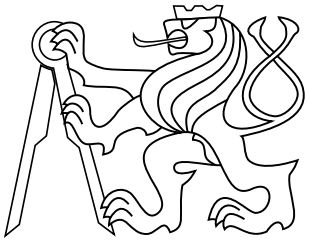
bakalářská práce

# Vizualizace stavu výroby v rozšířené realitě

*Daria Ozerova*



Květen 2020

Ing. Pavel Burget, Ph.D.

České vysoké učení technické v Praze
Fakulta elektrotechnická, Katedra řídicí techniky

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

## I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Ozerova** Jméno: **Daria** Osobní číslo: **466363**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra řídicí techniky**

Studijní program: **Kybernetika a robotika**

## II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

**Vizualizace stavu výroby v rozšířené realitě**

Název bakalářské práce anglicky:

**Production status visualisation in augmented reality**

Pokyny pro vypracování:

1. Seznamte se s platformou Unity a jejím použitím pro vývoj AR (augmented reality) aplikací pro Microsoft Hololens.
2. Navrhněte aplikaci pro vizualizaci stavu produktu vyráběného laboratorní výrobní linkou tak, aby byla aplikace snadno rozšiřitelná o nové modely a vyráběné produkty.
3. Aplikaci implementujte v Unity pro PC s Microsoft Windows či GNU/Linux pro vizualizaci v okně. Dále aplikaci implementujte pro Microsoft Hololens pro vizualizaci v AR.

Seznam doporučené literatury:

1. Unity User Manual. https://docs.unity3d.com.
2. Microsoft Docs: Microsoft Hololens. https://docs.microsoft.com/hololens.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

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

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **07.01.2020** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **30.09.2021**

| Ing. Pavel Burget, Ph.D. | prof. Ing. Michael Šebek, DrSc. | prof. Mgr. Petr Páta, Ph.D. |
|---|---|---|
| podpis vedoucí(ho) práce | podpis vedoucí(ho) ústavu/katedry | podpis děkana(ky) |

## III. PŘEVZETÍ ZADÁNÍ

Studentka bere na vědomí, že je povinna vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____ Podpis studentky

Datum převzetí zadání

## Poděkování

Děkuji Pavlu Burgetovi za nabídku práce na takto zajímavém projektu a za poskytnutí špičkového vybavení a zázemí. Dále děkuji Ondřeji Novákovi za cenné rady v průběhu mé práce. Jsem velmi vděčná paní Pinkové za rady tykající se pravopisu. Děkuji své rodině za morální a finanční podporu. Děkuji svým spolužákům Marišce a Kosťovi za to, že i ve zlé časy jsem nepřišla o rozum. Největší díky patří mému příteli Vítkovi, který mě při celém bakalářském studiu hodně podporoval.

## Prohlášení

Prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedla veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Datum

Podpis

# Abstrakt

Moderní průmysl umožňuje zákazníkovi si nejen objednat nějaký výrobek, ale i svoji objednávku různými způsoby nakonfigurovat. U některých výrobků se to může týkat barvy či materiálů, u jiných přidání dalších dílů. Výroba se pak může skládat z unikátních výrobků, které mají stejný typ, ale liší se konfiguracemi, které si určují zákazníci. Jelikož se ve velkých firmách dbá na prakticky nepřetržitý provoz, je velmi důležité umět efektivně kontrolovat, jestli jsou všechny unikátní výrobky v pořádku. Tato bakalářská práce se zabývá vytvořením aplikace, která sleduje, interpretuje a zobrazuje stav výrobků. V brýlích smíšené reality Microsoft Hololens a zároveň na stolním počítači se uživateli v reálném čase zobrazí modely zvolených výrobků. Aplikace byla navržena pro jednoduchou rozšiřitelnost o nové modely a byla otestována na dvou různých druzích výrobků.

## Klíčová slova

vizualizace robotických aplikací, rozšířená realita, Hololens, Unity

# Abstract

The modern industry provides customers with a possibility to not only order but also to configure a product. In some cases material or colour can be changed, other products could have additional parts. Manufacturing can consist of the same type of products with unique configuration chosen by a customer. Since large companies deploy resources on supporting the non-stop operation of machines, it is very important to be able to effectively control the state of all unique products. In this thesis, I implement an application that monitors, interprets, and visualise the state of manufacturing of products. Models of products are displayed to a user in real-time in smartglasses, Microsoft Hololens, and at the same time on a desktop computer. A great deal of emphasis has been placed on the easy extendability of the application. The application supports two types of products and it is easy to extend with new ones.



## Keywords

visualisation of robotic applications, augmented reality, Hololens, Unity

# Obsah

# Zkratky

**API** Application programming interface

**AR** Augmented reality

**HTTP** Hypertext Transfer Protocol

**IP** Internet Protocol

**MRTK** Mixed Reality Toolkit

**OOP** Object-oriented programming

**PDDL** Planning Domain Definition Language

**REST** Representational state transfer

**SDK** Software development kit

**SLAM** Simultaneous localization and mapping

**TCP** Transmission Control Protocol

**UDP** User Datagram Protocol

**URL** Uniform Resource Locator

**VR** Virtual reality

# 1 Úvod

Informační technologie mají obrovský vliv na změny ve všech oborech. V průmyslu díky tomu vnikl pojem tzv. *chytré továrny* [1]. Chytrá továrna funguje jako prostředí, ve kterém jsou stroje a výrobky digitálně propojeny. Prvky mezi sebou mohou komunikovat po síti, což otevírá lepší možnost pro automatizaci výroby, která pomáhá zvýšit produkci a navýšit počet nových výrobků a služeb. Princip chytré továrny snižuje počet chyb při výrobě a zlepšuje kvalitu výrobků, a díky tomu se zlepšuje spokojenost klientů [2].

Jedním z důležitých úkolů v průmyslu je monitoring stavu výroby nebo výrobní linky. Velké firmy dbají na to, aby jejich linky fungovaly nepřetržitě a mnoho úsilí je věnováno tomu, aby se jakýkoliv problém odhalil včas. Malá změna ve stavu nějakého výrobku může znamenat časově náročnou a drahou opravu nebo vyřazení z provozu [3]. Vizualizace stavu výrobků poskytuje další informaci pro monitoring a diagnostiku.

Cílem práce je vytvoření aplikace, která bude sloužit ke sledování stavu dokončení výrobku na průmyslové robotické lince. Práce popisuje implementaci aplikace pro PC a brýle Microsoft Hololens, která komunikuje s výrobní linkou a zobrazuje aktuálně vyrábějící se výrobky. Pracovníci továren mohou tímto způsobem pozorovat a kontrolovat výrobu i na dálku.

Aplikace by mohla umožnit klientovi odkudkoliv pozorovat každý krok výroby. Pokud se vyrábí několik výrobků najednou, uživatel si může zvolit, na který výrobek se v daném okamžiku chce dívat. I mimo továrnu si tedy může zkontrolovat, jestli je všechno v pořádku na modelu částečně sestaveného výrobku. Pokud to umožňuje výrobní linka, lze vidět i to, jak bude aktuálně vyráběný výrobek vypadat po dokončení. Model tak znázorňuje, které části výrobku jsou již postavené, a které zatím nikoli.

Augmented reality (AR) umožňuje uživateli prohlédnout si výrobek v různých velikostech a z různých stran. Mimo jiné klient tento výrobek může vidět i v reálném prostředí. To je relevantní zejména u velkých výrobků - například automobilů.

## 1.1 Cíle práce

Aplikace by měla disponovat několika základními funkcemi:

1. Komunikovat s centrálním serverem
2. Zobrazovat stav dokončení výrobku v reálném čase
3. Být jednoduše rozšířitelná o nové výrobky
4. Umět stahovat grafické modely součástek za běhu programu

5. Zobrazovat budoucí (cílový) stav výrobku (pokud je to možné)

Každý z těchto úkolů a některé další podrobněji popíšu v následujících kapitolách.

## 1.2 Struktura práce

Práce je rozdělena do šesti kapitol. V Kapitole 2 krátce vysvětlím, co je AR, uvedu hlavní rozdíly mezi rozšířenou a virtuální realitou, stručně popíšu princip a typy AR technologií. Později v kapitole představím aplikace AR v průmyslu a jejich využití. V Kapitole 3 podrobně vysvětlím úkoly aplikace zobrazující stav výrobků a konkrétní techniky pro její implementaci. Dále uvedu nástroje, ve kterých je aplikace implementována a jejich hlavní výhody. Implementace je popsána v Kapitole 4, její podkapitoly budou věnovány konkrétním řešením zadaných úkolů. V Kapitole 5 popíšu testování aplikace. Nakonec v Kapitole číslo 6 shrnu svoji práci a uvedu nápady na zlepšení vytvořené aplikace.

# 2 Rozšířená a virtuální realita v průmyslu

Následující kapitolu věnuji krátkému úvodu. V sekci 2.1 krátce vysvětlím rozdíl mezi AR a Virtual reality (VR), stručně popíšu základní druhy AR a technologie použité v aplikaci. V podkapitole 2.2 uvedu několik příkladů využití AR v průmyslu.

## 2.1 Technologie AR

AR projektuje do skutečného světa objekty namodelované na počítači, tj. doplňuje svět další grafickou, zvukovou apod. informací. VR je počítačem vytvořený svět, ke kterému můžeme získat přístup pomocí speciálních zařízení - brýlí, sluchátek nebo rukavic. Hlavní rozdíl technologií VR a AR je, že první interaguje s uživatelem, druhá s reálným světem.

Mezi nejdůležitější komponenty AR patří senzory a kamera. Zmíněné zařízení umožňují použití různých technologií, mezi které patří například Simultaneous localization and mapping (SLAM), který řeší problém mapování okolí a následnou orientaci v něm. Dalším příkladem je depth tracking, používaný pro detekci hloubky a vzdálenosti objektů. V [4] a [5] jsou některé technologie podrobně popsány a jsou uvedeny principy moderní AR. V této podkapitole jen krátce popíši její čtyři základní typy.

1. **Marker-based AR** používá kameru (telefonu, tabletu nebo brýlí) a počítačové vidění pro detekci libovolného vizuálního objektu (fotka, QR code, obrázek atd.), kterému se říká *marker* [4]. Počítačové vidění umožňuje zpracování obrazů a videí (například z kamery, fotek atd.). Díky tomu je možné v zařízení analyzovat obraz z kamery, detekovat pozici a orientaci markeru [6]. Marker se následně doplňuje o další obrázek, který má stejnou pozici a orientaci.

2. **Markerless AR** používá GPS, akcelerometr, gyroskop nebo kompas pro zjištění pozice a orientace zařízení. Pomocí dat ze senzorů, zařízení vytváří mapu určité lokality a orientuje se v ní. Pokud zařízení obsahuje kameru, používá se i počítačové vidění umožňující rozpoznání určitých objektů, které pomáhají zařízení orientovat se v okolí. Také to umožňuje informování různými doplněnými obrázky, které uživatel vidí přes kameru spolu se skutečným světem.

3. **Projection-based AR** je jeden z nejjednodušších typů AR. Technologie vytváří projekci na nějaký povrch. Projection-based AR může zjišťovat pozici, orientaci a hloubku objektu, na který projektuje. Vzdálenost nebo hloubka objektu se detekuje pomocí měření času odrazu světla.

