
**Budapesti Műszaki és Gazdaságtudományi Egyetem**
Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai Tanszék

Boros István

Tóth Szilveszter

# IPARI MUNKAVÉGZÉST TÁMOGATÓ KERETRENDSZER AUGMENTED REALITY MÓDSZEREK HASZNÁLATÁVAL

KONZULENS

Dr. Forstner Bertalan

BUDAPEST, 2018

# Tartalomjegyzék

**Összefoglaló** ..... **4**
**Abstract** ..... **5**
**1 Ipar 4.0: a negyedik ipari forradalom** ..... **6**
1.1 Történelem ..... 6
1.2 A vízió ..... 7
1.2.1 Az Ipar 4.0 fő szempontjai: ..... 7
1.3 Megvalósítás ..... 9
1.4 Kutatásunk témája ..... 10
**2 Az okos gyártás technológiái** ..... **11**
2.1 Kiber-fizikai gyártósorok [3] ..... 13
2.2 Virtuális és kiterjesztett valóság megjelenése az iparban ..... 14
2.2.1 Virtuális valóság ..... 15
2.2.2 Kiterjesztett valóság ..... 17
2.3 Kommunikációs architektúrák ipari környezetben ..... 23
2.3.1 Hagyományos architektúra ..... 23
2.3.2 Javasolt architektúra ..... 25
**3 Általunk javasolt megoldás** ..... **26**
**4 Webszerver** ..... **29**
4.1 Célkitűzés ..... 29
4.2 Felhasznált technológia ..... 29
4.3 Kommunikációs protokoll ..... 29
4.3.1 JSON üzenet felépítése ..... 30
4.4 A kiajánlható rendszer modellje ..... 33
4.5 Egy rendszer modellezése ..... 34
4.6 A kiajánlható rendszer beavatkozási modellje ..... 35
4.7 A rendszerekhez rendelt interaktív útmutatók kezelése ..... 36
4.8 Az útmutatóhoz tartozó adatok konverziója ..... 37
**5 Útmutató szerkesztő** ..... **39**
5.1 Célkitűzés ..... 39
5.2 Megvalósítás ..... 39
5.3 Útmutató létrehozása ..... 41

5.4 Felhasználói tapasztalok ..... 42
**6 Kliens ..... 43**
6.1 Célkitűzés ..... 44
6.2 Felhasznált technológia ..... 44
6.3 Első kísérleti fejlesztés ..... 45
6.4 Felhasználói felület felépítése ..... 46
6.4.1 Rendszercsempe ..... 47
6.4.2 Rendszerpanel ..... 48
6.4.3 Útmutató megjelenítő bemutatása ..... 51
**7 Új rendszer monitorozása és megjelenítése a kliens segítségével ..... 55**
7.1 Raspberry Pi csatolása ..... 56
**8 Integrálási tapasztalok gyártósori beszállítótól ..... 59**
**9 Összefoglaló ..... 62**
**10 Irodalomjegyzék ..... 64**

# Összefoglaló

Napjainkban sok más ágazat mellet az ipar is rohamos ütemű fejlődésen megy keresztül. A változó igények és a moduláris gyártás megjelenésével, szükségszerűvé válik a bevett módszerek eljárások fejlődése is.

A gyártási folyamat során előforduló eszközökben található szenzorok nagy mennyiségű adatot generálnak, melyek általában egy központi helyen érhetők el, ez azonban korlátozza az ebből kinyerhető információ mennyiségét, minőségét. Ezen problémára megoldást jelenthetnének a mobil klienssel is rendelkező SCADA rendszerek, azonban ezek a teljes rendszer képét jelenítik meg, ami megnehezíti a kényelmes navigációt. További problémát jelent ezen kliensek esetén, hogy az információ megjelenítéséhez szükség van kéz használatára, ami akadályozhatja a munkavégzést. Az imént említett problémák megnehezítik a gyártást felügyelő alkalmazottak gyors és hatékony döntéshozását. Kihívást jelentő feladat, hogy a megfelelő információ, a megfelelő helyen és megfelelő időben jelenjen meg a gyártást támogató személyzet számára, oly módon, hogy ez a munkavégzést minél kisebb mértékben befolyásolja.

A gyártás során gyakran előforduló probléma, hogy a munkafolyamatokat végző gépek meghibásodnak. Ez jobb esetben csak lassítja, rosszabb esetben meg is béníthatja a teljes folyamatot, mely már rövidebb kiesés esetén is komoly anyagi károkat okozhat. Jelenleg egy gép kiesése esetén hozzáértő szakemberekre van szükség a javítás elvégzéséhez, akik gyakran papír alapú dokumentációk segítségével kísérelik meg a probléma elhárítását, mely sok időbe telhet és a folyamat során könnyen félreértések történhetnek. Kihívást jelentő feladat erre a problémára egy olyan megoldást találni, amely különösebb szaktudás szükségessége nélkül lehetővé teszi a javítási feladat elvégzését. Mindezt oly módon, hogy a javítási időt és az esetleges félreértések számát a lehető legjobban lecsökkentjük.

A fent említett kihívások megoldásában segítségét jelenthetnek a különböző Augmented Reality eszközök használata. A kutatási eredmények validálására elkészült egy Microsoft HoloLens alapú, kontextus specifikus monitorozó és javítási útmutató prototípus keretrendszer, amely a fent említett szempontok alapján próbál egy általános megoldást kínálni.

# Abstract

Nowadays, along with many other sectors, the industry is undergoing rapid change and growth. With the emergence of changing needs and demands and the appearance of modular production, the development of well-established methods also becomes inevitable.

Sensors in the manufacturing process generate a large amount of data that are usually available in a central location, but this limits the amount and quality of the information that can be extracted. SCADA systems with mobile clients could be the solution to this problem, but they display the entire system image, which makes convenient navigation difficult. There is a further problem with these clients, the use of hands is needed to access the information, which may hinder work. The aforementioned problems make quick and effective decisions-making difficult for the production supervisory staff. A challenging task is to provide the right information at the right place and the right time for the support staff in a way that affects the work as little as possible.

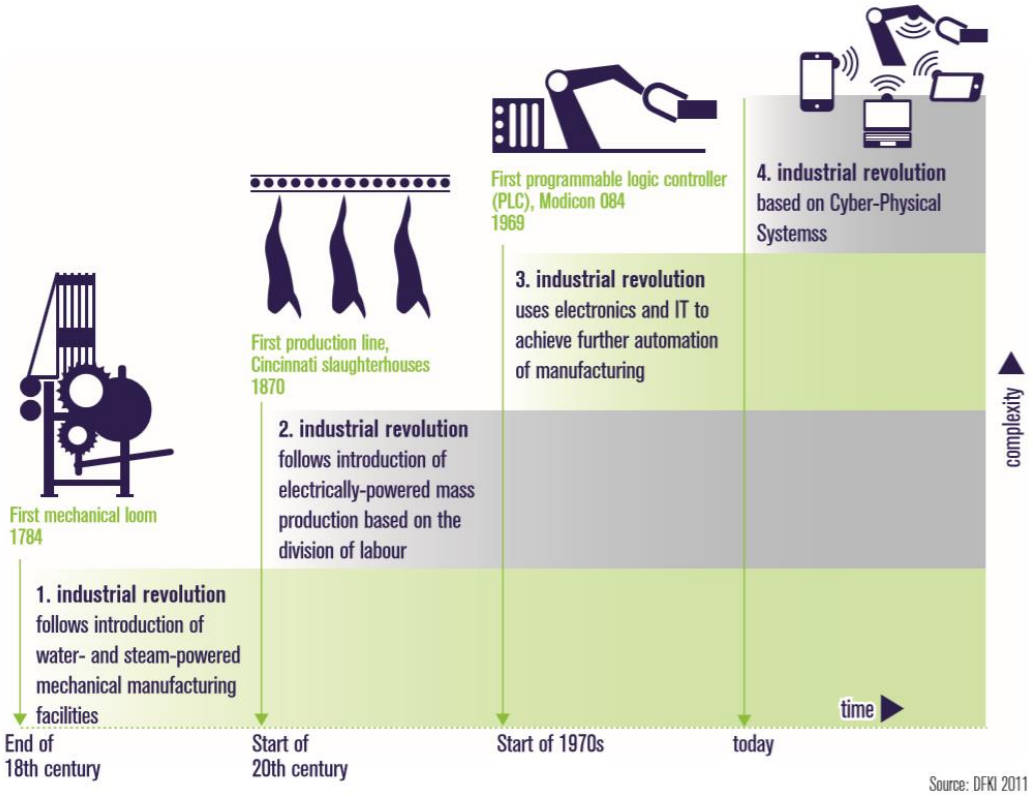
A problem often occurring during the manufacturing process is the failure of machines, that perform the work processes. At best this is only slowing down, but at the worst, it can halt the entire manufacturing line. Currently, failure of a machine may require skilled personnel to perform the repairs, who often try to solve the problem with the help of paper-based documentation, which may cause longer repair times, and in the process, misunderstanding can easily occur. A challenging task is to find a solution to this problem that makes it possible to perform repair tasks without the need for special expertise. All this in a way, that minimizes the repair time and the number of possible misunderstandings as much as possible.

The use of Augmented Reality technology could be an indispensable component to solve the challenges mentioned before. To validate our assumptions, a Microsoft HoloLens based monitoring and repair instructions prototype was developed, in line with the aforementioned needs, to offer a general solution to the problem.

# 1 Ipar 4.0: a negyedik ipari forradalom

## 1.1 Történelem

Az „ipari forradalom” kifejezést a közgazdász Arnold Toynbee tette ismerté az 1884-ben publikált jegyzetei által, melyek az 1760 és 1840 közötti időszak változásairól szóltak. Azóta történészek előszeretettel használják ezt a szókapcsolatot olyan időszakok meghatározására melyben a technológia innovációk nagy hatással vannak a társadalomra.




**1. ábra Az ipari forradalom négy szakasza**[1]

Az iparosítás a 18. század végi mechanikus gyártóberendezések bevezetésével kezdődött, amikor a gépek, mint például a mechanikus szövőszék forradalmasították az áruk előállításának módját. Ezt az első ipari forradalmat egy második követte, amely a 20. század fordulóján kezdődött, magába foglalva a villamos energiát felhasználó tömegtermelést munkamegosztás alapján. Ezt váltotta fel a harmadik ipari forradalom, mely az 1970-es évek elején kezdődött és tart a mai napig is. Ez a fázis az elektronikát és

[1] Forrás: [1]

az információs technológiákat alkalmazza a gyártási folyamatok fokozott automatizálása érdekében, amivel a gépek nem csak a kézi munka jelentős részét, hanem bizonyos szellemi munkákat is átvettek (1. ábra). [1]

## 1.2 A vízió

Az üzleti folyamatok lebonyolítását egyre nagyobb mértékben befolyásolják az ellátási lánc tevékenységeiben alkalmazott információs technológiák. Többen meg vannak győződve arról, hogy ezen változások által keletkező lehetőségek kiaknázásához le kell térni a megszokott útról és megnyitni az ajtót az ipar egy új korszaka, paradigmája előtt.

Az „Industrie 4.0” koncepció 2011-ben fogalmazódott meg Németországban, melynek neve előzetesen utal a negyedik ipari forradalomra. Ekkor írták le és foglalták össze a gyártási technológiák változásainak sorát és határozták meg a koherens politikai keretrendszer prioritásait a német ipar globális versenyképességének fenntartása érdekében. [2] Azóta ezen szófordulat rendkívül népszerűvé vált, mostanra világszerte tanácskoznak kutatók és szakemberek egyaránt a gyártási lánc digitális transzformációjának módszereiről és paradigmáiról.

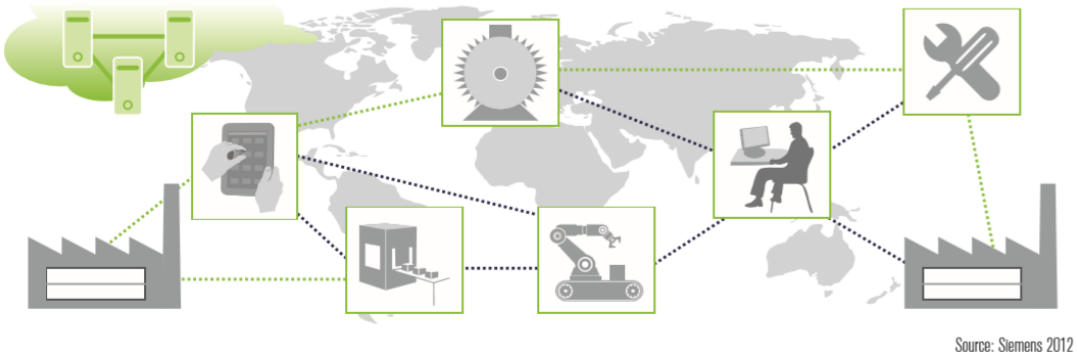
Az Ipar 4.0 a gyártási folyamatok egy olyan, technológián alapuló átszervezését írja le, melynek keretében az eszközök autonóm módon kommunikálnak egymással az érték lánc mentén: a jövő egy olyan „okos” gyárát létrehozva ezzel, amelyben a számítógépek által vezérelt rendszerek nyomon követik a fizikai folyamatokat, létrehozzák a fizikai valóság digitális mását és decentralizált döntéseket hoznak önszerveződő mechanizmusok alapján. A koncepció egyik fő hajtóereje a feldolgozóipar nagymértékű számítógépesítettsége, ahol a fizikai objektumok gond nélkül integrálódnak az információs hálózatba. Az előzők következményeként a gyártási rendszerek olyan változásokon fognak átesni, melyek lehetővé teszik a gyártás valós idejű menedzselését, a megrendelés pillanatától kezdve a szállításig.

### 1.2.1 Az Ipar 4.0 fő szempontjai:

#### 1.2.1.1 Értékhálózatokon keresztüli horizontális integráció

Az üzleti tervezés és a gyártás különböző állomásain használt IT rendszerek integrációja, amelyek felelősek az anyagok, energia és információ cseréjéért, a vállalton

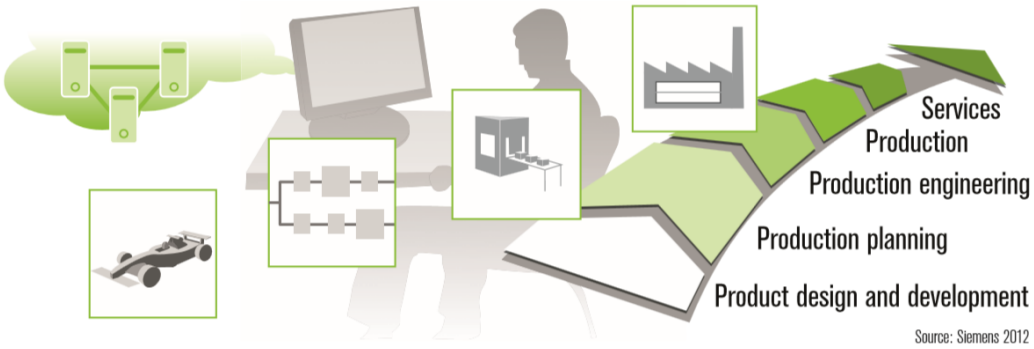
belül és több partner vállalat között is. A cél egy end-to-end megoldás létrehozása a fogyasztók számára (2. ábra).




**2. ábra Értékhálózatokon keresztüli horizontális integráció**[2]

### 1.2.1.2 Mérnöki munka integrációja a teljes értéklánc minden pontján

A cél a mérnöki munka end-to-end digitális integrációja oly módon, hogy a digitális és valós világ integrációját megvalósítja a termék teljes életciklusán keresztül, több vállalat között is, a vásárlók igényeit szem előtt tartva (**3**. ábra).

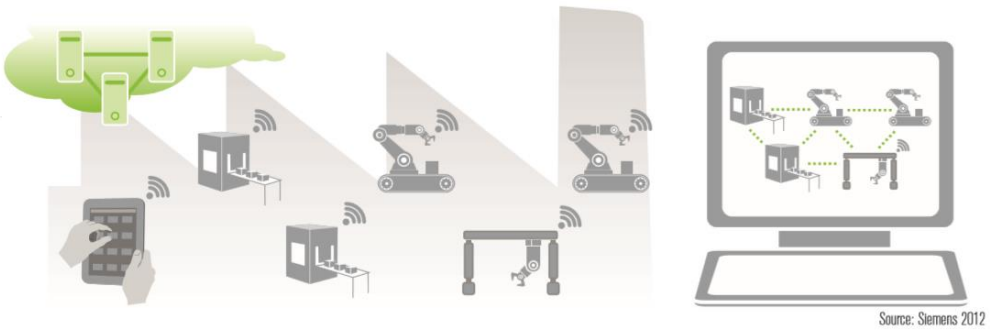



3**. ábra Mérnöki folyamatok integrációja a teljes értékláncon**[2]

### 1.2.1.3 Vertikális integráció és hálózatba kapcsolt gyártási rendszerek

A különböző hierarchia szinteken használt IT rendszerek integrációja (szenzorok, gyártás irányítás, gyártás felügyelet és menedzsment szoftverek). Célja, hogy a jövő gyárában a fix, előre meghatározott gyártási folyamat helyett konfigurációs szabályok szerint esetektől függően létrejöhessen a gyártási topológia automatikusan (4. ábra).

[2] Forrás: [1]

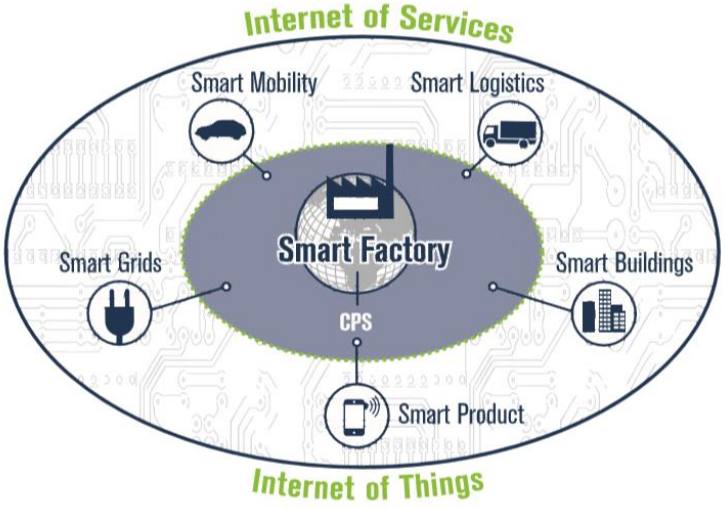



**4. ábra Vertikális integráció és hálózatba kapcsolt gyártási rendszerek[3]**

## 1.3 Megvalósítás

Az említett fejlesztések a klasszikus ipar és szolgáltatások közti megkülönböztetést idejétmúlttá teszik, mivel a digitális technológia által összekapcsolt ipari termékek és szolgáltatások olyan hibrid termékeket hoznak létre, melyek egyértelműen nem sorolhatók sem az áruk, sem a szolgáltatások közé. [2]

Az eddig említett változások eléréséhez nem elég csupán a gyárakat okosabbá tenni. A dolgok és szolgáltatások internete (Internet of Things and Services) éreztetni fogja a jelenlétét az összes kulcsterületen, mint az okos energiaellátó hálózatok, a fenntartható szállítmányozási stratégiák (okos szállítmányozás, okos logisztika), okos épületek és az okos egészségügy (5. ábra).




**5. ábra Ipar 4.0 ökoszisztéma elemei[3]**

[3] Forrás: [1]

Mi ebben a dolgozatban azonban az okos gyárakra és az ott használható technológiákra helyezzük a hangsúlyt, így a továbbiakban ezekkel foglalkozunk behatóbban.

