Debreceni Egyetem, Informatikai Kar

Távirányítható autó és Android vezérlőszoftver

Konzulens:

Dr. Kocsis Gergely
Adjunktus

Készítette:

Makár Péter Ákos
Mérnökinformatikus szak

Debrecen
2022

Tartalom

1. Bevezetés

A témaválasztás során, olyan lehetőséget kerestem, ahol a lehető legtöbb területén dolgozhatok az informatikának. A célom egy olyan projekt elkészítése volt, amelyben foglalkozhatok hardverrel, szoftverrel és szükséges hozzá némi hálózati alapismeret is.

A dolgozat hardveres részének megvalósításához felhasználtam a Digitális technika és az Elektronika tantárgyak teljesítése során szerzett tapasztalataimat. A szoftveres résznél a Mobilmegoldások és a Szoftverfejlesztés mérnököknek tárgyak ismereteire volt szükségem. A szerver kódját pedig a Számítógépes hálózatokon tanultak segítségével oldottam meg.

A tervezés során olyan komponenseket szerettem volna használni, amelyekkel már rendelkezem, ez az oka, hogy a robot alapja egy régi távirányítós autó a szerver pedig egy egyszerű laptop.

A megvalósítás során többször is újra kellett terveznem a kódokat vagy a hardver fizikai megvalósítását. A szerverként használt laptop a java kódot sokszor nem tudta tökéletesen futtatni a sok párhuzamosan futó szál miatt. A távirányítós autó kicsi mérete miatt a Raspberrys időnként túlmelegedhet, és a kábelek elvezetése is aprólékos tervezést igényelt.

Az alap ötletet az Amazon raktáraiban használt raktározó robotok adták. Ezek az eszközök polcokat hordoznak a rakodófelületükön. A robotok vonal kódokat olvasnak a padlón és úgy haladnak az előre kijelölt útvonalukon. Más cégeknél a betonpadlóba ágyazott mágneseket követik a robotok. Szerintem az Amazon megoldása a jobb, mert így több információt lehet az eszköznek átadni.

Az én ötletem egy olyan eszköz elkészítése volt, amely képes egy üzletben egy QR-kódokkal kijelölt útvonalon közlekedni, és a megfelelő parancs olvasása után fényképet készíteni, vagy irányt váltani. Amennyiben a beolvasás nem sikerül, a felhasználó képes a szerveren keresztül a telefonjával interakcióba lépni és módosítani a robot haladását.

Ha az eszköz olyan területre téved, ahol nem áll rendelkezésre Wifi hálózat, akkor a kódokat követve mozog tovább. Az eszköz "agya" egy Raspberry Pi 4. Minden vezérlést ez az integrált áramkör végez. Egyik nagy előnye az alacsony fogyasztás, valamint az ehhez mért nagy számítási kapacitás.

A megvalósítás során arra törekedtem, hogy minden komponens bővíthető legyen a későbbiekben. Minden hardveres elem szabadon cserélgethető, de az egyes komponensek cserélése esetén a szoftvert kalibrálni kell.

Egy ehhez hasonló őrrobot alkalmazása esetén egy boltban, vagy raktárban erőforrásokat lehet megspórolni, hiszen az eszköz egészen kis helyen is elfér, képes bemenni a polcok alá vagy mögé és ott fényképeket készíteni.

Az eszköz nagy előnye a fix kamerákkal szemben, hogy több kamerát is kiválthat, hiszen képes a helyváltoztatásra. Így a megfigyelést nem igénylő területeken nem szükséges elhaladnia, míg a megfigyelést igénylő területeken frekventáltan is közlekedhet.

A szerver jelenlegi kapacitása 2 autó és 2 telefon klienst tud kiszolgálni, de egy erősebb hardverrel ellátott számítógép ennél jóval több eszközt is ki tud szolgálni

1.2 Eszköz céljának bemutatása

A robotnak képesnek kell lennie egy közepes méretű boltban egy előre meghatározott pályán haladnia és szükség esetén fényképeket készítenie a beépített kamerával. Egy gyengébb felbontású kamera segítségével haladás közben dolgozza fel az információkat, amelyeket QR kódokon keresztül kap meg. A másik nagyobb felbontású kamerával fényképeket készít és menti vagy továbbítja a szerver és a másik kliens felé.

A sebesség a PWM érték módosításával befolyásolható. Az alapértelmezett PWM kitöltés 30%. Az egyharmados kitöltöttség az áramforrás élettartamára is kedvezően hat. A PWM mértékét befolyásolja a talaj anyaga is, szőnyegen sokkal nagyobb teljesítmény szükséges, mint parkettán. Az álatalam használt autóban a fogaskerekek kopottsága miatt, az előre és a hátra haladáshoz szükséges kitöltési tényező nem egyenlő. (Hátra könnyebben halad az autó.)

Az irányváltáshoz egy villanymotort használ, amit szintén PWM elven lehet vezérelni. Itt magasabb kitöltési tényezőt kell használni, mert a kerekek forgatása közben a súrlódás nagy. A kerekek forgatása is nagyban függ a talajtól. A gumiabroncs a legnagyobb súrlódást a lakkozott parkettán fejti ki, míg a legkisebbet a csempén. Ha a kerekek elérték a maximális kitérést az egyenestől, a szoftver nem enged további kerékszögváltozást az adott irányba.

A teljesen kifordított kerék előbb egyenes állapotba áll vissza, ha utasítást kap (például.: max bal + jobb = egyenes) és még egy utasítás szükséges a másik szélsőértékre való beálláshoz.

A kamera képe alapján képes a QR kód távolságának meghatározására, ezt a bejövő kép és a QR kód méretének arányából számítja, 2cm-es pontossággal. A minimális QR kód beolvasási távolság a kamerától függ. A maximális QR beolvasási távolság nagyjából 230 cm ezt a kamera sajátosságai befolyásolják. Kamera legnagyobb látótávját szoftveresen is lehet állítani. A fényviszonyok, és a beolvasandó objektum mérete és minősége is hatással van a beolvasás sikerességére, ezért erre a felhasználási területen figyelni kell. A felismerési ráta maximalizálására a legjobb körülmények a hideg fényű fényforrások és a stabilan elhelyezett QR kóddal ellátott táblák, valamint a stabil, folytonos rázós felületektől mentes talaj. Amennyiben a talaj nem megfelelő a vertikális irányú rázkódás sokszor használhatatlanná teszi az adott képkockát.

A tápegység a környezettől és a végrehajtandó utasításoktól függően tud 5V egyenfeszültséget biztosítani a Raspberry-nek. Ideális esetben, amit a sima talaj és a nagy, jól felismerhető táblák, nagy egyenesek kevés kanyar, jelentenek akár másfél órás üzemidő is elérhető. Használható akkumulátor is, mert az alapot képező távirányítós autó elemmel is és akkumulátorral is működtethető.

A korai tesztekben a Raspberry tápját egy hordozható 5V 2A 10 000mAh teljesítményű akkumulátor látta el. Ez nem bizonyult hatékony konstrukciónak, mert a nagy súly miatt nem tudott mozogni megfelelően.

A fénykép elkészítése két módszerrel történhet:

1 Megállás után elkészül a fotó majd tovább halad. Ezzel a módszerrel minimalizálható az elmosódás, de időigényesebb a feladat végrehajtása. Célszerű rosszabb látási

viszonyok esetén használni. Álló és mozgó objektumokról szerzett információk rögzítésére is egyaránt alkalmas.

2 Haladás közben is készülhet a fénykép. Ezzel a módszerrel viszont van esély elmosódásra ami az adott képkocka használhatatlanságát is okozhatja. A végrehajtás kevésbé időigényes, de a minőség is gyengébb. A tesztek alapján többnyire álló objektumok fényképezésére alkalmas.

Amennyiben haladás közben készül a fénykép, figyelni kell arra, hogy nem szabad kanyarodás közben fotózni. Ilyenkor 2 irányú lehet az elmosódás. Mert a fordulás horizontálisan a rázkódás pedig vertikálisan okoz elcsúszást.

Az eredeti koncepció szerint a kódolvasó kamera a földet pásztázta volna, de az a magasság, ami az autón maximálisan elérhető nem volt elegendő a megfelelő fókusztávolság beállítására, így az autó 2 kamerája előre néz.

2.Hardver

2.1. Raspberry kliens bemutatása

Az egylapkás rendszer, vagy System on Chip (SoC / SOC) egy olyan integrált áramkör, amely egy számítógép, vagy más elektronikai rendszer összes komponensét egyetlen lapkán tartalmazza.

Egyes nagyobb teljesítményű változataik Linux vagy Windows rendszereket is futtathatnak, ha kompatibilisek a szükséges processzorcsaláddal, és a minimálisan előírt hardveres elvárásokat képesek teljesíteni. [5]

Az egylapkás rendszer célja, az integráció fokának növelése a gyártási költségek csökkentése és a kisebb rendszerek létrehozására való törekvés.

A Raspberry Pi egy bankkártya méretű BCM2835 alapú egylapkás rendszer. Különböző Linux disztribúciók futtatására képes. [1]

A hivatalosan ajánlott operációs rendszer a laphoz a Raspberry Pi OS (korábban Raspbian), ami a Debian Linux kifejezetten Raspberry Pi-re optimalizált változata. Minden Raspberry Pi modellben integrálva megtalálható [2]:

- CPU
- RAM
- GPU
- USB port
- MIPI Kamera interfész
- HDMI port
- 3,5 mm jack port
- memória kártya foglalat

- GPIO lábak
- 5V tápcsatlakozás

Az USB, HDMI GPIO portok mennyisége és minősége a különböző modellek között eltérő lehet. A CPU sebessége, a RAM mennyisége is változhat a modelltől függően. Az egységek fizikai mérete közel azonos, a Pi Zero-t leszámítva (ez körülbelül feleakkora).

Néhány Raspberry Pi eszköz jellemzője [2]:

1 Pi Zero: alacsony teljesítményű és árú eszköz, 1Ghz-es 1 magos processzorral és 512MiB RAM-mal, HDMI, USB és micro-USB porttal. Valamit egy 40 pines foglalattal, amely tartalmaz 26 darab GPIO, 2darab 3.3V-os, 2darab 5V-os és 8darab test lábat.

2 Pi 1 - 3: Normál teljesítményű, 700Mhz - 1.2Ghz processzorral, 512MB - 1024MB RAM-mal, HDMI, USB, micro-USB porttal szerelt. Tartalmaz egy 40 pines GPIO foglalatot. Wifi és Bluetooth kapcsolatra is képes.

3 Pi 4: Az elődeinél nagyobb teljesítményű. 4 magos Cortex-A72 64bites 1.5Ghz-es processzorral, 2 - 8 GB LPDDR4-3200 SDRAM-mal szerelt rendszer. 2.4 GHz és 5.0GHz IEE 802.11ac wifi, Bluetooth 5.0, BLE Gigabit Ethernet biztosítja a kommunikációt a többi eszközzel. 2 db USB 3.0, 2 db USB 2.0, 2 db micro-HDMI, MIPI DSI kijelző, MIPI CSI kamera port található rajta. A robotot egy ilyen integrált lapka vezérli.

Meg lehet különböztetni A / A+ / B / B+ modelleket. Ezek többnyire a portok számában, vagy azok teljesítményében különböznek.

2.2. GPIO (General-purpose input/output)

Általános célú bemeneti / kimeneti csatlakozó. Egy integrált, vagy elektromos áramkörön biztosít digitális csatlakozást a lábakon keresztül. Használható bejövő jelek fogadására és kimenő jelek vezérlésére is. Képes a felhasználó futási időben vezérelni. [3]

A GPIO lábaknak (pin) nincs előre meghatározva a felhasználás célja. A felhasználási célt az áramkör felhasználója határozza meg.