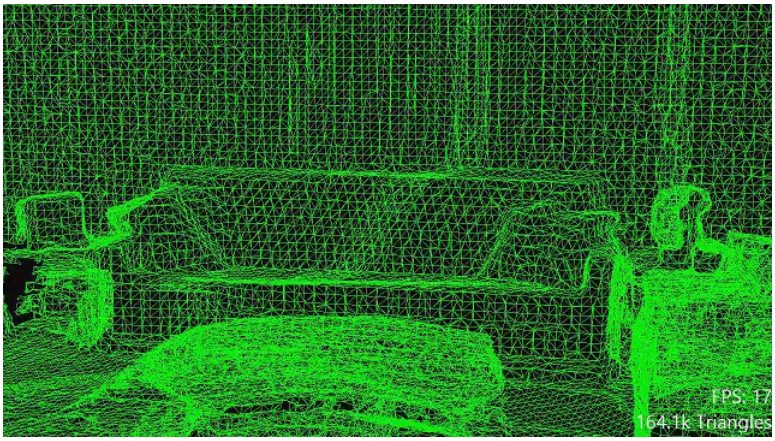
4. **Superimposition-based AR** využívá kameru a počítačové vidění pro detekci určitých objektů, které kompletně zaměňuje novými objekty podobného typu. Takovým objektem je například lebka, kterou pomocí Superimposition-based AR v [7] zaměňují za lidský obličej.

Některá zařízení umí podporovat několik typů AR. V aplikacích Microsoft Hololens se nejčastěji využívá markerless AR. Brýle “se orientují” v místnosti díky technologii

*spatial mapping*. Pomocí hloubkové a obyčejné kamery a speciálního software se vytváří 3D mapa místnosti, ve které se nachází uživatel. Spatial mapping zlepšuje vnímání reálného a virtuálního světa [8]. Uživatel si v reálném čase umísťuje virtuální objekt do místnosti, ve které se nachází. Kombinací vstupů ze senzorů s technologií SLAM je možné udržení objektů tam, kam je uživatel umístil.

Microsoft Hololens disponují několika obyčejnými kamerami, díky kterým lze použít Marker-based AR. Implementace Marker-based AR aplikací v Hololens je možné v herním engine Unity s použitím například *Vuforia Augmented Reality* Software development kit (SDK).

V této bakalářské práci využívám markerless AR s technologií spatial mapping. Za použití těchto technologií sledované výrobky mohou být v harmonii s místnosti. Výrobu uživatel uvidí na fyzickém povrchu (například na povrchu stolu). Díky tomu je dojem z výrobku mnohem realističtější.




**Obrázek 2.1** Příklad vizualizace spatial mapping místnosti v Microsoft Hololens [1]

## 2.2 Využití AR v průmyslu

Technologie AR a VR se využívají v medicíně, vzdělávání, reklamě a dalších oborech. V této podkapitole se zaměříme na jejich aplikace v průmyslu.

AR a VR jsou důležitou součástí chytré továrny anebo Průmyslu 4.0 [9, 10]. Technologie se mohou využívat ve všech stadiích výrobního cyklu, od vyřízení objednávky a projektování až po prodej. Výrobek se zákazníkovi ukáže ještě než se začne vyrábět. Díky tomu mohou firmy nabízet ukázku své produkce na výstavách, v kancelářích a v místech s vysokým počtem lidí.

Někteří výzkumnici předpokládají, že virtuální konfigurace a prototypování budoucích výrobků a vizualizace technologických a výrobních procesů v sobě skrývá největší ekonomický přínos [11]. Z toho vyplývá, že technologie virtuální a rozšířené reality mohou v průmyslu mít velmi užitečné aplikace. Díky tomu se každým rokem zvětšuje počet firem, které zmíněné technologie integrují do svých výrobních plánů [12].

[1] Zdroj: https://docs.microsoft.com/en-us/windows/mixed-reality/spatial-mapping

Ve výzkumu [13] se popisuje mobilní aplikace využívající AR pro trénování údržby vzdušného a kosmického prostoru. Cílem práce je vzdělávání a navádění nových technických pracovníků, kteří se zabývají údržbou a opravou letadel. Mimo jiné, může aplikace sloužit ke zpětné vazbě práce odborníků v reálném světě, a díky tomu eliminovat chyby při provozu. Aplikace funguje tak, že si začátečník nasadí brýle rozšířené reality, přes ně vidí součástky letadla s textovými poznámky o jejich skladbě a návod na údržbu. Díky použití aplikací takového typu je vzdělávání odborníků mnohem levnější a efektivnější [14].

V článku [15] je popsána aplikace, která zajišťuje podporu vzdálené průmyslové údržby. Techničtí pracovnici v továrně u sebe mají tablet s podporou AR, který po síti posílá obraz z kamery zkušenému odborníkovi, který do něj může po přijetí kreslit užitečné informace. Doplněný obrázek pak vidí na tabletu pracovník v továrně, a díky tomu je celková údržba levnější a je potřeba méně zkušených odborníků.

Jedním z nejčastějších využití virtuální reality v průmyslu je simulace a prototypování budoucích výrobku [10]. Příkladem aplikace může být *sketching*. Sketchning je představen většinou na papíře, ale občas je vhodné vidět určitý objekt i ve 3D (například model budovy nebo design bytu), kde použití technologií je nezbytné [10].

V laboratoři Testbed pro Průmysl 4.0 v CIIRC ČVUT v Praze byla vytvořena AR aplikace pro Microsoft Hololens. Aplikace umožňuje virtuální propojení dvou výrobních linek s roboty mezi sebou. Při použití na první robotické lince uživatel vidí modely robotů z druhé robotické linky, která se ve skutečnosti nachází mimo laboratoř. Pohyby modelů jsou shodné s pohyby skutečných robotů nacházejících se na druhé robotické lince.

# 3 Návrh aplikace

V této kapitole popíšu princip fungování aplikace zobrazující stav výrobní linky. V sekci 3.1 uvedu strukturu aplikace, stručně popíšu její nejdůležitější prvky. V podkapitolách 3.2 a 3.3 vysvětlím princip fungování demonstrační výrobní linky a vyjmenuji výrobky, které se na ní vyrábí. V sekci 3.4 krátce popíšu zvolené řešení pro komunikaci aplikace se serverem. V podkapitolách 3.5 a 3.6 vysvětlím, jak řeším stahování modelů a jejích konfigurací v reálném čase. V podkapitole 3.7 popíšu způsob zpracování zprávy přijaté od serveru. Nakonec v sekci 3.8 uvedu nástroje, které budu používat pro implementaci aplikace a její hlavní výhody.

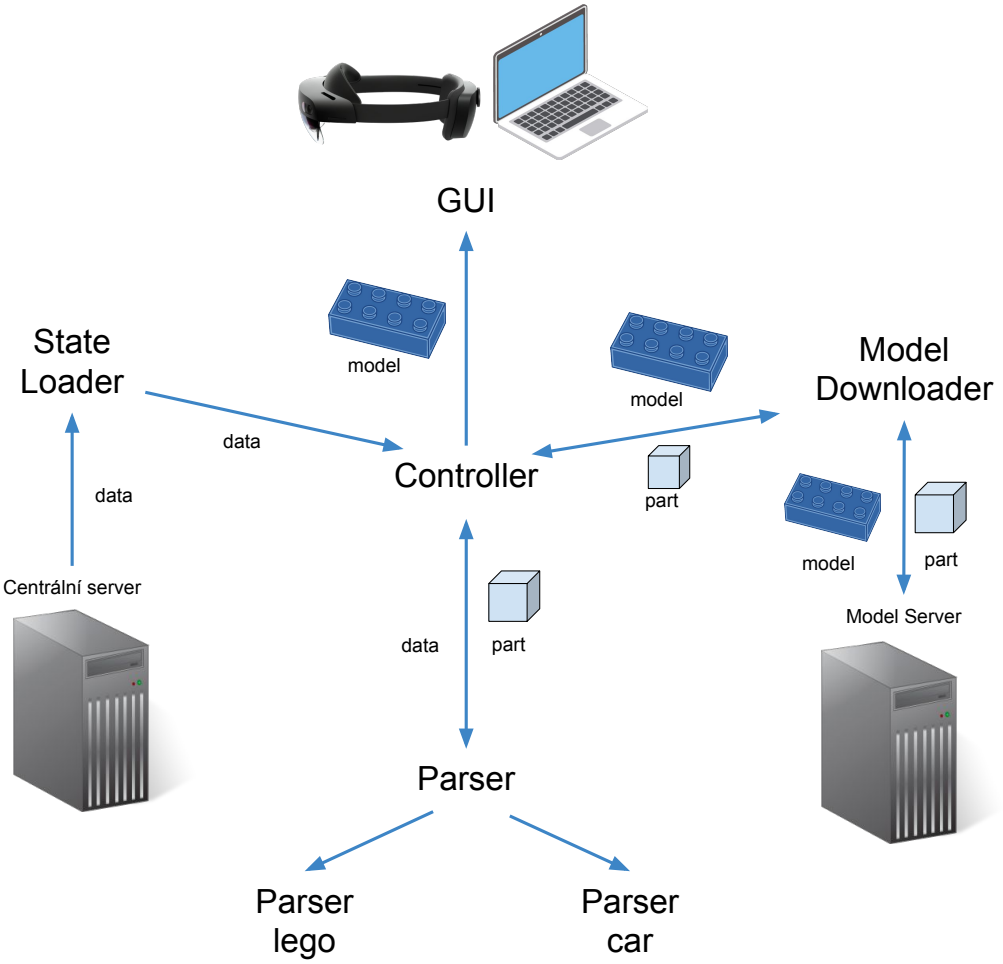
## 3.1 Struktura aplikace

Aplikace je rozdělena na několik části, které budu dále v textu nazývat následovně:

- `Model Server` je program, který uchovává a přeposílá modely součástek a jejich konfigurací (rozměry, orientaci, typ).
- `Model Downloader` stahuje modely a konfigurace určitých součástek ze serveru.
- `Centrální server` poskytuje stavy výrobní linky. Komunikuje s aplikací, ale není její součásti. Toto je styčný bod mezi mou aplikací a existujícím systémem.
- `State Loader` komunikuje s centrálním serverem a dostává informaci o současném stavu výrobní linky.
- `Parser` zpracovává informaci dodanou centrálním serverem.
- `GUI` je uzel odpovídající za grafickou vizualizaci modelů.
- `Controller` je hlavní uzel. Svazuje všechny uzly mezi sebou.

`Centrální server` vysílá zprávu obsahující informaci o stavu výrobní linky. Po přijetí dat se zpráva nejdříve zpracuje `Controllerem`. `Controller` určí, jaký typ výrobku se na výrobní lince vyrábí, a na základě toho vytvoří příslušný syntaktický analyzátor (`Parser`) pro analýzu konkrétního typu výrobku. `Parser` určí, co se na lince vyrábí, a najde veškerou informaci o současně vyrábějících se výrobcích (název, součástky, barvu, pozice, orientace atd.). `State Loader` dále podle názvu stahuje modely součástek těchto výrobků. Stažené modely se pak v určité konfiguraci zobrazují uživateli.

Části aplikace a jejich propojení jsou schématicky znázorněny na Obrázku 3.1. Funkcionalitou a konkrétní implementací každé části se budu zabývat v pozdějších kapitolách.




**Obrázek 3.1** Schéma aplikace [1]

## 3.2 Popis výrobní linky

Testování a realizace aplikace probíhají v laboratoři Testbed pro Průmysl 4.0, která se nachází v budově CIIRC ČVUT v Praze. Testbed je zaměřen na průmyslové aplikace. Mimo jiné je laboratoř vybavena výrobní linkou s roboty, u kterých jsou umístěny tzv. *stanice*. Mezi stanicemi jezdí několik vozíků dopravujících součástky pro výrobu. Na Obrázku 3.2 se nachází fotografie demonstrační výrobní linky v laboratoři.

Jedním z úkolů výrobní linky je tvorba výrobků, které se skládají z několika předem připravených součástek. Úkolem robotů je přenášení součástek výrobků ze stanic na správná místa na vozíku. Cíl je flexibilní a může jít například o postavení výrobku na určitém vozíku.

