Eötvös Loránd Tudományegyetem

Informatikai Kar

Programozáselmélet és Szoftvertechnológia Tanszék

Modellvasút terepasztal vezérlése szenzorok és mikrokontrollerek alkalmazásával

Témavezető:

Dr. Gludovátz Attila

adjunktus, Ph.D.

Szerző:

Petrovics Bence

Programtervező informatikus BSc.

Szombathely, 2021

Hivatalos és elfogadott Szakdolgozat témabejelentő

Az ipar fejlődésével, egyre több helyen kezd nélkülözhetetlenné válni a különböző informatikai eszközök jelenléte és az informatikai adatfeldolgozás. Egyik legfőbb feladat az eszközök, gépek vagy akár teljes gyártósorok rendszereinek irányítása. Ezeket a feladatokat általában a gyártósorokon elhelyezett PLC egységek végzik, azonban hasonló feladatokkal felruházható egy mikrokontroller is.

Szakdolgozatom célja egy olyan vezérlőprogram elkészítése, amely képes szenzorokból érkező adatok alapján egy modellvasúti terepasztal irányítására, vezérlésére, ezzel modellezve akár egy ipari rendszer működését. Hiszen a logika hasonló, szenzorok adatai alapján a szoftver döntést hoz, majd beavatkozik az adott folyamat végrehajtásába. Távolabbi jövőben pedig az itt szerzett tudásaimat szeretném hasznosítani.

A szakdolgozat a következőket tartalmazza:

1. **tervezési folyamat**
    a. szenzorkörnyezet és beavatkozó egységek elhelyezése bekötése.
    b. a rendszert leíró diagramok rajzolása
2. **Szoftver implementáció**
    a. **Automata vezérlés:**

        Automata vezérlési módban a felhasználói beavatkozás nem lehetséges oly módon, ami a modellek mozgását befolyásolja (pl. váltóállítás), mindaddig, amíg az automatikus végrehajtás le nem állt vagy le nem lett állítva. Ezen üzemmód legfőbb követelménye, hogy a valódi, nagyvasúti hálózatnak megfelelően mindenféle ütközést, balesetet el kell kerülnie.

    b. **Kézi vezérlés:**

        Kézi üzemmódban minden váltó, jelző szabadon állítható és minden vonat szabadon indítható is, kizárólag a felhasználón múlik a helyes vezérlés.

    c. **Világítás vezérlés:**

        Ebben a módban az előző két modul aktuális állapotától függetlenül lehetséges vezérelni az elhelyezett tereptárgyak fényeit.

3. Dokumentáció

A vezérlő szoftver Linux alatt, java nyelven kerül implementálásra egy Raspberry Pi-on, továbbá USB kapcsolattal segédeszközként Arduino csatlakozik a Raspberryhez, hogy minden szenzor és beavatkozó vezérlése biztosított legyen. A fent említett modulok megfelelő működéséhez párhuzamosítást szeretnék alkalmazni, amely biztosítja a modulok önálló működését.

Tartalomjegyzék

1. Bevezetés

Az elmúlt néhány évtizedben az elektronikai ipar hatalmas fejlődésen ment keresztül. Ennek egyik fő irányvonala az informatika, a számítógépek fejlődése. Elég abba belegondolnunk, hogy az elmúlt 15 évben a személyi számítógépek, PC-k teljesítménye többszörösére nőtt. Ez a fejlődés elősegítette, hogy az informatikai eszközök egyre szélesebb körben elterjedtek és mára a háztartások túlnyomó többségében megtalálhatóak legyenek.

Az elektronikai iparág nagymértékű fejlődése természetesen hatással van a többi ágazatra is. Egyre szélesebb körben jelennek meg a számítógépek, különböző kommunikációs eszközök, technológiák a nagyvállalatoknál, nem csak az irodákban, hanem az üzemekben is. Az elmúlt évtized második felében ez a folyamat jelentősen felgyorsult az Internet of Things (Dolgok Internete) és az ehhez kapcsolódó ipari megoldások kapcsán, amiket hazánkban Ipar 4.0-nak hívnak, ami a 4. ipari forradalom technológiai változásaira utal. A 4. ipari forradalom legnagyobb „találmánya” az IoT, amelyet immár az ipari területeken is egyre inkább alkalmaznak, úgynevezett „kiber-fizikai rendszereket” létrehozva.

Az Ipar 4.0 projektek célja a gyárak, gyártósorok digitalizálása, olyan informatikai megoldásokkal, melyek segítségével a vállalatok magasabb minőséget képesek biztosítani termékeiknek, továbbá (akár mesterséges intelligencia használatával) lehetőség nyílik a különféle erőforrás felhasználások optimalizálására is (ehhez olyan már ismert rendszereket is integrálnak a Dolgok Internetébe, mint például a vállalatirányítási rendszerek – ERP[1]).

Az informatika térhódítása azonban nem áll meg itt. Egyre több helyen fellelhetőek az új technológiák mindennapi életünk során is, a háztartási gépekben, az egészségügyben, a mezőgazdaságban, erdőgazdálkodásban stb. Ez a sor szinte végtelen, azonban ide tartozik egy olyan ágazat is, amely valamilyen módon a Földön szinte minden embert érint, a közlekedés.

[1] ERP: Enterprise Resource Planning, vállalati erőforrás tervező rendszer vagy vállalatirányítási rendszer

Napjainkban gyártott gépjárműveket tekintve talán nem túlzó azt állítani, hogy mindegyikben megtalálható egy miniszámítógép, igaz, ez korántsem olyan, mint amit az asztalnál ülve megszokhattunk. Mesterséges intelligencia, robot vezérli az önvezető autókat, buszokat. Ez a technológia mindennapi használatban (nem teszt jelleggel) már hazánkban is jelen van, bár nem a közúti közlekedést szolgálja, hanem a budapesti 4-es metró szerelvényeit vezérli. Ez a metróvonal mára teljesen automatizált, az utasokon kívül a járművön nincs operatív személyzet.

1.1. Szakmai előzmények, motivációk

Mivel a metró már egy kötöttpályás közlekedési eszköz itt kapcsolódnék a választott témámhoz. Gyermekkorom óta érdekel a vasút és minden, ami vele kapcsolatos. A gőzmozdonyok eltűnésével és a modern mozdonyok megjelenésével a vasút sem kerülhette el a digitalizációt.

Az ilyen, digitális rendszerek alapkövei a különféle szenzorok, hiszen a számítógépek tőlük szerzik az információt a külvilágról, mint például a hőmérséklet, folyadék szint ellenőrzések, nyomás szenzorok, vagy épp egy egyszerű fotocella vagy egy ipari robot akadály érzékelő szenzora, hogy említsek néhányat közülük. A szenzorokon kívül elengedhetetlen eszközök a „beavatkozók" is, melyek segítségével a számítógép utasításokat ad a külvilág felé, vezérli az adott eszközöket. Ilyen lehet például egy relé, egy lámpa, egy motor vagy bármely olyan eszköz, amely az automata vezérlésben részt vehet és számítógép által irányítható.

Érdekesség, hogy jelenleg Nyugat-Magyarországon a GySEV[2] Zrt. vasúthálózatának túlnyomó részén automata utastájékoztató rendszer üzemel, amely a nyílt vonalon elhelyezett érzékelők jelei alapján automatikusan kezeli az állomásokon a kijelzőket, hangosbemondókat, valamint az esetleges késést is így jelzi az utazóközönségnek.

1.2. A dolgozat tartalma

Szakdolgozatom célja, egy olyan kiber-fizikai rendszer megalkotása, amely képes szenzorok és beavatkozók segítségével egy vasútmodell terepasztal irányítására külső

[2] GySEV Zrt.: Győr Sopron Ebenfurti Vasút zrt. Egy magyar-osztrák vasúttársaság Nyugat-Magyarországon, melynek székhelye Sopron.

beavatkozás nélkül. Ezzel egy automatizált rendszer kicsinyített mását modellezem. Ezt láthatjuk a következő ábrákon (1. ábra, 2. ábra, 3. ábra, 4. ábra).

1. ábra - Személyvonat az állomáson

2. ábra - Mini világ

3. ábra - Esti fények

4. ábra - Nosztalgia vonat

A következő fejezetekben a fizikai környezetet, eszközöket és az őket kezelő programom használatát és felépítését fogom taglalni. Előbb a felhasználó számára szükséges információkat a felhasználói dokumentációban írom le. Ebben a szakaszban az olvasó megismerheti a program használatának módját és beállításait.

A következő részben pedig a programozói háttérinformációk kerülnek kifejtésre a fejlesztői dokumentációban. Itt az olvasó megismerheti a program részletes leírását, a logikai felépítést és a különböző vezérlő eszközök is bemutatásra kerülnek.

Végül összefoglalom a féléves munkámat és a Szakdolgozatomat. Itt fogom bemutatni még az elért eredményeimet és kitérek a rendszer továbbfejlesztési lehetőségeire is.

2. Felhasználói dokumentáció

Ebben a fejezetben a szoftvert felhasználói oldalról ismerhetjük meg. Részletesen kifejtésre kerül minden információ, ami szoftver használatával és beállításaival kapcsolatos.

2.1. A feladat rövid leírása

A Modellvasút Irányító rendszer vagy röviden MTCS (Model Train Control System) egy vasúti terepasztal vezérlő program, amely képes a sínbe épített érzékelők, motorok, relék segítségével vonatok és jelzők irányítására.

A program felhasználója képes tehát *manuálisan* vezérelni a terepasztal „résztvevőit”, mint például a mozdonyokat, lámpákat, váltókat. De lehetőséget biztosít arra is, hogy *előre megtervezett vezérprogramok*, úgynevezett „forgatókönyvek” szerint teszteléseket végezzen a felhasználó. Utóbbi esetben a mozdonyok, jelzőlámpák, váltók stb. bizonyos meghatározott bemeneti információk (például hány mozdony van a terepasztalon) mellett véletlenszerűen meghatározott értékek mentén (például hány kört menjenek a vonatok) futtasson le egy terepasztali szimulációt.

2.2. Hardveres és szoftveres előkövetelmények

A szoftver jelenlegi verziója egy Java nyelven készült program, amely specifikusan Raspberry Pi-ra lett kifejlesztve, hiszen a terepasztal vezérléséhez elengedhetetlenek a Raspberry Pi GPIO[3] portjai. A váltók és az épületvilágítások működtetéséhez Arduino Uno szükséges, amit a Raspberryhez USB-n keresztül csatlakoztatni kell.

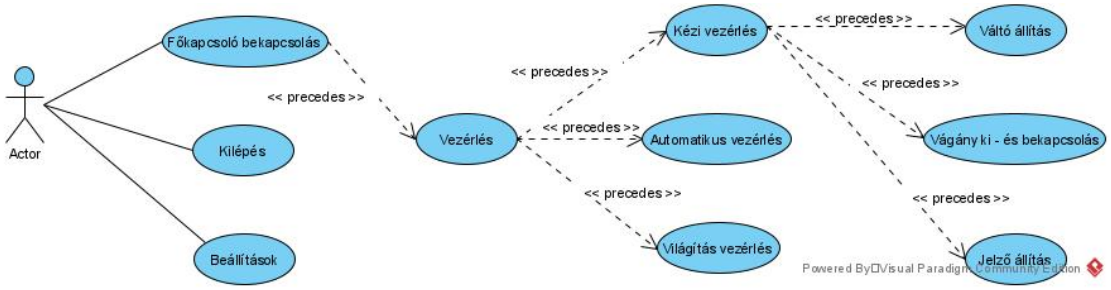
További követelmény az, hogy A Raspberry Pi-on előtelepített operációs rendszer (Raspbian) legyen, amin a program futtatásához elengedhetetlen Java virtuális gép is megtalálható, melynek verziója legalább 11-es (OpenJDK 11.0.8 verzión tesztelve).

