a integrace strukturovaných, polostrukturovaných a nestrukturovaných dat. Výzkum a implementace metadat se rozšiřuje o nové dimenze a zahrnuje nové koncepty. Data jsou dnes považována za "novou ropu" a jsou velmi cenná. Je potřeba je správně zpracovat a analyzovat, aby se získala jejich skutečná hodnota. V digitálních humanitních vědách jsou metadata vytvořená překřížením různých standardů považována za "chytrá data", která umožňují organizaci a integraci různých typů dat a zvyšují hodnotu velkých dat.

Celkově lze říct, že vývoj metadat se neustále rozšiřuje a vyvíjí ve všech oblastech. Metadatové projekty, standardy a slovníky se stále vyvíjejí a mají významný dopad na organizaci a integraci dat v digitálním světě.

### 3.2.2 Typy Metadat

Definování typů metadat závisí na kontextu a aplikační doméně. Gettyho výzkumný institut vydal publikaci Introduction to Metadata[42]: a zpřístupnil jeho elektronickou verzi na internetu. Její kapitola nazvaná "Setting the Stage" identifikuje typy metadat a jejich funkce jako:

- Administrativní metadata

  Metadata používaná při správě a administraci sbírek a informačních zdrojů (příklady zahrnují informace o akvizici, sledování práv a reprodukce, zákonné požadavky na přístup a informace o umístění).

- Popisná metadata

  Metadata používaná k identifikaci a popisu sbírek a souvisejících informačních zdrojů (příklady zahrnují katalogizační záznamy, finanční pomůcky, specializované indexy a kurátorské informace).

- Metadata pro uchovávání

  Metadata související s archivací informačních zdrojů a různých kolekcí (příklady zahrnují dokumentaci fyzického stavu zdrojů, opatření přijatá k zachování fyzických a digitálních verzí zdrojů (např. obnovení a migrace dat) a změny, ke kterým dochází během digitalizace nebo uchovávání).

- Technická metadata

  Metadata týkající se fungování systému nebo chování metadat (příklady zahrnují informace o hardwarových a softwarových požadavcích, technické digitalizaci (např. formáty, kompresní poměry a škálovací postupy) a autentizační a bezpečnostní údaje (např. šifrovací klíče a hesla).

- Užitná metadata

  Metadata související s úrovní a typem využívání sbírek a informačních zdrojů (příklady zahrnují záznamy o oběhu, záznamy o fyzických a digitálních výstavách, využívání, znovuvyužití, vyhledávání a sledování uživatelů)[42].

Brožura vydaná Národní organizací pro informační standardy (NISO – National Information Standards Organization) rozděluje typy metadat do tří souhrnných skupin:

- Popisná metadata

  Popisují zdroj pro účely, jako je vyhledávání a identifikace.

- Strukturální metadata

  Označují, jak jsou složené objekty sestaveny dohromady.

- Administrativní metadata

  Poskytují informace, které pomáhají spravovat zdroj, a zahrnují technická metadata, metadata o správě práv a metadata o uchovávání[28].

Jako první typ na vysvětlení se nabízí tradiční a nejrozšířenější typ – Popisná metadat. Pro popis publikace je třeba zachytit základní informace – název, autora, klíčová slova, datum vytvoření nebo vydání a typ zdroje. Tento proces se obvykle řídí určitými standardy, které kontrolují, které údaje je třeba zachytit a jak by měly být tyto údaje zaneseny do počítačem čitelného formátu. Katalogizační záznamy, finanční pomůcky, indexy a kurátorské informace jsou především popisné.

Metadata se od tradičních katalogizačních produktů liší několika důležitými vlastnostmi. Povaha knihovní katalogizace se výrazně změnila s tím, jak se do knihovních fondů dostávala stále větší rozmanitost netradičních formátů zdrojů. Současná knihovnická metadata se stále více zabývají digitálními zdroji, jak je reflektováno v RDA (Research Data Alliance). Kromě správy nových, netradičních formátů zdrojů pocházejících z formálních distribučních kanálů publikací přebírají knihovny také roli hostitele nebo asistenta institucionálních repositářů dokumentů a datových sad.[43]

Zatímco změna úložišť spolu s technologickým pokrokem, jako je například posun hudebních formátů od gramofonových desek přes kazety a CD až po formáty mp3, vyžaduje nové způsoby popisu věcí, zásadnější změna se týká požadavku na objevování a zpřístupňování základních jednotek v těchto úložištích (např. samotných hudebních skladeb). Takovou poptávku lze pozorovat u všech ostatních typů zdrojů, zejména u akademických publikací, zpráv a datových souborů. Organizace těchto zdrojů a poskytování služeb pro jejich vyhledávání a využívání je složitý proces, který vyžaduje různé typy metadat pro různé účely a funkce.

Jak uvedl Gilliland[42], všechny informační objekty, bez ohledu na to, jakou mají fyzickou nebo intelektuální podobu, mají tři rysy:

1. obsah (co objekt obsahuje nebo o čem je),
2. kontext (aspekty kdo, co, proč, kde a jak souvisí s vytvořením objektu)
3. strukturu (formální soubor asociací v rámci jednotlivých informačních objektů nebo mezi nimi)

- všechny tyto aspekty mohou, a měly by být reflektovány prostřednictvím metadat. Stává se z nich tedy více než jen popis.

Technická metadata mají zásadní význam pro životní cyklus každého digitálního zdroje. Kromě popisu vlastností zdroje mohou být metadata použita také k popisu platformy a softwaru potřebného k vykreslení digitálního objektu. Potíže mohou nastat v případě archivace digitálních zdrojů pro použití v budoucnosti (jako informace uložené na disketách nebo zastaralé aplikace), ale technická metadata mohou poskytnout vodítka o jejich vlastnostech.

Kromě toho metadata plní administrativní funkci samotných metadatových popisů: metadatový popis poskytuje informace o tom, kdy a jak byl metadatový popis vytvořen (kým, odkud a podle jakých norem), jaké technické údaje obsahuje a kdo má k uloženým metadatům přístupová práva. Všechny typy metadat využívaných najednou lze vidět na obrázku níže (Obrázek 3-1 Příklad Metadat). V nejjednodušším případě je každá stránka knihy naskenována do obrázku, který může dále vytvářet více filmů (obrázků a textových stránek) pro různé účely. Kromě popisných metadat musí být digitalizované stránky přehledně uspořádány v posloupnosti kapitol, oddílů a stránek pro potřeby čtenáře. Pro zachování správné posloupnosti tohoto digitálního zdroje jsou tedy zapotřebí strukturální metadata. Protože výsledkem digitalizovaného zdroje je často více souborů pro každou stránku, je třeba, aby technická metadata zaznamenávala informace o tom, jak byly stránky vytvořeny a jaká automatická nebo ruční úprava byla provedena. Některé výpisy technických metadat, jako je rozlišení, barevný režim, bitová hloubka, velikost filmu, rozměry, kompresní poměr a formát filmu každého obrázku, lze generovat automaticky. [44]

Metadat zabývající se správou práv jsou nejdůležitějším typem administrativních metadat. K digitálním zdrojům lze snadno přistupovat, kopírovat je, upravovat nebo mazat, což může následně vyvolat porušení autorských práv, přístupových oprávnění a licenčních pravidel. Metadata musí zaznamenávat informace o právech ke zdroji, které budou využívány pro administrativní úkoly a správu.

Metadata pro správu práv se starají o práva duševního vlastnictví knihy v původním formátu i v digitalizované podobě. Miniatury, obrázky připravené pro web a obrázky s nízkým rozlišením mohou mít práva a licence odlišné od práv a licencí k obrázkům s vysokým rozlišením. Metadata pro uchovávání by dokumentovala fyzické podmínky celku a částí fyzického zdroje a historii všech činností provedených s původními a digitalizovanými zdroji. Metadata o použití by uváděla informace o vyhledávání, oběhu a vystavování knihy. Společně tyto typy umožnují spravovat a organizovat elektronické zdroje, zajistit věrohodnost zdroje, archivaci a efektivní vyhledávání a využívání zdroje.

Užitná metadata mohou být zjištěna. Vydavatelé, sociální média a marketingové služby využívají takové údaje shromážděné na základě použití, například počet zobrazení, stažení, diskusí, recenzí, doporučení, sdílení nebo citací zveřejněných nebo zveřejněných položek. Amazon při doporučování knih využívá údaje, jako jsou „často společně zakoupené položky“, „zákazníci, kteří si tuto položku také zakoupili“, a průměrné hodnocení zákazníků (rozmezí hvězdiček). Popisy pomocí metadat doprovázející položku obvykle nejen ukazují, co je tato věc zač, ale také prozrazují, jak tato věc souvisí s jinými věcmi. Tvorba metadat může být přínosem pro modulované procesy, protože určité typy metadat mohou být vytvářeny různými poskytovateli nebo dokonce softwarem. Například administrativní metadata o právech a licencích mohou být dávkově zpracovávána s výchozími hodnotami a spravována odlišnou jednotkou než té, která vytváří popisná metadata. Technická metadata mohou být generována přímo počítačovými programy.


**Obrázek 3-1 Příklad Metadat**

## 3.3 Internet of Things

Díky prudkému rozvoji počítačových technologií byl zaznamenán rychlý nárůst počtu výrobků a předmětů, které spolu komunikují a jsou ovládány prostřednictvím internetu. Obohacení vztahů uvnitř lidské společnosti a mezi společností a životním prostředím vytváří základ pro přechod do nové etapy technologického rozvoje, která je charakterizována pojmem "internet věcí". Pojem "věc" je zde vykládán v nejširším slova smyslu. Může se jednat o předmět (např. nějaký druh výroby) nebo výrobek. Může být živá nebo neživá, ale měla by být spojen a komunikovat s digitálním světem prostřednictvím internetu a identifikována v čase a prostoru. Důležitými součástmi internetu věcí jsou již zmiňované CPS, v nichž jsou různé objekty propojeny.

V ekosystému IoT se poté, co objekt IoT odešle svá data do cloudu nebo databázového systému, data analyzují a obvykle se použijí v jednom ze tří možných scénářů[45]:

1. Analyzovaná data se používají ke zlepšení výkonu daného objektu nebo jiných objektů v systému v nereálném čase
2. Shromážděná data z fyzického objektu jsou analyzována v reálném čase a výsledky jsou odesílány do objektu s cílem zvýšit jeho výkonnost nebo pomoci objektu lépe se rozhodnout o jeho provozu
3. Data ze senzorů jsou analyzována v reálném čase a výsledky jsou odesílány jednomu nebo více objektům v síti, aby jim byly vydány příslušné příkazy týkající se jejich činnosti nebo aby těmto objektům pomohly učinit lepší rozhodnutí s cílem zvýšit jejich výkonnost.

# 4 Současná obdobná řešení

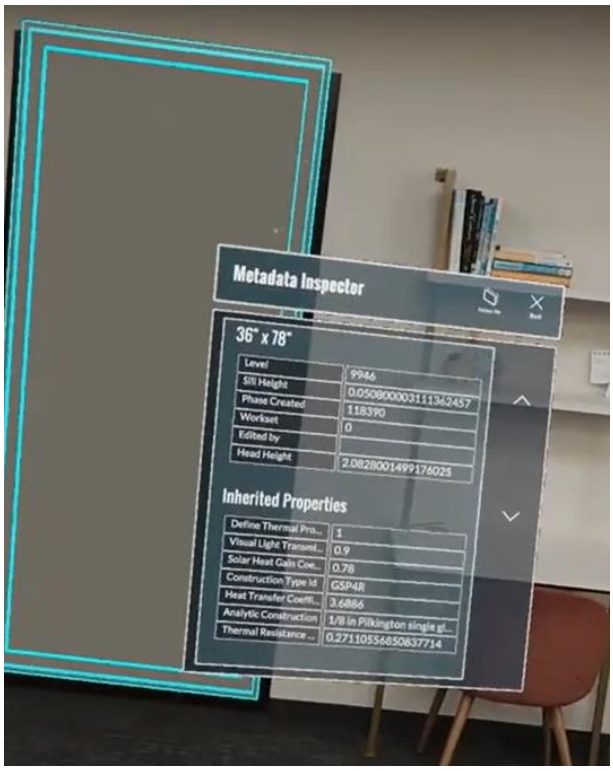
Metadata mají ve světě digitálních dat zásadní roli při poskytování informací o vlastnostech a atributech souborů. Existuje mnoho nástrojů a prohlížečů metadat, které umožňují uživatelům analyzovat, kontrolovat a spravovat metadata v různých typech souborů. ExifTool[46] je příklad všestranného softwaru, který pracuje s metadaty napříč různými formáty souborů jako obrázky, videa, zvuky či PDF.

Existuje pestrá škála možností, jak pracovat s metadaty, z nichž každá je přizpůsobena specifickým potřebám a preferencím. Dalšími příklady mohou být:

- Adobe Bridge – aplikace pro správu digitálních prostředků vybavenou robustním prohlížečem metadat, který umožňuje je organizovat, upravovat a publikovat.[47]
- MediaInfo – multiplatformový nástroj určený k poskytování podrobných technických informací o mediálních souborech. [48]
- FOCA (Fingerprinting Organizations with Collected Archives) - Specializovaný nástroj, který se zaměřuje na analýzu metadat v dokumentech, jako jsou soubory PDF, a pomáhá uživatelům získávat informace pro bezpečnostní účely.[49]

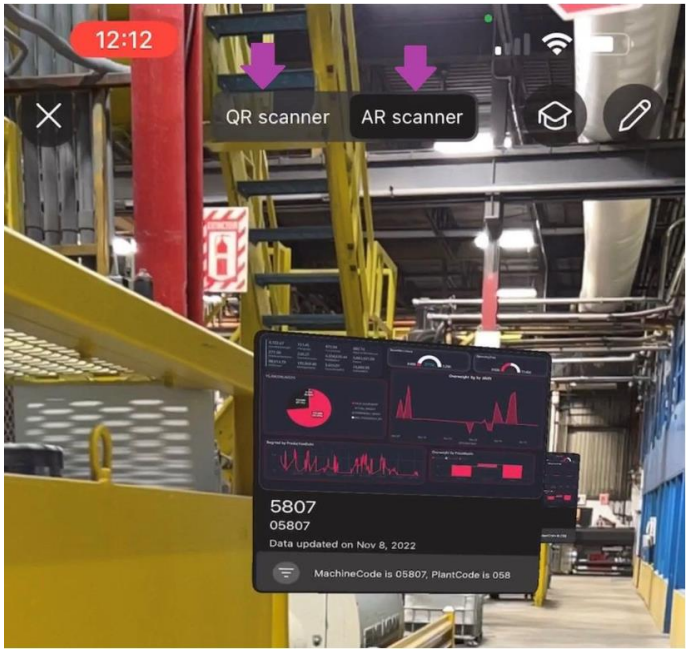
Pokud se ale přesune pohled směrem k augmentované realitě, tato škála se velmi rychle vytratí. V tuto chvíli neexistuje nástroj přímo dedikovaný vizualizaci průmyslových metadat, ale existuje několik aplikací, které mají funkci (do určité míry) zobrazit metadata. Všechny se zatím potýkají s jedním společným problémem – uživatelsky příjemné prostředí. Výzva spočívá v plynulém začlenění metadat do reálného kontextu. Pro pozitivní uživatelský zážitek je nezbytné zajistit, aby se metadata zobrazovala relevantním a nerušivým způsobem, aniž by uživateli bránila ve výhledu na fyzické prostředí.

Prvním řešením je Argyle Build[50], AR aplikace pro stavební dělníky, která pomáhá zefektivnit práci a dodržet časový plán. Aplikace vizualizuje BIM na staveništi a umožňuje interagovat s vytvořeným prostředím, například pomocí checklistů, a následně zasílá tyto informace do databáze, ze kterých se dále vytvářejí reporty. Lze přepínat různé prvky v BIM a filtrovat tak vše co se zobrazuje. Aplikace je k dispozici jako plugin pro aplikace Revit a Navisworks od společnosti Autodesk. Vizualizace metadat je v tomto případě limitovaná na pouhé zobrazení, jak je vidět níže.




**Obrázek 4-1 Metadata Inspector aplikace Argyl Build**

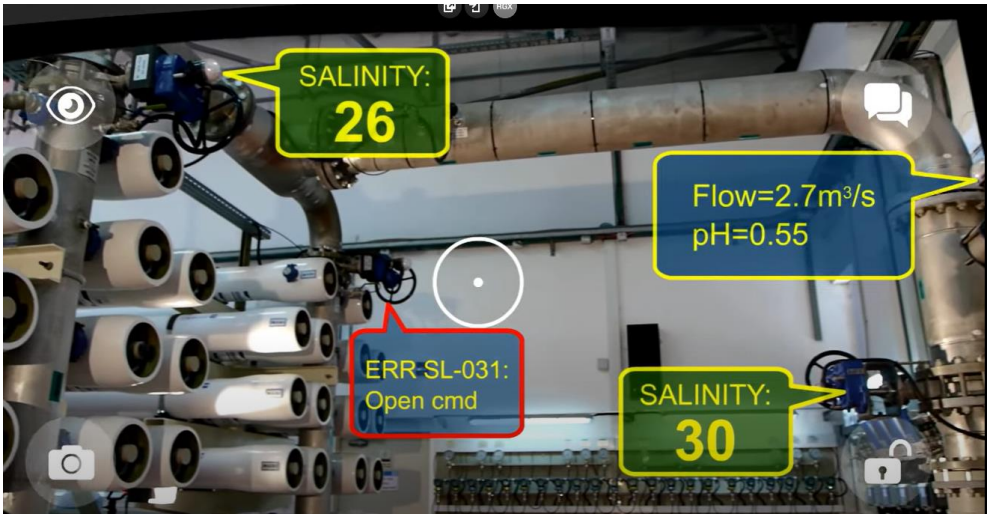
Jako jedno z možných řešení se jevil modul Data in Space[51], který kombinoval nástroje Power BI a Azure Spatial Anchors. Power BI, vyvinutá společností Microsoft, je služba pro podnikovou analýzu, která poskytuje interaktivní vizualizace a funkce business intelligence s rozhraním dostatečně jednoduchým pro koncové uživatele, aby si mohli vytvářet své reporty a panely. Umožňuje uživatelům připojit se k různým zdrojům dat, transformovat surová data do smysluplných poznatků a sdílet tyto poznatky v rámci celé organizace. Power BI se široce používá k analýze dat, vytváření reportů a rozhodování v podnicích všech velikostí. Bohužel se Tým Power BI rozhodl ukončit podporu funkce Data in space, která byla uvolněna jako náhled v aplikaci Power BI IOS, na základě blíže nespecifikovaných důvodů.



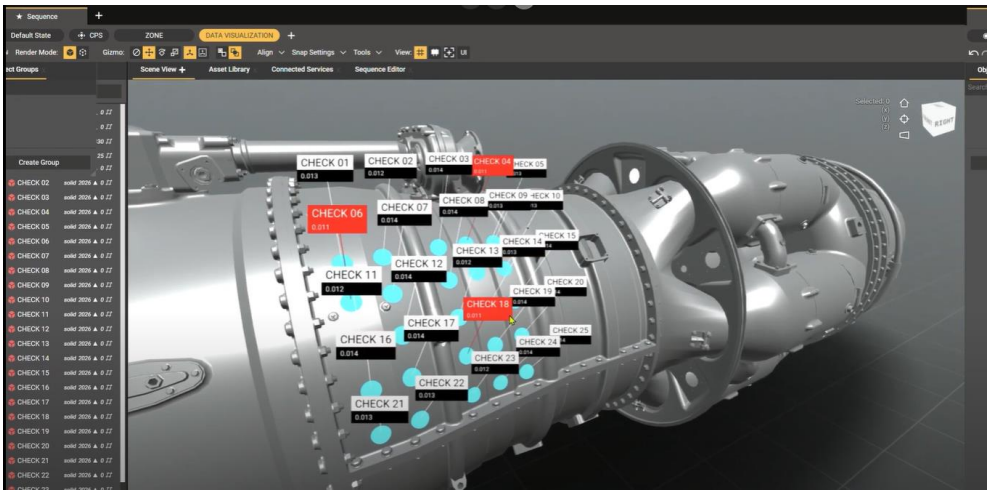

**Obrázek 4-2 Ukázka modulu Data in Space**

Posledním zmíněným řešením v rámci AR jsou dvě aplikace, které plní velmi podobnou funkci. Těmito aplikacemi jsou Help Lightning Fieldbit (HLF)[52] a Scope AR[53]. Obě aplikace slouží ke vzdálené podpoře, rozdílem je, že HLF se více soustředí na pomoc při údržbě a Scope na výcvik nových zaměstnanců. Obě aplikace ale poskytují požadovanou funkci – zobrazení dat spojených s konkrétním objektem v rámci rozšířené realitě.

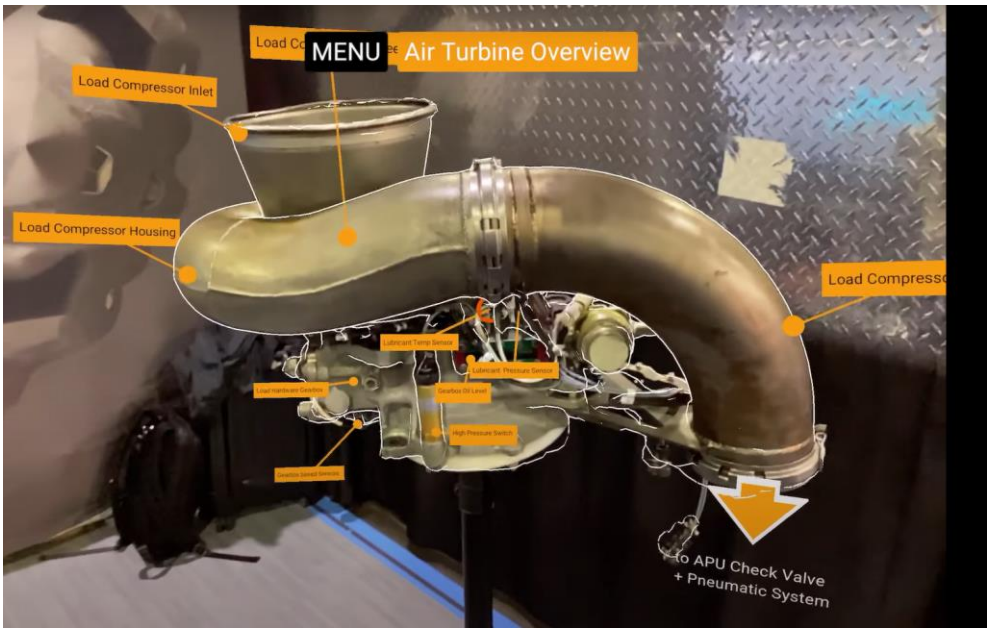
HLF poskytuje i další funkce, jako je navigace a propojení rozšířené reality se znalostními položkami (například videem, soubory PDF atd.). Princip funkce znalostních položek spočívá v tom, že si můžete prohlédnout nejen online data spojená s konkrétním strojem/zařízením, ale máte také snadný přístup k jeho manuálu apod. Princip navigační funkce spočívá v tom, že jakmile se nacházíte na scéně, můžete vyhledat konkrétní přístroj nebo senzor (dříve spojený s AR NODE). Aplikace uživatele navede k danému přístroji.[54]




**Obrázek 4-3 Ukázka aplikace Help Lightning Fieldbit**




**Obrázek 4-4 Ukázka historie v aplikaci Scope AR**

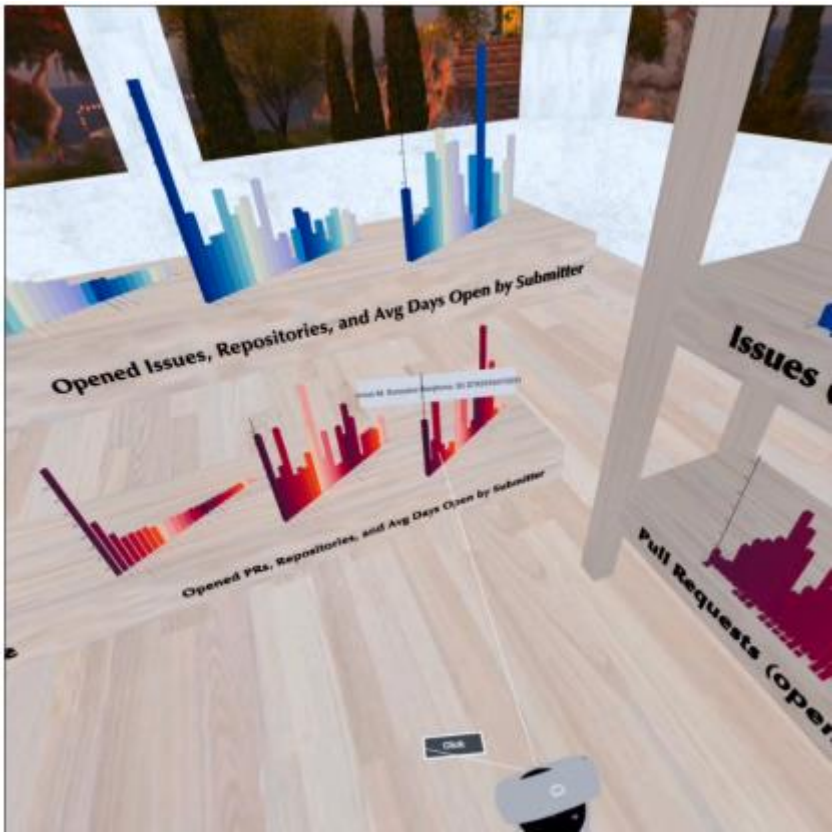



**Obrázek 4-5 Popis částí objektu v aplikaci Scope AR**

BabiaXR[55] je software s otevřeným zdrojovým kódem, který je určen k usnadnění experimentů s vizualizací dat rozšířené reality ve webových prohlížečích. I přesto, že se jedná o prostředí XR, dává článek náhled na možné zpracování a vizualizace dat. Sada nástrojů poskytuje komponenty pro vytváření komplexních vizualizací dat a jejich transformaci do scén vhodných pro provádění experimentů s lidskými subjekty. Podporuje různé metody analýzy, plánování a funkce pro provádění experimentů. BabiaXR je rozšiřitelný, založený na frameworku A-Frame JavaScript pro XR, a lze jej používat v jakémkoli zařízení XR podporujícím WebXR i v desktopových a mobilních zařízeních s omezenými možnostmi.

BabiaXR zjednodušuje proces plánování, provádění a vizualizace dat pro experimenty XR prostřednictvím řady funkcí a komponent. Ve fázi plánování experimentu mohou uživatelé vytvářet scény s různými vizualizacemi dat, libovolně je přizpůsobovat a flexibilně získávat data. Sada nástrojů poskytuje automatickou tvorbu prostředí a umožňuje bezproblémovou konfiguraci scén, vizualizací, prvků pro sběr dat a kamery pomocí komponenty babia-experiment. Nabízí také automatickou konfiguraci kamery, která přizpůsobuje nastavení kamery na základě zařízení, včetně ovladačů a interakcí.

Během provádění experimentu poskytuje BabiaXR uživatelům flexibilitu při výběru jejich fyzické přítomnosti, protože umožňuje zaznamenávat odpovědi na úkoly, demografické údaje a zpětnou vazbu pomocí mikrofonu zařízení, čímž se eliminuje potřeba osobní kontroly experimentu. Sada nástrojů rovněž usnadňuje měření času, záznam odpovědí, sledování polohy a natočení a automatické polohování úloh. Kromě toho BabiaXR nabízí řadu konfigurovatelných vizualizací, komponent pro dotazování do databází, možností zpracování dat a funkcí pro správu experimentů, které jsou postaveny nad rámcem A-Frame JavaScript pro vytváření scén WebXR.




**Obrázek 4-6 Data v prostředí BabiaXR**

Další článek[56] se zabývá zkoumáním a porovnáváním nástrojů založených na uživatelském rozhraní pro transformaci dat v prostředí stolních počítačů i virtuální reality (VR). Cílem studie je pochopit vliv interakčních technik a výpočetních prostředí na provádění základních operací transformace dat. Zdůrazňuje rostoucí trend poskytování nástrojů založených na uživatelském rozhraní, které snižují vstupní bariéru pro datovou vědu a omezují chybovost, čímž zpřístupňují datovou vědu netechnickým pracovníkům. Studie se zabývá potenciálními přínosy technik vestavěné a ztělesněné interakce, výhodami VR pro úlohy transformace dat a srovnáním rozhraní WIMP (okna, ikony, nabídky, ukazatel) s rozhraními založenými na gestech v prostředí stolních počítačů i VR.

Výsledky výzkumu naznačují, že VR vykazuje potenciální výhody pro strategické myšlení a podporu provenience během procesu transformace dat, ačkoli bylo zjištěno, že časový výkon je mezi desktopovým a VR prostředím podobný. Studie také odhalila, že interakce založené na gestech pro operace transformace dat na desktopu a ve VR byly považovány za intuitivní, flexibilní a snadno použitelné, přičemž prostředí VR bylo vnímáno jako vhodnější pro určité úkoly. Byly však vyjádřeny obavy ohledně omezeného prostoru a problémů s objevováním v prostředí stolních počítačů i VR, zejména při používání interakcí založených na gestech a problémů s funkčností ve VR.


**Obrázek 4-7 Transformace dat ve VR**

Další přístup, nyní již v rámci AR, popisuje článek využívající architekturu SOMARA[3][57], která se zaměřuje na podporu muzejních aktivit orientovaných na služby prostřednictvím spolupráce mezi mobilní platformou AR a rámcem webových služeb. Zdůrazňuje význam technologií mobilní rozšířené reality pro podporu digitálního zkoumání artefaktů v muzejních vzdělávacích scénářích a zmiňuje rostoucí potřebu aplikací, které usnadňují přístup k digitálnímu kulturnímu dědictví a jeho využívání. SOMARA podporuje realizaci virtuálních muzejních expozic založených na otevřené platformě, která umožňuje získávání a využívání obsahu prostřednictvím konzumace rozhraní API služeb třetích stran. Architektura zajišťuje úlohy interoperability, včetně získávání a využívání metadat, multimédií a 3D obsahu, a podporuje rozhraní API, jako jsou rozhraní poskytovaná Victoria and Albert Museum a Europeana.

Článek se zabývá implementací hlavní komponenty SOMARA, mobilního klienta AR, která zahrnuje využití Metaio AR SDK pro sledování a konfiguraci obsahu, rozšiřování objektů a personalizaci prostředí AR. Představuje pseudokódy a procesy pro sledování a konfiguraci obsahu, zpracování dokumentů XML a JSON a interakce, jako je výběr objektu a ukládání vybraného obsahu. Personalizace prostředí AR je zdůrazněna jako podpůrný modul v systému SOMARA, který umožňuje uživatelům vybrat preferovaný obsah a uložit jej do databáze založené na objektech pro pozdější použití. Tento personalizovaný obsah lze poté vizualizovat v prohlížeči AR Browser, což uživatelům umožňuje vytvářet a prohlížet si svá osobní prostředí AR mimo prostředí muzea.