Jakmile vozík zastaví před robotem, robot postaví jeden nebo více dílů na souřadnice specifikované řídícím systémem. Poté se vozík přesouvá k dalšímu robotovi, kde se

[1] Zdroj obrázku brýlí Microsoft Hololens: https://docs.microsoft.com/cs-cz/hololens/

postaví další součástka a tak pokračuje dále, dokud výrobek nedosáhne cílového stavu. Po cílovém stavu se vozík vyprázdní a cyklus se opakuje.


**Obrázek 3.2** Výrobní linka v laboratoři Testbed pro Průmysl 4.0 v CIIRC ČVUT [2]

O tom, jak přesně se pohybuje robot, a kam a odkud přesouvá součástky, rozhoduje řídící program, který je schopný vykonávat dynamicky vytvořený plán podle stanoveného cíle. Pro vytvoření plánu je také potřebný současný stav linky, který je uložený v tzv. *digitálním dvojčeti linky*.

Díky tomu, že server ukládá současný stav výrobní linky, je možné vytvořit aplikaci, kterou se zabývá tato bakalářská práce.

## 3.3 Druhy výrobků

V současné době je na robotické lince možné vyrábět dva druhy výrobků. Prvním je libovolná struktura postavená z LEGO kostiček, druhým je model auta. V případě auta vozík postupně přijede k až třem robotům, podle aktuálního stavu zásob dílů u jednotlivých robotů a každý z nich přidá na vozík jeden nebo více dílů (podvozek, kabina, kufr).

Informace o stavu výrobní linky je reprezentován jako text v Planning Domain Definition Language (PDDL) syntaxi, nebo jako soubor s koncovkou .problem. Tuto informaci budu dále referovat jako *problem* a označovat ji kurzivou.

Výrobky mají odlišné popisy digitálního dvojčete. Část souboru popisující stav výrobní linky při výrobě auta a struktury z LEGO kostiček je znázorněna na Obrázcích

[2] Zdroj: https://www.ciirc.cvut.cz

3.3 a 3.4.

Lze konstatovat, že popis stavu výrobní linky je různý pro různé modely, a proto je třeba každý z nich analyzovat trochu jinak. V Kapitole 4 popisuji význam jednotlivých částí digitálního dvojčete a způsoby jejich analýzy.

```
(FINAL_PRODUCT TARGET X4 Y2 Z1 BLUE)
(FINAL_PRODUCT TARGET X4 Y5 Z1 BLUE)
(FINAL_PRODUCT TARGET X4 Y6 Z1 BLUE)
(FINAL_PRODUCT TARGET X4 Y9 Z1 BLUE)
(GRIPPER_HOLDS BRICK-4X2 O-1 BLUE)
(OCCUPIED SOURCE X-1 Y-11 Z0)
(OCCUPIED SOURCE X-1 Y-12 Z0)
(OCCUPIED SOURCE X-1 Y-15 Z0)
(OCCUPIED SOURCE X-1 Y-16 Z0)
(OCCUPIED SOURCE X-1 Y-3 Z0)
(OCCUPIED SOURCE X-1 Y-4 Z0)
(OCCUPIED SOURCE X-1 Y-7 Z0)
(OCCUPIED SOURCE X-1 Y-8 Z0)
(OCCUPIED SOURCE X-1 Y0 Z0)
(OCCUPIED SOURCE X-1 Y0 Z1)
(OCCUPIED SOURCE X-1 Y1 Z0)
(OCCUPIED SOURCE X-1 Y1 Z1)
(OCCUPIED SOURCE X-10 Y-11 Z0)
(OCCUPIED SOURCE X-10 Y-11 Z1)
(OCCUPIED SOURCE X-10 Y-12 Z0)
(OCCUPIED SOURCE X-10 Y-12 Z1)
(OCCUPIED SOURCE X-10 Y-15 Z0)
(OCCUPIED SOURCE X-10 Y-15 Z1)
(OCCUPIED SOURCE X-10 Y-16 Z0)
(OCCUPIED SOURCE X-10 Y-16 Z1)
(OCCUPIED SOURCE X-10 Y-3 Z0)
(OCCUPIED SOURCE X-10 Y-3 Z1)
```

**Obrázek 3.3** Část popisu stavu výrobní linky při výrobě struktury z LEGO kostiček

```
    PART-TYPE-T-TRANSPORT
    PART-BLACK-CHASSIS
    X0_Y5_Z5_R0
)
(RESOURCE_CONTAINS_PART_AT
    SHUTTLE5
    PART-TYPE-T-TRANSPORT
    PART-T-TRANSPORT
    UNSPECIFIED
)
(RESOURCE_CONTAINS_PART_AT
    SHUTTLE5
    PART-TYPE-T-TRANSPORT
    PART-WHITE-CABIN
    X0_Y21_Z7_R90
)
(RESOURCE_CONTAINS_PART_AT
    SHUTTLE5
    PART-TYPE-T-TRANSPORT
    PART-WHITE-STAKEBED
    X0_Y5_Z7_R0
)
(RESOURCE_IN_STATION R1 S110)
(RESOURCE_IN_STATION R1 S12)
(RESOURCE_IN_STATION R1 SVR1)
(RESOURCE_IN_STATION R10 S110)
(RESOURCE_IN_STATION R10 SVR10)
```

**Obrázek 3.4** Část popisu stavu výrobní linky při výrobě modelu auta

I přestože je význam informace v jednotlivých souborech odlišný, *problem* vždy obsahuje následující informace:

- názvy stanic a vozíků,
- seznam součástek,
- souřadnice součástek,
- parametry výrobků,
- a některé další.

## 3.4 Komunikace aplikace s centrálním serverem

Důležitou funkcí aplikace je komunikace se serverem. Jelikož aplikace může být zapnutá na několika zařízeních zároveň, je vhodné použít model Klient-Server. Aplikace je tedy

klientem komunikujícím s centrálním serverem. Všechna zařízení, fungující jako zobrazovač se mohou dotazovat serveru na současný stav.

Pro implementaci komunikaci bylo zvoleno použití komunikace Transmission Control Protocol (TCP). Na rozdíl například od User Datagram Protocol (UDP) se mezi strany nejdříve nastaví oboustranné spojení, které existuje, dokud ho strany nezavřou [16]. Aplikace není časově kriticky náročná. Chceme-li zobrazit veškerou možnou informaci, není UDP pro aplikaci vhodné. TCP zajišťuje garanci správného přenosu.

## 3.5 Stahování modelů ze serveru

Aplikace běží na PC a v brýlích smíšené reality Microsoft Hololens. Aplikace má uživatelské rozhraní, ve kterém si uživatel vybere vozík, na který se chce podívat. Pro lepší znázornění výrobku v brýlích je zobrazen i hologram zvoleného vozíku nebo stanice.

Dalším problémem aplikace je distribuce modelů. Výrobků a součástek může být hodně a mohou se časem měnit. Není proto vhodné jejich grafické modely ukládat přímo na zařízení. Jedním z řešení problému je vytvoření v aplikaci dalšího klienta, který bude stahovat modely z jiného zařízení. Díky použití serveru není třeba po změně nebo přidání modelů znovu kompilovat program na každém zařízení, které obsahuje aplikaci zobrazující stav výrobků.

Je vhodné pro klienta a server použít Hypertext Transfer Protocol (HTTP) z důvodu volně dostupných programovacích balíků.

Veškerá komunikace se serverem se redukuje na čtyři základní operace s daty: získaní (GET), přidání (POST), modifikace (PUT) a mazání (DELETE). Representational state transfer (REST) Application programming interface (API) jsou obecné principy organizace komunikace klienta se serverem pomocí HTTP protokolu. Klient posílá žádosti na server, který mu poskytuje potřebná data.

V aplikaci zobrazující stav výrobků stačí použít žádost GET, která žádá server o určitá data (v našem případě o modely součástek). Klient pomocí GET pošle název určité součástky a server ve své odpovědi pošle její model (soubor formátu .obj).

## 3.6 Konfigurace modelů

Ne všechny grafické modely mají stejné rozměry, rotaci nebo střed. Může se stát, že je model součástky orientován jinak, než má být ve finálním zobrazení, nebo má jiné než požadované rozměry. Je proto nutné všechny modely pozměnit tak, aby se s nimi dobře pracovalo.

Konfiguraci (rozměry, rotaci a střed) musíme určit na straně serveru a posílat klientovi spolu s modelem. Data s konfiguracemi modelů uložíme do souboru ve formátu JSON. Použití JSON je velmi přehledné jak pro uživatele tak pro počítač [3]. Uživatel, který do aplikace přidává nové součástky, zadává jejich konfiguraci a cestu k souboru s modelem přímo do souboru formátu JSON. Je to velmi praktické, protože se nemusí

[3] Více informace o JSON: https://www.json.org/json-en.html

nic měnit v samotné aplikaci. Distribuce je poté snazší.

```
{
  "name": "BRICK-6X2",
  "model": "files/models/lego/6x2.obj",
  "type": "lego",
  "obj_config": [
    { "scale" : "0.06329", "rot" : "-90", "centerBias" : "-1.25"},
    { "scale" : "0.06329", "rot" : "0", "centerBias" : "0.00"},
    { "scale" : "0.06329", "rot" : "0", "centerBias" : "-0.25"}
  ]
},
{
  "name": "CHASSIS",
  "model": "files/models/car/Base.obj",
  "type": "car",
  "obj_config": [
    { "scale" : "0.06329", "rot" : "0", "centerBias" : "-0.25"},
    { "scale" : "0.06329", "rot" : "180", "centerBias" : "1.57"},
    { "scale" : "0.06329", "rot" : "0", "centerBias" : "-2.111"}
  ]
},
```

**Obrázek 3.5** Část souboru formátu JSON obsahující informaci o modelech součástek

Na Obrázku 3.5 je znázorněna část souboru obsahující informaci o modelech. V JSON je uvedená cesta k souboru s modelem, typ součástky a její název. Konfigurace modelů se nachází v poli pod klíčem `obj_config`. V poli jsou vždy tři řádky jako tři komponenty vektoru. Klíče `scale`, `rot` a `centerBias`, pak uvádí rozměr, rotaci a střed objektu.

## 3.7 Vizualizace výrobků

Centrální server posílá zprávu s popisem digitálního dvojčete. Po přijetí klientem se zpráva zpracovává. Nejdříve se podle klíčových slov hledají názvy vozíků a stanic. Dále se v textu hledají výrobky, které tyto vozíky či stanice obsahují. Nalezené výrobky se zobrazují ve výsledné aplikaci.

Pokud v souboru existuje informace o tom, jak vypadá finální výrobek, můžeme informaci využít a na zvoleném vozíku zobrazit v scéně obrysy hotového výrobku, které budou v průběhu výroby postupně doplňovány. V aplikaci se této funkcionalitě říká *ghost*. Ghosta je možné zobrazit, jako průhledný model hotového výrobku. Pokud cílový stav není známý, průhledného ghosta nemáme, a jakmile se reálná součástka postaví na vozík, zobrazíme ji hned barevně.

## 3.8 Nástroje pro vývoj v AR

Pro vizualizaci stavu výroby se v této bakalářské práce používá AR, a proto nejdříve stručně uvedu i vývojová prostředí, ve kterém se dá s AR pracovat. Současně aplikace pro Microsoft Hololens je možné vyvíjet s níže uvedenými herními enginy.

- Unity
- Unreal Engine (zatím jen beta verze)

Jedním z nejpoužívanějších herních enginů je Unity. Unity se používá pro vývoj 2D nebo 3D her, rozšířenou a virtuální realitu, vývoj počítačových simulací apod. Unity, na rozdíl od mnoha jiných herních enginů, je multiplatformní software. Jelikož cílem této bakalářské práce je vývoj aplikace pro minimálně dvě zařízení, použití Unity pro implementaci je velmi výhodné.

Unity samotné nepodporuje vývoj aplikací používajících rozšířenou realitu. Existuje několik nástrojů, které se dají do Unity nainstalovat a používat ho jako nástroj pro AR. Brýle smíšené reality Microsoft Hololens pro vývoj aplikací používají Mixed Reality Toolkit (MRTK) do Unity. V dokumentaci [17] je vysvětleno, jak se instaluje software použitý pro vývoj AR aplikace.