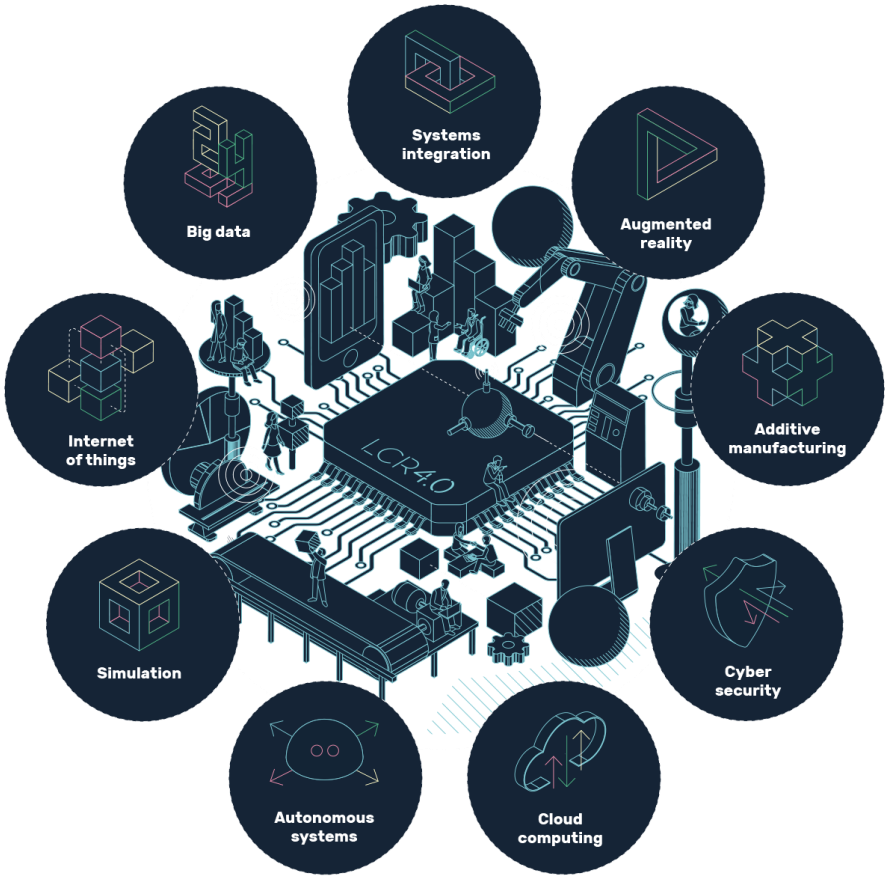
## 1.4 Kutatásunk témája

Az ipari fejlődés negyedik hulláma nem csak a gyártás folyamatára és a fogyasztókra lesz nagy hatással, hanem a gyárban dolgozó személyzetre is. Ahhoz, hogy a termelési folyamat megnövekedett komplexitásával minél jobban meg tudjanak birkózni, szükségük lesz olyan intelligens asszisztens rendszerekre, melyek a rendelkezésre álló adatok elemzésével képesek az adott helyzetben a lehető legjobb döntés meghozatalát támogatni. Az ilyen rendszerek azonban még gyerekcipőben járnak.

A kutatásunk során célunk egy olyan kontextusfüggő döntéstámogató rendszert megtervezni, illetve a megvalósíthatóságát felmérni egy prototípus megvalósításával, mely a felhasználó számára szükséges információkat jeleníti meg a megfelelő helyen, megfelelő időben. Ehhez először megvizsgáljuk az okos gyárak számunkra releváns technológiáit, ezek tulajdonságait, illetve lehetséges felhasználási módjait, majd ezekre építve javaslunk egy megoldást, ami támogatni képes a felmerült igényeket.

# 2 Az okos gyártás technológiái

A gyártási folyamat digitalizációja nem valósítható meg csupán egyetlen új technológia felhasználásával, a kitűzött célok eléréséhez több terület közötti együttműködés megvalósítása szükséges. Az ipar új paradigmájának megalkotásában az alábbi ábrán látható technológiák játszanak kulcsszerepet (6. ábra).




**6. ábra Ipar 4.0 fő technológiái**[4]

Tekintsük át, hogy ezek miként jelennek meg ebben a transzformációban:

- **Dolgok internete:** az internetre kapcsolt eszközök folyamatosan növekvő infrastruktúrája, melyek a végpontok között adatok és utasítások megosztására képesek.

[4] Forrás: http://lcr4.uk/wp-content/uploads/2016/11/big_illustration_8bit_bold.png

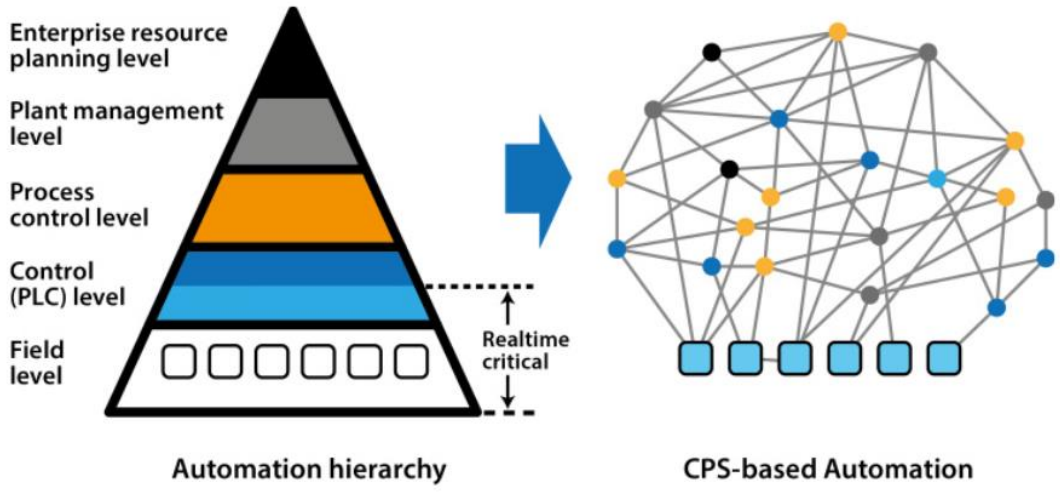
- **Big Data:** a vállalatok hatalmas mennyiségű adatot hoznak létre, melyekből ennek a technológiának a segítségével, eddig fel nem fedezett összefüggéseket vizsgálhatunk meg, amelyek segíthetik a növekedést és a fejlődést.
- **Szimuláció:** segítségével a virtuális világban próbálhatunk ki bizonyos eseteket, fizikai prototípusok építése nélkül, amely így csökkenti a költségeket és nagyobb flexibilitást biztosít kísérletek elvégzéséhez.
- **Additív gyártástechnológia:** gyártási eljárás, amely vékony rétegek lerakásával készít tárgyakat szemben a hagyományos megmunkálással, melynek során egy nagyobb nyers darabból választják le a felesleges anyagot, és a megmaradó rész lesz a késztermék. Egyik legismertebb eszköze a 3D nyomtató. Legnagyobb előnyei a gyors mintapéldány készítés és a kisebb alkatrészek, segédeszközök beszerzése terén domborodnak ki, az átfutási idő radikális csökkentése és a testreszabás kiterjesztése által.
- **Kiterjesztett valóság:** a gyártási folyamatok és a termékek esetén, hatalmas lehetőséget biztosít, hogy az általunk látott dolgokat virtuálisan kiegészíthetjük adatokkal és a nem látható rendszerek elemeivel.
- **Autonóm rendszerek:** a robotika és mesterséges intelligencia eljutott arra a pontra, hogy magas elvárásoknak eleget téve tud különböző feladatokat elvégezni, emberi beavatkozás nélkül.
- **Rendszerintegráció:** a különböző hardverek és szoftverek kombinációja bizonyos esetekben sokkal nagyobb értéket képviselhet, mint külön-külön.
- **Kiberbiztonság:** a hálózatba kötött eszközöknek ellen kell állnia a kívülről történő behatolásnak az információ védelme érdekében.
- **Felhőalapú számítás:** segítségével az IT szolgáltatások költségét és helyigényét csökkenteni tudjuk, ezen felül lehetőséget ad új, innovatív modellek létrehozására.

## 2.1 Kiber-fizikai gyártósorok [3]

A kiber-fizikai gyártósorok (cyber-physical production systems, továbbiakban CPPS) olyan autonóm, kooperatív elemekből és alrendszerekből állnak, amelyek képesek szituációfüggő kommunikációra egymás között, a gyártási szinteken átívelően, a termelést végző gépektől kezdve a logisztikáig.

Ahhoz, hogy ez meg tudjon valósulni, a rendszereknek képesnek kell lenniük a tevékenységük modellezésére és a különböző tevékenységek során végzett cselekedeteik előrejelzésére, továbbá az irányításra. Előreláthatólag az analitikának és a szimulációnak az integrációja lesz az egyik legfontosabb mozzanat ennek elérésében. Ehhez meg kell küzdeni a szenzorhálózatok által generált adatmennyiség feldolgozásával, az adatok biztonságos begyűjtésével és reprezentálásával. Azonban ezen út során képesek leszünk megvalósítani egy olyan ökoszisztémát, ami megfelelő módon tudja támogatni az ember-gép kommunikációt.

A CPPS-ek megjelenése részben meg fogja változtatni a klasszikus automatizációs piramist (7. ábra bal oldala). A hagyományos irányítási szint ugyanúgy jelen lesz, ami magába foglalja a technikai folyamatokhoz közeli PLC-ket a kritikus feladatok lehető leghatékonyabb ellátására, azonban a magasabb szintű hierarchiaszinteket egyre jobban a decentralizált működés jellemzi a CPPS-ekben (7. ábra jobb oldala).




**7. ábra Az automatizáció hierarchiájának dekompozíciója elosztott szolgáltatásokká**[5]

[5] Forrás: [3]

A CPPS lehetővé fogja tenni és támogatni fogja a kommunikációt az emberek, gépek és termékek között egyaránt, mivel képes lesz az adatok gyűjtésére és feldolgozására, bizonyos feladatok autonóm irányítására és megfelelő interfészeken keresztül interakcióba lépni emberekkel.

## 2.2 Virtuális és kiterjesztett valóság megjelenése az iparban

Az ipar 4.0 paradigma szerint egy gyár minden objektuma képes önálló feldolgozásra, és fel van szerelve a megfelelő kommunikációs képességekkel. Egy jelenleg ezt nem alkalmazó gyár esetén ez nemcsak a gépek közötti kommunikációt fogja befolyásolni, ugyanis jelentős hatása lesz az emberek és a technológia közötti kölcsönhatásra is. Ahelyett, hogy az emberek teljesen kimaradnának az automata, önszerveződő gépekből és komponensekből álló kiber-fizikai rendszerből, integrálódniuk kell a rendszerbe a megfelelő képességek és tehetségek felismerésével és alkalmazásával.

Egy kiber-fizikai struktúra leírja a kapcsolatot az emberek és a kiber-fizikai rendszer (továbbiakban CPS) között, amely feloszthatő egy fizikai komponensre és egy virtuális, digitális komponensre. A kommunikáció az ember és a CPS között történhet direkt manipuláció segítségével (például megnyom egy gombot egy rendszeren), vagy történhet egy közvetítő felhasználói felületen keresztül. Felmerül a kérdés, hogy hogyan érdemes reprezentálni az ember számára a releváns információt, és hogyan érdemes megvalósítani a beavatkozási lehetőségeit?

Egyre több CPS implementáció jelenik meg, melyeknek kiterjedt képességük van az adatok összegyűjtésére, cserélésére (küldés-fogadás) és feldolgozására, ami megnövekedett információmennyiséghez vezethet. Ezzel együtt teljesíteni kell számos követelményt is úgy mint, az adatok beszerzése, aggregálása, reprezentálása és újrafelhasználása a felhasználók és a rendszer számára egyaránt. Ezeknek a követelményeknek a teljesítése szükséges, ugyanis a növekvő számú adatot szolgáltató összekapcsolódott komponensek adataira csak úgy tud az ember helyes stratégiai döntést hozni, hogy az eseményeket megérti és egy könnyedén érthető vizualizációt kap a jelenlegi gyártási folyamatokról. A rendszernek pedig azért van szüksége a követelmények teljesítésére, hogy a CPS könnyedén tudjon létrehozni kapcsolatot a már meglévő gyártási, információs technológiákkal standardizált, platform-független interfészeken keresztül.

A közvetítő interfész a felhasználó és a CPS között történhetne akár úgy, ahogy a SCADA rendszereknél is láthatjuk, hogy például egy mobil alkalmazásban hozzáférünk a rendszerek szenzoradataihoz. Azonban egy ilyen rendszer esetén a korábbiakra hivatkozva egy olyan bonyolultságú rendszerről beszélhetünk, melyhez szükség van a teljes rendszer reprezentációjára, amit egy egyszerű mobilalkalmazás, amely csak az információt hordozza, nem képes megvalósítani úgy, hogy egy jól átlatható információt kapjunk a rendszerről, melyek alapján képesek vagyunk helyes döntést hozni, ha szükségszerű.

A közvetítő interfész jól megvalósítható virtuális és kiterjesztett valóság eszközökkel. A továbbiakban erről a két technológiáról és ezeknek a lehetőségeikről fogunk tárgyalni. [4]

### 2.2.1 Virtuális valóság

A virtuális valóság nem egy új technológia, inkább több régebbi technológia egybeolvasztásának eredménye. Olyan technológiák, amelyek például számítógépes grafikával, megjelenítéssel vagy számítógépes interfészek létrehozásával foglalkoznak. A virtuális valóságot gyakran azonosítják a virtuális környezetekkel (Virtual Environments). Számos definíciója van a virtuális valóságnak, ezek közül felsorolunk néhányat:

- A valós világ számítógép által generált szimulációja
- Szintetikus környezetben való részvétel illúziója, nem pedig az ilyen környezet külső megfigyelése
- Számítógép által generált háromdimenziós környezet szimulációja, ahol a felhasználó láthatja és manipulálhatja a környezet tartalmát

A virtuális valóság egy háromdimenziós, interaktív, számítógép által generált környezetből áll, amely modellezhet valós, vagy képzeletbeli világot is.

A virtuális valóság megjelenítéséhez kétféle mód terjedt el. Az egyik egyszerűen megjeleníti a virtuális világot egy számítógép monitorján. A másik az úgynevezett Immersive Virtual Reality, amikor a képernyőt felváltja egy fejre helyezhető képernyő vagy szemüveg, amely képes a világot háromdimenzióban a felhasználó látóterét kitöltő módon megjeleníteni a képet. A meghajtáshoz a szemüvegen kívül szükséges egy

nagyteljesítményű számítógép, vezérlő eszközök, illetve pozíció nyomon követő eszközök. [5]

**8. ábra VR Box virtuális valóság fejre helyezhető szemüveg[6]**

A virtuális valóságot először az űrkutatási és a repülőgép iparban használták, ahol a mára már jól ismert repülőgép szimulátorok tökéletes kidolgozása volt a cél. Manapság azonban számos területen megjelenik ez a technológia úgy mint: termék vizualizáció, egészségügyi alkalmazások, betanítás/tanító programok, karbantartási rendszerek, építőipar. Az építőiparban például az első megoldások célja az volt, hogy be tudja járni a felhasználó a látványterveket. [5]

Az ipar 4.0 paradigmára tekintve a virtuális valóság lehetővé teszi a felhasználó számára, hogy szimulálja és interaktív módon vizsgálja, felfedezze a CPS-alapú gyártási rendszer viselkedését, melyet a gyártási folyamatok valóságnak megfelelő feltérképezésével, szimulációjával érhetünk el. Ez lehetőséget biztosít, hogy információt kapjunk jól átlátható módon a rendszer komponenseiről jelenidőben, illetve ha további feltételek adottak, szimuláljuk a jövőbeni viselkedést. [4]

Számos területen problémát jelent, hogy a virtuális valóságot megjelenítő fejre helyezhető szemüvegek ellehetetlenítik a valóságban létező elemekkel való interakciót, ugyanis a felhasználó teljes egészében a virtuális valóságot látja. Például egy eszköz összeszerelését segítő tartalom sokkal hatékonyabb lenne, ha a valóságban láthatná a felhasználó az alkatrészeket és az instrukciókat egyszerre. Az ilyen korlátozások miatt az iparban a kiterjesztett valóság sokkal nagyobb szerepet kap szemben a virtuális valósággal.

[6] Forrás: [13]

### 2.2.2 Kiterjesztett valóság

A kiterjesztett valóság egy olyan technológia, amely háromdimenziós tartalomként feltérképezi a teret, és számítógép által generált modelleket helyez el benne, mindezt a felhasználó szemszögéhez viszonyítva valós időben megjelenítve úgy, hogy eközben a felhasználó látja a valóságot is.

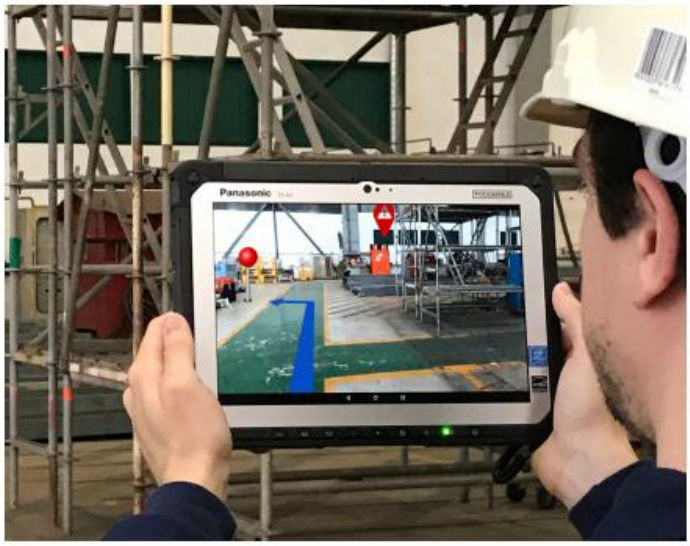
Ez a technológia elfogadottá és népszerűvé vált számos területen, egészen a fogyasztóknak szánt játékalkalmazásoktól az ipari alkalmazásokig. Különösen a gyártás és összeszerelés területek táplálják a fejlesztést, ugyanis a múltban már jól bizonyította ez a technológia, hogy komoly fejlődés érhető el az alkalmazásával.

Például komoly mértékben le lehet csökkenteni az összeszerelés betanítását azáltal, hogy a digitális objektumok a valós világban jelennek meg, és megmagyarázzák a felhasználó számára, hogy melyik alkatrészt hova kell helyezni. Ez természetesen eltünteti annak szükségességét, hogy hosszasan tanulmányozzák a részletes leírásokat, melyek az összeszerelésre vonatkoznak, hiszen ezt teljes egészében kiváltja a háromdimenziós tartalom. [6]

Egy másik jó példa a gyártási területre a különböző rendszerek monitorozása és azokba való beavatkozás, akár javítása. Kiterjesztett valóság segítségével, megvalósítható az, hogy egy rendszerről valós idejű, releváns információkat kapjunk, ha például a rendszer előtt állunk a valóságban. Mindemellett, ha megoldjuk azt, hogy a rendszer képes legyen előre jelezni egy alkatrész meghibásodását, akkor a technológia segítségével be tudunk avatkozni a rendszer működésébe, és javítani tudjuk a problémát útmutató segítségével. Ezzel lényegesen le tudjuk csökkenteni a kiesések számát, valamint képes lesz egy a rendszerhez szakértői tudással nem rendelkező ember is megoldani a problémákat.

