ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

**NEUMANN JÁNOS
INFORMATIKAI KAR**

OKOSIRÁNYTŰ

SMART COMPASS

OE-NIK Hallgató neve: **Fila Norbert László**
2022 Hallgató törzskönyvi száma: **T/005625/FI12904/N**

Óbudai Egyetem
Neumann János Informatikai Kar
Kiberfizikai Rendszerek Intézet

SZAKDOLGOZAT FELADATLAP

Hallgató neve:	**Fila Norbert László**
Törzskönyvi száma:	T/005625/FI12904/N
Neptun kódja:	[redacted]

A dolgozat címe:

Okos iránytű
Smart compass

Intézményi konzulens:	Somlyai László
Külső konzulens:	
Beadási határidő:	2021. december 15.
A záróvizsga tárgyai:	Számítógép architektúrák IoT, beágyazott rendszerek és robotika specializáció

A feladat

Tervezzen és valósítson meg egy olyan irányt mutató eszközpárt, melyek egy-egy Androidos mobiltelefonhoz kapcsolódva képesek meghatározni a másik eszközhöz tartozó irányt a hozzájuk kapcsolódott mobilhoz tartozó helyadatok alapján. Az eszköz legyen képes önkalibrációra, továbbá ehhez tartozzon egy mobilapplikáció is, mellyel képesek vagyunk az eszközt másik, ugyanilyen eszközzel párba állítani, de lehetőségünk legyen az eszköz mutatóját akár egy helyszín felé beállítani. A rendszer kialakításakor, a tervezés és megvalósításkor vegye figyelembe a hasonló működésű aktivitásmérőket, illetve okosórákat. Készítsen kimutatást a hasonló eszközök népszerűségéről, árairól. Részletesen dokumentálja a tervezés, fejlesztés, megvalósítás és tesztelés egyes lépéseit, valamint a jövőbeli ideális árára vonatkozó becslést, mely a hasonló rendszerek (okosóra, aktivitásmérő) megvizsgálásából adódik.

A dolgozatnak tartalmaznia kell:

- a megoldandó feladat pontos leírását és részletes elemzését,
- a hasonló eszközök működésének vizsgálatát,
- a hasonló eszközök külső jellemzőinek vizsgálatát marketing szempontból,
- a felhasznált alkatrészek választásának okait,
- a rendszertervet és a döntési indoklásokat,
- a megvalósítás menetét,
- a tesztelési tervet, a tesztelés eredményeit, a fejlesztés során felmerült problémákat és azok megoldásait,
- az elkészült rendszer felhasználási- és továbbfejlesztési lehetőségeit.

ÓBUDAI EGYETEM * Neumann János Informatikai Kar * Budapest I.

Ph.

.....
Dr. Fleiner Rita
intézetigazgató

A szakdolgozat elévülésének határideje: **2023. december 15.**
(OE TVSz 55.§ szerint)

A dolgozatot beadásra alkalmasnak tartom:

.....
külső konzulens

.....
intézményi konzulens

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat / diplomamunka saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatomban / diplomamunkámban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja.

Budapest, 2022. május 13.

hallgató aláírása

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar
9. sz. melléklet

KONZULTÁCIÓS NAPLÓ

Hallgató neve: Fila Norbert László

Neptun kód: [redacted]

Tagozat: Mérnök informatikus BSc 2017

Telefon: [redacted]

Levelezési cím (pl: lakcím): [redacted]

Szakdolgozat / Diplomamunka[1] címe magyarul:

Okosiránytű

Szakdolgozat / Diplomamunka[2] címe angolul:

Smart compass

Intézményi konzulens: Somlyai László

Külső konzulens:

Kérjük, hogy az adatokat nyomtatott nagybetűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2021.02.26.	Téma kiválasztása	Somlyai László
2.	2021.04.15.	Felmerült kérdések megbeszélése	Somlyai László
3.	2021.05.08.	Felmerült problémák megbeszélése	Somlyai László
4.	2021.05.12.	A dokumentum ellenőrzése	Somlyai László

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplomamunka 2 / Diplomamunka 3 / Diplomamunka 4[3] tantárgy követelményét teljesítette, beszámolóra / védésre[4] bocsátható.

Budapest, 2021.05.14.

Somlyai László
intézményi konzulens

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

KONZULTÁCIÓS NAPLÓ

Hallgató neve: FILA NORBERT LÁSZLÓ

Neptun Kód: [redacted]

Tagozat: MÉRNÖK INFORMATIKUS BSC 2017

Telefon: [redacted]

Levelezési cím (pl.: lakcím): [redacted]

Szakdolgozat / Diplomamunka[1] címe magyarul:
OKOSIRÁNYTŰ

Szakdolgozat / Diplomamunka[2] címe angolul:
SMART COMPASS

Intézményi konzulens: SOMLYAI LÁSZLÓ

Külső konzulens:

Kérjük, hogy az adatokat nyomtatott nagy betűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2022.03.07.	Felmerült problémák megbeszélése	Somlyai László
2.	2022.04.04.	Felmerült problémák megbeszélése	Somlyai László
3.	2022.04.28.	Felmerült problémák megbeszélése	Somlyai László
4.	2022.05.10.	A dokumentum ellenőrzése	Somlyai László

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplomamunka 2 / Diplomamunka 3 / Diplomamunka 4 [3] tantárgy követelményét teljesítette, beszámolóra / védésre [4] bocsátható.

Somlyai László

Intézményi konzulens

Budapest, 2022.05.18.

Tartalomjegyzék

1 Bevezetés

Egy olyan okosiránytű megépítése a cél, mely képes kommunikálni egy Androidos telefonnal. Az eszköz a telefon helyadatai alapján képes meghatározni egy másik telefon, egy szolgáltató vagy egy térképen megadott pont irányát és távolságát.

Az okosiránytű az első elképzelések alapján egy olyan ajándéktárgy lett volna, aminek célközönsége főleg (táv)kapcsolatban álló emberek. Azonban a gondolatot tovább víve kiderült, hogy általánosságban véve a gyalogosoknak és bicikliseknek is hasznos lehet egy ilyen eszköz. Segítséget nyújthat a nehezebben tájékozódó, illetve a környéket nem ismerő emberek számára, ha sürgősen keresünk egy ATM-et, akkor három pöccintés alatt már tudjuk is, merre kell mennünk, és még térképet sem kell leolvasnunk hozzá, nem kell behatárolnunk, hogy éppen merre kellene elindulnunk. Emellett a személyes találkozókat is segítheti, az app használatakor véletlenül sem tudjuk elkerülni a másikat. Kerékpározók, motorosok navigációját is megkönnyítheti egy ilyen apró, egyszerűen leolvasható és kezelhető eszköz.

Habár a GPS vagy telefon pontosan megmutatja, milyen úton kell menni egy adott hely felé, ezeket az eszközöket nem kényelmes útközben a kezünkben tartani, valamint a telefontolvajok is kockázatot jelentenek, hiszen mégiscsak rengeteg személyes adatunk van a mobilunkon. A lezárt autónkban sem hagyunk jól látható helyeken értékes tárgyakat, hiszen bárki feltörheti azt. Ugyanígy nem a legbiztonságosabb, ha drága telefonunkat tartjuk a kezünkben útközben.

Az eszközhöz tartozik egy szoftver is, ezzel lehetséges az egyes eszközöket párba állítani. Egy iránytűhöz több másik iránytű (profil) is tartozhat, de a szoftver maga is tartalmaz egy virtuális iránytű funkciót, így magára a hardverre nem feltétlenül van szükség ahhoz, hogy ezt a navigációs technikát alkalmazhassuk.

Ahhoz, hogy az eszköz eladható és versenyképes legyen a hasonló, illetve még „okosabb” eszközökkel (például okosórákkal és aktivitásmérőkkel) szemben, fontos, hogy az iránytű legyártása minél olcsóbb legyen, vagyis a lehető legolcsóbb komponenseket kell kiválasztani a megépítés során.

A piacon jelenleg két hasonló eszköz található meg. Mindkettőt ugyanaz a cég értékesíti. Ezek az eszközök letisztult módon képesek a navigációra: egy egyszerű nyíl mutatja, merre van a célunk, illetve mutatja, hol kell lekanyarodnunk. Mivel ez főként a kerékpárosok és motorosok navigálását segíti, hiszen nem kell a telefont elővenniük ahhoz, hogy megnézzék, jó irányba haladnak-e, ezért a kerékpárosok és motorosok a cég célcsoportja.

2 Irodalomkutatás

Okosiránytűből a piacon két darab termék van jelenleg, illetve ezen kívül még található két másik projekt. Az irodalomkutatás során megismert technológiák a későbbiekben részletezésre kerülnek.

2.1 Okosiránytűk

A „smart compass” kifejezésre való keresés után mindössze négy eszközt lehet találni az interneten, azokat is kissé nehezebben, ugyanis nem a Google első találatai között találhatók meg. Ebből kettő a piacon valóban megjelent termék, de mind a kettő ugyanazon cégé. Egy másik egy házi projekt, illetve a harmadik a kickstarter nevű oldalon található projekt. Ez a piacon nem jelent meg, illetve a kommentszekcióban többen is arra panaszkodnak, hogy nem kapták vissza a pénzüket, átverés volt, a cég honlapja pedig elérhetetlenné vált.

2.1.1 Pilgrim

A Pilgrim [1] (lásd 2.1 ábra) nevű eszköz egy egyszerű, 2016-os házi projekt. Sajnos csak egy-két információt oszt meg a publikummal.

Használata nagyon egyszerű. Kiválasztjuk, hová szeretnénk menni, például egy kávézóba, és megadja a hozzánk legközelebb eső ilyen helyet. Emellett a jelzőfények azt is jelzik, mennyire vagyunk közel a célunkhoz.

Digitális kijelzővel nem rendelkezik, Arduino segítségével programozta le. A hely kiválasztását követően a kód a Yelp API-hoz fordul, ami visszaadja a legjobban illeszkedő eredmény koordinátáit. Ezután a mágneses tű a felhasználó aktuális pozíciójához képest elfordul a cél irányába.

A programozása egy Genuino MKR 1000 boarddal történik. Ennek nem csak nagyobb memóriája van, de lehetővé teszi, hogy az eszköz az internethez csatlakozzon. Ezen felül az eszköz rendelkezik magnetométerrel és gyorsulásmérővel is, ezekre a szenzorokra van szüksége ahhoz, hogy meghatározza a cél irányát. A léptetőmotor és a forgó kódkapcsoló pedig a tárcsához szükségesek [2].

2.1. ábra – Pilgrim

2.1.2 HAIZE

A HAIZE nevű projekt a kickstarter.com oldalon fellelhető [3], [4]. Még 2015. októberében indult, és a 911 felhasználó által összesen 63.055 font (átszámítva kb. 26.063.585 forint) támogatás érkezett rá. Utoljára 2018-ban jelentkeztek a készítők. Az itt olvasható információkat mindenképpen fenntartásokkal kell kezelni, ugyanis a projekt sikertelen volt, illetve több kommentelő is leírja, hogy csalás áldozataivá váltak. Viszont a tervezett megvalósításáról és funkcióiról érdemes írni.

Az eszköz kör alakú háza alumíniumból készült. 42.5 mm az átmérője, illetve 10 mm a magassága. A házba egy rugalmas szalag van integrálva, mellyel az eszköz a biciklikormányra rögzíthető. Az eszköz tömege körülbelül 35 gramm.

LED kijelzővel rendelkezik. A kijelzőn mindösszesen két színes pont található, ezek közül az egyik az irányt mutatja, a másik pedig a cél közelségét (lásd 2.2 ábra). Ez a pont közeledve a célhoz egyre gyakrabban villog. Az eszköz a LED-ek fényerejét automatikusan szabályozza a fényviszonyok függvényében.

2.2. ábra – A kék fényű LED az irányt mutatja.

[5]

Az okosórákhoz és más eszközökhöz hasonlóan ez is kapott egy pántot, így az ember csuklójára is lehet rögzíteni.

A termékben több szenzor is megtalálható. Van benne magnetométer, gyorsulásmérő, giroszkóp és fényérzékelő. A magnetométer a célhoz tartozó irányt, a saját pozícióját pedig a gyorsulásmérő és a giroszkóp segítségével határozza meg. A fényérzékelő a korábban említett automatikus fényerő szabályozás miatt szükséges.

Az eszköz egy 300 mAh-s akkumulátort kapott, így akár két hétig is használható. Feltöltése micro USB kábellel történik.

A hozzátartozó applikációval pedig Bluetooth-on (4.0) keresztül kommunikál. Az applikáció követi a felhasználó aktivitását, illetve méri a kalóriafogyasztást is. Képes értesíteni a tulajdonosát bejövő üzenetekről és hívásokról, valamint felismeri, ha balesetet szenvedett, és automatikusan értesíti a vész esetén értesítendő személyt. Ezen kívül olyan funkciója is van, mely lehetővé teszi, hogy ismerőseink, barátaink pozíciója felé mutassa az eszköz az irányt.

2.1.3 Beeline Velo

A Beeline egy londoni székhelyű vállalat, mely 2015-ben startolt. Jelenleg kétféle iránytűt gyártanak. Az egyiket kerékpárosoknak (Beeline Velo), a másikat motorosoknak (Beeline Moto) szánják [6], [7].

A Velo (lásd 2.3 ábra) egy 350 mAh-s akkumulátorral rendelkezik, ami micro USB kábellel tölthető, és amivel legkevesebb 10 órán keresztül használhatjuk az eszközt. Készenléti állapotban 2-3 hónapig bírja. A kijelzője 30 mm átmérőjű és 200 x 200 pixel felbontású, illetve MIP (Memory-in-Pixel) technológiával készült. Rendelkezik gyorsulásmérővel, giroszkóppal és magnetométerrel. Bluetooth-szal (4.0) képes kapcsolódni a mobilhoz. Az eszköz szilikon gumiból, nejlonból, ABS műanyagból és üvegből épül fel. Négy kapacitív érintőgombbal rendelkezik, illetve, ha hevederbe van szerelve, akkor ellenáll víznek és részecskének (IP66 vagy IP67-es besorolású, nem egyértelmű, ugyanis mindkettőt említik). Egy ARM Cortex processzor dolgozik benne. 65 mm átmérőjű, 18 mm magas, tömegét nem közlik. Ára: 99 font (kb. 41160 forint), ami tartalmazza magát az eszközt, egy hevedert, egy töltőkábelt, matricákat, illetve egy kulcstartót, amire rá lehet helyezni az iránytűt.

2.3. ábra – Beeline Velo

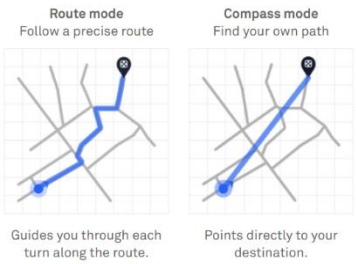
[8]

A Motoba (lásd 2.4 ábra) jobb teljesítményű, 400 mAh-s LiPo akkumulátor került, ami legkevesebb szintén 10 órán keresztül használható. Készenléti állapotban viszont akár 6 hónapig is bírja. Töltése USB-A típusú kábellel történik, amihez pogo tű interfész szükséges. A kijelzője szintén MIP technológiát használ transzflektív LCD-vel, LED háttérvilágítással. 26 mm átmérőjű és 208 x 208 pixel felbontású. Ugyanazokkal a szenzorokkal rendelkezik, mint a Velo, tehát van benne gyorsulásmérő, giroszkóp és magnetométer. Az eszköz szilikon gumiból, TPU (thermoplastic polyurethane) anyagból, ABS műanyagból és edzett ragasztott üvegből áll. Négy fizikai gombbal rendelkezik, így kesztyűben is könnyedén kezelhető. IP67-es besorolású. A mobiltelefonhoz való kapcsolódása Bluetooth-szal (4.0) történik. 49,8 mm átmérőjű, 18,6 mm magas, 30 grammot nyom. Ára: 149 font (kb. 61950 forint), ami kulcstartó helyett a motorhoz szükséges rögzítő eszközt tartalmaz.

2.4. ábra – Beeline Moto

[9]

Mindkét eszköz kétféle navigációs móddal (route és compass) rendelkezik (lásd 2.5 ábra). A route móddal pontos útvonalat kaphatunk, míg compass módban csak a cél felé mutat a rendszer, az útvonalat a felhasználóra bízza. Az alkalmazás a Google Maps adatait gyűjti be, így bárhol lehet használni. A Strava nevű appot is integrálták a saját szoftverükbe, ami lehetővé teszi, hogy a megtett utakat feltöltse a szerverre, illetve GPX-et használ az útvonaltervezéshez. Ezen kívül egyébként az út megkezdése óta eltelt időt, a pontos időt, a megtett távolságot és a sebességet is képes jelezni. Éjszaka automatikusan bekapcsol a háttérvilágítás. Az applikáció az angol, francia, német és japán nyelveket támogatja.



2.5. ábra – Beeline: route és compass mód

[4]

3 Technológiák

Az okosirányтűk vizsgálata után jöjjön a komponenseiknek, illetve tulajdonságaiknak a vizsgálata.

3.1 Anyagok

Az ABS (akrilnitril-butadién-sztirol) kemény és ütésálló anyag. -20 és +80 Celsius-fok között tartja meg kedvező tulajdonságait. Az időjárással szemben gyenge az ellenállása, így beltéri alkalmazásokban javasolt a használata [10].

A TPU kopás- és karcálló, vagyis jó választás az okosirányтű külső burkolatához, illetve akár a kábelekhez is használható [11].