# 4 Implementace

Následující Kapitolu věnuji podrobnému vysvětlení principu implementace aplikace. Nejdříve v podkapitole 4.1 zmíním, jak aplikace komunikuje s centrálním serverem. V sekci 4.2 vysvětlím důležitou funkcionalitou aplikace - syntaktickou analýzu. Dále v podkapitole 4.3 popíšu souřadnice popsané v digitálním dvojčeti a jejich použití při vizualizaci. V podkapitole 4.4 popíšu implementaci klienta a serveru využívající HTTP protokol. Nakonec v sekci 4.5 uvedu základní uživatelské rozhraní a jeho použití.

## 4.1 State Loader

Aplikace je rozdělena do dvou vláken. V jednom vlákně probíhá syntaktická analýza a vizualizace výrobku, ve druhém běží TCP klient, který čeká na zprávy od centrálního serveru.

Klient je vytvořen pomocí třídy `TCPClient` standardní knihovny jazyku C#. Příklad použití `TCPClient` v dokumentaci .NET framework [18] byl použit jako základ klienta v této aplikaci. Příklad je rozšířen o několik funkcí. Aby aplikace nebyla komunikací příliš vytížena, je v klientu implementován časovač, který kontroluje zprávy jen po určité době. Klient z příkladu je také rozšířen o funkci, která kontroluje, jestli je přijatá zpráva odlišná od předchozí. Funkce je velmi výhodná v případech, kdy se několik cyklů časovače nic neděje. Aplikace tak není přetížena zbytečným zpracováváním redundantních zpráv.

Po přijetí zprávu zpracovává `Controller`, který určuje typ výrobku a následně vytváří syntaktický analyzátor příslušného typu pro další analýzu dat.

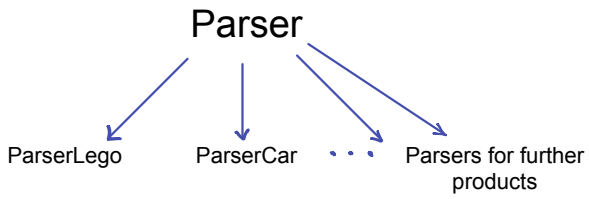
## 4.2 Syntaktická analýza stavu výrobní linky

`Parser` je jednou z nejdůležitějších částí programu. Jeho úkolem je syntaktická analýza *problemu* obsahující stav výrobní linky. `Parser` se můžeme ptát na současně obsazené souřadnice, existenci cílového stavu vozíku nebo například na seznam stanic v určitém *problemu*.

Jak jsem již zmiňovala v Kapitole 3, pro různé výrobky je výroba a její popis odlišný. Je proto vhodné navrhovat `Parser` tak, aby ho šlo jednoduše rozšířit o další modely a výrobky.

Jedním z řešení problému rozšíření aplikace je návrh `Parseru` jako tzv. *Interface*. Interface je konstrukční prvek v Object-oriented programming (OOP), který definuje rozhraní objektů bez jakékoli implementace. Jinými slovy interface určuje, jaké funkce mají mít třídy, které ho implementují.

Jelikož mám k dispozici dva druhy *problemů*, interface `Parser` implementují dvě třídy: `ParserLego` a `ParserCar`.




**Obrázek 4.1** Třídy, které implementují `Parser`

Aby aplikace byla jednoduše rozšiřitelná i o další výrobky, navrhla jsem funkce, které by měly mít všechny současné a budoucí třídy implementující interface `Parser`:

- `GetAllTargets(message)` vypíše názvy všech vozíků a stanic,
- `CountBuiltParts(message, target)` spočítá, kolik je postaveno součástek na daném vozíku (stanici),
- `IsThereGhost(message)` určí, jestli z popisu stavu výrobní linky lze zjistit cílový stav výrobků,
- `FindBuiltParts(message, target)` najde všechny nově postavené součástky,
- `ProcessParts(message, target)` zpracuje cílový stav výrobku (pokud ho lze zjistit).

Funkce `IsThereGhost(message)` má stejnou implementaci pro `ParserCar` a pro `ParserLego`. V souboru *problem* najdeme část, která se jmenuje `goal`. Pokud se v `goal` obsahuje klíčové slovo `None`, víme, že cílový stav výrobní linky není známý.

```
(:goal
    (and
        (OCCUPIED TARGET X3 Y-16 Z1)
        (OCCUPIED TARGET X-2 Y13 Z1)
        (OCCUPIED TARGET X1 Y13 Z1)
        (OCCUPIED TARGET X-2 Y9 Z1)
        (OCCUPIED TARGET X1 Y1 Z1)
        (OCCUPIED TARGET X-5 Y15 Z1)
        (OCCUPIED TARGET X1 Y-4 Z1)
        (OCCUPIED TARGET X1 Y5 Z1)
        (OCCUPIED TARGET X-4 Y-2 Z1)
        (OCCUPIED TARGET X4 Y-16 Z1)
        (OCCUPIED TARGET X-1 Y13 Z1)
        (OCCUPIED TARGET X-3 Y-9 Z1)
        (OCCUPIED TARGET X3 Y-3 Z1)
        (OCCUPIED TARGET X1 Y10 Z1)
        (OCCUPIED TARGET X-5 Y4 Z1)
        (OCCUPIED TARGET X-3 Y-13 Z1)
        (OCCUPIED TARGET X1 Y14 Z1)
```

**Obrázek 4.2** Příklad digitálního dvojčete obsahující cílový stav výrobků

```
    (RESOURCE_IN_STATION R2_TABLE SVR2)
    (RESOURCE_IN_STATION R3 S23)
    (RESOURCE_IN_STATION R3 SVR3)
    (RESOURCE_IN_STATION R3_TABLE SVR3)
    (RESOURCE_IN_STATION SHUTTLE2 S12)
    (RESOURCE_IN_STATION SHUTTLE3 S100)
    (RESOURCE_IN_STATION SHUTTLE4 S110)
    (RESOURCE_IN_STATION SHUTTLE5 S23)
    (RESOURCE_LOCKED R10_TABLE)
    (RESOURCE_LOCKED R1_TABLE)
    (RESOURCE_LOCKED R2_TABLE)
    (RESOURCE_LOCKED R3_TABLE)
    (RESOURCE_LOCKED SHUTTLE2)
    (RESOURCE_LOCKED SHUTTLE3)
    (RESOURCE_LOCKED SHUTTLE4)
    (RESOURCE_LOCKED SHUTTLE5
)
(:goal (and None))
(:metric MINIMIZE (TOTAL-COST))
```

**Obrázek 4.3** Příklad digitálního dvojčete s neznámým cílovým stavem výrobků

Funkcionalitu `ProcessParts(message, target)` vysvětlím na příkladě digitálního dvojčete linky při výrobě struktury z LEGO kostiček. Pro model auta se implementace liší jen v některých klíčových slovech. Funkce se volá jenom v případě, kdy je cílový

stav známý. Ve funkci zpracováváme část `goal`, lepší formulace které je znázorněna na Obrázku 4.2.

Každý řádek začíná slovem `OCCUPIED`, které nám říká, že na některých souřadnicích je postavena součástka, druhé slovo je název vozíku nebo stanice (v daném případě `TARGET`), `X`, `Y`, `Z` jsou obsažené souřadnice. Dané souřadnice nejsou světové a při vizualizaci je třeba je transformovat. Samotnou transformaci se zabývám později v sekci 4.3.

V části `goal` nám chybí důležitá informace. Nevíme totiž, jaké součástky se staví. Tyto informace najdeme v jiné části *problemu*, která je představena na Obrázku 4.4.

```
(FINAL_BRICK_SHAPE SOURCE X9 Y0 Z1 BRICK-2X2 R0)
(FINAL_BRICK_SHAPE TARGET X-3 Y-13 Z1 BRICK-4X2 R0)
(FINAL_BRICK_SHAPE TARGET X-3 Y-17 Z1 BRICK-4X2 R0)
(FINAL_BRICK_SHAPE TARGET X-3 Y-5 Z1 BRICK-4X2 R0)
(FINAL_BRICK_SHAPE TARGET X-3 Y-9 Z1 BRICK-4X2 R0)
(FINAL_BRICK_SHAPE TARGET X-3 Y1 Z1 BRICK-4X2 R0)
(FINAL_BRICK_SHAPE TARGET X-3 Y13 Z1 BRICK-4X2 R0)
(FINAL_BRICK_SHAPE TARGET X-3 Y17 Z1 BRICK-4X2 R0)
(FINAL_BRICK_SHAPE TARGET X-3 Y5 Z1 BRICK-4X2 R0)
(FINAL_BRICK_SHAPE TARGET X-3 Y9 Z1 BRICK-4X2 R0)
(FINAL_BRICK_SHAPE TARGET X-5 Y-17 Z1 BRICK-4X2 R90)
```

**Obrázek 4.4** Část *problemu* obsahující informaci o typech součástek

Klíčové slovo `FINAL_BRICK_SHAPE` uvádí tvar, pozici, orientaci součástky a na jakém vozíku se tato součástka staví.

Poslední informací o součástkách, kterou můžeme v *problemu* vyhledat, je její barva. Část *problemu* popisující barvu součástky, která se nachází na daných souřadnicích, je znázorněna na Obrázku 4.5.

```
(FINAL_PRODUCT SOURCE X9 Y1 Z1 BLUE)
(FINAL_PRODUCT TARGET X-1 Y-12 Z1 BLUE)
(FINAL_PRODUCT TARGET X-1 Y-13 Z1 BLUE)
(FINAL_PRODUCT TARGET X-1 Y-16 Z1 BLUE)
(FINAL_PRODUCT TARGET X-1 Y-17 Z1 BLUE)
(FINAL_PRODUCT TARGET X-1 Y-4 Z1 BLUE)
(FINAL_PRODUCT TARGET X-1 Y-5 Z1 BLUE)
(FINAL_PRODUCT TARGET X-1 Y-8 Z1 BLUE)
(FINAL_PRODUCT TARGET X-1 Y-9 Z1 BLUE)
(FINAL_PRODUCT TARGET X-1 Y1 Z1 BLUE)
(FINAL_PRODUCT TARGET X-1 Y10 Z1 BLUE)
```

**Obrázek 4.5** Část *problemu* obsahující barvy součástek

Zpracování postavených součástek probíhá v `FindBuiltParts(message, target)`. Funkce se volá pokaždé, když přijde nová zpráva s popisem stavu výrobní linky. `Parser` prohledává zprávu a podle klíčových slov (konkrétně `OCCUPIED` pro LEGO kostičky a `RESOURCE_CONTAINS_PART_AT` pro model auta) hledá nové součástky a potřebné informace, které se v aplikaci zatím nebyly zpracovány.

```
(RESOURCE_CONTAINS_PART_AT
    R2_TABLE
    PART-TYPE-T-STORAGE
    PART-T-STORAGE
    UNSPECIFIED
)
(RESOURCE_CONTAINS_PART_AT
    SHUTTLE2
    PART-TYPE-T-TRANSPORT
    PART-BLACK-CHASSIS
    X0_Y5_Z5_R0
)
(RESOURCE_CONTAINS_PART_AT
    SHUTTLE2
    PART-TYPE-T-TRANSPORT
    PART-BLUE-CABIN
    X0_Y21_Z7_R90
)
(RESOURCE_CONTAINS_PART_AT
    SHUTTLE2
    PART-TYPE-T-TRANSPORT
    PART-BLUE-OPENTOP
    X0_Y5_Z7_R0
)
```