Habár a kiterjesztett valóság nagy potenciált mutatott az útmutató alapú összeszerelés, javítás alkalmazásokban, a korábban elkészített eszközök erősen korlátozták a teljes potenciál elérését. A legtöbb eszközt leginkább kutatási céllal készítették, nem pedig kereskedelmi használatra, ami azt jelentette, hogy nem elérhetőek a piacon, valamint aki, ha hozzá is jutott egy ilyen eszközhöz, nem bízhatott a stabilitásában, zökkenőmentes futásában. Napjainkig a kiterjesztett valóság alkalmazások többnyire tabletekre, telefonokra, esetleg nagyméretű vezetékes fejre helyezhető szemüvegekre voltak fejlesztve a technológiai korlátok miatt. Ennek eredményeként

számos kiterjesztett valóság tartalom nem teszi lehetővé a kezet nem lefoglaló (érintésmentes) munkavégzést, akadályozva ezzel a felhasználó mozgását és interakcióját a valós fizikai világ objektumaival. Például egy javítási útmutató elvégzése során, ha ez egy tableten működik, akkor a felhasználó kénytelen felváltva fókuszálni az útmutató lépéseinek értelmezésére, a tablet képernyőjén lévő utasítások megfigyelésére és a tényleges lépések elvégzésére.




**9. ábra Ipari környezetben használt kiterjesztett valóság alkalmazás tableten megjelenítve**[7]

Azonban az elmúlt években a számítási teljesítmény és a megjelenítési technológiák fejlődéseinek köszönhetően megjelentek olyan új, kereskedelmi használatra szánt fejre helyezhető eszközök, amelyeket képesek vagyunk kezet nem lefoglaló módon (hands free) használni. Ezeknek az eszközöknek az előbb említett előnyük mellett a másik jelentős fejlesztésük, hogy a szemüvegbe bele van építve az alkalmazásokat futtató számítógép, így nem kell semmilyen módon hozzácsatlakoztatni egy asztali számítógéphez vagy bármilyen komolyabb számítási kapacitással rendelkező eszközhöz a használathoz. Ilyen eszköz például a DAQRI Smart Helmet és a Microsoft HoloLens. [6]

A kiterjesztett valóság az emberi érzékelés számítógépes támogatását jelenti virtuális objektumok használatával. Az ipar 4.0 mintához kötve a kiterjesztett valóság megjelenítésére képes okostelefonok, tabletek, de főleg az előbb említett szemüvegek a

[7] Forrás: [7]

leghatékonyabb eszközök a CPS-sel való interakcióra és a CPS-ből kinyert információk megjelenítésére.

Egy gyárban a bejövő információk sok különböző forrásból származhatnak. Ezek lehetnek például szenzoradatok, technikai dokumentációk, CAD modellek, statikus vagy dinamikus adatok. Az adatok újrafelhasználásához és a későbbi adatok keletkezéséhez fontos készíteni megfelelő sztenderdeket az információk elérése és cserélése érdekében. Az CPS által összegyűjtött adatokat tehát sztenderd platformfüggetlen interfészeken keresztül (mint például OPC UA) tesszük elérhetővé. Ezen a rendszeren keresztül a kiterjesztett valóságot megjelenítő kliensek közvetlenül hozzáférnek az aggregált, előre feldolgozott információhoz. Egy ilyen rendszer számos applikáció szcenáriót támogat főleg kiterjesztett valóság alkalmazásával [4]:

- Karbantartás, szervízelés, interaktív útmutató alkalmazás
- Gyártási folyamatok monitorozása, minőségmenedzsment, CPS státuszának megfigyelése
- Tervezés, gyártási folyamatok szimulációja, CPS-ek viselkedésének vizualizációja


**10. ábra Valós idejű információ kijelzése kiterjesztett valóság alkalmazásban[8]**

### 2.2.2.1 Microsoft HoloLens

A Microsoft HoloLens terméke az első kereskedelmi forgalomra szánt elérhető kiterjesztett valóság szemüveg, amely lényegesen olcsóbb a korábbi, egyénileg készített kísérleti szemüvegekkel szemben, és rendelkezik számos olyan képességgel, amelyek

[8] Forrás: [4]

alkalmassá teszik az ipari környezetre szánt alkalmazások létrehozására. A HoloLens fejlesztői verziója 2016-ban lett kiadva, ekkor kettő főbb termék versenytársa volt ezen a területen: a Google Glass és a Daqri Smart Helmet. A Google Glass szemben a HoloLens-el, számos képességgel nem rendelkezik, ilyen például a térbeli leképezés (spatial mapping) vagy a gesztúrák felismerése. Ezen felül a valós világgal való interakció nehézkes (például input bevitele). A Daqri Smart Helmet direkt ipari felhasználásra van tervezve, azonban ez a fejlesztésünk kezdetekor (2017 eleje) még nem jelent meg. [6]

Mi a Microsoft HoloLens-t használtuk saját alkalmazásunk fejlesztéséhez és teszteléséhez, valamint a Microsoft kiterjesztett valósághoz adott ajánlásait, kiinduló projektjeit és leírásait alkalmaztuk, így a továbbiakban, ha fejre helyezhető szemüvegről tárgyalunk, akkor a HoloLens képességeit vesszük alapul.

A HoloLens egy olyan szemüvegbe épített számítógép, amely tartalmazza a kiterjesztett valósággal kapcsolatos akkori state-of-the-art tudás lehető legtöbb elemét. A HoloLens számításait négy darab Intel Atom x5- Z8100 1.04 GHz Airmont logikai processzor végzi, a megjelenítéshez egy úgynevezett Holografikus feldolgozó egységet használ (HPU/GPU Holographic Processing Unit), 64 Gigabájt Flash memória áll rendelkezésre adataink és alkalmazásain tárolására, illetve 2 Gigabájt RAM memóriát használhatunk fel alkalmazásaink betöltésére (az operációs rendszert is beleértve). Az eszköz 2-3 órán át képes működni akkumulátorról. A feldolgozóegységek kettő darab 720 pixel széles HD (High-Definition) méretű fényforrás kép futtatását végzik, amelyek holografikus lencsék által fényt sugározva, összesen 2,3 millió fénypont értékét határozzák meg. A magas felbontású, térből leképzett 3D tartalmat az eszközbe épített rendszer generálja, amely tartalom tulajdonképpen a szemüveghez képest valós térben elhelyezkedő objektumok, élek és síkok alapján készül, melyhez a rendszer négy beépített környezet-feldolgozó (environment-processing camera) és egy mélységmérő kamerát használ fel (a Microsoft Xbox Kinect termékénél hasonló módon térképezik fel a környezetet). A rendszer a körülötte levő objektumok feltérképezése után meghatároz és engedélyez különböző interakciókat a valós és a virtuális világ viszonya alapján, miközben nyomon követi az eszköz pozícióját. A HoloLens tartalmaz egy úgynevezett inerciális mérőegységet (Inertial Measuement Unit vagy IMU) az elmozdulás, irány meghatározásához (giroszkóp, gyorsulásmérő segítségével), emellett tartalmaz egy RGB kamerát és 4 mikrofont. Számos további beépített képességgel is felruházták az eszközt, mint kézmozdulat felismerés, térbeli hangzás támogatás vagy hangfelismerés. [6]

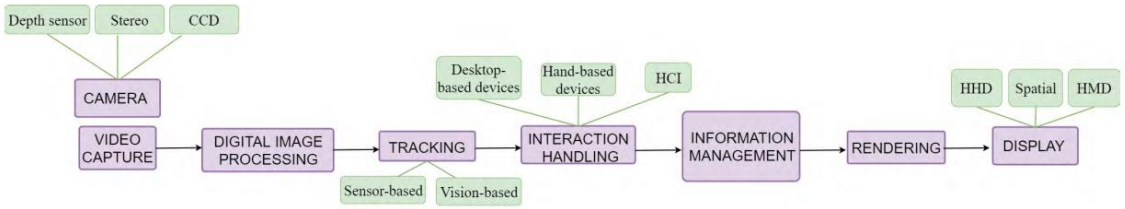
HoloLens-el és ahhoz hasonló eszközökkel könnyedén tudunk létrehozni egy útmutató alapú javítást, összeszerelést segítő alkalmazást, amely kényelmesen végig tudja vezetni a felhasználót a különböző lépéseken, miközben el is tudja végezni azokat a lehető legtöbb szükségtelen mozdulat eltüntetésével. Így nem kell váltani a figyelmet az útmutatót megjelenítő eszköz és a tényleges feladat között, hiszen minden szükséges információ a felhasználó szeme előtt van, és csak akkor kell (azt is érintésmentesen) interakciót végezni, amikor egy-egy résszel elkészült a felhasználó. Véleményünk szerint jelenleg a Microsoft HoloLens terméke a legalkalmasabb egy ilyen típusú alkalmazás megvalósítására.

**11. ábra Microsoft HoloLens[9]**

### 2.2.2.2 Kiterjesztett valóság rendszer működése

Ebben a fejezetben rövid kitekintést teszünk arra, hogy hogyan készíti el a kivetítendő képet a megjelenítő eszköz, milyen funkcionális modulokra van szükség ahhoz, hogy a képen a megfelelő helyen jelenjenek meg az objektumaink.

[9] Forrás: [6]




**12. ábra Egyszerűsített kiterjesztett valóság csővezeték[10]**

A 12. ábra Egyszerűsített kiterjesztett valóság csővezeték12. ábra egy egyszerűsített kiterjesztett valóság csővezetéket reprezentál, ahol a lila színű dobozok jelzik az egyes funkcionális modulokat, amelyeket egy kiterjesztett valóság eszköznek meg kell tudni valósítania.

Először az eszköz kamerája rögzít egy képet (Camera és Video capture), melyet a rendszer feldolgoz (Digital image processing), és megpróbálja megbecsülni a kamera pozícióját a valóságban lévő referenciaobjektumokhoz képest (Tracking). Az ilyen becslés nem csak a kamera/kamerák rögzített képeiből, képfeldolgozással keletkezhet, hanem a belső szenzorok értékeit is rögzítheti, mely információk segítségével nyomon tudjuk követni a referenciaobjektum pozícióját. A pontos pozícionálás szükségszerű, hiszen például a fej mozgatásával nagyon gyorsan és pontosan kell tudni elvégezni az elmozdulásokat, forgatásokat vagy nagyításokat a virtuális objektumokon ahhoz, hogy ne veszítsük el a helyes pozíciót. [7] HoloLens esetén megfelelő referenciaobjektum a térből generált háromdimenziós tartalom, melyet a környezet-feldolgozó, mélységmérő és RGB kamerák segítségével az eszköz folyamatosan frissíti, és a kameraképek és az inerciális mérőegység alapján a lehető legpontosabb nyomon követést valósítja meg.

A nyomon követés számítása után következik az interakció kezelés, mely a felhasználó által adott inputok feldolgozását végzi (Interaction handling). Ezután, ha bizonyos helyi vagy távoli információra van szükség a feldolgozáshoz (például adatra), a rendszer végrehajtja a kérést (Information Management). [7] HoloLens esetén a gesztúrák felismerését (például klikk gesztúra) az interakció kezelést végző modul dolgozza vel.

Az utolsóelőtti lépés a kivetítendő kép generálása, a háromdimenziós térben elhelyezkedő objektumok és adatok alapján egy kettődimenziós kép generálása, amely

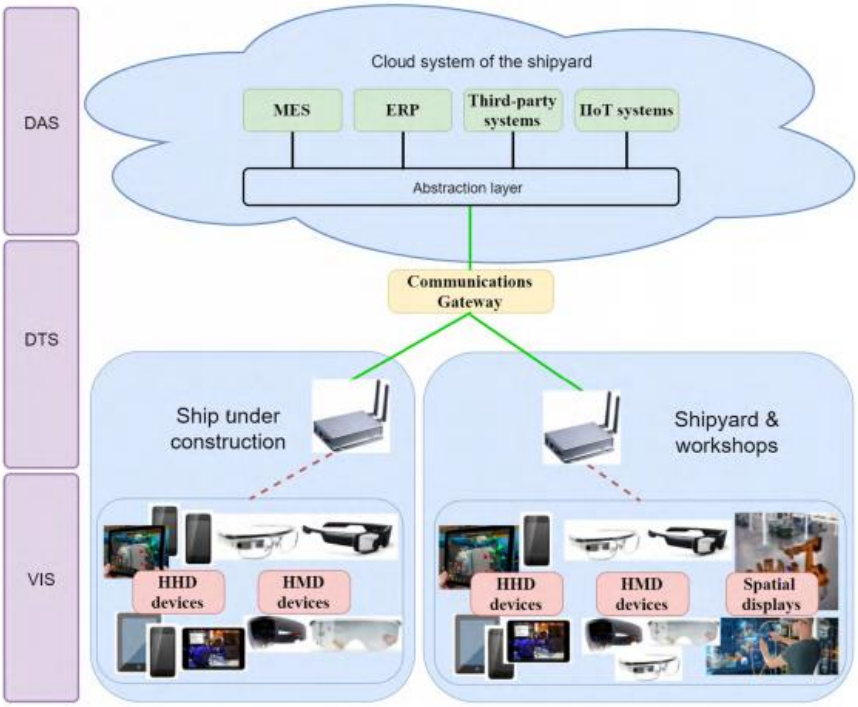
[10] Forrás: [7]

ezután utolsó lépésként kivetíthető az eszköz képernyőjére. HoloLens esetén a transzformált kép a holografikus lencsék kijelzőire kerülnek.

## 2.3 Kommunikációs architektúrák ipari környezetben

### 2.3.1 Hagyományos architektúra

Az iparban korábban a kiterjesztett valóság alkalmazásokhoz (és egyéb más alkalmazásokhoz is) kialakult egy általános architektúra, amely bizonyos szintig teljesítette a felhasználóbarát kliens készítéséhez szükséges követelményeket.




**13. ábra Hagyományos kommunikációs architektúra[11]**

A 13. ábra egy hajógyárra alkalmazott hagyományos kommunikációs architektúrát ábrázol. Bal oldalon látható, hogy három fő rétegre van szétbontva az architektúra kommunikáció szempontjából az adat keletkezésétől, egészen a kiterjesztett valóság szemüvegen megjelenő képért felelős komponensekig:

- VIS: Visualization and Interaction System, azaz a vizualizációért és az interakciók kezeléséért felelős réteg. Ez a réteg magába foglalja az összes

[11] Forrás: [7]

olyan eszközt, amely képes megjeleníteni a gyár eszközei által generált információt, és képes a felhasználótól érkező interakciókat lekezelni. Kiterjesztett valóságra épített architektúra esetén, az összes erre alkalmas eszköz megvalósítja ezt (HoloLens, tablet, mobiltelefon stb.).

- DTS: Data Transport System, adatszállításért felelős réteg. Ez a réteg felelős a (például kérésnek) megfelelő adatok összegyűjtéséért és mozgatásáért a felhő infrastruktúrából a helyi rendszerbe.
- DAS: Data Acquisition System, adatbeszerzést végző réteg. Innen származik a gyárral kapcsolatos összes adat: dokumentációktól kezdve egészen a szenzoradatokig. Ezt általában egy felhő infrastruktúra valósítja meg.

Ez a magas szintű architektúra a növekvő sávszélességigény és az egyre inkább valós idejű alkalmazások igénye miatt, több nehézséget is okozott. A kommunikáció a felhő és a kliens alkalmazás között Wi-Fi-n keresztül történik legtöbb esetben. Sok gyárban problémát okozhat a nagyméretű fém objektumok miatt az elektromágneses terjesztés, ami megzavarhatja a kommunikációt. Ha a kommunikációt javarészt vezetékeken keresztül szeretnénk végrehajtani, akkor a sávszélességet befolyásolhatja az elektromos interferencia, mely a magasfeszültségen működő eszközöktől származhat. Azonban számos helyen komoly javulást eredményez, hogy a kommunikáció vezetéken keresztül csatlakozik a DAS-hoz, és a másik végponton lokálisan egy magas sávszélességű Wi-Fi elérési pont található, amire a kliensek csatlakozni tudnak. A gyors válaszidő kritikus a kiterjesztett alkalmazások területén, hiszen a megfelelő felhasználói élmény eléréséhez az adatokat a lehető leggyorsabb módon szolgáltatnunk kell. Habár a statikus tartalmakat (például dokumentáció) el tudjuk tárolni a lokális tárban, azonban a dinamikusan változó adatokhoz (például szenzoradatok) folyamatos kommunikációra van szükség a felhővel.

Megoldást jelenthet a gyors elérésű központi jellegű proxy szerverek használata, amelyek ideiglenesen eltárolják a dinamikus adatokat, valamint a kliens alkalmazás előre megbecsüli, hogy milyen adatokra lesz szüksége, és ezt a háttérben letölti. Azonban ez a kliens számára energia és erőforráspazarló lehet, mind a többszálúság miatt, mind pedig a szükségtelen lekérések miatt (ha a becslés nem jó). Ha komolyabb számítást kell elvégezni a tartalom megjelenítéséhez, akkor ebben a rendszerben ezt vagy a felhőben

lévő gépeknek vagy kliensnek kell elvégezni, ami növeli a késleltetést és a kliens esetén terhelést is. [7]

### 2.3.2 Javasolt architektúra

Az utóbbi időben számos kutató keresett olyan architektúra alternatívát, amely javítja a válaszidőt és a megjelenítési teljesítményt kliensoldalon. Legtöbb eredmény azt hozta ki, hogy vegyünk használatba lokális, nagy számítási teljesítményre képes számítógépeket, amelyek tárolják a megjeleníteni kívánt adatot, és ha kell, feldolgozzák azokat (például videó feldolgozása, modell generálása stb.).

Az elmúlt években sokan javasolták az úgynevezett Edge Computing paradigma használatát az ipari környezetre szánt kiterjesztett valóság alkalmazások köré felépített architektúrákhoz. A paradigma megvalósítására számos megközelítés született úgy mint, Fog Computing, Mobile Edge Computing vagy Cloudlets. Mindegyik megközelítésnek ugyanaz a célkitűzése: mozgassuk a felhőben levő (a kliensek számára szükséges) feldolgozást a kliensekhez a lehető legközelebb, ami még lehetséges.

A Fog Computing alapú megoldások esetén, a számítási egységeket szétszórják a gyárban vagy az ipari környezetben, hogy a legtöbb helyen könnyen elérhetőek legyenek. Ezek az egységek feldolgozzák és tárolják az információkat, amelyek a kliensektől érkeznek, ezután (valamikor) továbbítják a felhőbe. Ellenkező irányban, a számítási egységek elkérik a releváns adatokat a felhőtől és ideiglenesen tárolják lokálisan, amelyet majd továbbküldik (akár az adatot feldolgozva) a kliensnek (például HoloLensnek). A Mobile Edge Computing hasonló a Fog Computinghoz, csak itt mobilhálózaton keresztül valósul meg a kommunikáció.

A hagyományos architektúra módosításával magas szinten kaptunk egy olyan architektúrát, amely a kliensekhez való közelségnek és a nagy számítási teljesítménynek köszönhetően tehermentesíthetjük a megjelenítő eszközöket és lecsökkenthetjük a dinamikus adatelérés késleltetését, amely architektúra alkalmassá teszi a hatékony, felhasználóbarát kiterjesztett valóság alkalmazások megvalósítását. [7]

# 3 Általunk javasolt megoldás

Az eddigiekben áttekintettük az Ipar 4.0 által vízionált jövő gyárát, annak céljaival, előnyeivel és a megvalósításhoz elengedhetetlen technológiákkal. Majd részletesebben megvizsgáltuk, miként fogja ez a transzformáció megváltoztatni a gyárak hierarchiáját, a virtuális és kiterjesztett valóság alkalmazási lehetőségeit és főbb nehézségeit, illetve a különböző ipari környezetben használatos kommunikációs architektúrákat.