A programot a saját mappájában lévő ModelTrainControlSystem.sh fájl megnyitásával, majd a futtatás (Execute) gombra kattintva tudjuk elindítani. A program működését és a felület komponenseit a következő alfejezet tárgyalja.

[3] General Purpose Input/Output

2.3. Felhasználói esetek

Az alábbi diagram azokat a funkciókat, illetve azok elérését (hierarchiáját) mutatja, amiket a felhasználói felületről el lehet érni. A funkciók részletes leírása a későbbiekben kifejtésre kerül.

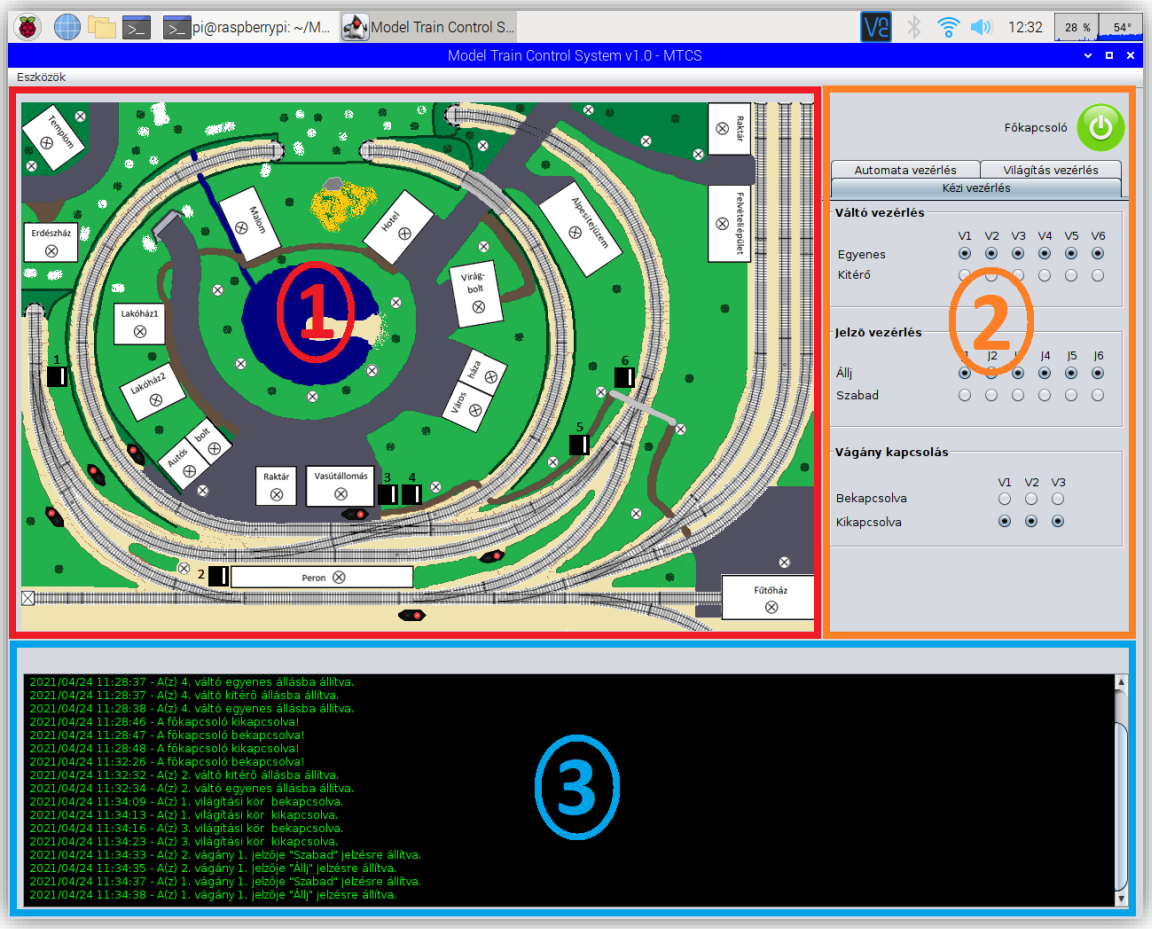


5. ábra - Felhasználói eset diagram

2.4. Kezelési útmutató

A kezdőfelület megjelenésekor szinte a teljes felület inaktív. Ennek biztonsági oka van, mégpedig az, hogy amíg a főkapcsoló nincs bekapcsolva, addig az asztalon semminemű vonatmozgás vagy más művelet nem engedélyezett, hogy a vonatok esetleges ütközéséből keletkező károk esélyét a szoftver csökkentse. A főkapcsoló bekapcsolásával a főrelé bekapcsol, így a pálya nem szakaszolt (szigetelt) részei áram alá kerülnek, amennyiben a menetszabályzó valamilyen áramirányban bekapcsolt állapotban van. Az MTCS programban három alapvető funkciót különböztetünk meg, amelyekkel a terepasztal bizonyos részeit tudjuk vezérlés alatt tartani vagy épp a számítógépre bízni az irányítást. Ez a három legfőbb modul pedig a kézi vezérlés, az

automatikus vezérlés és a világítás vezérlés. A módok közti váltás a kezdőfelületen érhető el. A kezdőfelület főbb részei a következők:

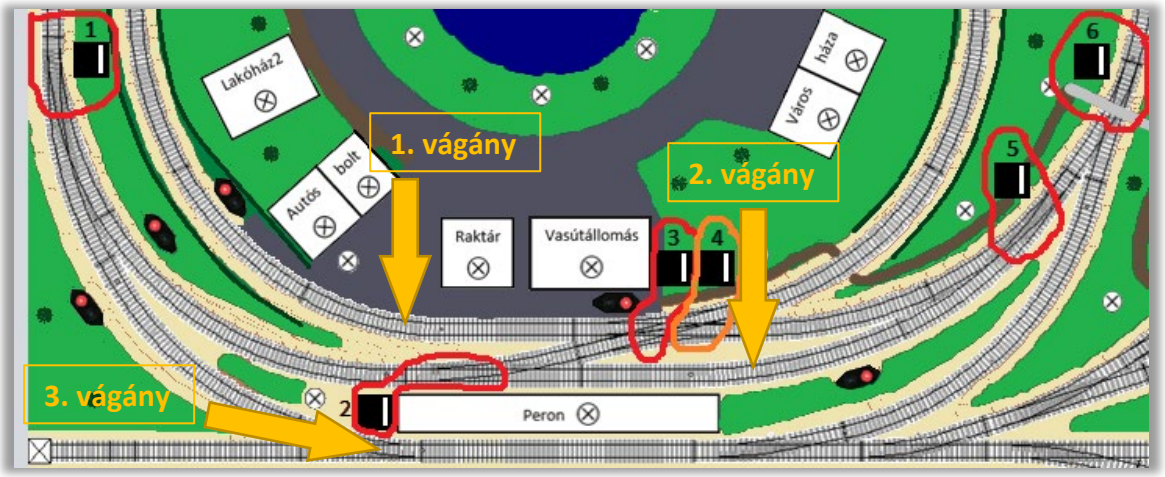


6. ábra - A rendszerkezdőfelülete

1. **<u>A terepasztal sematikus térképe:</u>** A program futása során a főablakban a térkép mindig látható. A térképen található jelző, váltó és vágány piktogramok (utóbbiak csak az állomás 3 átmenő vágányán) egyfajta gombként funkcionálnak, kattintásra az adott eszköz állapota megváltozik, azonban ez csak a kézi vezérlési üzemmódban használható.
2. **<u>Vezérlőpanel:</u>** A vezérlőpanelen elhelyezett fülek segítségével lehetséges a különböző módok közti váltás. Ezeket a módokat részletesen tárgyaljuk a 2.4.1, 2.4.2, 2.4.3 alfejezetekben.
3. **<u>Státusz ablak:</u>** Ezen a felületen jelennek meg az éppen végrehajtott utasítások, események és az aktuális állapot szöveges információi, mint például, hogy az adott vonat mennyi kört fog teljesíteni vagy, hogy milyen váltóállítás történt.

Ezen a területen található továbbá a főkapcsoló is, melynek szerepe már korábban említésre került.

A terepasztal váltóinak és vágányainak elhelyezkedését és számát az alábbi ábra mutatja. ()

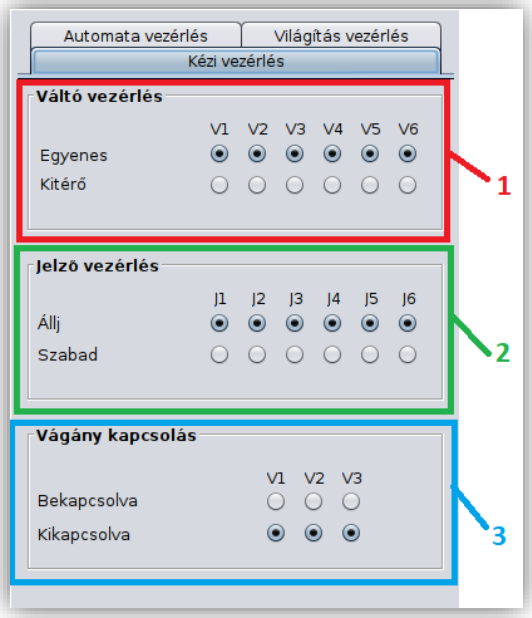


7. ábra - Vágányok és váltók

2.4.1. Kézi vezérlés

Kézi vezérlési üzemmód a rendszer alapállapota. Ekkor teljes mértékben felügyelhető az asztal eszközeinek vezérlése, a váltók, jelzők és vágányok szabadon állíthatók. Ebben az üzemmódban a rendszer egyáltalán nem felügyeli, hogy a vonatok ütközhetnek-e vagy sem egy kialakult szituációban, a teljes terepasztal vezérlés a felhasználó feladata.