3.2 Kommunikáció

A Bluetooth 4.0 használata alacsonyabb fogyasztást és 100 méteres hatósugarat biztosít. A Bluetooth 4.1 kiszűrte a 4G frekvencia zavaróhatásait, emellett az eszköz intelligens módon fel-le csatlakozik a vele párban álló eszközre, ezzel energiát spórolva meg. A 4.2-es verziónak javult az energiagazdálkodása, az adatok védelme erősebb lett. A Bluetooth 5 pedig már még nagyobb hatótávval, nagyobb adatátvitellel rendelkezik [12], [13].

A Bluetooth 5 beltéri használatkor kevesebbet fogyaszt, mint a korábbi verziók, azonban szabadtéri használatánál kicsivel, de többet fogyaszt [14]. Az okosirányтűnél fontosabb szempont az, hogy minél ritkábban kelljen feltölteni, mint az adatátvitel sebessége, mivel kevés adattal kell dolgoznia. Ezért a Bluetooth 4.x verzió lesz használva, azonban érdemes lehet a 4.1 vagy a 4.2-vel helyettesíteni.

3.3 Akkumulátorok

Az elektromos készülékekben leggyakrabban Lithium-ion és Lithium-polymer akkumulátorokat találhatunk [15], [16]. Bár a kijelentést nem bizonyítja, a termék árbecsléséhez szükséges kutatómunkát már elkezdtem, összesen 13 objektum, okosóra és aktivitásmérő vizsgálata történt eddig meg, és mind a 13 termék vagy Li-Ion, vagy Li-Poly akkumulátorokkal rendelkezett.

A Lithium-ion akkumulátorra jellemző, hogy olcsóbb, viszont nehezebb, és nem bírja a hőt. A Lithium-polymer akkumulátorok kisebbek, könnyebbek, és több energiát képesek eltárolni, mint a Li-Ion akkumulátorok [15].

3.4 Szenzorok

A vizsgált eszközökben csak a Pilgrim nem rendelkezik giroszkóppal, azonban mindegyik eszköznek van gyorsulásmérője és magnetométerje.

A gyorsulásmérő használható a gyorsaság, a sebesség és az orientáció meghatározásához. Ahhoz, hogy a dőlésszöget vagy oldalirányt is meg lehessen határozni, szükség van giroszkópra is. Egy-egy háromtengelyes gyorsulásmérővel és giroszkóppal zajtól mentes és érzékeny mérések végezhetők [17].

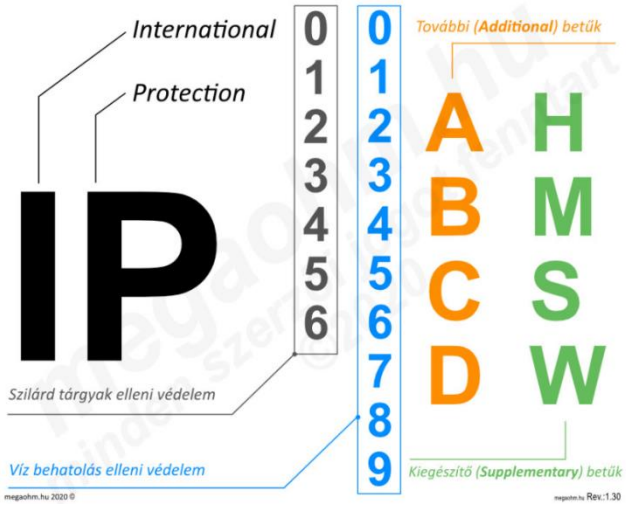
A telefonokban található magnetométer azért fontos, mert ezáltal mindig lesz egy kitüntetett pont, nevezetesen az északi irány, így a digitális térképek a telefon helyzetétől függően forgathatók [17].

3.5 Víz és részecske elleni védelem

A burkolatok különböző mértékű védelmet nyújtanak az elektromos készülékeknek. A Beeline Moto mellett IP67-es kód szerepel, a Beeline Velonál pedig IP66, bár ez utóbbi nem egyértelmű, mivel a honlapjukon az IP67-est is megemlítik ugyanennél a terméknél.

Az IP kódok [18] a védettségi fokozatnak az osztályozására szolgáló rendszer. A kód öt részből épül fel:

1. IP, mint International Protection.
2. Az IP után szereplő első érték a szilárd tárgyak elleni védettséggel kapcsolatos.
3. A második érték a víz behatolás elleni védettség nagyságát jelzi.
4. A számpár után szerepelhet további A, B, C, D, illetve
5. H, M, S, W karakter is.



3.1. ábra – A kód összetétele

[18]

Az IP66 azt jelenti, hogy a port egyáltalán nem engedi be a burkolat, viszont csak erős vízsugár ellen képes védeni. Vagyis abban az esetben tönkremegy a készülék, ha beletesszük egy pohár vízbe.

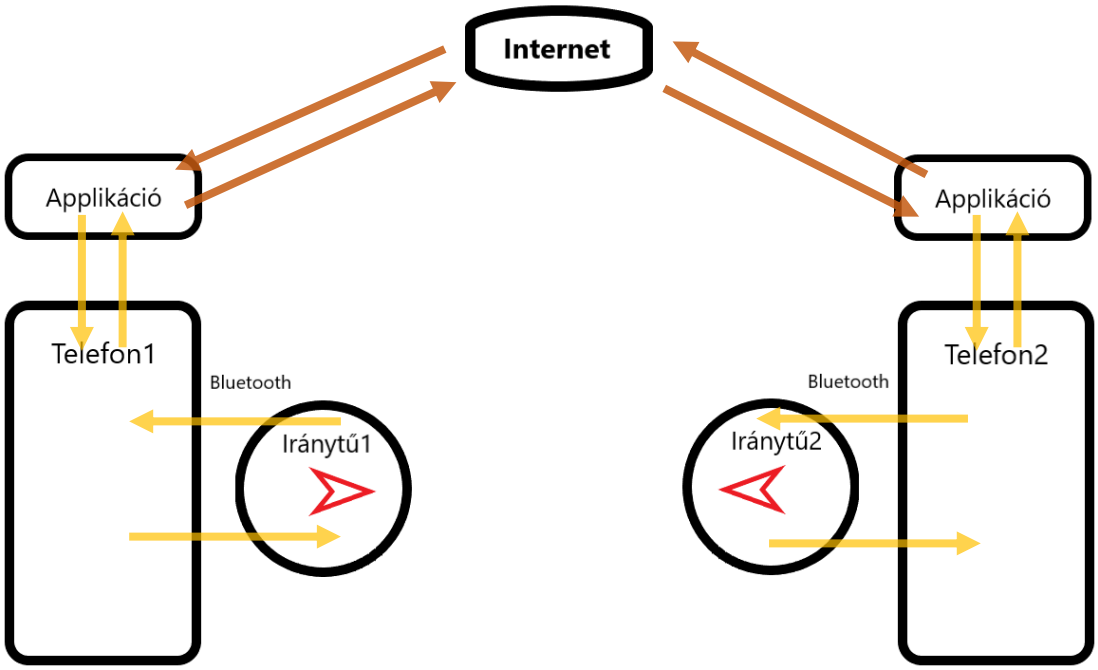
Az IP67-es kóddal ellátott készülék szintén nem engedi be a port a belsejébe, viszont ezt az eszközt már akár víz alá is lehet meríteni. Fontos megjegyezni, hogy csak bizonyos időn keresztül, bizonyos nyomás alatt képes csak védeni az eszközt a víz ellen.

3.6 GPS eXchange Format (GPX)

A Beeline képes GPX fájlokat konvertálni, de mik is azok a GPX fájlok? A GPX egy nyílt XML-szabvány, aminek segítségével helyszínek földrajzi koordinátáit és útvonalakat adhatunk meg. A GPX fájlból ki lehet olvasni egy adott ponthoz tartozó földrajzi szélességet és hosszúságot, magasságot és egyéb információkat is [19], [20].

4 Rendszerterv

A rendszer négy főkomponensből áll: egy szerverből, egy alkalmazásból, két telefonból és két iránytűből. Az egyik telefon elküldi az applikáción keresztül a szervernek a saját azonosítóját, illetve a GPS-koordinátáit. A szerverről ezeket az adatokat a másik felhasználó képes lekérdezni, és ez alapján meghatározni a másik eszközhöz tartozó irányt. A fogadó iránytűje megkapja mind a küldő, mind pedig a fogadó telefonjának a GPS-koordinátáit. Ebből kiszámításra kerül a másik eszköz iránya, és a tű abba az irányba fordul.



4.1. ábra – A rendszer egyszerű modellje

4.1 Telefon

Elsősorban Androidos telefonra történik a fejlesztés. Ennek oka, hogy mindig is androidos eszközeim voltak, illetve egyszerű blokk alapú programokat már készítettem MIT App Inventorban, így az Android áll hozzám a legközelebb.

4.2 Iránytű

Ma már a legtöbb Androidos telefonban van GPS adóvevő [21][1]. Mivel az egyik legfontosabb cél az, hogy minél olcsóbb legyen az eszköz gyártása, ezért az iránytű a telefon GPS jelét fogja használni, vagyis saját GPS vevője nem lesz. Ez némi

[1] „Are STRONGLY RECOMMENDED to include a GPS/GNSS receiver.”

pontatlansággal jár, de a mobiltelefon és az iránytű elég közel vannak egymáshoz, hiszen a telefon többnyire az embernél van, így a hiba nem lesz jelentős.

Bár GPS vevője nem lesz, magnetométerrel fel kell szerelni, különben az eszköz nem tudja, milyen irányba néz a világkoordináta-rendszerben. Ebből pedig az következne, hogy bár az alkalmazás meghatározná az irányt, ha az eszközt forgatnánk, a tű és az eszköz átmérője által bezárt szög nem módosulna, vagyis már nem a cél felé mutatna. Emellett lehetséges, hogy szükséges lesz az eszköznek giroszkópra és/vagy gyorsulásmérőre, amennyiben a léptetőmotor eltávolításra kerül. A Pilgrimben elegendő volt egy magnetométer és egy gyorsulásmérő az irány detektálásához.

Az akkumulátor típusa attól is függ, hogy az eszköz mennyi idő alatt merül le egy feltöltéssel. Amennyiben csak 24 óra vagy ennél kevesebb ideig bírja, érdemes megfontolni a Lithium-polymer akkumulátorok használatát. Ugyanakkor, ha a jövőben megfelelőnek bizonyul az élettartama, érdemes inkább Lithium-ion akkumulátort használni, ezáltal a termék előállítása kevesebb pénzből történik meg, és kisebb áron is eladható ugyanakkora nyereség mellett.

Az eszköz „kijelzője” LED-ekből fog állni. Mindig egy adott LED fog világítani attól függően, hogy melyik irányban található a cél.

Tartalmaznia kell egy Bluetooth modult, amivel képes kommunikálni a telefonnal.

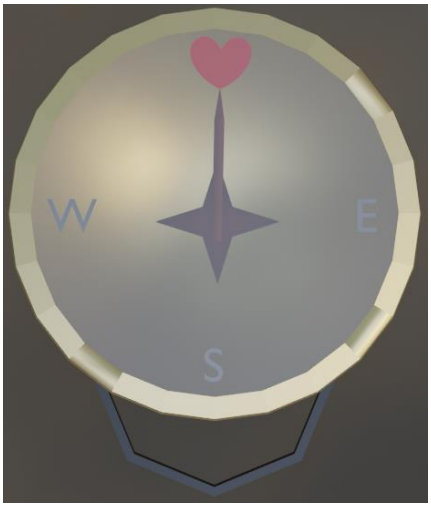
Az iránytű prototípusa egy Arduino Mega ADK-val lesz vezérelve, azaz nem kerül bele külön alaplap. Előnye, hogy a rendszer fejlesztése, tesztelése lépésről lépésre történhet, vagyis a LED-ek megfelelő bekötése után azonnal tesztelhetővé válnak. Utána a csatlakoztathatjuk a kapcsolásba a magnetométert, azt önmagában tesztelhetjük, majd utána tesztelhető az is, hogy a magnetométer forgatása során hogyan kell kiválasztani a megfelelő LED-et. végül pedig a kommunikációhoz szükséges modul tesztelése következik.

4.3 Az iránytű modellje

Az első modell még nagyon kezdetleges, sok módosítást kell még elvégezni rajta, mint például az oldalak kisimítása, más színkombinációk kipróbálása, más mintázatok kipróbálása, az üveglap domborítása, a kulcstartó rész újratervezése stb. A modellek általam készültek a Blender nevű szoftverben.

Az első modell elkészítésének koncepciója az volt, hogy az iránytű pároknak készül, ezzel lett párosítva az iránytű fogalma, azaz a modellen megjelennek az égtájak, illetve középen a csillag, de az északi égtájat felváltotta egy szív.

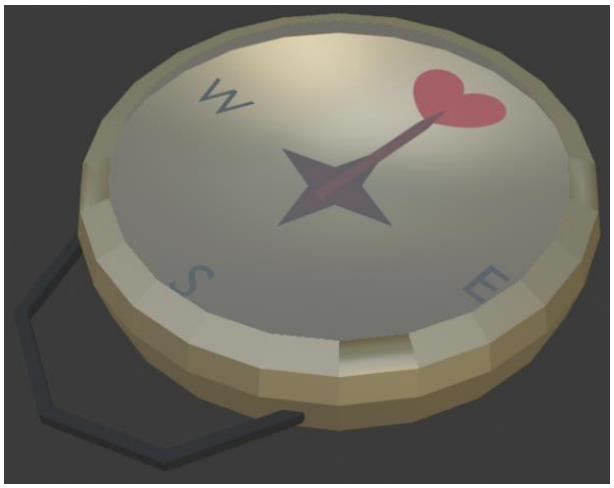
Mivel a termék célcsoportja időközben megváltozott, és már nem kifejezetten a párokat célozza, hanem futárokat is, így szükséges más kialakításokon és megjelenésen is gondolkozni. Az eszközön nincs digitális kijelző jelenleg, helyette egy mozgóalkatrész van (ez lenne a tű), ám a későbbiekben ez is változhat. A modellről hiányzik a léptetőmotor.



4.3. ábra – Felsőnézet

4.2. ábra – Alulnézet

4.5. ábra – Oldalnézet



4.4. ábra – Ferdeszögből

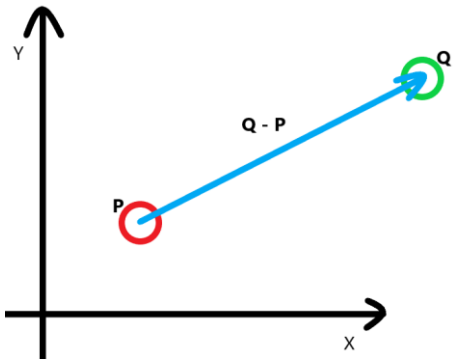
4.4 Applikáció

Az applikáció Androidos okostelefonokra készül, az iránytűnek az adatokat ezen applikáción keresztül lehet majd kiküldeni.

Az applikációval a felhasználónak lehetősége van arra, hogy kiválasszon egy célpontot. A célpont lehet akár egy konkrét GPS-koordináta, egy bizonyos típusú hely (pl. étterem), akár az a másik telefon, amivel párba állt olyan értelemben, hogy a másik telefon koordinátáihoz hozzáférhet. Amennyiben a másik telefont állítja be célnak, úgy ezek a helyadatok a felhasználó számára transzparensek, csak a távolságot és az irányt látja. A konkrét GPS-koordinátának megadását egy térképen tudja megtenni, a párban álló eszközök pedig egy listán jelennek meg, amik közül valamelyikre bökve be is állítja a célt.

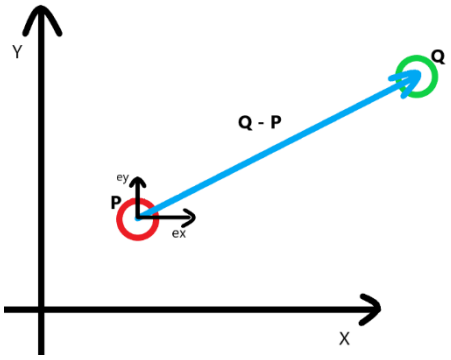
Az irány az első elgondolás szerint tehát a koordináták alapján kerül kiszámításra. A kiszámítása a következőképpen fog történni:

- kiszámítjuk a cél és a saját eszközünk különbségvektorát



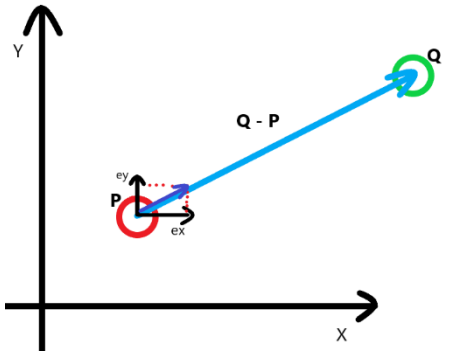
4.6. ábra – Különbségvektor
(P a saját eszköz, Q a cél)

- kiszámítjuk a különbségvektor hosszát
- kiszámítjuk a különbségvektor egységvektorait (ex és ey)



4.7. ábra – Egységvektorok

- végül kiszámítjuk a szinusz és koszinusz összetevőket, és megkapjuk az irányt



4.8. ábra – Megkaptuk a keresett irányt

Az majd csak a tesztelésnél fog kiderülni, hogy elegendő-e két GPS-koordináta az irány kiszámításához, vagy bonyolultabb számításokra van-e szükség. Nagyobb távolságok

esetén biztosan nem elegendő, hiszen akkor az iránytű akár a Föld alá is mutathat, amennyiben a cél a bolygó másik oldalán van. Ebben az esetben Azimuth szögeket kellene számolnunk [22].

Korábban említettem, hogy szükség van magnetométerre is. Ez ahhoz kell, hogy az X és Y tengelyek rögzítettek legyenek, és az eszköz forgatása esetén is megmaradjanak ezeknek a tengelyeknek az irányuk, azaz legyen egy világkoordináta-rendszer.