Több alkalmazási mód is lehetséges. Csak a GPIO láb elektromos időzítése, a szoftver sebessége befolyásolja a sebességét. Szabványos logikai jelszinteket használ, nem képes nagy teljesítmény átadására, tehát csak a vezérlő jeleket lehet vezérelni.

Néhány alkalmazási terület:

- LED állapotjelzők vezérlése
- más áramkörök működésének engedélyezése és letiltása
- fedélzeti kapcsolók leolvasása
- konfigurációs söntök leolvasása

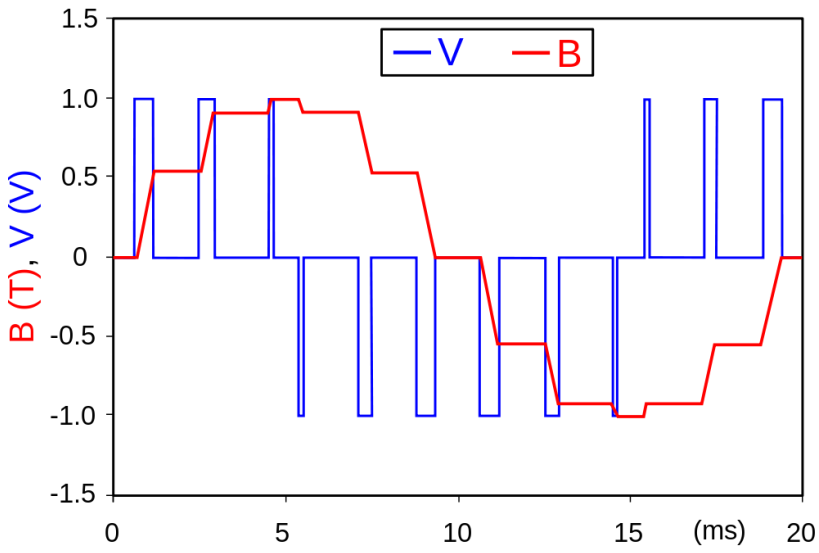
A GPIO-k alapvetően digitálisak, de gyakran használják őket lineáris folyamatok vezérlésére. Ilyen eset például a projekten a mozgást és fordulást végző motorok vezérlése. Ezt PWM-en keresztül lehet megtenni. A kimenő jel munkaciklusa határozza meg a folyamatvezérlő jel effektív nagyságát.

2.3. Impulzusszélesség-moduláció (PWM)

PWM, vagy pulse-width modulation Egy, az elektromos eszközök vezérléséhez használt széles körben elterjedt technológia. A korszerű teljesítményelektronika tett a gyakorlatban is használhatóvá [4].

A fogyasztóba táplált áramot és az átlagos feszültséget úgy állítják be, hogy azt gyors ütemben be és kikapcsolják. Minél nagyobb a bekapcsolt állapot aránya a kikapcsolthoz képest a fogyasztó annál nagyobb teljesítményt tud felvenni.

A be és kikapcsolások frekvenciájának sokkal nagyobbnak kell lennie, mint az a frekvencia, ami még hatással lehet a fogyasztóra. Egy lámpa fényerő szabályozásánál ez a frekvencia jellemzően 120Hz egy motorvezérlő esetében néhány kHz-től pár száz kHz-ig terjedhet.



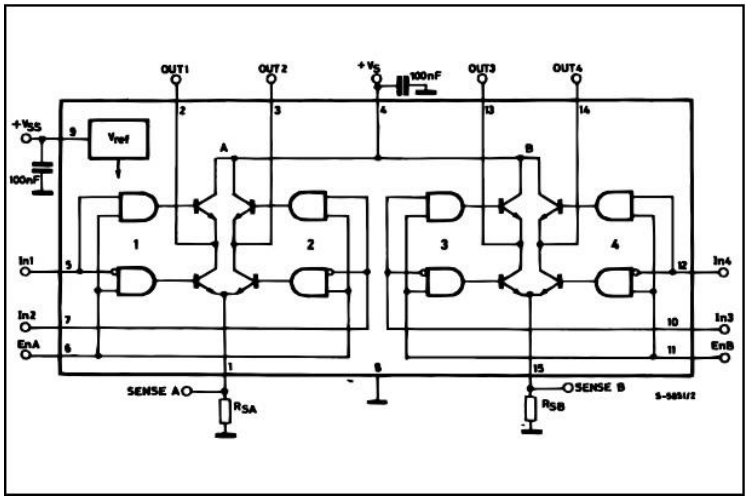
1. ábra Impulzusszélesség-moduláció [4]

A vonalfeszültséget (kék) a moduláció impulzusok sorozatává alakítja. Ez, szinuszos hullámformát indukál a fogyasztó áramkörében.

2.4. Autó részegységei

A robot 3 fő komponensből áll. A vezérlést végző Raspberry-ből, a távirányítós autóból és a benne található motorokból, valamint a kettőt összekötő vezérlőelektronikából.

L298 motorvezérlő: Egy integrált monolitikus vezérlő. Magas feszültségű, magas áramú vezérlő, ami a standard TTL logikai szintekhez készült. Feladata léptetőmotorok, mágnesszelepek, relék vezérlése. Két bemeneti foglalta van, amelyeken keresztül lehet engedélyezni, vagy tiltani a rájuk kötött eszközöket. A bemeneteket a különböző jelszintekkel lehet vezérelni, amit a GPIO lábakon keresztül végez a Raspberry.



2. ábra L298 block diagram [6]

Az L298 integráltan tartalmaz két teljesítménykimenetet. A teljesítmény kimenet vezérelhető PWM elven vagy folyamatosan, ez csak a bemenetek állapotától függ. Minden híd kapcsolatot a négy bemenet állapota határoz meg, melyek az In1; In2; In3; In4; EnA, és Eb. Az In bemenet vezérli a híd állapotát akkor, ha az En kapcsoló is engedélyezve van (magas feszültség érték van rajta). Alacsony feszültségi érték, azaz tiltás esetén gátolja az input kimenetet. Minden input TTL kompatibilis. Minden bemenetnek a lehető legrövidebb úton kell csatlakoznia a vezérlőjel forrásához, mert a hosszú vezetékek zajt vehetnek fel a környezetből. Be és kikapcsolás előtt az En lábnak alacsony, tiltó állapotban kell lennie. A kimeneti feszültség használható arra, hogy szabályozza a kimeneti áram értékét az amplitúdó levágásával vagy túláram elleni védelmet lehet biztosítani, a kimenetek alacsonyra állításával. [6]

A fék funkcióhoz (gyors motorleállítás) a maximális névleges teljesítmény nem lehet több mint 2 amper. Ha az ismétlődő áram maximuma több mint 2 amper, akkor párhuzamos konfiguráció is használható.

Külső diódákból álló hídra van szükség, ha induktív terheléseket hajtanak végre és az IC bemeneteire érkező jelek szaggatottak. (Shottky diódákat célszerű használni.) Ez a megoldás 3 Amperig képes működni egyenáramú üzemmódban és 3,5 A-ig ismétlődő csúcsáram üzemmódban.

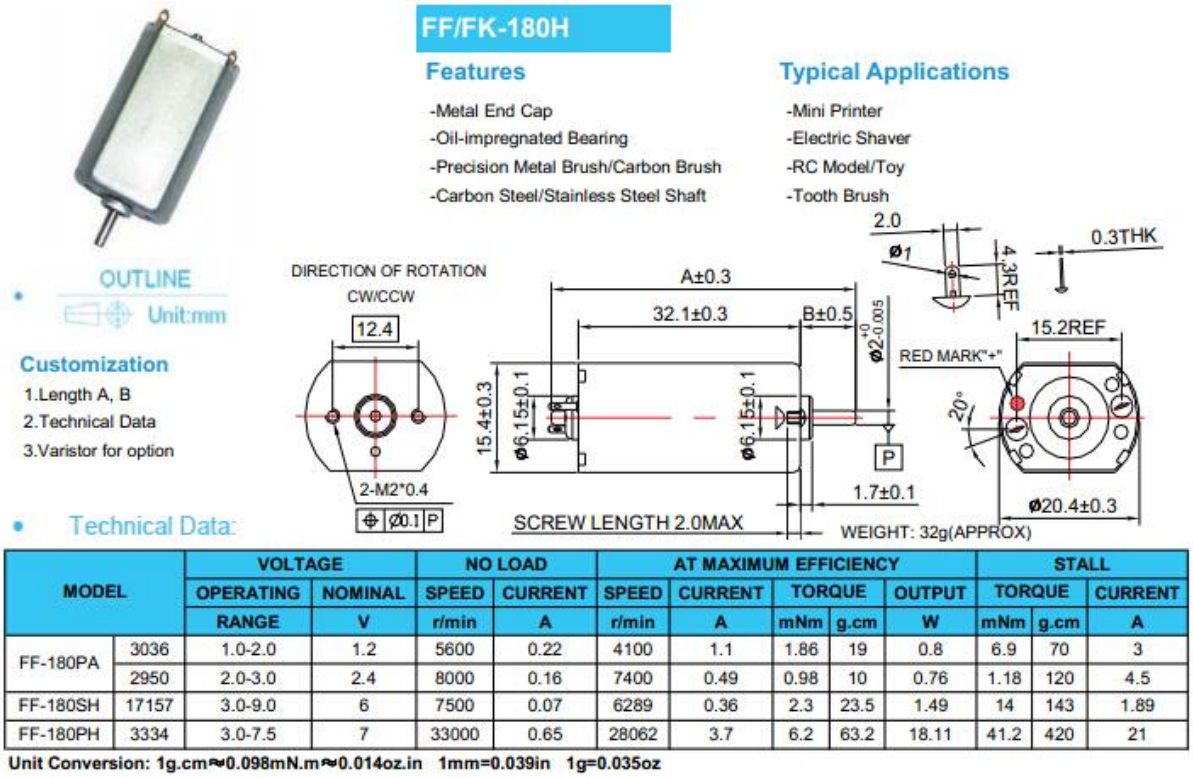
Guinness Mini DV Y2000 Webcam: Egy apró méretű webkamera. Feladata a robot előtt található QR kódok beolvasása. A felbontása alacsony 600 x 480 másodpercenként 30 képkockát továbbít, AVI videó formátumot használ a videóhoz. A videó Ezek a paraméterek ideálisak, mert az alacsony képminőség miatt könnyebb feldolgozni a képkockákat. A készített képeket a szoftver JPG formátumban menti el a háttértárra a megfelelő utasítás beolvasása után. Egy Mini USB - USB kábelen keresztül történik a kommunikáció. [7]

Egyes modellekben egy apró akkumulátor is található. A jelenlegi projektnél a tápellátást az USB port biztosítja. Képes egy max 32Gb-os SD kártyára rögzíteni, de ez a funkció nincs kihasználva. Méretei meglehetősen kicsik 26mm x 27mm x 26mm. Ez megkönnyíti a beszerelést.

3. ábra Guiness Mini DV Y200 Kamera [8]

ff-180sh DC motor: Az alapként használt távirányítós autó gyári motorja. Az olcsó kínai eszközökben ez a típus vagy valamely klónja eléggé elterjedt, de a nevükön kívül az összes főbb paraméterük megegyezik. Hasonló változatok még az FF-180PA, FF-180SH, FF-180PH. A változatoktól függően a teljesítménye 1 - 2 wattól majdnem 20 watt közötti lehet. Ez függ a típushoz rendelt optimális feszültségtől. A fordulatszám is akár 28 000 fordulat / perc is lehet, de ennél a projektnél jelentősen kisebb. A motor fordulatszáma ugyanis egy áttételezéssel le van csökkentve, így a nyomaték növekedik.