**Obrázek 4.6** Popis některých postavených částí modelu auta

```
(OCCUPIED TARGET X-1 Y-12 Z1)
(OCCUPIED TARGET X-1 Y-13 Z0)
(OCCUPIED TARGET X-1 Y-13 Z1)
(OCCUPIED TARGET X-1 Y-16 Z0)
(OCCUPIED TARGET X-1 Y-16 Z1)
(OCCUPIED TARGET X-1 Y-17 Z0)
(OCCUPIED TARGET X-1 Y-17 Z1)
(OCCUPIED TARGET X-1 Y-4 Z0)
(OCCUPIED TARGET X-1 Y-4 Z1)
(OCCUPIED TARGET X-1 Y-5 Z0)
(OCCUPIED TARGET X-1 Y-5 Z1)
(OCCUPIED TARGET X-1 Y-8 Z0)
(OCCUPIED TARGET X-1 Y-8 Z1)
(OCCUPIED TARGET X-1 Y-9 Z0)
(OCCUPIED TARGET X-1 Y-9 Z1)
(OCCUPIED TARGET X-1 Y1 Z0)
(OCCUPIED TARGET X-1 Y1 Z1)
(OCCUPIED TARGET X-1 Y10 Z0)
(OCCUPIED TARGET X-1 Y10 Z1)
(OCCUPIED TARGET X-1 Y13 Z0)
(OCCUPIED TARGET X-1 Y13 Z1)
(OCCUPIED TARGET X-1 Y14 Z0)
(OCCUPIED TARGET X-1 Y14 Z1)
(OCCUPIED TARGET X-1 Y17 Z0)
```

**Obrázek 4.7** Informace o souřadnicích postavených LEGO kostiček

Každou součástku reprezentuje objekt třídy `Part` obsahující údaje o jejich rozměrech, poloze, orientaci a vzhledu. Jakmile se ve skutečném světě postaví jedna nebo více součástek, v `Parseru` se zpracuje informace o stavu výrobní linky a vytvoří se objekty reprezentující součástky jednotlivých výrobků.

Funkce `GetAllTargets(message)` podle klíčových slov vyhledá názvy všech vozíků a stanic. V případě LEGO můžeme vozíky a stanice najít v části s klíčovým slovem `FINAL_BRICK_SHAPE`, která byla popsána výše. V případě auta jsou všechny stanice a vozíky vypsány na začátku *problemu*.

## 4.3 Transformace souřadnic

Jak bylo výše zmíněno, souřadnice uvedené v *problemu* nejsou světové. V aplikaci jim říkáme *LEGO-souřadnice*. LEGO má vlastní souřadnice, protože při tvorbě LEGO výrobků se používá vozík obsahující tzv. *piny*, na které se umisťují kostičky. Souřadnice popsané ve stavu výrobní linky určují, na kterém z pinů je postavena LEGO kostička.

Vozík, na kterém se staví struktura z LEGO kostiček, má velikost 25 x 37 LEGO pinů. Na ose $x$ je 25 pinů, na ose $y$ je 37. Pozice každé součástky se počítá od středu vozíku. Například všechny součástky popsané na Obrázku 4.8 mají $x$ a $y$ souřadnice záporné, a tedy jsou umístěny vlevo dole na ploše.

```
(OCCUPIED TARGET X-3 Y-12 Z1)
(OCCUPIED TARGET X-3 Y-13 Z0)
(OCCUPIED TARGET X-3 Y-13 Z1)
(OCCUPIED TARGET X-3 Y-16 Z0)
(OCCUPIED TARGET X-3 Y-16 Z1)
(OCCUPIED TARGET X-3 Y-17 Z0)
(OCCUPIED TARGET X-3 Y-17 Z1)
(OCCUPIED TARGET X-3 Y-4 Z0)
(OCCUPIED TARGET X-3 Y-4 Z1)
(OCCUPIED TARGET X-3 Y-5 Z0)
(OCCUPIED TARGET X-3 Y-5 Z1)
(OCCUPIED TARGET X-3 Y-8 Z0)
```

**Obrázek 4.8** Příklad popisu souřadnic některých LEGO kostiček

Model vozíku či stanice vytvořeny v Unity má určité globální souřadnice, které se při běhu aplikace mohou měnit. Aby se nemusely zvlášť přepočítat globální souřadnice všech součástek umístěných na stanovišti, je vhodné pro jejich popis používat lokální souřadnice, tj. v daném případě souřadný systém vozíku nebo stanice. LEGO souřadnice pak reprezentují pozici každé součástky vhledem k pozici stanoviště.

Při přidání nového modelu, ho nejdříve umístíme na vozík, což v Unity realizujeme pomocí *parentingu* (vozík je *parent* všech na něm umístěných součástek). Lokální souřadnice součástek následně změníme na LEGO souřadnice uvedené v *problemu* vynásobené velikosti jednoho pinu. Velikost jednoho pinu je reprezentována vektorem $[0.5\ 0.6\ 0.5]^T$, což jsou rozměry pinu v souřadnicích $xyz$.

## 4.4 Model Server a Model Downloader

LEGO kostička velikosti $n \times m$ pinů má určený střed v levém dolním pinu. V daném případě známe rozměry každé kostičky, a proto se střed kostičky v aplikaci jednoduše spočítá. Středy jiných výrobků (například auta) obecně nelze vypočíst. Je proto potřeba informaci zadávat ručně do souboru `data.json`, který obsahuje konfiguraci součástek a cestu k soubory s jejími modely.

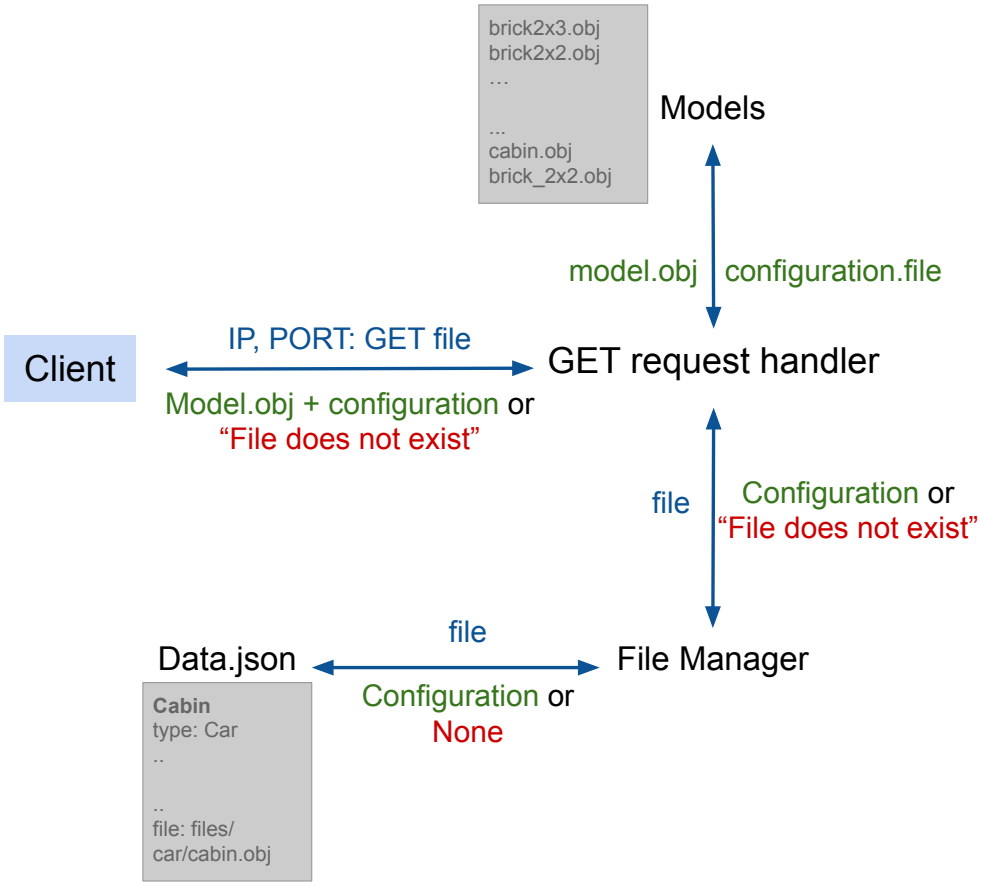
HTTP Server jsem vytvořila v jazyce JavaScript s použitím knihovny `Node.js` [19]. Server ukládá modely součástek (soubory typu .obj) a jejich konfiguraci.

Klient posílá žádosti na Uniform Resource Locator (URL) s Internet Protocol (IP) počítače a portem, na kterém běží aplikace. Žádost GET obsahuje název souboru, který klient chce stáhnout. Server zkontroluje, jestli soubor s daným názvem v systému existuje. Pokud ano, pošle ho spolu s jeho konfigurací a se zprávou, že soubor je nalezen. V opačném případě pošle oznámení, že soubor s daným názvem neexistuje.

Přijatou konfiguraci modelu (rozměry, rotaci a střed) `Controller` spojuje s údaji získanými z *problemu* (pozicí, rotací, barvou atd.) a má tak informaci pro vytvoření modelů stavějících se součástek na ploše v Unity.

Jelikož za běhu programu existuje možnost měnit vozík nebo stanici, může se stát, že na dvou zvolených stanovištích se staví výrobek stejného typu (například je na obou model auta). Není proto optimální jejich součástky pokaždé stahovat ze serveru. Načítání modelů může být pomalé, protože modely se stahují přes síť. Pokud ale již stažené

modely uložíme do dočasného cache, můžeme je využít při analýze další zprávy, a proto při zvolení dalšího vozíku či stanice se čas načítání modelů může mnohokrát zmenšit.




**Obrázek 4.9** Schéma funkcionality `Model Serveru`

## 4.5 Uživatelské rozhraní

Aplikace pro Microsoft Hololens je vytvořena pomocí MRTK nástroje v herním engine Unity. Uživatelské rozhraní obsahuje několik možnosti pro uživatele. Za prvé, je možné zvolit IP adresu a port pro oba servery, které aplikace obsahuje. Aby uživatel při každém spuštění aplikace nemusel znovu zadávat stejné údaje, IP adresy a porty se po každé změně ukládají do textového souboru. Po opětovném spuštění aplikace se načtou uložená nastavení.

Vstupní hodnoty musíme zkontrolovat. Je potřeba, aby obsahovaly platnou IP adresu a port. Toho můžeme docílit pomocí tzv. *regulárního výrazu*. Regulární výraz je řetězec definující vzor, který popisuje celou množinu řetězců. Řetězec s IPv4 adresou vždy obsahuje tři tečky. Počet číslic mezi tečkami se může lišit, a proto je použití regulárního výrazu v tomto případě výhodné.

Programovací jazyk C# má na regulární výraz speciální knihovnu `Regex` obsahující funkci `Match(Regex r, string s)`, která vrací *true*, pokud řetězec $s$ odpovídá vzoru $r$.