A szoftver fejlesztése Visual Studióban fog történni. Emellett elképzelhető az Android Studio használata is, illetve az MIT App Inventor lehetőséget biztosít arra, hogy egy gyors képet adjon a felhasználói felület kinézetéről és működéséről [23].

5 A hardver specifikációja

Az általam választott elektronikai eszközöket árusító webáruházból igyekeztem a lehető legolcsóbb megoldásokat kiválasztani. Az áruk mellett azt is figyelembe vettem kiválasztásuk során, hogy könnyen fellelhető forrásaik legyenek, hogy a fejlesztés minél gyorsabban, egyszerűbben és átláthatóbban történhessen meg.

5.1 Kijelző

A kijelző tulajdonképpen egy kör alakban elhelyezett LED sorból fog állni. Két ilyen LED gyűrűt rendeltem (WS2812 és WS2812B), az egyik 12, a másik 16 NeoPixel LED-del rendelkezik. Tulajdonképpen csak az egyikre volt szükség, a másikat csak a biztonság kedvéért vásároltam, ha esetleg az egyik a fejlesztés, tesztelés során valamilyen figyelmetlenség vagy egyéb hiba miatt tönkremenne. A programozásukban nincs számottevő különbség.

A NeoPixel vezérléséhez az adafruit készített egy jól dokumentált Arduino könyvtárat [24], [25], emellett pedig számos példát találtam a használatára, amelyek segítettek az eszköz leprogramozásában [26].

5.2 Magnetométer

A magnetométer ahhoz szükséges, hogy az eszköz forgatásakor a megfelelő LED gyulladjon fel, miközben az előző égő LED kapcsoljon le.

Ennek kiválasztása is hamar megtörtént, a legolcsóbb modulok egyike volt a HMC5883L jelölésű modul, és a nevére rákeresve azonnal találtam megfelelő forrást hozzá [27], amivel könnyedén el tudtam indulni a későbbiekben.

5.3 Bluetooth (nem került be a kapcsolásba)

Egy nRF24L01 modulra esett a választás, erről is számos forrást találtam. Azonban ezzel a modullal volt a legtöbb probléma, és bár az időbe még pont belefért volna egy másik eszköz megrendelése, nem mertem megkockáztatni annak a leprogramozását, hátha az új eszközzel is komolyabb nehézségekbe ütközök, így végül kénytelen voltam más megoldást találni ahhoz, hogy az eszköz képes legyen kommunikálni a mobiltelefonnal. Ez a modul tehát végül nem lett beépítve a rendszerbe.

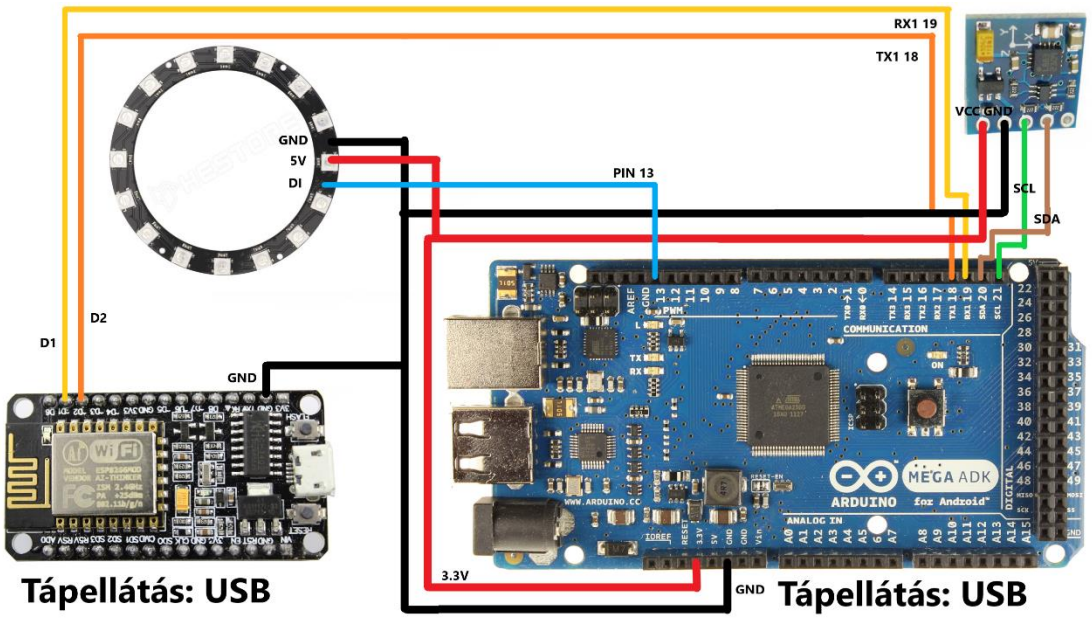
5.4 Wi-Fi (Bluetooth helyett)

A Bluetooth modullal tehát problémák adódtak, és ahhoz, hogy az eszköz tesztelhető legyen, egy Wi-Fi modul (NodeMCU ESP8266) került a kapcsolásba. Ezzel az eszközzel találkoztam az elsők között az egyetemen, egy kisebb projektben is használtam, így ismerős volt számomra. Hátránya, hogy ahhoz, hogy adatot küldhessünk neki, le kell

válnunk a hálózatról, és rá kell csatlakozni a NodeMCU-ra. Ez meglehetősen körülményessé, kényelmetlenné teszi az iránytű használatát

5.5 Kapcsolás

Az Arduino és a NodeMCU USB-n keresztül kapják az áramot, az Arduino pedig biztosítja a LED gyűrű és a magnetométer tápellátását. A LED gyűrű az Arduino 13-as PIN-jéhez csatlakozik, ezen keresztül kapja meg az utasításokat. A magnetométer SCL és SDA kimenete az Arduino SCL és SDA PIN-jeihez csatlakozik. A NodeMCU pedig soros kommunikációt folytat az Arduinóval. D1-es PIN-je az Arduino az RX1 19, a D2-es PIN-je pedig az Arduino TX1 18-as PIN-jéhez csatlakozik.



5.1. ábra – Kapcsolás

6 Az applikáció specifikációja

6.1 Kliens

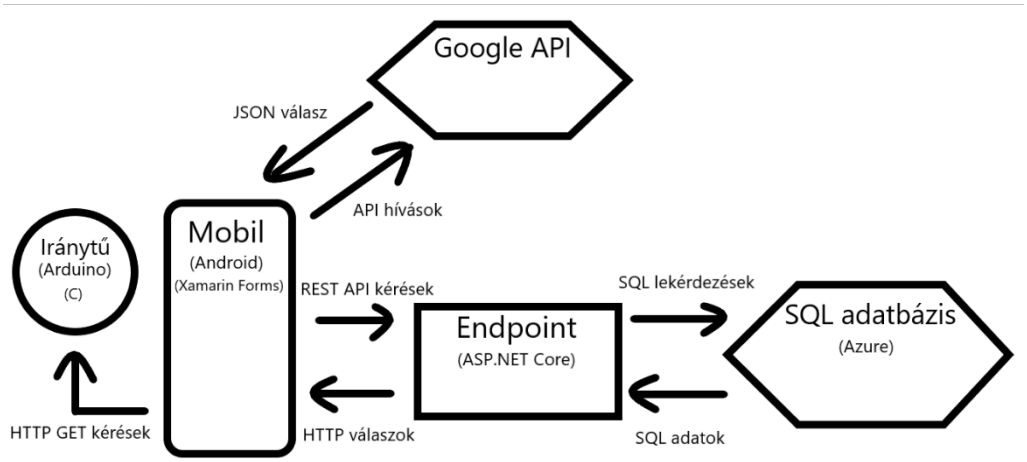
A klienst Android XAML Appban készítettem el, mivel ennek megértése egyszerűbb volt, hiszen a fejlesztés így C# és XAML nyelven történhetett, amikben már tapasztalt vagyok. Az úgynevezett vékony kliens kommunikál az endpointtal, az endpoint felé intéz kéréseket, majd az onnan visszaérkező adatokat jeleníti meg a felhasználónak.

6.2 Endpoint

Az endpoint ASP.NET Core-ban készült el. Közvetlen kapcsolatban áll az adatbázissal, azaz ő tartalmazza az SQL lekérdezéseket, amiket az adatbázis felé közvetíteni lehetséges. Ha a kliens felől érkezik egy kérés, akkor azt az endpont továbbítja az adatbázis felé SQL lekérdezés formájában, a kapott választ pedig visszaküldi a kliensnek.

6.3 A rendszer felépítése

Összefoglalva tehát a kliens REST API kéréseket intéz az endpoint felé, ezeket a kéréseket ő továbbítja az adatbázis felé. Az adatbázis eredményét feldolgozza az endpoint, majd HTTP válaszokban küldi vissza a kliensnek. A kliens emellett API hívásokat intéz a Google felé (amennyiben a térképet szeretné megnyitni vagy adott szolgáltatást keres a környékén), a válaszok pedig JSON formátumban érkeznek vissza. A kliens jelenleg csak nagyon körülményesen, Wi-Fin keresztül képes kommunikálni az iránytűvel. A kliens a cél irányát (szögét) küldi el az iránytűnek.



6.1. ábra – A fő komponensek közötti kommunikáció

7 Külső szolgáltatók

7.1 Adatbázis

Adatbázis-szolgáltató kiválasztásának szempontjai közé tartozott a megbízhatóság, jól dokumentáltság, illetve az, hogy az általam használt szoftverek minél könnyebben tudjanak ezekkel a külső termékekkel együttműködni.

Először a Google szolgáltatásainak kezdtem utánajárni, de hamar elvesztem a rengeteg opció között, nem is igazán tudtam, hogy pontosan melyiket kellene használni. Valójában azért ezzel kezdtem, mert egyértelmű volt, hogy a Google térkép szolgáltatásait fogom használni az alkalmazásban, és szerettem volna, ha egy helyről kapom meg a szükséges funkciókat, komponenseket. Biztonsági szempontból nem biztos, hogy a legjobb, ha egy helyről származik minden, azonban a Google megbízható vállalat, hiszen egy világcégről van szó.

Ha viszont a Microsoft fejlesztőkörnyezetét használom, akkor a legkézenfekvőbb, ha az Azure szolgáltatásait veszem igénybe. Ettől az elején ódzkodtam, mivel korábban végeztem már kisebb kutatómunkát az adatbázisokról, és az Azure felülete nem állt közel hozzám, számomra nem elég letisztult a felülete. Azonban itt jóval hamarabb megtaláltam azt a szolgáltatást, amit szerettem volna használni, illetve a Visual Studióban is percek alatt képes voltam rá csatlakozni. Végül tehát az Azure adatbázis-szolgáltatására esett a választásom.

Az Azure adatbázisának használata az első hónapban ingyenes. Ha lejárt az egy hónap, akkor blokkolják az adatbázishoz való hozzáférést. Ahhoz, hogy újra elérhessük, jelezni kell az Azure felé 1 hónapon belül, különben törlődik az adatbázis. Nekem le is járt az egy hónap, a tesztelések azonban addigra már nagyjából megtörténtek, ennek ellenére szükség volt előfizetni. Most úgy tűnik, az adatbázis használata hóvégére nem éri el az 1 eurót sem. Remélem, ez így is marad.

7.2 Térkép szolgáltatások

Ha térkép, akkor egyértelműen Google. Rengeteg API-val rendelkezik, jól dokumentált, és nagyon sok videó van az egyes szolgáltatásuk használatáról. Az API-k használata 2 hónapig ingyenesen használhatók. Ha és amennyiben jól értelmeztem, a 2 hónap lejárta után is ingyenesek maradnak, és csak abban az esetben kell utánuk fizetni, ha a használatuk egy bizonyos forgalmat elérnek. Az ára is a forgalom függvényében változik. A Google API-k közül kettőre volt szükség az alkalmazás jelen változatában.

7.2.1 Maps SDK for Android

Az alkalmazásban a felhasználónak lehetősége van a térképen kiválasztani egy pontot, és beállítani azt célként. Ezt a térképet valósítja meg a Maps SDK for Android [28] nevű

API. Egy szokványos Google térképet kapunk vele, amit aztán a projektjeink igényeihez mérten módosíthatunk.

7.2.2 Places API

Az alkalmazás egyik fő funkciója, hogy a felhasználó minél kényelmesebben, egyszerűbben és gyorsabban juthasson el a hozzá legközelebb lévő, általa kiválasztott szolgáltatáshoz. A Places API [29] lehetővé teszi, hogy az ilyen szolgáltatásokra (pl. étterem, hipermarket, ügyvéd, könyvtár, kávézó stb.) rákereshessünk. Az eredményt (7.1 ábra) JSON formátumban adja vissza, melyben rengeteg adat megtalálható, többek között a szolgáltató koordinátái, neve, vagy hogy nyitva van-e éppen.

Ez egyébként az egyik lehetőség, ugyanis nem kizárólag egy megadott rádiuszban tudunk helyekre rákeresni az API által, hanem tudunk konkrét szövegre is keresni, így, ha egy bizonyos hely típus nem elérhető, abban az esetben a Places API-nak a Text Search nevű keresési típusát használhatjuk. Az applikáció most csak a Nearby Search-öt használja.


```
{
   "html_attributions" : [],
   "results" : [
      {
         "business_status" : "OPERATIONAL",
         "geometry" : {
            "location" : {
               "lat" : 47.5438004,
               "lng" : 19.0285645
            },
            "viewport" : {
               "northeast" : {
                  "lat" : 47.5451247802915,
                  "lng" : 19.0298709802915
               },
               "southwest" : {
                  "lat" : 47.5424268197085,
                  "lng" : 19.0271730197085
               }
            }
         },
         "icon" : "https://maps.gstatic.com/mapfiles/place_api/i
         "icon_background_color" : "#4B96F3",
         "icon_mask_base_uri" : "https://maps.gstatic.com/mapfil
         "name" : "dm-drogerie markt Kft.",
         "opening_hours" : {
            "open_now" : false
         },
         "photos" : [
            {
               "height" : 3024,
               "html_attributions" : [
                  "\u003ca href=\"https://maps.google.com/maps/c
               ],
               "photo_reference" : "Aap_uEAbiCeIdL-CuUot8zKraYAS
QRl73GvTuk9n2WQZi",
               "width" : 4032
            }
         ],
         "place_id" : "ChIJd18a1tDZQUcR2UYsiPFU5_4",
         "plus_code" : {
            "compound_code" : "G2VH+GC Budapest, Magyarország",
            "global_code" : "8FVXG2VH+GC"
         },
         "rating" : 4.4,
         "reference" : "ChIJd18a1tDZQUcR2UYsiPFU5_4",
         "scope" : "GOOGLE",
         "types" : [
            "drugstore",
            "supermarket",
            "grocery_or_supermarket",
            "electronics_store",
            "home_goods_store",
            "food",
            "health",
            "point_of_interest",
            "store",
            "establishment"
         ],
         "user_ratings_total" : 54,
         "vicinity" : "Budapest, Bécsi út 136"
      },
      {
         "business_status" : "OPERATIONAL",
         "geometry" : {
            "location" : {
               "lat" : 47.5369171,
               "lng" : 19.0324022
```


7.1. ábra –Places API-tól érkező JSON válasz egy-egy részlete

8 Megvalósítás leírása, tapasztalatok

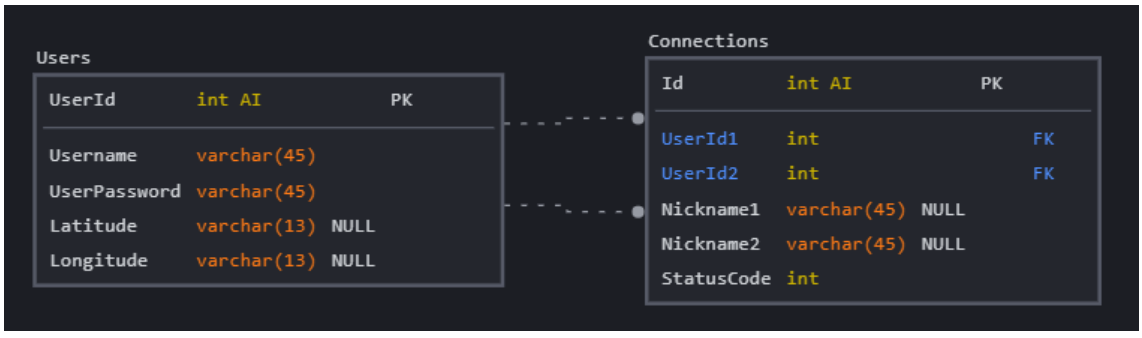
Amíg a megrendelt elektronikai modulokra vártam, nekiláttam az applikáció fejlesztésének, majd később párhuzamosan haladtam az applikációval és a kapcsolás összerakásával.

8.1 SQL-adatbázis

Első lépésként az Azure-ban létrehoztam egy adatbázist, majd utána megterveztem a tábláit.

Az adatbázis nagyon egyszerű, két táblából áll. Az egyik a regisztrált felhasználók felhasználónevét, jelszavát, illetve koordinátáit (szélességi kör és hosszúság), valamint a felhasználóhoz tartozó azonosítót, ennek az értéke automatikusan növekszik. A másik a felhasználók közötti kapcsolatot tárolja, a felhasználók azonosításához van két idegen kulcsa, ami a Users táblára referál. Az applikációban nincs implementálva, de az alábbi ábrán látható, hogy az adatbázis beceneveket is eltárol, vagyis a felhasználók képesek egymásnak becenevet adni. A tábla továbbá tartalmaz egy állapotjelző oszlopot, ami két értéket vehet fel: 1, ha a két felhasználó kapcsolata megerősített, azaz az egyik visszaigazolta a másik kérését, és 2, ha az egyik felhasználó elküldte a kérését, de arra még nem érkezett válasz. Ha a felhasználó elutasítja a kapcsolat kérését, akkor a táblából törlődik az adott rekord. Értelemszerűen egy felhasználó csak akkor képes lekérdezni a másik felhasználó pozícióját, ha az állapotjelző oszlopban 1-es érték szerepel.