Az állandó mágnesek az álló részben helyezkednek el, időben és térben állandó mágneses mezőt biztosítanak a motor számára. A forgórész mágneses mezejét a forgás megegyező sebességű, de ellentétes irányban kell mozgatni az optimális működés érdekében. A mező mozgatásáért a kommutátor a felelős. Ez, egy szelvényezett komponens amely a tekercseket mindig úgy látja el árammal, hogy azok a megfelelő időben indukáljanak mágneses mezőt. [9]



MODEL		VOLTAGE		NO LOAD		AT MAXIMUM EFFICIENCY					STALL		
		OPERATING	NOMINAL	SPEED	CURRENT	SPEED	CURRENT	TORQUE		OUTPUT	TORQUE		CURRENT
		RANGE	V	r/min	A	r/min	A	mNm	g.cm	W	mNm	g.cm	A
FF-180PA	3036	1.0-2.0	1.2	5600	0.22	4100	1.1	1.86	19	0.8	6.9	70	3
	2950	2.0-3.0	2.4	8000	0.16	7400	0.49	0.98	10	0.76	1.18	120	4.5
FF-180SH	17157	3.0-9.0	6	7500	0.07	6289	0.36	2.3	23.5	1.49	14	143	1.89
FF-180PH	3334	3.0-7.5	7	33000	0.65	28062	3.7	6.2	63.2	18.11	41.2	420	21

Unit Conversion: 1g.cm≈0.098mN.m≈0.014oz.in 1mm=0.039in 1g=0.035oz

4. ábra FF/FK-180H [10]

3. Szoftver

3.1. Programozási nyelvek

Az Android és a szerver kód Java nyelven íródott, míg a Raspberry-n futó program Python nyelven készült. A Pythonra a kamerák miatt van szükség, ugyanis az OpenCV Pythonnal működik tökéletesen.

Java: A Java egy egy objektumorientált és funkcionális elemeket tartalmazó programozási nyelv, amelyet a Sun Microsystems a '90-es években fejlesztett egészen 2009-ig amikor az Oracle tulajdonába került a cég. Jelenleg az Apache fejleszti. A szerveren és a kliensen - mind az Androidos mind a PC kliens beleértve - Java nyelven készült el a kód. A nyelv egyik nagy előnye a platformfüggetlenség. A megírt kód hardvertől és operációs rendszertől függetlenül fut a Java Virtual Machine-nek (JVM) köszönhetően, mert a fordító bájtkódra fordít. A bájtkód közvetlenül futtatható a processzoron. A nyelv a szintaxisát a C valamint a C++ nyelvtől örökölte [11]. Ezt kihasználva lehet a kódot apróbb módosításokkal átültetni x86 alapú PC-ről az ARM processzoros Android operációs rendszerrel rendelkező telefonra, és a Raspberry Pi 4-re.

A projekt jellegéből adódóan szükséges volt az objektum-orientáltság használata. Az elképzelés szerint a szerver a különböző folyamatokat (adatok fogadása a klienstől, adatok továbbítása, szálak leállítása és indítása, kliensek kezelésének szétválasztása) különböző objektumokként kezeli, amelyek külön szálakon futnak és állnak le, ha szükséges.

A JVM-et a Linux rendszerekre (Ubuntu, Raspberry Pi OS) nem, de a Windowsra telepíteni kell, mert nem tartozéka az alap szoftvercsomagnak. A Java Standard Edition (Java SE) nem része egyik rendszernek sem, így a szoftver fejlesztéséhez ezeket is be kellett szerezni.

Python: A Python egy általános célú magas szintű programozási nyelv. A tervezési filozófia az olvashatóságot és az egyszerű alkalmazást tartja fontosnak, a futási sebességgel szemben. Funkcionális és Objektumorientált paradigmákat is támogatja. A típusokat és a memóriát dinamikusan kezeli. Interpreteres nyelv, így nincs a tárgy és a forráskód különválasztva. Python értelmező segítségével azonnal futtatható. Az értelmezőt számos operációs rendszerre elkészítették így széles körben alkalmazható. [12]

A Raspberry-n futó klienst újra kellett írni, mert nem volt alkalmas a kamera integrálására OpenCV-vel. Az új verzió Python nyelven készült el. A program szálainak funkciói majdnem teljesen megegyeznek a Java prototípusával, így sok felmerülő kérdést, problémát kész megoldásokkal lehetett elhárítani.

A Pyhton kiválóan alkalmasnak bizonyult a Java-nál felmerülő problémák orvosolására. Meg lehet valósítani az objektumorientáltságot, szálkezelést, socket programozást és az OpenCV segítségével integrálható a kamera modul is.

A Raspberry Pi OS rendszer alapértelmezetten támogatja a Python3-at, de szükséges volt az OpenCV telepítése, ami parancssorral könnyen megoldható. (A telepítésnél vigyázni kell, mert az OpenCV legújabb verziója nem telepíthető Raspberry-re így a projekt 3.4.16-ot használ az aktuális 4.5.4 helyett.) A Python meglehetősen pazarlóan bánik az erőforrásokkal, így az első próbálkozások a futtatással nagyon magas CPU használat volt megfigyelhető. Ezt a problémát a megfelelő kamerák kiválasztásával ki lehetett javítani.

OpenCV: Az OpenCV egy C++ nyelven írt ingyenes szoftveres képfeldolgozásért felelős függvénykönyvtár. A C++ alapok ellenére elérhető Python, Java és Matlab nyelven is. Ezen interfészek API-ja az online dokumentációkban érhető el. [13]

A projekten belül a Raspberry-n futó Python kliens használja a QR kódok tartalmának azonosítására. Ehhez, a cv2.QRDetector-t kellett alkalmazni. Könnyen integrálható python3-ban, képes kezelni az USB-n és a Raspberry alaplapján található kamera porton keresztül kapcsolt webkamerákat is. A tesztek alapján a Raspberry egy 600 x 800-as felbontású kamera képét tudja a legkönnyebben feldolgozni. Ez az aranyközépút, mert a nagyobb felbontással könnyebben megtalálja a szoftver az objektumokat, de a több adatot lassabban dolgozza fel, míg a kisebb felbontásnál gyorsabb a bejövő képek ellenőrzése, de a kevesebb részlet miatt nehezebben azonosítja az objektumokat.

A bejövő képeket a kívánt formátumban le lehet menteni a háttértárra (vagy socketen keresztül továbbítani). Mivel a képek mentése .png formátumban történik, a szerver és a telefon kliens oldalon nem szükséges az OpenCV használata.

3.2. Socket programozás

Socket: A számítógép-hálózatokban a socket egy Internet Protocol-alapú hálózati folyam eleme. Valamilyen számítógépes hálózatban például az interneten kétirányú adatkommunikáció végpontja.

A kifejezést a TCP/IP protokollkészlet (általában az operációs rendszer által biztosított) API-jának megnevezésére is használják. Az internet socketek hozzárendelik az adatcsomagokat a megfelelő alkalmazásokhoz és szálakhoz, a megfelelő port és ip kombinációjának segítségével. [14]

Socket elemei [14]**:**

1 A socket címe, ami egy IP cím, ami egy eszközhöz csatolja
2 A socket portja ami egy alkalmazáshoz kapcsolja az adatokat. Ezzel a két adattal egyszerűen azonosítható az interneten az alkalmazások.

Socketek típusai [14]**:**

1 Raw socket: routerekben és egyéb hálózati eszközökben találhatóak. Itt a csomagok fejléce nincs levágva, az alkalmazások számára hozzáférhető kimarad a szállítási réteg
2 Stream sokcet: ezek a TCP protokoll szerint működnek, kapcsolat-orientáltak. Az adatcsomagok sorrendhelyesen továbbítódnak.

Datagram socket: kapcsolatmentes adattovábbításra valóak, az User Datagram Protocolt (UDP) alkalmazzák. Az adatcsomagok nem sorrendhelyesen továbbítódnak.

3

Szerver: Egy szerveroldali socket kommunikációt valósít meg. Fogadja a kliensek kéréseit és továbbítja a megfelelő cél kliensnek.
Kezdetben a "listening" állapotban várakozik, ilyenkor a netstat által jelzett távoli cím 0.0.0.0 a port pedig 0. Megszólítás után veszi fel a kliens adatait, majd egy új szálon "listening" állapottal indít egy új szerver socketet.

A szerver tehát egyidejűleg több különálló socketet is tud kezelni. Az operációs rendszer ezeket különálló socketekként kezeli. TCP kapcsolat esetén azonos IP-hez és porthoz több socket is tartozhat.

Kliens: Egy kliensoldali socket kommunikációt valósít meg. Fogadja a szerverektől kapott üzeneteket és a bejövő adatok alapján végrehajtja a kívánt utasításokat majd válaszol a szervernek. Ismeri a szerver IP-címét és a megfelelő portot és erre küldi az adatcsomagokat. A kapcsolatot a szerver megszólításával kezdi, majd a csatlakozás után a TCP protokoll szerint kommunikál a szerverrel, azon keresztül a másik klienssel.

A kliens 1 socket szálat kezel, ezen keresztül történik minden adatcsere. Nincs is szükség többre, hiszen egy klienshez egy szerver tartozik. UDP kapcsolat esetén nincs állandó

kapcsolat, az adatok nem sorrendhelyesen továbbítódnak. A TCP kapcsolat esetén egyszer létrejön a kapcsolat majd ezen a csatornán keresztül történik az adatcsere.

A socket kommunikáció szempontjából nincs jelentősége, hogy Python vagy Java nyelven íródott a programkód

3.3. QR-kód

Az eszköz a navigálást QR-kódok olvasásával teszi. A kódok tartalmazzák a megfelelő üzeneteket a kormány és a mozgást biztosító motorok vezérléséhez. A rendszer ezeket a kódokat egy kamera segítségével érzékeli, majd a kliens program dolgozza fel és adja ki a GPIO lábakra a megfelelő utasításokat. A QR-kód (Quick Response-kód) magyarul gyors válasz egy olyan vizuális kódolási forma (pontkód), amely a kódolandó üzenetet egy meghatározott struktúra szerint pixelekké alakítja át. A működése hasonló a vonalkódokéhoz, de több információt lehet benne tárolni. A név utal a gyors dekódolási sebességre és a felhasználó által igénybe vett gyors reakcióra. [15]

A fejlesztés 1994-ben kezdődött a japán Denso Wave cég által. A széleskörű elterjedésének köszönhetően ma már a legtöbb mobiltelefon képes ezeket a kódokat felismerni és a benne található utasításnak megfelelően eljárni.

Bármilyen irányból olvasható, köszönhetően a három pozícionáló mintáról a kód három sarkában. Ezek a pozícionáló mintázatok azonosak minden kódban, szabvány szerintiek. Akkor is értelmezhető az üzenet ha a kód ferde.

5. ábra QR kód váz, adatok nélkül [15]

A kódbélyegek skálázhatóak, amit Verziók jelölnek Verzió 1-től egészen Verzió 40-ig. A verziók közötti különbséget az adattárolási képesség és a hibatűrési határ adja. Jelenleg a leginkább elterjedt változat a 2008-ban bevezetett "Level L". Az elterjedés oka, hogy ez a szabvány már egészen sok információt képes tárolni, ellentétben az egydimenziós vonalkóddal.

Adattárolási képességek:

- Számok:7089 karakter
- Betűk: 4296 karakter
- Bináris, 8 bites adatok: 2953 bájt

- Kandzsi / Kana: 1817 karakter

Hibatűrési képességek:

- Level L: max 7% veszteség
- Level M: max 15% veszteség
- Level Q: max 25% veszteség
- Level H: max 30% veszteség

Ezek a veszteségek a megadott hibahatáron belül visszaállíthatók. A hiba itt a nem megfelelő scannelésből ered. Ha a beolvasás nem tökéletes akkor a képpontok nem mindig láthatók tökéletesen, esetleg nem lehet őket megkülönböztetni.

A hibajavítás a kódba szabvány szerint előre definiált kiegészítő jelzésekkel történik, ennek segítségével a dekódoló program bizonyos szintű torzulást képes helyreállítani.

