



Pázmány Péter Katolikus Egyetem
Információs Technológiai és Bionikai Kar

ÖNÁLLÓ LABORATÓRIUM

Laborautomatizáció a rendszerbiológiában

Bakó András [FZPA8B]
Mérnökinformatikus Bsc

2025

Témavezetők: Pongor Csaba István

Tartalomjegyzék

Témabejelentő	i
1. Bevezetés	1
1.1. Feladatom	1
2. Előzmények	3
2.1. Piaci alapú megoldások	3
2.1.1. Folyadékkezelő robotokba integrálható megoldások	3
2.1.2. Különálló megoldások	5
2.1.3. Integrált nagy rendszerek	6
2.2. Nyílt forráskódú megoldások	8
3. Tervezés	10
3.1. Választott megoldások indoklása	10
3.1.1. Felhasznált Szoftverek	10
3.1.2. Kiválasztott Hardver	11
4. Eredmények	14
4.1. Hardverösszeállítás és működés	14
4.1.1. ATX táp átalakítás	14
4.2. Megvalósított szoftver	16
4.2.1. Felhasználói felület (GUI)	17
4.2.2. Kommunikáció és vezérlés	17
4.2.3. Képfeldolgozás	18
4.2.4. Konfigurációs fájlok	19
5. Összefoglalás	20
5.1. Eredmények	20
5.2. Jövőbeli tervek	20

1. fejezet

Bevezetés

A biológiai laborautomatizálás célja, hogy a laboratóriumi munkafolyamatokat hatékonyabban, pontosabban és lehetőleg emberi beavatkozás nélkül valósítsa meg. Egy ilyen gép feladatai közé tartozik a Petri-csészék mozgatása, a minták precíz elhelyezése a táptalajon, valamint a kolóniák növekedésének időbeli nyomon követése képfeldolgozással. A rendszernek képesnek kell lennie a minták szabályos és egységes telepítésére, a csészék pozícionálására a kamerarendszer alatt, illetve a képek rögzítésére és elemzésére. A cél nem csupán az adatrögzítés automatizálása, hanem az is, hogy a kísérleti folyamat közben bekövetkező változások (például sejtnövekedés, kolóniaelágazás vagy színváltozás) is nyomon követhetők legyenek.

Az iparban léteznek teljesen integrált automatizált rendszerek, azonban ezek zártak, drágák és nehezen testreszabhatók. Ezért döntöttünk egy saját fejlesztésű, nyílt forráskódú szoftverekre és szabványos hardverekre (pl. Arduino Mega, RAMPS 1.4, Marlin firmware) épülő megoldás mellett, amely egyszerűbb és költséghatékony alternatívát nyújthat kutatási célra. A nyílt architektúra lehetővé teszi a rendszer rugalmas bővítését, valamint a funkciók és algoritmusok testreszabását a konkrét biológiai feladat igényei szerint.

1.1. Feladatom

Önálló laboratóriumi munkám során egy olyan rendszer részleges megvalósításában vettettem részt, amelynek célja a Petri-csészék automatizált kezelése, valamint a bennük növekvő baktériumkolóniák megfigyelése. A projekt jelenlegi állapotában a teljes rendszer még nem tekinthető késznek. Feladatom a hardveres vezérlés működésbe hozatala, valamint a szoftveres kommunikációs támogatás biztosítása volt.

A munkám során elsődlegesen a rendszerhez szükséges vezérlőelektronika (Arduino

Mega, RAMPS 1.4, léptetőmotorok) működésének összehangolását, a mechanikus egységek tesztelését és a Marlin konfigurálása után a firmware-re épülő mozgásvezérlés beállítását végeztem el. Emellett gondoskodtam arról, hogy a magasabb szintű vezérléshez szükséges szoftveres alapok — például a G-kód parancsok küldéséhez szükséges soros kommunikáció — Python nyelven megvalósításra kerüljenek.

A rendszer irányítása egy grafikus felhasználói felületen keresztül történik, amelyhez a vezérlési logika és a szoftveres működés alapjainak megtervezésében is részt vetttem. Ezeket a funkcionális lehetőségeket a témavezetőmmel közösen határoztuk meg, figyelembe véve a projekt későbbi bővíthetőségét és kutatási célú alkalmazhatóságát.

2. fejezet

Előzmények

2.1. Piaci alapú megoldások

A baktériumtenyésztés és -megfigyelés automatizálására számos korszerű technológiai megoldás létezik a piacon. Ezek a rendszerek különböző automatizálási szinteket és funkcionálisokat kínálnak, lehetővé téve a laboratóriumi munka hatékonyságának növelését, a hibalehetőségek csökkentését, valamint a reprodukálhatóság javítását. A megoldások skálája a teljesen integrált rendszerektől a részben automatizált vagy integrálható eszközökig terjed, beleértve azokat a modulokat is, amelyeket meglévő folyadékkezelő robotokba lehet beépíteni.