Ebben a fejezetben ezen ismeretek felhasználásával, kidolgozunk egy megoldási javaslatot egy kontextusfüggő döntéstámogató rendszerre, mely a felhasználó számára a szükséges információkat, a megfelelő helyen, a megfelelő időben jeleníti meg.

Célunk egy olyan keretrendszer kidolgozása, mely képes a felhasználó látómezőjében található rendszer adatait jól strukturált formában megjeleníteni számára oly módon, hogy közben a kezeit szabadon tudja használni. Ezen felül az adatok értékeit elemezve a felhasználó kapjon releváns javaslatokat a rendszeren végezhető beavatkozási lehetőségekről, azok paramétereiről, ezzel is elősegítve a gyors döntések meghozatalát. Továbbá támogatni szeretnénk interaktív útmutatók létrehozását és lejátszását, aminek segítségével a hagyományos papír alapú leírások és az általuk okozott hosszú javítási idők és félreértések száma jelentősen csökkenne.

A megoldásunkhoz elengedhetetlennek tartjuk, egy olyan komponens létrehozását, amely képes a különböző kommunikációs protokollokat használó, a gyártás során előforduló eszközök és rendszerek által szolgáltatott adatok gyűjtésére és azok egységes tárolására. Ez nagyban megkönnyíti a különböző forrásokból származó adatokon történő utófeldolgozást, analitikát. Ezen felül, az egységes tárolásnak köszönhetően egyszerű megoldást tudunk szolgáltatni a különböző kliensek adatokkal történő kiszolgálására és megteremtjük annak a lehetőségét, hogy több, teljesen különböző rendszer releváns adatait tudjuk kombinálni egy „rendszerré”, ami bizonyos esetekben nagyban megkönnyítheti a döntéshozást.

Példaként, egy éppen készülő terméknél, nem csak a gyártáshoz kapcsolódó adatokról, hanem a logisztikai rendszert integrálva, a szükséges alapanyagok raktárkészletéről, egy prediktív analízis modult felhasználva pedig a jövőben szükségesnek ítélt alapanyag mennyiségről is képet kapunk. Véleményünk szerint ez

nagyban meggyorsítja a döntéshozást, hiszen ahelyett, hogy több rendszer adatait kéne manuális módon összevetni, a megfelelő kliens használatával rögtön, akár a futószalag mellett is rendelkezésre állnak a szükséges információk és a döntés ott helyben meghozható. Ezáltal illeszkedve a 2.1 fejezetben tárgyalt automatizációs hierarchia dekompozíciójához elosztott szolgáltatásokká.


**14. ábra Ipar 4.0 implementálásának kihívásai[12]**

Egy ilyen komponens szükségességét alátámasztja a BITKOM, VDMA és ZVEI által végzett felmérés, amely az Ipar 4.0 vízió különböző aspektusairól tett fel kérdéseket. Ezek közül az egyik az Ipar 4.0 implementációjához kapcsolódó legnagyobb kihívásokról szólt. Az erre érkezett válaszok alapján a legnagyobb problémát a sztenderdek hiánya okozza, ezt követve a folyamatok és munka szervezése, majd a szükséges termékek elérhetősége (14. ábra).

A fentiek miatt, a megvalósítás során törekedni fogunk a megoldásunk kommunikációs protokolloktól való függetlenítésére, illetve ahol lehetőség adódik az adott területen legígéretesebb protokoll használatára. Például az iparban használt

[12] Forrás: [1]

eszközök esetén az egyre népszerűbb OPC UA kommunikációt használjuk, amelyet felmérések szerint már több mint 47 millió automatizációs eszköz támogat világszerte [8].

Fő kihívásnak a kommunikációs protokoll független megvalósítást és a rendszerek egységes kezelését tekintjük, különös tekintettel a beavatkozásra.

A rendszerünk másik elengedhetetlen része a kliens szoftver elkészítése. Ehhez először ki kellett választani egy Augmented Reality eszközt, ami az említettek szerint a Microsoft HoloLens szemüvege, mivel ennek köszönhetően a kezek szabad használata megoldottnak tekinthető. Ennek a kliensnek támogatnia kell a már említett információk, interaktív útmutatók megjelenítését, és meg kell valósítania a szerverrel történő kommunikációt is.

A megvalósításhoz vezető út főbb kihívásainak a felhasználóbarát felület kialakítását, a rendszerek felismerését, az interaktív útmutatók 3D elemeinek térbeli pozícionálását, illetve az ehhez szükséges 3D objektumok előállítását tekintjük.

Ezen felül kihívást jelentő feladat egy olyan szerkesztő elkészítése, melynek segítségével képesek leszünk az imént említett útmutatókat elkészíteni. Mindezt oly módon, hogy a felhasználó részéről ne igényeljünk különösebb előtudást, mégis kényelmes legyen az útmutatók készítése és a szerkesztő elérése is a lehető legegyszerűbb legyen.

A dolgozat keretein belül elkészített prototípusunk tartalmazni fogja a szerver komponenst, azonban ahelyett, hogy ajánlatokat tennénk a megjelenített információra és végezhető akciókra, megjelenítjük a rendszerhez felvett összes elemet, mivel adat hiányában nem tudunk megfelelő analízis motort készíteni. Véleményünk szerint ennek ellenére a dolgozatban felvetődő kérdésekre megfelelő válaszokkal tudunk szolgálni.

A kliens oldali prototípus alkalmazásunk az imént leírtaknak megfelelően, teljes egészében el fog készülni. Az útmutató szerkesztőnek pedig egy olyan változata készül el webes technológiák használatával, amely kétdimenziós objektumok elhelyezésére képes egy megadott térbeli markerhez viszonyítva. Azonban a jövőben háromdimenziós, a HoloLens környezetfeltérképezését használó verziót szeretnénk megvalósítani, amivel az Augmented Reality technológia által nyújtott lehetőségeket teljes mértékben ki tudjuk aknázni.

# 4 Webszerver

## 4.1 Célkitűzés

A rendszer működéséhez fontos, hogy a klienshez a megfelelő, jólformált adatok jussanak el. Ennek a kritériumnak a teljesítése a szerver feladata. Egy ipari monitorozó rendszerből, ami a felhasználó számára érdekes adatokat szolgáltat, adott periódusonként lekérdezi ezen adatokat, majd előállít egy előre meghatározott formátumot, amit egy REST interfészen keresztül hozzáférhetővé tesz a csatlakozó kliensek számára. A szerver további feladata, hogy elérhetővé tegye a csatlakoztatott rendszeren végezhető műveleteket úgy, hogy az megfeleljen a Richardson Maturity Model hármas szintjének. A műveletvégzésre irányuló kéréseket pedig a megfelelő rendszer megfelelő végpontjára irányítsa a felhasználó által megadott paraméterekkel.

## 4.2 Felhasznált technológia

A webszervert ASP.NET Core[13] technológiával valósítottuk meg, ami egy nyílt forráskódú platformfüggetlen keretrendszer, mely modern felhő alapú webszolgáltatások, mobil és IoT háttérrendszerek készítésére alkalmas.

A választás oka a platformfüggetlenség és egyszerűség mellett az volt, hogy ugyanabban a környezetben (C#, Visual Studio) tudtuk fejleszteni a szervert, amiben a klienst is fejlesztettük.

## 4.3 Kommunikációs protokoll

A kommunikációhoz a JavaScript Object Notation (JSON) formátumot választottuk, mivel viszonylag tömör, könnyen olvasható emberi szemmel és programozottan is könnyen kezelhető. A tömörség azért kedvező tulajdonság, mert így lassabb hálózati sebesség mellett is hatékony tud lenni a kommunikáció.

A szerver által előállított JSON üzenetet egy megadott előre definiált formátumba állítjuk elő, mivel így a kliensek számára nagyban egyszerűsödik a rendszer

[13] https://www.asp.net/core/overview/aspnet-vnext

struktúrájának és adatainak a megjelenítése, a felhasználói felület dinamikus felépítése. A következő pontban ennek a JSON üzenetnek a felépítését ismertetjük.

### 4.3.1 JSON üzenet felépítése

Különböző felhasználási eseteket megvizsgálva, szemantikai szempontok alapján, az üzenet öt fő logikai csoportra osztottuk, amelyeket egyesével ismertetünk a következőkben a saját rendszerünkből vett példák segítségével.

#### 4.3.1.1 Information

A rendszer alapvető információit tartalmazza, mint például név, leírás, az esetleges kézikönyvek és az erőforrás azonosító.

```
"information":{
   "name":null,
   "description":"Coffee Machine",
   "location":"office",
   "guides":[
      {
         "name":"How to Use a Coffee Maker",
         "type":"text/html",
         "uri":"http://www.wikihow.com/Use-a-Coffee-Maker"
      }
   ],
   "uri":"https://resources.com/coffemachine"
}
```

#### 4.3.1.2 Components

A rendszer komponenseit tartalmazza, melyekről megtudhatjuk a nevüket, leírásukat, típusukat és ezen felül tartalmazhat a komponenshez tartozó kézikönyveket és képeket. A komponenseknél típus szerint megkülönböztetjük a szenzorokat és az összetett komponenseket, amelyek több szenzorból épülhetnek fel. Egy összetett komponenst ugyanúgy kezelhetünk rendszerként, mint a jelenlegi rendszer alrendszere.

Ha komponens szenzor típusú, akkor tartalmaz egy adat mezőt is, ahol a szenzor adatait tároljuk el. Egy adatnak van neve, mértékegysége és értéke.

```
"components":[
   {
      "name":"watersensor",
      "description":"Checks water level and temperature.",
      "type":"Component",
      "guides":[ … ],
      "image":{ … }
   },
   {
      "data":[
         {
```

```
            "name":"Empty",
            "unit":"logical",
            "value":false
         },{ … },{ … }],
      "name":"coffeebeansensor",
      "description":"Checks coffee bean level.",
      "type":"Sensor",
      "guides":[ … ],
      "image":{ … }
   }
]
```

#### 4.3.1.3 Actions

Az action-ök megadják, hogy milyen műveleteket lehet végezni a rendszeren. Egy action objektum tartalmazza a nevét, leírását és a híváshoz szükséges információkat, mint a használandó HTTP Verb, az URI és a hívás paraméterei. Egy hívási paraméternek van neve, típusa és lehetnek elfogadási tartományai a megadható értékekre.

```
"actions":[
   { … },
   {
      "name":"brewcoffee",
      "description":"Brew Coffee",
      "method":{
         "method":"POST"
      },
      "uri":"https://coffe.com/brewcoffee",
      "parameters":[
         {
            "name":"type",
            "type":"enum",
            "allowedValues":[
               "Espresso",
               "Cappuccino",
               "Americano",
               "Caffe Latte"
            ]
         }
      ]
   }
]
```

#### 4.3.1.4 Connections

A rendszerhez csatlakozó és elérhető erőforrásokat tartalmazza, azok erőforrás címével.

```
"connections":[
   {
      "name":"printer",
      "uri":"https://resources.com/printer"
   }
]
```

#### 4.3.1.5 Person

A rendszerrel valamilyen kapcsolatban lévő csoportokat tartalmazza. Egy csoporthoz tartozik egy név és a tagok listája. Egy taghoz tartozik a neve, titulusa, elérhetőségei, képe.


```
"person":[
   {
      "name":"Producer",
      "members":[
         { … },
         {
            "name":"Michael L. Evans",
            "rank":"Administrator",
            "phone":"727-892-7656",
            "email":"mic_eva@coffezone-inc.info",
            "image":{ … }
         }
      ]
   }
]
```


#### 4.3.1.6 Tutorial

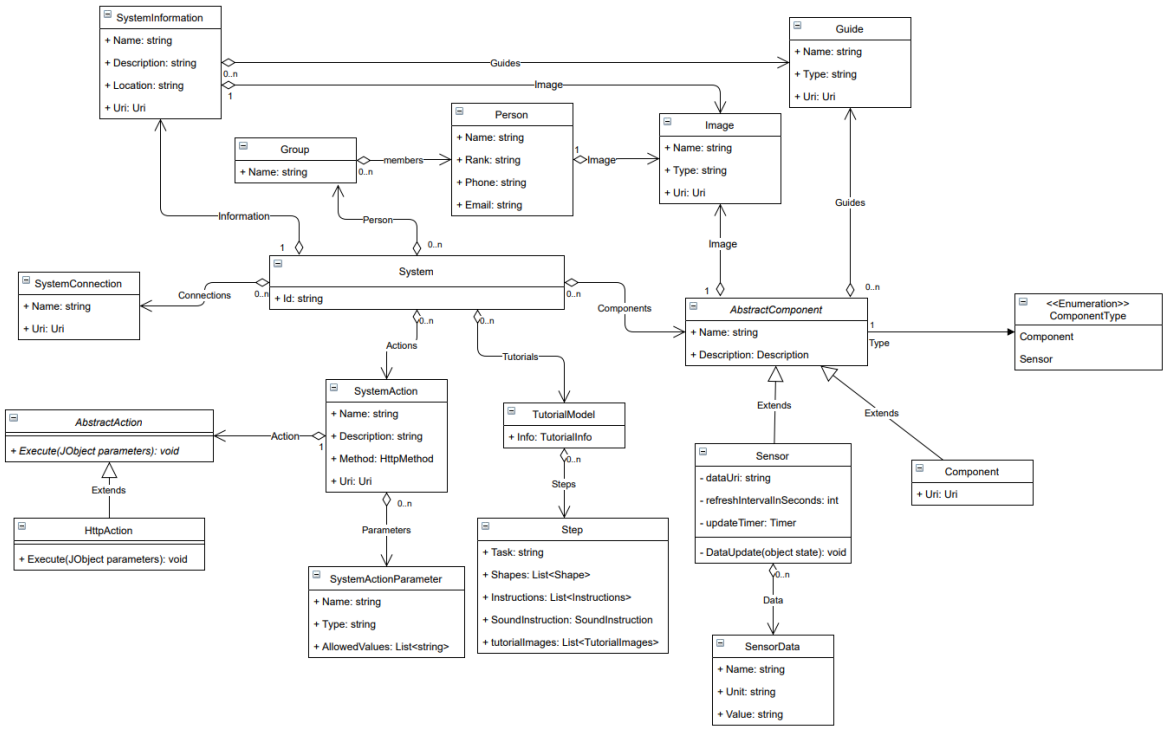
A rendszerhez kapcsolódó interaktív útmutatók gyűjteménye, melyek lehetővé teszik a felhasználó számára különböző feladatok vezetett elvégzését.

Minden útmutatóhoz tartozik név, leírás, a végrehajtani szükséges lépések száma és a végrehajtáshoz szükséges becsült idő. Majd ezután következnek az egyes lépések, amelyek tartalmazzák a megjelenítendő objektumokat (téglalap, kör, kép), a szöveges instrukciókat, illetve egy URI-t a hangfájlra, ami az adott lépéshez tartozó beszédhangos útmutató.


```
"tutorials": [
{
    "info": {
    "name": "Example Tutorial",
    "description": null,
    "duration": "12 minutes",
    "step_num": 3
    },
    "steps": [
        {
            "task": "prepare",
            "shapes": [ { ... } ],
            "instructions": [],
            "soundInstruction": { ... },
            "tutorialImages": []
        },
        { ... }
    ]
}]
```

## 4.4 A kiajánlható rendszer modellje

A szerver megvalósítása során a cél az volt, hogy egy olyan struktúrát hozzunk létre, amivel viszonylag egyszerű egy általános rendszert leírni, arról adatot szolgáltatni és reagálni különböző beavatkozási kérésekre.




**15. ábra A modell szerkezete**

Ahogy az a modell felépítéséből is látszik (15. ábra), egy rendszer több különböző tulajdonságot tartalmaz. Az általános információ tartalmazza a rendszer nevét, egyszerű leírását, pozícióját és az erőforrás azonosítót, amin keresztül elérhetjük azt, illetve tartozhatnak a rendszerhez különböző útmutatók, kézikönyvek, amik szintén itt kerülnek tárolásra.

Egy irányító rendszer felépítése egész komplex is lehet, mivel több szenzorból, vagy akár szenzorokból felépülő rendszerből áll, ezért erre fel kell készülni már a tervezés során. Mi a composite minta használatát találtuk megfelelőnek a céljainkhoz, ezért ezt implementáltuk a megoldásunkban az `AbstractComponent`, `Sensor` és `Component` osztályok segítségével. A `Sensor` osztály tartalmaz egy `SensorData` tulajdonságlistát, amiben megtalálhatóak az adott eszköz által közölni kívánt értékek. Illetve minden egyes komponensnek tudnia kell magáról, hogy ő szenzor vagy összetett komponens, mivel ez fontos információval szolgál a kliens számára.

Egy irányító rendszer attól válik tényleg azzá, ha lehetőségünk van azon különböző műveleteket végezni. Ezt úgy közelítettük meg, hogy egy rendszer tudja magáról mit is képes csinálni, így erről tárol egy listát `SystemAction` objektumokból. Ezek az objektumok tartalmazzák azt, hogy mi is az adott művelet, hol érhető el és milyen HTTP ige használata szükséges az eléréshez. A `SystemAction` objektumok listát vezetnek `SystemActionParameter` objektumokról, amik azt mondják meg, az adott műveletnek mik a bemenő paraméterei, azoknak mi a típusa és esetlegesen azt, hogy milyen megengedett értékeket vehetnek fel ezek a paraméterek.

Az irányító rendszerek hibamentes működését több személy teszi lehetővé, ezért az adott rendszerrel kapcsolatban lévő személyek listája is hasznos információt jelent. Ennek a megoldására bevezettünk csoportokat, hogy a személyeket szervezni tudjuk. Azért gondoltuk ezt szükségesnek, mivel előfordulhat, hogy egy adott rendszernek a belső személyzeten kívül, vannak külső karbantartói/üzemeltetői is. Emiatt a létrehozott csoportokhoz rendeljük a személyeket, a rendszer pedig a csoportokat tartja nyilván.

Mivel egy rendszer több másik rendszerrel is kapcsolatban lehet, ezért hasznos lehet tudni ezeket az összeköttetéseket, illetve azt, mily módon tudjuk esetleg elérni a kapcsolat másik szereplőjét. Ehhez vezettük be a `SystemConnection` osztályt, ami tárolja a másik fél nevét és a hozzá tartozó erőforrás azonosítót.

## 4.5 Egy rendszer modellezése

Amikor a fent ismertetett modell szerint leírunk egy mögöttes rendszert, akkor definiálnunk kell az alapvető információit, a hozzá tartozó megjeleníteni kívánt adatokat, a beavatkozási lehetőségeket és egyéb a fenti modellben látható tulajdonságokat.

A legfontosabb itt a beavatkozási lehetőségek felvétele a megfelelő paraméter és hálózati végpont információkkal és a megjelenítendő adatok definiálása.

Amikor felvesszük a megjelenítendő adatokat, akkor az adatok kliens oldali megjelenítéséhez szükséges információk mellett meg kell adjuk a szervernek szükségeseket is. Ezek a szervernek szükséges információk a lekérdezendő adat hálózati elérési pontja, illetve az, hogy az adatot milyen gyakorisággal szeretnénk frissíteni.