A torzulás oka lehet:

- Képfelbontási korlát
- Nem megfelelő fényviszony
- A bélyeg kis mérete
- Kopás vagy szennyeződés a kód felületén

A nagy népszerűsége miatt 2000-ben ISO/IEC 18008 nemzetközi szabvánnyá vált. A kód ingyenesen használható nyílt szabvány, minden szabadalom a Denso Wave tulajdona, de bárki szabadon felhasználhatja ingyenesen.

Van lehetőség a kód esztétikai megjelenésén is módosítani, de ez sokszor rosszabb felismerést okoz. A kód színezése, logokkal ellátása megengedett a magas hibatűrési határ miatt. Ilyen módosításokat akkor célszerű eszközölni ha a kód valamilyen reklám grafikus részét képezi. [15]

3.4. Vezérlést végző QR kódok

A projekt célja, egy olyan távirányítható eszköz elkészítése volt, amely képes egy bolti őrrobot funkcióit ellátni. Az eszköz feladata egy közepes méretű bolt bejárása, és a falakra elhelyezett QR kódokból kinyert utasítások végrehajtása.

Képes ezen kívül szükség esetén egy Androidos telefonról utasításokat fogadni (ezek többnyire a QR kódból kinyerhető utasítások) és azokat végrehajtani. A külső féltől származó utasításokat egy PC-n futó szervertől fogadja.

A kinyomtatott QR kódok a következő utasításokat tartalmazzák:

1 **FORWARD:TRUE:** Ilyenkor PWM vezérelve elindul az autó, annak érdekében, hogy a további kódokat könnyebben be tudja olvasni csak 30%-os kitöltöttségű PWM-et van beállítva, de ez módosítható

2 **FORWARD:FALSE:** Ilyenkor az autó abbahagyja az előre haladást. A motorban található fogaskerekeknél fellépő súrlódás miatt azonnal megáll, nem szükséges fékezni, de egy 10%-os kitöltöttségű tolatással ez is elérhető.

3 **BACKWARD:TRUE:** Ilyenkor PWM vezérelve elindul az autó hátra. A motorban található fogaskerekek kopottságából adódóan itt 50%-os PWM-et kellett alkalmazni, de ez is módosítható.

4 **BACKWARD:FALSE**: Ilyenkor az autó abbahagyja a tolatást.

5 **LEFT:TRUE** : Ilyenkor elforgatja a kerekeket PWM vezérelve balra ügyelve arra, hogy ne terhelje a végpont elérése után a kormányzásért felelős motort. Amennyiben a kormány a jobboldali maximális végállásban van akkor egyenesbe forgatja a kereket.

6 **LEFT:FALSE:** Ez egy elkészített, de sosem használt funkció. Abbahagyja a kerekek balra forgatását.

7 **RIGHT:TRUE:** Ilyenkor elforgatja a kerekeket PWM vezérelve jobbra ügyelve arra, hogy ne terhelje a végpont elérése után a kormányzásért felelős motort. Amennyiben a kormány a baloldali maximális végállásban van akkor egyenesbe forgatja a kereket.

8 **RIGHT:FALSE:** Ez egy elkészített, de sosem használt funkció. Abbahagyja a kerekek jobbra forgatását.

9 **PICTURE_SEND:** Fejlesztés alatt. Készít egy képet az erre a célra felszerelt kamerával és elküldi a kliensnek.

A robot vezeték nélküli internet segítségével képes kapcsolódni egy szerverre, de csak helyi hálózat (LAN) esetén. A szerver fejlesztése Windows operációs rendszer alatt történt NetBeans, Maven felhasználásával. A további fejlesztési fázisokban Linux operációs rendszer alatt volt használva, nincs tapasztalat működést megakadályozó hibával kapcsolatban.

A szerver célja, hogy igény esetén több robot és telefon-kliens is képes egymással kommunikálni, ezen kívül a képküldés is itt valósul meg, így a bejövő adatok szabadon módosíthatók például.: a készített képeket a szerveren is le lehet tárolni. A beérkező képet továbbítja a célként kiválasztott kliensre.

3.5. Raspberry kliens Java prototípus

A Raspberry-n futó kliens első változata, egy Java nyelven írt program. A Java kód könnyedén mozgatható a különböző operációs rendszerek között (Windows, Ubuntu, Raspberry OS, Android), így sok időt lehetett nyerni a fejlesztésével. Java swing elemekkel készült hozzá egy egyszerű grafikus felület is, amely csak a felhasználást könnyíti meg de az összes funkció elérhető a konzolablakból is.

Indulás után bekéri a szerver IP-címét, a saját és a cél kliens felhasználónevét. Megkísérel csatlakozni a szerverre, ha ez sikeres elküldi a saját felhasználónevét, hogy a későbbiekben azonosítani lehessen. Ehhez socket kapcsolatot használ.

A kapcsolódáshoz szükséges adatokat alapértelmezetten a grafikus felületről kéri be, a nem megfelelő vagy hiányos kitöltés esetén figyelmezteti a felhasználót.

Tesztelési céllal lehetőség van szöveges üzenetek küldésére és fogadására. A felhasználó a küldeni kívánt üzenetet egy erre a célra elhelyezett szövegdobozba írhatja be, amit a küldés gombbal tud a szerverre eljuttatni.

A kliens képes a fogadott üzenetek elemzésére. Ha a bejövő üzenet végrehajtható utasítást tartalmaz akkor végrehajtja azt. Ilyenkor az előre beállított GPIO lábakat vezérli. Mivel ez a kód csak a fejlesztés első szakaszában volt használva, ezért nincs benne PWM, de a végleges változatban már szerepel.

A GPIO lábak vezérléshez a következő csomagokat kell használni [16]:

- com.pi4j.io.gpio.GpioController;
- com.pi4j.io.gpio.GpioFactory;
- com.pi4j.io.gpio.GpioPinDigitalOutput;
- com.pi4j.io.gpio.PinState;
- com.pi4j.io.gpio.RaspiPin;

Ezek a csomagokat az internetről kell letölteni és feltelepíteni a Linux rendszernek megfelelően. Java Runtime, WiringPi Native Library szükséges hozzá.

A kódba való importálás után a dokumentációnak megfelelően használható. Be kell állítani a vezérelni kívánt lábakat, majd a beépített függvényekkel lehet be és kikapcsolni a lábakat. [17]

A Raspberry-re rögzített kamera képének feldolgozását, OpenCV segítségével terveztem megvalósítani, de ennek Java környezetbe való integrálása akadályokba ütközött. Gondok adódtak a telepítés során, majd a kamerát nem észlelte a csomag. Rövid kutatás után arra a döntésre jutottam, hogy az OpenCV Python nyelvvel való könnyű integrációja miatt elkészítem a kliens Python változatát.

3.6. Raspberry kliens Python

A második változat Python nyelven íródott. A Raspberry OS alapértelmezetten biztosít a Python nyelvű fejlesztéshez minden szükséges eszközt (Thonny, Python fordító). Ennek a változatnak nagyobb erőforrásigénye van, így az újraírás során erre figyelni kellett.

A Python könnyű érthetősége és egyszerű felépítése miatt nagyon hamar elsajátítható. Így a kliens viszonylag gyorsan elkészült. Ez annak is köszönhető, hogy a Java változat szerkezeti elemeiből sokat megtartott, így a logikát már csak át kellett ültetni egy másik programozási nyelvre. Ennél a változatnál már cél volt a kód átláthatóbbá tétele is, így a logikailag összekapcsolódó elemek csoportosítva lettek a nekik megfelelő osztályokba és az azoknak megfelelő forrásfájlokba.

A változat már tartalmazza az OpenCV csomagokat. Képes a kamera képét feldolgozni, igény szerint egy ablakban megjeleníteni. Az egyes képkockákon megtalálja a QR-kódokat és szöveges formátummá alakítja a tartalmukat.

Az OpenCV integrálása a Python nyelvbe az interneten található dokumentáció szerint történt:

1 Telepíteni kell az OpenCV csomagot a "pip" parancs segítségével
2 Ha szükséges az OpenCV korábbi változatait ennél a lépésnél lehet kiválasztani
3 (A „pip” a Python nyelv csomagkezelő szoftvere a segítségével lehet új csomagokat beszerezni)
4 A telepítés végeztével az OpenCV importálhatóvá válik a Python számára "import cv2"

5 Importálás után egy objektumot beállítunk a kamera stream-re (0 a beépített kamera 1 az USB-s külső eszköz)
6 Beolvassuk az adatokat a kamerától
7 A legegyszerűbb módszer a tesztelésre, ha megjelenítjük a képet

A Python kliens is több szálat használ, amelyek kommunikálnak egymással. Az egyes szálak funkciója megegyezik a Java változatban megtalálható változataikkal. Ennek a feladatelosztásnak az egyik nagy előnye, hogy így, ha az egyik folyamat valamilyen nem várt esemény miatt leáll, akkor nem okozza az egész program leállását. Egy esetleges új folyamattal akár figyelni is lehetne a szálakat és ha hibát észlel akkor a problémás folyamatokat újraindítja. Amennyiben a probléma ismételten jelentkezik akkor tájékoztatni lehetne a felhasználót vagy a fejlesztőt róla. Ez megkönnyíti a hibajavítás folyamatát.

A jelenlegi változat is tartalmaz egy kezdetleges hibakezelést, de csak az olyan eseményeket tudja kezelni, amiket ismer. Egy váratlan kamera leállást például nem képes felismerni (például az eszköz haladás közben beleakad valamilyen tereptárgyba és kihúzódik a kábel). Képes a szerverről való lecsatlakozást észlelni és erről a felhasználót tájékoztatni. Amennyiben a kamera a futtatás elejétől kezdve nem elérhető, vagy nincs megfelelően beállítva erről is értesíti a felhasználót.

3.7. Raspberry kliens felépítése

1 **`Main_class`**: Az összes további osztály és szál szülője. Feladata a szükséges konfigurációs lépések majd a funkciók végrehajtásáért felelős szálak elindítása.
   - Tájékoztatja a felhasználót az alap információkról, segít a helyes konfigurálásban.
   - Ellenőrzi az adatok helyességét, de akár úgy is beállítható, hogy mindent csináljon meg automatikusan. Mindezt parancssorból szöveges formában végzi.
   - Csatlakozik a szerverre (erről tájékoztatja is a felhasználót).
   - Elindítja a küldőszálat
   - Elindítja a fogadó szálat
   - Elindítja a kamerakezelő szálat
   - A szálaknak átadja a közös adatokat tároló objektumot, amely minden olyan adatot tárol, amit egynél több szálnak kell használnia.