2.1.1. Folyadékkezelő robotokba integrálható megoldások

Azok számára, akik már rendelkeznek automatizált folyadékkezelő rendszerekkel, ezek a megoldások lehetővé teszik a kolóniaválogató funkciók egyszerű bővítését. Ez különösen előnyös nagy mintaszám kezelésekor, illetve ahol már kialakult robotikai infrastruktúra áll rendelkezésre.



2.1. ábra. PickoloFreedom EVO 150 package

Tecan Colony Picking Package: A Tecan pipettázó rendszereibe integrálható kolóniaválogató modul, minőségellenőrzési célokra optimalizálva [1].



2.2. ábra. easyPick

EasyPick – Hamilton STAR platformhoz: Automatikus kolóniakiválasztó modul a Hamilton folyadékkezelő rendszerekhez, különösen jól alkalmazható nagy mintaszám esetén [2].

2.1.2. Különálló megoldások

Ezek a rendszerek lehetőséget biztosítanak arra, hogy csak bizonyos részmunkafolyamatokat automatizáljunk, vagy kiegészítsük meglévő berendezéseinket. Előnyük, hogy viszonylag egyszerűbben és költséghatékonyabban integrálhatók egy meglévő laboratóriumi infrastruktúrába.



2.3. ábra. QPix

QPix – Molecular Devices: Kolónia-válogató sorozat, amely nagy áteresztőképes-ségű válogatást tesz lehetővé, fluoreszcencia vagy morfológia alapján [3].



2.4. ábra. PIXL

PIXL – Singer Instruments: Félautomata kolóniaválogató rendszer, sötétmezős

képalkotással és pontos mintavételezéssel [4].



2.5. ábra. Copan Colibri crop640

Universal Colony Picker – Hudson Robotics: Kombinálja a kézi és automatizált működést, és különféle környezetekbe integrálható [5].

2.1.3. Integrált nagy rendszerek

Az integrált rendszerek olyan komplex megoldásokat kínálnak, amelyek képesek a teljes baktériumtenyésztési és kolóniaválogatási munkafolyamat automatikus végrehajtására. Ezek ideálisak nagy áteresztőképességű laboratóriumi környezetekben, ahol fontos a valós idejű megfigyelés és a manuális beavatkozások minimalizálása.



2.6. ábra. ScanStation 300

ScanStation 300 – Interscience: Egy teljesen automatizált valós idejű inkubációs és kolóniaszámláló rendszer, amely akár 300 Petri-csészét is képes párhuzamosan kezelní és figyelni a kolóniák növekedésére [6].



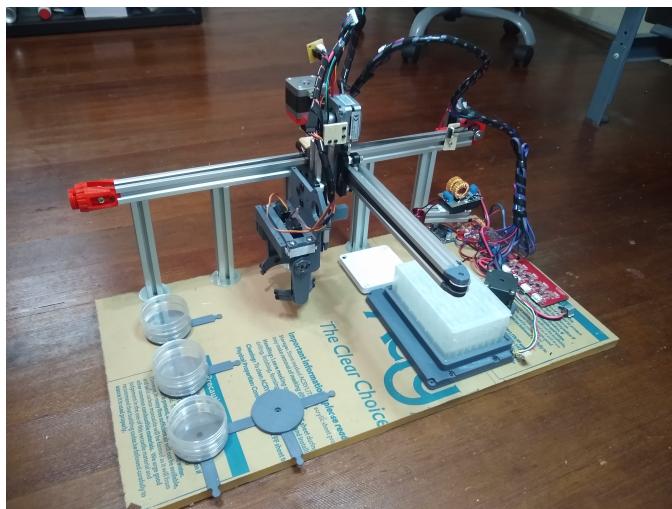
2.7. ábra. SciRobotics

PetriPlater és Pickolo – SciRobotics: A PetriPlater spirálmintázatban adagolja a mintát, míg a Pickolo automatikusan azonosítja és kiválasztja a kívánt kolóniákat, ezzel

egy teljesen automatizált tenyésztő és válogató rendszert alkotva [7].

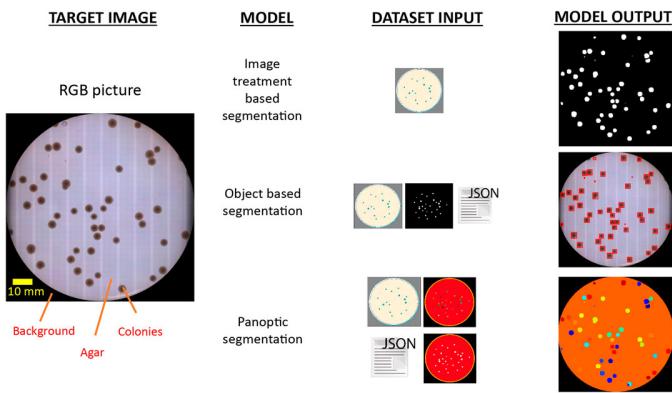
2.2. Nyílt forráskódú megoldások

A laboratóriumi automatizálás iránti növekvő igény és a költséghatékony szempontjai számos nyílt forráskódú kezdeményezést hívtak életre. Ezek a projektek lehetőséget biztosítanak arra, hogy alacsonyabb költségvetésből, testreszabott automatizált rendszereket hozzanak létre különböző kutatási igényekre szabva.