## 4.6 A kiajánlható rendszer beavatkozási modellje

A beavatkozási modellnél szem előtt tartottuk, hogy a különböző eszközök különböző kommunikációs elvárásokat támaszthassanak a rendszer felé, tehát tudjunk kezelni olyan rendszereket is, amelyek HTTP kommunikációt támogatnak, de olyanokat is amelyek csak TCP vagy Bluetooth segítségével tudnak kommunikálni. Ezt úgy akartuk megtenni, hogy a lehető legkevesebb módosítással lehessen egy új kommunikációs protokollt elérhetővé tenni a lehető legtranszparensebb módon.

Ezt úgy értük el, hogy a szervernél létrehoztuk az `AbstractAction` osztályt a command mintának megfelelően, ami definiálja a kommunikációs interfészt a szerveren belüli használatra. Ha egy új kommunikációs protokollt akarunk felvenni, akkor ebből az osztályból kell származtatnunk azt és felüldefiniálni az `Execute` metódust a kommunikáció megvalósításával.

Ezt azért tartottuk fontosnak, mivel bizonyos beágyazott rendszerek nem feltétlenül adnak lehetőséget HTTP feletti kommunikációra, hanem például csak az alacsonyabb szintű TCP kommunikációt támogatják. Ha mi a szerverre beérkező HTTP kéréseket megfelelő mögöttes rendszer kommunikációs képességeihez igazodva tudjuk továbbítani, akkor a szerver és kliens között a beavatkozásra is tisztán HTTP kommunikáció használható, amivel így egységes, átlátható kommunikáció alakulhat ki.

A beavatkozási modell egyik fő eleme a rendszerhez rendelt kontroller osztály, amelyben tároljuk a rendszerünket és kiajánljuk a kliensek számára a rendszer interfészét, ami a szolgáltatott adatok lekérdezéséből és a beavatkozási végpontból áll.

Egy specifikus rendszer esetén, amikor létrehozzuk az azt leíró osztályt és definiáljuk a `SystemAction` objektumokat meg kell adjuk annak a nevét, leírását, hogy a szerver mely végpontján hívható ez a beavatkozás. Ezen felül természetesen a szükséges paramétereket is itt tudjuk definiálni, illetve itt tudjuk megadni azt, hogy milyen típusú esemény ez a beavatkozás. Ez alatt azt kell érteni, hogy meghatározható az action kommunikációja a tényleges szerveren modellezett rendszer felé, ahogy azt fentebb is tárgyaltuk az imént.

Amikor a rendszer kontrolleréhez beérkezik egy kérés, hogy a mögöttes rendszer végrehajtson egy műveletet, akkor ki kell keressük a rendszerhez tartozó beavatkozások közül a megfelelőt, amiben segít az, hogy a kérés törzsében a paraméterek mellett megtalálható a beavatkozó művelet neve is. Ennek a tudtában végig haladunk a

modellezett rendszer action objektumain és megkeressük a név szerint egyezőt. Ezután már a megkapott rendszerhez tartozó beavatkozási lehetőség action objektumát használhatjuk, amivel így megfelelő kommunikációs protokollt fogunk használni a mögöttes rendszer felé. Ezt az teszi lehetővé, hogy a kommunikáció típusa a rendszer definiálásakor rögzítésre került a hívandó végponttal együtt, a hívás interfésze pedig minden kommunikációtípusra azonos. Tehát amikor megvan a kommunikációhoz szükséges action objektum, akkor csak a kapott beavatkozási kérés törzsében található paraméter objektumot kell átadni a híváshoz, ami így eljut a modellezett rendszerhez.

## 4.7 A rendszerekhez rendelt interaktív útmutatók kezelése

Mivel a fejlesztés során megjelent azon igényt szerettük volna kielégíteni, hogy az alkalmazás felhasználóit támogassuk különböző feladatok elvégzésében vezetett módon, ezért szerver oldalon meg kellett valósítanunk az útmutatók megfelelő kezelését.

A felmerült problémának a megoldását úgy terveztük meg, hogy egy webes felületen a felhasználó létre tudjon hozni különböző útmutatókat, amiket majd eltárolunk a szerveren a megfelelő rendszerekhez rendelve. Természetesen szerettük volna megadni a lehetőséget a kész/félkész leírások szerkesztésére is, ami azt tekintve, hogy az egyes megjeleníteni kívánt elemek mérete, távolsága egy webes felületen pixelben, a HoloLens kliensen pedig méterben értendő új kihívásokhoz vezetett.

Az elképzelt működéshez a szerveren létrehoztunk egy új kontrollert (`TutorialController`), ami az útmutatókért felel, illetve a szükséges modell osztályokat, hogy kényelmesen hozzáférhessünk az elkészült útmutatóhoz. Támogatnunk kellett a mentést/visszatöltést a webes felületről, amit a megfelelő meglehetősen primitív végpontok implementálása könnyek teljesített. Az egyedüli bonyodalmat az imént tárgyalt probléma szolgáltatta, mivel az itt kapott/küldött információk a szerkesztőnek szólnak (pixel a mértékegység) meg kellett oldanunk, hogy az útmutatók HoloLens-en is megfelelő formában, megfelelő méretekkel, arányokkal jelenjenek meg. Erre azt a megoldást választottuk, hogy a webes szerkesztőben történt mentés esetén átkonvertáljuk a létrehozott útmutatót a HoloLens kliens számára és ezt tároljuk el a rendszer adatai között, amit a kliens kérésére küldünk. A következő részben (4.8) tárgyaljuk részletesen az imént említett folyamatot.

Mivel kívánatos volt az interaktív útmutató egyes lépéseihez rendelhető hangfájlok és a képek megjelenítésének lehetősége, ezért a szerveren létrehoztunk egy

újabb kontrollert (`FileController`), ami lehetőséget biztosít a webes szerkesztő számára az ebbe a két kategóriába tartozó erőforrásfájlok tárolására. Amikor pedig a szerkesztőben egy fájl feltöltésre kerül, akkor válaszként adódik ezen fájl relatív URI-ja, amivel kényelmesen lehet hivatkozni az erőforrásra az útmutató leírójában.

## 4.8 Az útmutatóhoz tartozó adatok konverziója

A szerkesztő kimenetének a kliens számára megfelelő formátumba történő konvertálását a szerveren végezzük el. Mivel ez a formátum adott, ezért részletesen csak az elhelyezési és méretezési koordinátákra térnénk ki.

A szerkesztőben a teljes szerkesztési tér bal felső sarka adja a 0,0 pontot, valamint az y lefele növekszik, az x pedig jobbra, a koordináták pedig pixelben adottak. A kliens prototípusnál a 0,0 pont a pozicionáló marker középpontja, az y felfele növekszik, az x pedig jobbra, a koordináták méterben értendők, a marker mérete pedig a valóságos méretével egyezik. A szerkesztőből érkező adatok a pixelinformációkon túl tartalmazzák azt is, hogy mekkora valóságban a pozicionáló marker. Az alakzatok a körön kívül ugyanúgy adódnak meg mindkét helyen referencia pont tekintetében, ez pedig a bal felső sarok. Kör esetén a kliensnél a középpont a referenciapont, a szerkesztőben pedig a bal felső sarok (értelemszerű az átalakítás).

Mivel a Unity [14]fejlesztőkörnyezetben a pozicionáló marker esetén a hosszabbik oldal van tekintve 1 méternek, ezért mi referenciaként a hosszabbik oldalt választottuk. Egy x,y koordináta meghatározása során először kiszámítjuk, hogy hány pixel széles a marker hosszabbik oldala, valamint meghatározzuk, hogy hol helyezkedik el pixel tekintetében a középpontja. Ezután kliensoldali x meghatározása esetén egyszerűen az adott objektum x koordinátájából kivonjuk a pixelben megadott marker középpontját, majd ezt elosztjuk a hosszabbik oldalának pixelméretével (ekkor megkaptuk, hogy hány markernyivel kell eltolni az objektumot a középponttól), végül megszorozzuk a hosszabbik oldal valóságos méretével. Így megkaptuk, hogy hány méterrel kell eltolni a középponthoz képest az objektumot x irányban. Y esetén ugyanezt tesszük csak a kivonás fordított, hiszen ellenkező irányban növekszik. Ezután az objektumok méretének meghatározása értelemszerű a korábbiakra építve.

[14] https://unity3d.com/

Szöveg esetén a méret meghatározása trükkös, a szerkesztőből jött információk elárulják, hogy mekkora a font mérete, illetve, hogy egy pontnyi fontméret milyen magas pixelszámban. A két szám összeszorzásával megkapjuk a szöveg magasságát pixelben. A Unity fejlesztőkörnyezet azonban nem méterben várja a font méretet, hanem pontban, mégpedig úgy, hogy 10 pont az 1 cm. Ezért, ha elvégezzük a szerkesztőből kapott szövegmagasságra a korábban leírt konvertálást, akkor ezt még be kell szorozzuk 100-al, hogy cm-t kapjunk, majd további 10-el, hogy pontot kapjunk, így lesz arányos a szerkesztőbeli méret a kliensen láthatóval.

# 5 Útmutató szerkesztő

## 5.1 Célkitűzés

Az előző fejezetben említett igény teljesítésének egyik legfőbb komponense e szerkesztő, mivel ez teszi lehetővé a felhasználók számára a véleményük szerint hasznos interaktív útmutatók létrehozását. Mivel az igényeinknek megfelelő, vagy akár hasonló feladatot ellátó szerkesztőt nem találtunk, ezért egy saját prototípus készítése mellett döntöttünk.

Az útmutató szerkesztőjének a prototípusunk kifejlesztésénél a legszükségesebb eszközöket akartuk megadni. Fontosnak tartottuk, hogy lehetőséget biztosítsunk bizonyos dolgok bejelölésére, amivel egy útmutató végrehajtójának interakcióba kell lépnie valamilyen formában, pl. ránéz, megnyomja, lehúzza stb. Ezen felül még jogos elvárás a szöveges útmutatók megjelenítése, amik elhelyezhetők tetszőleges helyen és a képeké, amik lehetőséget adnak arra, hogy bizonyos apró dolgokat jobban specifikáljunk a segítségükkel, pl. egy kis gomb megnyomása a kapcsolótáblán, amit nem feltétlenül egyszerű/lehetséges bejelölni pontosan HoloLens-en. Ezen felül támogatjuk még egy beszédhangos instrukció megadását minden egyes lépéshez, amely a kliensen az adott lépésre érkezéskor automatikusan lejátszásra kerül.

## 5.2 Megvalósítás

Mivel azt szerettük volna elérni, hogy a felhasználók számára a lehető legkönnyebb/kényelmesebb legyen a szerkesztő használata, ezért egy webes alkalmazás elkészítése mellett döntöttünk.

Az alkalmazást az Angular [15]keretrendszer használatával készítettük el, a `canvas` kezelésének megkönnyítésére pedig a Fabric.js[16] könyvtárat használtuk.

A szerkesztővel szemben támasztott követelmények ezen szakaszban már meghatározásra kerültek, melyek a következők voltak:

[15] https://angular.io/

[16] http://fabricjs.com/

- Háttérkép megadása, ami segíti a felhasználót a szerkesztés során.
- A rendszer (részének) azonosításához (amelyhez az adott útmutató tartozik) szükséges Vuforia ImageTarget bejelölése és valós méretének megadása méterben.
- Az útmutató nevének és a végrehajtáshoz szükséges becsült időnek meghatározása.
- Útmutató lépések létrehozása, és azok elnevezése.
- Különböző jelzők, képek szövegek elhelyezése.
- Beszédhangos instrukció megadása az egyes lépéseknél.

A `canvas` háttérképének megadásánál a Fabric.js nagy segítséget nyújtott, mivel lehetőséget biztosít arra, hogy egy URL segítségével háttérképet adjunk meg. Így nem volt más dolgunk, mint feltölteni a szerverre a felhasználó által kiválasztott képet és az URL-t megadni, a feltöltésre kapott válasz alapján.

A Fabric.js ezen felül biztosította a különböző alakzatok (téglalap, kör) és a képek egyszerű megjelenítését, így a mi feladatunk a megfelelő eseménykezelés megvalósítása volt. Azt szerettük volna elérni, hogy a megfelelő eszköz kiválasztása után rajzolni tudjunk a `canvas`-re, erre pedig az `onMouseDown` és `onMouseUp` eseményeket tudtuk használni. Elsőként úgy valósítottuk meg a rajzolást, hogy a gombnyomás hatására elindult a rajzolás, ha van kiválasztott rajzeszköz és a gomb felengedésére véglegessé válik az alakzat, ez azonban nem mutatta az éppen készülő alakzatot. Ezért fel kellett használjuk az `onMouseMove` eseményeket is és kezelni attól függően, hogy milyen állapotban van éppen az alkalmazás (folyamatban van-e szerkesztés), temporális alakzatot kirajzolni a kiválasztott eszköznek megfelelően.

Ezt kissé tovább bonyolította az a tény, hogy lépésenként szerettük volna a szerkesztést végezni, tehát meg kellett oldani, hogy az egyes lépések közti váltások alkalmával, az elhagyni kívánt lépés állapota kerüljön elmentésre, a cél lépésé pedig betöltésre. Ennek megvalósításához segítséget nyújtott a Fabric.js, mivel támogatja a `canvas` JSON formába történő sorosítását. Ezt kellett kiegészítenünk a számunkra fontos attribútumokkal, amiket az egyes `fabric.Object` példányokhoz rendeltünk (pl. id, aminek segítségével meg tudtuk valósítani a törlést). Majd az így keletkező állapotleírást

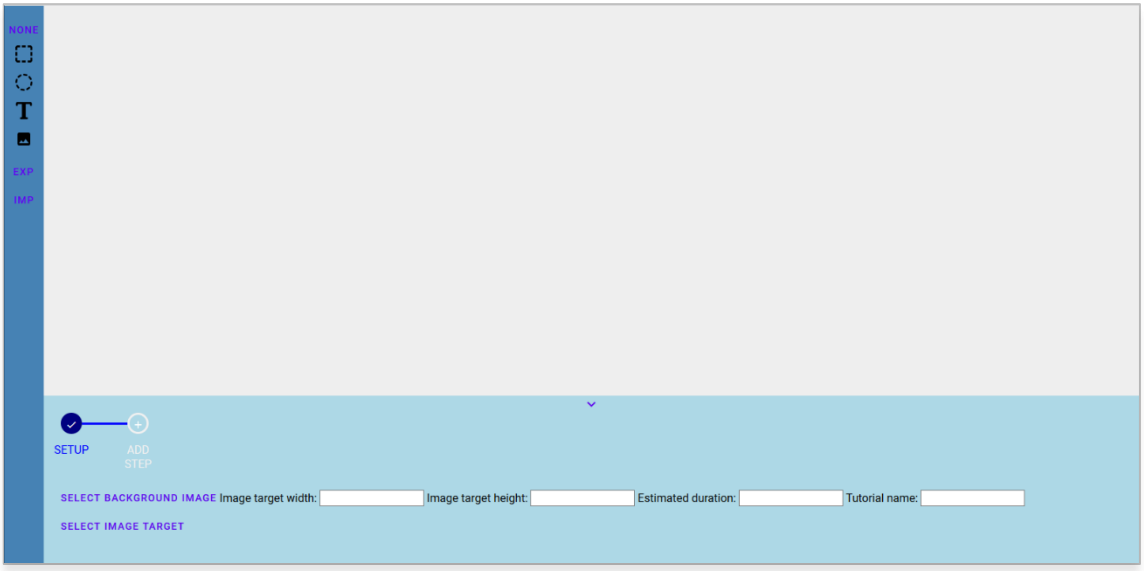
kellett tárolni az egyes lépésekhez, amihez egy listát használtunk, ami tárolja az egyes lépéseket és azok attribútumait.

A követelmények alapján a lépésekhez tartozhatnak hangfájlok is, amit viszonylag egyszerűen meg tudunk valósítani, mivel a felhasználó által kiválasztott fájlt feltöltjük a szerverre, majd a visszakapott URI-t eltároljuk az éppen szerkesztett lépésnél.

Miután az alakzatok rajzolása megfelelően működött, az azonosításra használt marker meghatározása is viszonylag egyszerűen megvalósítható volt. Erre bevezettünk egy konfigurációs fázist, ami minden létrehozott útmutatónál megjelenik és lehetőséget ad a marker bejelölésére, dimenzióinak megadására és az útmutató információinak kitöltésére, amelyek a `tutorial` objektumban tárolásra kerülnek. A marker berajzolására a már elkészített téglalap rajzolást használtuk fel kissé más paraméterezéssel.

## 5.3 Útmutató létrehozása

Amikor létrehozunk egy útmutatót a következő felület fogad minket, ahol a szerkesztés konfigurációs fázisában járunk (16. ábra).



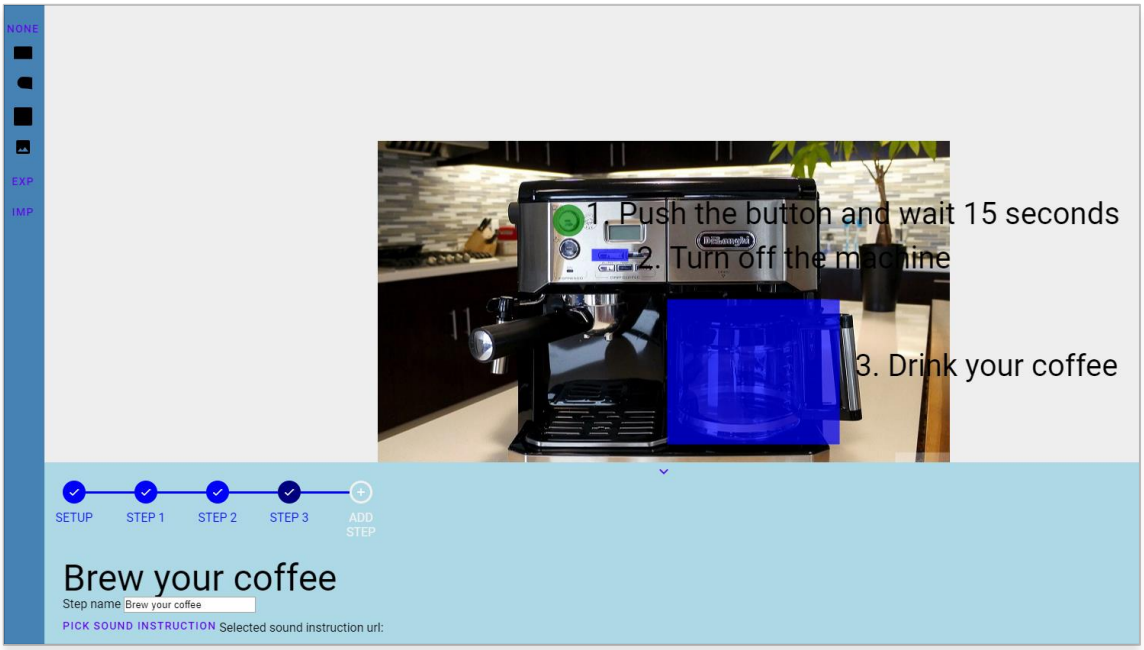

**16. ábra A szerkesztő konfigurációs felülete**

Itt először:

- adjunk meg egy szerkesztést segítő háttérképet,
- ezen jelöljük be a markert,

- majd adjuk meg a valós dimenzióit méterben,
- ezután adjuk meg a végrehajtás idejét és az útmutató nevét.

Ezután létrehozhatunk új lépéseket az Add Step gombra kattintva, aminek hatására a következő képernyő fogad minket, amin az eszközöket felhasználva végezhetjük is a szerkesztést (17. ábra).




**17. ábra Útmutató szerkesztése közbeni állapot**

A szerkesztés során a bal oldalon található eszközsávról választhatjuk az eszközöket és a szerkesztő alján van lehetőségünk megadni a lépés nevét és kiválasztani a hangfájlt, amit le szeretnénk játszani.