A kézi vezérlési üzemmódban, mint korábban említésre került a térkép piktogramjaira kattintva befolyásolhatóak az adott eszközök. Emellett a vezérlőpanelen a kézi vezérlést kiválasztva a megfelelő választógombok segítségével is irányíthatóak a váltók, jelzőlámpák és vágányok. A kézi vezérlés panel (8. ábra) részei a következők:

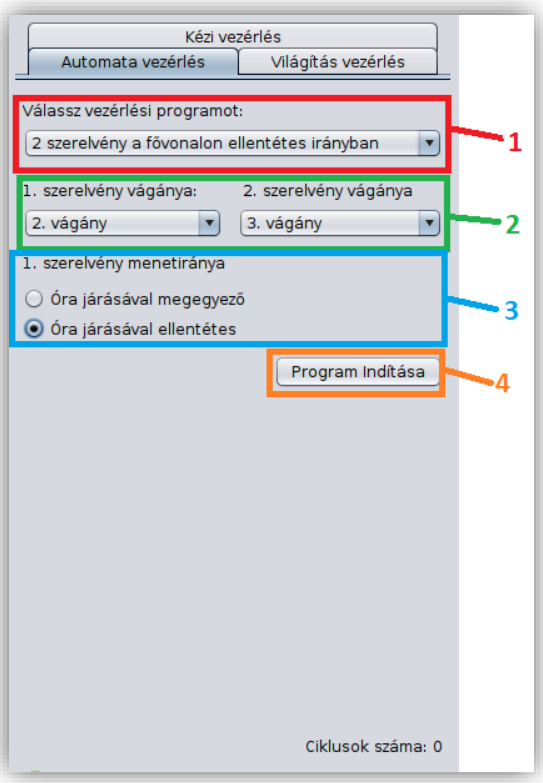


8. ábra - Kézi vezérlés panel

1. Váltó vezérlés: A 6 pár kapcsoló gombbal az összesen 6, digitálisan állítható váltót lehet kitérő vagy egyenes állásba állítani.
2. Jelző vezérlés: Ezen a területen szintén 6 pár választógomb helyezkedik el, melyekkel a 6 fényjelző vezérelhető. A gombok állásától függően a jelzők mutathatnak vörös, azaz állj jelzést vagy zöld, azaz szabad jelzést.
3. Vágány kapcsolás: Az állomás 3 átmenő vágányának reléit a 3 választógomb segítségével lehet ki- vagy bekapcsolni.

2.4.2. Automata vezérlés

Az automatikus vezérlés során a szoftver maga vezérli a teljes vasúti hálózatot, ekkor nem szükséges, sőt nem is lehetséges felhasználói beavatkozás a váltók, jelzők és relék kezelésébe. Automata vezérlés során a kézi vezérlés panel, valamint a térkép gombjai letiltásra kerülnek, ekkor a programot kizárólag leállítani lehet, valamint a főrelé is kikapcsolható. Az automatikus vezérlés paneljén (9. ábra) a következő lehetőségek találhatóak:



9. ábra - Automatikus vezérlés

1. **<u>Vezérlési program kiválasztása:</u>** A legördülő menüből összesen 6-féle automatikus vezérlési program választható ki, melyeket a későbbiekben részletesen tárgyalunk.
2. **<u>Induló vágányok:</u>** A vezérlési program megadása után aktívvá válnak a szerelvények induló vágányainak legördülő mezői. Az induló vágányok megadásakor figyelni kell arra, hogy az 1. és a 2. szerelvény nem indulhat azonos vágányról, ellenkező esetben hibaüzenetet dob a program.

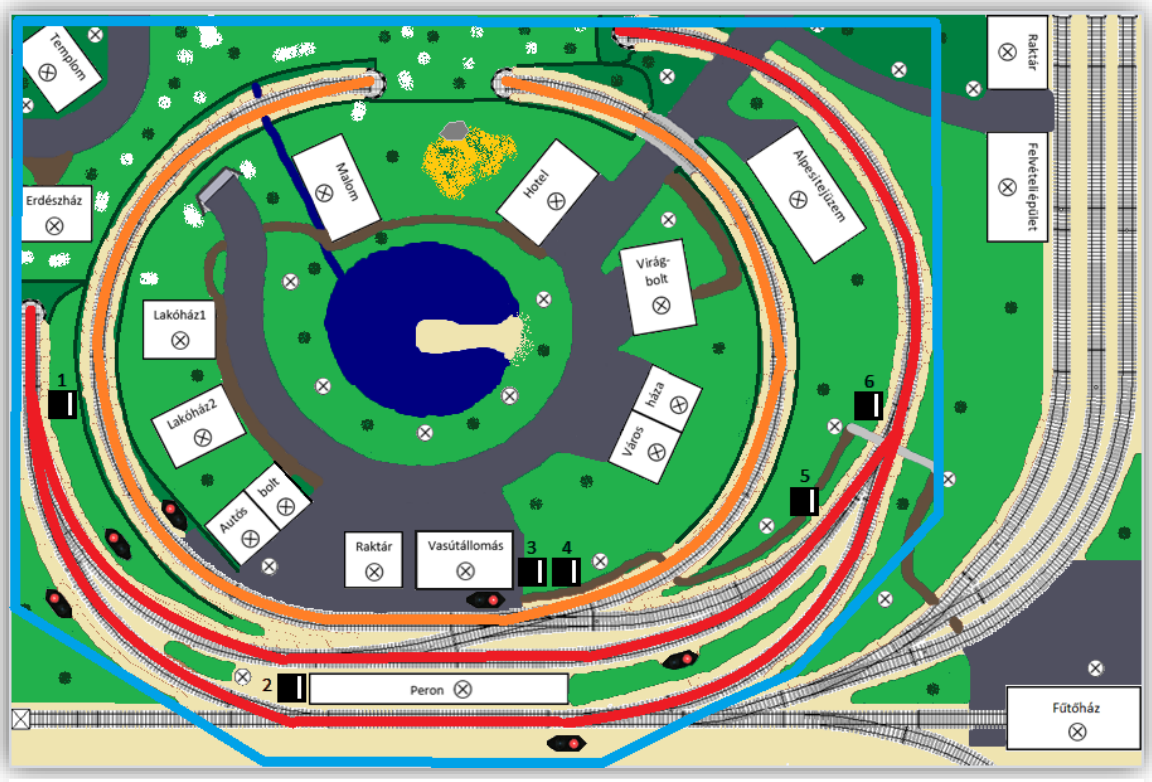
3. **<u>Menetirány választás:</u>** Az első szerelvény menetirányának pontos megadása a vezérlési program normál futásának egyik alapfeltétele. A kiválasztott vezérlési programból és az első vonat menetirányából a szoftver meg tudja határozni, hogy a két vonat megfelelő közlekedtetéséhez mely szenzorokat mikor kell figyelnie. Az itt megadott iránynak meg kell egyeznie a menetszabályzón (10. ábra) beállított áramiránnyal, ami attól függ, hogy az adott áramirány mellett a mozdony a terepasztalon az óra járásával megegyező vagy épp ellentétes irányban mozog.

10. ábra - Menetszabályzó

4. **<u>Indítás és leállítás:</u>** Amennyiben minden paraméter beállításra került az indító/leállító gomb is aktív állapotú lesz, ekkor a gombra kattintva elindul a kiválasztott vezérlési program a megadott paraméterekkel. Az indítás után a gomb felirata megváltozik és „Program leállítása” felirat lesz rajta, ekkor az automatikus program megállítására szolgál. Az automatikus program csak akkor áll le, ha minden vezérelt szerelvény az állomásra ért.

Automatikus vezérlőprogramok:

A vezérlőprogramok kizárólag a terepasztal digitális egységekkel felszerelt területén képesek irányítani a vonatokat. Ezt a területet a 11. ábra kék színnel jelöli. Amennyiben a vonat ezekről a vágányokról kihalad, a szoftver már nem tudja megfelelően vezérelni. A terepasztal program által vezérelhető területét a későbbiekben vezérelt területként említjük. Továbbá fontos megkülönböztetni a fővonalat és a mellékvonalat.



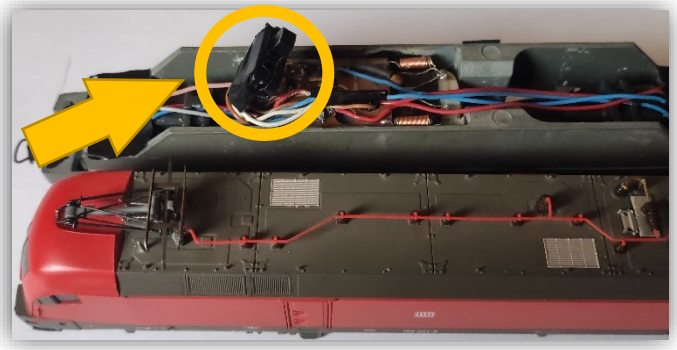
11. ábra - A vezérelt terület, a fővonal és a mellékvonal

A fővonal a „külső” pálya (piros vonallal jelöli a térkép), míg a mellékvonal a „belső” (narancssárga színnel jelöli a térkép), mindezt a 11. ábra mutatja.

- **1 szerelvény a teljes asztalon:** Ehhez a vezérlési programhoz a teljes vezérelt területről el kell távolítani a szerelvényeket (pl. a tárolóvágányokra kell állni velük manuálisan) kivéve azt az 1-et, amelyet a program vezérelni fog. A vezérlés során a szoftver véletlenszerűen dönti el, hogy a vonat megáll-e az állomáson vagy tovább halad, illetve hasonlóan, ha megállt, akkor elindul-e. Ezen felül azt is a program határozza meg, hogy a vonat egy pályán mennyi kört tegyen meg és ezt követően melyik vágányon folytassa útját. Az automata program indításakor az induló vágány tetszőleges lehet, ügyelni kell arra, hogy a

megadott vágány megegyezzen a vonat valós vágányával és az irány is a valós iránnyal.

- **<u>2 szerelvény a fővonalon:</u>** Ebben a vezérlési programban megkülönböztetünk ellentétes és azonos irányú vezérlést. Azonos irányú vezérlés esetén a 2 vonat a 2. és 3. vágányon egy irányban felváltva közlekedik. Az egy-egy vonat által megtett körök számát a program véletlenszerűen állapítja meg, amikor az egyik szerelvény elérte a meghatározott körszámot a másik szerelvény indul el, majd fordítva. Az ellentétes irányú vezérlés szinte teljesen megegyezik az azonos irányúval, annyi különbséggel, hogy az egyik mozdony irányváltó kapcsolóját a másik mozdonnyal ellentétesen kell kapcsolni. A mozdonyok irányváltó kapcsolói a burkolatuk alatt találhatók (12. ábra). A vezérlőprogramok a futás során a fővonalat használják, azonban ennél a két automata vezérlési programnál a vágányok megadása tetszőleges, mielőtt az automatikus vezérlési program elindul a vonatokat megfelelő vágányokra állítja. Ügyelni kell arra, hogy a vágányok és az első szerelvény irányának megadása megegyezzen a valósággal.



12. ábra - Az irányváltó kapcsoló

- **<u>3 szerelvény a teljes asztalon:</u>** Ez a vezérlési program lényegében az előző (2 szerelvény a fővonalon) bővített változata. A különbséget az jelenti, hogy 1 szerelvényt az állomás első vágányára is be kell állítani és ez csak a belső körön, azaz a mellékvonalon fog mozogni, míg a fővonalon a korábban leírt módon történik a közlekedés. Ennél a programnál is megkülönböztethetünk

13. ábra - Tehervonat érkezik az állomásra

azonos és ellentétes irányú vezérlést a fővonalon. Azonos irányú vezérlésnél mindhárom szerelvény egy irányban fog mozogni. Ellentétes irányú vezérlésnél pedig az egyik fővonali mozdony irányváltó kapcsolóját ellentétes állásba kell helyezni a másik 2-höz képest. Továbbá ügyelni kell arra, hogy a mellékvonalon futó szerelvény iránya a fővonali 1. szerelvény irányával egyezzen meg, máskülönben a vezérlési program nem fogja megfelelően vezérelni a mellékvonali vonatot. Ezeknél a vezérlési programoknál ügyelni kell arra, hogy a két fővonali vonat a 2. és a 3. vágányon helyezkedjen el, míg a mellékvonali az 1. vágányon. Fontos, hogy az 1. szerelvény iránya megegyezzen a valós iránnyal.