Při kontrole IP adresy použijeme regulární výraz

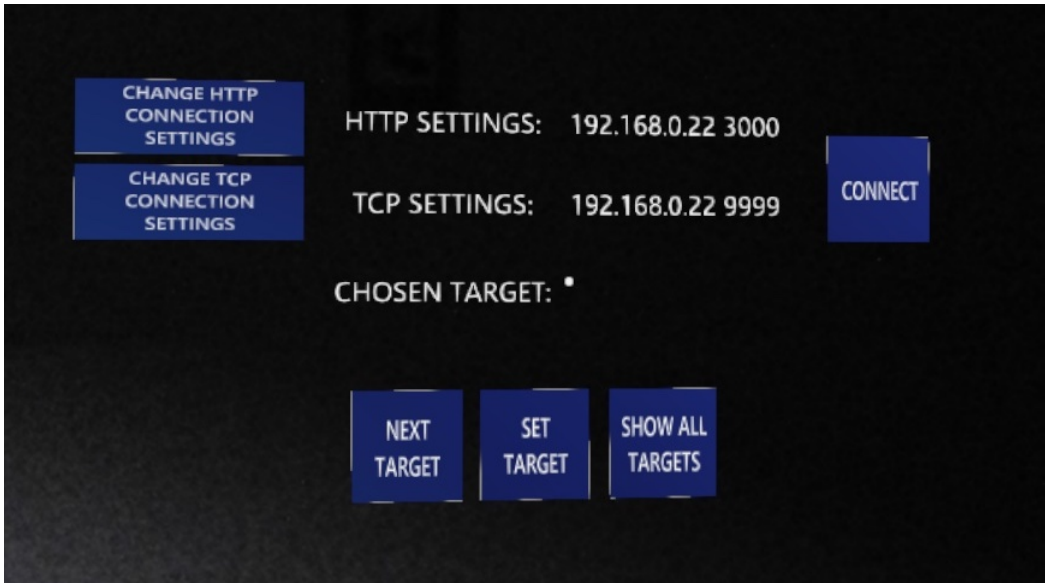
$$\hat{}([0-9]\{1,3\})\backslash.([0-9]\{1,3\})\backslash.([0-9]\{1,3\})\backslash.([0-9]\{1,3\}),$$

který zkontroluje, jestli řetězec obsahuje tři tečky, před a za kterými jsou jedna až tři číslice. Pro kontrolu portu použijeme jednoduchý výraz

$$\hat{}[0-9]\{1,4\},$$

který zjišťuje, jestli port obsahuje jednu až čtyři platné číslice.

Po zkontrolování základních nastavení, uživatel potvrdí tlačítko `Connect` pro připojení k centrálnímu serveru a k serveru, odkud se budou stahovat modely. Pokud servery fungují a jsou zadané správné IP adresy a porty, klient dostane popis digitálního dvojčete. Před uživatelem se objeví první vozík či stanice a pomocí tlačítka `Next` se může listovat mezi dalšími stanovišti popsanými v digitálním dvojčeti. Na Obrázku 4.10 je znázorněno uživatelské rozhraní aplikace pro v Microsoft Hololens.




**Obrázek 4.10** Menu v aplikaci pro Microsoft Hololens

Uživatel si může zvolit buď jeden vozík či stanici pomocí tlačítka `Next` nebo vybrat `Show all targets` a pozorovat všechny vozíky a stanice najednou.

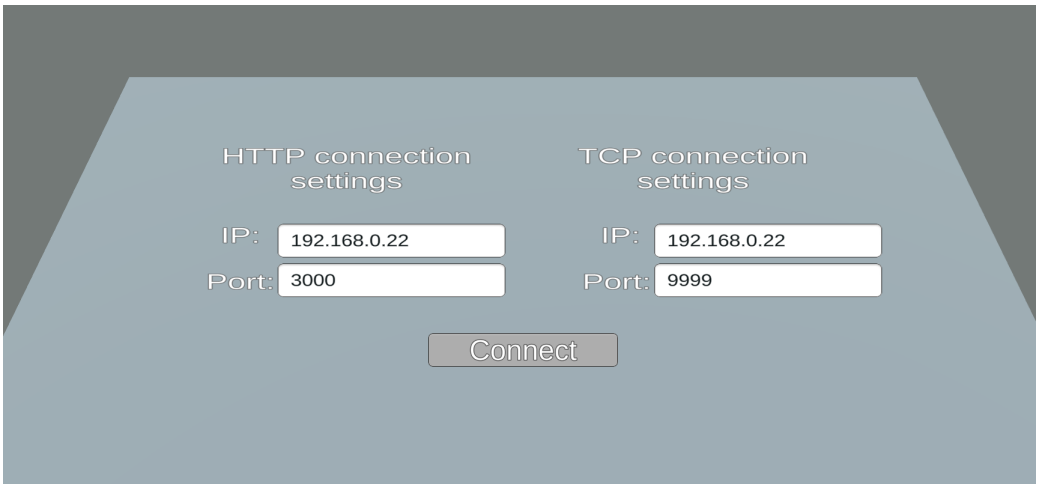
Vozíky a stanice jsou reprezentovány plochou v Unity. Pomocí základních funkcí MRTK jsem přidala možnost změny pozice, orientace a velikosti plochy. Pokud si uživatel později zvolí jiný vozík, orientace, pozice a velikost budou stejné jako u předchozího. V případě, kdy je zvoleno více vozíků, je možná změna konfigurace každého zvlášť.

# 5 Vizualizace stavu výroby

V následující Kapitole vysvětlím použití aplikace, popíšu a znázorním její funkcionalitu a možnosti.

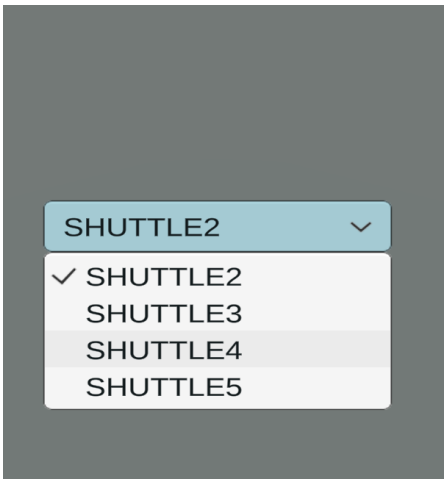
## 5.1 Testování aplikace

Po zapnutí aplikace před sebou uživatel vidí menu disponující možnostmi nastavení komunikace s centrálním serverem a se serverem, odkud se budou stahovat modely. Uživatel může změnit IP adresu a port. Po změně vybere tlačítko `Connect`. Na Obrázcích 5.1, 5.2 a 5.3 je znázorněno menu aplikace na PC.




**Obrázek 5.1** Část menu aplikace na PC s nastavením IP adresy a portu pro komunikaci


**Obrázek 5.2** Část menu aplikace na PC s nastavením modelu vozíku

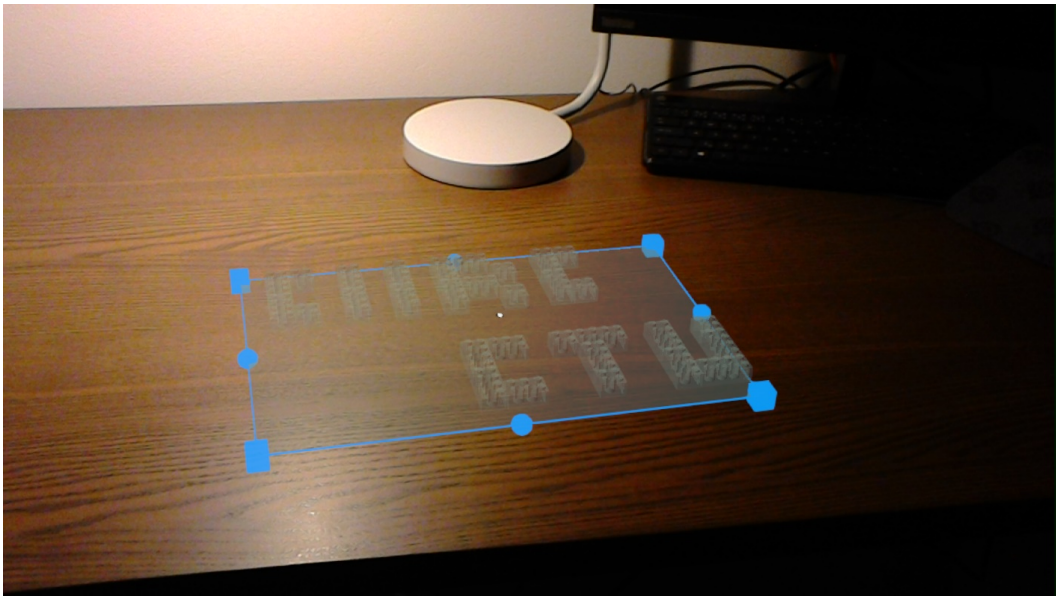



**Obrázek 5.3** Výpis všech vozíků popsaných v digitálním dvojčeti

Po připojení na server se před uživatelem objeví pracovní plocha jednoho z vozíků, na kterém se současně něco staví.

**Obrázek 5.4** Vozík s výrobkem umístěný na zdi v aplikaci pro Microsoft Hololens

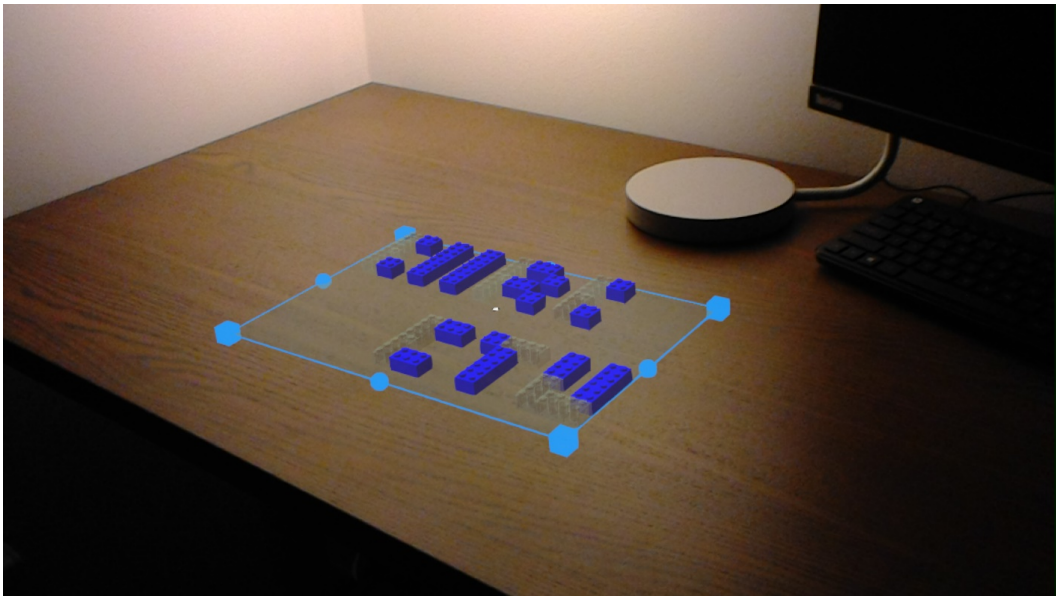
Na Obrázcích 5.4, 5.5 a 5.6 je znázorněna funkcionalita aplikace v brýlích Microsoft Hololens na příkladě výroby nápisu "CIIRC CTU" z LEGO kostiček na pracovní ploše vozíku.