2.8. ábra. BioPick eredménye

A projektünk kiindulópontjaként a *BioPick* nevű nyílt forráskódú fejlesztést vettük alapul, amelynek célja egy automatizált rendszer megvalósítása volt Petri-csészék kezelésére és megfigyelésére [8]. Nagy előnye a projektnak, hogy egy 3D nyomtató vázat könnyedén át lehet alakítani BioPick célokra, így ez a komplexitást tovább csökkenti. Bár a BioPick projekt nem készült el teljesen, a szoftveres megoldásai – különösen a Python alapú vezérlés és a G-kód kommunikáció – hasznos kiindulópontot jelentettek számunkra. Mi majd egy hasonló rendszert szeretnénk egy ipari robotba beépíteni. A BioPick sajton nem tartalmaz kélpfeldolgozást. Így jön képbe a következő nyíltforrású hardveres és szoftveres megoldás.



2.9. ábra. Szegmentálási módszerek evolúciója agarlemez-elemzéshez

Open-source colony picker platform: Egy moduláris, nyílt forráskódú rendszer, amely képfeldolgozással támogatott kolóniakiválasztást, valamint robotkar-vezérlést tesz lehetővé, Arduino és Raspberry Pi alapokon [9].

3. fejezet

Tervezés

3.1. Választott megoldások indoklása

A projekt célja egy olyan automata rendszer létrehozása volt, amely képes ellátni Petri-csészékkel végzett biológiai kísérletek során felmerülő alapvető feladatokat, mint például a táptalajjal való feltöltés, a minták telepítése és a baktériumkolóniák növekedésének nyomon követése. A piacon elérhető rendszerek jellemzően zártak, drágák és korlátozottan testreszabhatók, ezért a saját megoldásunkat nyílt forráskódú, 3D nyomtatókhöz tervezett hardver- és szoftveralapokra építünk.

3.1.1. Felhasznált Szoftverek

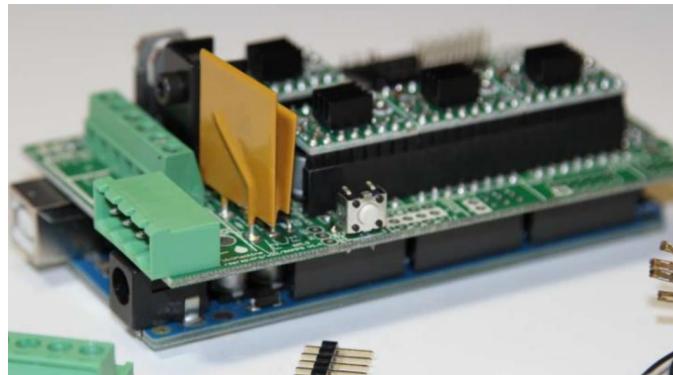
Python

A rendszer magasabb szintű logikáját és felhasználói felületét egy saját fejlesztésű **Py**
thon-alapú alkalmazás biztosítja, amely soros kommunikációt keresztül lép kapcsolatba a Marlin firmware-rel. A **PySerial** könyvtár segítségével megbízható adatkapcsolat építhető ki az Arduino és a számítógép között, míg a **PyQt** segítségével platformfüggetlen grafikus kezelőfelületet valósítottunk meg. Bár vannak más GUI-k amik gyorsabbak, de a PyQt adta lehetőségekkel kevés másik könyvtár tud versenyezni. A képfeldolgozási feladatokat az **OpenCV** könyvtár támogatja, amely számos eszközt biztosít a képek elemzéséhez, például a baktériumkolóniák szegmentálásához és nyomon követéséhez. Azért kiváló választás ez a könyvtár, mert a háttérben C++-ban implementált függvényeket használ, amelyek Pythonból is elérhetők, így ötvözi a magas teljesítményt a Python egyszerűségével. [10]

Marlin

A rendszer alacsony szintű vezérlését a **Marlin** firmware látja el, amely egy nyílt forráskódú, elsősorban 3D nyomtatókhoz fejlesztett vezérlőszoftver. Előnye a rugalmasság, a széles hardvertámogatás és az aktív közösségi háttér, amely megkönnyíti a testreszabást és a hibakeresést. A Marlin firmware közvetlenül értelmezi a **G-kód** parancsokat, és valós időben vezérli az **Arduino Mega 2560** mikrokontrolleren keresztül a **RAMPS 1.4** bővítőkártyára csatlakoztatott hardverelemeket. A Marlin G-kód lehet vezérelni, ami szöveges utasításokból áll, amelyek meghatározzák a gép mozgásait, sebességét, valamint a perifériák működését [11]. Ezeket a parancsokat a számítógépen futó, Python-alapú szoftver továbbítja a Marlin számára egy soros porton keresztül, tipikusan USB-soros kapcsolaton át. Egyszerre egy soros porton csak egy kapcsolat létesíthető.