Az adatbázisba felvettem pár teszt felhasználót, illetve azokat párba állítottam. Az egyes SQL lekérdezéseket az Azure felületén teszteltem, és megfelelő kimenet esetén az endpointba integráltam.



8.1. ábra – Az adatbázis táblái

8.2 Applikáció fejlesztésének folyamata

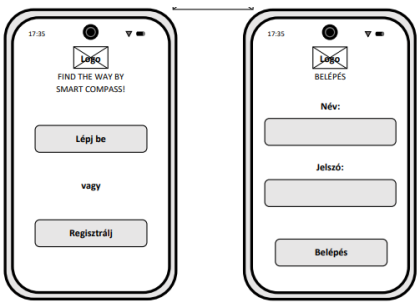
Az app egyes részeit, funkcióit külön solutionökben készítettem el, mivel így jóval gyorsabban tudtam haladni, illetve nem szemeteltem tele véletlenszerű kódokkal a projektet.

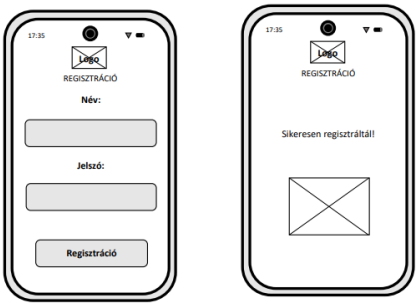
```
[assembly: UsesPermission(Android.Manifest.Permission.AccessCoarseLocation)]
[assembly: UsesPermission(Android.Manifest.Permission.AccessFineLocation)]
[assembly: UsesFeature("android.hardware.location", Required = false)]
[assembly: UsesFeature("android.hardware.location.gps", Required = false)]
[assembly: UsesFeature("android.hardware.location.network", Required = false)]
```

8.2. ábra – AssemblyInfóhoz fűzött sorok

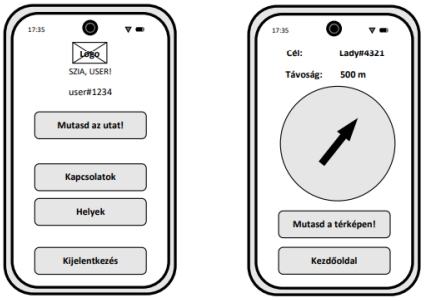
Először VS-ben tesztjelleggel csatlakoztam az Azure-os adatbázisra, majd miután ez sikerült, wireframe-eket készítettem a klienshez. Majd az AssemblyInfóhoz hozzá kellett adni pár sor kódot (8.2 ábra), melyek azért voltak szükségesek, hogy az alkalmazás hozzáférést kérhessen a mobil helyadataihoz [30]. A felhasználó helyadatait az alkalmazás jelenleg 30 másodpercenként automatikusan elküldi az adatbázis részére.

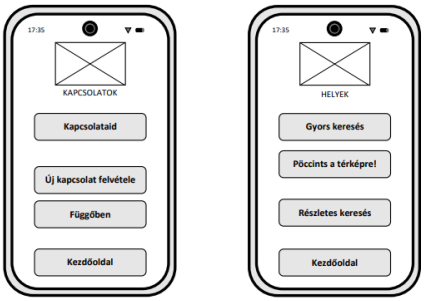
A 8.3 ábrán tulajdonképpen egy egyszerű bejelentkezési felület található. Ha felhasználó sikeresen bejelentkezett, akkor a következő felületre érkezik.





8.4. ábra – Regisztráció és bejelentkezés





8.3. ábra – Menü

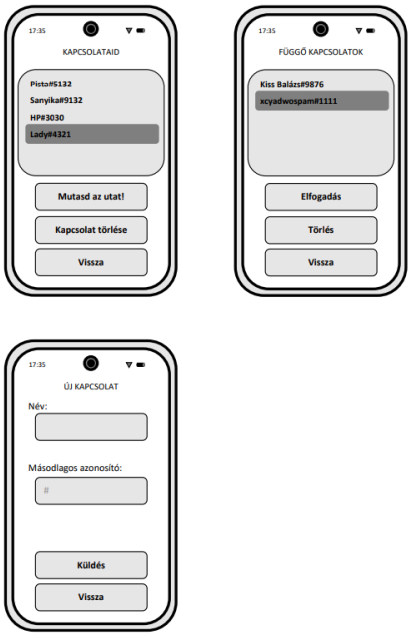
A menüben (8.4 ábra) a felhasználó három menüpont közül választhat: Kapcsolatok, Helyek, Mutasd az utat.

A Kapcsolataid menüpont alatt a más felhasználókkal való kapcsolatait tudja kezelni (8.5 ábra) (pl. új kapcsolat létrehozása, elfogadása, törlése), illetve itt tud célként megadni egy vele párban álló felhasználót is.

A Helyek menüpont alatt pedig háromféle mód áll rendelkezésre (8.6 ábra). Gyors keresés során a felhasználó egy szolgáltatás típust tud kiválasztani. A telefon 500 méteres rádiuszában történik keresés ebben az esetben. Ha van találat, akkor a legközelebbi szolgáltatóhoz irányítja a felhasználót, az alkalmazás átnavigál az irányító ablakra, és mutatja, merre kell mennie a felhasználónak, milyen messze van az adott hely, és hogy

mi a szolgáltató pontos neve. A Részletes keresés esetében is hasonló volt az elképzelés, azonban ez nem került implementálásra. Ebben a módban szintén megad a felhasználó egy adott szolgáltatást, illetve itt a keresés rádiuszát is ő maga állíthatja be. Itt viszont nem a legközelebbi szolgáltatás lesz automatikusan kiválasztva célként, hanem listázza a felhasználónak a találatokat. Tehát innen manuálisan tud kiválasztani a felhasználó egy adott szolgáltatás típusnak megfelelő szolgáltatót. Végül a Pöccints a térképre opció arra ad lehetőséget a felhasználónak, hogy a térképen felhasználó által megadott ponthoz irányítsa őt. Ebben az esetben az utca neve kerül kiírásra célként.

A wireframe-ek elkészítése után nekiláttam ezen felületek elkészítéséhez. Amikor az egyikkel elkészültem, annak leprogramozása is megtörtént, vagyis például a bejelentkezési lap elkészülésekor magát a bejelentkezést is megvalósítottam.



8.5. ábra – Kapcsolatok kezelése

8.6. ábra – Keresés helyekre

A bejelentkezés egyszerű, a felhasználónevet és a jelszót elküldi a kliens az endpointnak, ami ezen paraméterekkel küld el lekérdezést az adatbázis felé. Ha van találat, akkor a felhasználó sikeresen bejelentkezett, és tovább engedi a következő ablakra, ha nincs, akkor egy hibaüzenet jelenik meg a felhasználónak, mert nem sikerült a bejelentkezés.

A legközelebbi mérföldkő a kapcsolatok kezelése volt. Ha új kapcsolatot szeretne fölvenni a felhasználó, akkor be kell ütnie a másik felhasználó nevét. A wireframe-en egy négyjegyű azonosító is szerepelt (Discordhoz hasonlóan), ezt azonban elvetettem a későbbiekben. Ha létezik ilyen névvel felhasználó, akkor elküldi számára a kérést, ha nem, akkor egy üzenet jelzi, hogy nem létezik ilyen nevű felhasználó. Ha pedig már korábban küldött neki kérést, akkor a háttérben nem történik semmi, ennek ellenére ugyanúgy értesíti a felhasználót, hogy a kérés sikeresen ki lett küldve.

Törlés esetén a kliens kiküldi az endpointnak a saját és a másik felhasználó nevét, és ezek alapján történik törlés az adatbázisban. Akár egy már korábban elfogadott kapcsolat, akár egy új kérés törlése ugyanazt az utasítást hívja meg az endpointnál, vagyis az adatbázis Connections nevű táblájában található állapotjelző oszlop értékétől független a törlés módja.

Új kérés elfogadása esetén a kapcsolat állapotjelzője 2-es értékről 1-esre vált az adatbázisban. Innentől kezdve a felhasználó le tudja kérdezni a másik eszköz pozícióját, és így kiszámíthatóvá válik annak iránya.

Ahhoz, hogy a kliens csatlakozni tudjon az endpointhoz, szükség volt REST API-ra. Ennek alapját Kovács András kódja adta [31], ezen minimális módosítások történtek. Az endpointot a kliens alkalmazással párhuzamosan fejlesztettem, hiszen szükség volt arra, hogy az egyes funkciók tesztelhetők legyenek.

A virtuális iránytű oldalon található kis nyíl forgatása a telefonban található magnetométer segítségével történik [32]. Először megkapja azt, hogy milyen szögben kellene állnia ahhoz, hogy a cél felé mutasson, majd utána ezt a szöget a magnetométer jele szerint módosítja. A kezdeti szöget Forward Azimuth képlettel számolja ki [33] (8.7 ábra), a távolság pedig Haversine formula alapján kerül kiszámításra [34] (8.8 ábra).

```
private static double HeadingCalculator(Location user, Location destination)
{
    double toRad = Math.PI / 180.0;
    double deltaLongitude = destination.Longitude - user.Longitude;
    double x = Math.Cos(destination.Latitude * toRad) * Math.Sin(deltaLongitude * toRad);
    double y = Math.Cos(user.Latitude * toRad) * Math.Sin(destination.Latitude * toRad) - Math.Sin(user.Latitude * toRad)
        * Math.Cos(destination.Latitude * toRad) * Math.Cos(deltaLongitude * toRad);
    double rad = Math.Atan2(x, y);
    double degree = rad * (180 / Math.PI);
    return degree;
}
```

8.7. ábra – Forward Azimuth implementálása

```
private static double DistanceCalculator(Location user, Location destination)
{
    int R = 6371; // metres
    double latitude1 = user.Latitude * Math.PI / 180;
    double latitude2 = destination.Latitude * Math.PI / 180;
    double deltaLatitude = (destination.Latitude - user.Latitude) * Math.PI / 180;
    double deltaLongitude = (destination.Longitude - user.Longitude) * Math.PI / 180;

    double a = Math.Sin(deltaLatitude / 2) * Math.Sin(deltaLatitude / 2) +
               Math.Cos(latitude1) * Math.Cos(latitude2) *
               Math.Sin(deltaLongitude / 2) * Math.Sin(deltaLongitude / 2);
    double c = 2 * Math.Atan2(Math.Sqrt(a), Math.Sqrt(1 - a));

    double distanceInMetres = R * c; // in metres

    return distanceInMetres;
}
```

8.8. ábra – A Haversine formula implementálása

Legutoljára pedig a Google térkép került implementálásra. Ez a funkció teszi lehetővé, hogy a térképre pöccintve egy adott pont legyen célként meghatározva. Szerencsére ehhez is nagyon jó dokumentáció állt rendelkezésre [35]. Ahhoz, hogy használni tudjuk a

Google Maps API-t, egy API kulcsot kellett igényelni a Google-től, majd az AndroidManifest.xml fájl egy sorához hozzá kellett azt adni. Ennek implementálása egyelőre csak egy szimpla térképet ad. Ahhoz, hogy jelölőket is le tudjunk tenni, Pin objektumokat kell létrehozni [36], ezek egyébként személyre szabhatók [37]. A térképet úgy implementáltam, hogy megnyitásakor (ha még nincs) engedélyt kér a helyadatokhoz a felhasználótól, és ha azt megkapja, a térkép a felhasználó pozíciójára ugrik, a felhasználót a térképen is mutatja [38]. Ha a felhasználó a térkép egy pontján rákoppint, akkor megjelenik egy jelölő. Ha a jelölőre koppint, akkor megjelenik az utca neve. Ha pedig a térkép alatt található gombra pöccint, akkor ez a megadott hely lesz az új cél, és átnavigál a virtuális iránytű felületére.

A virtuális iránytű oldalán a fejlesztés végén bekerült egy gomb, amivel ki tudjuk küldeni az irányt a NodeMCU-nak. Ahhoz, hogy ezt megtegyük, először rá kell csatlakozni Wi-Fin keresztül a NodeMCU-ra, majd rányomni a gombra. Az alkalmazás ekkor automatikusan elküldi a megfelelő IP címre az adatot. Ezután visszacsatlakozhatunk a korábbi hálózatunkra. Ez meglehetősen körülményes, de mivel a BLE modul implementálása sikertelen volt, ezért ez egy olyan ideiglenes megoldás, ami lehetővé tette a kommunikáció, illetve a LED-ek iránytól függő vezérlésének tesztelését.

8.3 Hardver összerakásának folyamata

Első lépésként a LED gyűrűvel foglalkoztam. Először egy virtuális környezetben (tinkercad.com) szimuláltam a működését (16.1 melléklet) [39]. Sajnos az oldal más komponenssel nem rendelkezett, így csak ezt tudtam letesztelni ilyen módon. Ezután kivettem a dobozából, és elkezdtem a forrasztást. Kicsit aggódtam, hogy esetleg elrontom, forrasztásban kevés tapasztalatom volt, és a forrasztómnak a hegye is tönkrement már valamennyire, úgyhogy meglehetősen nehéz volt a feladat, de szerencsére sikerült, és a másik LED gyűrűt nem is kellett felhasználnom. Az Arduinóra való rákapcsolás után először csak az volt a cél, hogy egy LED világítson rajta. A LED-eknek szinte vakító fényük volt, szerencsére ennek beállításához is volt parancs. Miután ez sikerült, következhetett a magnetométer.

Arra, hogy képes legyen a magnetométer működni, egy egész napot rááldoztam. A modult nem kellett forrasztani, voltak megfelelő lyukai, azokhoz jártak rövid jumperek is, így könnyedén csatlakoztatni tudtam az Arduinóhoz. A magnetométer jelét soros monitoron figyeltem, viszont akárhogy forgattam, az érték nem változott. Azt hittem, a kóddal van probléma, rengeteg idő elment annak módosítgatásával, de nem jutottam eredményre. Közben a kapcsolást is folyamatosan variáltam, nem tudtam rájönni, hol lehet a hiba. Gondoltam arra is, hogy esetleg a jumperek nem érintkeznek megfelelően a modullal, de hiába próbáltam mozgatni, hiába cseréltem le a kábeleket, nem történt semmi. Végül, amikor már nem volt más ötletem, még intenzívebben mozgattam a jumpereket. Kiderült, hogy mégiscsak az okozta a problémát, hogy nem érintkeztek megfelelően. Úgyhogy

végül ezt is muszáj volt megforrasztani, szerencsére ennek a forrasztása is sikeres volt, és onnantól kezdve módosultak az adatok az eszköz forgatásakor.

A BLE modullal volt a legtöbb probléma, ezzel ment el a legtöbb idő, amit sajnos végül nem is tudtam megoldani. Az eszközről olyan forrásokat találtam, ahol Arduino kommunikált egy másik Arduinóval úgy, hogy mindkettőnek volt egy-egy ilyen BLE modulja, illetve találtam olyat is, ahol a BLE modul küld üzenetet a mobiltelefonnak [40]. Az én célom az volt, hogy a mobil tudjon küldeni adatokat a BLE egységnek [41], viszont ennek megvalósítása jóval bonyolultabb. Létezik hozzá applikáció egyébként [42], viszont ennek használata is nehézkes. A rendelt nRF24L01-es modul helyett HC-05-ös modullal egyszerűbb az ilyen funkciók megvalósítása, erről az egységről is rengeteg forrás található, azonban ez az egység drágább. Hozzáteszem, az Android emulátor nem képes bluetooth-os eszközre kapcsolódni, így ennek tesztelésekor rendes mobilt használtam.

Mivel a BLE modullal nem sikerült kapcsolatot teremteni a telefonon keresztül, ezért átmeneti megoldást kellett találni a problémára. Ez a megoldás lett a NodeMCU, ami mint Access Point [43] viselkedett a rendszerben. Találtam hozzá egy hasznos forrást [44], illetve a GitHub is a segítségemre volt [45]. Ezek alapján sikerült megvalósítani a mobil és a NodeMCU közötti kommunikációt. A mobilon meg kell keresni az elérhető Wi-Fik közül a NodeMCU-t, és csatlakozni kell rá. Ezután egy bizonyos IP címre HTTP requestet lehet küldeni, melynek tartalmaznia kell a cél irányát (double típusú adat). Ezt a modul megkapja, majd elküldi az Arduinónak. Az Arduino és a NodeMCU között soros kommunikáció történik. Az Arduino feldolgozza a bejövő adatot, majd kiszámolja, hogy melyik LED-nek kell égnie. Ez ugye attól is függ, hogy a magnetométer éppen milyen irányba néz.

A 16.1 mellékleten látható azon kódrészlet, ami kiszámítja, hogy épp melyik LED-nek kell égnie. Amelyik ég, abba az irányba kell fordulnia a felhasználónak ahhoz, hogy a cél irányába haladjon. Az eljárás két paramétert vár. A heading azt a szöget tartalmazza, amekkora szögben el kellene indulni ahhoz, hogy elérjük a célunkat (észak = 0 fok). A telefon -180 és +180 közötti értéket küld az Arduinónak, vagyis a heading paraméter egy -180 és 180 közötti érték lesz, viszont a magnetométer egy 0 és 360 közötti értékben határozza meg, hogy mekkora szöget zár be északkal (ha észak felé néz, akkor 0 értéket ad, ha dél felé, akkor 180-at, ha nyugat felé, akkor 270-et stb.). Éppen ezért, ha a heading változóban negatív érték szerepel, akkor azt át kell alakítani úgy, hogy a 360-hoz hozzáadjuk ezt a negatív értéket (tehát valójában kivonjuk a 360-ból a heading értékét). Az offset változó azt mondja meg, hogy amennyiben észak felé nézünk, úgy melyik LED-nek kellene világítania. A north változó azt mondja meg, hogy melyik LED-nek kellene világítania ahhoz, hogy észak felé mutasson.