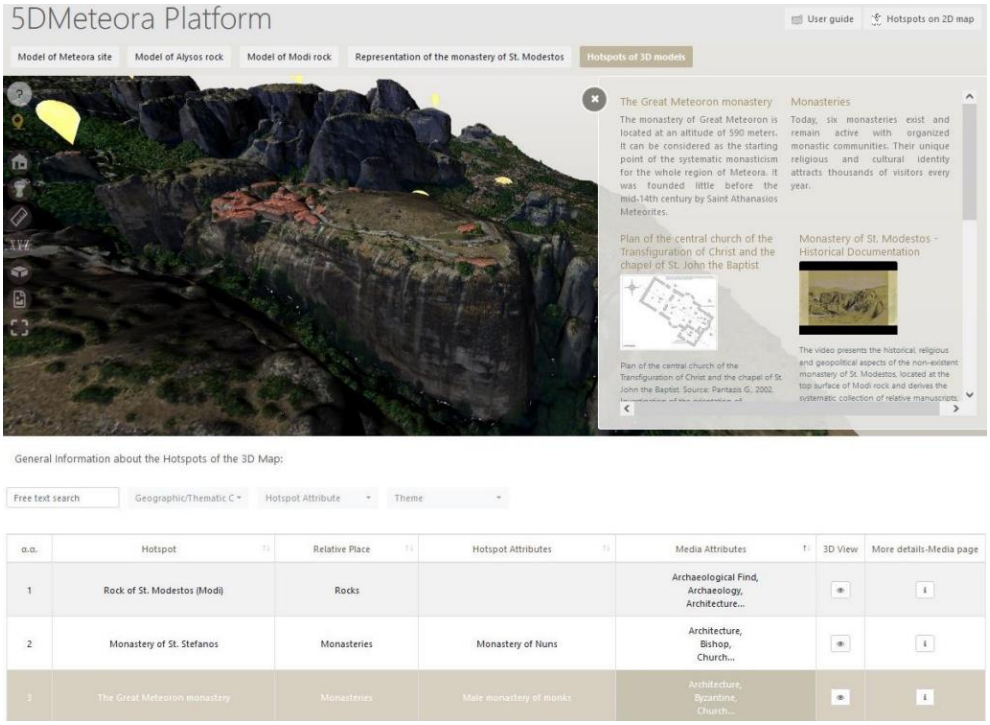

**Obrázek 4-8 Vizualizace pomocí AR aplikace**

Další článek[58] se zabývá rámcem 5dMeteora, webovou platformou určenou pro správu a publikování dat kulturního dědictví na webu, která integruje 3D vizualizaci, VR a AR. Cílem rámce je efektivně organizovat geoprostorová, multimediální a relační data a nabízet interaktivní 3D prohlížeč, VR a možnosti rozšířené reality. Zavádí také systém správy obsahu

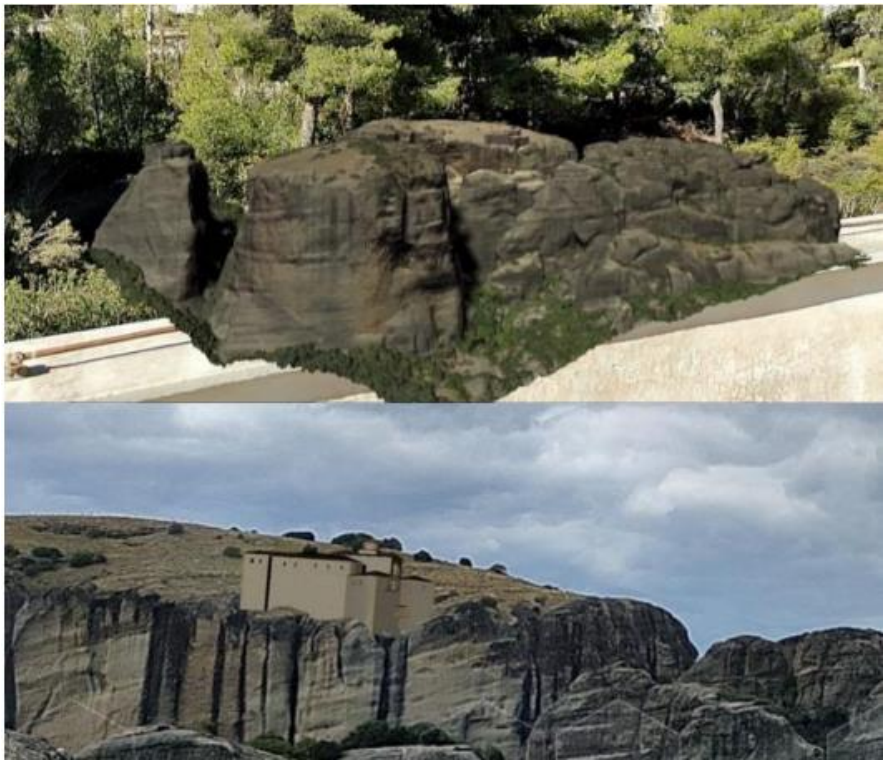
[3] Service Oriented Mobile Augmented Reality Architecture

(CMS) pro nahrávání, zpracování, publikování a aktualizaci obsahu, který umožňuje dynamickou správu obsahu a automatizaci operací s 3D daty. Platforma je postavena na systému správy databází MySQL a vyvinuta pomocí skriptů PHP, backendového Javascriptu a řadičů Ajax, což zajišťuje důvěrnost a integritu dat. Framework vychází vstříc potřebám odborníků v dané oblasti, kteří spravují multimodální data o kulturním dědictví, a poskytuje cenný nástroj pro šíření 3D a 2 D obsahu pro přizpůsobené zážitky z eXtented Reality (XR).

Framework je postaven na open-source technologiích stacku LAMP (Linux, Apache, MySQL, PHP) a nabízí uživatelsky přívětivý a intuitivní způsob vytváření, správy, publikování a úprav obsahu na platformě pro netechnické uživatele. Integruje také vědecké nástroje pro georeferencované modely a nabízí dynamickou scénu, sémantické obohacování a pokročilé dotazování a filtrování vyhledávacích formulářů napříč různými multimediálními sbírkami.




**Obrázek 4-9 Informace o lokaci v 5DMeteora**



**Obrázek 4-10 Zobrazení lokace pomocí AR**

# 5 Variantní přístupy

Tato kapitola je zaměřena na zodpovězení několika důležitých otázek:

- Na jaké platformě bude aplikace postavena?
- Jak bude zajištěna AR složka?
- Jak se budou data posílat?
- Kde se data budou skladovat?

Odpovědi na tyto otázky dají projektu směr a umožní prozkoumat možné přístupy k řešené problematice. Veškeré porovnání je založeno na porovnávacích nástrojích G2[59] a 6Sense[60], a na vlastní zkušenosti. Hodnocení je provedeno na stupnice 1-10, kde 1 je nejhorší skóre a 10 nejlepší. Např. pro cenu je tedy nejlepší varianta, když software je bezplatný, má tedy hodnotu 10.

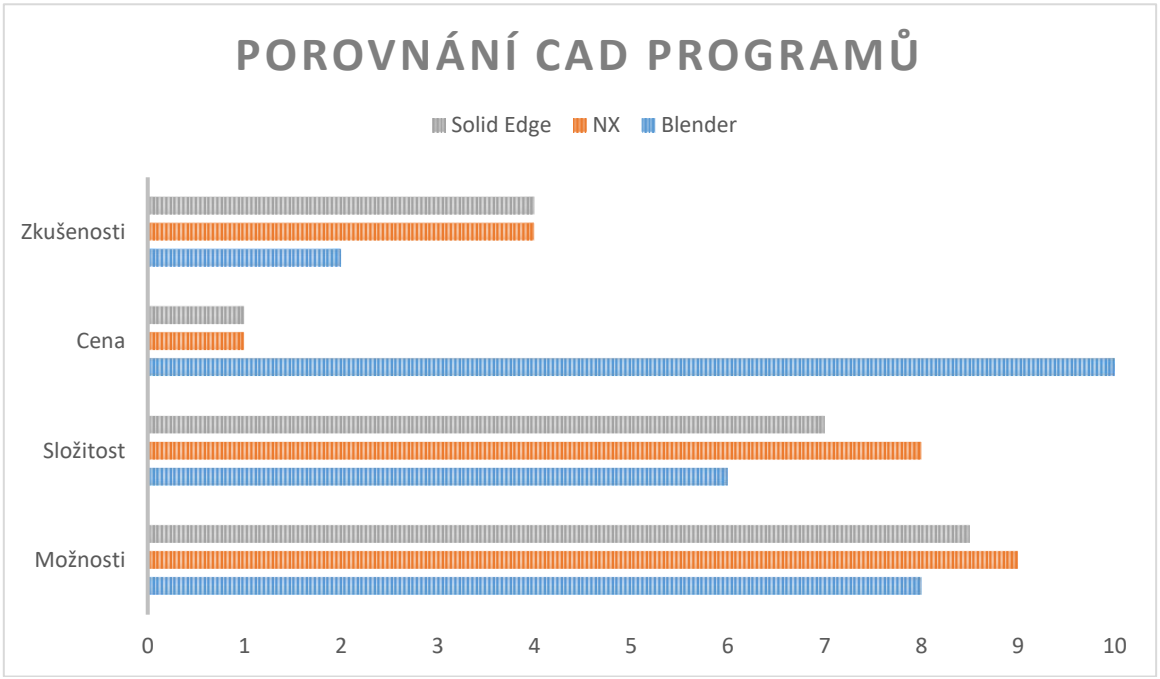
## 5.1 CAD

Pro hodnocení byli vybrány tři nástroje: NX a Solid Edge (dále jen SE) od společnosti Siemens a Blender. NX a SE jsou specializované programy určené pro profesionální vývoj produktů. Zaměřují se na vytváření přesných a funkčních 3D modelů pro výrobu. Blender je především 3D program, který vyniká v animaci, sochařství a VFX. Nabízí jisté možnosti, jak vytvořit CAD soubor, i přesto že to nejsou jeho přednosti

NX a SE nabízí funkce jako parametrické modelování, modelování sestav či poskytují inženýrské nástroje jako např. analýzu napětí, tolerancí a další výpočty. Blenderu tyt možnosti chybí, ale nabízí více možností v oblasti grafických úprav.

Co se týče ceny produktů, NX a SE nabízí bezplatný trial a studentskou verzi s limitovanými možnostmi. Blender je zcela bezplatný nástroj a poskytuje všechny funkce

Složitostí se zdají být všechny programy zhruba na stejné, s různými oblastmi, které uživatelé považují za lepší. Ke všem třem je možné vyhledat výukové programy a školící zdroje.




**Graf 1 Porovnání CAD programů**

Po zhodnocení byl vybrán program Blender, a to zejména z hlediska ceny a nastavení. Není nutné si zakládat licenci. Dodatečné funkce od NX a SE nejsou potřeba, jelikož se výsledný model bude používat pouze jako zdroj pro augmentaci.

## 5.2 Vývojová platforma

Jako vhodná vývojová platforma se nabízí Unity3D (dále jen Unity) a Unreal Engine (dále jen UE). Obě platformy jsou velmi populární a je na nich postavený nespočet aplikací.

Unity je známé svým uživatelsky přívětivým rozhraním a jednoduchostí, takže je přístupné i začátečníkům a nezávislým vývojářům. UE má na druhou stranu strmější křivku učení kvůli složitějšímu rozhraní a pokročilým funkcím.

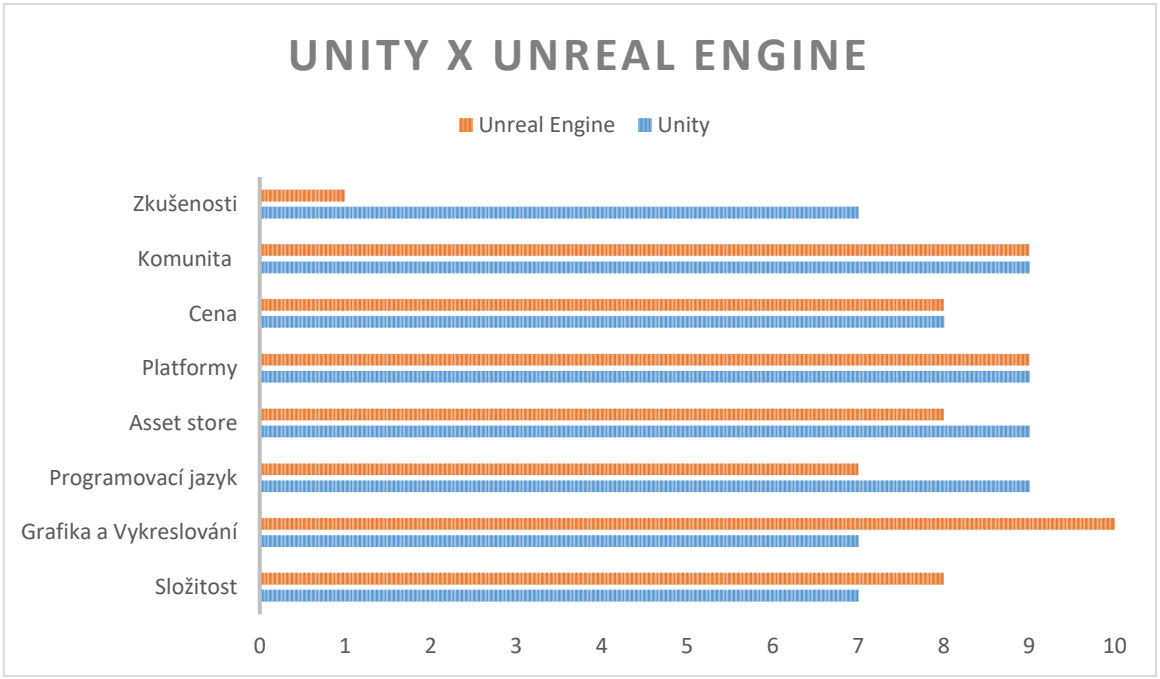
Unity poskytuje dobré grafické možnosti, ale může vyžadovat více úsilí k dosažení vysoce kvalitního vizuálního zpracování ve srovnání s UE, který je známý svou vysoce věrnou grafikou, pokročilým osvětlením a schopnostmi vykreslování, díky čemuž je preferovanou volbou pro vývoj AAA her a špičkových simulací.

Pro skriptování se používá v Unity především jazyk C#[61], který je široce používaný a pro začátečníky jednodušší na naučení. Primární skriptovací jazyk pro UE je C++[62], který nabízí vyšší výkon a flexibilitu, ale vyžaduje pokročilejší programátorské dovednosti.

Pro obě platformy existuje rozsáhlý Asset Store, ze kterého lze čerpat a usnadnit tak vývoj. Oba programy umožňují migrovat aplikace na různé platformy včetně PC, mobilních zařízení (iOS, Android), konzolí (PlayStation, Xbox), AR/VR a webu.

Unity nabízí bezplatnou verzi (Unity Personal) s omezenými funkcemi a omezením příjmů, zatímco Unity Pro vyžaduje model založený na předplatném s dalšími funkcemi a bez omezení příjmů. UE poskytuje svůj úplný zdrojový kód zdarma, ale u komerčních produktů si účtuje 5% licenční poplatek z hrubých příjmů po prvním milionu dolarů.

Obě platformy se mohou pochlubit velkou a aktivní komunitou s rozsáhlou dokumentací, fóry a výukovými programy, která poskytuje silnou podporu vývojářům.




**Graf 2 Unity x Unreal engine**

Byla vybrána platforma Unity, a to zejména z důvodu zkušeností s vývojem aplikací v Unity a s jazykem C#.

## 5.3 AR platforma

AR složka je hlavním cílem této práce, proto je výběr vhodného softwaru velmi důležitý. Hlavními parametry pro výběr budou trackování, možnosti softwaru, snadnost použití a kompatibilita s platformou Unity[63]. Pro využití v projektu jsou zvažovány následující nástroje: Vuforia, Wikitude, ARCore, ARFoundation a EasyAR.

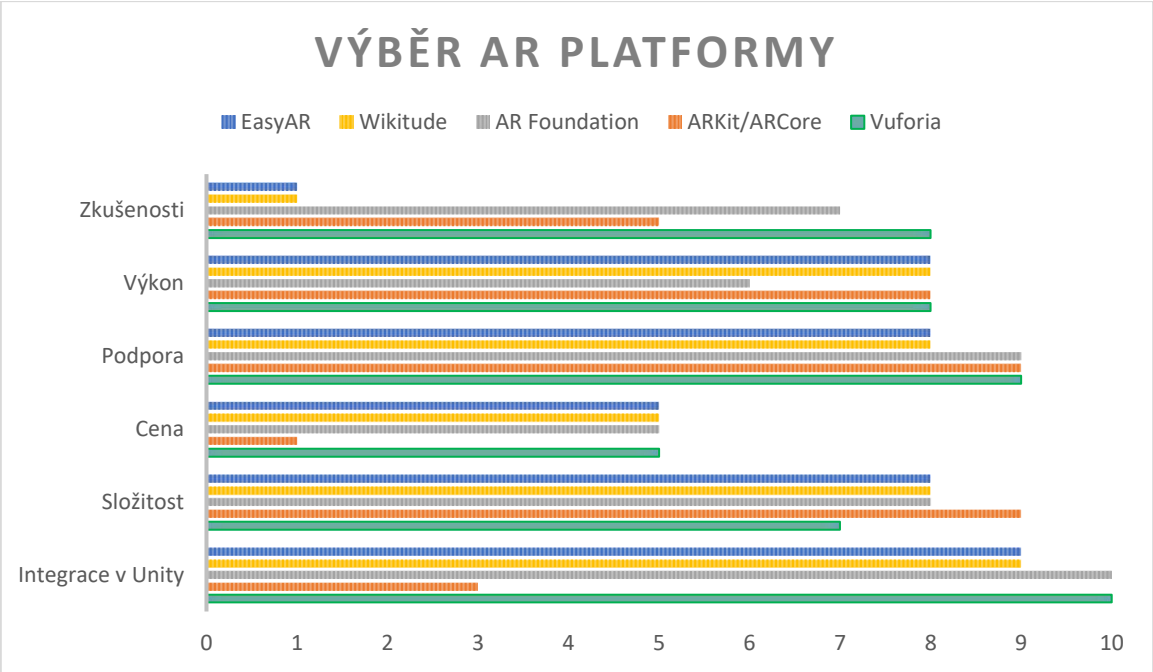
**Vuforia**[64] je jedním z nejpopulárnějších softwarů pro vývoj AR aplikací. Nabízí robustní funkce, jako je sledování na základě značek, rozpoznávání obrazu, rozpoznávání 3D objektů a podpora platforem iOS a Android. Vuforia poskytuje poměrně uživatelsky přívětivé rozhraní a obsáhlou dokumentaci, takže je přístupná jak začátečníkům, tak zkušeným vývojářům. Vuforia nabízí bezplatnou úroveň s omezenými funkcemi a placené plány s dalšími možnostmi, které jsou vhodné jak pro nezávislé vývojáře, tak pro podnikové projekty. Vuforia obecně nabízí velmi dobrý výkon, velmi užitečné nástroje a pomocí plug inu se jednoduše integrovat do Unity.

**AR foundation**[65] je nativní řešení pro tvorbu AR aplikací přímo v Unity. Poskytuje vývojové prostředí pro rozšířenou realitu s funkcemi, jako je podpora různých platforem, sledování 3D objektů, detekce rovin a integrace s dalšími prostředky Unity. Sama o sobě je ale velmi limitovaná a potřebuje dodatečná rozšíření, např. Google ARCore XR plugin pro funkci na Androidu.

**Wikitude**[66] nabízí funkce, jako je sledování na základě markerů, rozpoznávání obrazu, SLAM, geolokační AR a podpora platforem, jako jsou iOS, Android a chytré brýle. Wikitude lze integrovat s populárními vývojovými rámci, jako jsou Unity a Xamarin, což zvyšuje jeho flexibilitu a kompatibilitu. Nabízí flexibilní cenové plány vhodné pro nezávislé vývojáře, startupy i firemní zákazníky s možnostmi podle požadavků projektu a způsobu využití.

**EasyAR**[67] v základní verzi nabízí snadno použitelnou alternativu Vuforie. Nabízí funkce, jako je sledování na základě značek, rozpoznávání obrazu, 3D sledování objektů a sledování na základě SLAM na různých platformách. Podporuje vývoj pro systémy iOS, Android a Unity a nabízí tak kompatibilitu napříč platformami pro vytváření aplikací rozšířené reality. EasyAR nabízí bezplatné i placené plány s možnostmi přizpůsobenými pro individuální vývojáře, malé firmy i podnikové zákazníky. EasyAR je známý svou optimalizací výkonu a efektivní správou zdrojů, což umožňuje plynulé používání AR na mobilních zařízeních.

**ARCore**[68] alternativa **ARKit**[69], poskytují nativní funkce AR pro Android, respektive iOS, včetně sledování pohybu, porozumění prostředí a odhadu světla. Tyto platformy se bezproblémově integrují s příslušnými mobilními operačními systémy a nabízejí optimalizovaný výkon a přístup k funkcím specifickým pro danou platformu. Vývoj pomocí ARKit a ARCore obvykle vyžaduje znalost nativního vývoje mobilních aplikací pomocí Swiftu pro iOS a Javy/Kotlinu pro Android. Zakomponování v rámci Unity přestalo být podporováno, a je možné pluginy používat pouze ve verzích 2019 a starší.




**Graf 3 Výběr AR platformy**

Vybraným SDK pro vývoj aplikace byla Vuforia. Je to velmi všestranný a spolehlivý software, který je dobře podporován a nabízí další rozšíření, které ostatní aplikace neposkytují. Pro tento projekt byla vybrána základní verze, která je plně dostačující.

## 5.4 Databáze

Databáze je velmi důležitá při vývoji IoT aplikace, jelikož senzory generují obrovské množství dat. Tato data je nutné ukládat a organizovat pro budoucí analýzy, vizualizaci a rozhodování. Provádění analýz nad těmito daty je krokem k prediktivní údržbě strojů a odhalení vzorů a identifikace rizik. IoT aplikace často mají distribuovanou povahu, s mnoha zařízeními, která komunikují s centrálním systémem. Databáze umožňují efektivní a spolehlivou synchronizaci dat mezi těmito zařízeními a centrálním systémem.

Budou hodnoceny dvě relační databáze, MySQL a PostgresSQL, a jedna nerelační databáze, MongoDB.

MySQL[70] je známá pro své uživatelsky přívětivé rozhraní a širokou rozšířenost, díky čemuž je pro vývojáře relativně snadné začít. PostgreSQL[71] je zejména pro ty, kteří jsou obeznámeni s relačními databázemi, je však stále uživatelsky přívětivější. MongoDB[72] vyžaduje změnu myšlení, jelikož není postavená na tradičních relacích. Nerelační databáze často nemají předem definované schéma. To znamená, že mohou flexibilně ukládat různé typy dat bez nutnosti předem definovat jejich strukturu. Nerelační databáze jsou často vhodné pro aplikace, které pracují s nestrukturovanými nebo polostrukturovanými daty, jako jsou Big Data aplikace, či pro situace, kde je důležitá rychlost čtení a zápisu dat.

Co se týče škálovatelnosti, MySQL nabízí dobré možnosti s podporou clusteringu a replikace, vhodné pro rostoucí aplikace. PostgreSQL je vhodná v případě rozsáhlých nasazení, podporuje pokročilé funkce clusteringu a rozdělování. MongoDB je navržena s ohledem na horizontální škálovatelnost, takže je vhodná pro distribuované systémy a velké datové soubory.

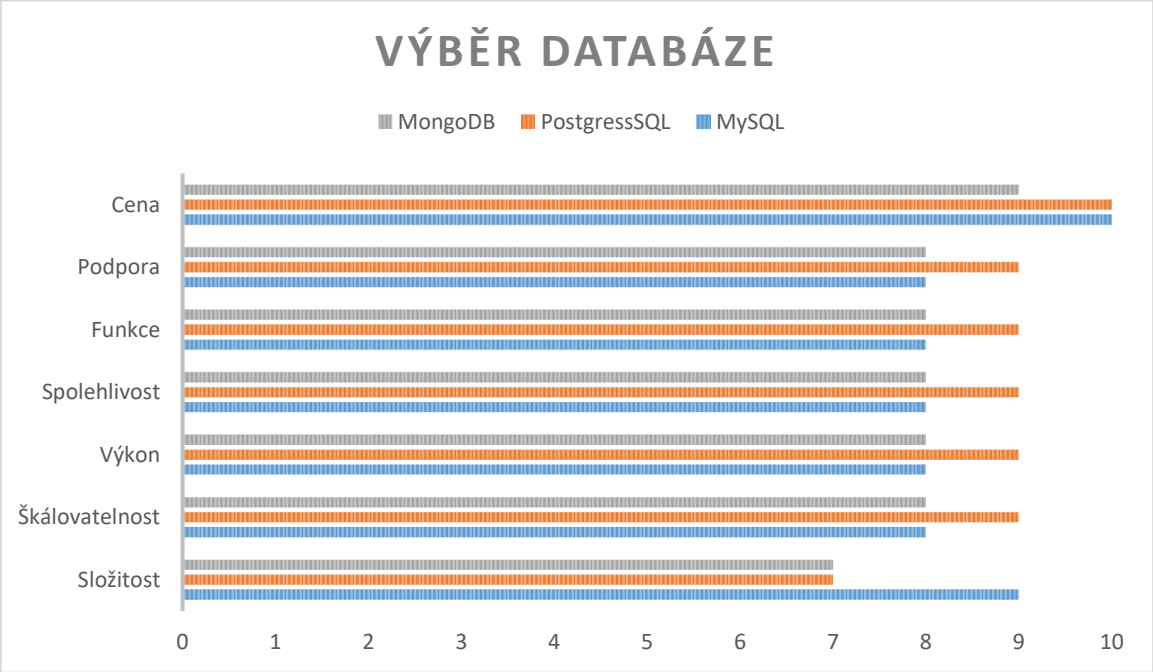
MySQL nabízí dobrý výkon pro většinu běžných případů použití, zejména při správném indexování a optimalizaci dotazů. PostgreSQL známý svým vynikajícím výkonem, zejména při

složitých dotazech a velkoobjemových transakcích. MongoDB poskytuje vysoký výkon při čtení a zápisu, zejména pro aplikace s velkými objemy nestrukturovaných dat.

MySQL je obecně spolehlivá, s dlouhou historií stabilních verzí a širokým průmyslovým využitím. PostgreSQL je vysoce spolehlivá, má robustní mechanismy obnovy po havárii a integrované funkce pro zajištění integrity dat. MongoDB je spolehlivá, s integrovanými funkcemi replikace a převzetí služeb při selhání pro vysokou dostupnost.

MySQL má nabízí solidní sadu funkcí pro správu relačních databází, včetně podpory transakcí, spouštěčů a uložených procedur. PostgreSQL známá pro svou rozsáhlou sadu funkcí, včetně podpory pokročilých datových typů, možností indexování a integrované podpory JSON. Stejně tak MongoDB má bohatou sadu funkcí pro dokumentově orientované databáze, včetně podpory dynamických schémat, geoprostorových dotazů a shardingu.

Všechny tři databáze těží z velké a aktivní komunity s rozsáhlou dokumentací a dostupnou podporou třetích stran. Stejně tak jsou všechny zdarma, opensource aplikace, které mají placené verze.




**Graf 4 Výběr databáze**

I přesto, že Postgress databáze má vše, co MySQL a více, byla vybrána právě MySQL. Tato databáze je velmi jednoduchá, snadno udržovatelná a v tomto projektu není třeba vysoké zabezpečení dat. Navíc tato databáze funguje jako pilotní, je to tedy jenom návrh, jak by finální databáze mohla vypadat.

## 5.5 Komunikace

**MQTT**[73] (Message Queuing Telemetry Transport): MQTT je odlehčený protokol pro zasílání zpráv navržený pro efektivní komunikaci mezi zařízeními v sítích s nízkou šířkou pásma, vysokou latencí nebo nespolehlivými sítěmi, které se běžně vyskytují v aplikacích internetu věcí. Řídí se modelem publish-subscribe, kdy klienti (zařízení) publikují zprávy do témat a ostatní klienti se k těmto tématům přihlašují, aby mohli zprávy přijímat. MQTT podporuje úrovně kvality služeb (QoS), které zajišťují doručení a spolehlivost zpráv, a používá architekturu založenou na zprostředkovateli, kde centrální zprostředkovatel MQTT usnadňuje směrování a doručování zpráv mezi klienty.

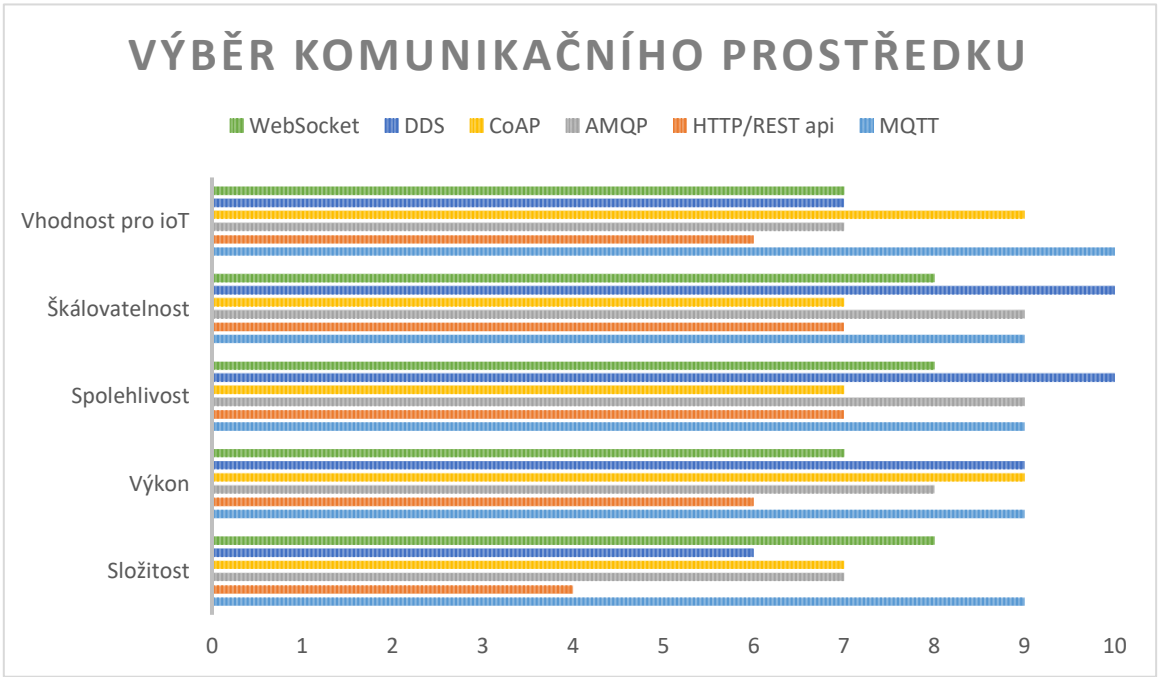
Rozhraní **API HTTP/RESTful**[74]: Komunikace založená na protokolu HTTP s využitím rozhraní RESTful API (Representational State Transfer) není sice specificky navržena pro internet věcí, ale je široce používána a chápána. Zařízení mohou komunikovat mezi sebou nebo se servery pomocí požadavků a odpovědí HTTP. Tento přístup je jednoduchý a široce podporovaný, ale pro určité aplikace nemusí být tak lehký nebo efektivní jako MQTT.