3.1.2. Kiválasztott Hardver

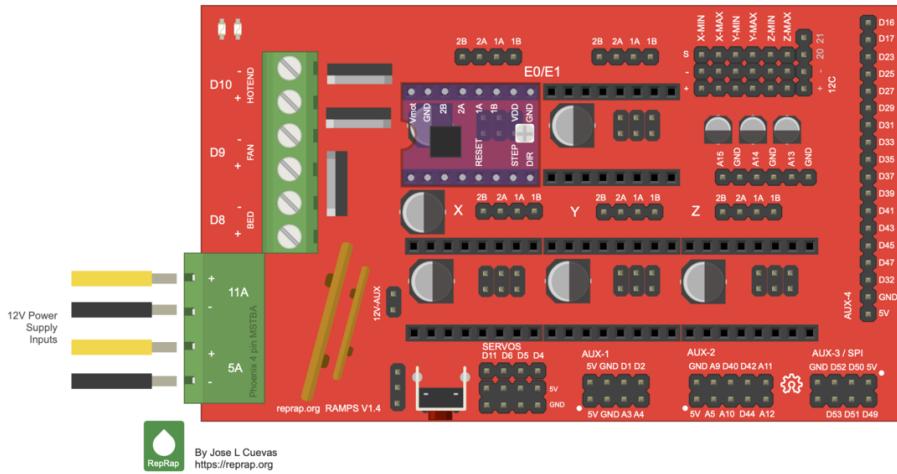


3.1. ábra. Arduino Mega + RAMPS 1.4

Arduino Mega 2560

Az **Arduino Mega 2560** egy nyílt forráskódú mikrokontroller alapú fejlesztőpanel. Nagy lábszáma és memóriakapacitása miatt kiválóan alkalmas összetett vezérlési feladatokra. Dokumentáció: [12]

RAMPS 1.4



3.2. ábra. RAMPS 1.4

A **RAMPS 1.4** (RepRap Arduino Mega Pololu Shield) egy kifejezetten az Arduino Mega 2560-hoz tervezett bővítőkártya [13], [14], amely megkönnyíti a perifériák csatlakoztatását és vezérlését. A hozzá tartozó dokumentáció részletes leírást ad a bekötésekről.

Jelentés	Példa	Funkciók
Analóg bemenet	A0 - A15	Szenzorjelek olvasása (pl. hőmérséklet, feszültség)
Digitális bemenet/kimenet	D0 - D69	Motorvezérlés, kapcsolók, LED-ek, SPI, I2C stb.

A Marlin firmware szigorúan szabályozza, hogy az Arduino Mega 2560 mikrokontroller egyes bemeneti–kimeneti lábait (I/O) hogyan lehet használni, annak függvényében, hogy milyen modulokat engedélyezünk a konfigurációs fájlokban (Configuration.h, Configuration_adv.h, pins_RAMPS.h). A digitális I/O lábak (D0–D69) közül csak azok érhetők el közvetlen vezérlésre, amelyeket a firmware nem foglal le rendszerszinten — például léptetőmotor vezérlésre, végálláskapcsolókra, fűtőelemekre vagy kommunikációs célokra.

A digitális lábak vezérlése korlátozott, mivel Marlinban a legtöbb kimenethez előre definiált funkciók tartoznak (pl. D10 a hotend fűtésére, D8 a bed fűtésére, D9 ventilátorhoz), és ezek működése csak a hozzájuk tartozó G-kód parancsokon keresztül történhet (pl. M104, M140, M106). Általános célú digitális kimenetek vezérlésére csak az úgynevezett „M42” parancs alkalmas, amely a firmware-ben opcionálisan engedélyezhető. Az M42

lehetővé teszi, hogy bármely szabadon hagyott digitális lábat manuálisan kapcsoljunk ki- és be. (Ezekre később a világítás és egyéb lehetőségek miatt van szükség.)

Az analóg bemenetek (A0–A15) jellemzően hőmérséklet-érzékelők csatlakoztatására szolgálnak. Ezek értékeit a Marlin automatikusan olvassa be és skálázza át a konfigurációban megadott karakterisztika szerint.

ATX tápegység



3.3. ábra. CODEGEN 300xx

A projekt során szükség volt egy megbízható tápegységre a rendszer komponenseinek energiaellátásához. Egy meglévő, elfekvőben lévő Codegen **CODEGEN 300xx 400W** típusú ATX tápegységet választottunk, amely bár nem rendelkezett részletes gyártói dokumentációval, az Intel ATX szabvány alapján meghatározhattuk a szükséges paramétereit. Továbbá, a rajta lévő címke elegendő információt biztosított. Ezt apró módosítás után munkára tudtuk fogni. [15]–[17].