A jobb érthetőség érdekében itt egy példa:

Ha a heading -90, akkor az azt jelenti, hogy balra kell fordulnunk. Ezt átváltjuk, vagyis 270 fokban kell elfordulnunk jobbra. A LED gyűrű 16 LED-ből áll, vagyis ahhoz, hogy a bal oldali LED világítson, ezt a 270-es értéket el kell osztanunk 22.5-lel (mert 360 / 22.5 = 16, ezzel az osztással tehát minden LED-et lefedtünk). Az osztás után kapjuk, hogy 12, vagyis a 12-es indexű LED-nek kellene felgyulladnia, ez a 12-es érték kerül az offset változóba. Ha az északi irányba nézünk, akkor a north változóban nulla érték fog szerepelni, így a 12-es indexű, vagyis a bal oldali LED fel fog gyulladni.

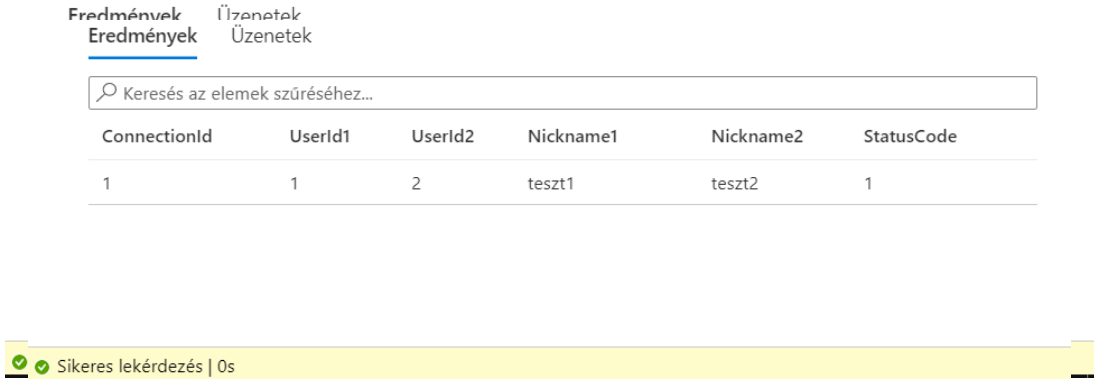
Viszont ha már nem észak felé nézünk, akkor a north változóban is egy 0-tól különböző szám lesz. A north változóban ugyanilyen logikával kerül be egy index, viszont -1-gyel szorozzuk. A -1-gyel való szorzásra azért van szükség, mert ha az óramutató járásával megegyező irányban fordulok, akkor az eddig világító LED-től bal oldalra található LED-nek kell felkapcsolnia, vagyis mindig ellenkező irányban gyulladnak fel a LED-ek, mint ahogyan fordulunk.

Ha az előző példát vesszük, akkor amennyiben jobbra fordulok, úgy a north értéke 4 lesz (90 / 22.5 = 4, jobb oldali LED indexe). Ezt az értéket megszorozzuk -1-gyel, és hozzáadjuk az offsethez, vagyis 12 + (-4) = 8. A 8-as érték pedig pont az alsó LED indexe, vagyis ha balra van a cél, de mi jobbra fordulunk, akkor mögöttünk lesz a cél, és emiatt az alsó LED fog világítani.

Előfordulhat azonban, hogy a north értéke -16. Ilyenkor 12 + (-16) = -4, ilyen indexű LED viszont nincsen. Ha negatív indexet kapunk, akkor az tulajdonképpen annyit jelent, mintha tettünk volna egy fordulatot. Vagyis 16-ból (a 0. indexű LED) ki kell vonni 4-et, így megkapjuk a 12-őt újra, ami a 12-es indexű LED-et fogja felkapcsolni. Másik eshetőség, mikor 15-nél nagyobb értéket kapunk. Ekkor is ugyanaz a szituáció, csak ilyenkor ebből a 15-nél nagyobb értékből kell levonnunk 16-ot, és így megkapjuk a megfelelő LED indexét.

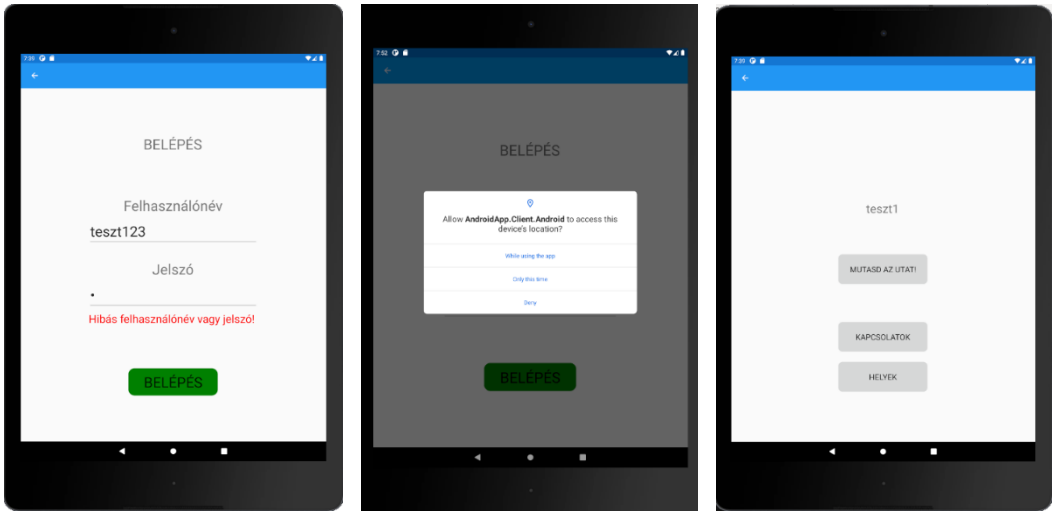
8.4 Applikáció tesztelése

Az adatbázisban három teszt felhasználó található (8.10 ábra). Ezek közül a „teszt1” és a „teszt2” áll kapcsolatban egymással (8.9 ábra).



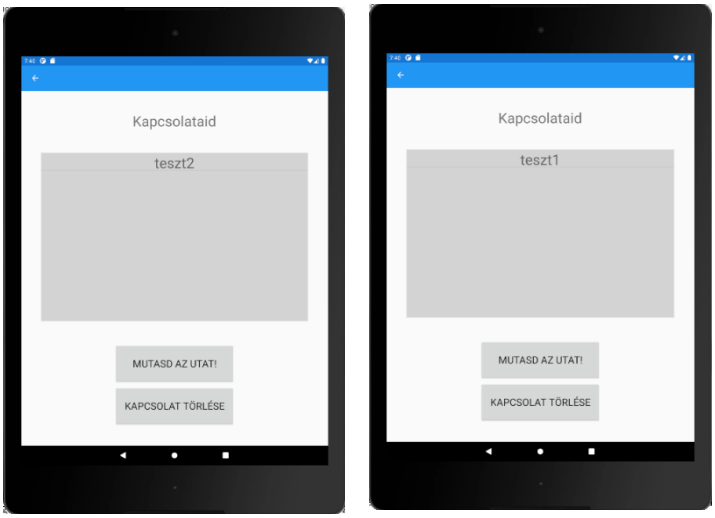
8.9. ábra – Teszt kapcsolatok a felhasználók között (Azure SQL adatbázis)

A bejelentkezés folyamán rossz jelszó (vagy felhasználó) esetén megjelenik egy hibaüzenet, megfelelő adatok megadásakor pedig engedélyt kér, hogy hozzáférhessen az eszköz pozíciójához, majd beenged a menübe (8.11 ábra).

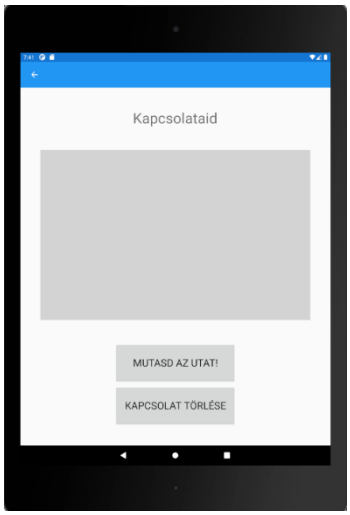


8.11. ábra – Bejelentkezés

Teszt1 nevű felhasználónál töröljük a Teszt2-vel való kapcsolatát. Teszt2 kitörlődött Teszt1 barátai közül, Teszt2 fiókjába átlépve pedig látjuk, hogy már Teszt1 sem létezik nála elérhető kapcsolatként (8.13 és 8.13 ábra).

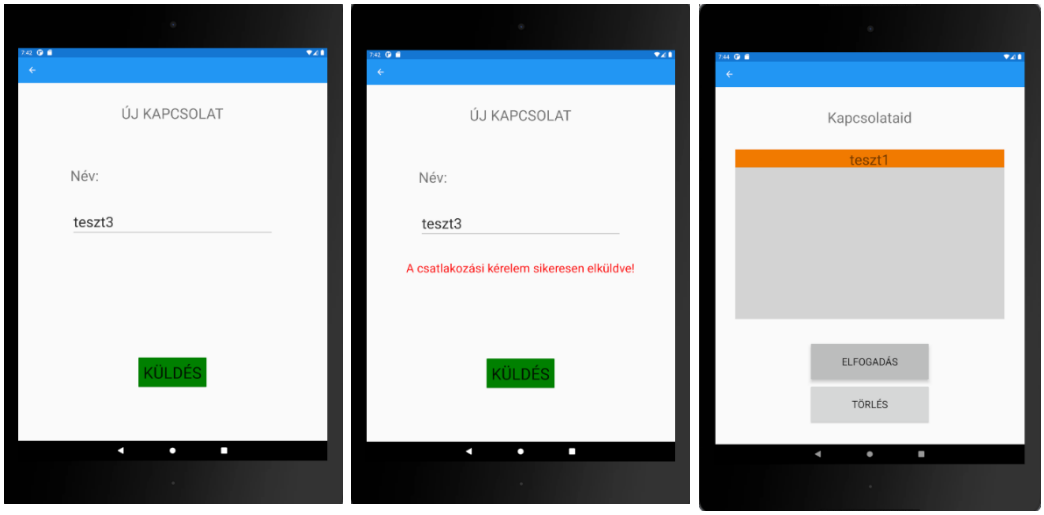


8.13. ábra – Teszt1 (bal) és Teszt2 (jobb) kapcsolatai



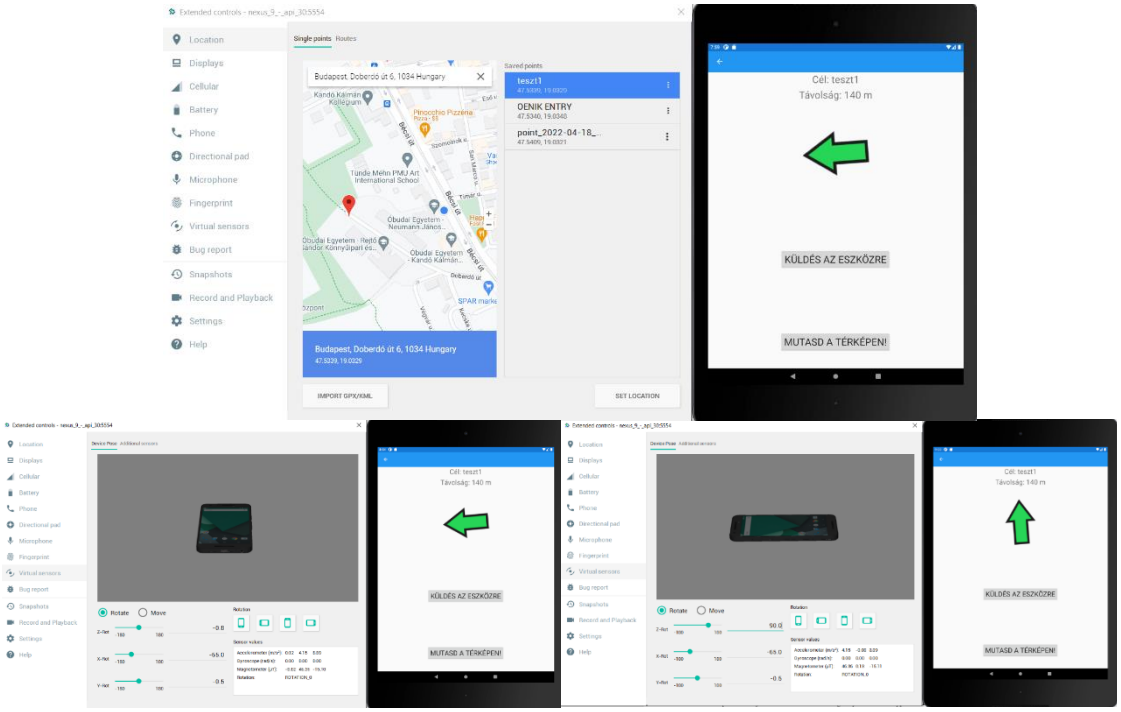
8.13. ábra – Teszt2 törlése után

Teszt1 fiókjából küldjünk kérelmet Teszt3 számára, majd Teszt3 fiókjába átlépve a „Függőben lévő” kapcsolatoknál Teszt1 kérelmét fogadjuk el (8.15 ábra).



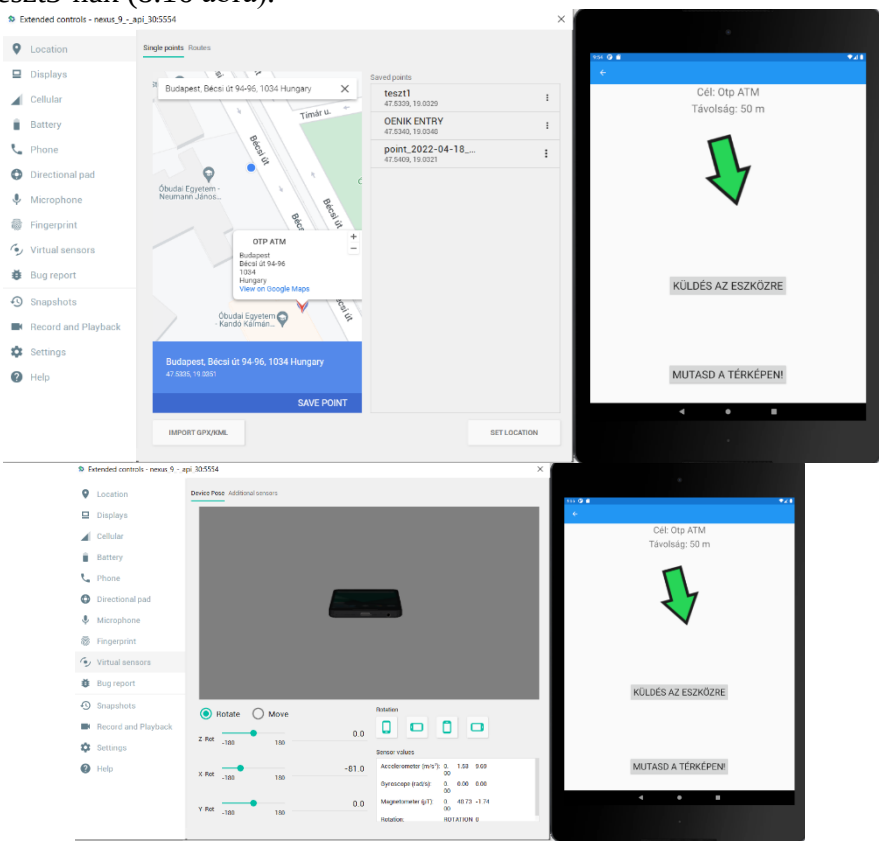
8.15. ábra – Új kapcsolódási kérelem küldése és elfogadása

Elfogadás után Teszt1 átkerült az elérhető kapcsolatok listájába. Állítsuk be Teszt1-et célként. Teszt1 nyugatra van Teszt3-tól, a virtuális iránytű ennek megfelelően balra mutat. Ha a szimulált Android eszközt forgatjuk, akkor a nyíl is fordul, és tartja az irányt Teszt1 felé (8.14 ábra).