A szerkesztés befejeztével az eszközsávról válasszuk az exportot, hogy elmentsük a szerverre a munkánkat. Ha később pedig tovább dolgoznánk akkor az import segítségével betölthetjük azt.

## 5.4 Felhasználói tapasztalok

A 8. fejezetben kifejtett versenyen jelen lévő szakembereknek elnyerte tetszését az általunk kigondolt koncepció. Különösen tetszett nekik, hogy a prototípus használata egyértelmű volt és nem igényelt különösebb előtudást, így rögtön a szerkesztésre koncentrálhattak.

# 6 Kliens

A korábbiak alapján a kiterjesztett valóság szemüvegek ipari felhasználásának előnye egyértelmű, azonban az ilyen platformra tervezett alkalmazások fejlesztése felvet egy újabb problémát: a háromdimenziós felhasználói felületek létrehozását. Az egyéb platformokkal szemben, az ilyen szemüvegek nem alkalmaznak érintőképernyőt, vagy billentyűzetet az alkalmazásokon belüli navigációra, ezért sokkal inkább a gesztúra interakciók megvalósítására kell koncentrálni. Sajnos hiány van az ilyen jellegű felhasználói felületek megvalósítására vonatkozó iránymutatásokból. A kevés elérhető irányelv általában kísérleti folyamatok eredményeként készült, és a fejlesztés során alakult ki, melyek alkalmazási szempontból nincsenek visszaigazolva, megerősítve tudományos cikkek által. Szerencsére, ha a virtuális valóság témában kutatunk, több útmutatást találhatunk a felhasználóbarát felületek létrehozásához. [6]

Kutatásaink haladtával rátaláltunk a Microsoft által létrehozott Mixed Reality Toolkit [9] (korábban HoloToolkit) nevű csomagjára. Ez a csomag tulajdonképpen egy szkriptekből és komponensekből álló kollekció, mely a Microsoft HoloLens és a Windows Mixed Reality headset (virtuális valóság szemüveg) eszközökre való alkalmazások fejlesztését segíti. A csomag, a nevéből is adódóan támogatja (mixelt módon) a virtuális valóság és a kiterjesztett valóság alkalmazások fejlesztését. Tartalmaz számos példát és előkészített elemet a felhasználói felületekhez, melyek magukba foglalják a virtuális valóság fejlesztői világában kialakult tervezési elveket és iránymutatásokat, valamint HoloLens esetén figyelembe veszik a Microsoft által meghatározott iránymutatásokat.

A 18. ábra bemutatja a Mixed Reality Toolkit támogatási területeit. Látható, hogy a bemenet kezelésétől a virtuális objektumokon végezhető interakciókon át egészen a beszédhangfelismerésig meg van valósítva eszközspecifikus módon a felhasználás, és ha igényünk van rá, testreszabható, kiegészíthető a legtöbb funkció. Például egy általunk létrehozott objektumot könnyen tudunk a HoloLens számára értelmezhetővé tenni, például kattinthatóvá, a megfelelő gesztúra hatására.


**18. ábra Mixed Reality Toolkit eszköspecifikus támogatási területei témákra bontva**[17]

Az alkalmazásunk monitorozó és útmutató megjelenítő komponenseinek elkészítéséhez megfelelőnek találtuk és felhasználtuk a Mixed Reality Toolkit csomagot.

## 6.1 Célkitűzés

A kliens feladata, hogy a szerverünkkel kommunikálva, az azonosított rendszer adatait és útmutatóit megjelenítse, illetve lehetővé tegye, hogy a felhasználó különböző beavatkozásokat végezzen azon. A kliens további feladata, hogy a rendszerhez tartozó útmutatók elvégezhetők legyenek. A leírt viselkedést a kliensnek ergonomikus módon kell megvalósítania HoloLens felhasználásával.

## 6.2 Felhasznált technológia

HoloLens-re történő fejlesztés megvalósulhat C++ programnyelven DirectX API segítségével, vagy amit kezdetnek a Microsoft is ajánl, kiindulhatunk egy Unity [10] projektből is, amely lényegesen egyszerűsíti és felgyorsíthatja a fejlesztés kezdetben.

A Unity egy játékmotor, mely segítségével háromdimenziós videojátékokat, illetve egyéb interaktív jellegű háromdimenziós tartalmakat lehet létrehozni. A Microsoft

[17] Forrás: [9]

és a Unity partnerségével elérhetővé vált a HoloLens-re történő kiterjesztett valóság alkalmazások gyors és egyszerű fejlesztése.

Felhasználtuk a korábban már említett Mixed Reality Toolkit Unity csomagját, amely rengeteg eszközspecifikus megvalósítást tartalmaz.

A Unity projekt önmagában nem telepíthető a HoloLens-re, így ha tesztelni szeretnénk az eszközön, először le kell fordítani a projektet egy UWP (Universal Windows Platform) [11] projektté, amelyben ha szeretnénk, (Unity objektumai szempontjából korlátozottan) fejleszthetünk is (használhatjuk az újabb C# képességeit és több fejlesztői csomag elérhető), és ezt követően telepíthetjük az eszközre.

## 6.3 Első kísérleti fejlesztés

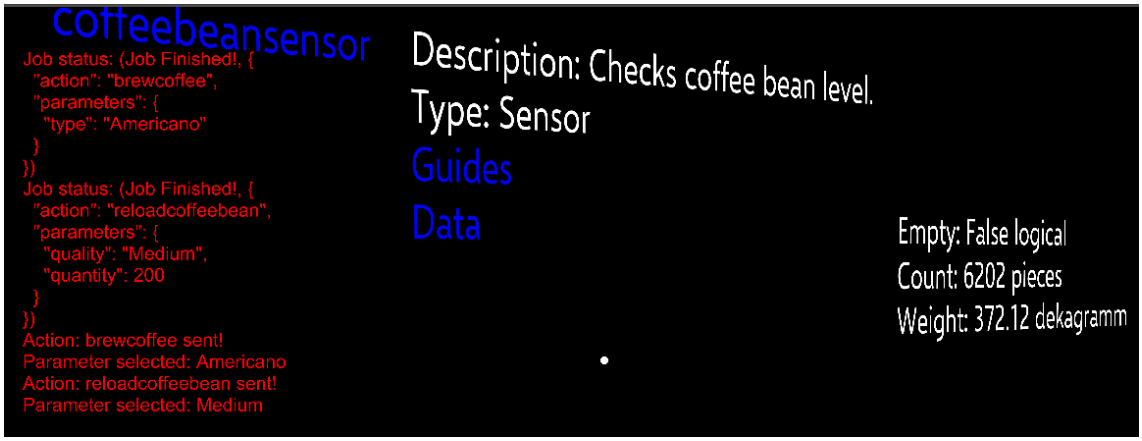
Az első kísérleti fejlesztés egy prototípus prototípusának tekinthető alkalmazás volt, amely célja a korábban bemutatott kiszolgálóval való kommunikáció validálása, valamint a gesztúra interakciók tesztelése és a szöveges megjelenítés megvalósítása.

A projekt alapvetően MVC mintát követve valósult meg, ugyanis a modell és a kontroller hatékonyan újrafelhasználható és jól rá lehet illeszteni a projektünkre. Kliensünk forráskódja két részre válik szét: A Unity-s projekt, amelyben a View részt a C# szkriptek alkotják, valamint a Unity-s projektből lefordított UWP-s projekt, amelyben a Model, a Controller és egyéb logika helyezkedik el.

Az alkalmazás lényegében a kiszolgálótól kapott JSON üzenet alapján szövegeket jelenít meg, és ha az adott szöveghez tartozik egy vagy több alkomponens (például akciók), akkor azt szín szerint megkülönböztetjük és kattinthatóvá tesszük. Vannak speciálisan viselkedő kattintható szövegeink, ilyen például egy kérés elküldése a szerver felé (Send). A 19. ábra bemutatja, hogy a JSON hierarchiája alapján egy balról jobbra felépülő menürendszer jön létre.


**19. ábra Felépített menühierarchia**

A kérés paramétereinek kiválasztását, valamint a kiszolgálóval való kommunikáció eredményét egy folyamatosan látható státusz szövegként jelenítünk meg.




**20. ábra Értesítések megjelenítése**

A projekt célja az elsődleges prototípus kialakításán felül egy további projekt keretének célszerű kialakítása, az ilyen platformra való fejlesztés megismerése, és a szervezési, fejlesztési stratégiák a tapasztalatok alapján való kialakítása volt.

## 6.4 Felhasználói felület felépítése

A felhasználói felület tervezésénél törekedtünk letisztult, egyértelmű és ergonomikus felület megvalósításra. Ötletet merítettünk a Microsoft csapata által kiadott Mixed Reality Design Labs [12] mintáiból, amely a Mixed Reality Toolkit könyvtárat felhasználva jó példákat, ajánlásokat mutat be felhasználói felületekre, dialógus ablakokra, státuszjelzőkre, információs panelekre és sok egyéb gyakran előforduló komponensre, mindezt megvalósítva kiterjesztett valóság környezetben. Mind felhasználói felület tervezés, mind architektúra szempontjából túlnyomórészt az

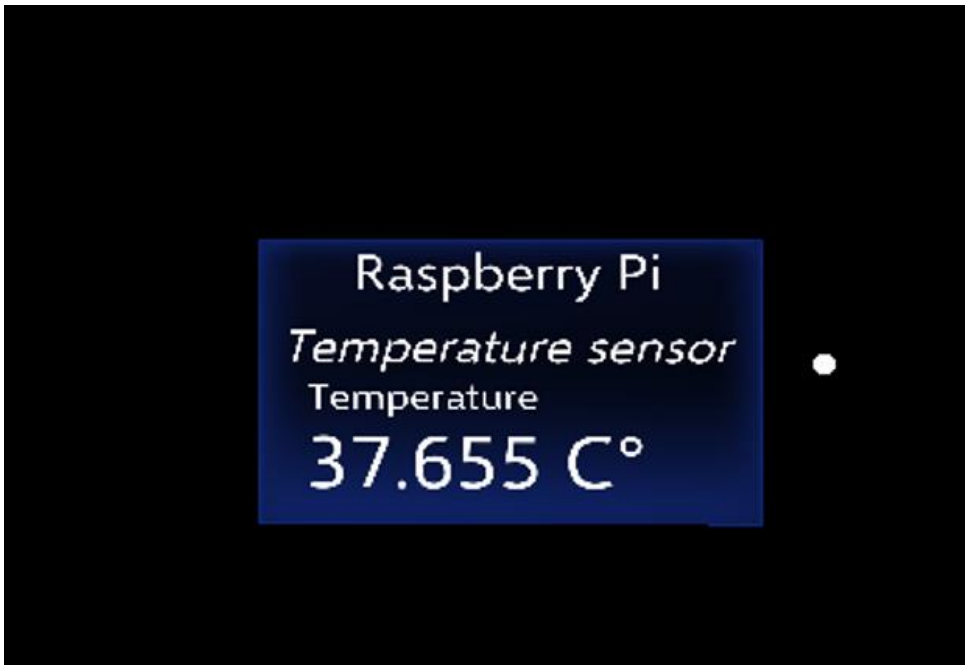
MRDesign Labs Periodic Table of the Elements [12] projektjéből indultunk ki, melyben a periódusos rendszer van modellezve.

A felhasználói felület megtervezésénél figyelembe vettük, hogy a szervertől kapott bizonyos típusú adatok számossága dinamikusan változhat, mint például a szenzorok adatai, az útmutatók vagy a beavatkozási lehetőségek. Ezért olyan megoldást kellett választanunk a felületek szempontjából, amelyek nem jelenítenek meg minden információt, hanem lehetőséget adnak bizonyos interakciók hatására az adatok megjelenítésére, vagy az éppen láthatók elrejtésére.

Felhasználói felületünk legalapvetőbb eleme az úgynevezett csempe. Ez egy kétdimenziós lap, egy áttetsző színre is képes háttérrel, melyen különböző szöveges információk jelenhetnek meg. Később látni fogjuk, hogy a felületünk legtöbb eleme ilyen csempékből épül fel.

### 6.4.1 Rendszercsempe

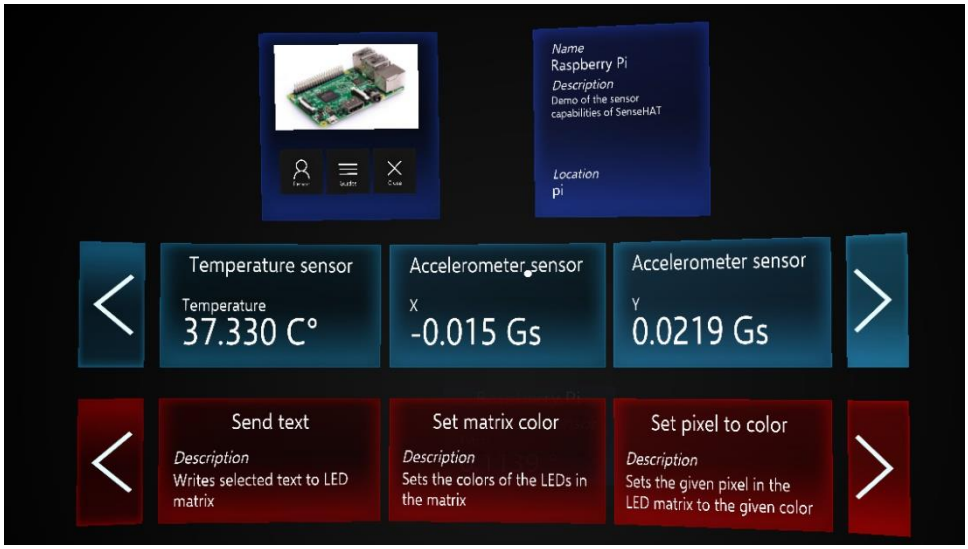
Amikor egy hívás (vagy trigger) hatására (például Vuforia érzékeli/észreveszi a rendszer egy jelölőjét vagy markerjét) lekérjük a rendszertől az információkat, akkor megjelenítünk a rendszerről egy csempét, illetve az összes hozzátartozó (al)komponenséről egyet-egyet. A rendszer csempéjén megjelenítünk egy-egy szenzor egy-egy adatának aktuális értékét vagy néhány szenzor aktuális értékéből származtatott információt úgy, hogy egyszerre csak egy érték látható, de valamennyi (jelenleg kettő) másodpercenként körforgásszerűen váltjuk a megjelenített információt. A megjelenített csempékkel interakcióba léphetünk a kattintás gesztúrával (vagy levegő koppintással, angolul: air tap). Ha egy komponensre kattintunk, akkor alatta egy sorban megjelenik a komponens rendszere, és annak az alkomponensei. Az alkomponensek esetén ugyanúgy egy rendszercsempe jelenik meg, csak egy újabb sorban (rekurzív megoldás). Itt feltételezzük, hogy nincs annyi komponens, és nem olyan mély a komponens fa, hogy ne férne ki a valós látótérben, azonban a későbbiekben erre szofisztikáltabb megoldást szeretnénk. A 21. ábra bemutatja azt a szituációt, amikor komponens nélküli rendszerünk van, és nem végeztük semmilyen interakciót a csempével, csak kapcsolódtunk a rendszerhez (például Vuforia érzékelte a megfelelő markert).




**21. ábra Rendszercsempe**

## 6.4.2 Rendszerpanel

Amennyiben a rendszer csempéjére kattintunk (amelyik a szenzor információkat jeleníti meg), akkor a 22. ábra által ábrázolt felület jelenik meg.




**22. ábra Rendszerpanel**

### 6.4.2.1 Rendszerinformációs panel

A 22. ábra felső részén látható a rendszerinformációs panel, melyet a két fő (sötétkék színű) panel testesít meg. A jobb oldali megjeleníti a rendszer nevét, leírását és helyét. A bal oldali megjelenít egy képet a rendszerről, valamint három menüpontot ad:

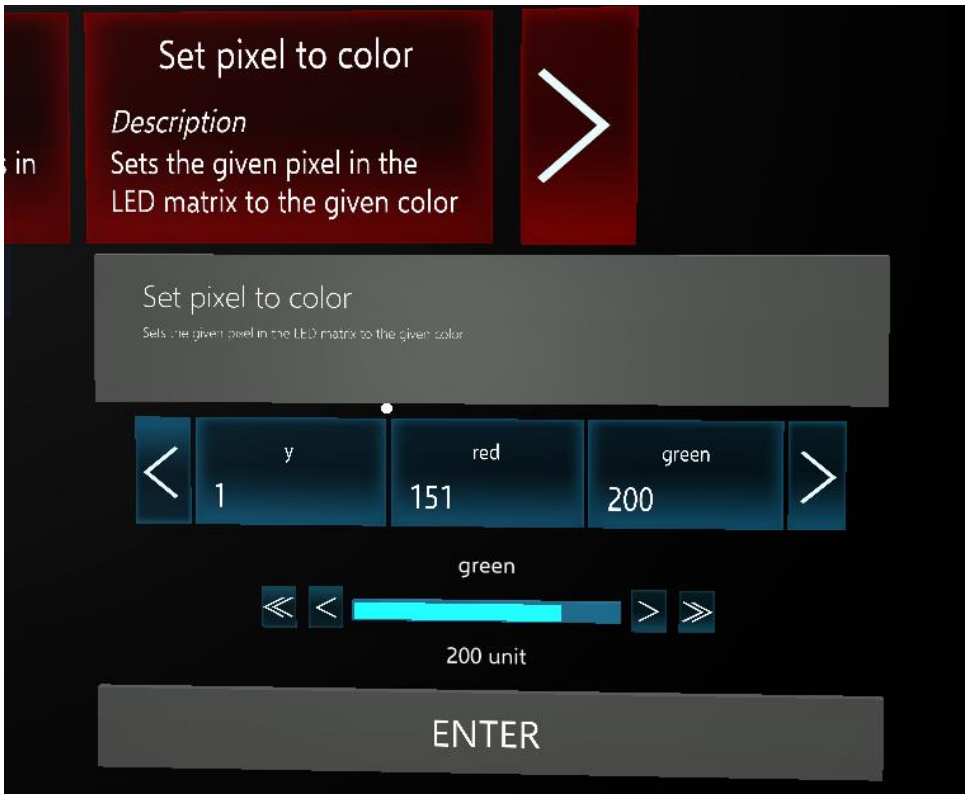
- Person: A rendszerhez bármilyen módon kapcsolódó emberek alapvető adatainak és elérhetőségeinek megjelenítése.
- Guides: Útmutató-kiválasztó megjelenítése, melyről a következő fejezetben tárgyalunk bővebben.
- Close: A rendszer felület bezárása. Ekkor a felület összecsukódik, eltűnik, és csak a rendszercsempe lesz látható.

#### 6.4.2.2 Szenzorpanel

A két panel alatt látható egy lapozható csempézett felület, mely három csempét jelenít meg. Ha három csempénél több megjelenítendő elem van, akkor lehet lapozni. Ez a felület a szenzorpanel, amely panel a rendszer szenzorainak adatait jeleníti meg, úgymint szenzor neve, adat neve, értéke és mértékegysége.

A szenzorpanelen levő csempék nem feltétlen egy szenzornak az információit jelenítik meg. Ahogy korábban is beszéltünk róla, a megjelenített információ lehet egy szenzor adatából vagy adataiból érkező információ, de lehet több szenzor vagy adatot szolgáltató objektum együttes adataiból származtatott információ. Például egy objektum mozgását figyelő rendszer több szenzoradatból dolgozhat, de (a példa kedvéért) mi csak azt szeretnénk megjeleníteni, hogy mozog-e az objektum vagy sem. Továbbiakban a szenzor szó alatt ebben a kontextusban egy olyan objektumra gondoljunk, mint egy vagy több információt szolgáltató objektum.