   [18] [19] [20] [21] [22]
2 **`Static_variables:`** Egy olyan objektum osztálya, amely minden szálak közötti kommunikációhoz szükséges adatot tartalmaz. A benne található adattagokat az összes futó folyamat meg tudja figyelni, képes módosítani. A tárolt információk:
   - mozgással kapcsolatos adatok
   - a hálózati kommunikációhoz szükséges adatok (ip, port, felhasználónév, célkliens felhasználóneve)
   - első kerekek kitérésnek értéke
   - mozgás és fordulás PWM kitöltési tényezője

- GPIO vezérléshez szükséges lábak
- bejövő kimenő adatok

3 **`Reciever_class:`** Feladata a szervertől érkező utasítások fogadása. Folyamatosan ellenőrzi, hogy érkezik-e valamilyen adat. Ha van bejövő üzenet akkor azt dekódolja UTF-8-as kódolásnak megfelelően, majd a mindenki által elérhető objektum megfelelő elemét beállítja a bejövő üzenetnek megfelelően. Szabályos leálláskor értesíti a felhasználót egy "Szerveroldali fogadószál leáll!" üzenettel. [23]

4 **`Sender_class`**: Feladata az adatok továbbítása a szerver felé a socketen keresztül. Indulás esetén értesíti a felhasználót a "Küldő és bemenet kezelő szál elindul!" üzenettel. Szabályos leállás esetén értesíti a felhasználót a leállásról a "Küldő és bemenet kezelő szál leáll!" üzenettel. Folyamatosan figyeli, hogy van-e valamilyen küldendő üzenet a stat_var objektumban. Küldéskor a célkliens felhasználónevét beilleszti az üzenet elejére. [22]

5 **`Camera_handler_class:`** Feladata a kamerából érkező adatok feldolgozása. Indulásakor a „Kamera kezelő szál elindult!” üzenettel tájékoztatja a felhasználót.
   - A felhasználó által beállított kamerából érkező adatokat beolvassa [21] (OpenCV felhasználásával)
   - A felhasználói beállítástól függően megjeleníti a "látottakat" egy új ablakban.
   - QR kódot keres az adott képkockában.
   - Ha talál arról tájékoztatja a felhasználót.
   - Beírja az üzenetet az adatokat tároló objektum qr kódot tároló mezőjébe és azt azt a változót amely a bejövő adatról ad visszajelzést igazra állítja. Utóbbi lépésnek a kimenetkezelő szálnál lesz jelentősége.
   - Amennyiben nem talál adatot akkor a qr kódot tartalmazó adattag értékét üresre, és a qr kód találatot jelző adattag értékét hamisra állítja.
   - Ha a felhasználó kéri a kamera képének megjelenítését, akkor a képen található QR kódot bekeretezi.
   - A QR kód méretét és a bejövő kép méretét összevetve képes kiszámolni a kód távolságát. Ehhez a lépéshez szükség lesz még egy szorzótényezőre is, amely a kamera látószögétől függ. Ezt a tényezőt a legkönnyebben úgy lehet kiszámolni, hogy 100cm-re helyezzük a QR kódot a kamerától. A szoftver kiszámolja a QR kód és a képkocka arányát majd ezt az értéket megszorozzuk annyival, hogy 100-at kapjunk. Ezzel a módszerrel nagyjából 2,5m-ig pontos számításokat kapunk. Utána már nem elég a webkamera felbontása a stabil QR kód értelmezéshez.

   [24] [25] [26] [27]

   Sikeres leálláskor a „Kamera kezelő szál leáll!” üzenettel tájékoztatja a felhasználót.

6 **`Client_output_thread_class:`** Feladata a kliens eszköz kimeneteinek és az azokhoz szükséges beállításoknak a kezelése. Mivel a robot mozgásra képes ezért ez is egy kimenet forma. Meghívja a GPIO lábakat beállító függvényt. (Ez a függvény egy

önellenőrzést is végrehajt a program indulásakor.) Ha a bejövő üzenetek közül, vagy a kamera kezelő folyamat által talált QR kódban végrehajtandó üzenet van akkor az azoknak megfelelő függvényeket meghívja. Ilyenkor a kerékforgatás vagy a mozgás idejére indít egy új szálat, ami végrehajtja a kívánt utasítást. Majd azt a változót, ami az adott utasítást tartalmazza hamisra állítja. Ha a felhasználó szeretne a kerekek irányán manuálisan állítani azt is megteheti ilyenkor a maunális kerékállítást végző művelet hívódik meg. Itt a felhasználó a kerekek irányát finom hangolhatja, hogy a null érték pontosan előre nézzen.

7 **`GPIO_output_handler_thread_class:`** Feladata a GPIO lábak beállítása a megfelelő portokon, valamint az induláskor történő önellenőrző folyamat vezérlése. Ilyenkor a robot előre - hátra mozog, majd a kerekeket jobbra és balra forgatja majd vissza egyenesbe. Ezekről a lépésekről tájékoztatja is a felhasználót. A felhasználónak lehetősége van a fentebb említett manuális kerékbeállításhoz. Ilyenkor a 'j' és a 'b' billentyűk segítségével 0,01s időre tudja forgatni a kerekeket jobbra és balra. Ha a kerekek iránya megfelelő akkor a 'q' billentyű megnyomásával tud kilépni a menüből.

8 **`Move_thread_class:`** Feladata a GPIO lábak vezérlése az előre vagy hátra haladás érdekében. Figyeli az adatokat tároló objektum előre és hátra haladást tároló változóit. Megvalósítja a motor PWM elvű vezérlését. Az előre beállított kitöltési értéket tartja, a GPIO láb időzített be és kikapcsolásával. Miután végzett a vezérléssel leáll, ezzel is kímélve az erőforrásokat. Leállás előtt az általa vezérelt lábakat alaphelyzetbe, azaz kikapcsolt állapotba teszi. Ez egy egyszerű védelmi intézkedés, amivel el lehet kerülni a telep gyors merülését és a motor leégését.

9 **`Turn_thread_class:`** Feladata a GPIO lábak vezérlése a jobbra vagy balra fordulás érdekében. Figyeli az adatokat tároló objektum jobbra és balra fordulást tároló változóit. Megvalósítja a motor PWM elvű vezérlését. Az előre beállított kitöltési értéket tartja, a GPIO láb időzített be és kikapcsolásával. A fordulást több szakaszban oldja meg, de a vezérlés megoldható akár a QR kódba írt olyan utasítással is ami a fordulás mértékét befolyásolja, de nem ez az alapértelmezett. Az alapértékből a fordulás a szélső értékig történik. Onnan 2 ellenkező irányú fordítási utasítás kell, hogy a másik szélső értékre elérjen. Például, ha jobbszélső állásban van: balra, balra utasítást kell kapnia mert először az alapállapotba kerül majd onnan fordul a baloldali szélsőértékhez. Miután végzett a vezérlés leáll, ezzel is kímélve az erőforrásokat. Leállás előtt az általa vezérelt lábakat alaphelyzetbe, azaz kikapcsolt állapotba teszi. Ez egy egyszerű védelmi intézkedés, amivel el lehet kerülni a telep gyors merülését és a motor leégését.

3.8. Képek kezelése

AVI (Audio Video Interleave): Jelentése Audio Video Összefésülés. Egy fájlformátum, melyet a Microsoft 1992-ben mutatott be. Feladata a hang és a videó adatok egy csomagban való tárolása. Jelen esetben csak a kép továbbítódik.

Egy speciális esete a RIFF formátumnak (Resource Interchange File Format - erőforrás-cserélő fájlformátum) mely a fájl adatait tömbökbe "chunk"-okba darabolja.
A RIFF formátumon belül az AVI egyetlen tömböt alkot. Ez két kötelező és egy nem kötelező altömbre van bontva.:

1 Az első altömb a "hdrl" címkével van ellátva. Ez a fájl fejléce és a meta adatokat tartalmazza (adat az adatról). Információt találunk itt többek között a kép szélességéről és magasságáról, a képkockák számáról.
2 A második altömb a "movi" címkével van jelölve. Ez a kép és hang adatokat tartalmazza, ezek alkotják az AVI videót.
3 A nem kötelező altömb az "idxl" címke jelöli, feladat a helymeghatározás a fájlon belül.

A RIFF formátumon keresztül a "movi" adatai dekódolhatóak és kódolhatóak egy modul segítségével. Ez a modul a kodek. Feladata a nyers adat a "movi" tömbön belül használt adatra fordítása. Ezért az AVI a médiatartalmat gyakorlatilag bármilyen tömörítőeljárásnak megfelelően tartalmazhatja, feltéve, hogy a tömörítési eljárás megfelel valamely sémának, de tömörítetlenül is tartalmazhatja (például: Full Frames, Intel Real Time Video, QPEG, MPEG-4). A jelenlegi egyik legnépszerűbb megoldás az, ha a videót DivX-be a hangot Mp3-ba tömörítjük. [28]

JPEG (Joint Photographic Experts Group): Képek tárolására alkalmas fájlformátum. A JPEG formátumot 1992-ben fogadták el, mint különböző képtömörítési eljárásokat leíró normát. A kép tartalmát veszteségesen tömöríti. A szabvány tartalmaz egy veszteségmentes tömörítést is de ezt nem használják. A tömörítés ugyan információveszteséggel jár, de egy 10-100x-osan tömörített kép a visszaállítása után is élvezhető marad. Fényképek és rajzok tárolására alkalmas első sorban. A Grafikonok és egyéb hirtelen szín átmenetes ábrákhoz a PNG a megfelelő formátum.

A módszer különböző színmélységeket képes kezelni, használható szekvenciális és progresszív módon is:

- szekvenciális: soronként történik a tömörítés.
- progresszív: először durva formákat, majd a finomabb részleteket dolgozza fel.

JPEG-ben nem a képpontok tárolódnak le, hanem a képet transzformálják a frekvencia tartományban DCT-vel (diszkrét koszinusz transzformáció).

Továbbfejlesztett változata a JPEG2000, mely a DCT helyett Wavelet transzformációt használ. Ez a 8x8 pixeles elemi blokkok határán jelent nagy különbséget.

A szabvány ugyan a tömörítést leírja, de nem határozza meg a tárolási módokat. A képeket

többnyire a JFIF (JPEG File Interchange Format) formátumban tárolják. A kép feldolgozása többlépcsős:

1. Színtérszámítás RGB színtérből YCbCr színtérbe CCIR601 szerint
2. Cb és Cr színkomponensek szűrése és letapogatása.
3. 8x8-as blokkokra történő felosztás majd a blokkok diszkrét koszinusz transzformációja
4. Kvantálás
5. Átrendezés
6. Entrópiakódolás

Az 1,2,3,4-es lépéseknél képződik a veszteség. Ezt az emberi szem egészen a 1,5 - 2 bit/pixel adatmennyiségig nem látja, 0,3bit/pixel alatt a kép használhatatlan., mivel a tömörítés során olyan mintázatok kerülnek a képre, amelyek korábban nem voltak ott, blokkok képződnek színek jelennek meg és tűnnek el, szürkül a kép a JPEG2000 ezeket a hibákat orvosolta.

Ha a forrásfájl 24-bit-RGB fájl, akkor 12-15-szörös tömörítésnél még szabad szemmel nem látható a veszteség, 35-ig még jó minőségűnek nevezhető a kép. Ezek a számok függenek a kép tartalmától is. Ha sok az apró részlet a képen ezek elveszhetnek a tömörítési eljárás során. [29]

3.9. PC teszt kliens

A fejlesztés egy korai állapotában készült egy kliens PC-re, amely az Androidos telefonon futó klienst váltotta ki annak elkészüléséig. Ez a Java nyelven írt program volt az alapja a később Python nyelven elkészített Raspberry-n futó kliensnek.

A program segítségével könnyen és gyorsan lehet szimulálni a telefonos változat által kiadható utasításokat. Próbaüzemben lehetett vele vezérelni a robotot, valamint üzenetet küldeni neki és fogadni. Az Androidos kliens alapjai is ebből a kódból származnak, köszönhetően a Java platformfüggetlenségének.

Ez a változat tartalmaz:

- egy fogadó szálat
- egy küldő szálat
- egy grafikus felületet az egyszerű kezelhetőség érdekében

A program nagyon kevés szálon fut, így nem kellett az optimalizálással sok időt eltölteni. Egy fogadó és egy küldő szál fut használat közben. A program indítása után, meg kell adni a felhasználónevet, a cél felhasználót, és a szerver IP címét. A program Socketen kapcsolaton keresztül csatlakozik a szerverre. Sikeres csatlakozás esetén elküldi a felhasználónevét a szervernek, amellyel a továbbiakban azonosítani lehet majd.

Egy szövegdobozban tájékoztatja a felhasználót a futással kapcsolatos információkról:

- Sikeres csatlakozás a szerverre
- Sikertelen csatlakozás a szerverre

- Ha elveszti a kapcsolatot a szerverrel
- Hiányzó felhasználónév
- Hiányzó cél kliens felhasználónév
- Hiányzó IP cím.