**AMQP**[75] (Advanced Message Queuing Protocol): AMQP je další protokol pro zasílání zpráv určený pro spolehlivou komunikaci orientovanou na zprávy. Ve srovnání s MQTT podporuje pokročilejší funkce, jako je řazení zpráv do fronty, směrování a zabezpečení. AMQP se často používá v podnikových systémech pro zasílání zpráv a aplikacích, kde jsou spolehlivost a pokročilé funkce kritické.

**CoAP** [76] (Constrained Application Protocol): je specializovaný webový přenosový protokol určený pro omezená zařízení a omezené sítě, jaké se vyskytují například v nasazeních internetu věcí. Je navržen tak, aby byl lehký a efektivní, takže je vhodný pro zařízení s omezenými zdroji a sítě s nízkou spotřebou energie. CoAP se řídí podobným modelem požadavek-odpověď jako HTTP, ale je optimalizován pro omezená prostředí.

**DDS**[77] (Data Distribution Service): DDS je protokol pro předávání zpráv typu publish-subscribe určený pro systémy reálného času a kritické systémy. Poskytuje funkce, jako je datově orientovaná publish-subscribe komunikace, řízení kvality služeb (QoS) a vestavěná podpora odolnosti proti chybám a škálovatelnosti. DDS se často používá v aplikacích, jako je průmyslová automatizace, letectví a obrana.

**WebSocket**[78]: WebSocket je komunikační protokol, který poskytuje plně duplexní komunikační kanály prostřednictvím jediného spojení TCP. WebSocket sice není speciálně navržen pro internet věcí, ale lze jej použít k vytvoření trvalých obousměrných komunikačních kanálů mezi zařízeními a servery. WebSocket se často používá v aplikacích vyžadujících aktualizace v reálném čase nebo interaktivní komunikaci.




**Graf 5 Výběr komunikačního prostředku**

Jednoznačným vítězem je zde protokol MQTT. Je zcela bezplatné, velmi jednoduché a nenáročné řešení, které dokáže propojit velké množství zařízení s velkou rychlostí.

# 6 Inicializace příslušných prostředí a frameworků

Tato kapitola je zaměřená na instalaci a nastavení všech vybraných částí vybraných výše, tedy:

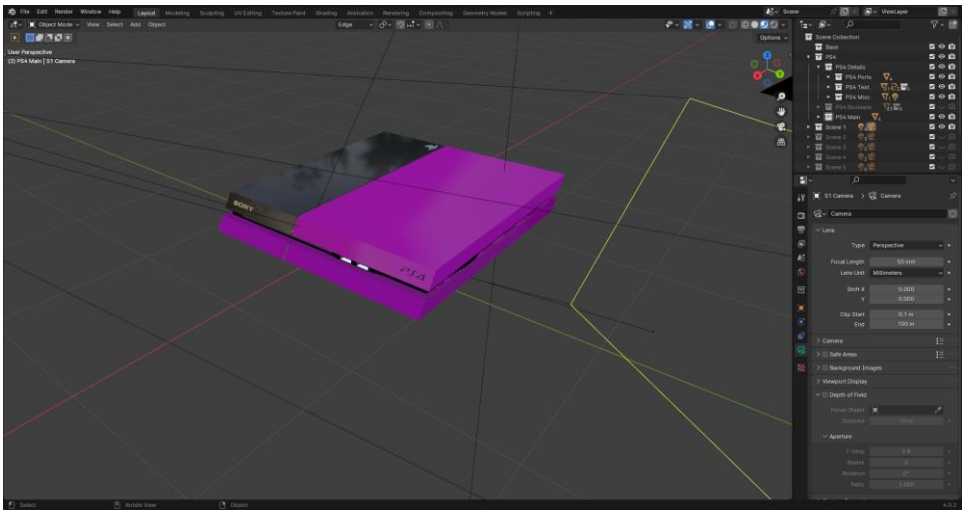
- CAD – Blender
- Vývojová platforma Unity
- AR platforma – Vuforia
- Databáze – MySQL
- Komunikace – MQTT

Následně bude popsán Node-Red, vizuální nástroj pro vytváření a řízení datových toků a automatizaci procesů.

## 6.1 CAD

Blender je vybraný CAD systém pro tvorbu modelu. Tento model, kterým je prozatím PlayStation 4, bude fungovat jako „Model Target“, neboli objekt, který bude vyhledáván kamerou a bude spouštět požadovanou augmentaci. Jedná se o nadstandardní řešení, které je třeba otestovat. V případě nekvalitních výsledků je možné využít standardní značky, která je velmi stabilní i v neperfektních podmínkách.

Pro využití programu Blender není třeba vytvářet uživatelský účet, pouze ho stáhnout z oficiální stránky blender.com a nainstalovat. Dále stačí vytvořit model a exportovat do požadovaného formátu. Velkým plusem je obrovské množství tutoriálů a návodů, jak s tímto programem pracovat.

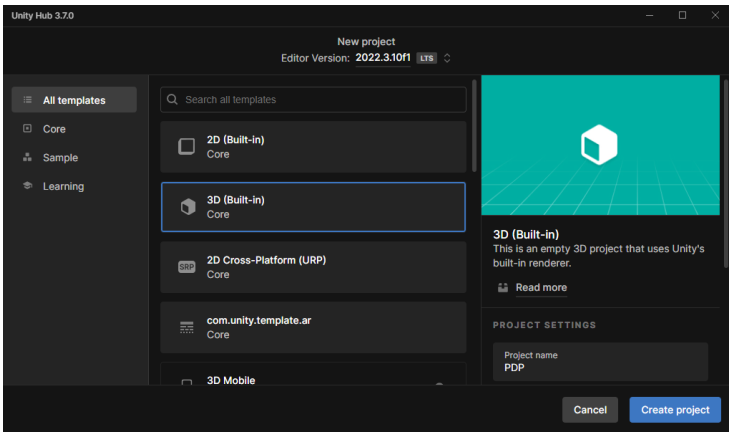


**Obrázek 6-1 PS4 model vytvořený v Blenderu**

## 6.2 Platforma Unity

Pro využití softwaru Unity je zapotřebí se registrovat na oficiálních stránkách unity.com. Vytvořený účet umožní stažení programu Unity Hub, který umožňuje využívání různých verzí Unity. Byla zvolena bezplatná verze Unity, která postačuje k vytvoření požadované aplikace.[79]

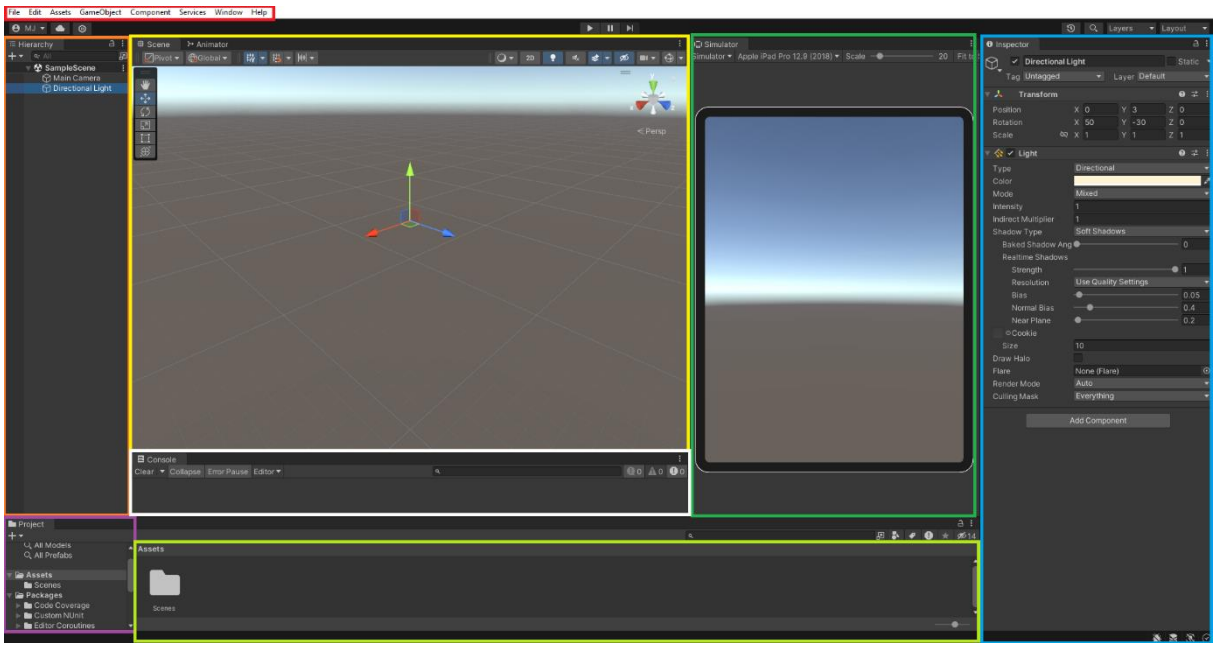
Po spuštění aplikace je nutné se přihlásit pomocí dříve vytvořeného účtu. Po přihlášení se vytvoření a pojmenuje projekt, jak je vidět níže.




**Obrázek 6-2 Vytvoření nového projektu**

Aplikace má několik oken, se kterými je nutné se seznámit a jsou k vidění níže.

- Toolbar – Je v obdélníku označeným červeně. Skládá se z 8 základních ovládacích prvků, které umožňují práci s programem jako takovým.
- Hierarchy – Je v obdélníku označeným oranžově. Obsahuje všechny prvky umístěné ve scéně, včetně zohlednění (ne)aktivních prvků a rodičovských vztahů.
- Scéna – Je v obdélníku označeným žlutě. Zobrazuje pozici objektů a umožňuje s nimi manipulovat.
- Simulátor – Je v obdélníku označeným zeleně. Ukazuje finální zobrazuje ve skutečnosti.
- Inspector – Je v obdélníku označeným modře. Zobrazuje podrobné informace o vybraném objektu a vlastnosti všech připojených komponent, jako např. skripty. Úprava těchto vlastností umožňuje upravit funkcionalitu objektů ve scéně.
- Console – Je v obdélníku označeným bíle. Zobrazuje všechny zprávy generované aplikací. Vypisují se sem všechny chybové zprávy, ale také zprávy pro lepší fungování aplikace, jako např. příjem informace, nalezení modelu apod.
- Project – Je v obdélníku označeným fialově. Zobrazuje strukturu všech složek v projektu. Po vybrání se zobrazí v okně označeným světle zeleně.



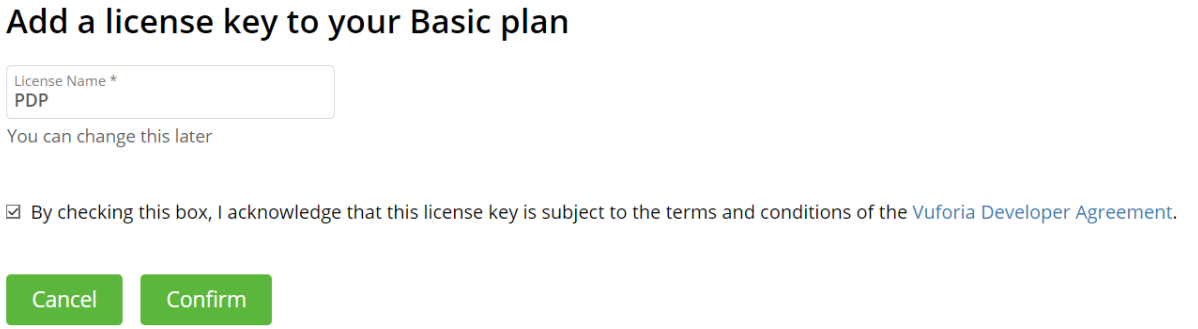
**Obrázek 6-3 Rozložení oken v Unity**

## 6.3 SDK Vuforia a Model Target Generator

Pro využívání Vuforie je opět nutné si vytvořit uživatelský účet z oficiálních stránek developer.vuforia.com. Po vytvoření účtu je možné stáhnout celkem 5 verzí programu. Jelikož se využívá platforma Unity, byla vybrána verze pro tento program. Po stažení je nutné tento balíček importovat do Unity *Assets -> Import Package -> Custom Package* a v otevřeném okně tento balíček vybrat. Tímto se do unity nahraje Vuforia Engine jak je vidět níže.


**Obrázek 6-4 Importování Vuforie do Unity**

Dalším nutným krokem pro využívání tohoto rozšíření je vytvoření licence na oficiálních stránkách. Byla vybraná základní licence, která je postačující pro tento projekt.




**Obrázek 6-5 Založení licence Vuforia**

Takto vytvořenou licence je třeba rozkliknout, a samotný klíč zkopírovat a vložit do Unity. Cesta k místě umístění je následující: v okně Hierarchy vytvořit AR kameru *Vuforia Engine -> AR Camera*, vybrat takto vytvořený objekt a v okně Inspector vybrat *Open Vuforia Engine configuration*, nakopírovat licenční klíč, jak je vidět níže. Nyní je možné plně využívat Vuforia Engine.


**Obrázek 6-6 Licenční klíč Vuforia v Unity**

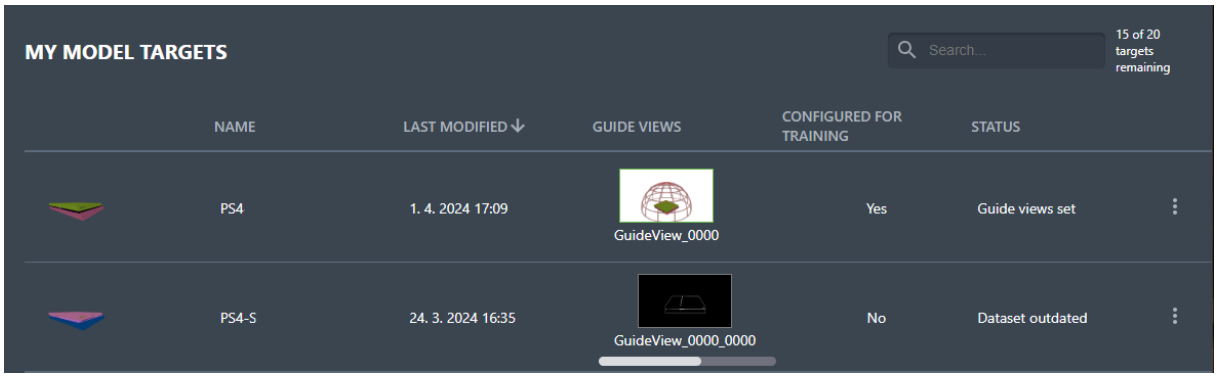
Model Target Generator (MTG)

MTG je klíčový nástroj pro transformaci CAD modelu, rozpoznávání a sledování objektů v rozšířené realitě. Tento nástroj je dostupný opět na oficiálních stránkách. Po stažení a přihlášení se do programu je třeba nejdříve vytvořit Model Target, vybrat CAD soubor a pojmenovat ho. Dále je nutné nastavit několik parametrů:

- Model Up Vector – upraví souřadnicový systém tak, aby byl model správně položený, jak má být.
- Model Units – nastavení jednotek. Může se stát, že se model nepřenese absolutně přesně. V tomto případě je nutné zkontrolovat poměr stran, a pokud poměry sedí, je možné rozměry upravit následně v Unity.
- Coloring – Zde jsou k dispozici dvě možnosti – Realistic a Non-Realistic appearance. První možnost se použije v případě, že model má stejnou texturu a barvu jako fyzický objekt který reprezentuje. V případě využití Advanced Model Targetu (vysvětleno dále) se bude vyhledávat pouze objekt, který má stejné barvy a textury jako model. Je tak možné mít více stejných fyzických objektů jiných barev, a ke každým mít jinou augmentaci.
  Non-realistic Appearance se využije, pokud model nemá stejné textury a barvy, nebo existuje více barevných variant objektu. Po vybrání této možnosti se obarví všechny plochy náhodnými barvami. Pro tento projekt byla vybraná tato varianta.
- Dalším oknem je Complexity, která poskytuje informaci o částech a počtu trojúhelníků, ze kterých se model skládá. Vyhodnotí se tak komplexita modelu. V tomto případě vyhodnotil program komplexitu modelu PS4 jako Good neboli dobrá. Znamená to, že trackování byl mělo být dostatečné, ale mohou nastat potíže s přesným překrytím.
- Optimize tracking je specializovaný mód, které může ovlivnit kvalitu trackování objektu. Jsou tři možnosti. Default je standardní mód, který aplikovatelný na většinu objektů. Doporučuje se komplexnější geometrii a objekty v dobrých podmínkách. Tento mód podporuje objekty, které se pohybují, a automaticky je stabilizuje. Pro tento projekt byla vybrána tato varianta.
  Low Feature Objects je mód, který se používá pro jednoduché objekty, objekty s jednolitou barvou, hladké a reflexní objekty. Příkladem mohou být kovové části jako lisované plechové díly. Je třeba aby trackování objekt zůstal statický.
  AR Controller je velmi specifický mód pro případy kdy trackování objekt je vždy v pohybu a je držen v ruce, jako např. AR/VR controller.
- Posledním krokem je rozhodnutí mezi Guide View a Advanced Model Target. V případě Guide View se vytvoří pohled na model, který je zapotřebí pro aktivaci

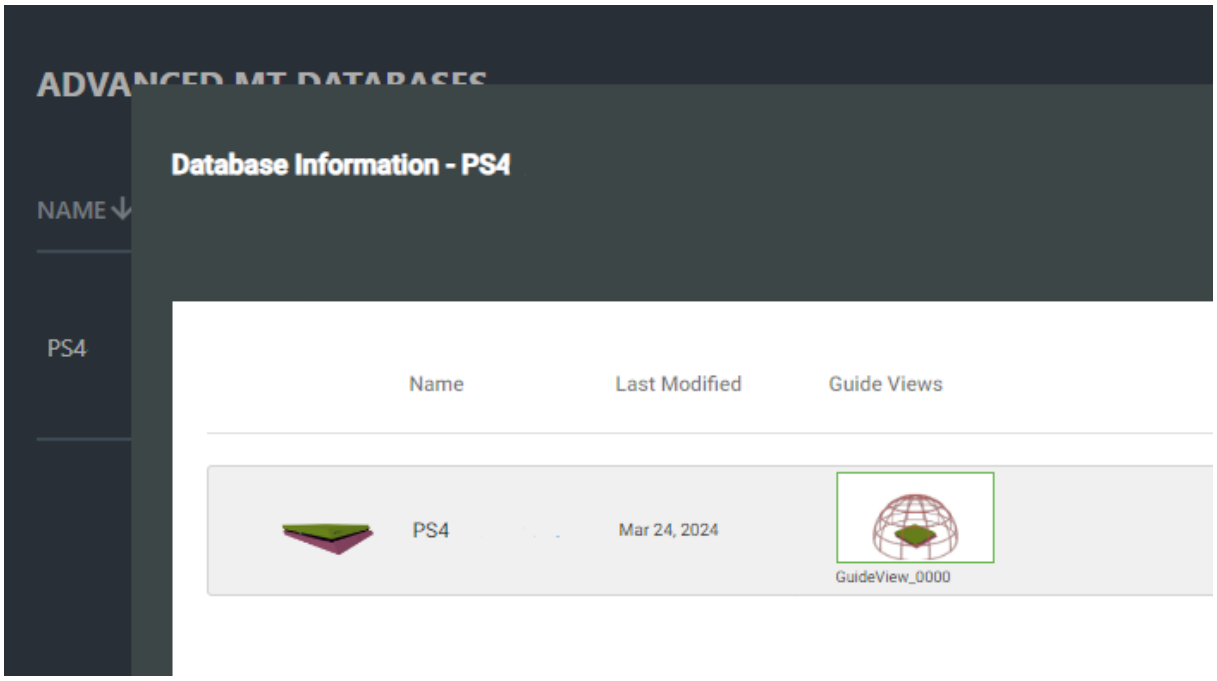
augmentace. Tato možnost byla testována, a výsledek byl velmi nekvalitní z hlediska vyhledávání objektu.

Druhá možnost nabídne vytvoření oblastí pro pozorování objektu jako např. 360° view, který umožní trackovat objekt ze všech úhlů. Pro projekt byl vybrán možnost Advanced Model Target, varianta Dome, která umožňuje sledování ze všech stran, kromě strany spodní, která v tomto případě není nutná.



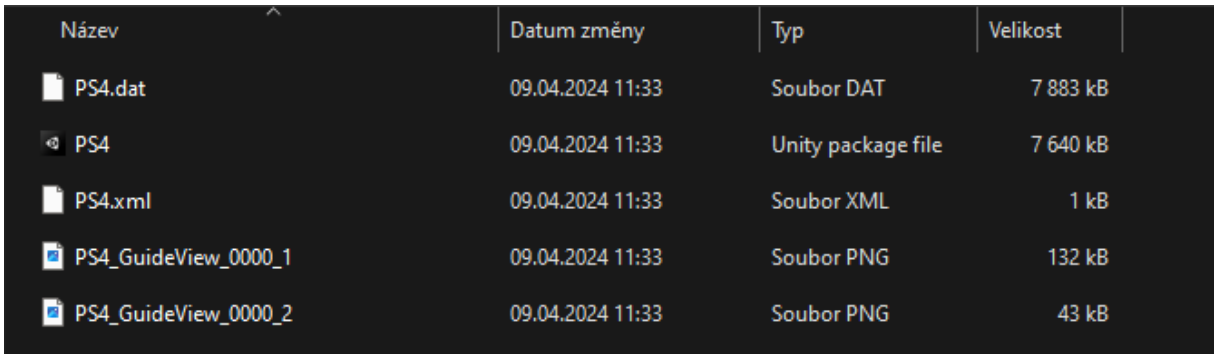

**Obrázek 6-7 Vytvořené modely v MTG**

Po vytvoření Model Targetu je nutné (pro případ Advanced Model Target) Advanced MT databázi. Tato databáze analyzuje pohledy a „vytrénuje“ se, aby bylo možné najít objekt ze všech požadovaných pohledů. Čas nutný pro toto trénování je závislý na komplexnosti objektu, počtu úhlů a nepochybně výkonem počítače. Trénink pro tento případ trval 2 hodiny.



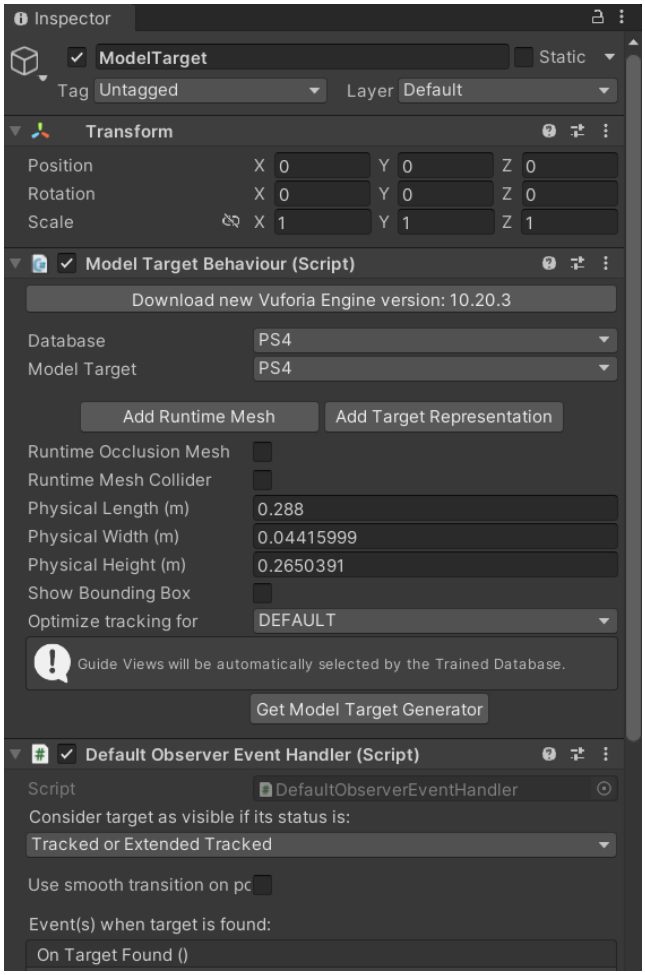

**Obrázek 6-8 Vytvořená databáze**

Po vytvoření databáze je nutné exportovat výsledek. Po kliknutí na tlačítko Export v MTG je nutné vybrat cílovou složku. Výsledek Exportu je k vidění níže. Tyto soubory je nutné importovat do Unity.



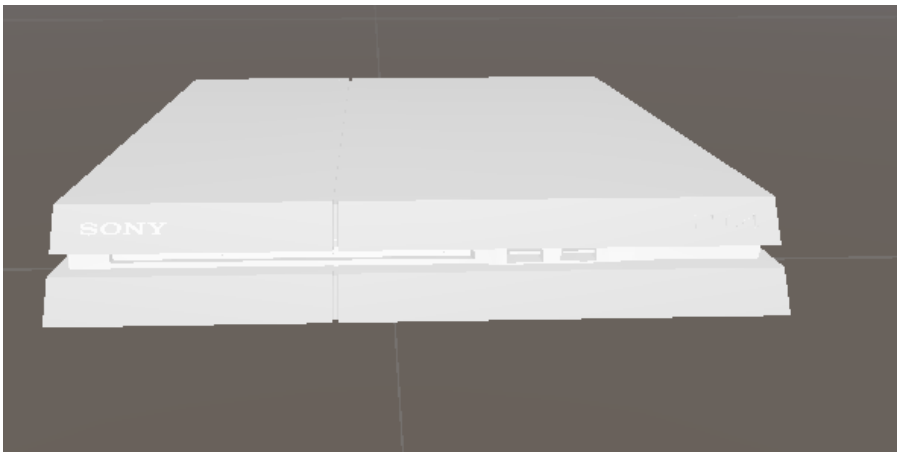

**Obrázek 6-9 Exportovaná databáze**

Dalším krokem je vytvoření objektu Model Target v okně Hierarchy v Unity *Vuforia Engine -> Model Target*. Níže je k vidění ModelTarget v okně Inspector. Zde je nutné nejdříve vybrat databázi, ze které se bude čerpat, a Model Target, který se bude vyhledávat. Je nutné také překontrolovat parametry modelu.



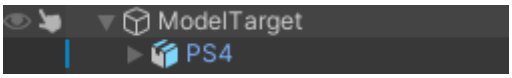

**Obrázek 6-10 Model Target Inspector**

Poté bude vytvořen mock up objektu ve scéně, který je k vidění níže, a představuje tak objekt spouštějící augmentaci.




**Obrázek 6-11 Mock up objektu PS4**

Po vytvoření Model Targetu je nutné přidat požadovanou augmentaci, která prozatím bude samotný model. Je nutné, aby veškerá augmentace měla nastavený Model Target jako rodiče, viz níže




**Obrázek 6-12 Model Target jako rodič augmentace**

Nyní je možné otestovat, zda lze trackovat požadovaný objekt a zdali augmentaci správně překrývá fyzický objekt.


**Obrázek 6-13 Trackování PS4 a doplnění augmentace**

## 6.4 MQTT

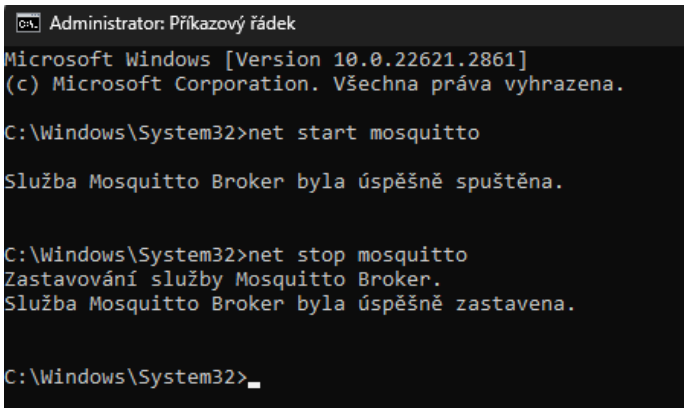
MQTT je klíčovou složkou pro komunikaci mezi jednotlivými zařízeními a přenos dat mezi klienty. Pro zprovoznění komunikace skrze MQTT je nutné provést tři kroky:

- Instalace brokeru
- Instalace MQTT Exploreru
- Propojení MQTT a Unity

Instalace brokeru

Prvním krokem je nainstalování MQTT brokeru, který bude zajišťovat komunikaci mezi zařízeními. Tímto prostředníkem může být Eclipse Mosquitto, Open Source broker, který právě využívá MQTT protokolu. Představuje to jednoduché, „lightweight“, řešení vhodné pro použití na všech zařízeních od plnohodnotných serverů až po lokální servery na laptopu. Program lze naistalovat ze stránky mosquitto.org.

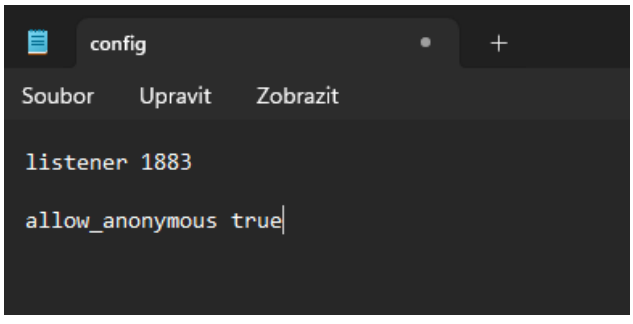
Ovládání služby mosquitto probíhá přes příkazový řádek (CMD). Důležité je CMD otevřít jako správce, jinak bude přístup odepřen. Služby brokeru se spustí zadáním následujícího příkazu – **net start mosquitto**. Pokud je třeba brokeru přestat využívat lze zadat – **net stop mosquitto.** Oba příkazy jsou k vidění na obrázku níže.




**Obrázek 6-14 Spuštění a zastavení služby Mosquitto**

Dále je třeba nastavit základní nastavení – zejména port a povolení připojení. Je třeba vytvořit soubor .txt přímo v místě instalace (pokud nelze umístění najít ve vyhledávači tak cesta bude nejpravděpodobněji – Tento počítač> OS (C:)> Program Files> mosquitto). Do tohoto textového souboru se zadá využívaný port (default port je 1883) a umožní se připojení neznámých zařízení (toto je pouze testovací program, takže otázka bezpečnosti se nemusí řešit).