Ha egy szenzornak több adata van, azt külön csempére tesszük (például a 22. ábra középső részén a gyorsulásérzékelő (accelerometer). A szenzorokat megjelenítő lapozható csempefelület alatt egy ugyanilyen felület jelenik meg azzal a különbséggel, hogy ez a rendszeren végezhető akciókat jeleníti meg. Látható, hogy egy akció csempén megjelenítjük az akció nevét, és leírását. Ezek a csempék kattinthatóak. Ha interakcióba lépünk velük ilyen módon, akkor a 23. ábra által mutatott felület jelenik meg alatta (Set pixel to color-ra kattintva).




**23. ábra Akció paramétereinek beállítása**

A 23. ábra mutatja, hogy megjelenik egy felső szürke panel, amelyen az akció neve ismét rajta van, illetve ha hosszabb a leírás, és nem fér ki az akció csempéjén, akkor itt nagyobb eséllyel kifog férni, hiszen elég nagy felületet választottunk a betű méretéhez viszonyítva. Alatta látható szintén egy lapozható csempefelület, amely csempéken látható az akció egy paraméterének neve, és beállított aktuális értéke. Ezek a csempék kattinthatóak. Ha az egyikre kattintunk, akkor az alatta látható csúszka jelenik meg, mely számérték esetén a látható módon néz ki. A csúszka felett jelezzük, hogy éppen melyik paramétert állítjuk, valamint a telítettségével érzékeltetjük, hogy az alsó-felső határoktól milyen messze vagyunk. A jobbra és balra nyilakkal tudjuk állítani az aktuális értéket, mely érték a csúszka alatt látható. A dupla nyilakkal tudunk konfigurálható (pl.: szervertől érkező ajánlásra alapozva) módon nagyobb értékeket ugrani. Ha például enumerációt vagy igaz/hamis értéket kell állítanunk, nagyobb ugrási lehetőség nincsen. Ha beállítottuk paramétereinket, akkor lehetőségünk van végrehajtani az akciót az enter panelra kattintva, mely elküldi a felparaméterezett kérést a szerver felé. Ha rákattintunk egy másik akció csempére, akkor az előző megnyitott panel az bezárul. Ugyanez történik, ha lapozunk az akciópanelen.

### 6.4.3 Útmutató megjelenítő bemutatása

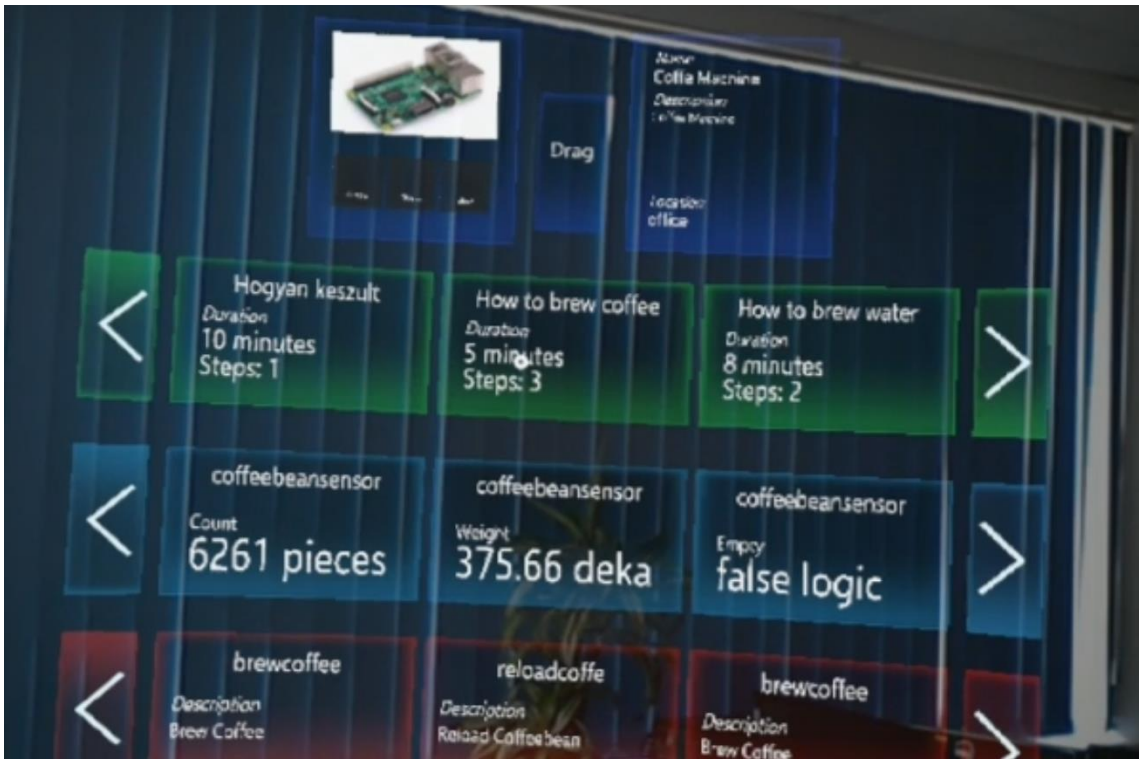
A felhasználói felületünket kiegészítettük az útmutató megjelenítővel, mely egy új modult csatolt hozzá az egy rendszerhez tartozó felhasználói felülethez.

A rendszerpanelen a „Guides” menüpontra kattintva, legördül egy újabb lapozófelület úgy, hogy letolja a szenzorokat és az akciókat megjelenítő lapozó felületet. Ezen a felületen tudjuk kiválasztani, hogy melyik útmutatót szeretnénk elindítani a rendszerhez kapcsolódóan. Ha a „Guides” menüpontra kattintunk az útmutató lapozófelület megjelenített állapotában, akkor az bezárul, és a többi lapozófelület feljebb kúszik, tehát a rendszerpanel kiindulóállapota lesz látható.

A felületen megjelenő csempék hasonlóan kattinthatók, mint a rendszerhez tartozó akciók csempéi. Egy útmutató csempén megjelenítjük az útmutató nevét (vagy címét), körülbelül mennyi idő elvégezni és mennyi lépésből áll. Egy lépésnek tekintjük az útmutató elemek megjelenítéséből már egyértelműen elvégezhető cselekvést (pontosabban az útmutató készítője által már egyértelműnek vélt cselekvést).

#### 6.4.3.1 Drag and drop funkció beillesztése

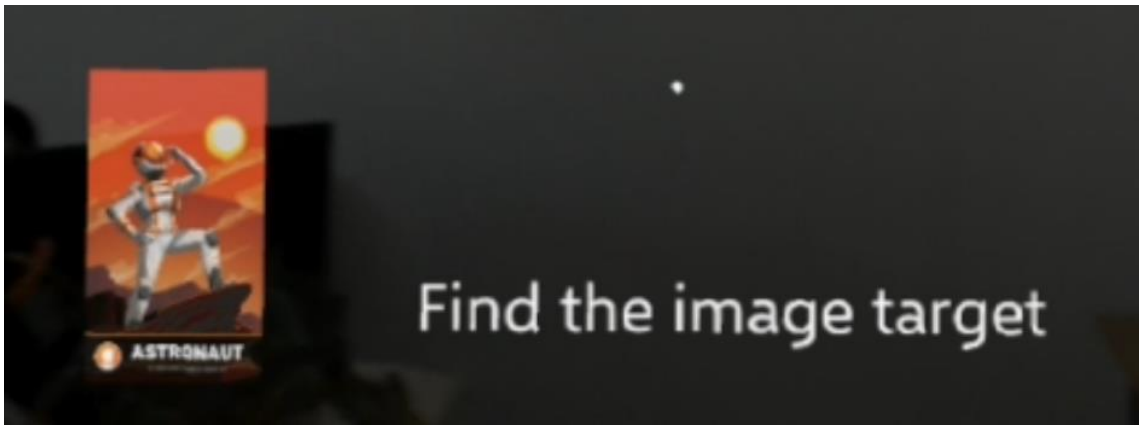
Eddigi megvalósításunk szerint a rendszercsempe (és a rendszerpanel) elhelyezése a rendszer (Vuforia segítségével) valós térbeli elhelyezkedéséhez volt viszonyítva fix, beégetett módon. Tehát fennált a lehetősége, hogy nem kifejezetten kényelmes helyre kerültek ezek a csempék. Azonban felhasználás szempontjából ez nem jelentett végzetes problémát. Az útmutató modul beillesztésekor elengedhetetlenné vált, hogy a rendszerpanel térben mozgatható legyen a felhasználó által, ugyanis elképzelhető volt az az esemény, hogy az útmutató objektumai belelógnak (például az adott gépnek azon a részén kell megnyomni egy gombot) a rendszerpanel csempéibe. A rendszer rendszerpaneljét így kiegészítettük egy újabb csempével, melyen alapértelmezés szerint „Drag” felirat van. Ha rákattintunk, akkor „Drop” felirat lesz látható, és az egész csempe követni fogja a kurzorunkat, valamint a szemünktől való távolságot megtartja. A csempe elhelyezkedését a 24. ábra mutatja.




**24. ábra Áthelyezhető rendszerfelület megnyitott útmutatókkal**

### 6.4.3.2 Útmutató végrehajtása

Ha egy útmutató csempére kattintunk, akkor különböző instrukciók jelennek meg a szemünk előtt. Először az alkalmazás megkér arra, hogy keressük meg az útmutatóhoz tartozó képet (Image Target). Ezt tekinthetjük egy referenciapontnak, amelyhez képest megjelennek az útmutatóhoz tartozó különböző feliratok és objektumok (melyeket az útmutató szerkesztőben specifikáltunk), melyek a lépések elvégzését segítik. Az előbb említett kérelmet az alkalmazás egy szövegként megjeleníti („Find the image target”) a szemünk előtt fix pozícióban. Itt jelenítjük meg későbbiekben az adott lépéshez tartozó instrukciót. A 25. ábra alapján, bal oldalon láthatjuk kép formájában, hogy milyen képet, alakzatot vagy térbeli objektumot keressünk.



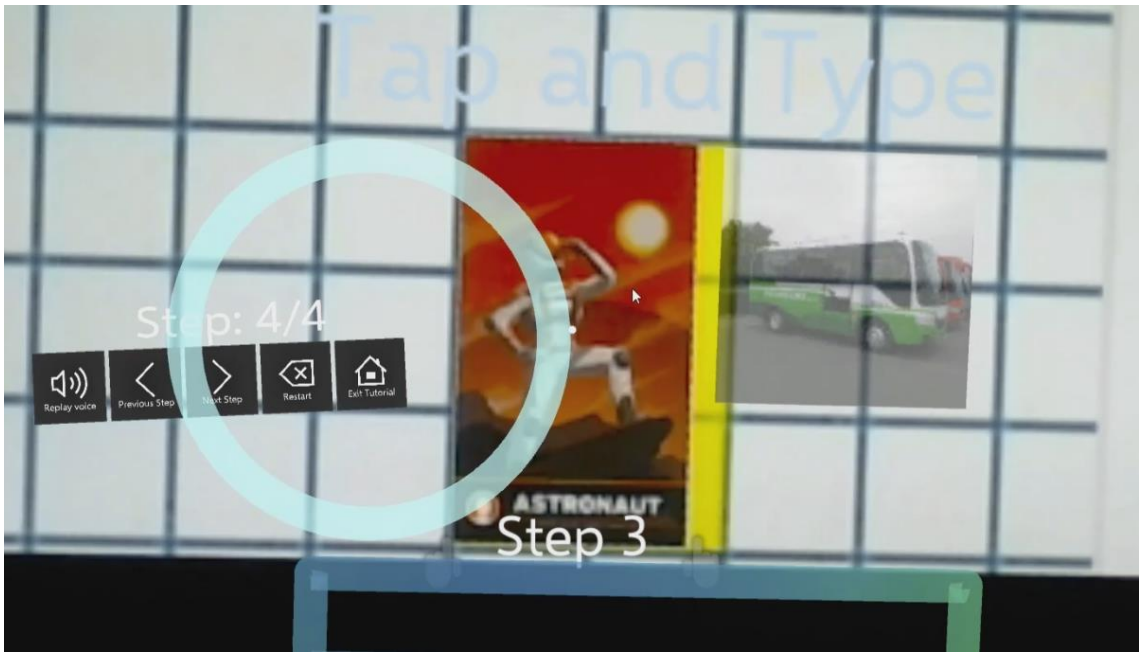

**25. ábra Felismerendő kép (Vuforia által) és az utasítás**

A leírtak mellett az útmutató megnyitásakor megjelenik egy eszköztár (toolbar) is, amellyel tudjuk vezérelni az útmutató végrehajtását. Öt gomb található az eszköztáron:

- Replay voice: lejátssza az adott lépéshez tartozó hangot, ha már aktív az útmutató (akkor aktív, ha megtalálta az keresett objektumot)
- Previous Step: ha van korábbi útmutató lépés akkor visszalép egyet
- Next Step: a következő útmutató lépésre lép, ha van ilyen
- Restart: elölről kezdi az útmutatót (megjeleníti az első lépést)
- Exit Tutorial: kilép az útmutatóból, eltűnik az összes olyan elem, amely az útmutatóval kapcsolatos

Az eszköztár fölött megjelenik egy felirat is, amely jelzi, hogy éppen hányadik lépésnél tartunk.

Miután megtalálta a felhasználó a felismerendő objektumot, megjelennek az objektum körül az útmutató szerkesztőben megadott elemek vele egy síkban. Ezek az elemek a következők lehetnek: szöveg, kör (háromdimenzióban tórusz), téglalap (háromdimenzióban keret), kép. A felhasználó előtt levő felirat az első lépés utasítására módosul, és a későbbiekben itt lesz látható, hogy az adott lépésben mit kell tenni összefoglalás-szerűen (amíg az útmutató meg van nyitva). A térbeli objektumok és feliratok kijelölik a pontos helyet, ahol végezni kell a cselekvést, és meghatározzák, hogy az adott helyen mit kell tenni. Ha elvégeztük az adott lépést, akkor az eszköztár segítségével továbbléphetünk. Az eszköztár nem veszik a szemünk elől, a kurzort követi, ha egy bizonyos távolságon túl van hozzá képest. A 26. ábra szemléltet egy útmutatót, amely a negyedik lépés objektumait jeleníti meg.




**26. ábra Az eszköztár és az útmutató objektumok megjelenése a felimerendő objektum körül**


**27. ábra Fejlesztési példa: kávéfőzés útmutató futtatása HoloLens-en**

# 7 Új rendszer monitorozása és megjelenítése a kliens segítségével

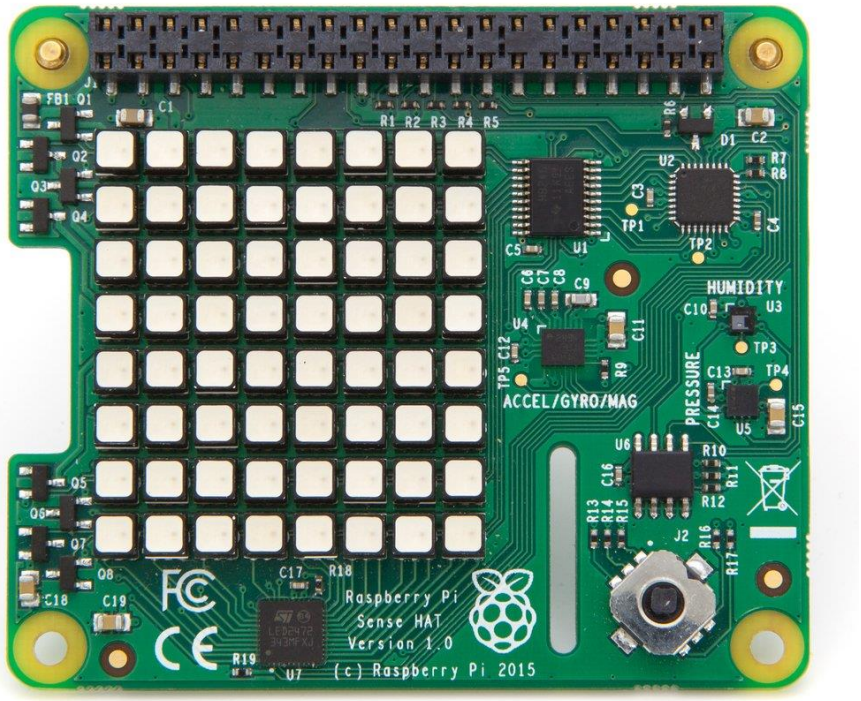
Ahhoz, hogy új rendszereket tudjunk megfigyelni a szerverünk segítségével és ezek adatait a kliensek el tudják kérni, a következőket kell megtennünk:

1. Fel kell vegyünk a szerver projektben egy rendszert a System mappában statikus osztályként, amiben definiáljuk a szükséges tulajdonságokat, melyek a következők:
    a. Szenzorok/Komponensek (`Sensor/Component` osztály) hozzáadása a listához.
    b. Tevékenységek (`Action` osztály) hozzáadása a megfelelő listához. Itt meg kell adjuk a kiajánlott eszközön elérhető végpontokat, a megfelelő kommunikációs protokollal. (Az elérhető kommunikációs lehetőségeket az `Actions` mappában láthatjuk. Az osztályok az `AbstracAction`-ből származnak és annak a megfelelő metódusát definiálják felül. Így van lehetőség új protollok támogatására is.)
    c. A rendszer információinak leírása (`SystemInformation` osztály). Ha azt szeretnénk, hogy a rendszerről megadott kép megjelenjen a szerveren, akkor a `wwwroot` mappába kell másolni azt, hogy statikus tartalomként ki tudja szolgálni a szerver.
    d. A rendszerhez csatlakozó eszközöket a `SystemConnection` osztály segítségével adhatjuk meg.
    e. Megadhatjuk a rendszerhez a hozzá tartozó személyeket csoportosítva a `Group` és `Person` osztályok segítségével.
    f. Ezután létre kell hozni egy `Get` függvényt, aminek a segítségével a fenti definiált rendszert összerakjuk és visszatérünk egy `ControlledSystem` objektummal.
2. Adjuk hozzá a System mappában található `ControlledSystemDB`-ben található dictionary-hez az újonnan felvett rendszert, hogy arra majd később hivatkozni tudjunk.

3. Ha definiáltuk a rendszerünket, akkor létre kell hozni hozzá a megfelelő kontrollert, amire jó minta lehet a `Controllers` osztályban található `RaspberryPiController` osztály. Itt a `ControlledSystemDB`-ben található dictionary-ből kell a megfelelő, előbb létrehozott eszközt elérni.
4. Ezután el kell indítani a szervert.
5. A kliens projektben módosítani kell az elérési címet az újonnan hozzáadott eszköz címére. Ezt a `ViewBuilder` osztály `Start` metódusában tehetjük meg.
6. Most már lefordíthatjuk a klienst és feltölthetjük azt az eszközre.

## 7.1 Raspberry Pi csatolása

Hogy meggyőződjünk arról, hogy a rendszerünk ténylegesen általánosra sikerült, ezért a Raspberry Pi Sense HAT kiegészítőjét (28. ábra) felvettük eszközként az előző pontban ismertetettek szerint.