A program többnyire a szerverrel egy eszközön fut, így a hálózati kapcsolódásból eredő problémák felderítésére nem alkalmas. A hálózati kapcsolat sebessége nagyban befolyásolja a szerver-kliens közötti kapcsolatot. Így ezzel csak a program megfelelő futása ellenőrizhető. Ha a telefon vagy a Raspberry WIFI kapcsolatával valamilyen hiba van akkor az üzenetek késhetnek, vagy el is veszhetnek. Ha a kliens és a szerver egy helyen fut ez nem fordulhat elő, legfeljebb a hibás port beállításból következhet sikertelen kapcsolódás. [34]

Egy nem megfelelő router választása esetén akár másodpercekben is mérhető az az idő ami az üzenet elküldése és a parancs végrehajtása között eltelik. Ez nem feltétlenül jelent problémát, mert a robot képes az önálló működésre, de a távirányítású üzemelést akár el is lehetetleníti.

A fogadó szál a szervertől fogadja az üzeneteket. Fogadás után megjeleníti azok tartalmát az erre a célra kijelölt szövegmezőben. Képes felismerni, ha az üzenet irányváltoztatáshoz kapcsolódó parancsot tartalmaz. Ezt jelzi is az információs szövegmezőben, az ettől eltérő üzeneteket megjeleníti az egyszerű üzeneteknek szánt mezőben. Az üzenet megjelenítését a felhasználónév megjelölésével kezdi, majd az üzenet tartalmát írja ki.

A küldő szál a felhasználó által beírt, majd a "küldés" gomb megnyomásával elküldött üzenet továbbításával foglalkozik. A szöveg elejére illeszti a célfelhasználó nevét egy "@" karakterrel elválasztva. Így a @ karakter használata nem megengedett az üzenet szövegében. Az elválasztó karakter megváltoztatható, de akkor a többi klienst is fel kell a változásra készíteni.

A kliens átalakítható egy chat klienssé pár egyszerű lépésben. Le kell cserélni a felhasználónevet az üzenettől elválasztó karaktert egy olyan karaktersorozatra amely nem valószínű, hogy megjelenik az üzenetküldés során, vagy két lépésben kell elküldeni. Először a célfelhasználó nevét majd az üzenetet. Ilyenkor a szervert is fel kell készíteni a megváltozott adattovábbítási módra. A szerver előre meghatározott időközönként elküldi az aktív felhasználók listáját, amit a kliens megjelenít.

A Grafikus felületen találhatóak gombok, melyekkel a robot haladási irányára lehet hatni. Ezeknek a megnyomásával ugyanazt a parancsot lehet elküldeni a szerveren keresztül a Raspberry-n futó kliensnek, mint amit az, a QR kódokból kiolvas. Így távirányíthatóvá válik az eszköz. Létezik a kliens PC-s párja is ami a Raspberry-n futó változatot szimulálja. Itt a gombok helyett sima szövegdobozok jelzik a haladással kapcsolatos információkat, minden másban megegyezik ezzel a változattal.

A felület az alap Swing elemekből épül fel:

- Görgethető szövegmezők
- Gombok
- Szövegdobozok

3.10. PC Java Szerver

Feladata a Raspberry-n és az Androidon futó kliensek összekötése. Ezt socket kapcsolattal teszi lehetővé. A fejlesztés Windows alatt történt, de a Linuxon történő futtatás is lehetséges. A Java szerver és egy kezdeti fázisban maradt PC kliens egyszerre készültek. A Java nyelv azért volt előnyös választás mert így a fejlesztés menete nagyon hasonló lehetett az Androidos eszközön, a Raspberry-n és a Windows vagy Linux operációs rendszert futtató PC-n.

A csatlakozott kliensektől érkező üzeneteket továbbítja az üzenetben elhelyezett címzettnek (ha van ilyen). Az üzeneteket nem tárolja, így, ha az egyik kliens offline csak az utolsó üzenetet kapja meg a csatlakozásakor. A projekt jellegéből adódóan nem is szükséges az üzeneteket tárolni, de a megvalósítás lehetővé teszi, hogy némi módosítás után rendelkezzen az üzenet tárolásának lehetőségével. Így, akár egy chatprogram szervere is lehet. Egy ilyen működést a kliensek is támogatnak a szükséges módosítások elvégzése után.

Mivel a szerver feladata csak az adatok továbbítása a címzettek felé ezért más célokra is felhasználható. Kisebb módosításokkal chat program szervere is lehet. Ilyen felhasználás esetén az áthaladó üzeneteket nem kell megjeleníteni. Egy frissen csatlakozott felhasználó megkapja az éppen aktív felhasználók listáját annak érdekében, hogy ki tudja választani kinek akar üzenni. Egy adatbázis csatlakoztatásával az ismert felhasználók tárolhatók. Az üzenetküldés lehet előzetes regisztrációhoz kötött, így csak az ismert felhasználók tudnak üzeneteket küldeni egymásnak. Az adatbázisban tárolhatók az üzenetek így, ha egy felhasználó offline akkor a csatlakozása után megkapja a korábban küldött üzeneteit.

Egy esetleges továbbfejlesztett változatban a szerveren áthaladó képeket ki lehetne elemezni. Mivel a Raspberry teljesítménye nem elegendő az adatok helyben történő feldolgozásához ezért az, csak továbbítaná a kamera képét a szervernek. A szerver pedig tetszőleges képfeldolgozási eljárásokat hajthatna végre a kapott adatokon például objektumok felismerése. Az objektumfelismerés alkalmas lehet egy bolt árukészletének nyilvántartására. Az OpenCV támogatja az arcfelismerést ha kellő számú minta áll rendelkezésre. Egy ilyen felhasználás során az eszköz lefényképezi az előtte álló személyt a szerver pedig azonosítja az adatbázis alapján. Ha nincs egyezés akkor értesíti az illetékeseket.

Mivel a szerver megkapja a robot által beolvasott információkat ezért nyomon is tudja követni az aktuális állapotát. Egy sematikus ábrán lehet ábrázolni az eszköz aktuális pozícióját, hasonlóan a buszokon található monitorhoz.

A Java nyelvben választása öt fő okra vezethető vissza:

1. A párhuzamos szálakkal történő programozást egyszerűen lehet benne megvalósítani
2. A platformfüggetlensége miatt az összes szükséges operációs rendszert támogatja
3. A nyelv objektumorientált. A részegységek egységbezárása miatt és a logikai elemek elkülönítése miatt volt fontos.
4. A Socketen keresztül történő kommunikáció megvalósítására könnyen használható eszközök állnak a rendelkezésre.

5 Egyszerűen készíthető grafikus vezérlőfelület (GUI) a korábban csak konzolos felületen karakteres megjelenítő helyett.

A párhuzamos szálkezeléssel segítségével lehet megvalósítani a kliensek elválasztását egymástól. Mivel két klienst kell egyszerre kezelni, ezért, ha a megvalósítás jó, akkor nem igényel sok erőforrást. Ilyenkor a kliensekhez kötődő folyamatok a szerveren egymástól függetlenül futnak, egy esetleges hiba nem állítja le az egész programot.

A szálak egymással egy mindenki számára elérhető és módosítható objektumon keresztül kommunikálnak. Ha egy szál megszerzi a neki szükséges információt akkor törli azt, annak érdekében, hogy legközelebb már ne olvassa ki. Ha megvan az üzenet akkor továbbítja a kliensnek, vagy ha a szervernek szóló tartalom van benne akkor végrehajtja azt.

Az első változatban a megvalósítás nem volt tökéletes, így a szervernek igen magas lett a hardverigénye. Ennek oka a szálak kezelésében volt keresendő, valamint a szálak futása során elvégzett tevékenységekben. A futtatás nagy erőforrásigénye miatt a szerver több váratlan hibát is produkált:

1 Lefagyott
2 Eldobta a kapcsolatot a kliensekkel
3 Nem küldte el az üzenetet
4 Többször is elküldte az üzenetet
5 Váratlanul kilépett
6 A grafikus felület fagyott le, de a működés továbbra is hibamentes volt

Ezeket a hibákat a szálkezelés rendberakása után nem produkálta. A megoldás a szálak futásának időzítése és a szálak futásának szüneteltetése volt. Ha egy kliens üzenetet küld akkor azt a program továbbítja vagy végrehajtja majd a száll futása szünetel egy darabig. Ha egy kliens lecsatlakozik, vagy nem válaszol az ellenőrző üzenetre akkor a szál leáll. Korábban előfordult, hogy a lecsatlakozott kliens után a hozzá rendelt folyamat még tovább futott és pazarolta az erőforrásokat. Egy klienshez egy fogadó és egy üzenetküldő szál társul. Így minden felvett új kliens kettővel növeli a futó folyamatok számát.

A kliensek csatlakozásának ellenőrzését a szerver egy watchdog timer szerű működéssel oldja meg. Egységes időközönként üzenetet küld a klienseknek, akik válaszolnak az üzenetre. Ha nincs válasz akkor megismétli a következő periódusban. Ha ekkor sem kap választ akkor az adott szálat leállítja. Ilyenkor a kliens újra csatlakozhat.

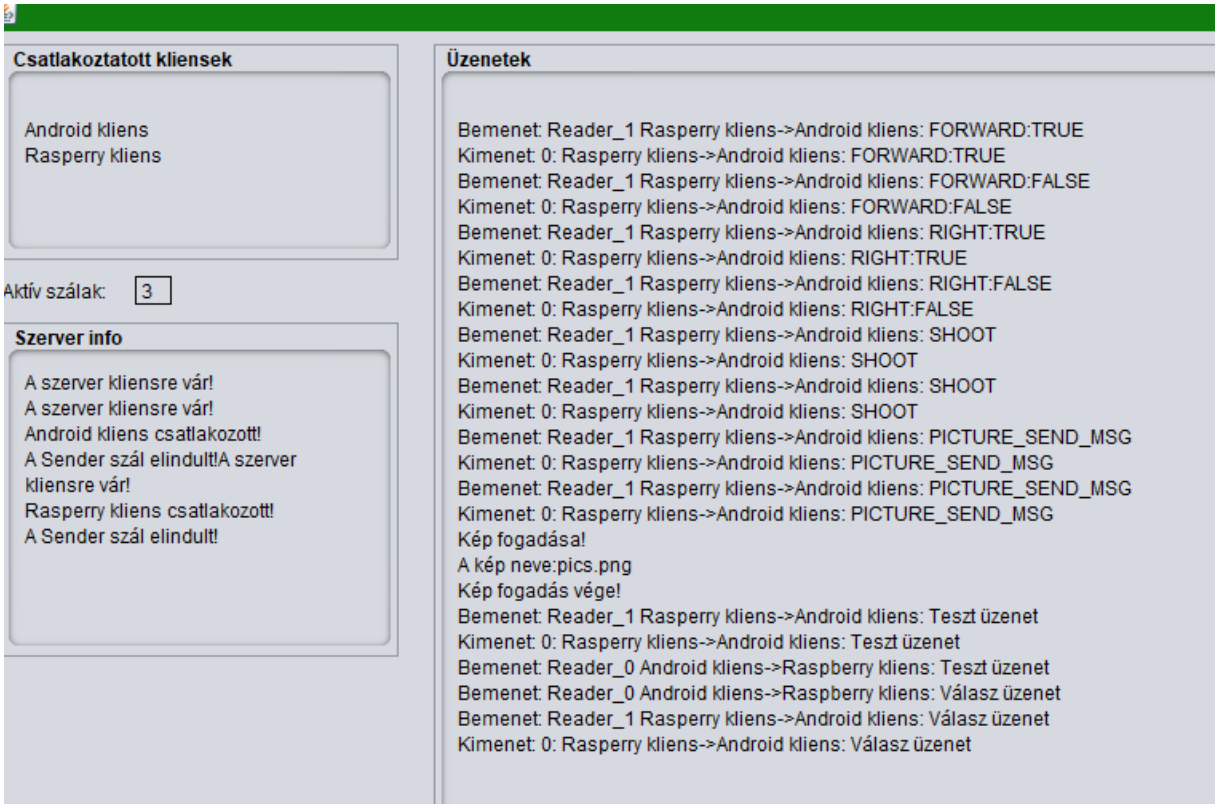
Java szerver felépítése: A szerver 2 fő részegységből áll. Az egyik a működést megvalósító egység (modell) amely végzi a kliensekkel történő kommunikációt, a szálkezelést, kliensek ellenőrzését stb. A másik a grafikus felület (GUI) amelyen keresztül tájékoztatja a felhasználót az aktuálisan futó folyamatokról, csatlakozott felhasználókról, kiírja a hibaüzeneteket, mutatja, hogy hány szál fut aktuálisan. Egy szövegdobozban kiírja a kliensek közötti üzeneteket. Ha egy chat program szervereként üzemelne ezt a privát üzenetek esetén nem tenné, vagy a kliensek valamilyen kódolt formában üzennének egymásnak és csak a hash kódokat jelenítené meg.