- **Tisztító vonat:** Mivel a modellvasút mozdonyai a sínből kapják a működésükhöz szükséges áramot, így mozgásuk során az érintkezés és apró kisülések miatt a sínekre szennyeződés kerül, amely rontja az érintkezést ezáltal egyre több szennyeződés kerül rá. Ennek elhárítása érdekében a síneket tisztítani kell. Amíg a szennyeződés csak kisebb mértékű ez a művelet manuális tisztítás helyett tisztító vonattal is elvégezhető. A tisztító vonatot (14. ábra - 1 mozdonyt és egy speciális tisztító kocsit) az állomás 4. vágányára (11. ábra balra lent) kell állítani, majd a programot kiválasztva a képernyőn megjelenő utasítások követésével elindítani a tisztítást. A szerelvény a program során előre megadott számú kört teljesít (beállítási paraméterekről később) a teljes vezérelt terület minden vágányán meghatározottan az óra járásával ellentétes irányban.

14. ábra - Tisztító vonat

2.4.3. Világításvezérlés

Ebben az üzemmódban a vezérlőpanel világítás vezérlés fülét kell kiválasztani. Ez a rendszer futása során bármikor megtehető függetlenül attól, hogy fut-e épp automatikus vezérlés vagy sem. A terepasztalon 15 épületben továbbá a peronokkal és a közvilágítással együtt összesen 10 világítási kör van, melyek külön-külön kapcsolhatók. Ezek a következők:

15. ábra - Világítás vezérlés panel

1. Közvilágítás
2. Üzemi terület térvilágítása
3. Peronok
4. Tejüzem, 2-es számú lakóház, hotel
5. Rendezőpályaudvar épületei
6. Malom, erdészház, állomási raktár
7. Autóalkatrész üzlet – bolt (épületrész), rendőrség (épületrész)
8. Autóalkatrész üzlet – lakóház (épületrész), városháza (épületrész)
9. 1-es számú lakóház, virágbolt, templom
10. Állomás épület, fűtőház

2.4.4. Beállítások

A terepasztalon a vonatok automata működésébe közvetlenül nem lehet beavatkozni, azonban néhány paraméter megadására van lehetőség. A különböző vezérlési paraméterek megadását az Eszközök menü Beállítások menüpontjában tehetjük meg.

16. ábra - Beállítási paraméterek

A terepasztal paraméterei 3 csoportba sorolhatók, ahogy a jobb oldali ábrán (16. ábra) is látható. Az 1. csoportba tartoznak a különböző várakozási idők, amelyek a vonatok indulási, áthaladási vagy egyéb időzítéseit tudják befolyásolni.

A 2. csoport a körök száma, itt adható meg egy adott vonat által megtett körök száma egy vezérlési programban és a tisztító körök száma is.

Végül az Arduino beállítások, itt lehet megadni az arduino panel (később, továbbfejlesztés esetén: panelek) tulajdonságait.

A programban a következő paramétereket lehet beállítani:

<u>Várakozási idők:</u>

- **Indulási késleltetés:** Amikor egy adott vágányról a vonat elindul nem azonnal a zöld jelzésre teszi ezt. A jelző szabad jelzése és a szerelvény tényleges elindulása között eltelt idő beállítására szolgál ez a paraméter. Értékét másodpercben, az 1-20 tartományban lehet megadni, alapértéke: 3 másodperc.

- **Jelző áthaladási várakozás:** Amikor a vonat az állomásról elindult elhalad a kijárati jelző mellett. Ezzel a paraméterrel megadható egy olyan időintervallum, ami biztosítja, hogy a teljes vonat kiment az állomásról. Értékét másodpercben, az 1-20 tartományban lehet megadni, alapértéke 7 másodperc.
- **Általános várakozási idő:** Két vonat állomáson történő találkozása esetén az első vonat érkezésétől a második vonat indulásáig eltelt idő, nem beleértve a második vonat indulási késleltetését. Értékét másodpercben, az 1-20 tartományban lehet megadni, alapértéke 1 másodperc.
- **Állomási tartózkodási idő:** Amennyiben a program úgy dönt (például az 1 szerelvény teljes asztalon történő vezérlésénél), hogy a vonat nem indul tovább az állomásról, akkor a következő döntésig eltelt időt ezzel a paraméterrel lehet beállítani, a paraméter megadott értéke határozza meg, hogy a vonat mennyit áll az állomáson, ha épp nem indul tovább. Értékét másodpercben, az 1-60 tartományban lehet megadni, alapértéke 15 másodperc.

Körök száma:

- **Egy menetben megtett körök száma:** Bár a szoftver az automatikus program során az adott vonat megtett köreinek számát véletlenszerűen sorsolja, lehetőség van megadni a kisorsolható körszám maximumát. Ezen paraméter értéke határozza meg, hogy az automata vezérlőprogram során egy adott szerelvény mennyi kört tehet meg egy menetben, azaz a következő megállásig. A paraméter értékét az 1-10 tartományban lehet megadni, alapértéke 5.
- **Tisztító körök száma:** A tisztító programban a tisztító szerelvény ebben a paraméterben megadott számú kört teljesít a vezérelt terület minden vágányán. A paraméter értékét az 1-5 tartományban lehet megadni, alapértéke 3.

Arduino beállítások:

- **Arduino Uno:** A jelölőnégyzet(ek) kiválasztásával aktiválható vagy inaktiválható az adott Arduino panel. Amennyiben a jelölőnégyzet aktív az OK gombra kattintva a program kinyitja az USB portot, amennyiben lehetséges, ellenkező esetben hibaüzenet jelenik meg. Alapértelmezetten az Arduino Uno USB kapcsolata engedélyezve van, azaz a jelölőnégyzetben van egy pipa.

- **<u>Arduino időtúllépés:</u>** Amennyiben az Arduino panel a kiadott utasításokat nem nyugtázza, azaz nem válaszol a megadott időn belül, a program az USB kapcsolatot megszakadtnak nyilvánítja és mindennemű vezérlést és mozgást azonnal letilt. A paraméter értékét milliszekundumokban, az 50-1000 tartományban lehet megadni, alapértéke 300 milliszekundum.

3. Fejlesztői dokumentáció

Ebben a fejezetben a szoftver tervezésének, elkészültének technológiai oldala kerül bemutatásra. Részletesen kifejtve a specifikációt, valamint a felhasznált eszközök, módszerek leírását. Megismerhetjük a program működését, logikai felépítését is.

3.1. A probléma részletes specifikációja

Az elkészítendő szoftver egy olyan alkalmazás, amely képes **szenzorok adatai** alapján meghatározott **folyamatok irányítására** beavatkozó egységeken keresztül. Ezen feladat megvalósítását egy modellvasút terepasztalon hajtjuk végre, ahol a szoftver feladata a különböző vasúti objektumok (kitérők, jelzők stb.) és szerelvények vezérlése.

Szükséges, hogy a program rendelkezzen egy **könnyen kezelhető grafikus felülettel**, amely lehetővé teszi a felhasználó számára a funkciók közti egyszerű navigálást. Továbbá **lehetőséget biztosít** a terepasztal **vezérlési paramétereinek megadására** – konfigurációs fájlok használatával –, amelyek a program leállítása után is megtartják értékeiket.

Az elkészítendő alkalmazásban **3 alapvető funkciót** kell megkülönböztetni, melyek **a kézi vezérlés, az automata vezérlés** és végül a **világítás vezérlés.**

A **kezelőfelületen** meg kell jeleníteni a terepasztal sematikus **alaprajzát**, amelyen jelölve vannak a főbb útvonalak, az épületek és azok nevei, a közvilágítás lámpái és természetesen a sínhálózat, a jelzők, valamint a váltójelzők, amelyek a váltók egyenes vagy kitérő állását jelenítik meg. Ez **a térkép** a program futása során **mindvégig látható** és a fizikailag létező terepasztal valós idejű digitális mását mutatja a felhasználónak.

Szintén a kezdőképernyőn kap helyet a vezérlésmódok közti váltás, továbbá egy **státuszablak**, amelybe a rendszer **naplózza az aktuális műveleteket**. Ennek segítségével nyomon követhető a program belső működése, esetleges probléma esetén az okok kiderítése is egyszerűbbé válik.

A grafikus felület tervezésekor gondoskodni kell arról, **hogy veszélyes helyzet esetén** a teljes terepasztalon egy főkapcsolóval ki lehessen kapcsolni a vonatok mozgásához szükséges feszültséget. Ebben az esetben a kitérők, jelzők és lámpák állapota nem

változtatható meg, vagyis a program teljes grafikus kezelőfelülete (a menük, a státuszablak és a főkapcsoló kivételével) inaktív lesz.

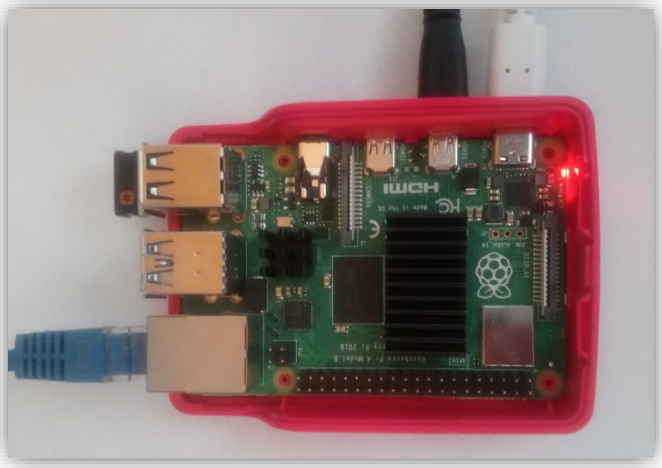
3.2. A rendszer fizikai felépítése

Ebben az alfejezetben a kiber-fizikai rendszerem fizikai szereplőit mutatom be.

3.2.1. Raspberry Pi és Arduino Uno

A teljes automata rendszer központi vezérlőegysége egy Raspberry Pi 4 [1]. A Raspberry nem más, mint egy tenyérnyi számítógép, úgynevezett SoC. Az **SoC** az angol „***System on Chip***" kifejezésből származik, amely lényegében azt jelenti, hogy egy teljes számítógépet integrálunk egyetlen áramköri lapra. Ezeknek az SoC lapoknak a rendes asztali számítógépekhez hasonlóan egy teljes operációs rendszer szükséges működésükhöz. A bennük található ROM-ban gyakorlatilag a BIOS-nak megfelelő, alapszintű szoftver található, amely az operációs rendszer elindítására szolgál.

A **Raspberry** Pi az egyik, ha nem a **legnépszerűbb** SoC a piacon. Az eszközt az Egyesült királyságban fejlesztették ki 2012-ben és eredetileg oktatási (programozás oktatási) feladatokra szánták.