**Obrázek 5.5** Model vozíku představující budoucí stav výrobku

Jak jsem zmínila v Kapitole 2 brýle smíšené reality Microsoft Hololens používají technologii spatial mapping. V aplikaci se tato technologie využívá pro propojení výrobků s místnosti, ve které se nachází. Na Obrázku 5.4 je představen model vozíku s výrobkem umístěný na zdi, na Obrázcích 5.5 a 5.6 je výrobek umístěný na povrch stolu. Umístění

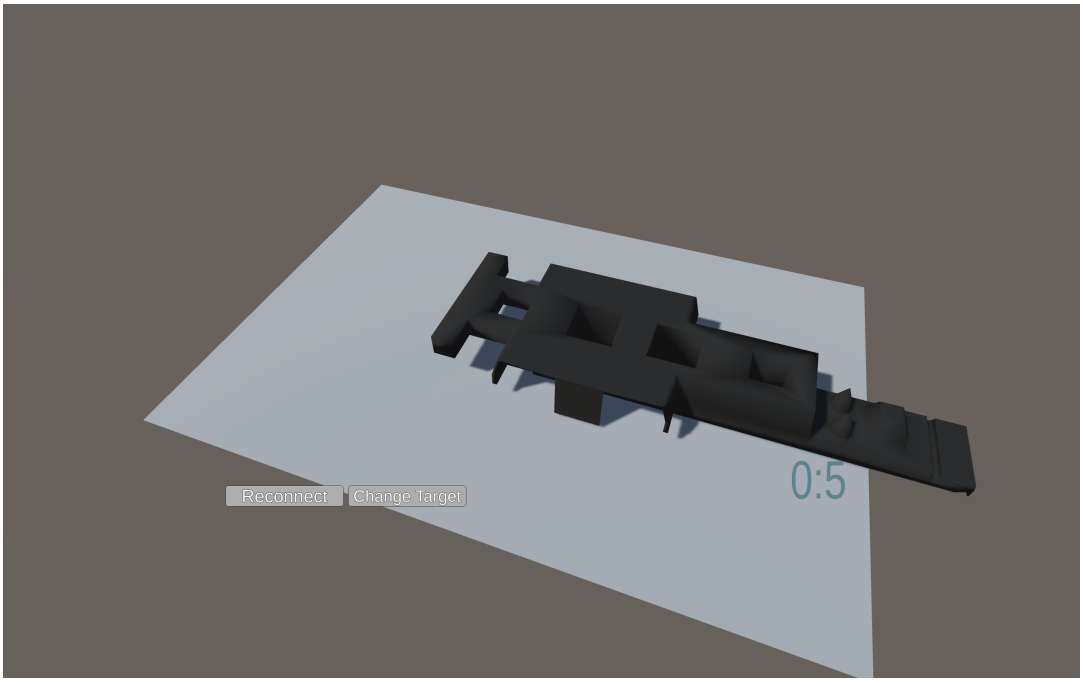
výrobků do místnosti je výhodné zejména pro velké objekty. Například představte si, že vidíte, jak se staví vaše nové auto ve vaši garáži.



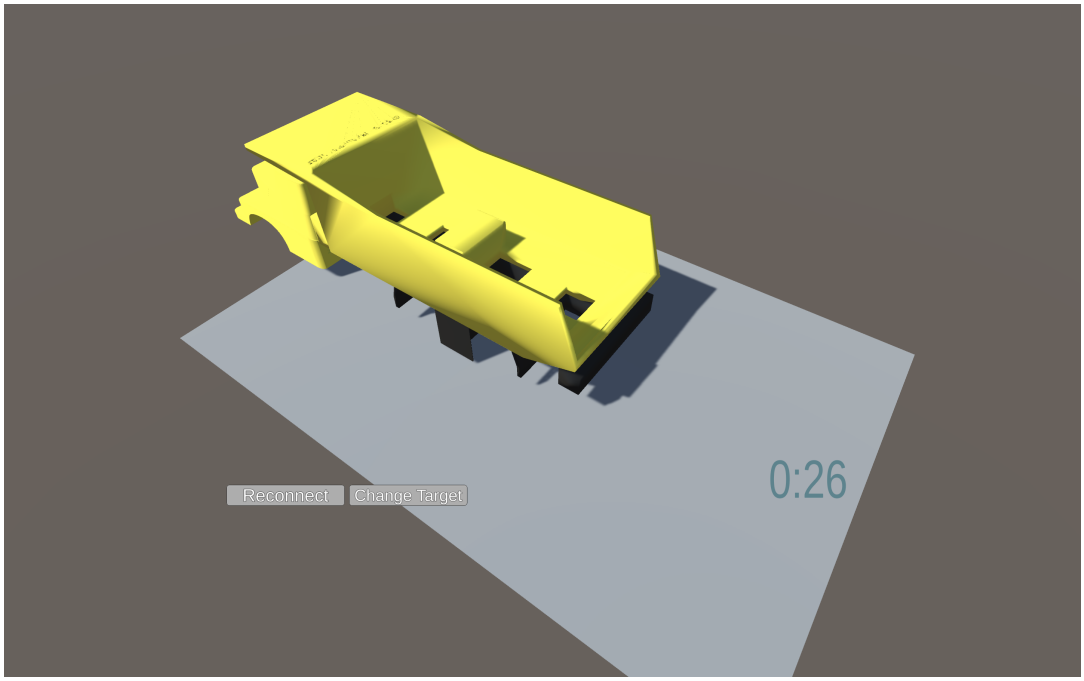
**Obrázek 5.6** Model vozíku obsahující výrobek v pokročilém stádiu výroby

Další možností nastavení pro uživatele je volba velikosti, orientace a polohy plochy, kterou před sebou vidí. Výrobky přitom kopírují transformaci plochy.

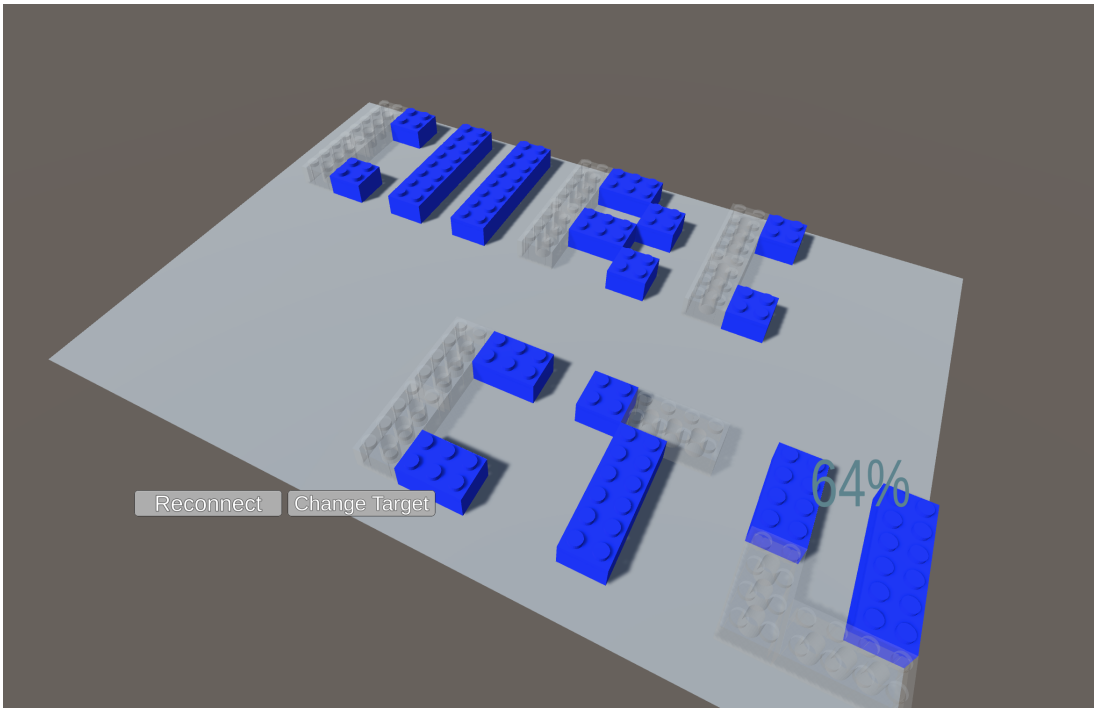
V aplikaci na PC takové funkce nejsou. Místo toho se plocha s výrobkem otáčí.




**Obrázek 5.7** Model auta s postaveným podvozkem v aplikaci na PC




**Obrázek 5.8** Hotový model auta v aplikaci na PC[1]

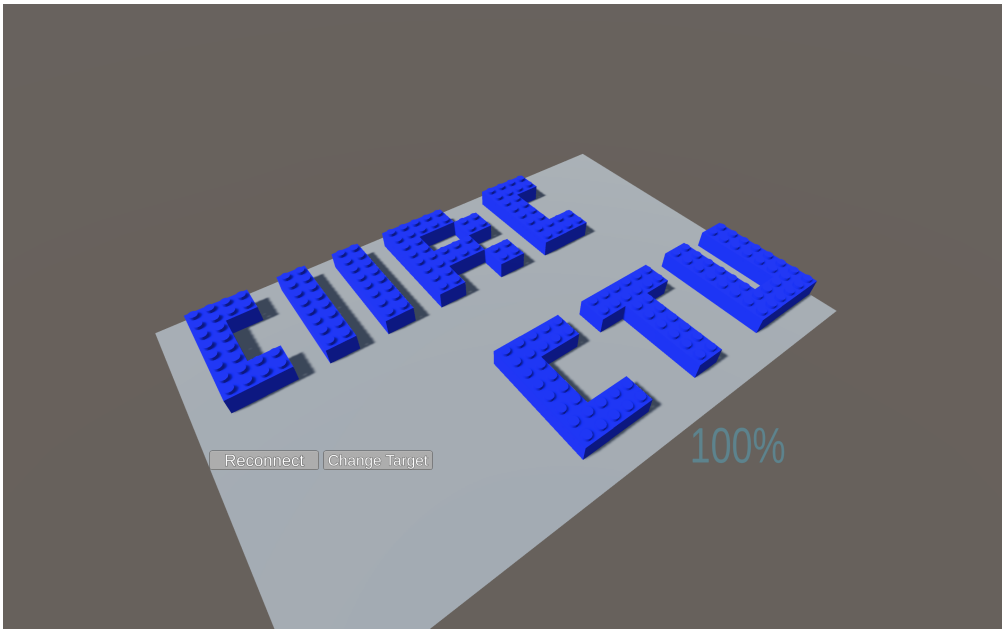



**Obrázek 5.9** Nápis "CIIRC CTU" z LEGO kostiček vyrobený na 64%

Po přijetí první zprávy se zkontroluje, jestli z ní lze zjistit cílový stav. Pokud ano, vytváří se ghost, jak je vidět na Obrázku 5.5. Průhledné součástky znamenají budoucí

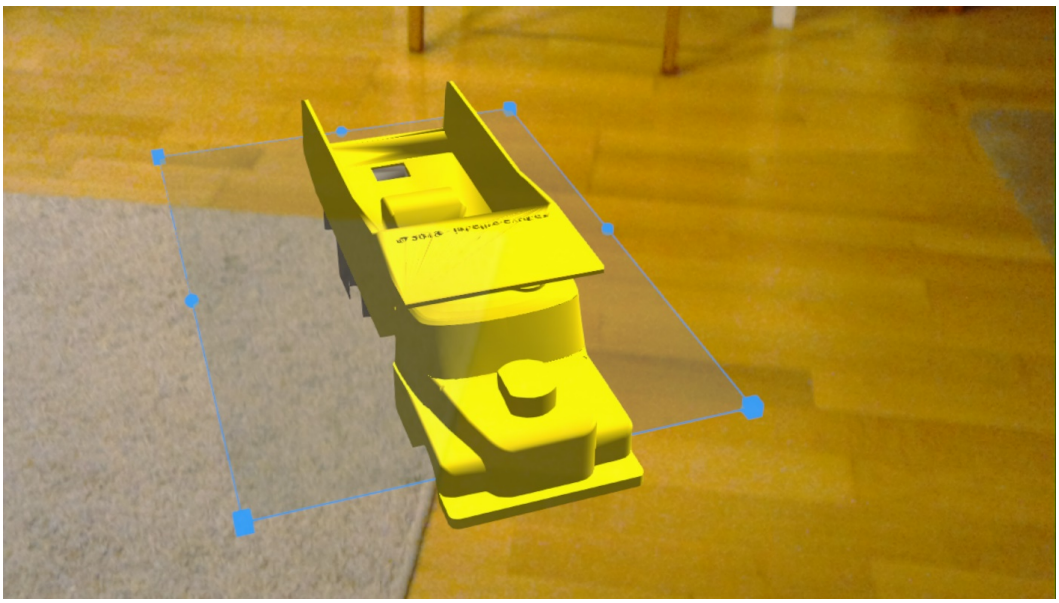
[1]Poznámka: Je vidět, že auto je lehce vystrčeno z vozíku. Ve skutečnosti souřadnice, na kterých se nachází některé součástky jsou opravdu mimo plochu, což se znázorňuje v aplikaci. Není to tedy chyba.

stav. Postupně po přijetí dalších zpráv se tyto součástky obarvují. Obrázek 5.6 znázorňuje výrobek v již pokročilejším stadiu výroby.