A kliensek csatlakozásuk után elküldik a saját felhasználónevüket, ezzel lehet majd őket azonosítani a továbbiakban. A később küldött üzeneteknek az első részben tartalmazniuk kell az adott üzenet címzettjét. Ha a kapcsolódás sikeres volt a szerver indít egy küldő szálat és egy fogadó szálat. Ha egy kliens üzenetet küld a fogadó szál kiolvassa belőle a címzettet és a megfelelő küldő oldali szál által elérhető változóban letárolja. Ha a küldő szál észleli, hogy a küldendő üzenetet tartalmazó string nem üres továbbítja az üzenetet. Továbbítás után törli a string tartalmát, ezzel elkerülve, hogy folyamatosan az utolsó kapott üzenetet továbbítsa. A kliens addig hatja végre az utoljára kapott utasítást amíg más nem kap vagy elér egy bizonyos végállapotot pl.: kerekek szélső értékig elfordultak.

Ha a szerver olyan üzenetet fogad, amelynek a címzettje nem létezik, akkor figyelmen kívül hagyja azt. Mivel 2 eszköz kommunikál ezért nem szükséges visszajelezni, hogy az adott felhasználó nem létezik, de egy chat szolgáltatást megvalósító szervernél ezt a lépést célszerű megtenni.

A szerver az indulást követően elvégzi saját magán a szükséges beállításokat. Nyit egy Server Socketet a 10000-res portra. Elindítja a csatlakozott felhasználókat figyelő szálat és a futó folyamatokat számoló szálat is. A grafikus felületen a szövegdobozok formázását html tag-ekkel lehet megvalósítani, így ezeket is be kell állítani.

Kliens csatlakozása után a hozzá rendelt fogadó és küldő szálat is elindítja. Jelzi a felhasználónak a csatlakozást, és ha megvan a kliens felhasználóneve akkor kiírja a megfelelő, csatlakozott felhasználókat listázó szövegdobozba.



6. ábra Szerver grafikus felülete [8]

Watchdog timer (felügyeletidőzítő): Beágyazott rendszerekben és más számítógépes vezérlésű eszközökben használják, ahol a felhasználó nem fér hozzá az eszközhöz, vagy nem tud időben reagálni a fellépő hibára. Egy ilyen rendszer nem függhet a felhasználótól, ha valamilyen probléma adódik önállóan kell arra megoldást találnia. A módszert széles körben használják az űrszondákban, hiszen fizikailag azok is hozzáférhetetlenek.

Operációs rendszerekben is használják, egy időintervallum áll bizonyos műveletek végrehajtására. Ha a művelet nem végez az intervallumon belül akkor a rendszer rögzíti a hibaüzeneteket majd leállítja a folyamatot.

Számos konfigurációban előfordulnak, gyakran a mikrovezérlők is tartalmazzák. A számítógépekben egy a processzorhoz közeli chipben vagy egy bővítőkártyán helyezkedhet el. Futhat a processzorral azonos és különböző órajelen is. [30]

Grafikus felület: A grafikus felület a Swing felületkezelővel készült. A Swing egy grafikus felület leíró eszköz, ami a Java nyelvvel kompatibilis. A JFC (Java Foundation Classes) részét képezi. Fejlesztése során a cél, egy AWT-nél (Abstract Windows Toolkit) részletesebb, kifinomultabb grafikus felületet lehessen vele létrehozni. Segítségével bármilyen grafikus felület elkészíthető.
Alapértelmezetten tartalmaz:

1. gombokat
2. jelölőnégyzetet
3. radio gombokat
4. csúszkákat
5. szövegdobozokat
6. szövegmezőket
7. görgetőpaneleket
8. füles paneleket
9. táblázatokat
10. listákat

Az AWT-vel ellentétben platformfüggetlen, mert a teljes felület leírható Java kóddal. Erősen moduláris architektúra, lehetővé teszi a modulok teljes egyénre szabását, majd ezekből egy egyedi megvalósítás létrehozását. A felhasználó a Java öröklési megoldásainak segítségével a modulok alapértelmezett opcióit szabadon felülírhatják ezzel elérve az egyedi megjelenést.

A JComponent osztály az őse az összes felület leíró elemnek. Az elemekhez kapcsolódó események aszinkron módon hívódnak meg, tulajdonságaik kötöttek dokumentált metódusként működnek. A felület akár futásidőben is gyorsan megváltoztatható.

A széleskőrű felhasználhatóság oka, hogy a Swing képes felülírni a gazdagépen futó operációs rendszer által használt grafikus felület vezérlőit a saját megjelenítése érdekében. A vezérlőfelületek megjelenítéséhez a Java 2D API-t használja fel.

A könyvtár nagyrészt függ az MVC modell (Modell View Controll - Modell Nézet Vezérlő) mintáitól. Ennek lényege, hogy a megtekintett adatok, a beviteli elemek és az ezek utasításait kezelő modelltől el vannak különítve. Az elemekhez társulnak modellek, amelyekhez a felhasználó írhatja meg a megvalósítást. A tipikus használat csak a fák és listák esetén igényelhet egyedi modellek létrehozását. A keretrendszer biztosítja az alapértelmezett implementációkat, melyek társítva vannak a Swing könyvtárban található gyermekosztályokkal. A listák, fák, táblák igényelhetnek egyedi modelleket, ha az adatstruktúrák alkalmazás-specifikusak.

A Swing a relatív elemelrendezést preferálja. Ez, a felületet alkotó vizuális elemek közötti helyzeti viszonyokat írja le. A másik elrendezési mód az abszolút elemelrendezési módszer. Itt az elemek helye és mérete pontosan rögzítve van. Utóbbi eljárás kevésbé kezeli rugalmasan az ablak méretének változásait. Nem megfelelő beállítások esetén a grafikus felület széteshet. [31]

- beérkező és kimenő üzenetek megjelenítése
- csatlakozott felhasználók felhasználónevei
- aktuálisan futó szálak

3.11. Android kliens

A program feladata a robot távirányítású vezérlése amennyiben az szükséges. Ha az eszköz letér a kijelölt útvonalról, vagy egy nem várt esemény miatt kitérőt kell tennie akkor ez megtehető a PC kliens vagy az Androidos kliens segítségével. Készíthető fénykép távirányírással és a szövegbeviteli mező segítségével lehet üzenni is az eszköznek, vagy utasításokat adni neki.

Az Androidra történő fejlesztés történhet Kotlin és Java nyelven is. Az utóbbi nyelvre esett a választás mert így a kód egyes részei egységesek lehettek a PC klienssel és a korai Raspberry-n futó klienssel is. Az egységes nyelvhasználatnak és az azonos kódrészleteknek köszönhetően a hibák megkeresése és javítási megoldásának kidolgozása is könnyebb. Egy esetleges probléma kijavítását egy helyen kell kitalálni és a kész megoldást át lehet ültetni más kódokba is.

A feladat elvégzésére a Kotlin nyelv is tökéletesen alkalmas, mert lefordítható a Java Virtuális Gépére és alkalmas arra, hogy a Java nyelven készített kóddal működjön együtt. A Java könyvtár egy részére épít például a Collections könyvtár. A Kotlin nyelv egyik előnye a Java-bal szemben, hogy sokkal tömörebben, kevesebb kód írásával lehet ugyanazt a programot megírni. A kevesebb kód kevesebb hibalehetőséget jelent. A nyelv 16 évvel fiatalabb, mint a Java ezért egy kezdő felhasználó jóval kevesebb segédanyagot, információt találhat az interneten. 2021-ben a Java a 2. legnépszerűbb nyelv volt 17,82%-os részaránnyal, míg a Kotlin a 13. 1,44%-kal a PYPL Index szerint átlagosan világszerte [32]. A jövőben a Kotlin valószínűleg nagyobb népszerűségre fog szert tenni a hosszútávú tendenciák alapján. Az Android studio egyformán támogatja a Java és a Kotlin nyelven való fejlesztést. A két nyelv támogatása 2017-ben jelent meg a szoftverben először. A Google célja, hogy a fejlesztők

elsősorban a Kotlin nyelvet használják az Android app fejlesztéshez, ugyanakkor a Java nyelv támogatása is hosszútávon megmarad. A Kotlin számtalan előnye mellett a választás a Java-ra esett a korábban szerzett tapasztalatok a platformfüggetlensége és az egységesen elvégezhető hibajavítás miatt.

A fejlesztés az Android Studio-val történt. A szoftver által alapértelmezetten biztosított grafikus felülettel készült fel a felhasználói felület. A felület leírása XML kóddal történik, amellyel könnyen lehet gombokat, szövegdobozokat, szövegmezőket létrehozni, akár csak a Swing segítségével. Az XML kóddal széleskörűen lehet a grafikus felület elemeit személyre szabni. A felület megalkotásához az Android Studio-ba integrált Layout Editor-t kell használni.

Az XML (Extensible Markup Language - kiterjeszthető jelölőnyelv) egy általános célú leíró nyelv. Adattípusok leírására képes. Elsődleges célja a strukturált szöveg leírása az interneten keresztül. Előnye, hogy az emberi szem számára is könnyen értelmezhető a leírás jellege. Képes a legtöbb adatstruktúra leírására. Platformfüggetlen így könnyen szállítható.

A kliens egy Android rendszerű telefonon kapott helyet. Az Android széles körben elterjedt jelenleg a legnépszerűbb operációs rendszer megelőzve a Windows-t és az IOS-t 39%-os részaránnyal 2021-ben. A nagy részesedésnek köszönhetően a megírt kliens rengeteg eszközön képes futni. A tesztelés egy gyengébb és egy közepes teljesítményű telefonnal történt. Mind a két eszközön megfelelően futott a program. Így, ha a rendszert egy nagyobb telephelyen kellene bevetni nem igényelne drága infrastruktúrát. Egy közép-alsó kategóriás eszközön már használható minőségben fut.

Minimális rendszerkövetelmények:

- Android 6.0 Marshmallow
- 1 GB szabad tárhely
- 2 GB RAM
- Minimum CPU: HiSilicon Kirin 650 vagy Qualcomm Snapdragon 430
- Ajánlott CPU: Qualcomm Snapdragon 720G
- Wifi kapcsolat

Android operációs rendszer: A fejlesztés 2005-ben indult a cél egy mobil operációs rendszer készítése volt. Elsősorban érintőképernyős eszközökre lett tervezve (telefonok, táblagépek). A fejlesztés az Andoid Inc kezdte meg de a Google felvásárolta a céget így a további munka már a Google felügyelete alatt folytatódott. Az első Android operációs rendszerrel ellátott eszköz 2008-ban jelent meg.