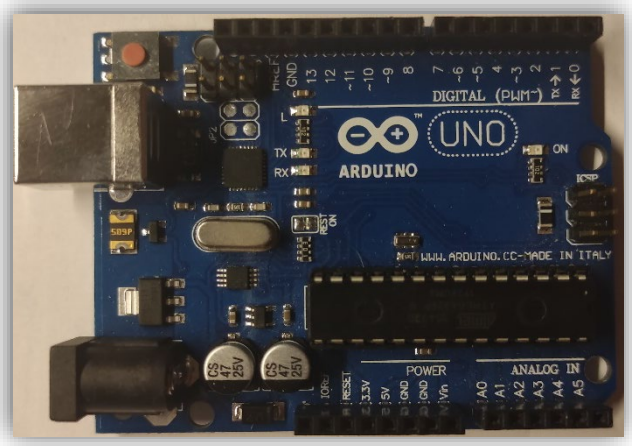


17. ábra - Raspberry Pi 4

Az elmúlt években több változat is megjelent, a legfrissebb modell jelenleg a Pi4, amely, a fent említettekkel szemben 4 magos 1,5 GHz-es ARM processzorral, illetve 2, 4 vagy 8 GB memóriával rendelkezik.

A Raspberry sajátosságához hozzátartozik azonban, hogy rendelkezik ún. **GPIO** portokkal, amelyek lehetőséget adnak **külső eszközök programokból történő vezérlésére** vagy valamilyen **szenzorokból érkező információk fogadásra**. Fontos megemlíteni, hogy egy Raspberry PC saját operációs rendszerrel rendelkezik, amelyet micro SD kártyán tárol.

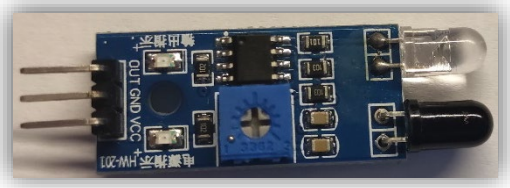
A terepasztal eszközeinek vezérlésének szerves része a Raspberry Pi mellett egy Arduino panel is [2]. Az Arduino egy szintén széles körben elterjedt eszköz. Ez azonban már sokkal inkább egy egyszerűbb mikrokontroller, mintsem egy SoC. Processzorának sebessége mindössze 16 MHz, flash memóriájának mérete 32 KB. Összesen 14 digitális kimenettel rendelkezik, melyből 6 úgynevezett **PWM**[4] láb. **A Raspberry-vel szemben az Arduino nem rendelkezik saját operációs rendszerrel** sem, felépítése sokkal egyszerűbb. Működés közben egyetlen programot képes futtatni, melyet korábban egy asztali PC-n vagy notebookon kell elkészíteni és az eszközre USB-n keresztül feltölteni.



18. ábra - Arduino Uno fejlesztői panel

3.2.2. Szenzorok és beavatkozók

Az automata vezérlés legfontosabb elemei a **szenzorok és beavatkozó eszközök**. A vezérlő szoftver a szenzorok segítségével szerez tudomást a külvilágról és a begyűjtött információk alapján a beavatkozó eszközök segítségével hajt végre műveleteket egy folyamatban. [3]



19. ábra - HW-201 infravörös szenzor

Ahogy a nagyvasúton, úgy a terepasztalon is szükség van szenzorok beépítésére ahhoz, hogy a program a beavatkozók segítségével a szerelvényeket megfelelően tudja vezérelni. **A szerelvények helyzetének meghatározására HW-201 típusú infravörös akadályérzékelő szenzorok** [4] (19. ábra) kerültek beépítésre minden jelző elé és mögé. Összesen 3 vágányon 6 jelző digitálisan vezérelt, ami azt jelenti, hogy 12 db infravörös szenzor került felhasználásra. Az adott jelző előtti szenzornak a vonat megállításában

[4] PWM: Pulse Width Modulation, impulzus szélesség moduláció.

20. ábra - Szenzor a jelző előtt

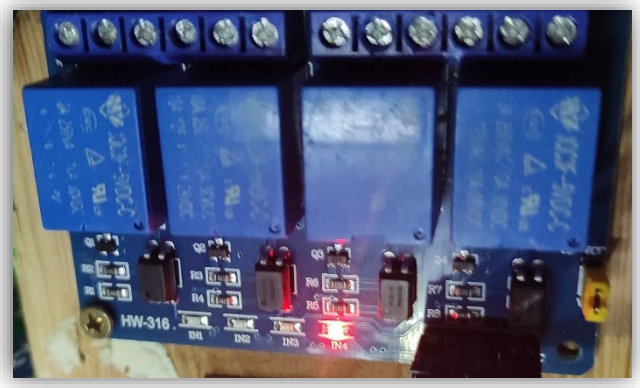
van kulcsfontosságú szerepe, amikor a vonat eléri ezt a szenzort a program automatikusan kikapcsolja a pályaszakaszhoz tartozó relét, ezzel megállítja a vonatot. A jelző mögötti érzékelő feladata, hogy a program tudomására juttassa, hogy a vonat elhagyta az állomást, ekkor a jelző a nagyvasút mintájára vörösre vált.

A szenzor működése viszonylag egyszerű [5]. Két dióda található rajta, melyek közül az egyik a kibocsátó, a másik pedig a vevő. A kibocsátó dióda (19. ábra, áttetsző) infravörös fényt bocsát ki, amelyet a vevő (19. ábra, sötét) érzékel, amennyiben a szenzor előtt vagy felett valamilyen tárgy, akadály van. Potenciométer (19. ábra, középen a „+ jel") segítségével az érzékenység szabályozható. A terepasztalba beépített szenzorok a stabil működés biztosítása érdekében minimális érzékenységgel működnek, mivel a környezeti fényben található infravörös komponensek hamis jeleket eredményeztek az eszközök tesztelése során. A minimális érzékenység miatt azonban a szerelvények érzékelése vált instabillá, ennek érdekében a mozdonyok forgóvázaira alufólia csíkok kerültek (21. ábra), amik tökéletesen biztosítják a szenzorok stabil érzékelését.

21. ábra - Fényvisszaverő fóliák a mozdonyok forgóvázain

Mint az korábban említésre került, az automata rendszer eszközeinek másik fontos csoportját a beavatkozó eszközök alkotják. A modellterepasztalon kétféle beavatkozó és egy jelző eszközt találhatunk. Beavatkozó eszközök a relék és a váltó állító szervomotorok. Jelző eszközök pedig a vasúti fényjelzők.

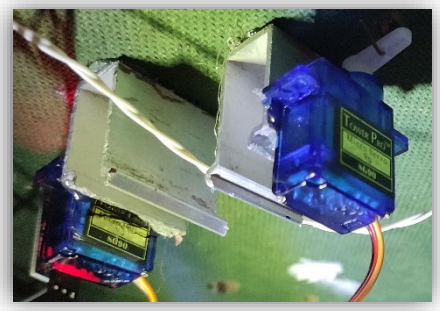
A terepasztalon egy vasútállomás található, amely összesen **3 átmenő vágánnyal rendelkezik**. Ezeken a vágányokon **szigetelt szakaszok kerültek kialakításra**. A vonatok indítása ezen szakaszok be- és kikapcsolása révén lehetséges, amit relék biztosítanak. Egy 4 csatornás relémodul [6] (22. ábra) 3 reléje látja el a szigetelt szakaszok vezérlését, míg a **4. relé egyfajta főrelé** szerephez jut. **Ennek segítségével a teljes terepasztal áramtalanítható**, abban az esetben, ha valamilyen probléma merül fel (például valamilyen okból egy vonat kisiklik). Bizonyos szituációkban a programnak automatikusan is ki kell kapcsolnia a főrelét a nagyobb problémák megelőzése érdekében.

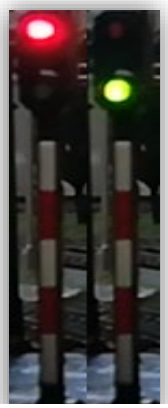


22. ábra - 4 csatornás relémodul (bekapcsolt főrelé)

24. ábra - SG-90 szervomotor

A teljes vágányhálózatban **összesen 11 váltó** található, ezek közül **6 vezérelhető szoftveresen**. Ezek állítását SG-90 típusú szervomotorok (24. ábra) látják el, amik a vágányhálózat alatt találhatók. A motorok mozgatása PWM jelekkel történik az Arduino panelen keresztül [7].


23. ábra - A 3-as és 4-es váltók motorjai


25. ábra - Fényjelző

A vasút és ezzel együtt a mini világ elengedhetetlen kellékei még a **vasúti fényjelzők**, melyek **vörös, illetve zöld fénnyel** jelzik (25. ábra), hogy a vonatnak meg kell állni vagy áthaladhat a jelzőn. Az állomáson összesen 6 darab fényjelző került elhelyezésre, ezek a program vezérlése szempontjából egyszerű LED-eknek minősülnek. A jelzők mindegyike a Raspberry Pi-hez csatlakozik.

További látványelemként megemlítendő, hogy az asztal összes, **több mint 10 épületébe** és a közvilágítás oszlopaiba **LED-ek kerültek bekötésre**. Ezeket a kezelőfelületen a világítás vezérlés lehetőségnél lehet ki- és bekapcsolni.

26. ábra - Épületvilágítás

3.3. Eszközök közti kommunikáció

Ahogy a korábbiakból kiderült a **terepasztal vezérlését** egy **Raspberry Pi és egy Arduino Uno** végzi. A két eszköz közül a tényleges irányítást a „vezér” azaz a Raspberry Pi végzi, az Arduino kisegítő eszközként működik. Erre legfőképp azért volt szükség, mivel a Raspberry Pi 4 összesen 28 GPIO porttal rendelkezik (néhány korábbi verzió kevesebbel), ez pedig nem elegendő számú port ahhoz, hogy minden szenzor, relé, motor és LED bekötésére lehetőség legyen. Továbbá a váltómotorok működtetéséhez PWM jelek szükségesek, amiket bár a Raspberry digitális PWM-mel képes előállítani, az Arduino analóg lábai pontosabb vezérlést tesztnek lehetővé, ami egy váltó milliméterpontos állításakor fontos szempont.

Ennek **a két eszköznek azonban valamilyen módon kommunikálni kell** egymással. Erre a célra USB[5] kommunikációt kell létesíteni köztük. Az **USB kommunikáció** [3] lehetővé teszi az eszközök közti **kétirányú információcserét.**

A vezérlés során **az Arduino** önálló döntést nem hoz, nem avatkozik bele a folyamatba, kizárólag **a Raspberry Pi utasításait hajtja végre**. A két eszköz közti kommunikáció a két irányban egymástól eltérő szintaxist használ. Az egyszerűbb irány az, amikor az Arduino választ ad, nyugtázza a kért utasítást. Ekkor egy OK, azaz „0” választ küld a Raspberry felé vagy, ha valamilyen módon ismeretlen funkciókódot kap, akkor „1” hibakóddal üzen vissza.