4. fejezet

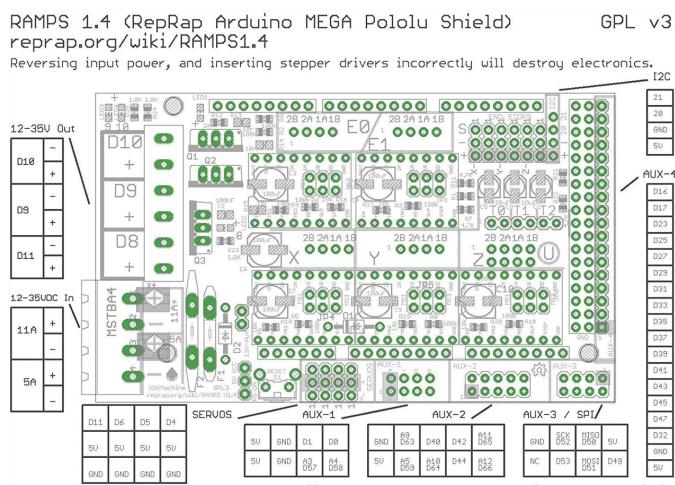
Eredmények

A rendszer felépítése kész hardverelemekből történt, amelyeket célirányosan illesztetünk össze. Az alábbi alfejezetek részletezik az egyes modulok integrációját, az ezekhez kapcsolódó beállításokat és a fejlesztés során szerzett gyakorlati tapasztalatokat.

4.1. Hardverösszeállítás és működés

4.1.1. ATX táp átalakítás

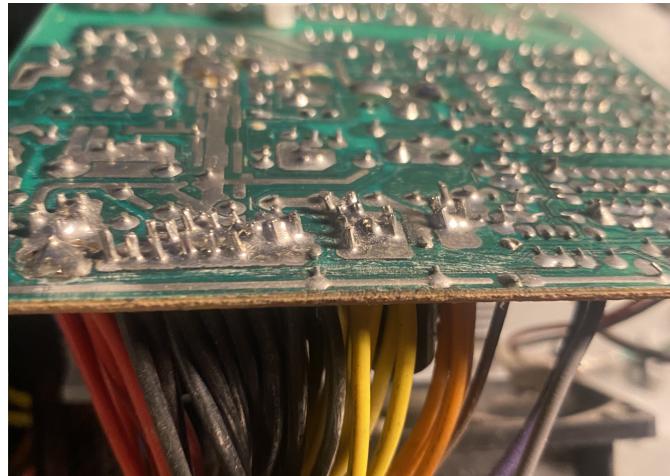
Ez elég intuitív feladat volt. Lényegében szekvenciálisan követtem a leírt lépésekét, kivéve a táp átalakításakor, mivel a RAMPS leírásban [13] nem térnek ki egy szerintem fontos dologra, ami a következő képen látszik:



4.1. ábra. RAMPS 1.4 csatlakozási és bekötési ábra

Ezen a képen az látszik bal alól, hogy 11A-re is szüksége lehet a "fűtött ágynak" (D8 kimenet) [17]. Mivel a baktériumok beültetése után a tü sterilitálására van szükség, és ezeket hővel szeretnénk megoldani, ezért a tervezés során biztosítani szerettem volna ehhez az erőforrásokat. Ahogy a tápegység címkéjén is látszik, 14A-t tud maximálisan kiszolgálni. 3.3

Viszont nekem ezzel volt egy problémám: az ATX tápok – főleg a régebbiek – esetén egyetlen 18 AWG sárga vezeték tipikusan 6A-t bír biztonságosan [18]. Erre egy könnyű megoldás a terhelés megosztás, de több sínes táp esetén nem javasolt párhuzamosan kötni két vezetéket, ha másik sínről származnak. Az előbb említett címke szerint a tápegység azt állítja, hogy fizikailag elkülönített 12V1 és 12V2 ág van. Mégis, szétszedés után azt tapasztaltam, hogy az egész egy közös sínen van.

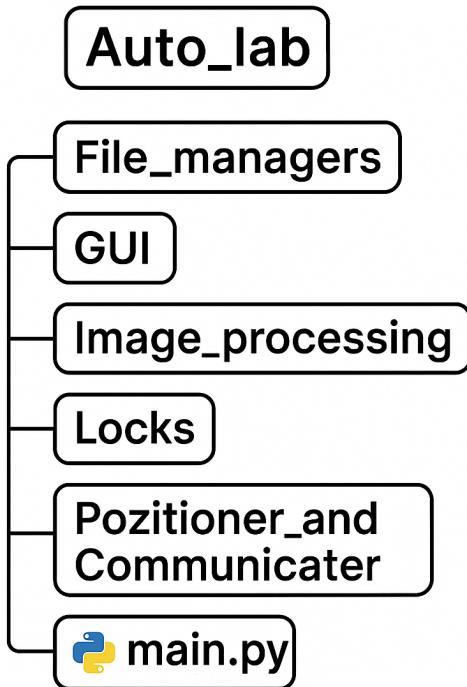


4.2. ábra. CODEGEN 300xx 400W közös +12V sín forrasztási pontjai

A vizsgálat alapján egyértelművé vált, hogy a kiválasztott két sárga vezeték közös sínről táplálkozik (mivel csak egy sín van), ezért azok párhuzamos kapcsolása biztonságosan megvalósítható.