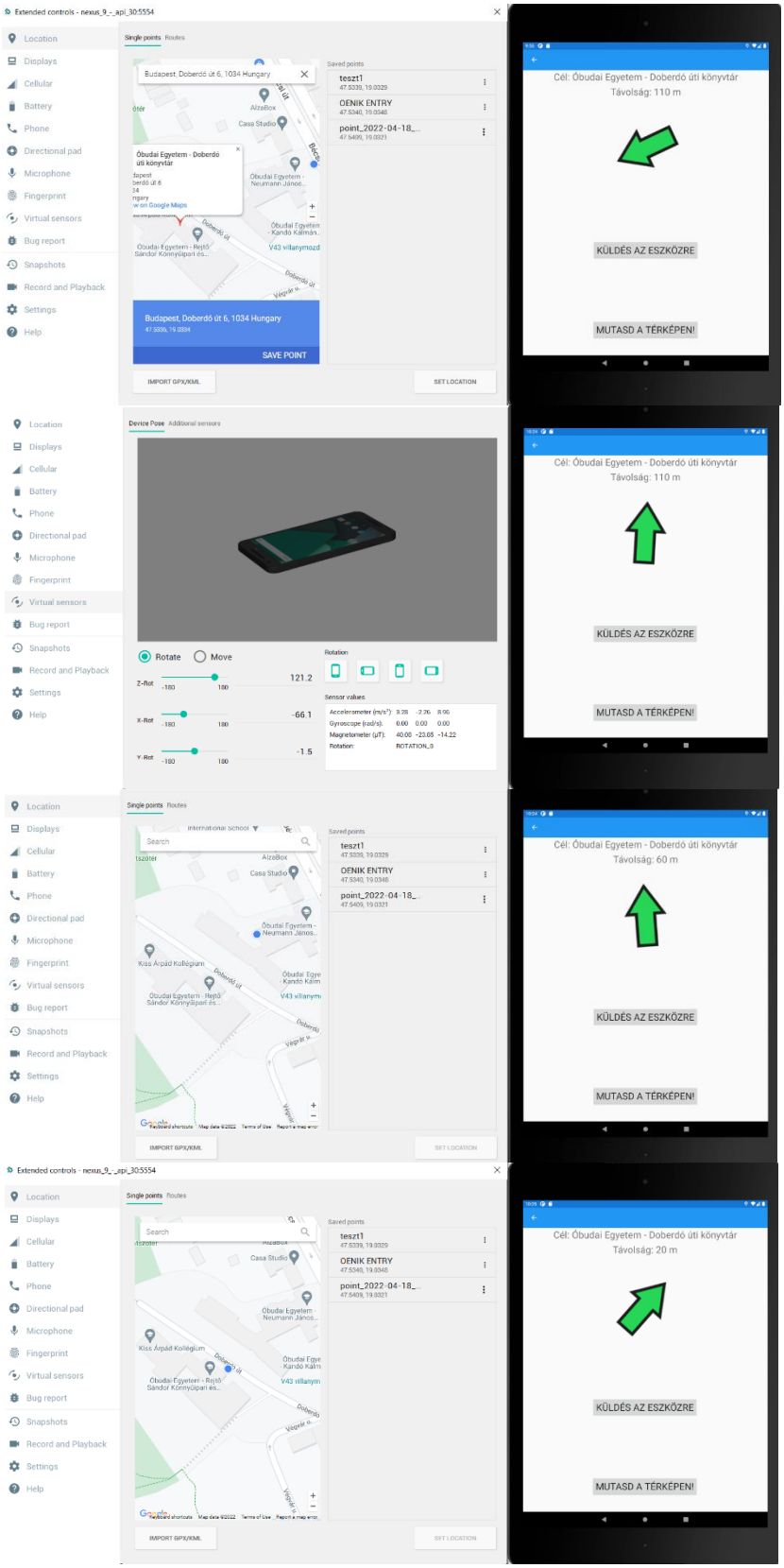
8.14. ábra – Térképen piros jelölő jelzi Teszt1 pozícióját, a nyíl iránya a telefon orientáltságától függ

Most a gyors keresés funkciót használva keressünk Teszt3 közelében valamilyen szolgáltatást. Az applikációban jelenleg választhatunk, hogy könyvtárat, kávézót, éttermet, boltot, ATM-et vagy esetleg egy ügyvédet keresünk a közelünkben. Keressünk most egy ATM-et. Amint látjuk, be is jött a navigációs ablak, ami mutatja, merre kellene mennie Teszt3-nak (8.16 ábra).



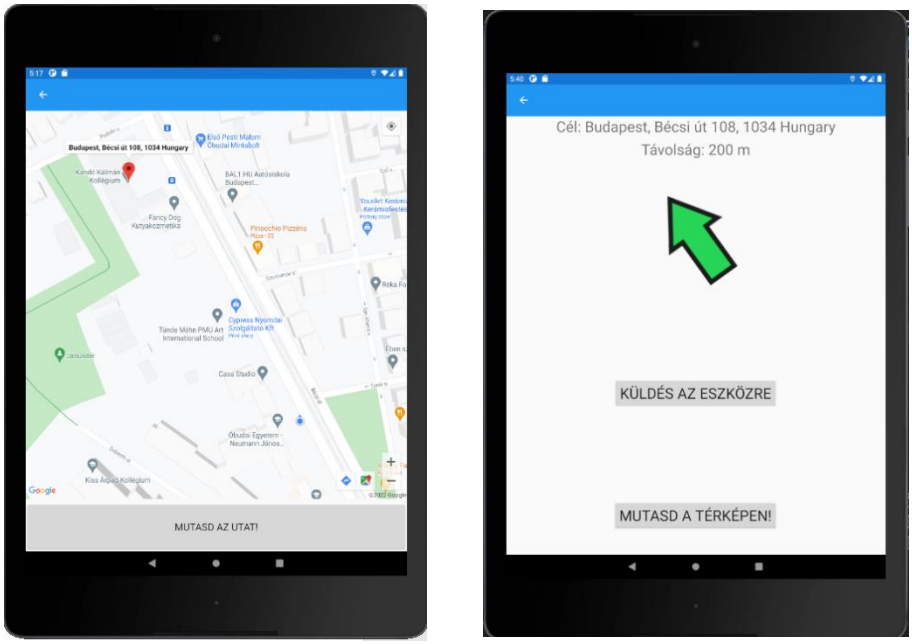
8.16. ábra – Gyors keresés funkció (ATM a közelben)

Egy újabb gyors kereséssel megtalálhatjuk a Doberdó úti könyvtárat. Ennél a példánál (8.17 ábra) az is látható, hogy ahogy Teszt3 elindul a könyvtár irányába, úgy a távolság is csökken. Először a telefon az eddigi példáknál is megszokott irányba néz, vagyis északra (X tengelye -65° körüli) Ezután elforgatjuk Z tengely mentén balra, hogy a telefon teteje mutasson a cél felé. Innentől csak a telefon pozíciója módosul, az orientációja nem. Ahogy közeledünk a könyvtárhoz, egyre kisebb a távolság, a nyíl pedig folyamatosan mutatja, hogy az aktuális pozíciónkhoz képest merre található a könyvtár azonos orientáltság mellett. Ahogy korábban láthattuk, az orientáció is befolyásolja a nyíl irányát, ezért nem tartottam indokoltnak, hogy ennél a tesztnél az orientációt is módosítsam.



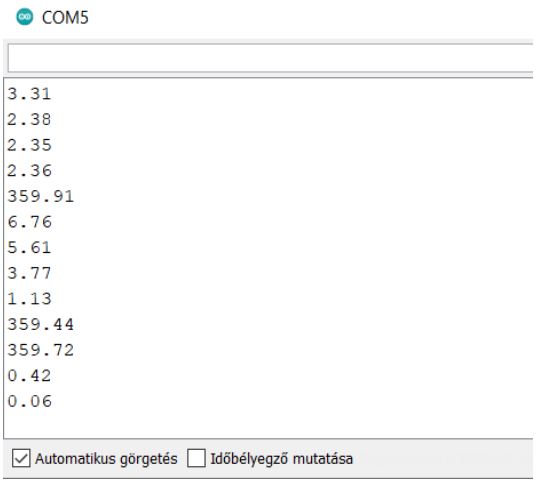
8.17. ábra – Könyvtár felé haladva frissül az irány és a távolság

A részletes keresés nem került implementálásra, helyette nézzük meg, hogyan működik a Pöccints a térképre! funkció. Mivel korábban már engedélyt adtunk az applikációnak, hogy lekérdezhesse a pozíciónkat, ezért a térképet megnyitva rögtön Teszt3 pozíciójára ugrik, de egyébként erre a Google API-ja is lehetőséget ad a felhasználó számára a térkép jobb felső sarkában található gombbal. Pöccintsünk a térképen a Kandó Kálmán Kollégium környékére. Ha a megjelenő piros jelölőre újra pöccintünk, kiírja a hely címét. Ez a cím jelenik meg akkor is, ha kiválasztjuk úticélként. Kiválasztás után megjelenik a virtuális iránytű, ami az eddigi példákkal azonos módon navigálja el a felhasználót a kollégium területére.

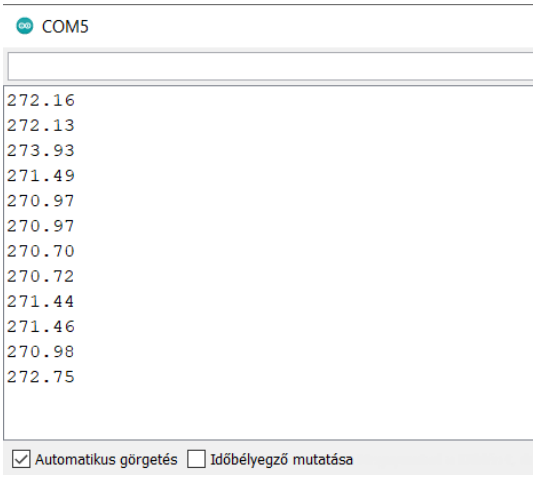


8.18. ábra – Pöccints a térképre mód

8.5 Hardver tesztelése



8.19. ábra – A szenzor kimenete, ha észak felé fordítjuk

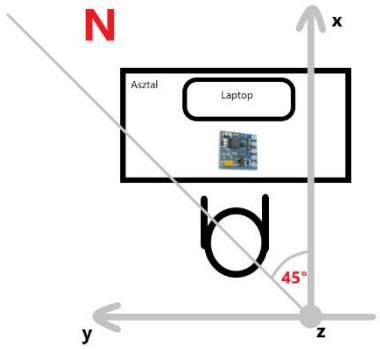


8.20. ábra – A szenzor kimenete, ha nyugat felé fordítjuk

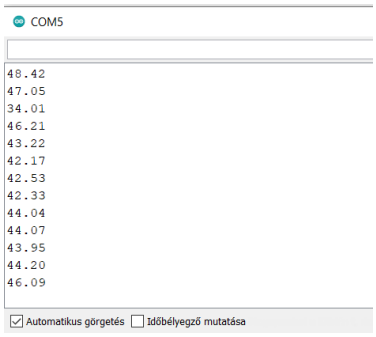
A magnetométer által küldött jeleket soros monitoron vizsgáltam. Ha a szenzort észak felé fordítjuk, akkor a kapott érték 0, illetve 360 érték körüli (8.19 ábra). Ha a szenzort nyugatra fordítjuk, akkor 270 körüli értéket kapunk (8.20 ábra). Tőlem jelenleg kb. 40 fokkal balra fordulva található észak (8.22 ábra), ez abból a kimenetből látható, amit az érzékelő akkor eredményez, amikor velem egy irányba néz (8.22 ábra).

A korábban részletezett kód egyes változóinak értékét vizsgálva (minden példában 2 mérés található a soros monitoron, egy mérés az orientációtól az index változóig tart) láthatjuk, hogy megfelelő eredményeket kapunk, és a megfelelő LED-ek világítanak az egyes esetekben.

Legyen a célunk észak (8.23 ábra), ebben az esetben a gyűrűnek mindig azon LED-jének kell világítania, amelyik északra mutatna. Tehát a cél észak, mi pedig nézzünk előre. Ahhoz, hogy belássuk, valóban jó eredményt kapunk, egy iránytűt is elhelyeztem az eszköz mellé. Látható, hogy amennyiben a szenzor előre néz, a 14-es indexű LED gyullad ki, és ez az eredmény elfogadható. Van némi hiba, az igazi iránytű által és a LED által mutatott irány nem esik egybe, azonban ezt okozhatják az egyes változók értékeinek kerekítései, az igazi iránytű pontatlansága vagy a szenzort zavaró elektromos eszközök. Mivel a LED



8.22. ábra – Észak tőlem 40°-kal balra fordulva található

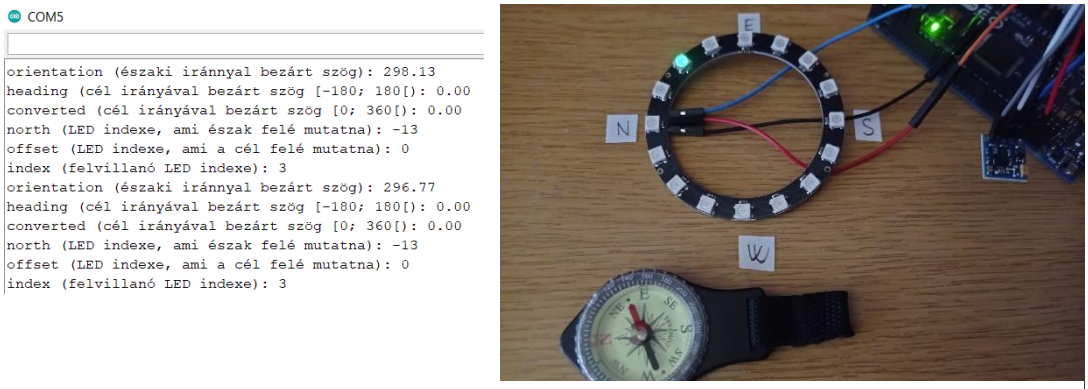


8.22. ábra – Magnetométer kimenete, ha előre néz

gyűrű és a szenzor nincs egymáshoz rögzítve, ezért valószínűleg ebből fakad a legtöbb hiba, de belátható, hogy a rendszer megfelelő pontossággal működik.

8.23. ábra – Előre nézünk, a cél pedig észak

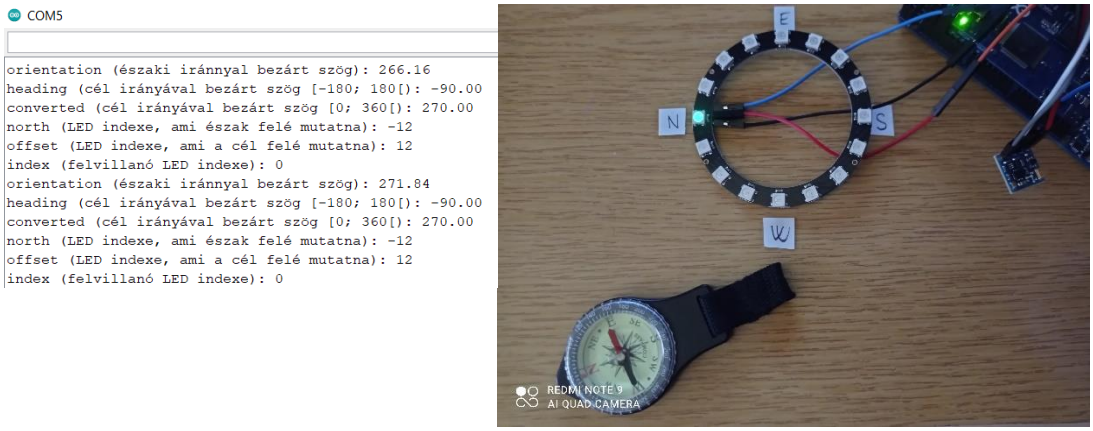
Ha mindent (az iránytűt, a LED gyűrűt és a szenzort) elforgatunk 90°-kal balra, akkor is a megfelelő, azaz az északi irányba mutató LED ég (8.24 ábra).



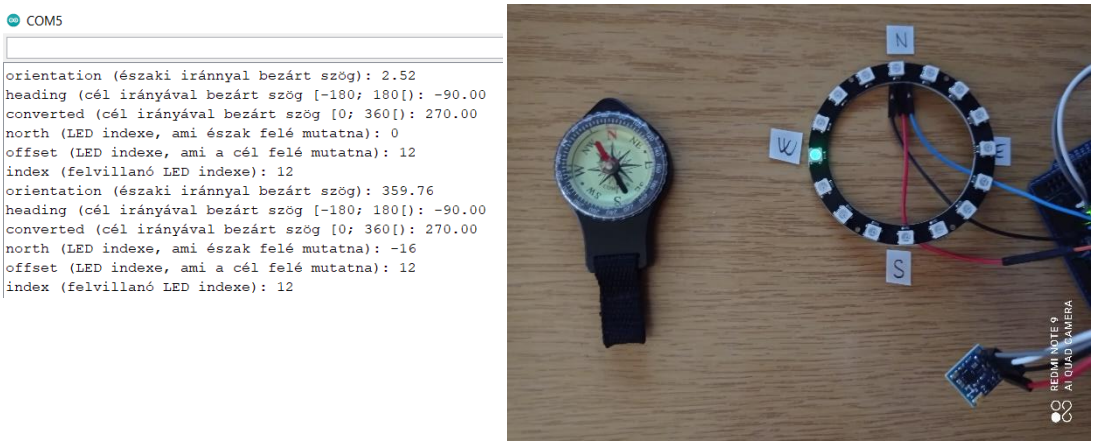
8.24. ábra – Balra nézünk, a cél pedig észak

Legyen a cél észak helyett nyugat, és most nézzünk északra. A kép kicsit csalóka, ugyanis a LED gyűrű nem lett északi irányba elforgatva, csak az iránytű és a szenzor, viszont láthatjuk, hogy a nyugati jelölésű LED gyullad fel, vagyis az elvárt kimenetet kaptuk (8.25 ábra).

Utolsó tesztként pedig nézzünk nyugat felé, a célunk pedig legyen nyugat. Itt is helyes kapott eredmény. Azt láthatjuk, hogy amennyiben nyugat felé nézünk és nyugat felé van a célunk, akkor előre kell mennünk (8.26 ábra).