A másik irányban az üzenet 3+1 tagból épül fel, ahol az első tag a **funkciókód**. Összesen 1 karakter hosszú, ez a tag határozza meg azt, hogy **a Raspberry pontosan milyen**

[5] USB: Universal Serial Bus – univerzális soros port

funkció végrehajtását kéri az Arduino-tól. A funkciókódokat a következő táblázat tartalmazza.

1. táblázat - Arduino funkciókódok

Funkciókód	A funkció leírása
0	Ping – nem történik semmilyen művelet végrehajtás, a Raspberry csak egy választ kap, hogy az USB kapcsolat él
1	DigitalWrite – az Arduino meghatározott digitális lábának ki- vagy bekapcsolását teszi lehetővé
2	ServoControl – meghatározott analóg lábra kapcsolt szervomotor megfelelő szögbe állítását teszi lehetővé (egy váltó állítása)
3	MultipleServoControl – egyedi vágánykapcsolati kód alapján a vágányút összes váltójának megfelelő állapotba állítását teszi lehetővé

A funkciókódot követi a **pin, vagyis a láb azonosítója**, amely 2 karakterből áll, amennyiben 10-nél kisebb indexű lábra vonatkozó utasítást kell kiadni, úgy a pin értéke például a 3-as láb esetén 03. Amennyiben az utasítás vágányút állításra, azaz váltók csoportos állítására vonatkozik, úgy a pin értéke a vágánykapcsolati kód (track connection code), amikből az Arduino előre definiáltan egyértelműen meg tudja határozni, hogy mely váltókat milyen állásba kell állítania. A vágánykapcsolati kódokat a következő táblázat tartalmazza.

2. táblázat - vágánykapcsolati kódok

Kód	Leírás
10	Vágányút beállítása az 1-es vágányon áthaladáshoz. A váltók állása a következő: - 4-es váltó egyenes állású - 3-as váltó kitérő állású
13	Az 1-es vágányról a 3-as vágányra történő vágányút állítás. A váltók állása a következő: - 3-as és 4-es váltó egyenes állású - 5-ös és 6-os váltó kitérő állású
20	Vágányút beállítása az 2-es vágányon áthaladáshoz. A váltók állása a következő: - 2-es és 5-ös váltó egyenes állású - 1-es és 6-os váltó kitérő állású

30	Vágányút beállítása az 3-as vágányon áthaladáshoz. A váltók állása a következő: - 1-es és 6-os váltó egyenes állású
31	A 3-as vágányról az 1-es vágányra történő vágányút állítás. A váltók állása a következő: - 1-es, 2-es 3-as és 4-es váltó kitérő állású
default	Az összes váltót alaphelyzetbe, azaz egyenes állásba állítja

Végül az üzenet utolsó, 3. tagja az adott lábra **beállítani kívánt értéket** vagy **állapotot** tartalmazza. Ez a szakasz 3 karakter hosszú, vágányút állítás esetén értéke 000. Digitális utasítás esetén, ha az adott láb bekapcsolása a cél az érték 001, egyébként 000. Egyszeri váltóállításnál ez a tag tartalmazza az állítandó váltó beállítani kívánt szögét. Az üzenet végére zárótagként egy „@” karakter kerül, ebből az Arduino a karakterenkénti olvasás során észleli, hogy az üzenet véget ért, feldolgozható és végrehajtható. Az üzenet tagjait „;” választja el egymástól.

Példaként az Arduino digitális 5-ös lábának bekapcsolása a következő paranccsal tehető meg: **1;05;001**@

3.4. Program logikai és fizikai szerkezetének leírása

A rendszer architektúrája háromrétegű. Legfelső, a felhasználó által elérhető réteg a grafikus kezelői felület, ezen helyezkednek el az asztal felügyeletéhez, irányításához szükséges panelek, vezérlőegységek.

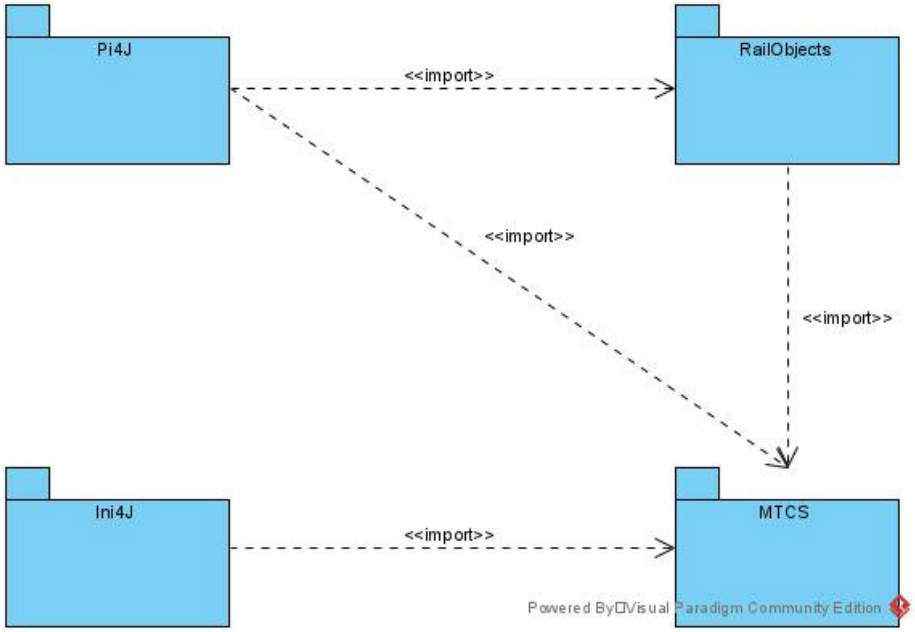
Következő réteg a program mag, az „üzleti logika”, amely közvetlenül kommunikál a grafikus felülettel köztes osztályokon keresztül. Ebbe a rétegbe tartozik az Arduino is.

A legalsó réteg a fizikai eszközök szintje, ahova az jelzések célba érnek, vagy szenzorok esetén az információ, utasítás innen érkezik. Itt találhatók meg a szenzorok, váltóállító motorok, relék és a jelzők.

3.4.1. Programcsomagok, modulok, osztályok

A rendszer kódszintű felépítése két fő csomagra bontható, melyeken kívül van további kettő külső csomag is, ezek a Pi4j és az Ini4J. A csomagok hierarchiáját a 27. ábra mutatja. Vegyük előbb a Pi4J csomagot. A Pi4J [8] egy szabadon felhasználható program könyvtár,

amely a Raspberry Pi GPIO portjainak használatát, soros kommunikációt, ezen felül egyéb kommunikációs lehetőségeket (I2C[6], SPI[7]) és szoftveres PWM megoldásokat tartalmaz. A rendszer működése során ezek közül a GPIO kezelés és a soros kommunikáció kerül a középpontba.



27. ábra - A szofver csomag architektúrája

A rendszer legfőbb osztályait az MTCS csomag tartalmazza. Ebben a csomagban találhatók azok az osztályok, amik a felhasználói grafikus felületet és a konkrét vezérlési feladatokat látják el. A csomagon belüli osztályok osztálydiagramját a 28. ábra mutatja.

[6] I2C: Inter-Integrated Circuit, soros kkommunikációs kapcsolat, általában mikrovezérlők között, illetve azok felsőbb szintű eszközzel való kommunikációjára használják

[7] SPI: Serial Peripherial Interface: szinkronizált soros kommunikációs interfész. Általában beágyazott rendszerek, pl. ipari vezérlőegységek (PLC-k) használják.

28. ábra - Az MTCS csomag osztálydiagramja

3.4.2. Az MTCS csomag

Főablak

A program grafikus felülettel rendelkezik, melynek főablaka a MainWindow osztályban került implementálásra. A kezelőfelület főablaka összesen 3 nagyobb panelből épül fel. Ezek a vezérlési panel, a státusz panel és a térkép, mint önálló panel.

Az asztal sematikus térképén az épületek, jelzők és a váltók piktogramjai az adott eszközök pillanatnyi állapotát jelölik valós időben. Ezt egyedi hátterű címkék teszik lehetővé, ahol az adott piktogramra kattintáskor a címke háttérképe cserélődik ki. A váltók, jelzők és az állomás átmenő vágányai emellett érzékelik a kattintást is, amit kézi

vezérléskor használhat a felhasználó. Az épületek piktogramjai csak az adott épület világításának állapotát mutatják.

A vezérlési panel tartalmazza a vezérlési módok egyedi paneljeit, amik között egy lapozható panellel (Javaban: jTabbedPane) lehet váltani. A lapozható panelen a három vezérlési panel helyezkedik el, ezek a kézi vezérlés, automata vezérlés és a világítás vezérlés. Amennyiben a felhasználó automatikus vezérlést indít a kézi vezérlés letiltásra kerül. A kézi vezérlés esetén a panel rádiógombjai vezérlik a váltókat, jelzőket és reléket. Az adott eszköz aktuális státusza megjelenik a térkép adott piktogramján akkor is, ha az utasítás a panelről, rádiógomb segítségével lett kiadva.

A világítási körök be- és kikapcsolására két állapotú gombok (jToggleButton) szolgálnak. Ezek ki- és bekapcsoláskor szinte közvetlenül az Arduinonak küldik az üzenetet. Az üzenetküldés mechanizmusa megegyezik a kézi vezérlésű váltóállítással. Amikor a felhasználó bekapcsol egy világítási kört az eseménykezelő a TrainConrol osztály világítás vezérlő függvényét (tcLightControl) hívja meg (váltó esetén az SWChange), majd ez továbbítja a megfelelő funkciókódot és az értéket a FunctionModule WriteArduPin metódusának. Ez a függvény már a kapott funkciókódot és átküldendő értéket a korábban taglalt szintaktikába rendezi és elküldi az USB kapcsolaton keresztül.

A főablak harmadik panelja a státusz panel. Ezen jelennek meg a program működésével kapcsolatos információk, üzenetek egy szövegmezőben. Ez a szövegmező görgethető, de nem szerkeszthető.

Beállítás ablak

A csomag másik grafikus osztálya a SettingsWindow. A felhasználó ebben az ablakban adhatja meg a terepasztal alap vezérlési paramétereit. Ezeket a paramétereket három csoportba soroljuk, amik a várakozási idők, körök száma és az Arduino beállítások. Mivel a cél, hogy a rendszer a beállításokat a Raspberry leállítása után is eltárolja, ezért erre a feladatra konfigurációs fájlt használ. A konfigurációs fájlok vagy ini fájlok egyfajta kulcs-érték párokat tartalmaznak, külön csoportokba sorolva. A csoportok jelenleg megegyeznek a paraméterek három csoportjával. Az ini fájlok kezelése a Java nyelvben kiegészítő osztálykönyvtárakkal lehetséges. Erre a célra szolgál a másik külső csomag, az

Ini4J [9]. A Pi4J csomaghoz hasonlóan szabadon hozzáférhető csomag, ami a konfigurációs fájlok java nyelvben történő használatát teszi lehetővé.

A program két alapvető csomagra osztandó. Egyik csomagba tartoznak azok az osztályok, melyek eszközei közvetlen kapcsolatban állnak a terepasztallal, a sínhálózattal. Ebbe, a vasúti eszközök (RailObjects) csomagba tartoznak a váltók, vágányok (relék, mint vágány objektum) és a jelzők. Ezzel szemben a másik, MTCS csomagba a grafikus felület és a program mag osztályai tartoznak, amelyek tartalmazzák a fent leírt modulokat, funkciókat, mint a kézi vezérlés, automata vezérlőprogramok stb.