Txt soubor bude vypadat takto:



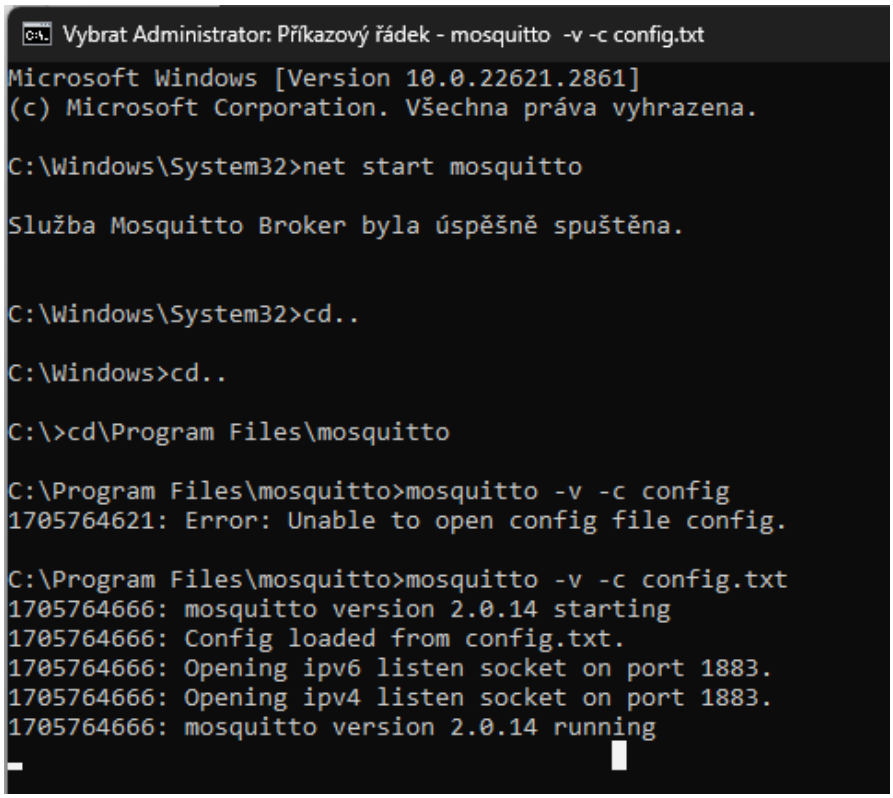

**Obrázek 6-15 Txt. soubor nastavení brokeru Mosquitto**

Dále je nutné napojit tento textový soubor. Postup je následující:

- Otevřít CMD jako správce
- Spustit mosquitto
- Odkázat se na disk C (pokud se složku na tomto disku nachází)
- Odkázat se na složku mosquitto
- napojit textový soubor (v tomto případě se soubor nazývá **config**)

(Pokud se složka mosquitto nachází jinde než v tomto případě (C:> Program Files> mosquitto) tak bude třeba zohlednit veškeré dodatečné složky či podsložky)

Celý postup je ukázán na obrázku níže:

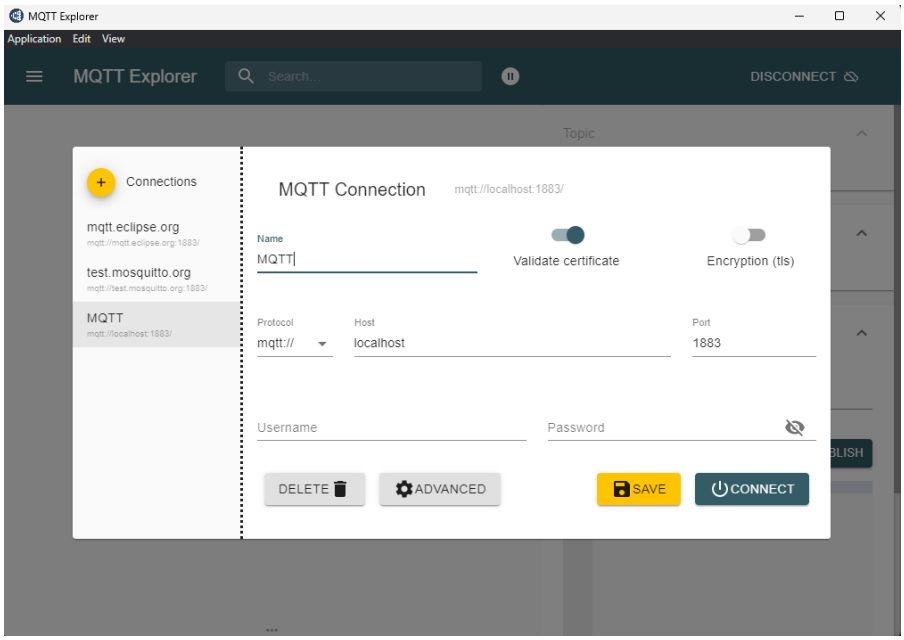



**Obrázek 6-16 Spuštění služby Mosquitto a připojení txt. souboru config**

Tento příkazový řádek musí zůstat otevřený pro využívání této služby, jelikož funguje jako správce serveru.

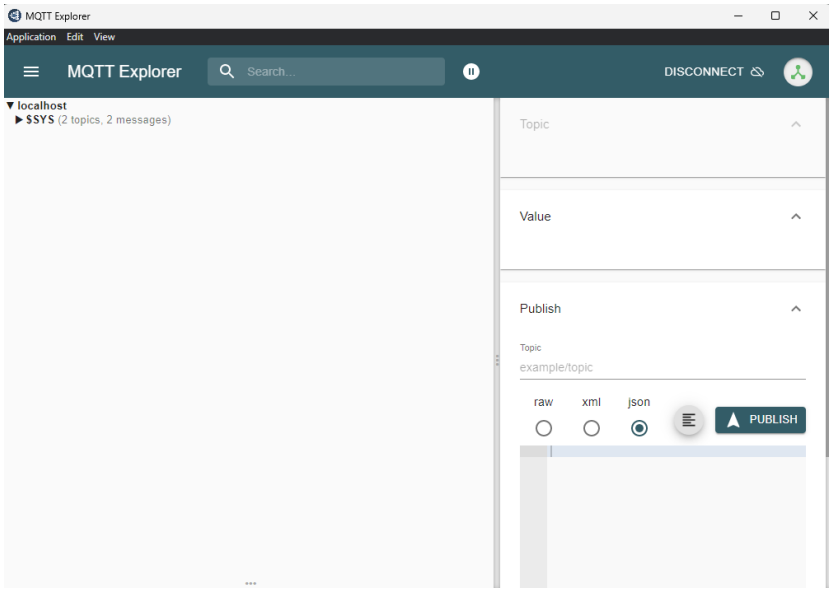
Instalace MQTT Explorer

Dalším krokem je stažení a nainstalování MQTT Exploreru dostupné z odkazu mqtt-explorer.com. Tento krok není povinný, ale je velmi užitečný pro přehledné zobrazení veškeré komunikace probíhající přes broker. Po nainstalování a spuštění je nutné vytvořit připojení, nastavit jméno, hosta, port a pokud je to nutné tak uživatelské jméno a heslo (v tomto případě nebylo nastavené, takže nutné není)




**Obrázek 6-17 Nastavení připojení Exploreru na Mosquitto**

Po připojení se zobrazí toto okno s veškerou komunikací procházející přes broker:



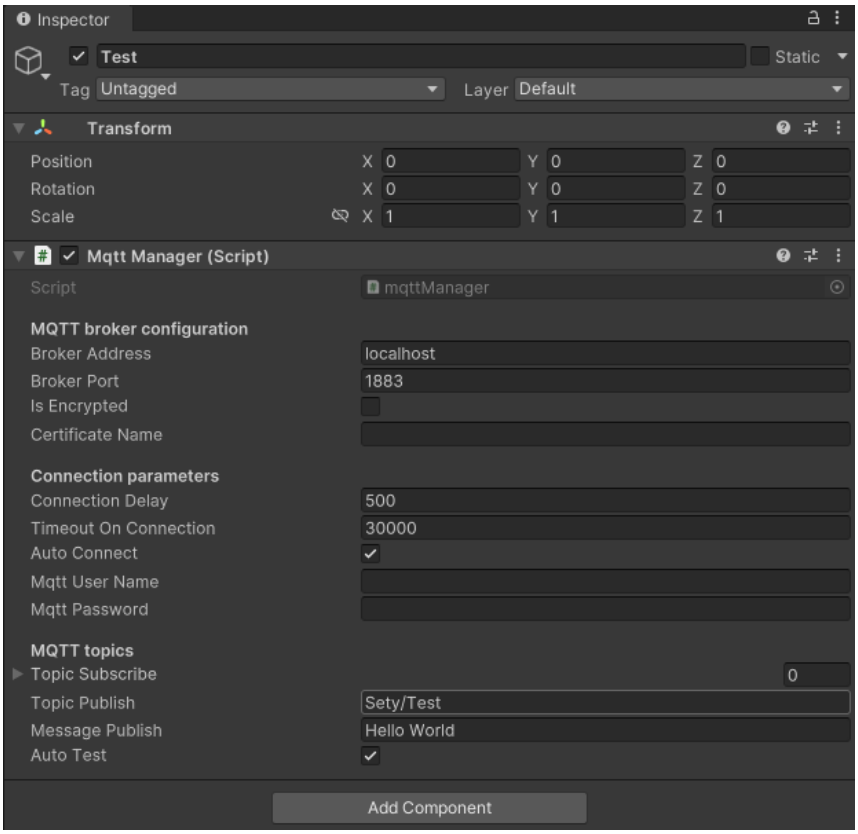

**Obrázek 6-18 Komunikace procházející přes Mosquitto**

Propojení MQTT-Unity

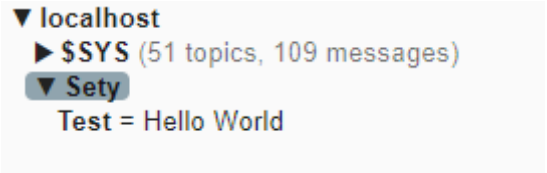
Pro možnost posílání a příjímání zpráv z brokeru v Unity je nutné provést několik kroků:

1. Spustit Mosquitto
2. Spustit MQTT Explorer
3. Mít k dispozici VS Code
4. Stáhnout M2MQTT knihovnu a MQTT Manager – lze stáhnout na stránce github

Po stažení obou souborů z kroku 4. je nutné je importovat do Unity. Soubor MQTT Manager, zajišťuje připojení na adresu, která je specifikovaná na GameObjectu, který má tento skript jako komponentu, viz níže. Je možné mít více takto vytvořených managerů odlišených Tagem, který je bude identifikovat. To umožní publikovat a přijímat zprávy z různých adres.




**Obrázek 6-19 Test MQTT managera**




**Obrázek 6-20 Poslaná zpráva z Unity na MQTT**

mqttController.cs

Tento skript ovládá akce, které se provedou při příjmu zprávy. Defaultně kód vypadá takto:

```
using UnityEngine;
public class OverviewManager : MonoBehaviour
{
    public string nameController = "Controller 1";
    public string tag_mqttManager = "";
    public mqttManager _eventSender;
    void Start()
    {
        if (GameObject.FindGameObjectsWithTag(tag_mqttManager).Length > 0)
        {
            _eventSender =
GameObject.FindGameObjectsWithTag(tag_mqttManager)[0].gameObject.GetComponent<mqt
tManager>();
        }
        else
        {
```

```
            Debug.LogError("At least one GameObject with mqttManager component
and Tag == tag_mqttManager needs to be provided");
        }
        _eventSender.OnMessageArrived += OnMessageArrivedHandler;
    }
    private void OnMessageArrivedHandler(mqttObj mqttObject)
    {}
```

Zde je stručný přehled klíčových součástí a funkcí kódu:

1.1 Proměnné:

**nameController**: proměnná představující název kontroléru, ve výchozím nastavení nastavená na "Controller 1".

**tagOfTheMQTTReceiver**: proměnná, která odkazuje na GameObject s připojeným scriptem mqttReceiver.

**_eventSender**: Instance třídy mqttReceiver, připojená k brokeru MQTT. Je zodpovědná za příjem zpráv MQTT.

1.2 Metoda Start:

V metodě Start je k události OnMessageArrived instance _eventSender připojena obsluha události pomocí metody OnMessageArrivedHandler. To znamená, že když přijde zpráva MQTT, zavolá se metoda OnMessageArrivedHandler.

1.3 Metoda OnMessageArrivedHandler:

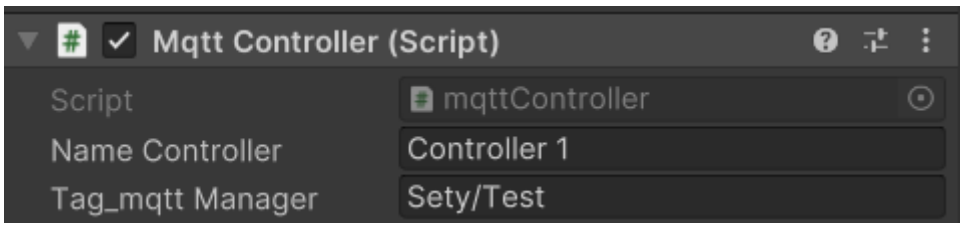
Tato metoda se volá vždy, když přidružený mqttReceiver obdrží zprávu MQTT. Pro otestování funkčnosti se do tohoto skriptu doplní následující funkce:

```
        this.GetComponent <TextMeshPro> ().text = newMsg;
        Debug.Log("Event Fired. The message, from Object " + nameController + " is
= " + newMsg);
```

Tato metoda aktualizuje text komponenty TextMeshPro připojené ke stejnému GameObjectu jako tento skript. Text je nastaven na hodnotu přijaté zprávy (newMsg). Do konzole Unity se také zapíše zpráva, která oznamuje, že událost byla vyvolána, a poskytuje informace o řadiči a přijaté zprávě.

Stručně řečeno, tento skript se připojí k přijímači MQTT (_eventSender), „naslouchá“ příchozím zprávám a provede popsanou akci (v tomto případě aktualizuje komponentu TextMeshPro).

Tento kód musí být připojený na objekt, jinak se nikdy nespustí. Bude tak vytvořen nový GameObject, který se nazve ControlledObject. V okně inspektora se nastaví Tag managera.




**Obrázek 6-21 Nastavení skriptu MQTT Controller**

Po spuštění programu lze poslat a zobrazit i z druhé strany, tedy z MQTT do Unity.


**Obrázek 6-22 Poslaná zpráva z MQTT do Unity**

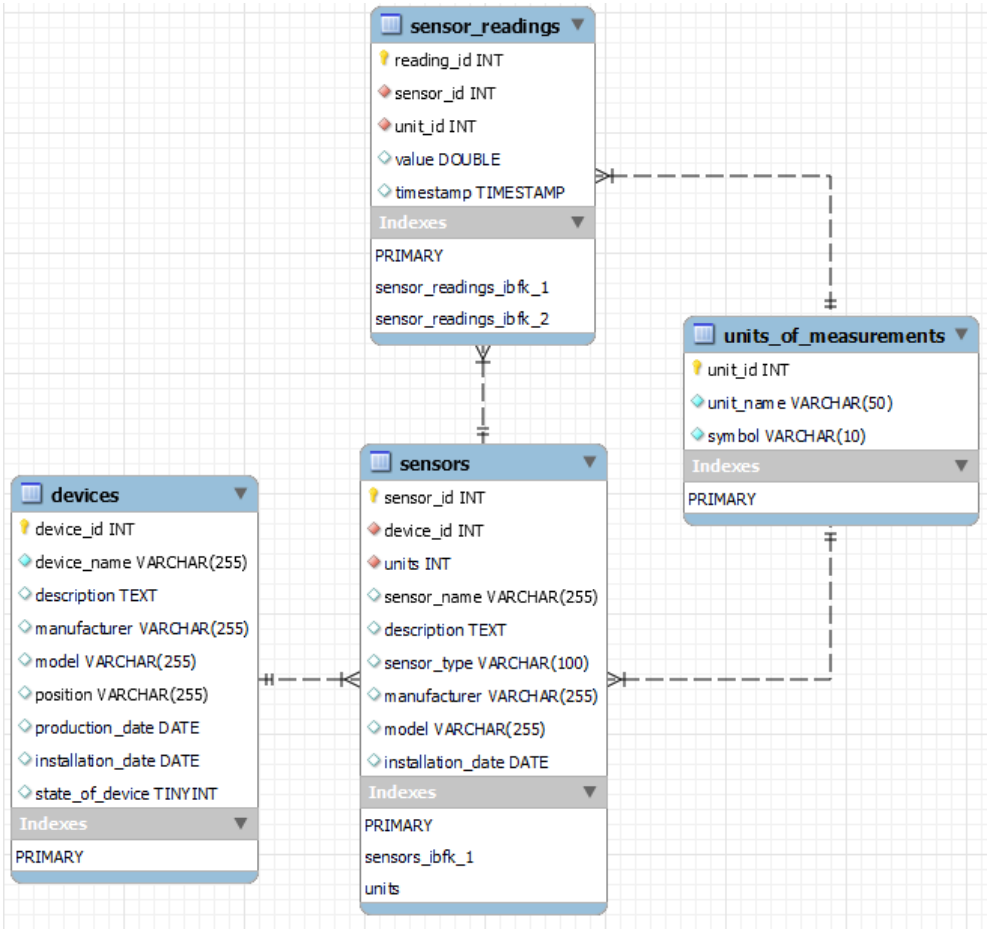
## 6.5 Databáze

Dalším důležitou komponentou, která zajistí funkci programu, je databáze. Správně nastavená databáze je klíčová pro uchování a správu dat, které se budou generovat a zpracovávat.

### 6.5.1 EER Diagram

EER neboli Enhanced Entity-Relationship diagram (rozšířený ER diagram), je rozšířená verze tradičního Entity-Relationship diagramu (ERD, také známý jako ERA). EER se používá k modelování složitějších datových struktur a vztahů, které nejsou snadno zachytitelné pomocí standardního ERD. Tento diagram byl vytvořen v rámci MySQL.

V diagramu lze vidět jednotlivé Entity (tabulky), jejich atributy (charakteristiky, které je popisují), jakého typu tyto atributy jsou (jako např. INT, VARCHAR apod.) a také vazby, které propojují jednotlivé entity. Vyskytují se zde pouze vazby 1:N.




**Obrázek 6-23 EER diagram databáze**

Tabulky budou blíže popsány v následující kapitole.

### 6.5.2 Vytvoření tabulek

Pro využití databáze MySQL bude opět nutné vytvořit uživatelský účet a program nainstalovat. Celkem byly vytvořeny 4 tabulky:

- Devices – tabulka, která popisuje obecné informace o zařízení jako je název, popis, výrobce, mode, pozice, datum výroby, datum instalace a zda je zařízení v provozu, které je zobrazeno 1, pokud je online, a 0 pokud je off-line. Každá položka tabulky je jednoznačně identifikována pomocí ID.

| device_id | device_name | description | manufacturer | model | position | production_date | installation_date | state_of_device |
|---|---|---|---|---|---|---|---|---|
| 1 | PlayStation 4 | Gaming Console | Sony | PS4 | room | 2024-01-01 | 2024-02-01 | 1 |

**Obrázek 6-24 Zobrazení dat v tabulce Devices**

- Sensors – tabulka, která popisuje informace o dostupných sensorech jako jméno, popis, typ sensoru, výrobce, mode a datum instalace. Každý sensor je identifikován pomocí ID a je přiřazen k zařízení, na kterém je naistalován pomocí cizího klíče na ID z tabulky Devices.

| sensor_id | sensor_name | description | sensor_type | manufacturer | model | installation_date | device_id |
|---|---|---|---|---|---|---|---|
| 1 | Current Sensor | Measures electrical current | Electrical | ManufacturerA | ModelA | 2024-02-01 | 1 |
| 2 | Voltage Sensor | Measures electrical voltage | Electrical | ManufacturerB | ModelB | 2024-02-01 | 1 |
| 3 | Temperature Sensor | Measures temperature | Environmental | ManufacturerC | ModelC | 2024-02-01 | 1 |
| 4 | Vibration Sensor | Measures vibration | Mechanical | ManufacturerD | ModelD | 2024-02-01 | 1 |
| 5 | Pressure Sensor | Measures pressure | Environmental | ManufacturerE | ModelE | 2024-02-01 | 1 |

**Obrázek 6-25 Zobrazení dat v tabulce Sensors**

- Units_of_Measurements – je tabulka, která popisuje všechny dostupné jednotky a jejich stanovený symbol. Všechny jednotky jsou identifikované pomocí ID

| unit_id | unit_name | symbol |
|---|---|---|
| 1 | Pascal | Pa |
| 2 | Volt | V |
| 3 | Ampere | A |
| 4 | Watt | W |
| 5 | Celsius (Temperature) | °C |
| 6 | Percentage | % |
| 7 | Bar | bar |
| 8 | Newton | N |
| 9 | RPM (Revolutions per Minute) | rpm |
| 10 | Lux | lux |
| 11 | Watts per meter square | W/m2 |

**Obrázek 6-26 Zobrazení dat v tabulce Units_of_measurement**

- Sensor_Readings – je tabulka, která skladuje všechny naměřené hodnoty ze senzorů do sloupce Value a rovněž i čas pořízení. Všechny měření jsou identifikovány pomocí ID měření, a poté pomocí dvou cizích klíčů, které je přiřadí k senzoru a jednotce.

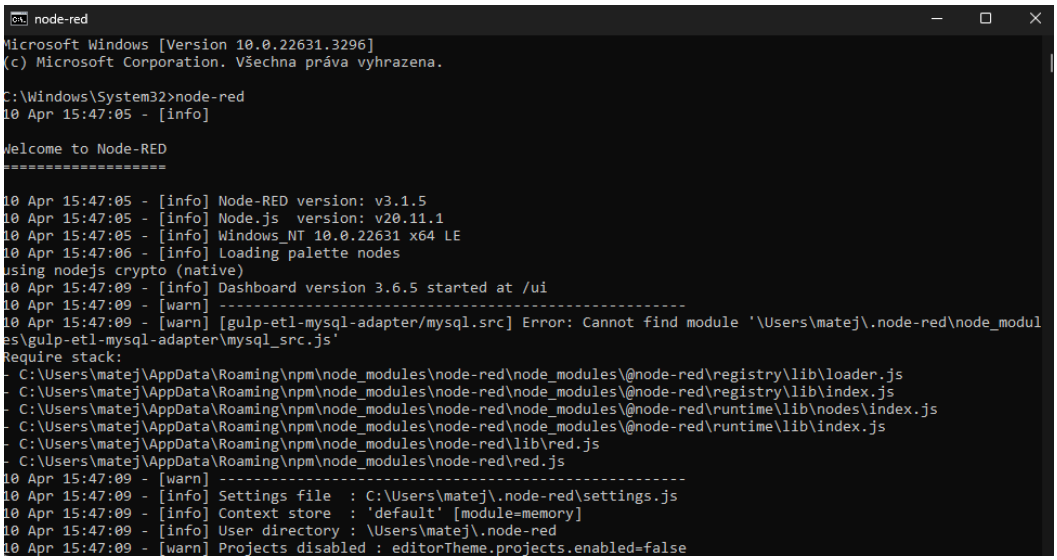
| reading_id | sensor_id | unit_id | value | timestamp |
|---|---|---|---|---|
| 433 | 3 | 5 | 36 | 2024-04-03 18:06:14 |
| 434 | 2 | 1 | 198 | 2024-04-03 18:06:14 |
| 435 | 1 | 3 | 130 | 2024-04-03 18:06:14 |
| 436 | 2 | 1 | 162 | 2024-04-03 18:06:19 |
| 437 | 1 | 3 | 131 | 2024-04-03 18:06:19 |
| 438 | 3 | 5 | 60 | 2024-04-03 18:06:19 |
| 439 | 1 | 3 | 135 | 2024-04-03 18:06:24 |
| 440 | 2 | 1 | 169 | 2024-04-03 18:06:24 |
| 441 | 3 | 5 | 53 | 2024-04-03 18:06:24 |
| 442 | 2 | 1 | 183 | 2024-04-03 18:06:29 |
| 443 | 1 | 3 | 130 | 2024-04-03 18:06:29 |
| 444 | 3 | 5 | 48 | 2024-04-03 18:06:29 |
| 445 | 1 | 3 | 118 | 2024-04-03 18:06:41 |
| 446 | 2 | 1 | 201 | 2024-04-03 18:06:41 |
| 447 | 3 | 5 | 44 | 2024-04-03 18:06:41 |
| 448 | 2 | 1 | 154 | 2024-04-03 18:06:46 |
| 449 | 1 | 3 | 123 | 2024-04-03 18:06:46 |
| 450 | 3 | 5 | 41 | 2024-04-03 18:06:46 |
| 451 | 2 | 1 | 180 | 2024-04-03 18:06:51 |
| 452 | 3 | 5 | 50 | 2024-04-03 18:06:51 |
| 453 | 1 | 3 | 113 | 2024-04-03 18:06:51 |
| 454 | 2 | 1 | 211 | 2024-04-03 18:06:56 |
| 455 | 1 | 3 | 125 | 2024-04-03 18:06:56 |
| 456 | 3 | 5 | 54 | 2024-04-03 18:06:56 |

**Obrázek 6-27 Zobrazení dat v tabulce Sensor_Readings**

## 6.6 Node-Red

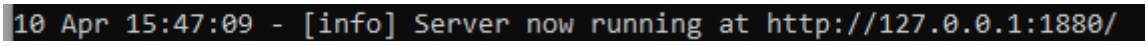
Node-RED (dále jen NR) je vizuální programovací nástroj pro propojování zařízení, senzorů a služeb na platformě Node.js. Umožňuje vytvářet tokové programy pomocí webového rozhraní, které využívá uzly (tzv. nodes) reprezentující různé funkce a operace. Node-RED je často používán pro vytváření IoT aplikací, automatizaci domácností, monitorování a řízení datových toků.

Jak již bylo zmíněno, NR je postaveno na platformě Node.js, využívá tedy jazyk Java Script. Proto je nutné nejdříve nainstalovat Node.js z oficiálních stránek nodejs.org. Poté, pomocí Node.js Package Manageru, lze nainstalovat NR příkazem *npm install – g --unsafe-perm node-red*. Po instalaci je třeba spustit instanci pomocí CMD příkazem node-red, viz níže.




**Obrázek 6-28 Instance Node-RED**

Takto vytvořenou instanci pak lze navštívit pomocí webového prohlížeče na adrese http://localhost:1880. Port adresy je k vidění v CMD.




**Obrázek 6-29 Adresa Node-RED**

Nyní je nutné importovat do NR dva moduly skrze Manage Palette:

- MQTT
- MySql

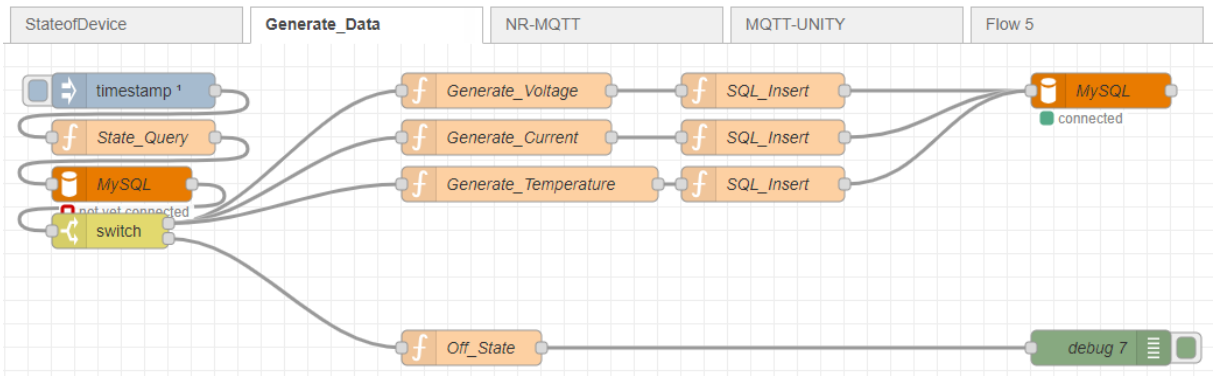
Tyto moduly umožní vytvořit

- MQTT in – node, který se připojí k MQTT brokeru a odebírá zprávy z nastaveného topicu
- MQTT out – node, který se připojí k MQTT brokeru a posílá zprávy do nastaveného topicu
- mysql – node, který umožní napojit se na databázi

S takto připraveným prostředím lze vyřešit zbývající problémy řešení:

- **Simulování měření ze senzorů**
- **Propojení s Unity**

Níže lze vidět flow, který řeší první problém.




**Obrázek 6-30 Flow pro generování dat ze sensorů**

Na začátku je node *timestamp*, který funguje jako spínač a je nastavený, aby se spustil každých 5 vteřin. Tento spínač pošle do vytvořené databáze jednoduchý SQL dotaz, zda je zařízení aktivní.

```
msg.topic = "SELECT state_of_device FROM devices WHERE device_id = 1"
return msg;
```

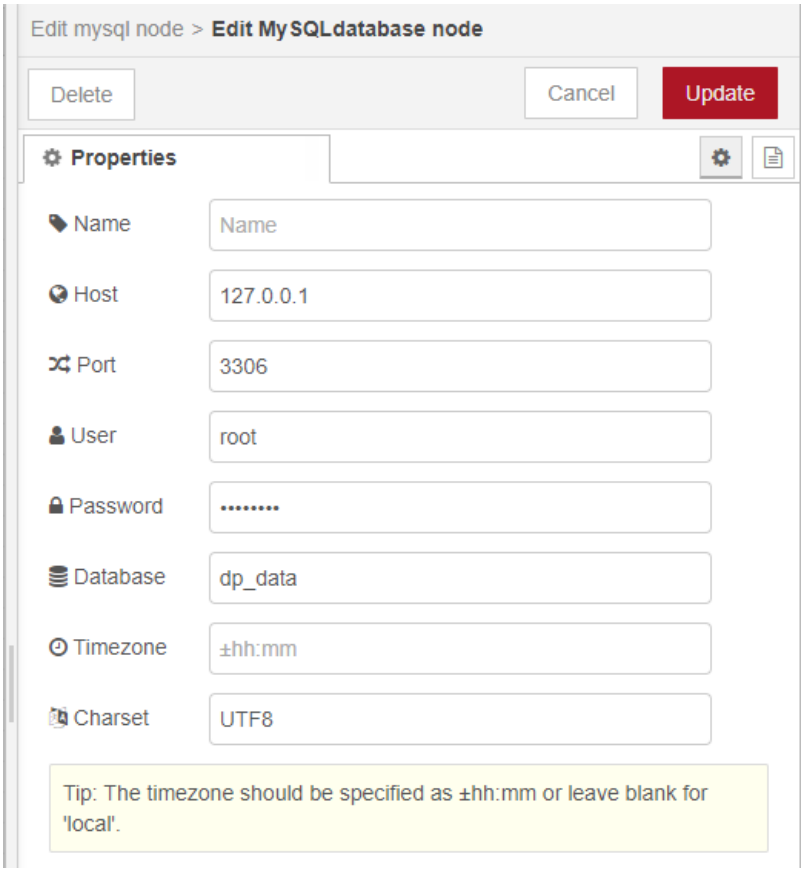
Dále je zde *switch*, který se vydá směrem nahoru, pokud je odpověď na dotaz 1, nebo se vydá dolů, pokud je odpověď 0. Spodní cesta vypíše do konzole „Device is Off“. Horní cesta ale spustí generování dat pomocí vzorce podle Box-Mullerovi transformace, viz níže. Jedná se statistickou metodu používanou ke generování náhodných čísel s normálním rozdělením.

```
function generateNormalDistribution(min, max) {
    const mean = (max + min) / 2;
    const stdDev = (max - min) / 6;
    // Box-Muller  transform
    const u1 = 1 - Math.random();
    const u2 = 1 - Math.random();
    const z0 = Math.sqrt(-2 * Math.log(u1)) * Math.cos(2 * Math.PI * u2);
    const result = Math.round(mean + stdDev * z0);
    return Math.max(min, Math.min(result, max));
}

const minValue = 100;
const maxValue = 150;
const generatedValue = generateNormalDistribution(minValue, maxValue);

msg.payload = generatedValue;
return msg;
```

Hodnota z této funkce se dále pošle do další SQL dotazu, který zapíše data pomocí příkazu INSERT do tabulky Sensor_readings.