8.26. ábra – Nyugatra nézünk, a cél pedig nyugat

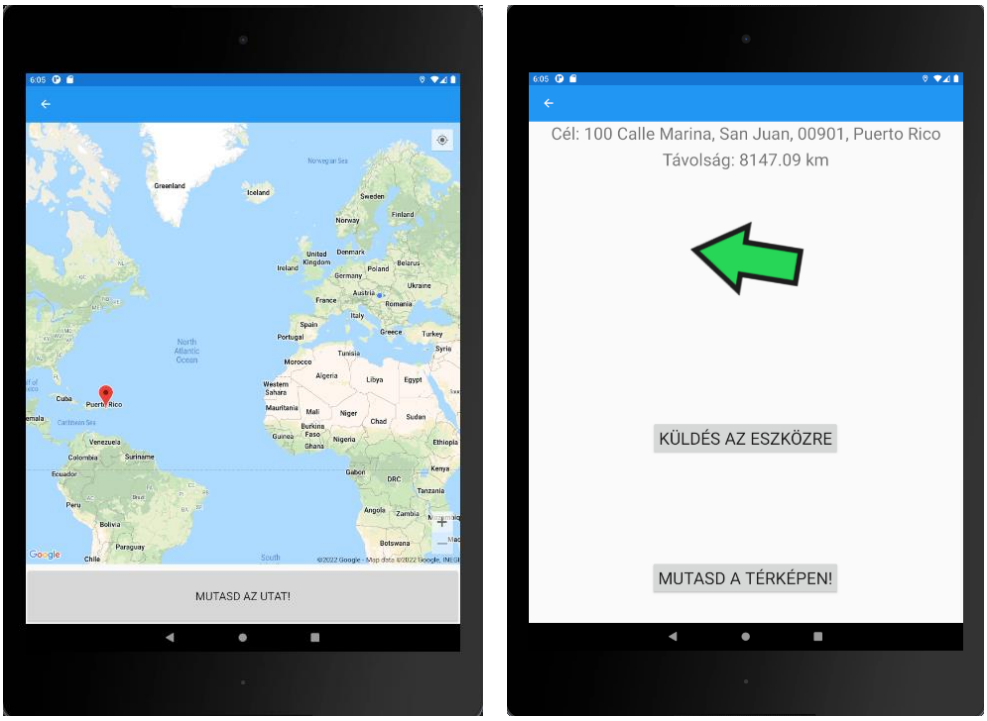


8.25. ábra – Északra nézünk, a cél pedig nyugat

8.6 Applikáció és hardver együttes tesztelése

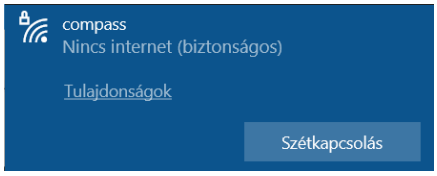
Láthattuk, hogy az applikáció és az iránytű külön-külön működnek. Beláthatjuk, hogy amennyiben a két eszköz között a kommunikáció sikeres, úgy a virtuális és a valóságos iránytű is ugyanabba az irányba fog jelezni.

Tegyük fel, hogy a korábbi példákhoz hasonlóan most is az Óbudai Egyetem a kiindulási pont, és szeretnénk San Juanba (Puerto Rico) utazni. Látható, hogy a virtuális iránytű kissé pontatlan (8.27 ábra). Ez ilyen nagy távolságban azért lehetséges, mert az Android emulátor virtuális magnetométerének paraméterein nem történt módosítás.



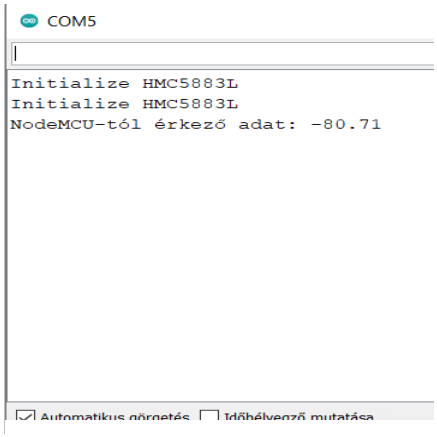
8.27. ábra – Nagyobb távolság esetén kisebb hiba az Android emulátoron

Csatlakozzunk rá a NodeMCU-ra (8.30 ábra), majd nyomjunk rá a „Küldés az eszközre” gombra (ezután le is csatlakozhatunk az eszközről).

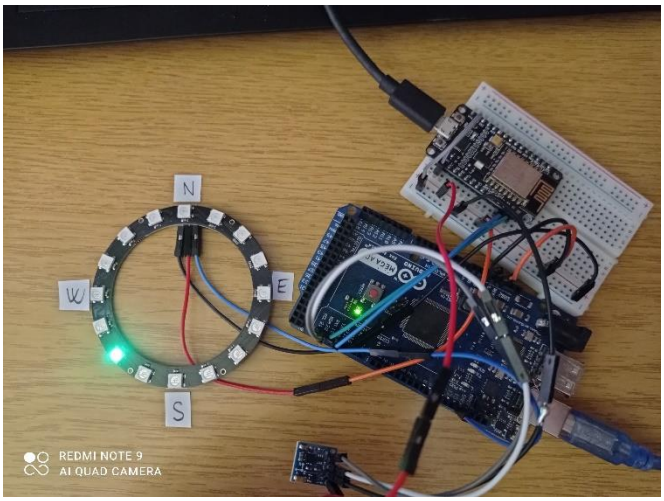


8.30. ábra – Csatlakozás a NodeMCU-ra

A soros monitoron látjuk, hogy megérkezett a szög, amerre a cél található. Ez a szög független a mágneses deklinációtól, mivel kizárólag a koordináták alapján kerül kiszámításra. Így tehát ha megnézzük a hardvert, mivel annak kódjában a mágneses deklináció meg lett adva [46], láthatjuk, hogy megfelelő irányba jelez (8.28 ábra).



8.29. ábra – NodeMCU-tól megérkezett az adat az Arduinóra



8.28. ábra – San Juan iránya

8.7 Fejlesztés közben felmerült problémák és azok megoldásai

A fejlesztés korai szakaszában egyszerre volt fent a laptopon a Visual Studio 2019 Enterprise és a Visual Studio 2022 Community verziója. Nem vagyok biztos abban, hogy ez okozta, de rengetegszer kaptam véletlenszerűen egy hibát a fordítótól, ha módosítottam valamit a kódban. A hiba szerint a fordító nem találta a Xamarin.Android.CSharp.targets nevű fájlt egy bizonyos elérési útvonalon. A fájl tényleg nem létezett, de még a mappák sem, amiben keresgélt. Néha sikerült debugolni, néha nem. Egy megoldást végül találtam egy microsoftos fórumon, de ez sem segített mindig [47].

Egyébként a teljes fejlesztés borzasztóan lassan haladt: minden debugolás alkalmával el kellett indítania a laptopomnak az endpointot, miközben futott rajta az Android emulátor, amibe szintén be kellett töltenie az alkalmazást, emellett pedig ott futott a háttérben a Google Chrome-ban megnyitott száznál több oldal. A laptopom 8 GB RAM-ja nem volt elég, rengetegszer lefagyott, rengeteget kellett várni, igaz, hozzá kell tenni, hogy már ráfér egy újratelepítés. Megoldás lehetett volna erre a problémára az, ha nem Android emulátort használok debugolásra, hanem a saját készülékemet, de a saját készülékem sem egy mai darab.

A hardverrel kapcsolatban már említettem, hogy a magnetométerhez nem illeszkedtek rendesen a jumperek, azért forrasztani kellett. A tesztelések folyamán észrevettem, hogy a modul túl közel van a mikrokontrollerhez, illetve a laptophoz is, és ezek miatt az elektromos eszközök miatt sokszor megzavarodott, rossz értéket küldött az Arduinónak. Hogy ezt megoldjam, újabb jumperekkel egészítettem ki a már meglévőket, hogy távolabb tudjam vinni a modult ezektől a zavaró tényezőktől, viszont az utolsó jumper annyira makacs volt, hogy először elhajlott, majd végül eltört a jumper tűje. Úgyhogy még egy utolsó forrasztásra szükség volt, de ez már csak másnap történt meg.

9 Biztonsági kockázat

Az iránytű és az alkalmazás jelenleg nem használható nyomkövetésre, azonban a jövőben elképzelhető, hogy az egyes célok (hely vagy felhasználó) térképen is megtekinthetők lesznek. Ha a felhasználó pozícióját a lehető legpontosabban kéri le az alkalmazás, úgy abban az esetben előfordulhatnak visszaélések. Ugyanis bár az applikációban egy felhasználónak vissza kell igazolnia a másik felhasználót ahhoz, hogy az lássa, merre és milyen távolságra található a másik, elképzelhető, hogy valakinél két telefon van, és esetleg az egyiket elrejtheti valahol, így megfigyelve bármit és bárkit. Jelen állapotában viszont ténylegesen két telefonra van szüksége az „elkövetőnek”, mivel a telefon helyadataival dolgozik a rendszer. Abban az esetben, ha a későbbiekben GPS modul is beépítésre kerül, úgy már elegendő, ha csak magát az iránytűt helyezi el adott helyen. Erre a problémára megoldást jelenthet, ha valamilyen módon értesíti a rendszer azt a felhasználót, akinek a közelében ilyen idegen eszköz található huzamosabb ideig hasonlóan ahhoz, ahogyan azt az Apple tervezi a saját készülékével [48].

10 A termék árának becslése

Ahhoz, hogy képet kapjunk arról, hogy a jövőben kb. milyen áron lehet eladható a termék, érdemes összehasonlítani a különböző okosórák és aktivitásmérők tulajdonságait, illetve azok árait. Az egyes tulajdonságok (X, mint bemenet) eredményezik a termék árát (Y, mint kimenet). Az adatgyűjtés nem terjedt ki a szakdolgozat elején ismertetett okosiránytűkre, a minta kiselemszámú, valamint egyes termékek esetében az információk hiányosak, mert az azokkal kapcsolatos adatok titkosak, vagy csak nagyon nehezen hozzáférhetők, valamint előfordulhat pontatlanság is az egyes források által közzétett adatok különbözősége miatt. Az adatokat, megállapításokat tehát fenntartásokkal kell kezelni, ennek ellenére jó kiindulópont lehet a későbbiekben, ha azt szeretnénk megállapítani, érdemes-e a terméket piacra dobni.

A kérdésre való választ, miszerint milyen áron lehetne eladni egy ilyen okosiránytűt, Solverrel adtam meg. Az egyes eredményeket a Magyar Internetes Agrár / Alkalmazott Informatikai Újság (MIAU) oldalán [49] található MY-X Factor-Y FREE szolgáltatásának termelési függvény becslő eljárást használva kaptam.

10.1 Vizsgált eszközök

Azért ezeket az eszközöket választottam ki, mert a legtöbb információ a termékekről könnyen hozzáférhető, az egyes árak között pedig nagyobb különbségek vannak, amitől tisztább képet várok az adatok elemzése után.

A vizsgált eszközök és áraik (forintban):

Termék neve	Termék ára (2021-es legkisebb átlagárak az arukereso.hu adatai alapján)
Amazfit Band 5	12961
Amazfit Bip	19980
Apple Watch SE	113187
Apple Watch SE + Cellular	139589
Apple Watch Series 3	80594
Apple Watch Series 6	163668
Fitbit Charge 4	51875
Fitbit Inspire 2	36232
Fitbit Sense	118114
Fitbit Versa 3	80913
Garmin Vivosmart 4	35990

Samsung Galaxy Watch 3	107629
Xiaomi Mi Band 4	10616

Először összeszedtem az általam fontosnak vélt tulajdonságokat („Adatok v1” munkalap), amiket úgy véltem, befolyásolhatják egy termék árát, illetve magukat az árakat is („Árak (Y)” munkalap). Összesen 39 ilyen tulajdonságot írtam fel. Ezeket egy Excel táblázatba importáltam, a sorok tartalmazzák az egyes termékeket, az oszlopok pedig a tulajdonságokat. Ezután feltöltöttem a táblázatot nyers adatokkal, amelyik termék tulajdonságára nem találtam információt, ott azt egy „Nincs adat” szöveggel jeleztem.

Miután összeszedtem minden elérhető információt az egyes márkák oldalairól, a hiányos cellákat (ahol nem találtam adatot az adott paraméterre vonatkozóan) is feltöltöttem valamilyen adattal, hogy a Solver tudjon velük is dolgozni. Szintén emiatt arra is szükség volt, hogy az egyes celláknak (amikben akár szöveges értékek is lehettek) olyan számértéket adjak, ami után össze tudjuk hasonlítani két termék jóságát. Megadtam, hogy az egyes oszlopoknál a nagyobb vagy a kisebb érték számít-e „jobbnak”. Egy érték akkor „jobb” egy másiknál, ha az jobban növeli a termék árát. Erre jó példa az „anyaszervezet” paraméter. Például a Fitbit anyaszervezete a Google. Ezt az információt úgy konvertáltam számértékké, hogy az anyaszervezet piaci értékét vettem. Az elgondolás e mögött az, hogy egy cég drágábban tudja eladni ugyanazt a terméket népszerűtlenebb társánál, egy cég népszerűségére pedig következtetni lehet a szervezet piaci értékéből. Minél nagyobb ez az érték, annál népszerűbb.

A következő feladat az volt, hogy az egyes cellákban található értékeket rangsoroljuk („Sorrend v1 (X)” munkalap). Az adott oszlopban tehát a legrosszabban teljesítő termék a 13-as értéket kaphatta meg, a legjobban teljesítő pedig az 1-est.

A termékek árai („Árak (Y)” munkalap) tulajdonképpen ezeknek a tulajdonságoknak a kimenetei. El kellett dönteni, hogy pontosan mely éveket vesszük figyelembe. Végül a 2020-as és 2021-es árakat gyűjtöttem ki, mivel a legtöbb általam vizsgált termék áráról csak 2020. óta léteznek adatok (legalábbis az árukereső oldalán). Az árukereső minimum árakat, legkisebb és legnagyobb átlagárakat jelenít meg, egy termékhez mindhárom árát kigyűjtöttem.

Ezután a bemeneti adatokat (X) és a kimeneti adatokat (Y) egy munkalapra gyűjtöttem („X + Y v1” munkalap). Ez úgy történt, hogy mivel 2020 és 2021-es kimeneti adatok álltak rendelkezésre, ezért a sorok, vagyis a bemeneti jelek megduplázódtak: egy sorhoz (termékhez) tartozik egy 2020-as és egy 2021-es kimenet. Vagyis egy ilyen táblázatnak van 13 * 2 sora és 38 + 1 oszlopa (38 paraméter (X), 1 kimenet (Y)). Ebből a táblázatból pedig három készült el, mivel azt akartam megtudni, hogy a termékek bizonyos paraméterei mennyiben befolyásolják a minimum árat, a legkisebb és legnagyobb átlagárat.

Ezt a három táblázatot egyesével megadtam a termelési függvénynek, majd az eredményeket átmásoltam egy külön munkalapra („COCO Solver v1” munkalap). A kimenet alapján elmondható, hogy a 38 paraméter közül az árra ezek a funkciók és tulajdonságok hatnak:

- női ciklus figyelő,
- akkumulátor kapacitás,
- mennyi idő alatt merül le (autonómia),
- töltési idő,
- a kijelző mérete,
- az anyaszervezet piaci értéke,
- magasságmérő,
- Wi-Fi,
- a kijelző felbontása,
- sport mód funkció,
- memória nagysága,
- hangszóró megléte,
- NFC kommunikáció támogatása,
- a tömeg,
- légnyomásmérő,
- illetve a Google Trend népszerűsége.

Bizonyos funkciók azért nem játszhatnak szerepet az árban (pl. a lépésszámláló funkció), mert a vizsgált termékek mindegyikében megtalálható. Emiatt feltételezhetjük, hogy az eredmény valamennyire pontatlan, mivel, ha változatosabb paraméterekkel rendelkező termékeket sorakoztatunk fel egymás mellett, lehetséges, hogy a lépésszámláló funkció is hatással lenne a kimenetre. A vizsgálatot tehát pontosabbá tudjuk tenni, ha több, egymástól különbözőbb terméket veszünk fel a táblázatok soraiba.

Az egyes funkciók értelemszerűen növelik egy termék árát. Ami számomra meglepő volt, az a tömeg árra való kihatása. A jelenlegi adatok alapján azt mondhatjuk, hogy minél könnyebb egy ilyen eszköz, annál drágább. Mondhatnánk, hogy azért könnyebb egy termék, mert kevesebb hardver van benne, ebben az esetben egyértelműen hibás lenne a kapott eredmény, hiszen, ha kevesebb a hardver, akkor kevesebb a funkció, ezáltal olcsóbbnak kellene lennie egy terméknek, nem pedig drágábbnak. A válasz az, hogy ha egy termék könnyű, akkor az drágább alapanyagokból, drágább feldolgozásból, megmunkálásból készült el pontosan azért, hogy a felhasználót ne zavarja az eszköz a tömege.

A becsült árak és a tényleges árak között a legnagyobb különbség a Fitbit Sense és a Fitbit Versa 3 termékeinél fordult elő mindhárom táblázatban kb. 20.000 Ft különbség van (delta oszlop) a tényleges és a becsült ár között, egyes esetekben 20.000 Ft-tal lehetne olcsóbb, illetve más esetekben pedig pont, hogy 20.000 Ft-tal lehetne drágább is adott

évben adott paraméterekkel. Ahol tehát a delta oszlopban található érték pozitív, az a termék abban az évben lehetne olcsóbb is.

Ahhoz, hogy meghatározzuk, hogy egy bizonyos paraméterekkel rendelkező termék milyen áron lehetne eladható (ami nem egy pontos, csak egy becsült érték), nincs más dolgunk, mint a termék egyes paraméterei alapján eldönteni, hogy az adott paraméterrel hányadik lépcsőn helyezkedik el, majd a megadott lépcsőknek a cellaértékeit összeadni, és ennek az összege lesz a termék becsült ára.

Mivel nem ismert az okosirányú összes paramétere, ezért a legtöbb cellában csak megközelítő értéket írtam. A bizonyos fokig véletlenszerűen megadott paraméterek alapján egyedül a kis tömege adott hozzá a minimum árához, aminek a becsült összege így 3936 Forint, vagyis minimum kb. 4000 Forint lenne a termék minimum ára a piacon.

Mivel kevés a felvett objektum, és a paramétereik is sokszor megegyezőek, azt lehet mondani, hogy jóval pontosabb becslést kaphatunk, ha több terméket több szempontból vizsgálunk meg.