FunctionModule

A FunctionModule osztályban a közvetlen hardveres kommunikációt megvalósító függvények kerültek implementálásra. Ezek a metódusok kezelik a Raspberry GPIO portjait, illetve ahogy korábban említésre került az Arduino USB kapcsolatát is ez az osztály használja.

Itt találhatóak továbbá azok az „elemi" vezérlő függvények, amiket az automata vezérlési programok hívnak meg. Ilyenek a TrainStrat, TrainStop, isBusyTrack stb. A függvények bővebb leírása a javadoc dokumentációban található meg.

TrainControl

A TrainControl osztály a rendszer egyik legfőbb központi osztálya. A grafikus elemeken kívül a program egészét vezérli. Ez az osztály tartalmazza azokat a RailTrackObj objektumokat (a RailObject csomagról később), amik a vágányokat, jelzőket és szenzorokat program szinten deklarálják. A TrainControl osztály a Thread osztályból örököltetett, ennek funkciója az, hogy amennyiben a felhasználó egy automata vezérlést elindít, az külön programszálon fog futni.

Az automata vezérlőprogramok függvényei itt kerültek implementálásra a OneTrain függvény egy szerelvényt vezérel a teljes terepasztalon, míg a two TrainsOnMainTrackDifferent és two TrainsOnMainTrackSame függvények két-két szerelvényt vezérelnek a fővonalon ellentétes, illetve azonos irányban. Ezen függvényekhez kapcsolódik 3 szerelvény esetén egy újabb szálon a OneTrainOnSecondaryLine függvény, ami a belső körön vezérel egyetlen szerelvényt. A cleaningTrain függvény a tisztító vonat mozgásáért felel.

Itt történik az Arduino panel (továbbfejlesztés esetén panelek) USB kapcsolatainak kezelése is, illetve a program indulásakor a beállítások beolvasása a konfigurációs fájlból.

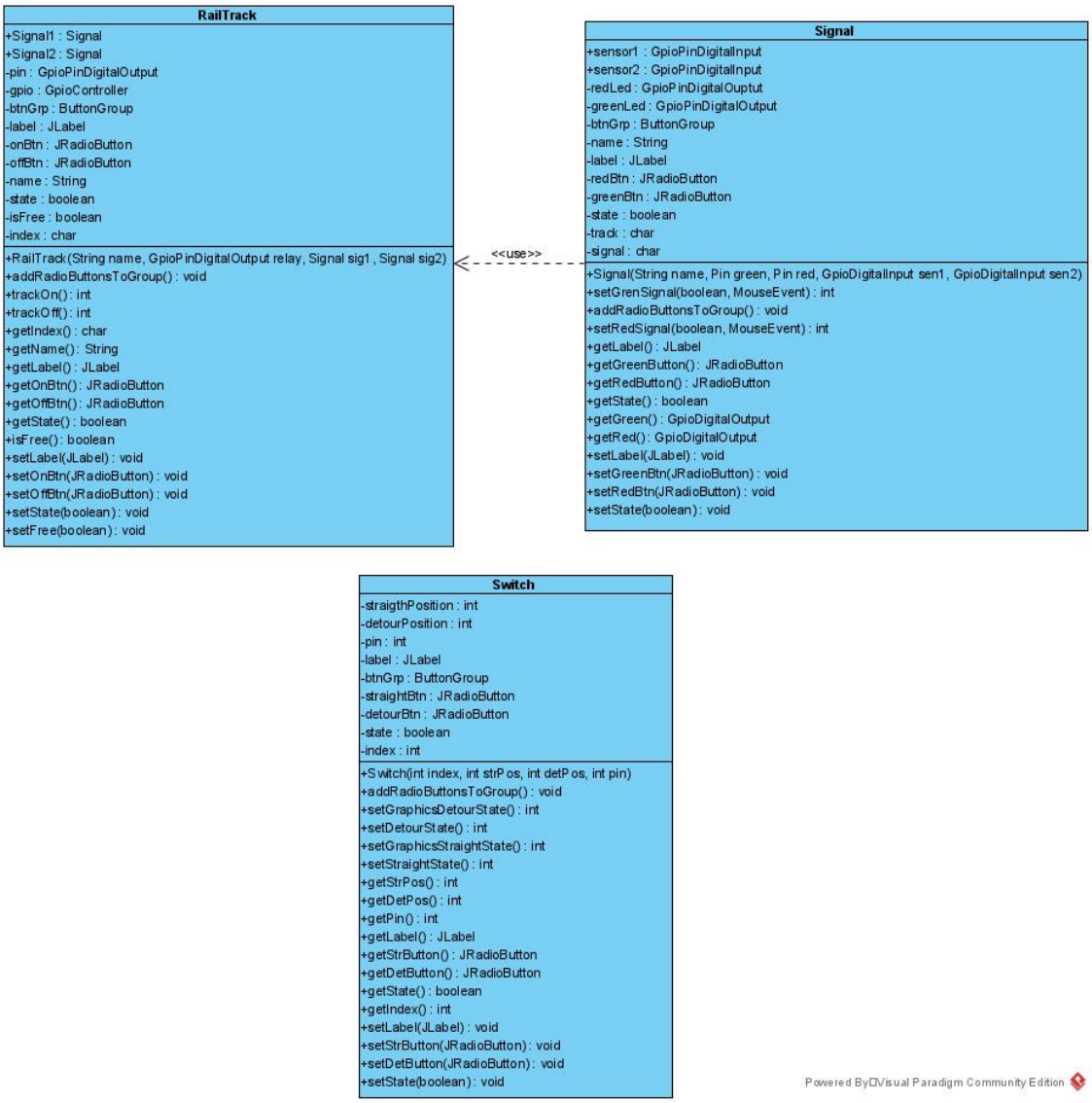
Arduino és LogFunctions

Az Arduino, mint objektum egy-egy valós Arduino panelt testesít meg a rendszer számára. Az adott Arduino objektumában tárolódik el annak neve, elérési útvonala (connectionString), a kapcsolathoz szükséges információk (baud rate, nyitott a port vagy sem (boolean flag) stb.) és a kapcsolathoz szükséges soros port számát (serial : int).

A LogFunctions osztály statikus metódusokat tartalmaz. Ezek a MainWindow státusz paneljének szövegmezőjét használják kimenetként. A metódusok különböző információkat, státuszüzeneteket írnak ki a kijelzőre, mint például a váltó állítás, ebben az esetben, hogy melyik váltó, milyen irányba állt.

3.4.3. A RailObjects csomag

Ebben a csomagban a terepasztalon lévő beavatkozó eszközök „interfész” osztályai találhatóak. Ezen objektumok metódusai a konkrét fizikai eszközt vezérlik. Ahogy korábban említésre került, a FunctionModule osztály „elemi” vezérlőfüggvényeiben kerülnek felhasználásra. Funkciójuk a jelzők zöld vagy vörös jelzésének beállítása vagy a váltók pozíciójának módosítása vagy épp egy relé ki- és bekapcsolása vágányok esetén. A csomag osztálydiagramját a 29. ábra szemlélteti.



29. ábra - A RailObject csomag osztálydiagramja

SignalObj

A SignalObj a jelzők működtetésére szolgáló objektum. Metódusaival állítható az adott jelző állapota. Az objektumban tárolódik az adott jelző címkéje, amit a főablak a térképen jelenít meg. Minden jelzőhöz tartozik továbbá két GPIO pin, amik a vörös és zöld LED-eket jelölik. Ahogyan „Szenzorok és beavatkozók” fejezet említi egy-egy jelző előtt és mögött szenzorok helyezkednek el. Ezeket a rendszer objektum szinten rendeli a jelzők objektumaihoz. A szenzorok nem saját speciális objektumok, hanem a Pi4J csomagból használható GpioPinDigitalInput típust használják.

RailTrackObject

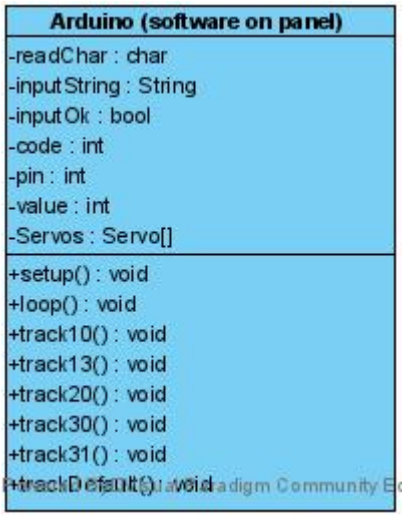
A vágány objektum. Szerkezete és feladata hasonló a jelző objektumhoz. Metódusaival befolyásolható a vágány tápellátásáért felelős relé állapota. A rendszer itt rendeli össze

a vágányokat a hozzá tartozó jelzőkkel, azaz minden vágányhoz tartozik két jelző objektum is. Így létrejön egy olyan komplex struktúra, amely kényelmesen és átlátható módon használható, hiszen a teljes rendszer alapjául szolgáló eszközöket összesen három objektum deklaráció által létre lehet hozni. Ezt a TrainControl osztály tartalmazza (track1, track2, track3).

Korábban említésre került a főrelé is. Program szinten ez is egy „vágány”, azonban a térképhez szükséges címke helyett a főkapcsoló címkéjével lett összerendelve és nem tartozik hozzá jelző és ezáltal szenzorok sem, csak a relé állapota változtatható. A főrelé nem egyetlen elszigetelt szakasz áramellátásáért felel, hanem a másik pólust szakítja meg, ezáltal a szigetelt szakaszokat akkor is képes áramtalanítani, ha azok egyébként bekapcsolt állapotban vannak.