**28. ábra Raspberry Pi Sense HAT[18]**

Ez a kiegészítő a következő, számunkra fontos szenzorokat tartalmazza:

[18] Forrás: https://www.raspberrypi.org/products/sense-hat/

- Gyorsulásmérő
- Giroszkóp
- Magnetométer
- Nyomásmérő
- Páratartalommérő
- Hőmérsékletmérő
- 8x8 méretű RGB LED tömb

A demonstrációhoz az imént felsorolt szenzorokat használtuk, azonban ehhez ezeket ki kellett ajánlanunk a szerver számára. Ezért kellett készíteni egy szolgáltatást, ami ezen szenzorok olvasását, illetve a LED tömb vezérlését elvégzi és elérhetővé teszi ezt hálózaton keresztül, hogy azt a szerver használni tudja.

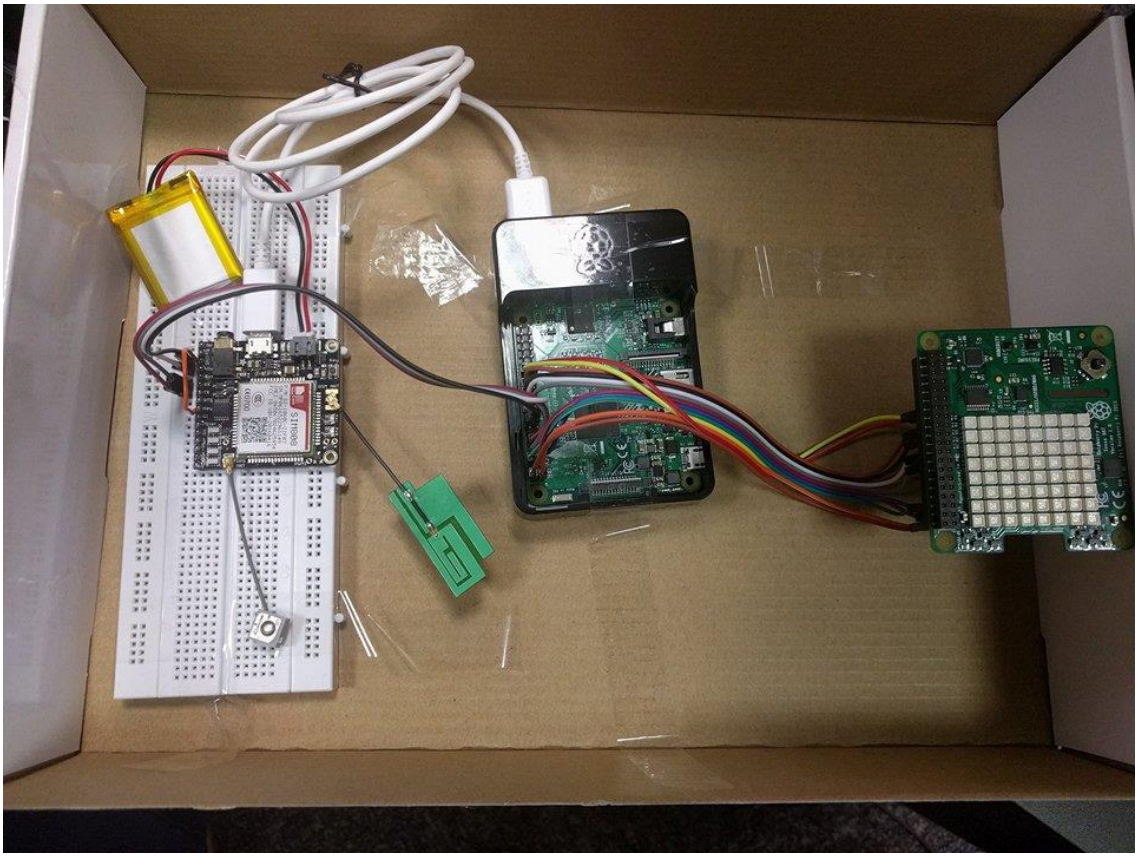
Hogy a szenzorokat tesztelni tudjuk, ezért viszonylag hordozható módon akartuk megvalósítani a fentieket. Ehhez egy GSM modult (29. ábra) használtunk fel, aminek segítségével az eszköz a szerverrel 2G hálózaton tud kommunikálni.


**29. ábra Adafruit FONA 808 GSM modul[19]**

[19] Forrás: https://www.adafruit.com/product/2542

Már csak azt a problémát kellett kiküszöböljük, hogy az eszköz bekapcsolásakor a szolgáltató mindig más IP-t ad az eszköznek, ez pedig a szerver oldalon probléma, mivel nem feltétlenül az eszköz definiálásakor adott címen lesz elérhető a rendszer. Ennek megoldására felállítottunk egy VPN szervert, amivel létrehoztuk a saját privát alhálózatunkat és ezzel meg is oldódott az előbb említett kommunikációs probléma, mivel az eszköz mindig ugyanazt a belső IP-t kapta a hálózatba csatlakozás után.

Annak az elérésére, hogy a rendszer bekapcsolás hatására egy idő után elkezdje szolgáltatni az adatait egy szkriptet készítettünk. Ennek a feladata az, hogy a bekapcsolás után elindítja a GSM modult, így az csatlakozni tud a szolgáltató hálózatára, majd bekapcsolódik a VPN-re és elindítja a szenzorokat kiajánló szolgáltatást.



**30. ábra Elkészült tesztrendszer a szoftverkomponensek ellenőrzésére**

Miután ezeket elvégeztük, kipróbáltuk az elkészült rendszert (30. ábra) és valóban sikerült csak az eszköz definiálásával megjeleníteni a szenzorok értékeit és beavatkozásokat végezni. Így elértük a célunkat és láthattuk, hogy a kialakított architektúra valóban általánosra sikerült.

# 8 Integrálási tapasztalok gyártósori beszállítótól

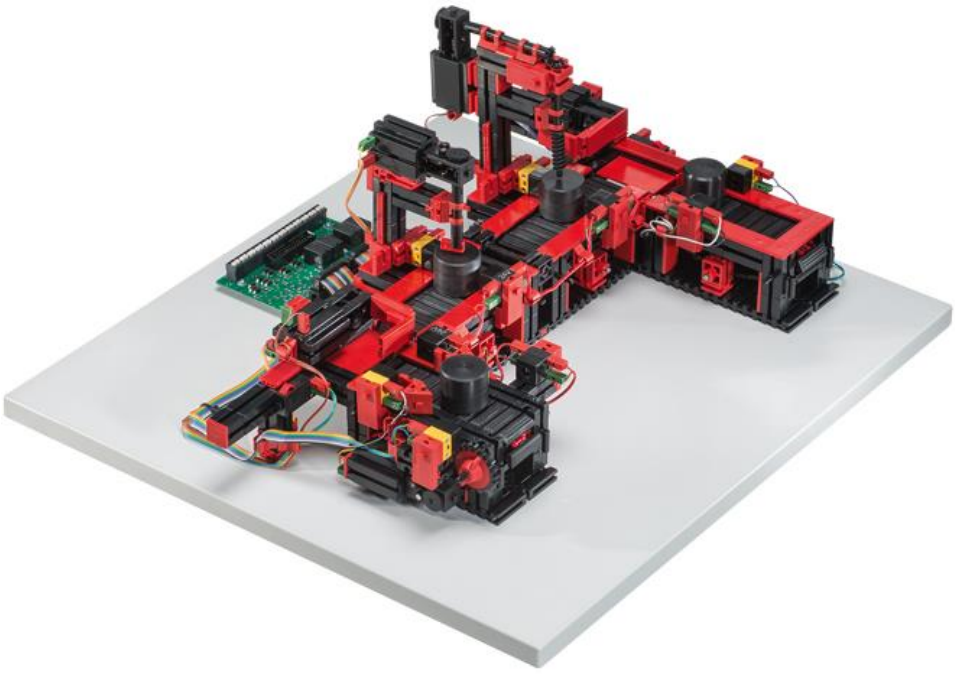
2018 tavaszán a Siemens cég meghirdette az Automation meets Edge nevű versenyét, melyen számos egyetem hallgatói (legalább kettő fős csapatokként) indulhattak saját ötleteikkel. A verseny célja az volt, hogy keressünk új ötleteket az (a 2.3.2 fejezetben is tárgyalt) Edge Computing potenciálját kihasználva, melyek különböző, iparban felmerülő kihívásokat oldanak meg. Az ipar 4.0 paradigmát figyelembe véve az alkalmazásunknak válaszolnia kellett az alábbi kérdések közül legalább egyre:

- Mit tudunk optimalizálni az alkalmazásunkkal az Edge Computing segítségével?
- Hogyan tesszük hatékonyabbá a gyárban lévő gépeket az alkalmazásunkkal? (például beszédfelismerés vagy mintafelismerést kihasználva)
- Milyen mesterséges intelligencia megoldások tehetik hatékonyabbá a gépeket?
- Milyen képességek, funkciók segíthetik a gépeket a jövőben? (kiterjesztett valóság, blockchain, új business modell)

Az alkalmazás egy részét vagy egészét a Siemens által fejlesztett SIMATIC Edge Device nevű eszközön kellett megvalósítani. Ez egy számítógép, amire egy a fejlesztők által módosított Debian operációs rendszer került. Célja, az Edge Computing minta alapján a gyárban levő specifikus eszközök (például PLC-k) adatainak összegyűjtése és velük való kommunikáció megvalósítása egy sztenderd, platformfüggetlen interfészen keresztül (például OPC UA), mely képességekkel csökkenhet az adatelérési késleltetés, valamint könnyebbé válik az olyan, az eszközön futó kiszolgáló alkalmazások fejlesztése, melyek különböző kliensekkel kommunikálnak (például monitorozó és beavatkozó kiterjesztett valóság alkalmazás). További célja, hogy tehermentesítse a klienseket a teljesítményigényesebb számításoktól (videófeldolgozás, modellgenerálás), ami tovább javíthatja a késleltetést.

A verseny követelményei és elvárásai alapján úgy gondoltuk, hogy az ötletünk jól illeszkedik feltételekhez és felkeltheti a szervezők érdeklődését, valamint lehetőséget láttunk arra, hogy a ténylegesen iparban tevékenykedő emberektől kapjunk visszajelzéseket, így indultunk a versenyen.

Az ötletünk elnyerte az értékelők tetszését, így meghívtak egy egyhetes hackathon-ra 2018 októberében Nürnbergben, ahol lehetőséget adtak a prototípusunk megvalósítására és bemutatására. A megvalósításhoz biztosítottak egy HoloLens-t, az említett SIMATIC Edge Device-ot, egy FischerTechnik szimulációs modellt és az azt vezérlő SIMATIC ET200SP PLC-t. Az alábbi képen látható a gyártósort szimuláló modell.



**31. ábra Fischertechnik gyártósori szimulációs modell[20]**

Az első kihívást az Edge Device és a szimulációs modell-t vezérlő PLC közti kommunikáció megvalósítása jelentette. Azonban a korábban említett szerverünk jó felépítésének köszönhetően könnyen kitudtuk egészíteni arra, hogy képes legyen OPC UA protokoll alapú kommunikációt létesíteni a nyilvántartott eszközökkel/rendszerekkel, ha ez a mód specifikálva van. Az Edge Device fejlesztőinek az alkalmazások telepítése szempontjából az egyetlen elvárásuk az volt, hogy Docker konténeren belül fussanak. Az első Docker konténerünkbe a szerverünket építettük, amely megvalósításában korábbiakon felül hozzáadtuk a szimulációs modellt új eszköz/rendszerként és felvittük a paramétereit, valamint a felé küldhető kéréseket akciókként specifikáltuk. A második

[20] Forrás: https://content.ugfischer.com/cbfiles/fischer/produktbilder/ft/96790-Taktstrasse-mit-2-Bearbeitungsstationen-24V.jpg

konténerünk az útmutatószerkesztőt futtatta. Ezután telepítve a kliens alkalmazásunkat a HoloLens-re elkészült a prototípusunk monitorozó része. Az útmutatókhoz készítenünk kellett egy jelölőt, amit a HoloLens Vuforia segítségével fel tud ismerni és el tudja rajta helyezni az útmutató objektumait. Ezt a modell mellé elhelyezve és azzal együtt lefényképezve, könnyedén tudtunk készíteni útmutatókat a modellhez, melyekkel már bemutathatóvá tudtuk tenni az útmutatókat is HoloLens-en.

A prototípust a hackathon végén a Siemens vezetői értékelték, és számos hasznos visszajelzést kaptunk. Alapvetően a korábbi kutatásainkat visszaigazolva nagyon jó koncepciónak tartották az Edge Computing és a Kiterjesztett valóság alkalmazások kombinációját. Többen jelezték, hogy a HoloLens szemüveg (hands-free) vezérlése nagyon előnyös ipari környezetben, ugyanis nem foglalja le folyamatosan az ember kezét például a tabletekkel szemben, így a platformra készült alkalmazások komolyan növelhetik a hatékonyságot a munkavégzés közben. A szerverünk Edge platformon való megvalósítását nagyon előnyösnek tartották (sok egyéb szempont mellett) biztonság szempontjából, ugyanis az adat nem kerül ki felhőbe, vagy olyan környezetbe, ahol akár illetéktelenek is hozzáférhetnek. Továbbá kiemelték, hogy a felületünk kontextusfüggő információ megjelenítése nagyon jó koncepció, ugyanis egy igen sok monitorozható eszközből álló ipari területen sokszor problémát okoz idő szempontjából az éppen keresett eszközhöz tartozó információ elérése. Kiemelték, hogy a megoldásunkkal a kontextus alapján több terület (mint például logisztika, üzemeltetés) információját tudjuk hatékony módon megjeleníteni.

Azonban néhányan kételkedtek abban, hogy ezt meg lehet valósítani és használni robosztus módon az iparban. Felmerült az is, hogy a HoloLens esetleg nem túl kényelmes a súlya, kialakítása és a látószöge miatt. Gondolatként megemlítették, hogy a szöveges útmutatók kiváltása jó ötlet, azonban ehhez kényelmes(ebb) útmutató szerkesztőre van szükség, hogy könnyen és gyorsan lehessen előállítani ezeket. Végül talán filozófiai kérdésként merült fel az, hogy ha mindenki az útmutatókra alapozik, és nem marad szakértő a gyárban, akkor ennek mi lesz a következménye?

# 9 Összefoglaló

A dolgozatot az Ipar 4.0 koncepció áttekintésével és főbb tulajdonságainak megvizsgálásával kezdtük abból a célból, hogy egy távoli, átfogó képet kapjunk a témakörről, amelyben felmerült problémák megoldásával foglalkozunk. Majd áttekintettük a legmeghatározóbb technológiákat, amelyek szerepet játszanak az gyártás új paradigmájának megvalósításában. Részletesebben kifejtettük a kiber-fizikai gyártósorok és a virtuális és kiterjesztettet valóság témakörét, mivel a dolgozat további részeit tekintve ezen technológiák alapszintű ismerete elengedhetetlen.

Ezután bemutattuk az általunk javasolt megoldást egy kontextusfüggő monitorozó rendszerre a szükséges komponenseivel és a megvalósítás kihívásaival együtt. Szerveroldalon a különböző kommunikációs protokollokkal való együttműködés és a rendszerek egységes kezelése voltak, különös tekintettel a beavatkozásra. A kliensoldali prototípus esetében pedig a felhasználóbarát felület kialakítását, a rendszerek felismerését, az interaktív útmutatók 3D elemeinek térbeli pozícionálását, illetve az ehhez szükséges 3D objektumok előállítását azonosítottuk, mint fő megoldandó problémák. Továbbá az interaktív útmutatók szerkesztésének lehetővé tétele is feladatunk volt, ahol kitételként szabtuk meg azt, hogy a felhasználó részéről ne igényeljünk különösebb előtudást, mégis kényelmes legyen az útmutatók készítése és a szerkesztő elérése is a lehető legegyszerűbb legyen.

A megoldási javaslatunk és az általunk azonosított főbb kihívások tárgyalása után végighaladtunk a megvalósításon. Részleteztük a fontosabb elemeket és bemutattuk azt, hogy a tervezett rendszer miként ad választ a fent említett problémákra. Ahol lehetőségünk volt, kitértünk a kapott visszajelzésekre és szemléltettük, hogy az általunk készített prototípus valóban a kívánt módon működik.

A dolgozat zárásaként megosztottuk az integrálási tapasztalatainkat, melyeket egy valós gyártásori beszállító, a Siemens közreműködésével hajtottunk végre. Az általuk szervezett Automation meets Edge nevű verseny remek lehetőséget szolgáltatott számunkra a koncepciónk kipróbálására valós ipari eszközökön, az Edge Computing potenciálját kihasználva. Miután a megoldásunk elnyerte a szakmai zsűri tetszését, másik kilenc csapattal együtt részt vettünk egy egyhetes rendezvényen, ahol implementálhattuk a megoldásunkat. A hackathon keretében szerzett tapasztalataink és a kapott

visszajelzések validációt szolgáltattak a témakörben általunk azonosított kihívásokra és az ezekre választ adó megoldásainkra is. Ezen felül, a kapott szakmai vélemények remek alapot szolgáltatnak a jövőbeni fejlesztések tekintetében, mivel több olyan, tapasztalatból adódó javaslatot kaptunk, melyek alátámasztják az általunk elképzelt eseteket.

Véleményünk szerint a megvalósíthatóság kritikus kérdéseit sikerült megválaszoljuk a dolgozat során a kifejlesztett prototípussal. Továbbá meggyőződhettünk arról is, hogy az általunk kijelölt irány jó, és van igény a hasonló ember-gép interakciót segítő megoldásokra. Azonban ezt a feltáró munkát még több akadály leküzdése választja el attól, hogy a javasolt megoldásunk valós ipari környezetben használható termék legyen, ennek elérése érdekében a továbbiakban a kijelölt útvonalon haladunk a fejlesztéssel.

# 10 Irodalomjegyzék


[1] I. 4. W. Group, „Recommendations for implementing the strategic initiative INDUSTRIE 4.0," 2013.

[2] P. D. E. a. S. Policy, „Industry 4.0 Policy Department Economic and Scientific Policy," 2016.

[3] L. Monostori, „Cyber-physical Production Systems: Roots, Expectations and R&D Challenges," 2014.

[4] M. S. M. L. D. Z. Dominic Gorecky, „Human-Machine-Interaction in the Industry 4.0 Era," Innovative Factory Systems, Kaiserslautern, 2014.

[5] T. A. a. L. I. G. Bouchlaghem N., „VIRTUAL REALITY APPLICATIONS IN THE UK's CONSTRUCTION INDUSTRY".

[6] J. M. M. I. P. A. M. E. H. W. Gabriel Evans, „Evaluating the Microsoft HoloLens through an augmented reality assembly application," Mechanical Engineering Conference Presentations, Papers, and Proceedings, 2017.

[7] T. M. F.-C. Ó. B.-N. M. A. V.-M. PAULA FRAGA-LAMAS, „A Review on Industrial Augmented Reality Systems for the Industry 4.0 Shipyard," 15403 Ferrol, Spain, 2018.

[8] C. Resnick és D. Clayton, „OPC Technology Well-positioned for Further Growth in Tomorrow's Connected World," 2018.

[9] „MixedRealityToolkit-Unity," [Online]. Available: https://github.com/Microsoft/MixedRealityToolkit-Unity.

[10] „Unity," [Online]. Available: https://unity3d.com/.

[11] „Universal Windows Platform," [Online]. Available: https://docs.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide.

[12] „Mixed Reality Design Labs," [Online]. Available: https://github.com/Microsoft/MixedRealityDesignLabs_Unity.

[13] J.-R. J. Hanas Subakti, „A Marker-Based Cyber-Physical Augmented-Reality Indoor Guidance System for Smart Campuses," 2016.