Ezt követően az alkatrészeket az előzetes tervek szerint összeállítottam [13], [14], majd a rendszert USB-n keresztül, soros porton csatlakoztattam a számítógéphez. Miután rátelepítettem az Arduino-ra a Marlin-t, utána Pythonból vezérelhetővé vált az eszköz. Ezt követően nekikezdtem a GUI és az alap könyvtár építésének.

4.2. Megvalósított szoftver



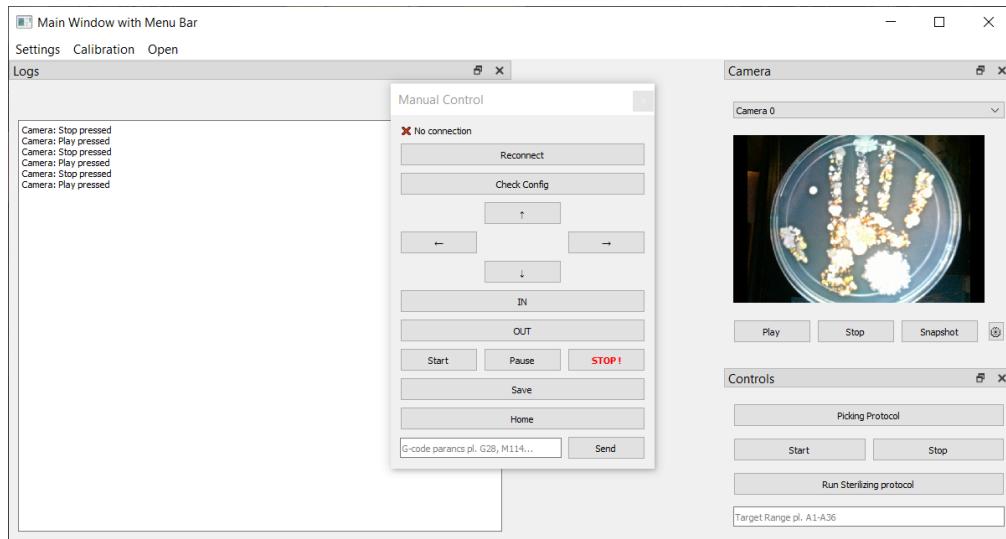
4.3. ábra. Python könyvtár

A szoftver több funkcionális egységre tagolódik, amelyeket külön könyvtárak és modulok valósítanak meg. A felhasználói felületet a `PyQt5` segítségével valósítottuk meg, ahol a fő komponenseket `QMainWindow`, `QWidget` és `QDockWidget` osztályok segítségével rendeztük el. A menürendszeret és az interaktív elemeket `QAction`-ökön keresztül kezeljük. A rendszer külön GUI-panelekben jeleníti meg a vezérlést, állapotinformációkat, képmegjelenítést és a G-kód küldést, ezzel támogatva a kényelmes és áttekinthető kezelhetőséget.

A felhasználói felület folyamatos és akadásmentes működéséhez a hosszabb ideig futó műveleteket – mint például a mikrokontrollerrel való kommunikáció vagy a képfeldolgozás – külön szálakon futtatjuk. Ehhez a `QThread` osztályt használjuk, amelyet a PyQt szignál-slot mechanizmusával egészítünk ki, így a háttérszálak biztonságosan kommunikálhatnak a főszállal. A párhuzamos szálak közös erőforrásokhoz való hozzáférését a `"Locks"` könyvtár biztosítja, amely zárolási mechanizmusokkal (pl. `threading.Lock`) védi az adatokat, különösen olyan kritikus pontokon, mint a soros port vagy megosztott adatstruktúrák használata. A főbb feladatokat most pontokba szedve ismertetem.

4.2.1. Felhasználói felület (GUI)

A Python és **PyQt** [19] alapú GUI lehetővé teszi a rendszer kényelmes kezelését. Alapvetően a PyQt adta lehetőségek közül a Dockereket használjuk. Ez azért előnyös, mert a widgeteket egymásba tudjuk ágyazni, és elrendezésüköt könnyen tudjuk menedzselni. A tervezés során az alapötlet az volt, hogy legyen egy "main_window", és ő hozza létre a benne található Dockereket. Így minden hasznos információnak ő lesz a központja és elosztója. Jelenleg így néz ki a GUI:



4.4. ábra. Main widget

4.2.2. Kommunikáció és vezérlés

A GUI és a mikrokontroller (Marlin firmware) közötti kommunikáció soros porton keretrendszerrel történik. Az adatátvitel a **pySerial** [20] könyvtár segítségével valósul meg. A "G-kódokkal" való kommunikációt a "Positioner_and_Communicator" könyvtár valósítja meg. A "G_communicate.py" fájl küldi a G-kódokat, és fogadja a válaszokat. Ez egy külön szál, ami a program indításakor indul. Eszköz csatlakozásakor betölti az előzetesen elmentett konfigurációt (ha van), és ezeket az alapbeállításokat elküldi az eszköznek. Ha csatlakozunk az eszközhöz, a következő alszálakat indítja el:

Szál neve	Queue neve	Továbbított G-code parancsok
<code>self.x_thread</code>	<code>x_motor_queue</code>	Csak G0 vagy G1 parancsokat, ha azok kizárolag az X tengelyt érintik (pl. G1 X100 F1500).
<code>self.y_thread</code>	<code>y_motor_queue</code>	Csak G0 vagy G1 parancsokat, ha azok kizárolag az Y tengelyt érintik (pl. G1 Y50 F1500).
<code>self.aux_thread</code>	<code>aux_queue</code>	Csak M42 parancsokat, amelyek digitális kimeneteket vezérelnek (pl. M42 P58 S200, M42 P58 S0).
<code>self.control_thread</code>	<code>control_queue</code>	Minden egyéb G-code parancsot, pl. (M119), konfiguráció (M906, M204), szüneteltetés (M0) és motorleállítás (M18).

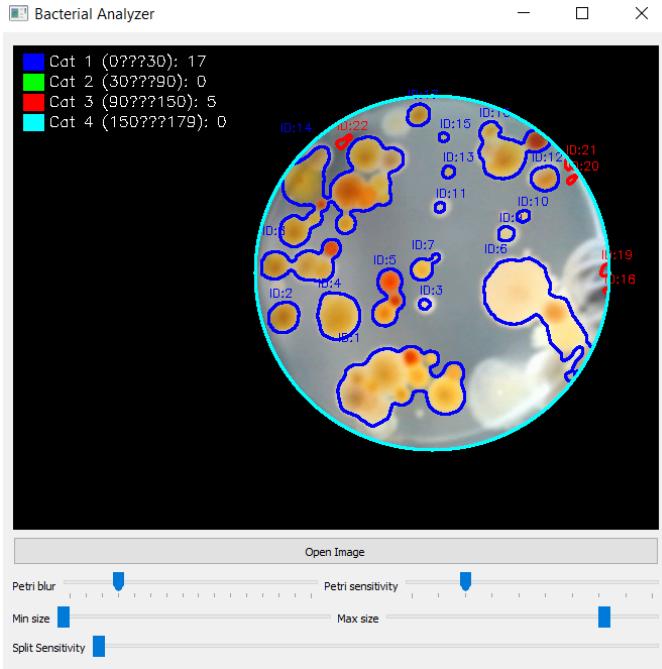
4.1. táblázat. A GCodeControl szálai és a Marlin felé továbbított parancsok

Erre azért van szükség, mert így egyrészt a buffer méretét nem fogjuk átlépni (a Marlin alapkonfigurációjában 12 utasítást tud eltárolni a buffer). Másrészt a válaszokra külön tudnak figyelni, és saját listát tudnak menedzselni a még ki nem küldött parancsokról.

4.2.3. Képfeldolgozás

Az OpenCV [21] egy olyan könyvtár, ami az alapvető képfeldolgozási és szegmentálási feladatokra jó eszközöket biztosít. Ez később még bővílhet. A jövőben ez lesz a legnagyobb feladat, hogy jól definiáltan és menedzselhető módon működjön. Ez a feladat jelenleg az "Image_processing" mappa alatt történik. Alapvetően három feladatot kell ellátnia a program ezen részének:

- Petri-csésze detektálása és a kívül eső részek levágása
- Baktériumok detektálása és kategorizálása
- Baktériumok növekedésének nyomon követése



4.5. ábra. Petri-csésze detektálás

Ezen feladatok közül egyelőre csak az első működik jól. A képen az látható, hogy a Petri-csészét jól detektálja. A kép széleit levágja (feketére festi), így csak a hasznos terület marad meg, ahol releváns információ van. Ezt követően a nagyobb összefüggő területeket szín szerint szegmentálja. Ennek fejlesztése lesz a későbbi cél: hogy ne csak szín, hanem alakzat alapján is tudjunk szűrni. Így a különböző kolóniák jól megkülönböztethetők lesznek.

4.2.4. Konfigurációs fájlok

Az applikáció beállításait célszerű menteni, és későbbi betöltéskor ezeket a beállításokat visszatölteni a jobb felhasználói élmény elérése érdekében. Ebben a **yaml** [22] könyvtár nyújt könnyen kezelhető lehetőséget. Ez a formátum jól olvasható, könnyen kezelhető és jól menedzselhető. Ilyen konfigurációs beállításokat használunk például: kamera beállításához, a csatlakozni kívánt eszköz paramétereihez, valamint a képfeldolgozás során.

Így ezeket csak akkor szükséges újra beállítani, amikor változtatni szeretnénk rajtuk.

5. fejezet

Összefoglalás

Feladat az volt, hogy egy működő, bővíthető, kutatási célra alkalmas szoftveres-hardware alapú rendszert kellett építeni, amelyre a jövőben már csak a konkrét alkalmazás logikáját kell ráépíteni.