11 Továbbfejlesztési lehetőségek

A Beeline, mint kiderült, remek referenciapont a rendszer későbbi funkcióit tekintve. A route nevű navigációs módjával lehetne bővíteni a szoftvert, a hozzátartozó pánt nagyon jó megoldásnak tűnik, könnyedén felszerelhető biciklik kormányaira. Amennyiben az iránytű több funkciót kap, úgy fizikai vagy kapacitív érintőgombokkal, ennek megfelelően pedig digitális kijelzővel is fel lehetne szerelni.

SignalR implementálása esetén a felhasználó értesítéseket kaphatna afelől, hogy érkezett-e új kapcsolat kérelme, így nem kellene manuálisan elvégeznie ezeket a lekérdezéseket (vagyis jelenleg újra meg kellene nyitnia az adott menüpontot a lekérdezéshez).

Az iránytű egy további változata kaphatna akár GPS és Wi-Fi modult is, így akár a telefontól teljesen függetlenül is működhetne.

12 Összefoglalás

Az eszköz jelenlegi verziója a Beeline-féle compass módot implementálja. Az alkalmazás a mobil GPS-koordinátáit magától a telefontól kapja meg, a cél GPS-koordinátáit pedig vagy az Azure vagy pedig a Google szerveréről kérdezi le. A kapott koordináták alapján kiszámítja az irányt, a LED gyűrűn pedig felkapcsolja a megfelelő LED-et figyelembe véve azt is, hogy az eszköz éppen milyen irányba néz. Ennek meghatározásához volt szükség a magnetométer szenzorja.

Az elsődleges cél az, hogy egy olyan termék jöjjön létre, ami csak az elengedhetetlen, működéséhez szükséges komponenseket tartalmazza, ilyen módon lehetővé téve a termék minél olcsóbb előállítását. A termék lehetséges jövőbeli ára a Beeline-féle okosiránytűk, illetve más okosórák és aktivitásmérők árai alapján kerül a jövőben kiszámításra. A számításhoz szükséges modell már létezik, az adatok, kapott eredmények elemzése még tart, illetve az már bizonyos, hogy a jelenlegi 13 vizsgált objektumnál több terméket kell beletenni a modellbe ahhoz, hogy az árakat képes legyen a Solver minél jobban megbecsülni. A modell arra fog majd választ adni, hogy a termék forgalomba helyezése esetén milyen minimum áron, minimum átlag- és maximum átlagáron lesz eladható. Amennyiben ezek az értékek nagyobbak, mint az előállítás (illetve az összköltség) értéke, úgy nagyobb bizonyossággal lesz kijelenthető, hogy a terméket megéri sorozatgyártásba fogni. Értelemszerűen a modell nem tud pontos választ adni a termék áráról, ahogyan arról sem, hogy mennyire lesz népszerű, mekkora lesz rá a kereslet, tehát a bizonytalanság bizonyos mértékben továbbra is megmarad majd.

13 Summary

The device's current version implements the Beeline-kind compass. The application receives the smartphone's GPS coordinates, and the destination's coordinates are provided by either Azure or Google's servers. Given the coordinates, the app calculates the poper direction and on the LED ring it switches on the appropriate light, also considering which way the device is facing. In order to carry out the calculations properly I needed a magnetometer sensor.

The primary goal is to create a device that only uses the essential components that are required to operate, facilitating the cheapest production possible. The product's possible future price will be calculated depending on the Beeline-kind smart compasses' and other smart watches and band's prices. The modell needed for this calculation already exists, the data and the received results are still bein analized, and I am sure now that I need to extend the list of the currently stuidied 13 objects with more products in the modell in order to make the Solver estimate more accurately. The modell will reveal the amount of the minimum price, the minimum average and maximum average price that the device could be sold on when put on market. In case these values are lower than the production cost (and total cost), then we could dedicate to starting serial production more confidently. Obviously the modell cannot give a real life accurate answer about the deveise's price, and cannot predict exactly its popularity and the market's interests, so the uncertanity will never vanish entirely.

14 Referencialista

[1] Arduino team, „Find the best bar with a smart DIY compass!”, *Arduino Blog*, 2016. május 3. https://blog.arduino.cc/2016/05/03/find-the-best-bar-with-a-smart-diy-compass/ (elérés 2021. május 13.).

[2] Arduino team, „Pilgrim képe”. https://blog.arduino.cc/wp-content/uploads/2016/05/1cover-1024x768-e1462259398634.png (elérés 2021. május 13.).

[3] The James Dyson Award, „Haize, the smart compass”, *James Dyson Award*. https://www.jamesdysonaward.org/en-NZ/2016/project/haize-the-smart-compass/ (elérés 2021. május 10.).

[4] onomo, „HAIZE - minimalist urban bike navigation”, *Kickstarter*. https://www.kickstarter.com/projects/onomo/haize-a-compass-reinvented-navigation-for-urban-cy (elérés 2021. május 10.).

[5] Coolthings, „HAIZE képe”. https://netdna.coolthings.com/wp-content/uploads/2015/11/haize-1.jpg (elérés 2021. május 10.).

[6] Beeline, „Bike Computer Beeline Velo | Beeline”. https://beeline.co/pages/beeline-velo (elérés 2021. május 10.).

[7] Beeline, „Motorcycle sat nav | Beeline”. https://beeline.co/pages/beeline-moto (elérés 2021. május 10.).

[8] Beeline, „Beeline Velo képe”. https://cdn.shopify.com/s/files/1/1897/8919/products/device_thumbnail_grey_600x.jpg?v=1574095888 (elérés 2021. május 13.).

[9] Beeline, „Beeline Moto képe”. https://cdn.shopify.com/s/files/1/1897/8919/products/Beeline-Moto-PR072019-19_600x.jpg?v=1568207890 (elérés 2021. május 13.).

[10] S. Olivera *és mtsai.*, „Plating on acrylonitrile–butadiene–styrene (ABS) plastic: a review”, *J. Mater. Sci.*, köt. 51, ápr. 2016, doi: 10.1007/s10853-015-9668-7.

[11] P. Alves, J. F. J. Coelho, J. Haack, A. Rota, A. Bruinink, és M. H. Gil, „Surface modification and characterization of thermoplastic polyurethane”, *Eur. Polym. J.*, köt. 45, sz. 5, o. 1412–1419, máj. 2009, doi: 10.1016/j.eurpolymj.2009.02.011.

[12] Wikipédia, „Bluetooth”, *Wikipédia*. 2020. május 3. Elérés: 2021. május 14. [Online]. Elérhető: https://hu.wikipedia.org/w/index.php?title=Bluetooth&oldid=22588257

[13] A. Nguyen, „Bluetooth 1.0 vs 2.0 vs 3.0 vs 4.0 vs 5.0 - How They Compare | Symmetry Blog”. https://www.semiconductorstore.com/blog/2018/Bluetooth-1-0-vs-2-0-vs-3-0-vs-4-0-vs-5-0-How-They-Differ-Symmetry-Blog/3147/ (elérés 2021. május 14.).

[14] M. Collotta, G. Pau, T. Talty, és O. K. Tonguz, „Bluetooth 5: a concrete step forward towards the IoT”, *ArXiv171100257 Cs*, nov. 2017, Elérés: 2021. május 14. [Online]. Elérhető: http://arxiv.org/abs/1711.00257

[15] Laptopakkumulator.eu, „Lithium-polymer akkumulátorokról szakszerűen - Laptopakkumul”. https://laptopakkumulator.eu/lithium_polymer_akkumulatorok (elérés 2021. május 14.).

[16] „Battery Statistics – Battery University”. https://batteryuniversity.com/learn/archive/battery_statistics (elérés 2021. május 14.).

[17] GSMArena, „Sensors - definition - GSMArena.com”. https://www.gsmarena.com/glossary.php3?term=sensors (elérés 2021. május 14.).

[18] „IP védettség táblázat és a védelmi fokozatok jelentése.” https://megaohm.hu/erintesvedelem/ip-vedettseg (elérés 2021. május 13.).

[19] „GPX”, *Wikipédia*. 2021. május 9. Elérés: 2021. május 14. [Online]. Elérhető: https://hu.wikipedia.org/w/index.php?title=GPX&oldid=23853491

[20] GPX, „GPX: the GPS Exchange Format”. https://www.topografix.com/gpx.asp (elérés 2021. május 14.).

[21] „android-11-cdd.pdf”. Elérés: 2021. május 10. [Online]. Elérhető: https://source.android.com/compatibility/11/android-11-cdd.pdf

[22] „Azimuth”, *Wikipedia*. 2021. április 13. Elérés: 2021. május 13. [Online]. Elérhető: https://en.wikipedia.org/w/index.php?title=Azimuth&oldid=1017628287

[23] H. Kang és J. Cho, „Case Study on Efficient Android Programming Education using Multi Android Development Tools”, *Indian J. Sci. Technol.*, köt. 8, sz. 19, aug. 2015, doi: 10.17485/ijst/2015/v8i19/75984.

[24] A. Industries, „NeoPixel Ring - 16 x 5050 RGB LED with Integrated Drivers”. https://www.adafruit.com/product/1463 (elérés 2022. május 9.).

[25] A. Industries, „Adafruit NeoPixel Library: Adafruit_NeoPixel Class Reference”. https://adafruit.github.io/Adafruit_NeoPixel/html/class_adafruit___neo_pixel.html (elérés 2022. május 9.).

[26] Sertronics Marketing, „NeoPixel Ring mit Arduino ansteuern: So wird’s gemacht”, *BerryBase Blog*, 2020. november 9. https://www.blog.berrybase.de/blog/2020/11/09/neopixel-ring-mit-arduino-ansteuern-so-wirds-gemacht/ (elérés 2022. május 9.).

[27] K. Jarzebski, „How to wire HMC5883L - Magnetometer (Compass) to Arduino Mega”. https://www.circuito.io/app?components=9442,11061,1234567 (elérés 2022. május 9.).

[28] „Maps SDK for Android – APIs & Services – Szakdoga – Google Cloud Platform”. https://console.cloud.google.com/apis/library/maps-android-backend.googleapis.com?project=szakdoga-345414 (elérés 2022. május 9.).

[29] „Google Maps Platform Documentation | Places API | Google Developers”. https://developers.google.com/maps/documentation/places/web-service (elérés 2022. május 9.).

[30] jamesmontemagno, „Xamarin.Essentials: Geolocation - Xamarin”. https://docs.microsoft.com/en-us/xamarin/essentials/geolocation (elérés 2022. március 27.).

[31] Kovács A., „Diasor és kódok letöltése | Nik Programming”. https://nikprog.hu/courses/halado-fejlesztesi-technikak/lecke/diasor-es-kodok-letoltese-11/ (elérés 2022. május 9.).

[32] jamesmontemagno, „Xamarin.Essentials: Compass - Xamarin”. https://docs.microsoft.com/en-us/xamarin/essentials/compass (elérés 2022. május 10.).

[33] C. Veness, „Calculate distance and bearing between two Latitude/Longitude points using haversine formula in JavaScript”. https://www.movable-type.co.uk/scripts/latlong.html (elérés 2022. május 10.).

[34] M. Nichat, „Landmark based shortest path detection by using A* Algorithm and Haversine Formula”, ápr. 2013.

[35] davidbritch, „Xamarin.Forms Map Initialization and Configuration - Xamarin”. https://docs.microsoft.com/en-us/xamarin/xamarin-forms/user-interface/map/setup (elérés 2022. május 10.).

[36] davidbritch, „Xamarin.Forms Map Pins - Xamarin”. https://docs.microsoft.com/en-us/xamarin/xamarin-forms/user-interface/map/pins (elérés 2022. május 10.).

[37] davidbritch, „Customizing a Map Pin - Xamarin”. https://docs.microsoft.com/en-us/xamarin/xamarin-forms/app-fundamentals/custom-renderer/map-pin (elérés 2022. május 10.).

[38] dotnet-bot, „Map.IsShowingUser Property (Xamarin.Forms.Maps)”. https://docs.microsoft.com/en-us/dotnet/api/xamarin.forms.maps.map.isshowinguser (elérés 2022. május 10.).

[39] S. D. School Science and Computer Science Teacher at Divine Child High, „8.1 – Introduction to NeoPixels”, *High School Maker*, 2017. március 20. http://www.highschoolmaker.com/electronics-with-micro-controllers/chapter-8-neopixels/8-1-introduction-to-neopixels/ (elérés 2022. május 10.).

[40] P. Abhiemanyu, „Sending Sensor Data to Android Phone using Arduino and NRF24L01 over Bluetooth (BLE)”. https://circuitdigest.com/microcontroller-projects/sending-sensor-data-to-android-phone-using-arduino-nrf24l01-over-bluetooth-ble (elérés 2022. május 9.).

[41] A. Parajuli, „NRF24L01 as BLE Module with Arduino”, *The IOT Projects*, 2021. június 29. https://theiotprojects.com/nrf24l01-as-ble-module-with-arduino/ (elérés 2022. május 9.).

[42] Nordic Semiconductor ASA, „nRF Connect Device Manager – Alkalmazások a Google Playen”. https://play.google.com/store/apps/details?id=no.nordicsemi.android.nrfconnectdevicemanager&hl=hu&gl=US (elérés 2022. május 9.).

[43] ESP8266 Arduino Core, „Soft Access Point — ESP8266 Arduino Core documentation”. https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/soft-access-point-examples.html (elérés 2022. május 9.).

[44] Random nerd tutorials, „ESP8266 Client-Server Wi-Fi Communication Between Two Boards (NodeMCU) | Random Nerd Tutorials”, 2020. január 9. https://randomnerdtutorials.com/esp8266-nodemcu-client-server-wi-fi/ (elérés 2022. május 9.).

[45] „ESPAsyncWebServer/examples at master · me-no-dev/ESPAsyncWebServer”, *GitHub*. https://github.com/me-no-dev/ESPAsyncWebServer (elérés 2022. május 9.).

[46] N. G. D. Center, „NCEI Geomagnetic Calculators”. https://www.ngdc.noaa.gov/geomag/calculators/magcalc.shtml (elérés 2022. május 12.).

[47] „who to resolve the problem of this.....?????” https://social.msdn.microsoft.com/Forums/en-US/7609f6f7-11b5-4d38-b080-79e5f5001304/who-to-resolve-the-problem-of-this?forum=xamarincrossplatform (elérés 2022. május 9.).

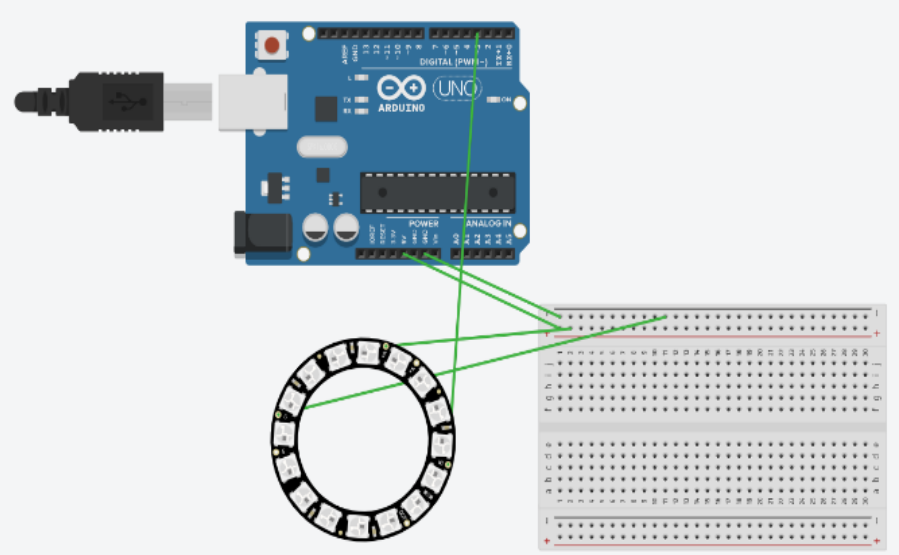
[48] Zrt H. K., „Lépett az Apple, biztonságosabb lehet a bármit követő és megtaláló AirTag”, *hvg.hu*, 2022. május 8. https://hvg.hu/tudomany/20220508_apple_airtag_biztonsagi_frissites (elérés 2022. május 10.).

[49] „MIAU 1998-2022”. https://miau.my-x.hu/miau2009/index.php3 (elérés 2022. május 13.).

15 Ábrajegyzék

16 Mellékletek

16.1 Forráskódok




```
#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
 #include <avr/power.h>
#endif

// Which pin on the Arduino is connected to the NeoPixels?
#define PIN 3

double analogPin = A0;

// How many NeoPixels are attached to the Arduino?
#define NUMPIXELS 16

double state = 0;
Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

#define DELAYVAL 1000 // Time (in milliseconds) to pause between pixels

void setup()
{
  Serial.begin(9600);
#if defined(__AVR_ATtiny85__) && (F_CPU == 16000000)
  clock_prescale_set(clock_div_1);
#endif


  pixels.begin(); // INITIALIZE NeoPixel strip object (REQUIRED)
}

void loop()
{
  state = analogRead(analogPin);
  Serial.println(state);
  pixels.setPixelColor(1, pixels.Color(0, 0, 150));  //blue
  pixels.show();   // Send the updated pixel colors to the hardware.
  delay(DELAYVAL); // Pause before next pass through loop
}
```

16.1. ábra – LED gyűrű szimulációja (tinkercad)

```
void setPixel(float heading, double orientation){
  pixels.clear();
  float converted = heading;
  if (heading < 0){
    converted = 360 + heading;
  }

  int offset = round(converted / 22.5);
  int north = -1 * round(orientation / 22.5);
  int index = offset + north;
  if (index < 0){
    index = 16 + index;
  }
  else if (index > 15){
    index = index - 16;
  }
  pixels.setPixelColor(index, pixels.Color(0, 150, 0)); // zöld
  pixels.setBrightness(1);
  pixels.show();   // Pixelek frissítése
```

16.2. ábra – Megfelelő LED indexének kiszámítása (kódrészlet)