3.4.4. Arduino szoftver

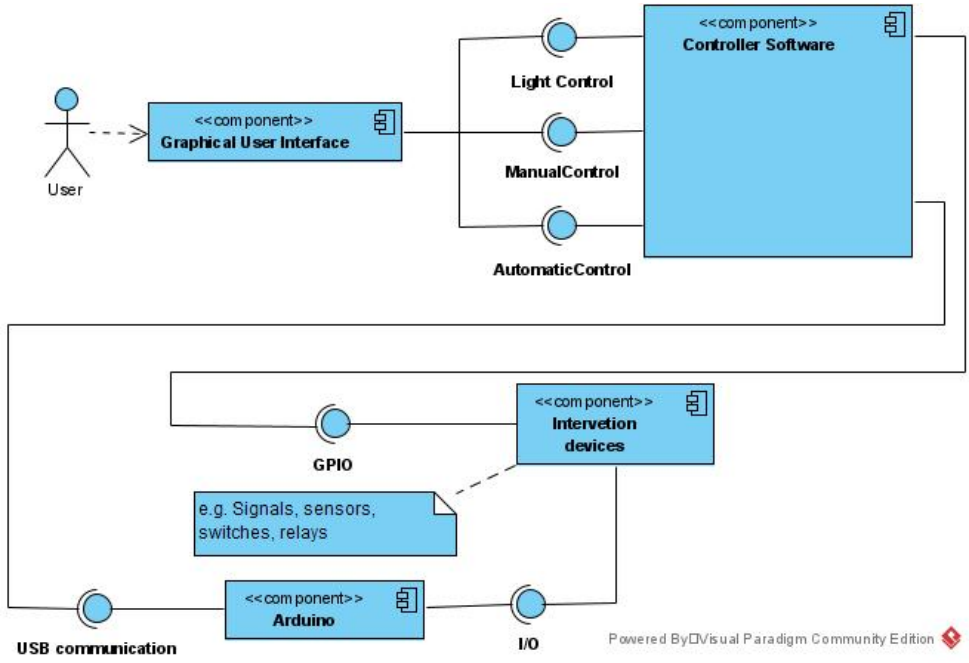
Mivel az Arduino Uno panelen futó program is egyfajta modul a Raspberry Pi-n futó Java program számára, így mindenképp említést érdemel a felépítése. Működése egyszerű. A folyton futó ciklusban az Arduino folyamatosan figyeli, hogy érkezett-e valamilyen utasítás az USB porton keresztül. Amennyiben üzenetet kap azt karakterenként egy változóba beolvassa, majd feldolgozza. Ekkor USB kapcsolaton keresztül nyugtázza, hogy az üzenetet megkapta és feldolgozta, majd végrehajtja a műveletet. Bár az Arduino programban nem beszélünk osztályokról, mivel egy C alapú programozási szintaktikát követ, de függvényeit és változóit az alábbi osztálydiagram jól szemlélteti.



30. ábra - Az Arduino panelen futó szoftver osztálydiagramja

3.4.5. A szoftver komponensei és a komponensek kommunikációja

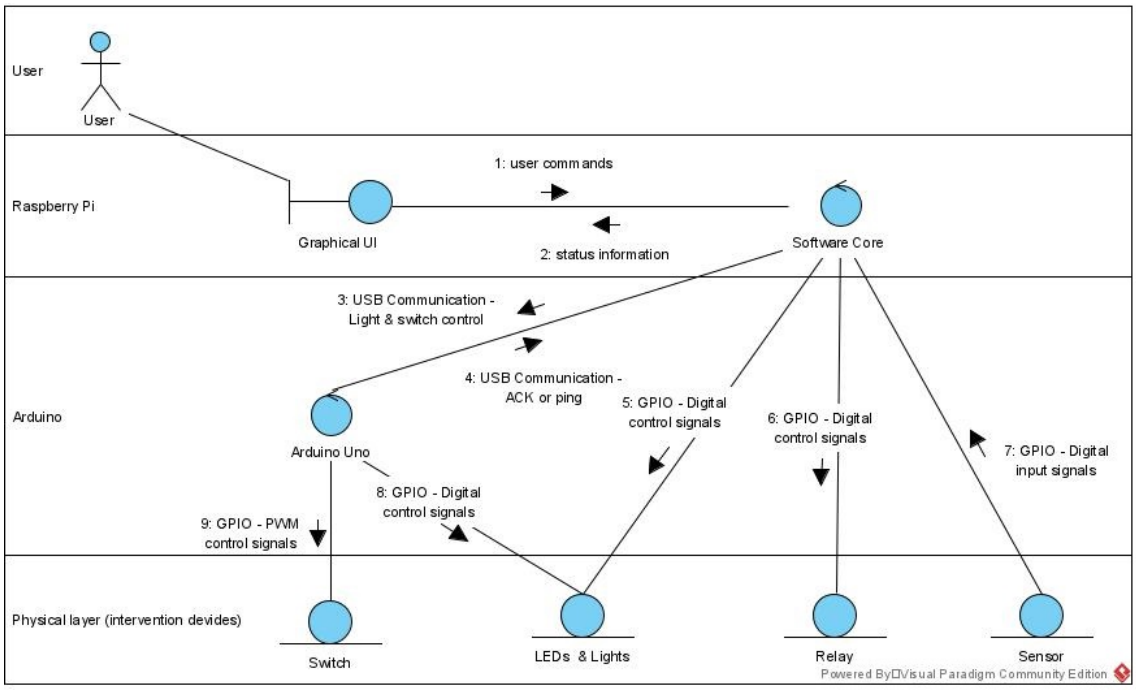
A program főbb egységeinek egymáshoz való viszonyát a komponens diagram ábrázolja.



31. ábra - Komponens diagram

Komponensek kommunikációja

A szoftver komponensei közti információcsere a program működésének alapját képezi. Ezeket a kapcsolatokat a kommunikációs diagram mutatja be.



32. ábra - Kommunikációs diagram

(Megjegyzés: a diagramok jó minőségben nagyítható formátumban a mellékletek között megtalálhatók).

3.4.6. Hibajelzések

Ha a rendszerben fellépő hiba egyszeri alkalommal fordult elő, akkor a beállítások ablakban kell lehetőséget biztosítani annak törlésére. Ha a hiba többször egymás után előfordul az alábbi táblázatban leírt hibakódokat kell megjeleníteni a grafikus felületen.

3. táblázat - Hibakódok

Hibakód	Leírás
0x01	Java belső hiba (IllegalStateException)
0x02	Hiba az Arduino Nano USB kapcsolatában
0x42	Hiba az Arduino Uno USB kapcsolatában
0x04	Időzítési hiba (InterruptedException)
0x08	A váltóállítás sikertelen
0x10	Nem értelmezhető funkciókód az Arduino Nano-nál (funkciókódokról az USB kapcsolatnál)

0x50	Nem értelmezhető funkciókód az Arduino Uno-nál (funkciókódokról az USB kapcsolatnál)
0x20	Általános hiba

3.5. A program tesztelése, tesztüzem

Egy elkészült szoftver tesztelése önmagában véve is egy elég komplex feladat. A helyzetet tovább árnyalja, ha az adott szoftver valamilyen eszközt vezérel vagy épp valamilyen eszközben, például mikrokontrollerben (Arduino) vagy PLC[8]-ben fut. Ezért a modellvasút vezérlő programja egyfajta tesztüzemben került tesztelésre. A korábban asztali számítógépen elkészült program elsőre az éles környezetben, a terepasztal szenzoraival került kipróbálásra. A részletes tervezés és implementáció eredményeként a szoftver kisebb hibák kivételével megfelelően futott, az automatikus vezérlő programokban nem adódtak problémák. A néhány, főként az épületvilágítást érintő hiba javításra került.

[8] PLC: Programmable Logic Controller – programozható logikai vezérlő

4. Összefoglalás és további fejlesztési lehetőségek

Az ipar és az informatika kettőse meglátásom szerint a következő években egyre szorosabbra fogja fűzni a kapcsolatát. Ezzel együtt az informatika az élet minden pontján meg fog jelleni, vagy már megjelent, gondoljunk csak egy mai mobiltelefonra, nem más, mint egy kisméretű számítógép, akárcsak a Raspberry Pi. Szakdolgozatom téma választásakor épp ezért tettem le voksom erre a területre. Emellett a vasút és a vonatok már gyerekkorom óta kedvelt dolgok az életemben. A modellvasút pedig már elég hosszú ideje meghatározó eleme a szabadidős tevékenységeimnek. Ezekből kiindulva lett célom, hogy egy olyan modellvasút terepasztal irányítást hozzak létre, amely némi kreativitással sokkal gazdaságosabb megoldást nyújt, mint a piacon lévő más, digitális vasúti vezérlések, melyek költsége akár 6 számjegyű összeg is lehet és az kizárólag a vezérlőegységet tartalmazza. Ezzel szemben az analóg rendszerek kiépítése kisebb költségekkel jár, azonban ezzel a megoldással a kompromisszumok száma csökkenthető.

A későbbiekben ezt a szoftvert szeretném tovább fejleszteni, hiszen rengeteg olyan funkciót tud még ellátni, amik a digitális rendszerekben adottak. Ilyenek például az állomási hangosbemondók, a vasúti átjáró automata biztosítása, ahol lehetőség van kézi vezérlésre is vagy a vonatok fokozatos lassítása, megállítása, illetve a gyorsítás, ezzel szemben a relé kissé hirtelen, egyszerűen a feszültség megszüntetésével állítja meg a vonatot. Azonban egy megfelelően bekötött relés kapcsolat lehetőséget ad arra, hogy szoftveresen változtatható legyen az áramirány, amellyel már nem csak a terepasztal jelenlegi vezérelt területén, hanem további szenzorok és váltók bevonásával akár tolatási műveletek is végezhetők a rendező pályaudvarokon. A modularitás és architektúra szempontjából egy kényelmes megoldást jelenthet egy TCP/IP alapú kommunikáció szerver és kliens között, ahol a szerver a Raspberry Pi a kliens pedig egy, a jelenlegivel megegyező vagy hasonló grafikus felületű kliens, amely futhat akár Windows, Linux asztali PC-n vagy épp egy mobiltelefonos applikációban.

Ezzel, az elkészült programmal egy régi célom valósult, hiszen sikerült egy olyan rendszert megalkotni, ami nem csak egy terepasztal vezérlését képes kiszolgálni, hanem az itt szerzett tapasztalattal az ipari rendszerek működése is világosabbá vált számomra.

5. Irodalomjegyzék

[1] G. Ruzsinszki, Programozható Elektronikák, 2019.

[2] R. Cseh, Arduino programozási kézikönyv, Budapest: TavIR, 2011.

[3] J. C. Shovic, Raspberry Pi IoT Projects, Liberty Lake, Washington, USA: Apress, 2016.

[4] Components 101, „Components 101," 30. augusztus 2018.. [Online]. Available: https://components101.com/sensors/ir-sensor-module. [Hozzáférés dátuma: 2. március 2021.].

[5] Silicon TechnoLabs, „components101.com," [Online]. Available: https://components101.com/asset/sites/default/files/component_datasheet/Datasheet%20of%20IR%20%20Sensor.pdf. [Hozzáférés dátuma: 2. március 2021.].

[6] Handson Technology, „www.handsontec.com," [Online]. Available: https://www.handsontec.com/dataspecs/4Ch-relay.pdf. [Hozzáférés dátuma: 3. március 2021.].

[7] Arduino, „Arduino Reference," [Online]. Available: https://www.arduino.cc/reference/en/. [Hozzáférés dátuma: 5. március 2021.].

[8] Pi4J, „The Pi4J Project," Free Software Foundation, 5 március 2019. [Online]. Available: https://pi4j.com/1.2/. [Hozzáférés dátuma: 16. február 2021].

[9] I. Szkiba, „[ini4j] The Java .ini library," 5. augusztus 2019.. [Online]. Available: http://ini4j.sourceforge.net/. [Hozzáférés dátuma: 14. március 2021.].

6. Ábrajegyzék

7. Mellékletek

Mappa/Fájlnév	Leírás
Diagrams könyvtár	A rendszer UML diagramjainak nagyítható változatait tartalmazza
Javadoc – ModelTrainControlSystem könyvtár	A java nyelvhez elérhető Javadoc fejlesztői dokumentáció
SourceCodes könyvtár	A rendszer forráskódjainak könyvtára
Petrovics_Bence_VESC5N_eredetisegi_nyilatkozat.pdf	A szakdolgozat eredetiségének nyilatkozata