5.1. Eredmények

A projekt során mind szoftveres, mind hardveres irányban sikerült elérnünk a kitűzött célokat. A hardver alapvető működése biztosított, további bővítése azonban már nem informatikai jellegű feladat. A szoftver oldalon létrejött egy jól strukturált, bővíthető alaprendszer, amely stabil alapot nyújt a jövőbeni fejlesztésekhez. Innentől kezdve a fókusz már kizárolag a konkrét, alkalmazás-specifikus funkciók megvalósítására helyezhető.

5.2. Jövőbeli tervezek

A következő nagy feladat a mozgások pontos inicializálása lesz a teljes rendszer mechanikai összeszerelését követően. Emellett a képfeldolgozó modul teljes kiépítése és a baktériumkolóniák növekedésének automatikus nyomon követése, valamint annak dokumentálása is kiemelt szerepet kap. Innentől kezdve már csupán a rendszer különálló komponenseinek összehangolt működése — azaz „egy teljes kép” megalkotása — van hátra. Meggyőződésem, hogy némi finomhangolással ez egy jól használható, testreszabható és költséghatékony laboratóriumi eszközökévé válhat, amely valódi segítséget nyújthat a kutatók és fejlesztők számára.

Hivatkozások

- [1] Tecan, *Sample plating and colony-picking application packages*, <https://www.tecan.com/colony-picking-package-offer>.
- [2] H. Company, *EasyPick – Reliable colony picking on the Microlab STAR platform*, <https://download.hamiltonsupport.com/wl/?id=bAgnQwJfazC8EMiGSqRm48KoJC>
- [3] M. Devices, *QPix 400-Series Microbial Colony Pickers*, <https://www.moleculardevices.com/products/clone-screening/microbial-screening/qpix-400-series-microbial-colony-pickers>.
- [4] S. Instruments, *PIXL – Precision Microbial Colony Picker*, <https://www.singerinstruments.com/solution/pixl-precision-microbial-colony-picker/>.
- [5] H. Robotics, *Universal Colony Picker Offers Automated and Manual Functionality*, <https://clpmag.com/lab-essentials/lab-automation/universal-colony-picker-offers-automated-manual-functionality/>.
- [6] Interscience, *ScanStation 300 – Real-time incubator and colony counter*. cím: <https://www.interscience.com/en/products/real-time-incubator-and-colony-counter/scanstation-300>.
- [7] SciRobotics, *PetriPlater and Pickolo – Petri dish processing system*. cím: <https://scirobotics.com/petridish-processing/>.
- [8] Dr3y, *BioPick – Open-source Petri dish automation system*. cím: <https://github.com/dr3y/BioPick>.
- [9] J. Doe és A. Smith, „Open-source automated colony picker for high-throughput applications“, *Frontiers in Bioengineering and Biotechnology*, cím: <https://www.frontiersin.org/journals/bioengineering-and-biotechnology/articles/10.3389/fbioe.2023.1202836/full>.

- [10] OpenCV Doc. Team, *OpenCV: Python Bindings Basics*, https://docs.opencv.org/4.x/da/d49/tutorial_py_bindings_basics.html.
- [11] Marlin Firmware Project, *Marlin G-code Reference*. cím: <https://marlinfw.org/meta/gcode/>.
- [12] Arduino Documentation, *Arduino Mega 2560 Datasheet*. cím: <https://docs.arduino.cc/resources/datasheets/A000067-datasheet.pdf>.
- [13] RepRap pdf, *RAMPS 1.4 Assembly Guide (PDF)*. cím: https://www.reprap.org/mediawiki/images/0/06/RAMPS_dossier.pdf.
- [14] RepRap Wiki, *RAMPS 1.4 Wiki*. cím: https://reprap.org/wiki/RAMPS_1.4.
- [15] TechnologyUK, *The Power Supply Unit - Computer Hardware*. cím: <https://www.technologyuk.net/computing/computer-hardware/power-supply-unit.shtml>.
- [16] Arduino Forum, *Wiring ATX for RAMPS 1.4*. cím: <https://forum.arduino.cc/t/arduino-mega-2560-ramps-1-4-atx-power-5v-and-12v/345667>.
- [17] RepRap Wiki, *Modify PC Power Supply*. cím: https://reprap.org/wiki/PC_Power_Supply.
- [18] Intel Corporation, *ATX12V Power Supply Design Guide*, <https://studylib.net/doc/18064382/atx12v-power-supply-design-guide>, Accessed: 2025-05-18, 2000.
- [19] Python Software Foundation, *PyQt - Python Wiki*, <https://wiki.python.org/moin/PyQt>.
- [20] pySerial Developers, *pySerial Documentation*, <https://pyserial.readthedocs.io/en/latest/pyserial.html>.
- [21] OpenCV Developers, *OpenCV: Introduction to OpenCV-Python Tutorials*, https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html.
- [22] PyYAML Developers, *PyYAML Documentation*, <https://pyyaml.org/wiki/PyYAMLDocumentation>, Accessed: 2025-05-16, 2025.