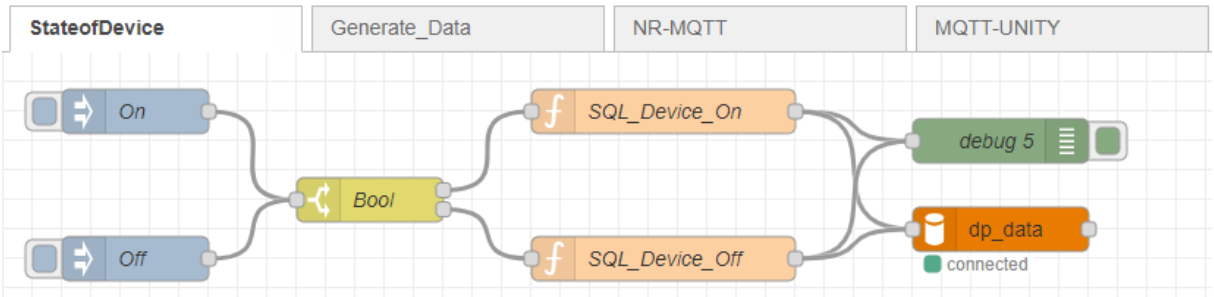
```
var sensor_id = 1;
var unit_id = 3;
var current = msg.payload

msg.topic = "INSERT INTO sensor_readings (sensor_id, value, unit_id)" +
    "VALUES (' " + sensor_id + "', ' " + current + "', ' " + unit_id + "');";
return msg;
```

Node mysql se ale musí nejdříve nastavit. Nejdříve je nutné založit novou třídu databáze, která se jmenuje MySQL. Takto vytvořené třídě se musí nastavit port, na kterém je databáze hostována, uživatelské jméno, heslo a název databáze.




**Obrázek 6-31 Nastavení databáze v Node-RED**

Dále je nutné vytvořit flow, který bude ovládat, zda je zařízení aktivní. Na začátku jsou dva nody *timestamp*, které vysílají zprávu typu Boolean, tedy TRUE nebo FALSE. Dále je switch, který aktivuje jednu z funkcí podle přijaté hodnoty. Uvnitř aktivované funkce je SQL dotaz, který aktualizuje sloupec state_of_device z tabulky devices.

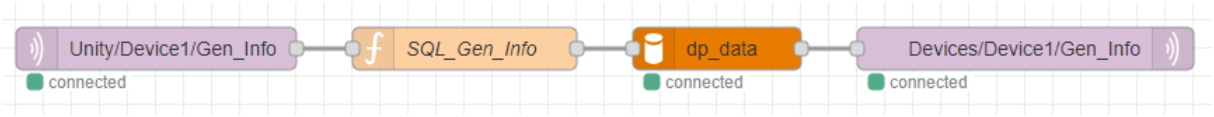



**Obrázek 6-32 Flow pro aktivaci/deaktivaci zařízení**

```
var value = 1
msg.topic = "UPDATE devices SET state_of_device = '" + value + "' WHERE
device_id = '1';"
msg.payload = "On"
return msg;
```

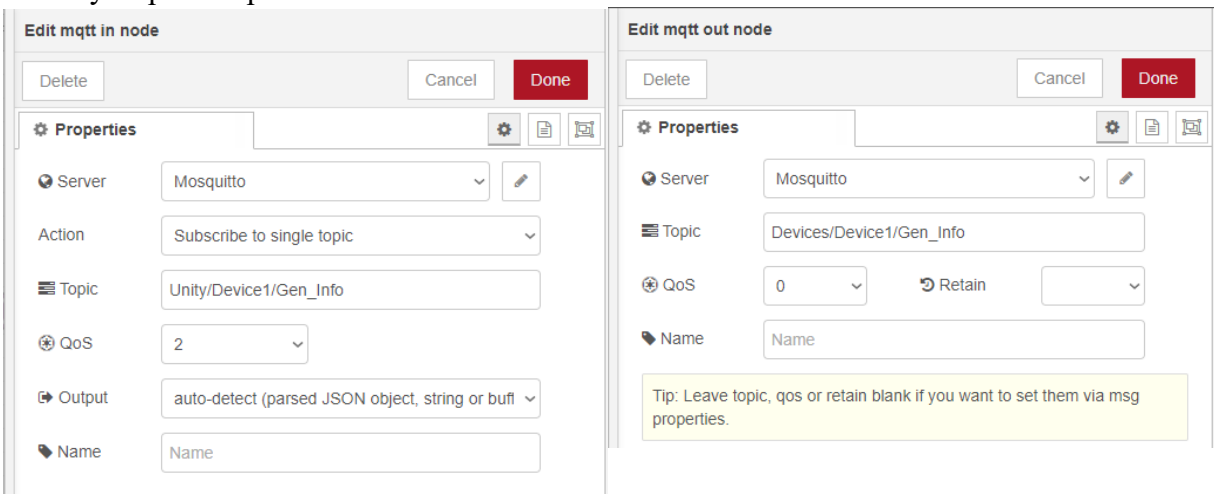
Druhý problém, tedy propojení s Unity, lze vyřešit následujícím způsobem. Využije se již implementovaná schopnost posílat z Unity do MQTT zprávu na příslušný topic. V NR je posluchač, který tento topic sleduje, a pokud sem přijde zpráva, tak vyšle signál dále. Na tento

node je navázaný SQL dotaz, který se vyšle do databáze, a odpověď na něj se pošle do jiného topicu v MQTT, který odebírá manager v Unity.

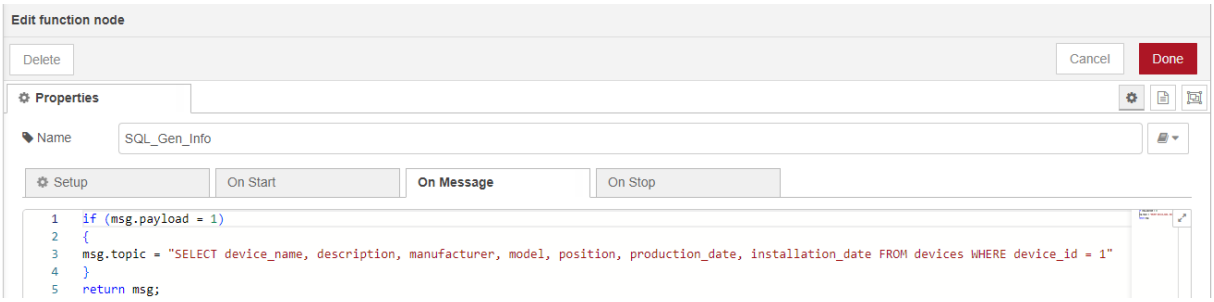



**Obrázek 6-33 Ukázka komunikace mezi MySQL a Unity**

Je nutné tyto MQTT nody nejdříve nastavit. Podobně jako u databáze je nutné vytvořit novou třídu, která se jmenuje Mosquitto. Dále je nutné specifikovat topic, který se odebírá či na který se posílá zpráva.

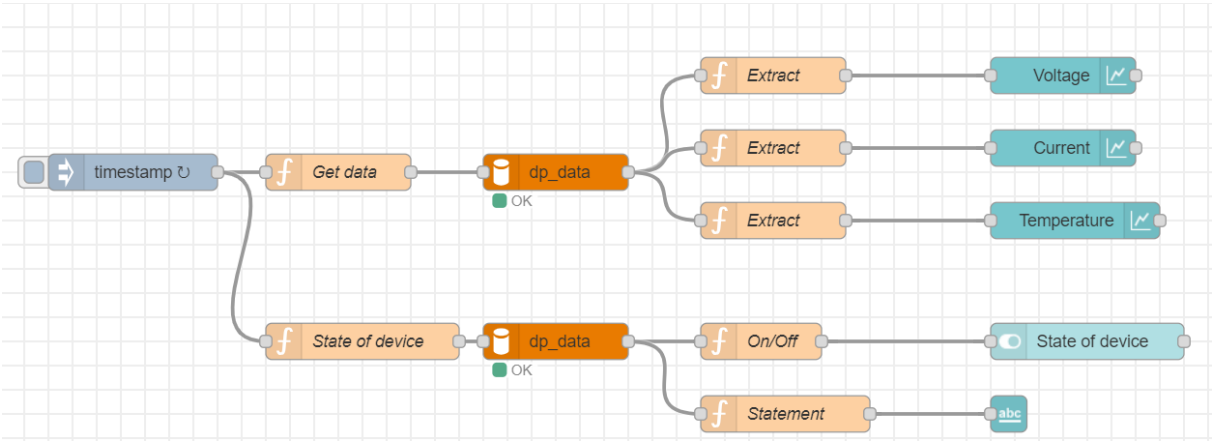



**Obrázek 6-34 Ukázka nastavení nodů MQTT in a MQTT out**




**Obrázek 6-35 Příklad SQL dotazu**

V NR je také možné vytvářet dashboardy, které umožní snadný přehled o hodnotách i mimo aplikaci v Unity. K tomu je nutné importovat další modul, *dashboard*. Tento modul poskytuje různé ovládací prvky jako např. textová pole, grafy, budíky apod. Flow pro vytvoření jednoduchého dashboardu je k vidění níže.




**Obrázek 6-36 Flow pro vytvoření dashboardu**

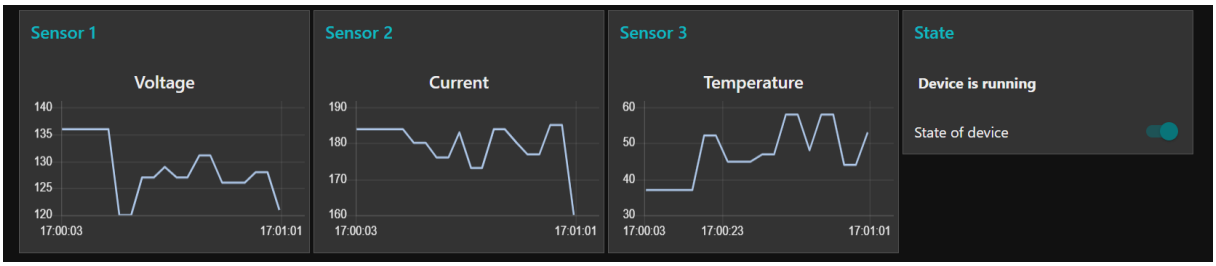
Na začátku opět stojí spínač, který se aktivuje každých 5 vteřin a navazují na něj dva SQL dotazy. State of device, který byl ukázaný výše, ovládá textové pole a switch. Get data funkce obsahuje SQL dotaz, který vybere nejnovější záznam ze sloupce value, z tabulky Sensor_readings, pro sensory 1, 2 a 3.

```
msg.topic = "WITH RankedReadings AS (SELECT sensor_id, value,ROW_NUMBER() OVER
(PARTITION BY sensor_id ORDER BY timestamp DESC) AS row_num FROM
sensor_readings WHERE sensor_id IN (1, 2, 3)) SELECT sensor_id, value FROM
RankedReadings  WHERE row_num = 1"
return msg;
```

Přijatá data z databáze jsou ve formátu JSON, a proto se musí požadované hodnoty extrahovat.

```
var incomingData = msg.payload;
var firstValue = msg.payload[0].value;
msg.payload = firstValue;
return msg;
```

Tyto data dále putují do grafů, které je vizualizují.

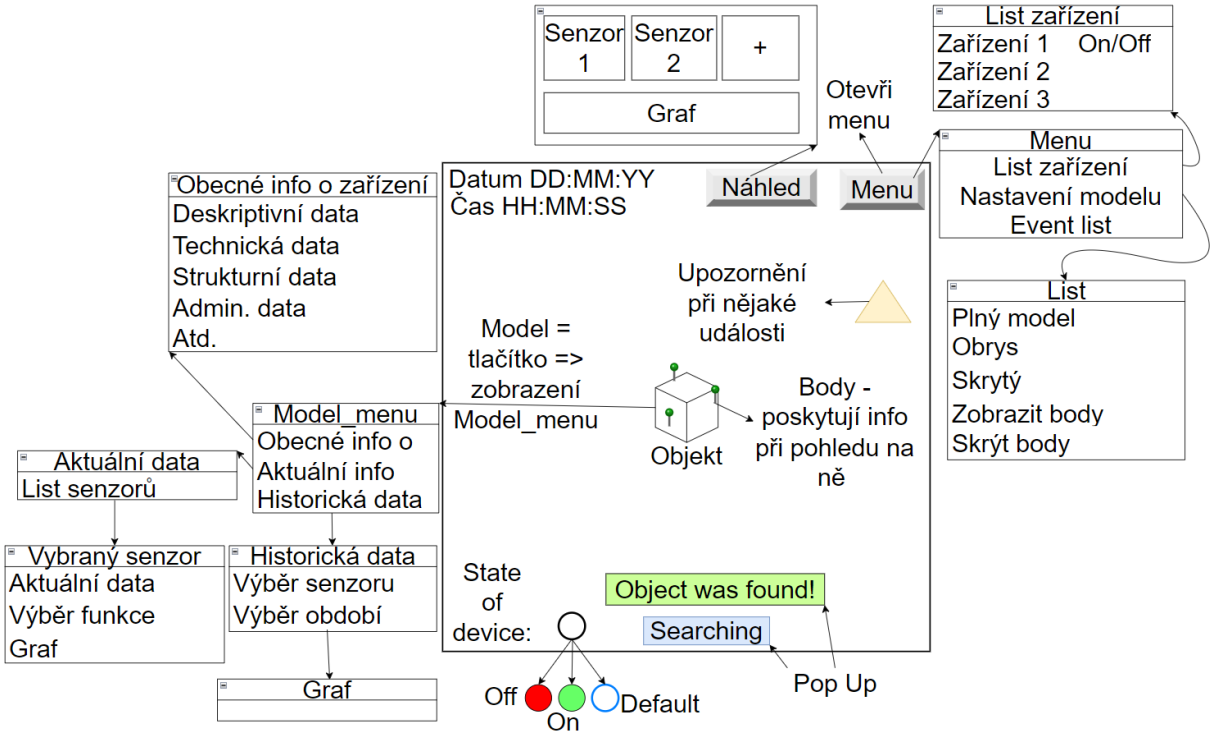



**Obrázek 6-37 Výsledný dashboard**

# 7 Popis pilotní aplikace

Po nastavení všech částí je nyní nutné vytvořit samotnou aplikaci. Prvním krokem bude vytvoření grafického rozhraní, tedy vytvoření všech menu a vytvoření jednoduchých funkcionalit pro jejich ovládání, doplnění augmentace a dokončení komunikace.

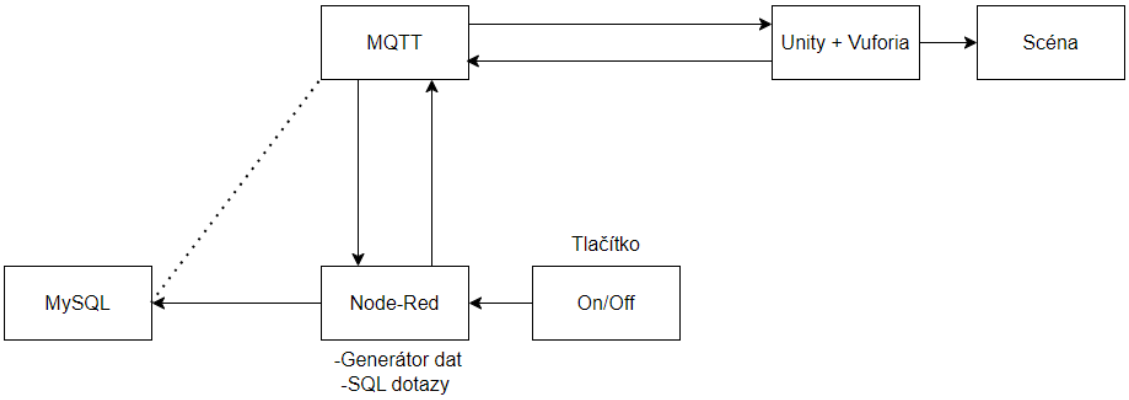
Na začátku byl vytvořen hrubý návrh, jak by mohlo GUI vypadat. Tento návrh je možné vidět na Obrázek 7-1 Návrh GUI.




**Obrázek 7-1 Návrh GUI**

Na ploše aplikace je zobrazené datum a čas, stav zařízení, upozornění při nějaké události jako např. překročení hranice hodnot senzoru, a vyskakovací hláška, která řekne že zařízení bylo nalezeno. V pravém horním rohu jsou dvě tlačítka – Overview a Menu. Overview bude zobrazovat všechny senzory na zařízení a vytváří graf. Menu umožňuje získat přehled o všech zařízeních, přehled o událostech, a umožňuje ovládat model. Dále je tlačítkem samotný model, který aktivuje model_menu. Zde jsou k dispozici tři možnosti – Obecné informace o zařízení, jako např. Model, výrobce, datum instalace apod., Aktuální data, které zobrazí měření ze senzorů a po kliknutí dodává detailní informace, Historická data, která umožní vybrat senzor a časové období, a na základě těchto proměnných vytvoří graf.

Obrázek 7-2 Komunikace mezi prvky představuje návrh komunikace mezi jednotlivými prvky. Detailně je tento proces popsán v následující kapitole.

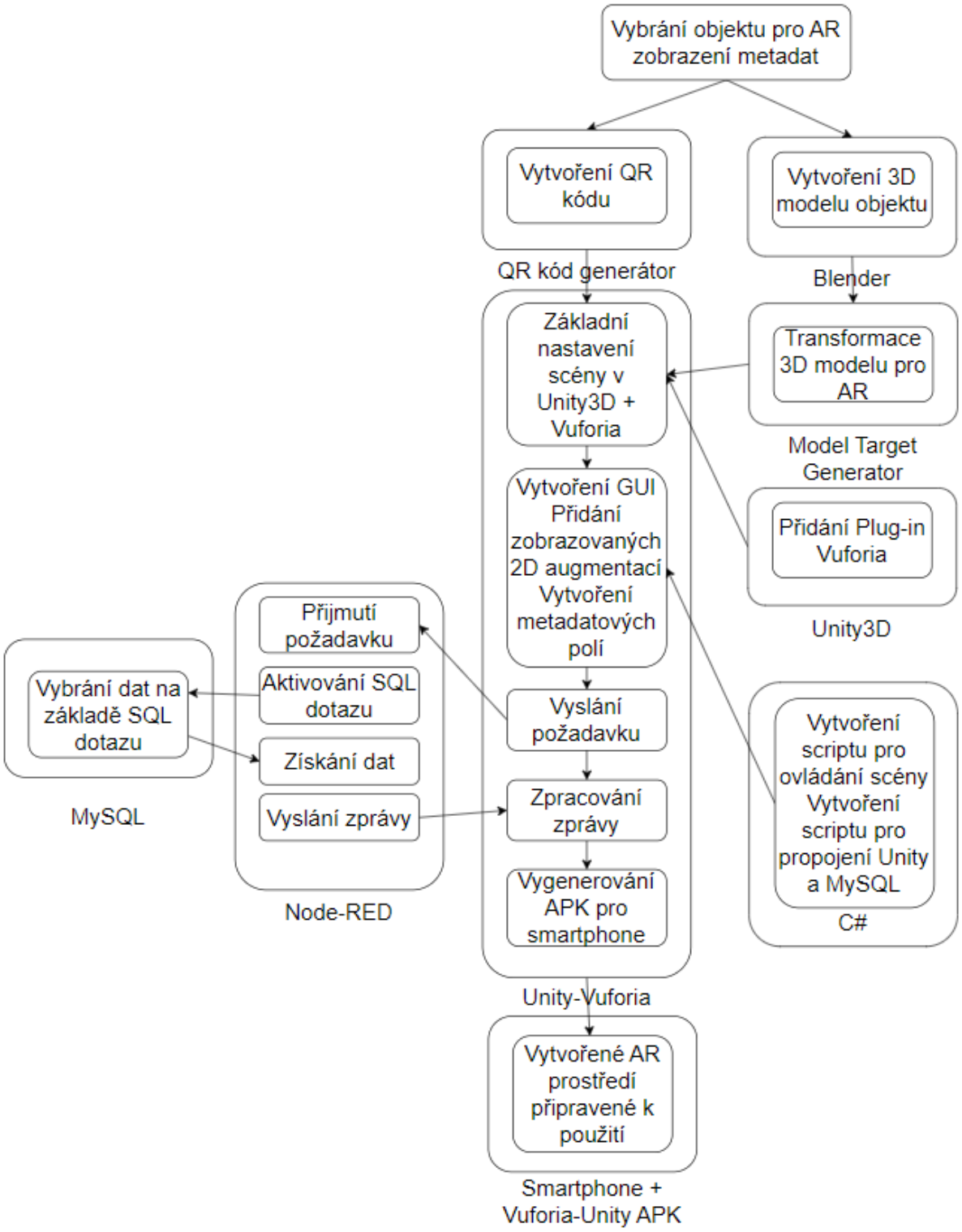



**Obrázek 7-2 Komunikace mezi prvky**

## 7.1 Business logika

Business logika je jádrová část softwaru, která definuje, jaké procesy a pravidla jsou aplikovány na data v rámci daného obchodního prostředí nebo domény. Jedná se o logiku, která řídí chování aplikace na základě obchodních pravidel a požadavků, nikoliv o aspekty týkající se uživatelského rozhraní nebo technických detailů implementace.

V rámci aplikace se datový tok aktivuje na základě trackovaného modelu, načež se aktivuje daná augmentace. Při otevření panelu, který má metadatová pole, např. panel Aktuální data, se vyšle požadavek na topic na serveru MQTT. Tomuto topicu „naslouchá“ NR, které na základě správy aktivuje SQL dotaz. V databázi MySQL se tento SQL dotaz spustí, a vyhledaná data se vyšlou zpět do NR, kde se zpracují a pošlou do jiného topicu MQTT. Tomuto topicu naslouchá mqttManager v Unity. Skript na aktivovaném panelu má rovněž funkci pro zpracování přijaté zprávy, která je formátu JSON. Zpráva se rozklíčuje a zobrazovaná scéna se aktualizuje s potřebnými informacemi.




**Obrázek 7-3 Business logika aplikace**

## 7.2 GUI, augmentace, import

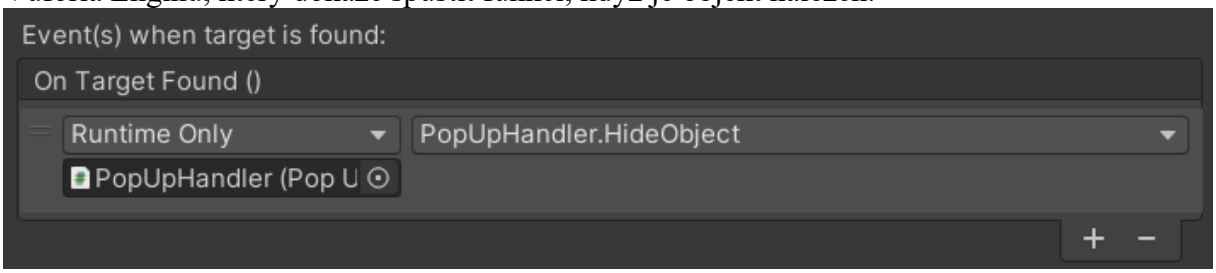
Grafické uživatelské rozhraní je důležité, jelikož poskytuje prostředí, ve kterém lze interagovat s obsahem a ovládat aplikaci. Důležité je, aby prostředí bylo jednoduché a intuitivní, ale aby poskytovalo dostatečné množství informací.

Nejdříve je nutné vytvořit Canvas, na který se budou umisťovat jednotlivé prvky:

- Datum a čas – Umístěný vlevo nahoře, je ovládaný scriptem currentTime

```
using System;
using TMPro;
using UnityEngine;

public class currentTime : MonoBehaviour
{
    public TMP_Text dateTimeText;
    void Update()
    {
        DateTime currentDateTime = DateTime.Now;
        string formattedDateTime = currentDateTime.ToString("\ndd-MM-
yy\nHH:mm:ss");
        dateTimeText.text = formattedDateTime;
    }
}
```

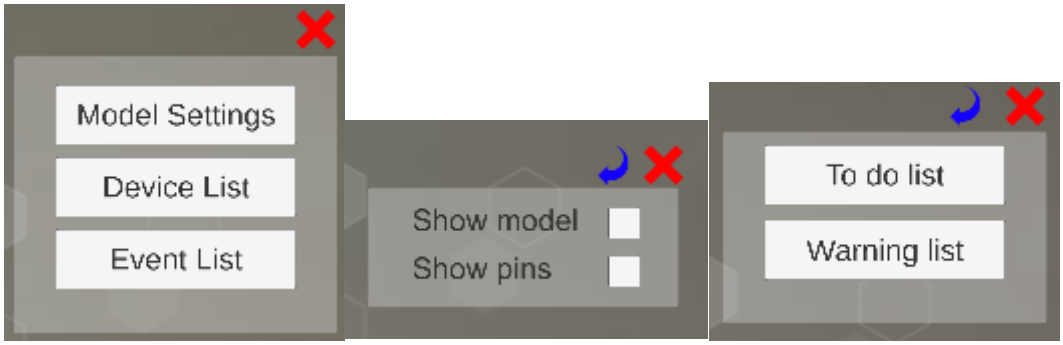
- State of device – V pravo dole, ve kterém se budou aktivovat prvky On a Off v závislosti na stavu zařízení
- Pop up – Uprostřed dole, upozornění, které bude ukazovat *Searching…* když kamera hledá objekt, a Object *was found!* když byl objekt nalezen. K tomuto je využito funkce Vuforia Enginu, který dokáže spustit funkci, když je objekt nalezen.




**Obrázek 7-4 Event trigger Vuforia Enginu**

Tento event je ovládaný skriptem PopUpHandler, který je k vidění v příloze 1.

- Tlačítka Overview a Menu v pravém horním rohu. Overview zobrazí panel, který ukáže rychlý přehled o všech sensorech. V menu bude možné ovládat prvky aplikace jako viditelnost modelu, zobrazení dodatečných dat, dá přehled o všech zařízeních a zobrazí event list.

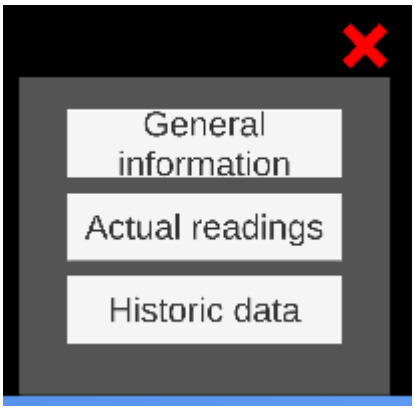



**Obrázek 7-5 Ukázka menu**


**Obrázek 7-6 Ukázka Overview**

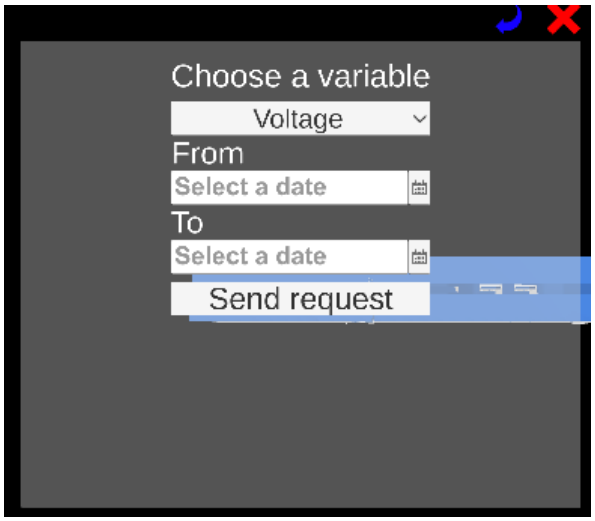
Barva, zaplnění ukazatele a ukazatel, zda je hodnota v pořádku je ovládána pomocí skriptu ColorChanger, který je k vidění v příloze 2. Dále je zde graf, který byl vytvořený za pomocí XCharts plug inu dostupného ze stránky github.

- Posledním elementem bude nastavení zobrazovaného modelu jako tlačítko. K tomu je zapotřebí přidat k objektu BoxCollider a vytvořit skript, který umožní na něj kliknout. Tento skript je k vidění v příloze 3. Po kliknutí na objekt se zobrazí model menu níže.




**Obrázek 7-7 Ukázka model menu**

- General Information bude zobrazovat všechny informace o zařízení. Actual readings bude zobrazovat aktuální data. Historic data umožní zobrazit starší data.




**Obrázek 7-8 Ukázka tabulky Historic data**

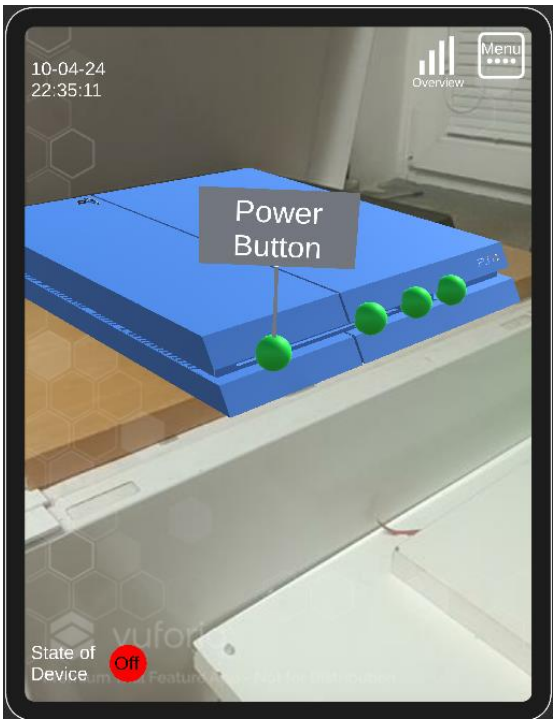
Kromě těchto elementů se doplní dodatečná augmentace samotného modelu. Ta spočívá v zobrazení důležitých částí objektu jako např. USB port, tlačítko start, CD-Rom apod. Tento „Pin“ se skládá ze 4 částí – sféra, která je počátkem následujících objektů, obdélník, který spojuje sféru a plochu, která má na sobě TMP objekt.