**Obrázek 5.10** Nápis "CIIRC CTU" z LEGO kostiček vyrobený na 100%

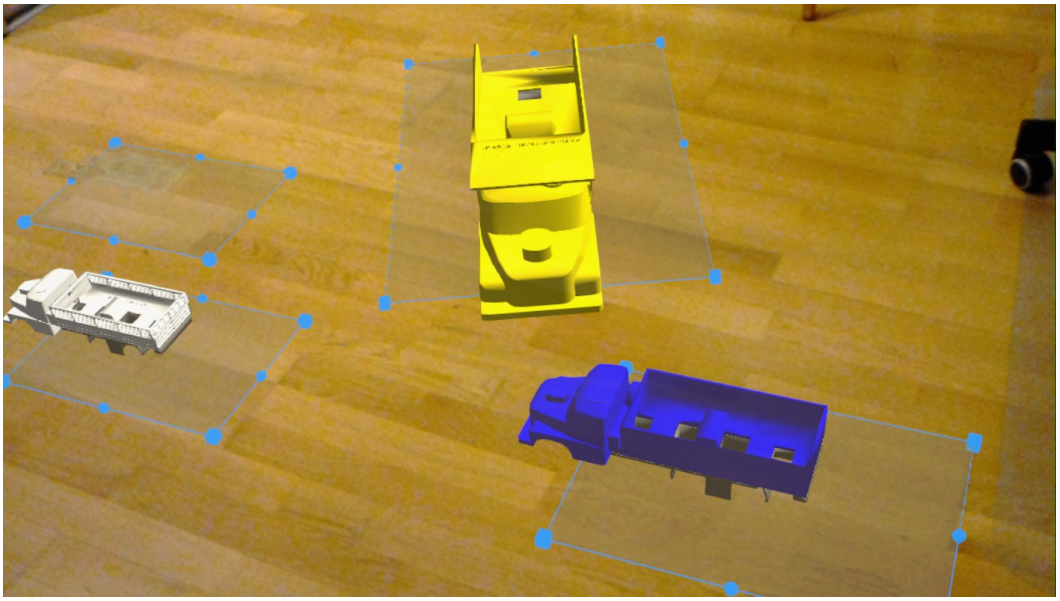
Pokud cílový stav zjistit nelze (jak je například při výrobě modelu auta), tak je plocha na začátku prázdná a již barevné součástky se postupně přidávají, což je vidět na Obrázcích 5.7 a 5.8, kde je nejdříve postaven jen podvozek auta, pak po nějaké době se postaví kabina a vlečka.



**Obrázek 5.11** Ukázka jednoho výrobku v aplikaci pro Microsoft Hololens

Pro znázornění stavu dokončení výrobku jsou v programu naimplementovány hodiny, které označují přibližný čas, během kterého se výrobek staví. Pokud je možné zobrazit cílový stav výrobku, vidíme na kolik procent je výrobek postavený, jak je znázorněno

na Obrázcích 5.9 a 5.10.



**Obrázek 5.12** Ukázka několika výrobků najednou v aplikaci pro Microsoft Hololens

V aplikaci existuje možnost zvolení několika vozíků či stanic najednou. To lze udělat i když si uživatel již jedno stanoviště zvolil a nějakým způsobem s ním manipuloval. V tomto případě zvolený vozík/stanice zůstane ve stejné pozici se stejnou orientaci a další vozíky či stanice se přidají pro další manipulaci. Lze to udělat i obráceně, tj. nejdříve zobrazit všechna stanoviště, potom zvolit jenom jedno. Na Obrázku 5.11 je představen jeden model auta na vozíku, na Obrázku 5.12 je znázorněno několik vozíků najednou.

# 6 Závěr

## 6.1 Splněné cíle

Aplikace zobrazující stav výrobní linky disponuje funkcionalitami, které byly součásti zadaných cílů: komunikace s centrálním serverem, zobrazení výrobků, stahování modelů ze serveru a zobrazení budoucího stavu.

Myslím si, že hlavním cílem vytvořené aplikace nebyly konkrétní kroky pro její realizaci, ale celkové inovativní řešení umožňující ukázat uživateli stav dokončení výrobků. Díky vývoji aplikace v rozšířené realitě je uživatel schopen výrobky nejen pozorovat, ale i s nimi interagovat, měnit jejich pozici, orientaci a rozměry tak, aby mu to nejvíce v konkrétní situaci vyhovovalo. Pro ukázku například na výstavě lze výrobky zvětšit na jejich skutečnou velikost, v kanceláři se mohou pozorovat zmenšeně, atd.

Brýle rozšířené reality Microsoft Hololens se nemusí používat vždy při sledování výroby. Když se testuje výrobní linka, není jich třeba. Pro zjednodušení lze používat aplikaci implementovanou na PC. Aplikace na brýlích a na několika PC přitom mohou běžet zároveň. Každá funguje jako klient dotazující centrální server a server poskytující modely součástek.

Aplikace umožňuje snadné rozšíření o další výrobky a modely. Po přidání nebo odstranění modelů na serveru se změny hned objeví v aplikaci zobrazující výrobky a nemusí se přitom znovu kompilovat. Pokud se na výrobní lince vyrábí nové druhy výrobků, je možné jednoduché rozšíření aplikace díky vytvořenému interface pro syntaktickou analýzu digitálního dvojčete. V tomto případě je už nutné kompilovat znovu.

Rozšířením základních funkcí aplikace byla realizována možnost ukázat součinnost několika vozíků nebo stanic zároveň. Tato funkcionalita umožní uživateli bez jakéhokoliv zásahu sledovat všechny stanoviště na lince najednou. Každý vozík zvlášť se dá nastavovat podle potřeb uživatele nebo mazat, když jej nepotřebuje.

## 6.2 Budoucí vývoj

Výše uvedené funkcionality zobrazovače stavu dokončení výrobků nejsou jediné, které se dají v aplikaci implementovat. Existují další možnosti, které ji mohou jednak rozšířit, jednak vylepšit

V budoucnu by mohla aplikace ukazovat i další diagnostická data - například, jestli se nějaká součástka nebo výrobek neopozdil, nebo jestli je na vozíků špatně postavený nebo pokažený výrobek.

Detekce a analýza chyb a neočekávaného chování ve výrobě mohou být implementovány mnoha způsoby, například pomocí komunikace se serverem, rozšířené syntaktické

analýzy digitálního dvojčete, sledování času postavení výrobků a detekcí velkého zpoždění. Díky této funkcionalitě se o chybě dozví i mimo továrnu, teoreticky může být i přibližně znázorněn konkrétní problém ve výrobě.

Místo syntaktické analýzy digitálního dvojčete linky by se data ze stanovišť mohla klientovi přeposlat přímo ze serveru. Pak by se struktura aplikace rozšířila o další možnost zpracování dat.

Pro zlepšení distribuce by se v budoucnu ze serveru mohly stahovat i syntaktické analyzátory, které implementují `Parser`, aby nebylo nutné při jakékoliv změně a přidání nových druhů výrobků aplikaci kompilovat.

Velmi dobrým zlepšením aplikace by byla ukázka cílového stavu přímo na vyrábějícím se výrobku. Například na struktuře z LEGO kostiček by se průhledné kostičky mohly zobrazovat na vozíku skutečné výrobní linky. Cílový stav je ve stavu výrobní linky popsán jen pro struktury z LEGO kostiček, ale teoreticky by se mohl zjistit z dynamicky vytvořeného planu řídicího programu. Pak v AR pomocí markerů by se našla popsaná v *problemu* stanoviště a na příslušná místa by se přidaly průhledné modely součástek. Po přidání na stanoviště skutečných součástek by se průhledné modely postupně odstraňovaly.

# Literatura


[1] Detlef Zuehlke. "SmartFactory—Towards a factory-of-things". In: *Annual Reviews in Control* 34.1 (1. dub. 2010), s. 129–138. ISSN: 1367-5788.

[2] Mohd Aiman Kamarul Bahrin et al. "Industry 4.0: A review on industrial automation and robotic". In: *Jurnal Teknologi* 78.6 (28. červ. 2016). Number: 6-13. ISSN: 2180-3722.

[3] Marco S. Reis a Geert Gins. "Industrial Process Monitoring in the Big Data/ Industry 4.0 Era: from Detection, to Diagnosis, to Prognosis". In: *Processes* 5.3 (zář. 2017). Number: 3 Publisher: Multidisciplinary Digital Publishing Institute, s. 35.

[4] Alan B. Craig. *Understanding Augmented Reality: Concepts and Applications*. Ed. Alan B. Craig. Boston: Morgan Kaufmann, 1. led. 2013, s. 39–67. ISBN: 978-0-240-82408-6.

[5] *What is Augmented Reality (AR) and How does it work*. [online]. URL: https://thinkmobiles.com/blog/what-is-augmented-reality/ (cit. 21. 04. 2020).

[6] Rajeev Sharma a Jose Molineros. "Role of computer vision in augmented virtual reality". In: Stereoscopic Displays and Virtual Reality Systems II. Sv. 2409. 30. břez. 1995, s. 220–231.

[7] Masuko Ishii et al. "Application of Superimposition-Based Personal Identification Using Skull Computed Tomography Images". In: *Journal of Forensic Sciences* 56.4 (2011), s. 960–966. ISSN: 1556-4029.

[8] James D. Westwood. *Medicine Meets Virtual Reality 19: NextMed*. IOS Press, 2012. ISBN: 978-1-61499-021-5.

[9] Francesco De Pace, Federico Manuri a Andrea Sanna. "Augmented Reality in Industry 4.0". In: *American Journal of Computer Science and Information Technology*. 2018.

[10] A. Y. C. Nee a S. K. Ong. "Virtual and Augmented Reality Applications in Manufacturing". In: *IFAC Proceedings Volumes*. 7th IFAC Conference on Manufacturing Modelling, Management, and Control 46.9 (1. led. 2013), s. 15–26. ISSN: 1474-6670.

[11] Yongxin Liao et al. "Past, present and future of Industry 4.0 - a systematic literature review and research agenda proposal". In: *International Journal of Production Research* 55.12 (18. červ. 2017), s. 3609–3629. ISSN: 0020-7543, 1366-588X.

[12] Steven J. Vaughan-Nichols. "Augmented Reality: No Longer a Novelty?" In: *Computer* 42.12 (pros. 2009), s. 19–22. ISSN: 1558-0814.

[13] T. Haritos a N.D. Macchiarella. "A mobile application of augmented reality for aerospace maintenance training". In: sv. 1. *24th Digital Avionics Systems Conference*. Říj. 2005, 5.B.3–5.1.

[14] Nirit Gavish et al. "Evaluating virtual reality and augmented reality training for industrial maintenance and assembly tasks". In: *Interactive Learning Environments* 23.6 (2. lis. 2015), s. 778–798. ISSN: 1049-4820.

[15] Francesco Ferrise Riccardo Masoni. *Supporting Remote Maintenance in Industry 4.0 through Augmented Reality*. In: *American Journal of Computer Science and Information Technology*. Břez. 2018.

[16] J. Postel. *Transmission Control Protocol*. [online]. Zář. 1981. URL: https://www.hjp.at/doc/rfc/rfc793.html (cit. 25. 03. 2020).

[17] *Learn the tools and architecture - Mixed Reality*. Library Catalog: docs.microsoft.com [online]. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/development (cit. 02. 04. 2020).

[18] *TcpClient Class (System.Net.Sockets)*. Library Catalog: docs.microsoft.com [online]. URL: https://docs.microsoft.com/en-us/dotnet/api/system.net.sockets.tcpclient (cit. 30. 03. 2020).

[19] *Node.js Documentation*. [online]. URL: https://nodejs.org/docs/latest-v13.x/api/ (cit. 30. 03. 2020).