**Obrázek 7-9 Ukázka Pinu**

Takto vytvořený Pin je viditelný vždy, a je čitelný pouze z jednoho pohledu. Proto je třeba zajistit, aby se otevřel pouze v případě, že je na něj namířená kamera, a aby se vždy natočil směrem k ní. To zajišťují tři skripty, které jsou k vidění v příloze 4, 5 a 6:

- Gaze – Tento skript spravuje informační obsah ve scéně na základě toho, na co se uživatel dívá. Sleduje všechny objekty pomocí skriptu InfoBehaviour a otevírá nebo zavírá jejich informace na základě interakce s uživatelem. Je připevněný na AR kameře
- InfoBehaviour – Tento kód otevírá a zavírá informace objektu. Je připevněný na objekt, který je počátek, tedy sféra.
- FaceCamera – Tento kód spravuje rotaci objektu v závislosti na poloze kamery. Je připevněny na objektu, který má informaci.

Tímto je dokončená augmentace a tvorba uživatelského rozhraní.




**Obrázek 7-10 Výsledná augmentace**

## 7.3 Komunikace

Tato kapitola je zaměřena na vytvoření komunikace mezi Unity a MQTT, a požadovanou odpovědí od NR. Komunikace v Unity je, jak již bylo zmíněno, postavená na MQTT manageru. Pro každou komponentu, která bude požadovat informace, je třeba vytvořit vlastní manager a controller. Těmito objekty jsou v tomto případě:

- Connection – Testovací zpráva při spuštění aplikace
- Device List – Zobrazuje všechny zařízení v databázi
- Overview – Rychlý přehled o sensorech
- Actual Data – Informace o jednotlivých sensorech včetně dodatečných informací jako např. průměr za den, maximální/minimální hodnota apod.
- General Information – Informace o snímaném zařízení
- Historic Data – Zobrazuje starší data na základě výběru
- State – Kontroluje stav zařízení

Následně se, k již vytvořeným tabulkám připojí MQTT controller, který začne posílat zprávu do MQTT, která funguje jako požadavek o informaci, pomocí následujícího kódu:

```
public void PublishMessage()
{

    _eventSender.Publish();

    Debug.Log("Message sent");
}
private void OnEnable()
{
    InvokeRepeating("PublishMessage", 0f, 2f);
}
private void OnDisable()
{
    CancelInvoke("PublishMessage");}
```

Všechny zprávy, které se přijmou z MQTT jsou ale ve formátu JSON. JSON (JavaScript Object Notation) je formát pro výměnu dat, který je snadno čitelný jak pro lidi, tak pro počítače. Je to lehký textový formát, který se skládá ze dvojic klíč-hodnota, a je často používán pro přenos dat mezi webovými službami a klienty.


```
[
  {
    "device_id": 1,
    "device_name": "PlayStation 4",
    "state_of_device": 0
  },
  {
    "device_id": 2,
    "device_name": "Test",
    "state_of_device": 1
  }
]
```


**Obrázek 7-11 Ukázka zprávy JSON**

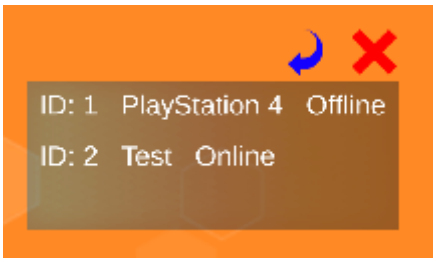
Je nutné tuto zprávu rozklíčovat. K tomu se použije plug in z asset storu Json.NET Converters. Příklad tohoto rozklíčování je ukázán níže.

```
private void OnMessageArrivedHandler(mqttObj mqttObject)
{
    foreach (JObject deviceJson in jsonArray)
    {
        int deviceId = deviceJson.Value<int>("device_id");
        string deviceName = deviceJson.Value<string>("device_name");
        int deviceState = deviceJson.Value<int>("state_of_device");
        string deviceStateText = deviceState == 1 ? "Online" : "Offline";
        TextMeshProUGUI tmp = Instantiate(tmpPrefab, tmpContainer);
        tmp.gameObject.SetActive(true);

        tmp.text = $"ID: {deviceId}   {deviceName}   {deviceStateText}";
```

Tento kód obsahuje metodu s názvem OnMessageArrivedHandler, která se spouští, když přijde zpráva na MQTT server. Tato metoda přijímá jeden parametr mqttObj, který obsahuje informace o příchozí zprávě.

Uvnitř metody se nejprve provádí analýza přijaté zprávy ve formátu JSON. Předpokládá se, že zpráva má formu pole JSON objektů, které jsou reprezentovány jako instanci třídy JArray z knihovny Newtonsoft.Json.

Následně je procházeno každé zařízení ve zprávě pomocí smyčky foreach. Pro každé zařízení se extrahují informace jako deviceId, deviceName a deviceState z odpovídajícího JSON objektu. Tyto informace jsou pak použity k vytvoření nové instance textového pole TextMeshProUGUI (TMP) pomocí metody Instantiate. Toto pole je inicializováno s textem, který obsahuje identifikátor zařízení, jeho jméno a stav. Celý kód je k vidění v příloze 7.

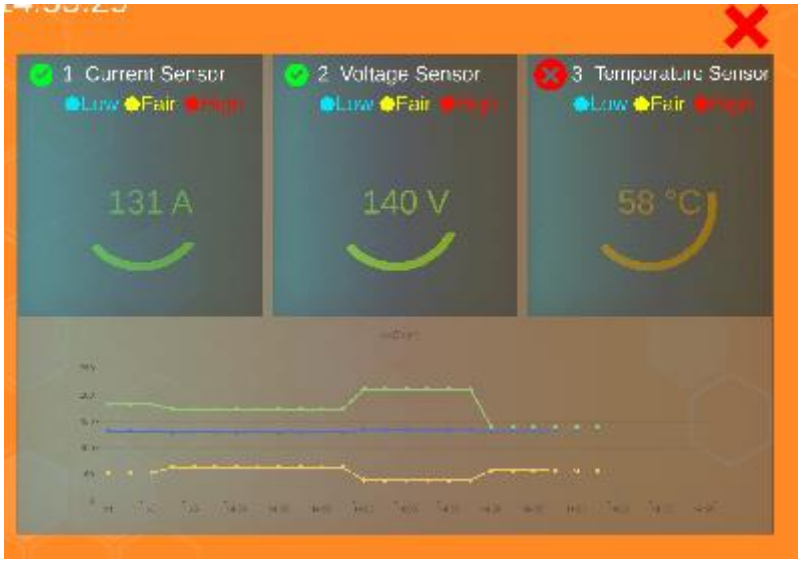



**Obrázek 7-12 Ukázka přehledu zařízení**

Podobným způsobem bude vytvořen zbytek komunikace s menšími úpravami v závislosti na obsahu zprávy. Dále zbývá pouze zařídit aktualizaci hodnot grafů. Postup bude vysvětlen na grafu v tabulce Overview. Bude vytvořen graf *XCharts -> LineChart -> BasicLine*. Nastaví se 3 datové série, každá bude čerpat z jiné proměnné, která se extrahuje z příchozí zprávy.

```
string currentTime = DateTime.Now.ToString("HH:mm");
lineChart.AddXAxisData(currentTime);
lineChart.AddData(0, float.Parse(devCurr));
lineChart.AddData(1, float.Parse(devVolt));
lineChart.AddData(2, float.Parse(devTemp));
```

AddXAxisData nastavuje osu X jako čas, a AddData nahrává jednotlivé proměnné do grafu v závislosti na čísle. Výsledek je vidět níže.




**Obrázek 7-13 Dokončené Overview**

# 8 Vytvoření druhé iterace

Tato kapitola se zaměří na celkové dokončení aplikace, kroky jsou následující:

- Doplnění sledovaných objektů
- Doplnění databáze o dodatečná zařízení
- Upravení žádosti na základě objektu
- Doplnění SQL dotazů
- Vytvoření APK balíčku pro instalaci

Předmětem kapitoly je i „bug-fixování“, neboli oprava či odstranění nežádoucích prvků v rámci aplikace.

## 8.1 Doplnění sledovaných objektů

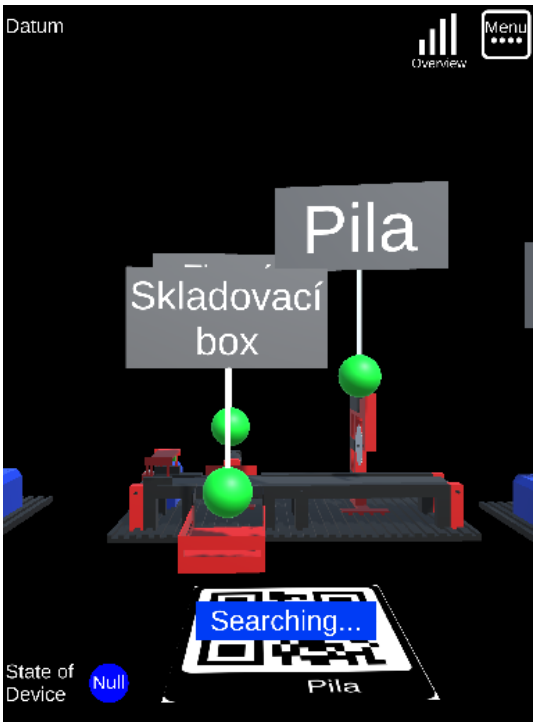
Je nutné aplikace doplnit o dodatečná zařízení, aby bylo možné simulovat skutečné využití v rámci podniku. Modely byly poskytnuté Ing. Matějem Dvořákem a následně přetvořeny Model Target podobně jako PS4. Při testování trackování bylo zjištěno, že modul nedokáže tyto modely najít. To je pravděpodobně z důvodu, že jsou „uvedeny do provozu“, tj. mají na sobě připojené kabely, jsou připevněny k sobě a v pozadí jsou další stanice, což znemožňuje jednoznačnou identifikaci. Bylo tak nutné opustit toto trackovací řešení a využít tak značku. Značka je vytvořena na základě QR kódu a textového označení pro snadnou identifikaci. Následně byla vytvořena databáze, která má všechny QR kódy, a přetvořena na plug-in, který se nahrál do prostředí Unity.

| Image | Target Name | Type | Rating | Status | Date Modified |
|---|---|---|---|---|---|
| | SenzorovaStanice | Image | ★★★★★ | Active | Apr 30, 2024 |
| | Rameno | Image | ★★★★★ | Active | Apr 30, 2024 |
| | Pila | Image | ★★★★★ | Active | Apr 30, 2024 |
| | FrezkaVrtacka | Image | ★★★★★ | Active | Apr 30, 2024 |
| | Svarecka | Image | ★★★★★ | Active | Apr 30, 2024 |
| | Sklad | Image | ★★★★★ | Active | Apr 30, 2024 |
| | Playstation4 | Image | ★★★★★ | Active | Apr 30, 2024 |

**Obrázek 8-1 Databáze značek**

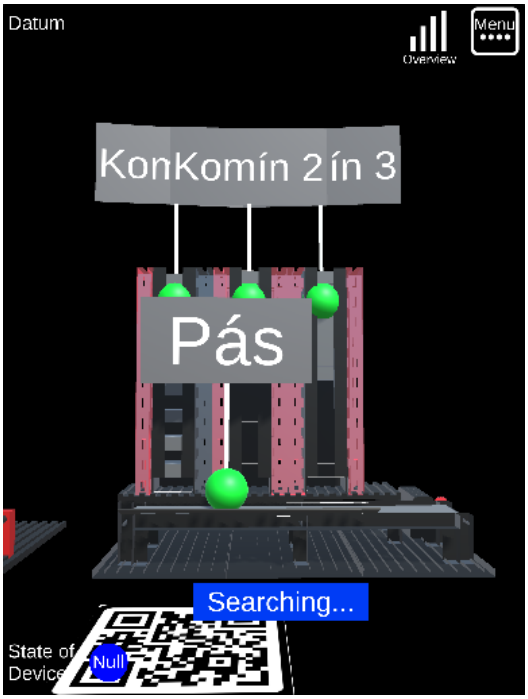
Pro jednotlivé stanice byli vytvořeny Image Targety s příslušným prvkem z databáze. Byla vytvořena jednoduchá augmentace jako v případě PS4, která je vidět na obrázcích níže. Celkem byly doplněny 4 stanice – Frézka/Vrtačka, Pila, Svářečka a Sklad.


**Obrázek 8-2 Augmentace Frézka/Vrtačka**




**Obrázek 8-3 Augmentace Pily**


**Obrázek 8-4 Augmentace Svářečky**




**Obrázek 8-5 Augmentace sklad**

## 8.2 Doplnění databáze o dodatečná zařízení

Jelikož byla doplněna zařízení je třeba upravit i MySQL databázi. Všechny čtyři objekty byli doplněny do tabulky Devices.

| devic | device_name | description | manuf | model | positio | production_c | installation_ | stat |
|---|---|---|---|---|---|---|---|---|
| 1 | PlayStation 4 | Gaming Console | Sony | PS4 | room | 2024-01-01 | 2024-02-01 | 0 |
| 2 | Test | Test | Test | Test | Test | 2024-01-01 | 2024-01-01 | 1 |
| 3 | Pila | Pila | ZČU | Fischer | room | 2024-01-01 | 2024-01-01 | 0 |
| 4 | Sklad | Sklad | ZČU | Fischer | room | 2024-01-01 | 2024-01-01 | 0 |
| 5 | Svářečka | Svářečka | ZČU | Fischer | room | 2024-01-01 | 2024-01-01 | 0 |
| 6 | Frézka Vrtačka | Frézka Vrtačka | ZČU | Fischer | room | 2024-01-01 | 2024-01-01 | 0 |

**Obrázek 8-6 Doplnění zařízení do tabulky Devices**

Ke všem zařízením byly přidány senzory, které se mohou vyskytovat na jejich skutečných protějšcích, např. stanice pila má nejenom senzory Proud, Napětí a Teplota, ale také Vibrace, Rotace a Kapacita.

| sensi | devic | sensor_name | description | sensor_type | manufacturer | model | installation_date | units |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | Current Sensor | Measures e... | Electrical | ManufacturerA | ModelA | 2024-02-01 | 3 |
| 2 | 1 | Voltage Sensor | Measures e... | Electrical | ManufacturerB | ModelB | 2024-02-01 | 2 |
| 3 | 1 | Temperature... | Measures t... | Environmental | ManufacturerC | ModelC | 2024-02-01 | 5 |
| 4 | 1 | Vibration Sen... | Measures v... | Mechanical | ManufacturerD | ModelD | 2024-02-01 | 13 |
| 5 | 1 | Pressure Sen... | Measures p... | Environmental | ManufacturerE | ModelE | 2024-02-01 | 7 |
| 11 | 3 | Current Sensor | Measures e... | Electrical | ManufacturerA | ModelA | 2024-02-01 | 3 |
| 12 | 3 | Voltage Sensor | Measures e... | Electrical | ManufacturerB | ModelB | 2024-02-01 | 2 |
| 13 | 3 | Temperature... | Measures t... | Environmental | ManufacturerC | ModelC | 2024-02-01 | 5 |
| 23 | 3 | Vibration Sen... | Measures v... | Environmental | ManufacturerD | ModelD | 2024-02-01 | 13 |
| 24 | 3 | Rotation Sen... | Measures r... | Mechanical | ManufacturerE | ModelE | 2024-02-01 | 9 |
| 28 | 3 | Capacity Sen... | Measures c... | Volume | ManufacturerF | ModelF | 2024-02-01 | 14 |
| 14 | 4 | Current Sensor | Measures e... | Electrical | ManufacturerA | ModelA | 2024-02-01 | 3 |
| 15 | 4 | Voltage Sensor | Measures e... | Electrical | ManufacturerB | ModelB | 2024-02-01 | 2 |
| 16 | 4 | Temperature... | Measures t... | Environmental | ManufacturerC | ModelC | 2024-02-01 | 5 |
| 25 | 4 | Capacity Sen... | Measures c... | Volume | ManufacturerF | ModelF | 2024-02-01 | 14 |
| 17 | 5 | Current Sensor | Measures e... | Electrical | ManufacturerA | ModelA | 2024-02-01 | 3 |
| 18 | 5 | Voltage Sensor | Measures e... | Electrical | ManufacturerB | ModelB | 2024-02-01 | 2 |
| 19 | 5 | Temperature... | Measures t... | Environmental | ManufacturerC | ModelC | 2024-02-01 | 5 |
| 29 | 5 | Pressure Sen... | Measures p... | Environmental | ManufacturerG | ModelG | 2024-02-01 | 7 |
| 30 | 5 | Flow rate Se... | Measures fl... | Environmental | ManufacturerI | ModelI | 2024-02-01 | 15 |
| 20 | 6 | Current Sensor | Measures e... | Electrical | ManufacturerA | ModelA | 2024-02-01 | 3 |
| 21 | 6 | Voltage Sensor | Measures e... | Electrical | ManufacturerB | ModelB | 2024-02-01 | 2 |
| 22 | 6 | Temperature... | Measures t... | Environmental | ManufacturerC | ModelC | 2024-02-01 | 5 |
| 26 | 6 | Vibration Sen... | Measures v... | Environmental | ManufacturerD | ModelD | 2024-02-01 | 13 |
| 27 | 6 | Rotation Sen... | Measures r... | Mechanical | ManufacturerE | ModeE | 2024-02-01 | 9 |

**Obrázek 8-7 Data z tabulky Sensors**

Nutné bylo také doplnit jednotky veličin, přesněji Vibrace, Kapacita a Průtok.

| unit_id | unit_name | symbol |
|---|---|---|
| 1 | Pascal | Pa |
| 2 | Volt | V |
| 3 | Ampere | A |
| 4 | Watt | W |
| 5 | Celsius (Temperature) | °C |
| 6 | Percentage | % |
| 7 | Bar | bar |
| 8 | Newton | N |
| 9 | RPM (Revolutions per Minute) | rpm |
| 10 | Lux | lux |
| 11 | Watts per meter square | W/m2 |
| 13 | Vibration | Hz |
| 14 | Capacity | Ks |
| 15 | Flow Rate | m^3/s |

**Obrázek 8-8 Doplněná tabulka Units_of_Measurements**

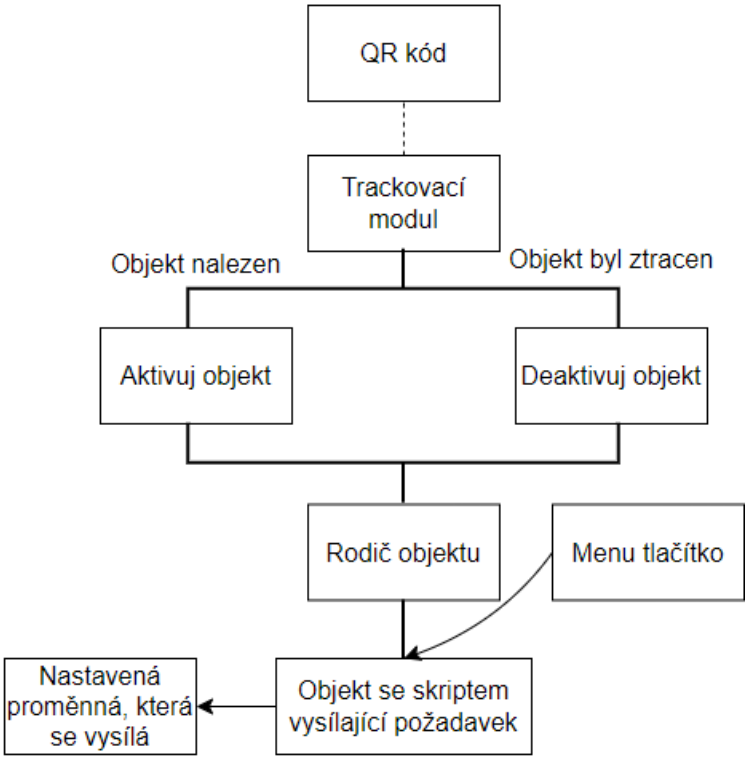
Níže bylo otestováno, zda se všechny hodnoty správně generují a propisují do databáze

| reading_id | sensor_id | unit_id | value | timestamp |
|---|---|---|---|---|
| 1042 | 1 | 3 | 127 | 2024-04-30 13:03:17 |
| 1043 | 2 | 1 | 178 | 2024-04-30 13:03:17 |
| 1044 | 3 | 5 | 56 | 2024-04-30 13:03:17 |
| 1045 | 15 | 1 | 152 | 2024-04-30 13:03:17 |
| 1046 | 14 | 3 | 119 | 2024-04-30 13:03:17 |
| 1047 | 16 | 5 | 65 | 2024-04-30 13:03:17 |
| 1048 | 1 | 3 | 129 | 2024-04-30 13:03:20 |
| 1049 | 2 | 1 | 191 | 2024-04-30 13:03:20 |
| 1050 | 3 | 5 | 44 | 2024-04-30 13:03:20 |
| 1051 | 15 | 1 | 204 | 2024-04-30 13:03:20 |
| 1052 | 14 | 3 | 139 | 2024-04-30 13:03:20 |
| 1053 | 16 | 5 | 56 | 2024-04-30 13:03:20 |
| 1054 | 2 | 1 | 208 | 2024-04-30 13:03:23 |
| 1055 | 1 | 3 | 116 | 2024-04-30 13:03:23 |
| 1056 | 3 | 5 | 39 | 2024-04-30 13:03:23 |
| 1057 | 15 | 1 | 184 | 2024-04-30 13:03:23 |
| 1058 | 14 | 3 | 132 | 2024-04-30 13:03:23 |
| 1059 | 16 | 5 | 58 | 2024-04-30 13:03:23 |
| 1060 | 1 | 3 | 122 | 2024-04-30 13:03:26 |
| 1061 | 2 | 1 | 202 | 2024-04-30 13:03:26 |
| 1062 | 3 | 5 | 48 | 2024-04-30 13:03:26 |
| 1063 | 15 | 1 | 194 | 2024-04-30 13:03:26 |
| 1064 | 14 | 3 | 112 | 2024-04-30 13:03:26 |
| 1065 | 16 | 5 | 48 | 2024-04-30 13:03:26 |
| 1066 | 2 | 1 | 181 | 2024-04-30 13:03:29 |

**Obrázek 8-9 Data z tabulky Sensor_Readings**

## 8.3 Upravení žádosti na základě objektu

S více objekty je třeba upravit způsob zasílání požadavku na MQTT server, jelikož každý objekt má jiné parametry, tj. jiné senzory. Tudíž je nutné aktivovat rozdílné SQL dotazy pro každý objekt. Způsob, jakým toto bylo vyřešeno je vidět na obrázku Obrázek 8-10 Příklad nastavení Event Listu Image Targetu.




**Obrázek 8-10 Příklad nastavení Event Listu Image Targetu**

Každý Image Target má jiný objekt, který aktivuje. Tento objekt je rodič objektu, který má na sobě podobný skript jako DeviceListManager, dostupný v příloze 7. Při nalezení (či ztracení) QR kódu se spustí kroky, které tyto rodiče (de)aktivují. Samotný dceřiný objekt je sám spravovaný příslušným tlačítkem, který ho aktivuje.

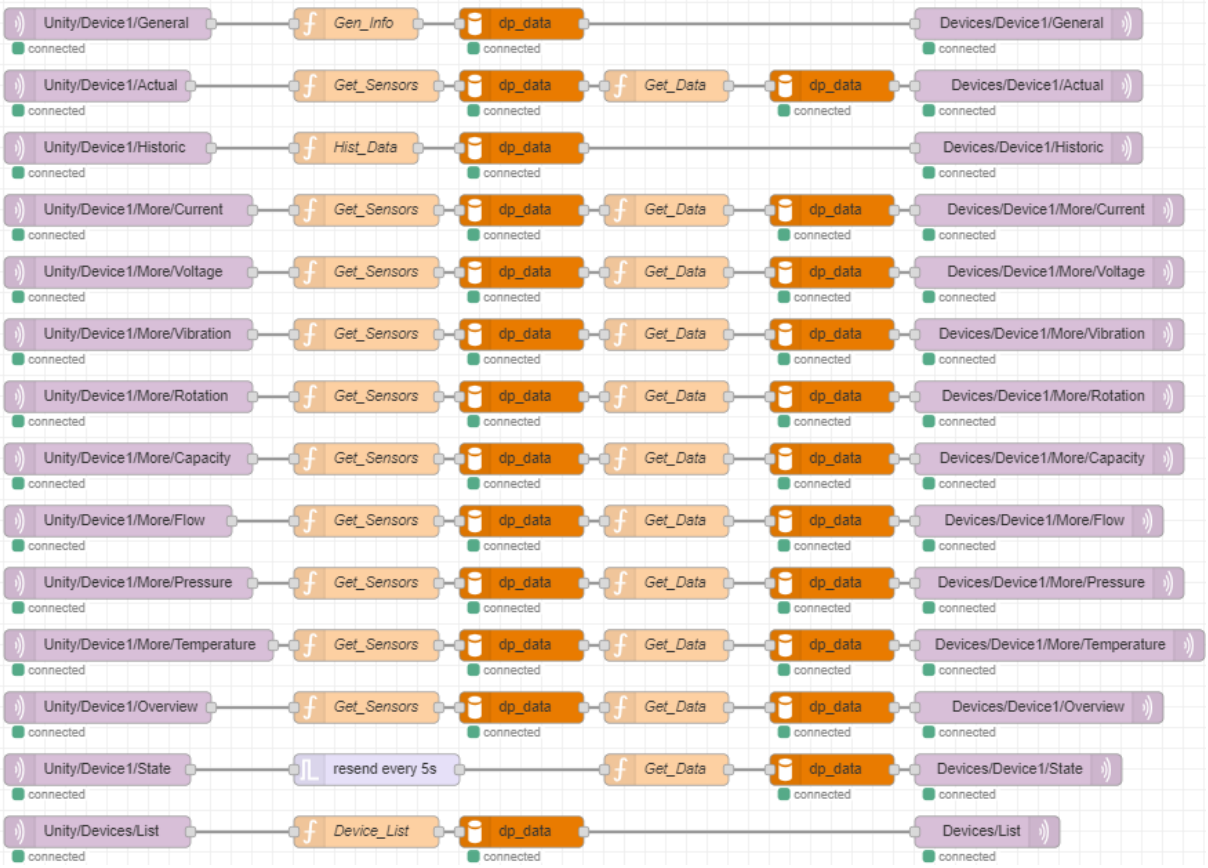
Zmiňovaný skript ale dokáže posílat pouze zprávu, která je zadaná v mqttManageru, je proto nutné ho upravit, viz níže. Úprava zahrnuje vytvoření veřejné proměnné (je tedy možné ji měnit mimo skript), která se převede do formátu String a nastaví se jako vysílaná zpráva.

```
public int DeviceID;

 public void PublishMessage()
 {
     _eventSender.messagePublish = DeviceID.ToString();

     _eventSender.Publish();

     Debug.Log("Message sent");
 }
```

Podobně se upraví zbytek tabulek, které vysílají požadavek.

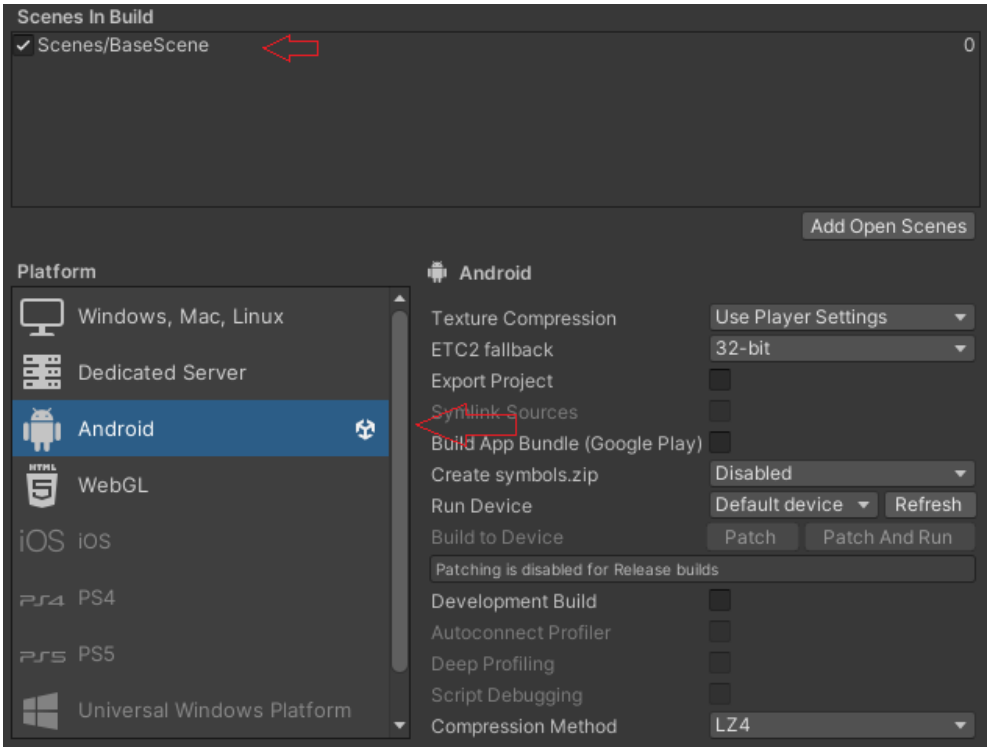
## 8.4 Doplnění SQL dotazů v rámci NR

Po umožnění vysílání různých požadavků na MQTT server, je třeba také upravit SQL dotazy v rámci NR. SQL dotaz byl lehce upraven tak, aby přijal číslo získané z MQTT, které koresponduje s ID zařízení v databázi. Podle tohoto čísla se načtou potřebné senzory a data z nich.




**Obrázek 8-11 Zpracování požadavku v NR**

## 8.5 Vytvoření APK balíčku pro instalaci

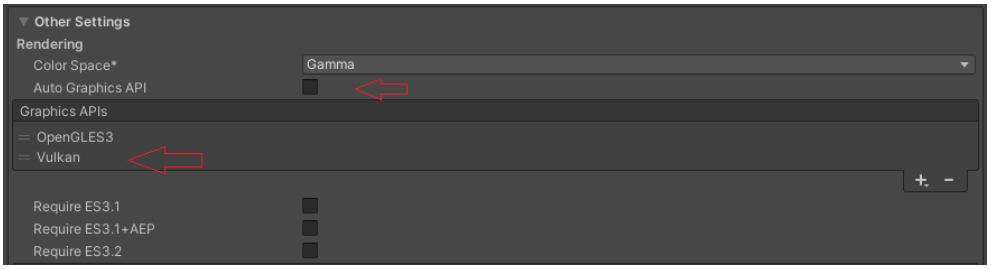



**Obrázek 8-12 Platforma Android**

Jelikož je aplikace tvořená s úmyslem využití na systému Android, je nutné vybrat tuto platformu. Dále je nutné vybrat scénu, která se má zpracovat.

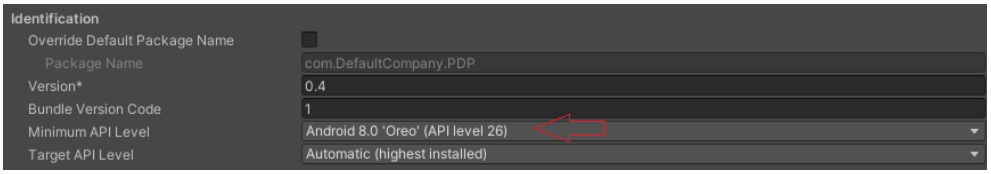

**Obrázek 8-13 Název a verze aplikace**

Nastavení názvu ulehčí jednoznačnou identifikaci aplikace. Změna verze rovněž tak umožní sledovat vývoj aplikace.



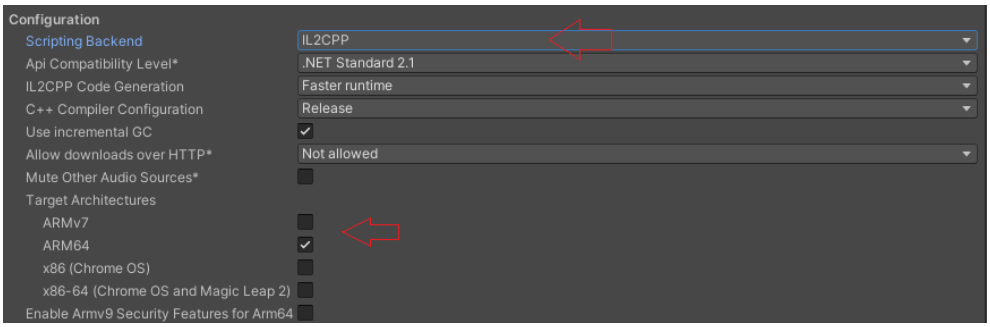

**Obrázek 8-14 Nastavení grafiky**

Dále je nutné odstranit grafické API Vulkan, které se zobrazí po deaktivování položky „Auto Graphics API“. Vulkan Graphics API je moderní, vysoko výkonné rozhraní pro práci s grafikou, které přináší výhody jako je nižší úroveň přístupu k hardwaru, lepší využití více jádrových procesorů a lepší správu paměti. Nicméně, Vulkan není podporován na všech zařízeních a platformách. Některá zařízení nebo platformy mohou mít omezenou nebo žádnou podporu pro Vulkan, což může způsobit problémy při spouštění aplikací vytvořených v Unity s tímto API.




**Obrázek 8-15 Nastavení minimální verze systému Android**

Nastavení správného minimálního API levelu v Unity je důležité z několika důvodů. Specifikováním minimálního API levelu se definuje nejstarší verze operačního systému, kterou aplikace podporuje. To zajišťuje, že aplikace bude kompatibilní s co největším počtem zařízení. Pokud je nastavený příliš vysoký minimální API level, může vyloučit starší zařízení, která nejsou schopna spustit novější verze operačního systému. Různé verze API nabízejí různé funkce a optimalizace. Nastavením vyššího API levelu lze využít pokročilejší funkce a optimalizace dostupné ve vyšších verzích operačního systému. Novější verze API obvykle zahrnují opravy chyb a zlepšení zabezpečení. Nastavením vyššího minimálního API levelu se zajistí, že aplikace bude chráněna proti známým zranitelnostem a bude mít přístup k nejnovějším bezpečnostním funkcím poskytovaným operačním systémem.




**Obrázek 8-16 Nastavení překladače a architektury**

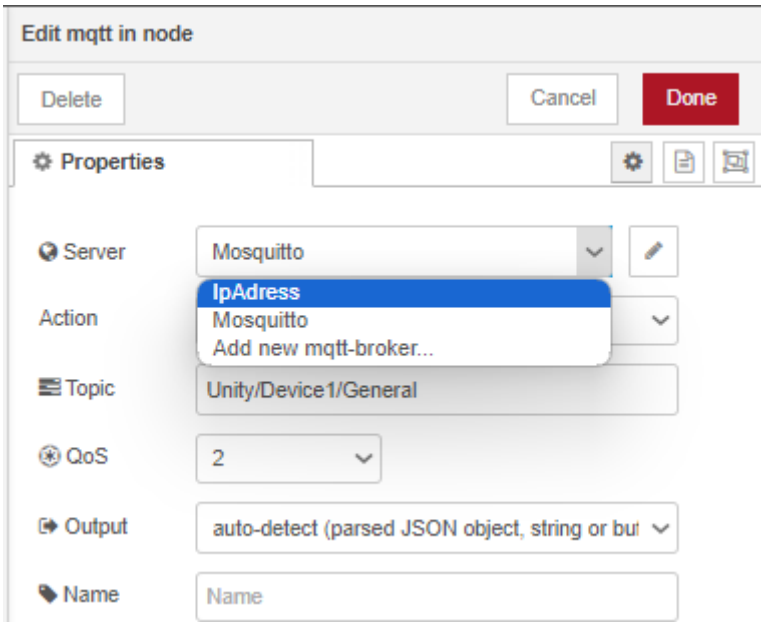
IL2CPP je zkratka pro "Intermediate Language to C++". Jedná se o překladač vyvinutý společností Unity Technologies, který transformuje kód napsaný v jazyce C# (a dalších jazycích podporovaných platformou Unity) do jazyka C++. Tento překlad umožňuje aplikacím vytvořeným v Unity běžet na různých platformách, jako jsou například iOS, Android, Windows, macOS a další. IL2CPP je také známý svou schopností generovat efektivní a optimalizovaný kód, což může vést k lepší výkonu a menší paměťové náročnosti aplikací.[80]

ARM64 je zkratka pro 64bitovou architekturu procesorů založených na technologii ARM[4]. Tato architektura je používána v mnoha moderních zařízeních, jako jsou chytré telefony, tablety, přenosné počítače, IoT zařízení a další. Procesory ARM64 jsou navrženy tak, aby poskytovaly vysoký výkon při efektivním využití energie, což je důležité pro mobilní zařízení s omezenou kapacitou baterie. Tato architektura se také stala stále populárnější i v segmentech jako jsou serverová infrastruktura a superpočítače díky svému výkonu a energetické účinnosti.[81]

## 8.6 Nutné kroky pro migraci aplikace do reálného prostředí

Prvním logickým krokem je využití reálných dat ze senzorů. Je možné využít jako řídící jednotku např. Raspberry Pi, velmi oblíbený a relativně levný jednodeskový počítač. Tento mini počítač může sloužit jako IoT server, kde komunikace je založena na MQTT protokolu. MQTT server vytvořený v diplomové práci je založený lokálně, je tedy nutné ho změnit na základě IP adresy. Rovněž i mqttManagery v rámci prostředí Unity a uzle v NR je třeba upravit. V mqttManageru se změní localhost na hostovací IP adresu. V NR je třeba vytvořit nový MQTT server, ve kterém se rovněž nastaví stejná IP adresa. Takto vytvořený server se musí nastavit u všech uzlů.

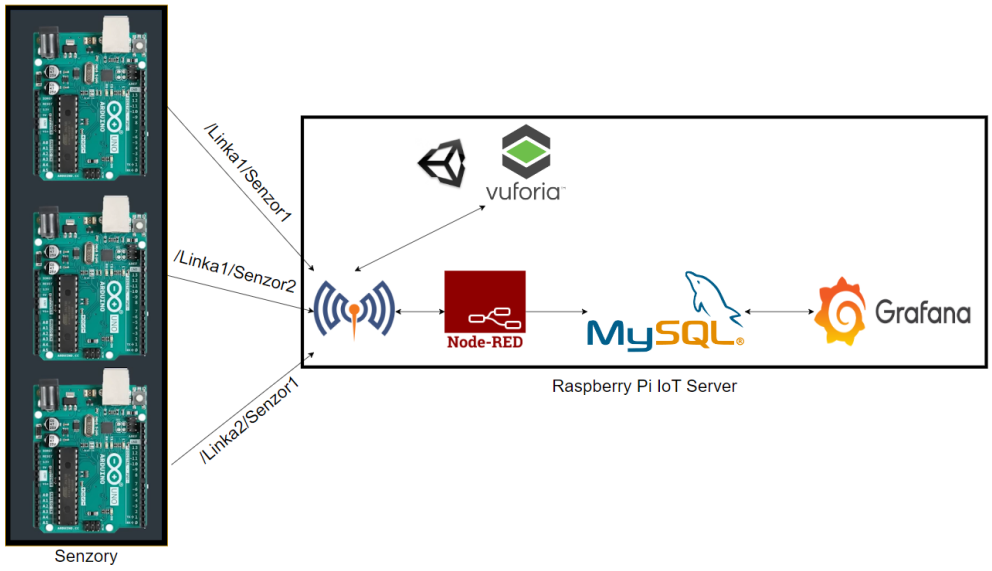
[4] Advanced RISC Machine




**Obrázek 8-17 Změna MQTT serveru v rámci NR**

Komunikace mezi NR a MySQL je beze změny. V rámci samotné databáze je ale nutné vytvořit uživatelské účty. Aplikace v této podobě využívá adminovský účet, což pro víceuživatelský přístup je nevhodné. Jelikož má adminovský účet všechna práva, má přístup k celé databázi a všem SQL příkazům. Je třeba vytvořit uživatelský účet, který bude mít přístup pro vybrané tabulky a příkaz SELECT. Zamezí se tak jakémukoli poškození integrity dat.

V rámci front-end řešení pro Unity-Vuforia bude třeba dodat augmentaci dle potřeby. Pokud bude architektura databáze stejná (tj. stejné názvy sloupců), není třeba ani měnit vytvořené skripty. Pro front-end řešení mimo AR je možné využít softwaru Grafana. Grafana je open-source platforma pro vizualizaci a analýzu dat, která umožňuje uživatelům vytvářet interaktivní a dynamické grafy a dashboardy




**Obrázek 8-18 Návrh řešení IoT serveru**

Pro ulehčení práce pro spouštění a správu jednotlivých částí je možné využít Docker. Docker je platforma pro kontejnerizaci, která umožňuje vývojářům balit své aplikace a všechny jejich závislosti do standardizovaných kontejnerů. Tyto kontejnery lze pak spouštět na jakékoli platformě, která podporuje Docker, bez ohledu na operační systém nebo infrastrukturu. Docker

poskytuje jednoduché rozhraní pro správu a nasazení kontejnerizovaných aplikací, což usnadňuje vývoj, testování a nasazování softwaru.

IoTStack je konkrétní sada nástrojů a softwaru, která využívá Docker pro implementaci a správu IoT aplikací a infrastruktury. Tato sada nástrojů obsahuje různé komponenty a služby, které jsou často používány v IoT prostředí, jako jsou časové řadové databáze, vizualizační nástroje, správci zařízení a další. IoTStack tedy využívá sílu Dockeru pro kontejnerizaci jednotlivých komponent a služeb, což umožňuje snadnou instalaci, konfiguraci a správu celé IoT infrastruktury. Tímto způsobem se minimalizuje složitost nasazení a integrace různých prvků v IoT prostředí.

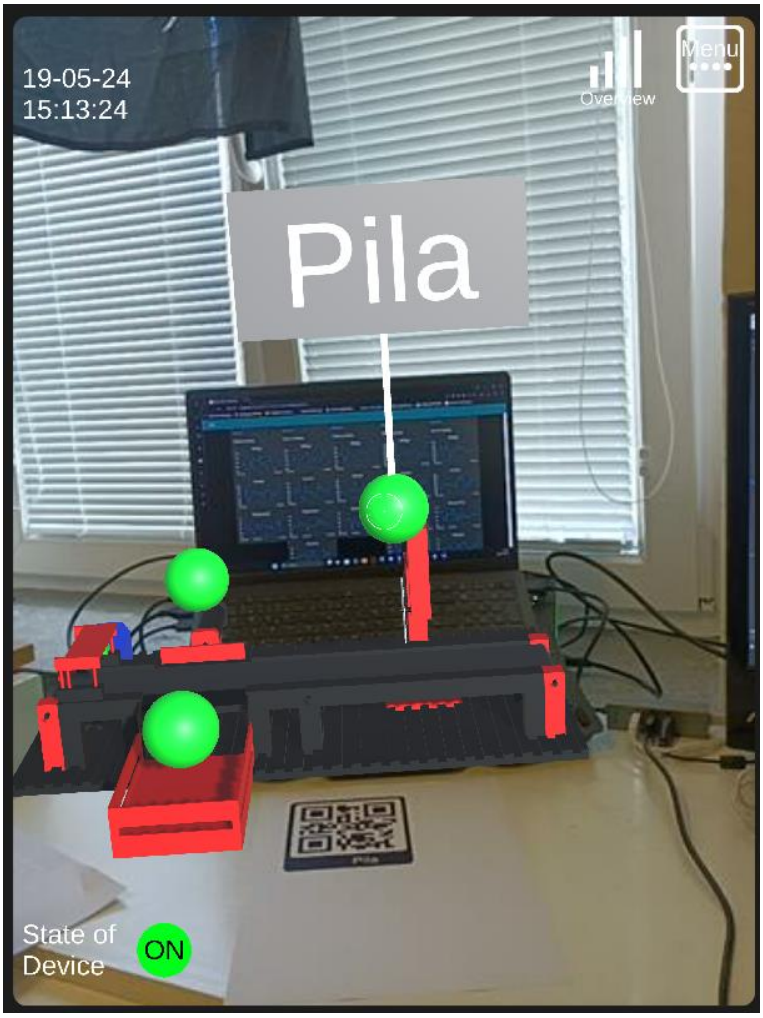
Při instalaci IoTStack je třeba nainstalovat všechny potřebné komponenty, tedy:

- Node-Red
- MySQL
- Mosquitto
- Grafana
- Portainer-ce

Portainer CE je open-source platforma pro správu kontejnerů, která poskytuje uživatelsky přívětivé rozhraní pro správu Docker kontejnerů a jejich prostředí. Jedná se o nástroj, který umožňuje uživatelům jednoduše spravovat jejich kontejnery, obrázky, sítě a další prvky prostřednictvím webového rozhraní.

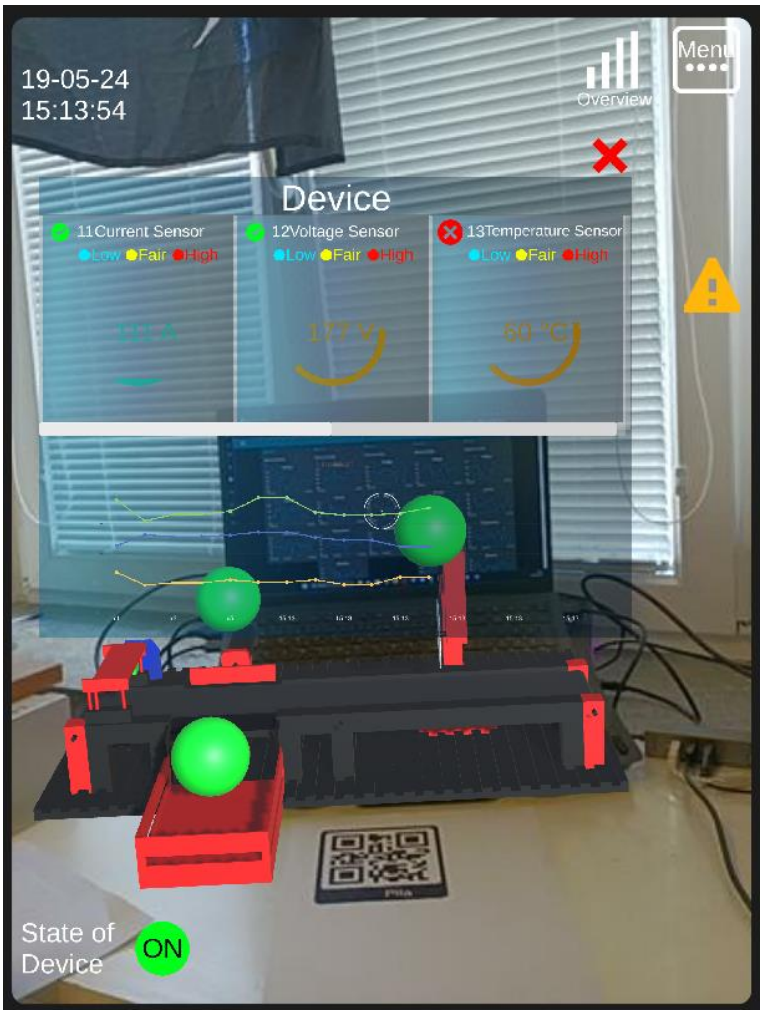
# 9 Zhodnocení druhé iterace aplikace

Druhá iteraci aplikace umožňuje získávat data o různých objektech. Je zde tedy celkem 5 objektů: PS4, Pila, Sklad, Svářečka a Frézka/Vrtačka. Každý tento objekt je reprezentovaný odpovídajícím QR kódem, jak je vidět níže.



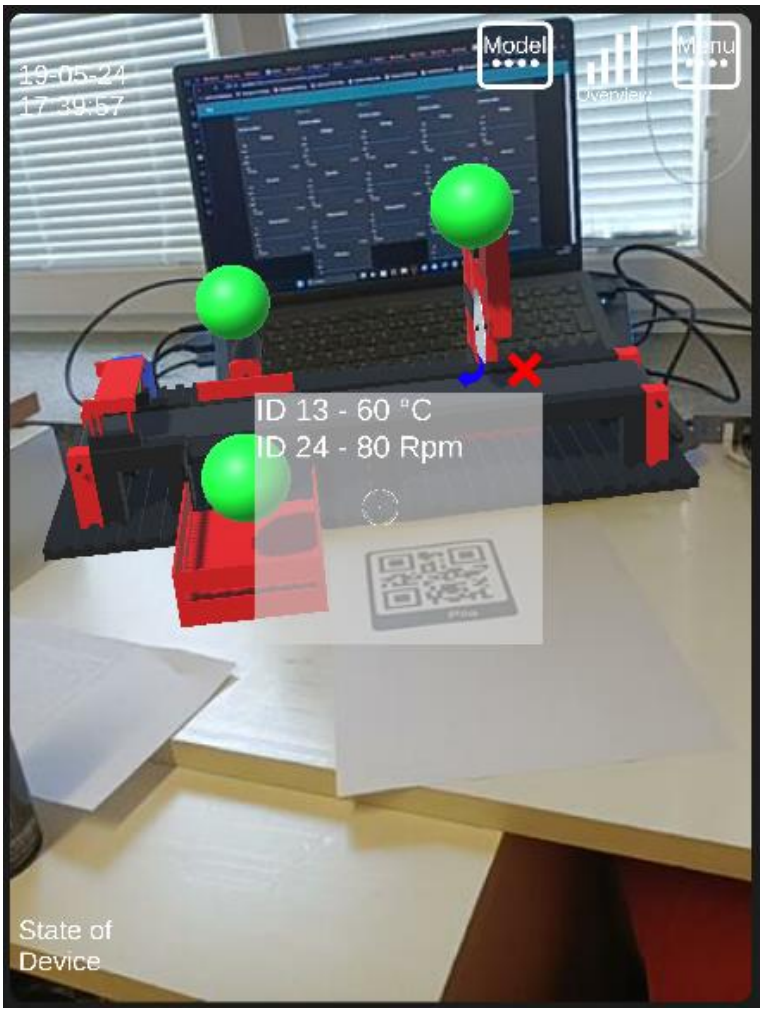

**Obrázek 9-1 Nalezení objektu včetně zobrazení Pinu**

Každý objekt je rovněž doplněný interaktivními sférami či „Piny“, které zobrazí informaci při pohledu na ně. Pro jednodušší zaměření pinu byl vytvořena muška, která ukazuje kam se kamera „dívá“. Na obrazovce lze rovněž vidět datum, čas a stav zařízení.



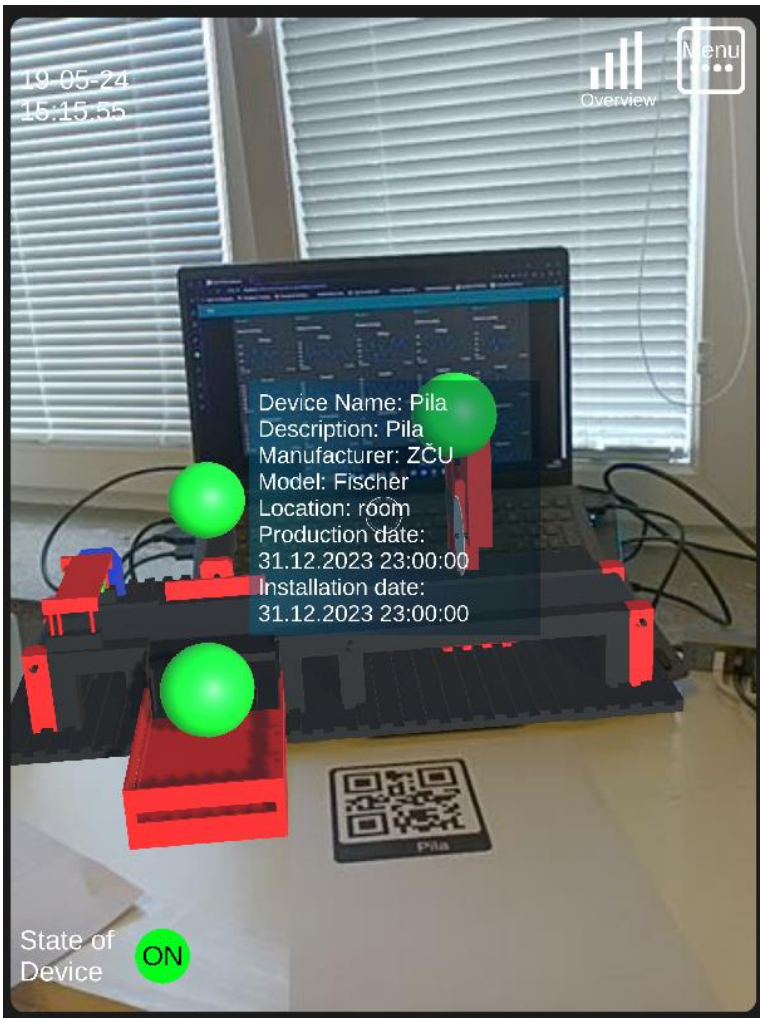

**Obrázek 9-2 Zobrazení panelu Overview**

Obrázek 9-3 je panel Overview, který zobrazuje aktuální data v číselném a grafickém vyjádření. Samotné okno má posuvník pro zobrazení zbylých senzorů. Pokud některá z hodnot přesáhne doporučenou hodnotu, jako je tomu v případě senzoru teploty č. 13, zobrazí se červený křížek a vyskočí upozornění na obrazovce. Tuto výstrahu lze zobrazit *Menu -> Event List -> Warning List* nebo jednoduše kliknutím na výstrahu. Níže je tato výstraha vidět.



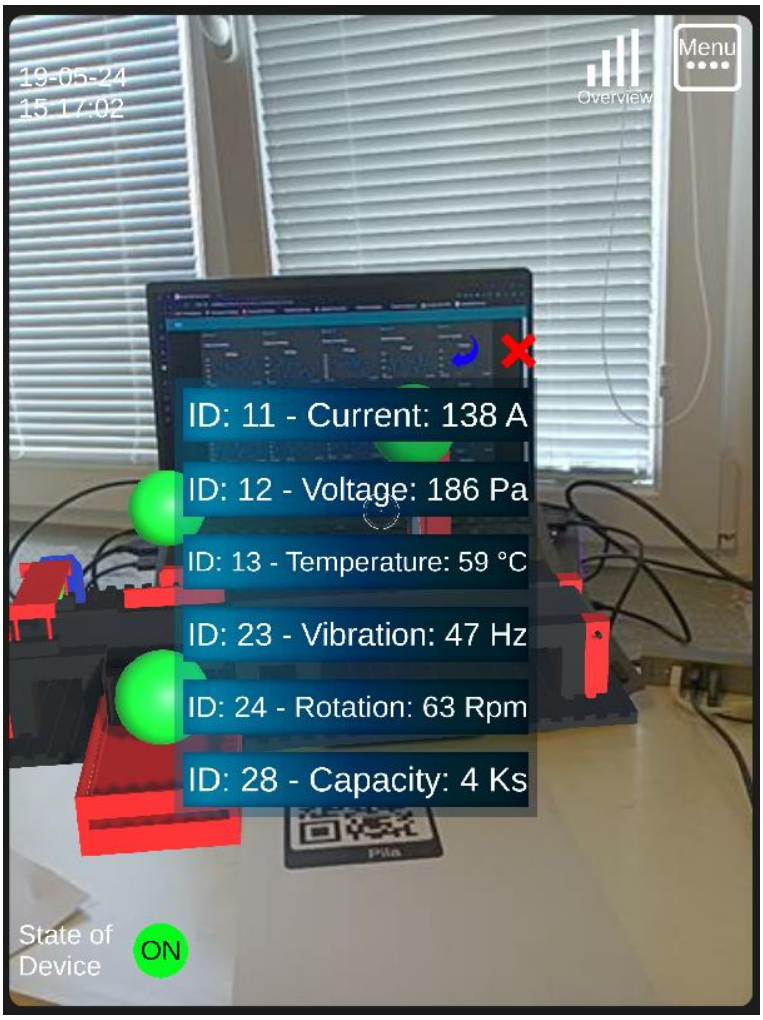

**Obrázek 9-3 Zobrazení výstrahy**

Do tohoto panelu se propisují všechny výstrahy zařízení. Je tu vidět zmiňovaný senzor 13, ale rovněž i senzor 24, který měří otáčky. Pokud by toto byla skutečná data, bylo možné odhalit důvod ke zvýšeným hodnotám a včas reagovat, čímž se podnik může posouvat směrem k prediktivní údržbě.




**Obrázek 9-4 Obecné informace o zařízení**

Při otevření Model_menu pomocí tlačítka model, nebo po kliknutí na samotný model na obrazovce, je možné zobrazovat všechny dostupné informace – Obecné informace o zařízení k vidění na Obrázek 9-4 Obecné informace o zařízení, Aktuální data na Obrázek 9-5 Přehled senzorů zařízení a Historická data na Obrázek 9-8 Sestrojený graf na základě požadavku.



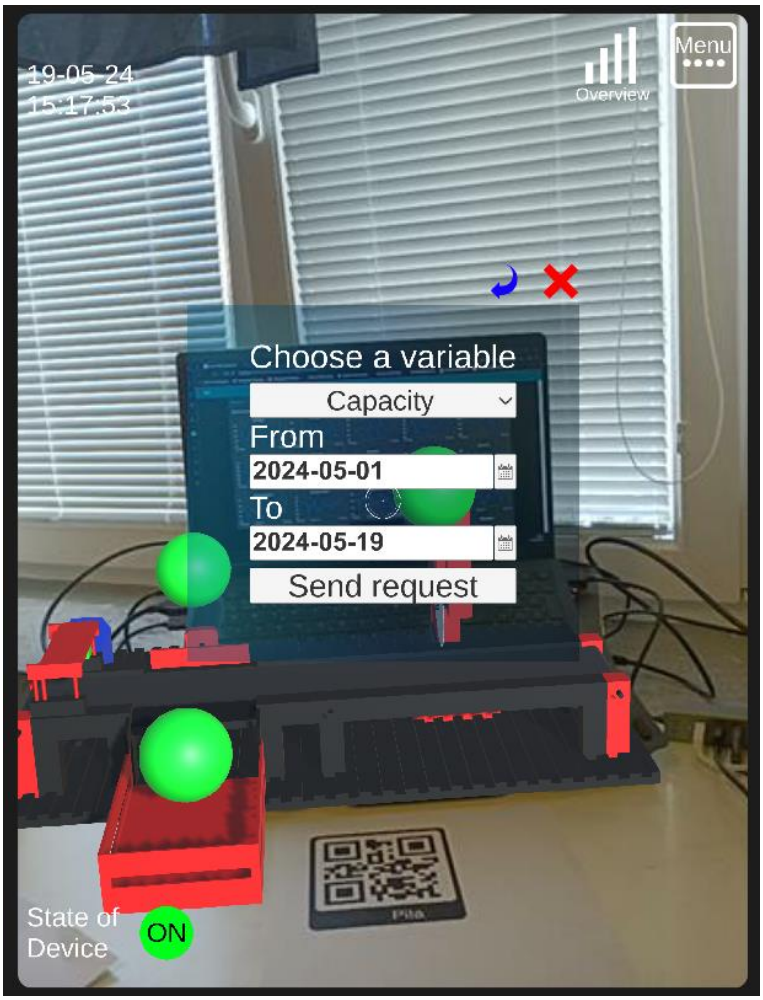

**Obrázek 9-5 Přehled senzorů zařízení**

Panel Aktuální Data zobrazuje všechny senzory na zařízení, jejich ID a zobrazuje jejich aktuální hodnotu. Tento panel tedy funguje podobně jako Overview, jelikož podává přehled o senzorech. Rozdíl je v tom, že jednotlivé senzory fungují jako tlačítka.


**Obrázek 9-6 Dodatečné informace o senzoru**

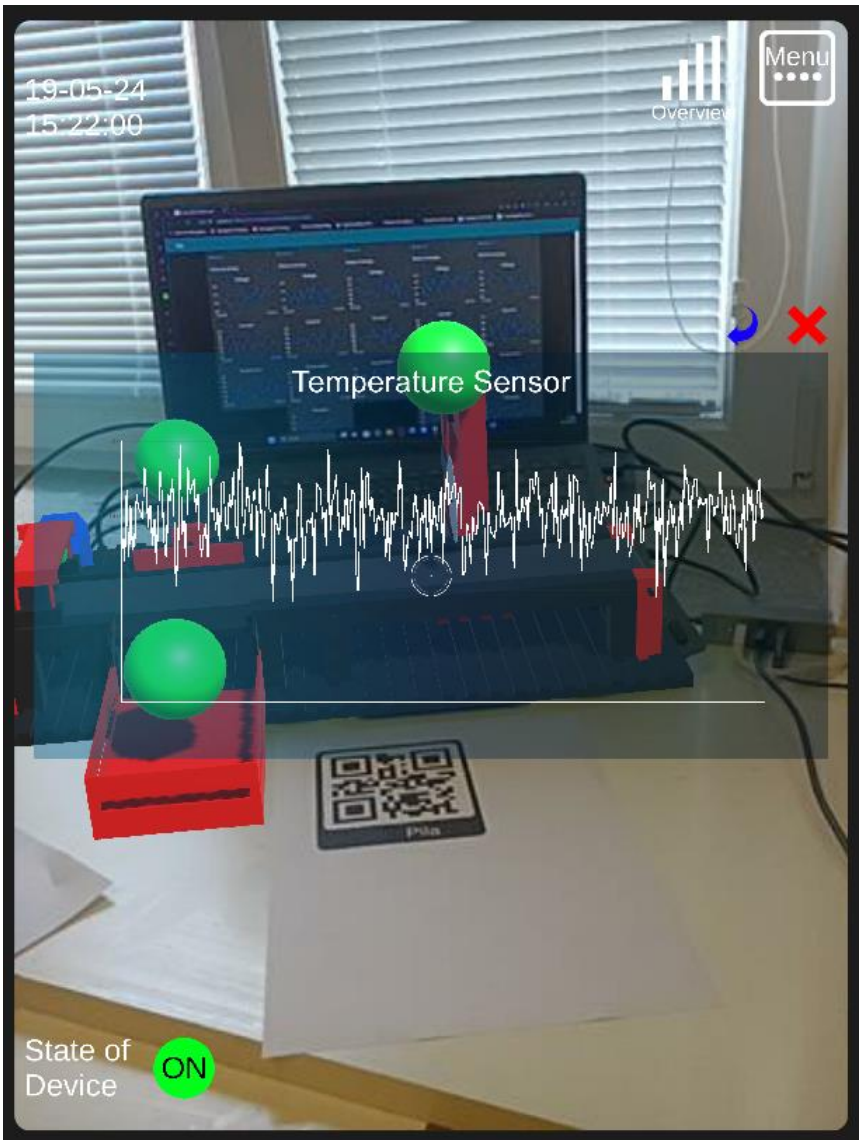
Po kliknutí na libovolný senzor se zobrazují dodatečné informace jako denní průměr, maximum a minimum, včetně již zobrazených informací jako je název senzoru a jeho ID.

Panel Historická Data umožňuje výběr požadované proměnné a datového úseku. Jsou zde tedy celkem tři interaktivní prvky. První je rozbalovací seznam, který zobrazí všechny dostupné proměnné dostupné k zařízení. Následují dva rozbalovací kalendáře kde uživatel vybere požadované datum. Po výběru všech třech nutných polí se odemkne tlačítko Send Request, které pošle požadavek a otevře panel s grafem.




**Obrázek 9-7 Výběr historických dat**

Na základě požadavku se vyberou požadovaná data z databáze, připravený skript je zpracuje a zobrazí v grafu.




**Obrázek 9-8 Sestrojený graf na základě požadavku**

Tato aplikace představuje pokročilý a inovativní přístup k zobrazování průmyslových metadat v rozšířené realitě. Kombinace Unity a Vuforia poskytuje silný základ pro AR funkce, MQTT zajišťuje efektivní komunikaci, MySQL spolehlivé ukládání dat a Node-RED flexibilitu v integraci a správě datových toků. I přesto, že jednotlivé složky aplikace je jednoduché nastavit, jednou z výzev j komplexita a zajištění vysokého výkonu a bezpečnosti, jelikož aplikace integruje více technologií. AR aplikace mohou být náročné na výkon, a je třeba optimalizovat jak grafiku, tak síťovou komunikaci. Co se týče zmiňované grafiky je třeba aplikaci „uhladit“ jelikož bylo úsilí směřováno zejména na funkčnost, grafická stránka je spíše strohá, kromě panelu Overview. I přesto že to nebyl významný cíl práce, bylo velkým zklamáním nefunkční trackování na základě modelu, které by velmi urychlilo uvedení do provozu, jelikož trackování na základě značek je časově náročně, i přes zvýšenou stabilitu. I přesto zmíněné nedostatky má tento projekt potenciál výrazně zlepšit efektivitu a interaktivitu průmyslových procesů.

# Závěr

Tato diplomová práce se skládá z teoretické a praktické části. Pro vysvětlení pojmů Průmyslová metadata a Rozšířená reality byla použita literatura z odborných publikací. Dále byli popsány jednotlivé moduly využité pro vytvoření aplikace, jež v této práci slouží pro simulaci reálného prostředí, spolu s jednotlivými výrobními stroji, které jsou reprezentovány značkou. Následovalo získání a nastavení těchto softwarů. Softwary byly porovnány jak z pohledu informací udávaných výrobcem a hodnocení ostatních uživatelů, tak z pohledu zkušeností autora této práce. V softwarech MySQL byla vytvořena databáze, která spravovala všechna vygenerovaná data ze „senzorů“ a informace o strojích. Nástroj Node-red byl nastaven tak, aby mohl simulovat hodnoty a zapisovat je do databáze. Rovněž byl nastaven tak, aby mohl reagovat na požadavky z prostředí Unity. Následně byla vytvořena aplikace v Unity, a pomocí plug-inu Vuforia bylo umožněno využívat tuto aplikace pro rozšířenou realitu.

Hlavním praktickým výstupem této práce je aplikace, která v rámci rozšířené reality dokáže vizualizovat generovaná data, a to včetně komunikace s databází. Byl také navržený způsob, jak tuto aplikaci migrovat pro využití v reálném prostředí a navrhnuty kroky pro další výzkum. Výstup práce bude také sloužit jako návod v této oblasti v rámci dizertační práce.

Lze konstatovat, že byla vytvořena snadno rozšiřitelná aplikace pro zobrazování průmyslových metadat v rámci rozšířené reality. Skladování průmyslových dat bylo zajištěno vytvořením databáze v MySQL. MQTT broker slouží jako komunikační prostředek mezi všemi moduly aplikace. Samotné jádro aplikace je pak vytvořené v rámci prostředí Unity s využitím plug-inu Vuforia. Při vývoji aplikace bylo také nutné vytvořit vlastní API, které bylo zpracováno v nástroji Node-RED. Toto API je snadno modifikovatelné, tudíž není problém jednoduše měnit a doplňovat požadované funkce. Bylo otestováno trackování na základě modelu, které se zatím jeví jako nestabilní. Je tedy nutné v tomto ohledu provést dodatečný výzkum a testování, jelikož přesné dimenzování modelu pro značku je velmi složité.

# Seznam použitých zdrojů

[1] MACHADO, C. a J.P. DAVIM. *Industry 4.0: Challenges, Trends, and Solutions in Management and Engineering* [online]. B.m.: CRC Press, 2020. Manufacturing design and technology series. ISBN 978-0-8153-5440-6. Dostupné z: https://books.google.cz/books?id=LN_htAEACAAJ

[2] FUNARTECH. *Industry 5.0* [online]. [vid. 2024-01-28]. Dostupné z: https://www.funartech.com/approach/industry5

[3] PASCUAL, D.G., P. DAPONTE a U. KUMAR. *Handbook of Industry 4.0 and SMART Systems* [online]. B.m.: CRC Press, 2019. ISBN 978-0-429-84968-8. Dostupné z: https://books.google.cz/books?id=ExGwDwAAQBAJ

[4] STEVEN M. LAVALLE. *VIRTUAL REALITY*. B.m.: Cambridge University Press. 2023

[5] MILGRAM, Paul a Fumio KISHINO. A Taxonomy of Mixed Reality Visual Displays. *IEICE Trans. Information Systems*. 1994, **E77-D, č. 12**, 1321–1329.

[6] AZUMA, Ronald T. A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments* [online]. 1997, **6**(4), 355–385. Dostupné z: doi:10.1162/pres.1997.6.4.355

[7] BROUM, Tomáš, Petr HOŘEJŠÍ, Miroslav MALAGA a Pierre GRZONA. Competencies of Industrial Engineers for Implementing Augmented Reality Metadata Systems. *Sustainability* [online]. 2023, **15**(1). ISSN 2071-1050. Dostupné z: doi:10.3390/su15010130

[8] HUANG, Zhanpeng, Pan HUI, Christoph PEYLO a Dimitris CHATZOPOULOS. Mobile augmented reality survey: a bottom-up approach. 2013.

[9] CHATZOPOULOS, Dimitris, Carlos BERMEJO, Zhanpeng HUANG a Pan HUI. Mobile Augmented Reality Survey: From Where We Are to Where We Go. *IEEE Access* [online]. 2017, **5**, 6917–6950. Dostupné z: doi:10.1109/ACCESS.2017.2698164

[10] DOWNEY, Sarah. *The Gigantic List of Augmented Reality Use Cases* [online]. 25. říjen 2016 [vid. 2024-01-28]. Dostupné z: https://www.uploadvr.com/augmented-reality-use-cases-list/

[11] MÜLLNER, W. *Potentiale, Risiken Und Grenzen Von „Augmented Reality"* [online]. B.m.: AV Akademikerverlag, 2013. ISBN 978-3-639-46180-0. Dostupné z: https://books.google.cz/books?id=aalKlQEACAAJ

[12] ROLLAND, Jannick, Yohan BAILLOT a Alexei GOON. A survey of tracking technology for virtual environments. *Fundamentals of Wearable Computers and Augmented Reality*. 2001.

[13] DIAZ, Estefania Munoz, Dina Bousdar AHMED a Susanna KAISER. 16 - A Review of Indoor Localization Methods Based on Inertial Sensors. In: Jordi CONESA, Antoni PÉREZ-NAVARRO, Joaquín TORRES-SOSPEDRA a Raul MONTOLIU, ed. *Geographical and Fingerprinting Data to Create Systems for Indoor Positioning and Indoor/Outdoor Navigation* [online]. B.m.: Academic Press, 2019, Intelligent Data-Centric Systems, s. 311–333. ISBN 978-0-12-813189-3. Dostupné z: doi:https://doi.org/10.1016/B978-0-12-813189-3.00016-2

[14] KLEIN, G. *Visual Tracking for Augmented Reality: Edge-based Tracking Techniques for AR Applications* [online]. B.m.: VDM Publishing, 2009. ISBN 978-3-639-07891-6. Dostupné z: https://books.google.cz/books?id=EDRbPgAACAAJ

[15] MEHLER-BICHER, Anett a Lothar STEIGER. *Theorie und Praxis* [online]. München: De Gruyter Oldenbourg, 2014 [vid. 2024-05-18]. ISBN 978-3-11-035385-3. Dostupné z: doi:doi:10.1524/9783110353853

[16] TANG, Arthur, CHARLES B OWEN, Frank BIOCCA a Weimin MOU. Experimental Evaluation of Augmented Reality in Object Assembly Task. In: *IEEE / ACM International Symposium on Mixed and Augmented Reality* [online]. Damstadt, Germany: IEEE, 2002, s. 265–266. Dostupné z: doi:10.1109/ISMAR.2002.1115105

[17] ZHANG, Xiang, S. FRONZ a N. NAVAB. Visual marker detection and decoding in AR systems: a comparative study. In: *Proceedings. International Symposium on Mixed and Augmented Reality* [online]. 2002, s. 97–106. Dostupné z: doi:10.1109/ISMAR.2002.1115078

[18] GARRIDO-JURADO, Sergio, Rafael MUÑOZ-SALINAS, Francisco MADRID-CUEVAS a Manuel MARÍN-JIMÉNEZ. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition* [online]. 2014, **47**, 2280–2292. Dostupné z: doi:10.1016/j.patcog.2014.01.005

[19] TÖNNIS, Marcus. *Augmented Reality*. B.m.: Springer Berlin, Heidelberg, 2010. ISBN 978-3-642-14178-2.

[20] UENOHARA, M. a T. KANADE. Vision-based object registration for real-time image overlay. *Computers in Biology and Medicine* [online]. 1995, **25**(2), 249–260. ISSN 0010-4825. Dostupné z: doi:https://doi.org/10.1016/0010-4825(94)00045-R

[21] ZHOU, Feng, Henry DUH a Mark BILLINGHURST. Trends in Augmented Reality Tracking, Interaction and Display: A Review of Ten Years of ISMAR. *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality* [online]. 2008, **2**, 193–202. Dostupné z: doi:10.1109/ISMAR.2008.4637362

[22] BLESER-TAETZ, Gabriele, Cedric WOHLLEBER, Mario BECKER, Didier STRICKER a Fraunhofer IGD. Fast and stable tracking for AR fusing video and inertial sensor data. 2006.

[23] WAGNER, Daniel, Gerhard REITMAYR, Alessandro MULLONI, Tom DRUMMOND a Dieter SCHMALSTIEG. Real-Time Detection and Tracking for Augmented Reality on Mobile Phones. *IEEE transactions on visualization and computer graphics* [online]. 2010, **16**, 355–68. Dostupné z: doi:10.1109/TVCG.2009.99

[24] LIU, Haomin, Guofeng ZHANG a Hujun BAO. Robust Keyframe-based Monocular SLAM for Augmented Reality. In: *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)* [online]. 2016, s. 1–10. Dostupné z: doi:10.1109/ISMAR.2016.24

[25] SCOTT, D.M. *Industrial Process Sensors* [online]. B.m.: CRC Press, 2018. ISBN 978-1-351-83584-8. Dostupné z: https://books.google.cz/books?id=pH50DwAAQBAJ

[26] NIHTIANOV, S. a A. LUQUE. *Smart Sensors and MEMS: Intelligent Devices and Microsystems for Industrial Applications*. 2013.

[27] ALA. *Committee on Cataloging: Description & Access (CC:DA)* [online]. [vid. 2023-09-23]. Dostupné z: https://alcts.ala.org/ccdablog/

[28] ORGANIZATION (U.S.), National Information Standards. *Understanding Metadata* [online]. B.m.: NISO Press, 2004. ISBN 978-1-880124-62-8. Dostupné z: https://books.google.cz/books?id=-QGPAAAACAAJ

[29] MARY S. WOODLEY, Gail CLEMENT a Pete WINN. *DCMI Glossary* [online]. Dostupné z: https://www.dublincore.org/specifications/dublin-core/usageguide/glossary/

[30] DUBLIN CORE. *Dublin Core™ Metadata Element Set, Version 1.1: Reference Description* [online]. Dostupné z: https://www.dublincore.org/specifications/dublin-core/dces/

[31] OCLC. *OCLC* [online]. [vid. 2023-10-26]. Dostupné z: https://www.oclc.org/en/home.html?redirect=true

[32] IPTC. *Photo Metadata* [online]. Dostupné z: https://iptc.org/standards/photo-metadata/

[33] FGDC. *CONTENT STANDARD FOR DIGITAL GEOSPATIAL METADATA* [online]. [vid. 2023-11-05]. Dostupné z: https://www.fgdc.gov/metadata/csdgm-standard

[34] TEI-C. *Guidelines* [online]. Dostupné z: https://tei-c.org/guidelines/

[35] ARCHIVISTS, SOCIETY OF AMERICAN. *Encoded Archival Description (EAD)* [online]. 2015. Dostupné z: https://www2.archivists.org/groups/technical-subcommittee-on-encoded-archival-standards-ts-eas/encoded-archival-description-ead

[36] TDWG. *Darwin Core* [online]. Dostupné z: https://dwc.tdwg.org

[37] EDITEUR. *ONIX (Online Information eXchange)* [online]. Dostupné z: https://www.ifla.org/references/best-practice-for-national-bibliographic-agencies-in-a-digital-age/resource-description-and-standards/metadata-formats/xml-formats/onix-online-information-exchange/

[38] IEEE Standard for Learning Object Metadata. *IEEE Std 1484.12.1-2020* [online]. 2020, 1–50. Dostupné z: doi:10.1109/IEEESTD.2020.9262118

[39] BRICKLEY, Dan a Libby MILLER. *FOAF Vocabulary Specification 0.99* [online]. Dostupné z: http://xmlns.com/foaf/spec/

[40] HORODYSKI, John. *Metadata Matters*. B.m.: Auerbach Publications, 2022. ISBN 978-1-00-059744-8.

[41] SCHEMA. *Welcome to Schema.org* [online]. Dostupné z: https://schema.org

[42] ANNE J. GILLILAND a Murtha BACA. *Introduction to metadata* [online]. 2016. vyd. B.m.: Getty Research Institute, nedatováno. Dostupné z: https://www.getty.edu/publications/intrometadata/

[43] GARTNER, Richard. *Metadata in the Digital Library: Building an Integrated Strategy with XML* [online]. 2021. ISBN 978-1-78330-486-8. Dostupné z: doi:10.29085/9781783304868

[44] MITCHELL, Erik. *Metadata Standards and Web Services in Libraries, Archives, and Museums: An Active Learning Resource*. B.m.: Bloomsbury Publishing, 2015. ISBN 978-1-61069-449-0.

[45] DIAN F. JOHN. *Fundamentals of Internet of Things: For Students and Professionals*. 2022. vyd. B.m.: Wiley, nedatováno. ISBN 978-1-119-84729-8.

[46] HARVEY, Phil. *exiftool* [online]. Dostupné z: https://exiftool.org

[47] ADOBE. *Adobe Bridge* [online]. [vid. 2023-10-07]. Dostupné z: https://www.adobe.com/cz/products/bridge.html

[48] MEDIAAREA. *MediaInfo* [online]. Dostupné z: https://mediaarea.net/en/MediaInfo

[49] ELEVEN PATHS. *FOCA - Fingerprinting Organizations with Collected Archives* [online]. Dostupné z: https://community.chocolatey.org/packages/foca

[50] ARGYLE. *AR for the jobsite* [online]. [vid. 2023-10-26]. Dostupné z: https://www.argyle.build

[51] SHENHAV, Maya a RELNOTES. *Data in space* [online]. 18. říjen 2022. Dostupné z: https://learn.microsoft.com/en-us/power-platform-release-plan/2022wave1/power-bi/data-space

[52] HELPLIGHTING. *HELP LIGHTNING FIELDBIT* [online]. [vid. 2023-11-16]. Dostupné z: https://helplightning.com/solutions/fieldbit-share-know-how-instantly/

[53] SCOPEAR. *scopear* [online]. Dostupné z: https://www.scopear.com/home

[54] GRZONA, Pierre a Broum TOMAS. Augmented Reality – Industrial use case. In: [online]. 2022, s. 26–33. Dostupné z: doi:10.24132/PI.2022.11146.026-033

[55] MORENO-LUMBRERAS, David, Jesus GONZALEZ-BARAHONA a Gregorio ROBLES. BabiaXR: Facilitating experiments about XR data visualization. *SoftwareX* [online]. 2023, **24**, 101587. Dostupné z: doi:10.1016/j.softx.2023.101587

[56] IN, Sungwon, Tica LIN, Chris NORTH, Hanspeter PFISTER a Yalong YANG. This is the Table I Want! Interactive Data Transformation on Desktop and in Virtual Reality. *IEEE transactions on visualization and computer graphics* [online]. 2023, **PP**. Dostupné z: doi:10.1109/TVCG.2023.3299602

[57] RATTANARUNGROT, Sasithorn a Martin WHITE. Development of service oriented mobile AR applications for museum learning activities. In: *2016 22nd International Conference on Virtual System & Multimedia (VSMM)* [online]. 2016, s. 1–8. Dostupné z: doi:10.1109/VSMM.2016.7863202

[58] BOUTSI, A.-M., I. TALLIS, I. PASTOS, S. VERYKOKOU a C. IOANNIDIS. 5DMETEORA FRAMEWORK: MANAGEMENT AND WEB PUBLISHING OF CULTURAL HERITAGE DATA. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* [online]. 2023, **X-M-1–2023**, 33–40. Dostupné z: doi:10.5194/isprs-annals-X-M-1-2023-33-2023

[59] G2A. *Where you go for software* [online]. Dostupné z: https://www.g2.com

[60] 6SENSE. *In-Depth Comparison* [online]. [vid. 2024-02-12]. Dostupné z: https://6sense.com

[61] OKITA, A. *Learning C# Programming with Unity 3D, Second Edition* [online]. B.m.: CRC Press, 2019. ISBN 978-1-138-33682-7. Dostupné z: https://books.google.cz/books?id=ctVSwwEACAAJ

[62] W3SCHOOLS. *w3schools* [online]. Dostupné z: https://www.w3schools.com/cpp/

[63] GLOVER, Jesse. *Unity 2018 augmented reality projects : build four immersive and fur AR applications using ARKit, ARCore, and Vuforia* [online]. Birmingham, UK: Packt Publishing, 2018. ISBN 978-1-78883-584-8. Dostupné z: https://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=1860852

[64] PTC VUFORIA. *PTC Vuforia* [online]. [vid. 2024-04-11]. Dostupné z: https://www.ptc.com/en/

[65] UNITY3D. *AR Foundation* [online]. Dostupné z: https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@6.0/manual/index.html

[66] WIKITUDE. *Wikitude* [online]. Dostupné z: https://www.wikitude.com/external/doc/documentation/latest/unity/

[67] EASYAR. *Easy AR* [online]. [vid. 2024-04-11]. Dostupné z: https://www.easyar.com

[68] GOOGLE. *Make the world your canvas* [online]. [vid. 2024-04-11]. Dostupné z: https://developers.google.com/ar

[69] APPLE. *Augmented Reality* [online]. [vid. 2024-04-11]. Dostupné z: https://developer.apple.com/augmented-reality/arkit/

[70] ORACLE. *MySQL* [online]. [vid. 2024-04-11]. Dostupné z: https://www.mysql.com

[71] POSTGRESQL GLOBAL DEVELOPMENT GROUP. *PostgreSQL: The World's Most Advanced Open Source Relational Database* [online]. [vid. 2024-04-11]. Dostupné z: https://www.postgresql.org

[72] MONGODB. *MongoDB* [online]. [vid. 2024-04-11]. Dostupné z: https://www.mongodb.com/lp/cloud/atlas/compare-mongodb-vs-postgresdb?utm_content=rlsavisitor&utm_source=google&utm_campaign=search_gs_pl_evergreen_atlas_competitor_retarget-nbnon_gic-null_emea-all_ps-all_desktop_eng_lead&utm_term=postgres&utm_medium=cpc_p

[73] WOLFSSL. *wolfMQTT Client Library* [online]. Dostupné z: https://www.wolfssl.com/products/wolfmqtt/?gad_source=1&gclid=Cj0KCQjwlN6wBhCcARIsAKZvD5jdI0GnPMTDmVnH0uZ2bmHiLiqUm1KiEGONmVRU9P4k4_4WikD5YA0aAgaxEALw_wcB

[74] AMAZON. *What is a RESTful API?* [online]. [vid. 2024-04-11]. Dostupné z: https://aws.amazon.com/what-is/restful-api/

[75] AMQP. *AMQP is the Internet Protocol for Business Messaging* [online]. [vid. 2024-04-11]. Dostupné z: https://www.amqp.org/about/what

[76] RADIOCRAFTS. *CoAP - Constrained Application Protocol* [online]. [vid. 2024-04-11]. Dostupné z: https://radiocrafts.com/technologies/coap-constrained-application-protocol/

[77] DDS FOUNDATION. *What is DDS?* [online]. [vid. 2024-04-11]. Dostupné z: https://www.dds-foundation.org/what-is-dds-3/

[78] IETF. *The WebSocket API (WebSockets)* [online]. [vid. 2024-04-11]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API

[79] UNITY. *Oficiální Unity3D návody* [online]. Dostupné z: https://learn.unity.com/

[80] UNITY3D. *IL2CPP Overview* [online]. Dostupné z: https://docs.unity3d.com/Manual/IL2CPP.html

[81] KINZER, Kelsey. *What Is ARM64 and Why Should You Use It?* [online]. Dostupné z: https://jumpcloud.com/blog/why-should-you-use-arm64

# Seznam příloh

# PŘÍLOHA č. 1

Skript *PopUpHandler*

```
using UnityEngine;

public class PopUpHandler: MonoBehaviour
{
    public GameObject objectToHide;
    public GameObject objectToShow;

    public void HideObject()
    {
        if (objectToHide != null)
        {
            objectToHide.SetActive(false);
            Invoke("ShowObject", 5f);
        }
    }

    private void ShowObject()
    {
        if (objectToShow != null)
        {
            objectToShow.SetActive(true);
            Invoke("HideObjectAfterDelay", 5f);
        }
    }
    private void HideObjectAfterDelay()
    {
        if (objectToShow != null)
        {
            objectToShow.SetActive(false);
        }
    }
```

# PŘÍLOHA č. 2

Skript *ColorChanger*

```
using System.Collections;
using System.Collections.Generic;
using TMPro;
using UnityEngine;
using UnityEngine.UI;

public class ColorChanger : MonoBehaviour
{
   public TMP_Text valueText;
   public Image imageComponent;
   public Color[] colorThresholds;
   public float[] thresholdValues;
   public string unit;
   public float minValue = 0f;
   public float maxValue = 100f;
   public GameObject toggleWarning;
   public GameObject toggleOk;
   public float warningThreshhold;

   void Update()
   {
      if (valueText != null && imageComponent != null)
      {
         string text = valueText.text;
         float value;
         string extractedUnit;
         if (TryExtractValueAndUnit(text, out value, out extractedUnit))
         {
            float normalizedValue = Mathf.Clamp01(Mathf.InverseLerp(minValue, maxValue,
value));
            Color newColor = DetermineColor(normalizedValue);
            imageComponent.color = newColor;
            imageComponent.fillAmount = normalizedValue;
            valueText.color = newColor;
```

```
            if (value > warningThreshhold)
            {
                toggleWarning.SetActive(true);
                toggleOk.SetActive(false);
            }
            else
            {
                toggleWarning.SetActive(false);
                toggleOk.SetActive(true);
            }
        }
    }
}

private bool TryExtractValueAndUnit(string text, out float value, out string unit)
{
    value = 0f;
    unit = "";
    string[] parts = text.Split(' ');
    if (parts.Length >= 2)
    {
        if (float.TryParse(parts[0], out value))
        {
            unit = parts[1];
            return true;
        }
    }
    return false;
}

private Color DetermineColor(float normalizedValue)
{
    if (colorThresholds.Length <= 1)
    {
        return Color.white;
```

```
        }


        for (int i = 0; i < colorThresholds.Length; i++)
        {
            if (i < colorThresholds.Length - 1)
            {
                if (normalizedValue <= thresholdValues[i + 1])
                {
                    float t = Mathf.InverseLerp(thresholdValues[i], thresholdValues[i + 1], normalizedValue);
                    return Color.Lerp(colorThresholds[i], colorThresholds[i + 1], t);
                }
            }
        }
        return colorThresholds[colorThresholds.Length - 1];
    }
}
```

# PŘÍLOHA č.3

Skript *ClickObject*

```
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.EventSystems;

public class ClickObject : MonoBehaviour
{
   public GameObject Object;
   public GameObject Panel;
   void Update()
   {
      if (Input.GetMouseButtonDown(0))
      {
         if (Object == GetClickedObject(out RaycastHit hit))
         {
            ToggleVisibility();
         }
      }
   }

   GameObject GetClickedObject(out RaycastHit hit)
   {
      GameObject target = null;
      var ray = Camera.main.ScreenPointToRay(Input.mousePosition);
      if (Physics.Raycast(ray.origin, ray.direction * 10, out hit))
      {
         if (!isPointerOverUIObject()) { target = hit.collider.gameObject; }
      }
      return target;
   }
   private bool isPointerOverUIObject()
   {
      PointerEventData ped = new PointerEventData(EventSystem.current);
      ped.position = new Vector2(Input.mousePosition.x, Input.mousePosition.y);
      List<RaycastResult> results = new List<RaycastResult>();
      EventSystem.current.RaycastAll(ped, results);
```

```
        return results.Count > 0;
    }
    void ToggleVisibility()
    {
        Panel.SetActive(!Panel.activeSelf);
    }
}
```

# PŘÍLOHA č.4

Skript *Gaze*

```
using System.Collections.Generic;
using System.Collections;
using UnityEngine;
using System.Linq;
public class Gaze : MonoBehaviour
{
List<InfoBehaviour> infos = new List<InfoBehaviour> ();
   void Start()
   {
      infos = FindObjectsOfType<InfoBehaviour>().ToList();


   }
   void Update()
   {
    if (Physics.Raycast(transform.position,transform.forward, out RaycastHit hit))
      {
         GameObject go = hit.collider.gameObject;
         if (go.CompareTag("hasInfo"))
         {
            OpenInfo(go.GetComponent<InfoBehaviour>());
         }
         else
         {
            CloseAll();
         }
      }
   }
   void OpenInfo(InfoBehaviour desiredInfo)
   {
      foreach (InfoBehaviour info in infos)
      {
         if (info == desiredInfo)
         {
            info.OpenInfo();
         }
```

```
            else
            {
                info.CloseInfo();
            }
        }
    }
    void CloseAll()
    {
        foreach (InfoBehaviour info in infos)
        {
            info.CloseInfo ();
        }
    }
}
```

# PŘÍLOHA č. 5

Skript *InfoBehaviour*

```
using UnityEngine;

public class InfoBehaviour : MonoBehaviour
{
    const float Speed = 6f;

    [SerializeField]

    Transform SectionInfo;

    Vector3 desiredScale = Vector3.zero;

    void Update()
    {
        SectionInfo.localScale = Vector3.Lerp(SectionInfo.localScale,desiredScale,
Time.deltaTime*Speed);
    }
    public void OpenInfo()
    {
        desiredScale = Vector3.one;
    }

    public void CloseInfo()
    {
        desiredScale = Vector3.zero;
    }
}
```

# PŘÍLOHA č. 6

Skript *FaceCamera*

```
using UnityEngine;

[ExecuteInEditMode]
public class FaceCamera : MonoBehaviour
{
    Transform cam;
    Vector3 targetAngle = Vector3.zero;

    private void Start()
    {
        cam = Camera.main.transform;
    }

    private void Update()
    {
        transform.LookAt(cam);
        targetAngle = transform.localEulerAngles;
        targetAngle.x = 0;
        targetAngle.z = 0;
        transform.localEulerAngles = targetAngle;
    }
}
```

# PŘÍLOHA č. 7

Skript *Device_ListManager*

```
using Newtonsoft.Json.Linq;
using System.Collections.Generic;
using TMPro;
using UnityEngine;


public class Device_ListManager : MonoBehaviour
{
    public TextMeshProUGUI tmpPrefab;
    public RectTransform tmpContainer;
    public string tag_mqttManager = "";
    public mqttManager _eventSender;
    public int textSpacing = 10;
    private List<TextMeshProUGUI> instantiatedTMPs = new List<TextMeshProUGUI>();
    void Start()
    {
        if (GameObject.FindGameObjectsWithTag(tag_mqttManager).Length > 0)
        {
            _eventSender = GameObject.FindGameObjectsWithTag(tag_mqttManager)[0].gameObject.GetComponent<mqttManager>();
        }
        else
        {
            Debug.LogError("At least one GameObject with mqttManager component and Tag == tag_mqttManager needs to be provided");
        }
        _eventSender.OnMessageArrived += OnMessageArrivedHandler;
    }
    private void OnMessageArrivedHandler(mqttObj mqttObject)
    {
        foreach (var tmp in instantiatedTMPs)
        {
            Destroy(tmp.gameObject);
        }
        instantiatedTMPs.Clear();
        JArray jsonArray = JArray.Parse(mqttObject.msg);
```

```
        float yOffset = 0f;
        float totalHeight = 0f;
        foreach (JObject deviceJson in jsonArray)
        {
            int deviceId = deviceJson.Value<int>("device_id");
            string deviceName = deviceJson.Value<string>("device_name");
            int deviceState = deviceJson.Value<int>("state_of_device");
            string deviceStateText = deviceState == 1 ? "Online" : "Offline";
            TextMeshProUGUI tmp = Instantiate(tmpPrefab, tmpContainer);
            tmp.gameObject.SetActive(true);
            tmp.text = $"ID: {deviceId}  {deviceName}  {deviceStateText}";
            Vector3 tmpPosition = tmp.rectTransform.localPosition;
            tmpPosition.y -= yOffset;
            tmp.rectTransform.localPosition = tmpPosition;
            yOffset += tmp.rectTransform.rect.height + textSpacing;
            totalHeight += tmp.rectTransform.rect.height + textSpacing;
            instantiatedTMPs.Add(tmp);
        }
        tmpContainer.sizeDelta = new Vector2(tmpContainer.sizeDelta.x, totalHeight);
    }
    public void PublishMessage()
    {
        _eventSender.Publish();
        Debug.Log("Message sent");
    }
    private void OnEnable()
    {
        InvokeRepeating("PublishMessage", 0f, 2f);
    }
    private void OnDisable()
    {
        CancelInvoke("PublishMessage");
}
```