A fejlesztők a rendszerre Java és Kotlin nyelven írhatnak menedzselt kódot, amely az eszközt vezérli a Google által biztosított programkönyvtáron keresztül. A platform megalkotásának célja, hogy a hordozható eszközök egy egységes nyílt forráskódú operációs rendszerrel legyenek ellátva. Az alapokat egy Linux rendszer adja a megfelelő átalakításokkal, hogy képes legyen az elvárt funkciók ellátására:

- Érintőképernyő kezelése
- Egynél több kamera egyidejű kezelése

- WIFI
- Bluetooth
- HSDPA (High-Speed Downlink Packet Access) - Nagy sebességű Csomagletöltési Hozzáférés, 4G

A Linux kernel tartalmazza a hardverhez szükséges drivereket, melyeket az eszközök gyártói készítenek el. A Linux rendszerekben alapértelmezetten megtalálható könyvtárak a kernel által biztosított szolgáltatásokat használják. Ezekre épül a Dalvik virtuális gép, amely Sun virtuális gépével nem kompatibilis, más az utasításkészlete és más bináris programot futtat. A Java programok egy. dex formátumú kiterjesztést használnak szemben a Java által használt több .class-al. Ezeknek a. dex (Dalvik Executable) fájloknak többnyire kisebb a méretük, mivel a több Java fájlban is megtalálható konstansokat csak egyszer fordítja le a Dalvik fordító. A Sun által fejlesztett virtuális gép és a Davlik rendszer között csak a Java a közös pont minden más elemükben eltérnek.

Az Android verziói a 2.2-es változattól egészen a 9-es verzióig az ABC betűi szerint sorban haladva kapják az elnevezésüket. Froyo, Gingerbead, Ice Cream Sandwich, Jelly Bean, KitKat, Lollipop, Marshmallow, Nougat, Oreo, Pie. A 9-es változattól felfelé már nem alkalmazzák az édességneveket.

Ha nem áll rendelkezésre wifi router akkor az eszköz távirányítható a telefon hordozható wifi hotspot szolgáltatásával. Ilyenkor a szerver programot futtató PC-t és a robotot csatlakoztatni kell a telefonhoz wifin keresztül. A megfelelő beállítások elvégzése után használható a rendszer.

Android kliens is több szálon üzemel. A mai okostelefonoknak már bőven van olyan teljesítménye, hogy egy ilyen programot futtatni tudjon. Indítás után meg kell adni a szerver IP címét a felhasználónevet és a cél kliens felhasználóját. Ezt követően történik a csatlakozás. A program Socketen keresztül csatlakozik a szerverhez. Elküldi a felhasználónevét amivel a továbbiakban azonosítható lesz. Ha a csatlakozás sikeres, a telefonnal lehet irányítani a robotot. Beállítástól függően a telefon felhasználója akár tájékoztatást is kaphat a robot aktuális állapotáról. A telefonról lehet a PC-s klienshez hasonlóan üzenetet küldeni az eszköznek, amiben egyéni utasítások szerepelhetnek.

Egy esetleges továbbfejlesztett változatban lehetne jelezni, hogy hol jár a robot vagy a robotba való GPS beépítésével a koordinátákat is lehetne továbbítani a telefonra. Ha a szerver rendelkezik a képfeldolgozáshoz szükséges képességekkel akkor a telefonra érkezhet a felhasználónak tájékoztatás az azonosított személyekről.

A felhasználói élmény nagyban függ a telefon teljesítményétől a router teljesítményétől, a robot távolságától a wifi jelforrásához viszonyítva. Célszerű egy megfelelő lefedettségű területen használni a rendszert. [33]

4. Beüzemelés és hibaelhárítás

4.1. A beüzemelés lépései

Előkészítés: A felhasználónak meg kell győződnie, hogy a felhasználási területen megfelelő Wifi lefedettség van, valamint a robot vezérlését végző táblák megfelelő helyre vannak e kihelyezve. A sima csúszós felületektől mentes talaj előnyt jelent például: beton, járólap. Le kell ellenőrizni, hogy az akkumulátor vagy az ennek helyettesítésére szolgáló 4db ceruzaelem rendelkezik-e a megfelelő töltöttségi szinttel.

A kamerának:

- A megfelelő irányba kell néznie
- Nem lehet szennyeződés a lencsén
- A kamera kábele nem lehet megtörve, megszakadva
- A kábelt csatlakoztatni kell a Raspberry-hez és a kamerához is.

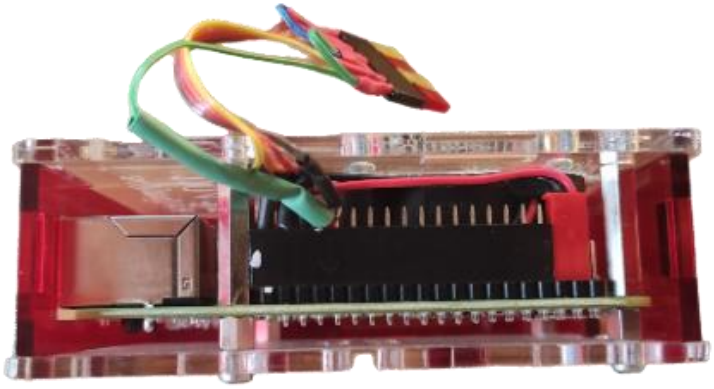
Ha a kamera nem megfelelő irányba néz, manuálisan be kell állítani. A kamera felfogatása olyan, hogy engedi a forgatást. Erre azért van szükség mert a kamera a járműtest fölött helyezkedik el és szállírás közben megsérülhet. Szállítás során a kamera visszahajtható a hátsó részbe.

7. ábra Robot, hátrahajtott kamerával [8]

Autó:

- A jármű alján található kapcsolónak kikapcsolt állapotban kell lennie az ellenőrzés során.
- Az eszköz teleptárolójába el kell helyezni az akkumulátorokat vagy az elemeket.
- Az L298-as IC-n a jumpereknek a megfelelő helyen kell lenniük.
- A Raspberry-t a ház belsejében kell elhelyezni, az erre kialakított üregben.
- A forgatható kerekeknek előre kell nézniük.
- Ha a Raspberry a helyén van és a kamera, a táp és a szalagkábel is csatlakoztatva van akkor a csavarokkal rögzíteni kell a ház tetejét. A rögzített ház megakadályozza a Raspberry pattogását elcsúszását. Ha a Raspberry elmozdul a kábelek kicsúszhatnak a portjukból.
- A kábeleknek mindig a házon belül kell lenniük. Ha egy kábel kilóg az beleakadhat a tereptárgyakba, vagy feltekeredhet a kerékre. Ez akár súlyosan is károsíthatja a Raspberry-t, a kamerát vagy a vezérlő IC-t.

Ha a kerekek nem előre néznek, azt egy későbbi lépésben korrigálni kell a "wheel_set" paranccsal. A robot a bekapcsoláskori kerékállást tekinti az alapértelmezettnek.



8. ábra Raspberry GPIO lábai és a pozícionáló fehér pötty [8]

Raspberry:

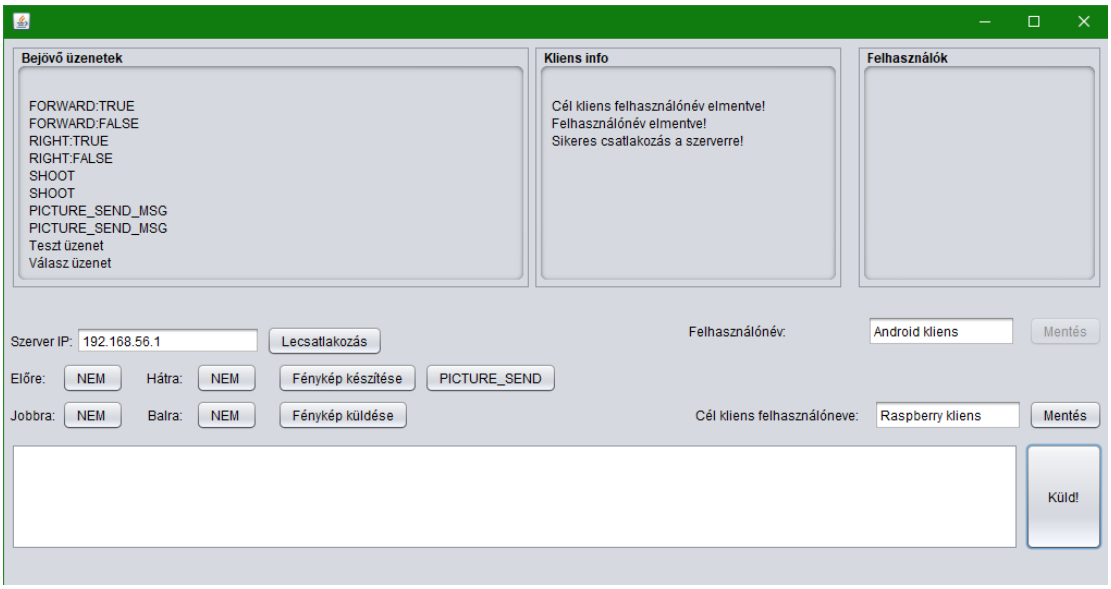
- A kék kapcsolónak kikapcsolt állapotban kell lennie. Ez azért szükséges mert az ellenőrzés során a Raspberry nem kaphat áramot.
- A szalagkábelnek megfelelően kell csatlakoznia a GPIO lábakhoz. A fehér pöttyel jelölt lábnak a fehér pöttyel jelölt lábhoz kell csatlakoznia.

- A hűtést biztosító ventilátornak csatlakoznia a kell a GPIO lábakhoz. A robot háza meglehetősen zárt, ezért szükség van erre a ventilátorra, hogy ne melegedjen túl a processzor.
- A kamerának a megfelelő portra kell csatlakoznia.
- A kábelek nem lehetnek megtörve vagy megszakadva és nem feszülhetnek
- Minden kábelnek a csatlakozásába könnyen kell illeszkednie, nem szorulhat, feszülhet. Ez az adott port károsodásával járhat.

Telefon:

- Minimum Android 7.0 Nougat Androiddal kell rendelkeznie.
- Képesnek kell lennie kapcsolódni a közös wifi hálózatra.
- Fel kell telepíteni rá a kliens programot. Ez a legkönnyebben az Android Studio segítségével tehető meg.
- Ha a robotot távirányítani kell akkor a wifi lefedettségén belül kell maradni.

Ezek mellett az eszközök mellett szükség van még egy olyan router-re amely képes internettel ellátni az összes eszközt. (vezeték nélküli vagy vezetékes kapcsolattal)



9. ábra PC-n futó Java kliens grafikus felülete [8]

Szervergép:

- A számítógépnek tudnia kell futtatni a szervert.
- Windows: telepíteni kell rá a Java Virtual Machine-t és a Java Standard Edition-t.
- Linux: csak a Java Standard Edition-t kell telepíteni.
- Rendelkeznie kell hálózati kapcsolattal. Azonos hálózati elosztóra kell kapcsolódnia, mint amire a Raspberry és a telefon fog. Mivel a szerver nem fog mozogni, ezért a kapcsolat lehet vezeték nélküli is.

- A szervergép elvárt teljesítménye a csatlakoztatott kliensek számától függ. Két kliens esetén 6 futó szállal kell számolni.

Tesztelési céllal a szerveren futhat a PC kliens is. Ennek használata kevésbé körülményes, mint a telefonos változaté. Másik előnye, hogy nem függ a hálózattól a csatlakozása.

Ha minden előkészítő lépés kész van kezdődhet a rendszer beüzemelése. Első lépés a szerver elindítása. A számítógépnek csatlakoznia kell a hálózatra, majd el kell indítani a szervet. Szükség lesz a szerver IP címére, ezt Windows operációs rendszer esetén a legkönnyebben a parancssorba begépelt "ipconfig" utasítással kaphatjuk meg. Linux esetén az IP-t a parancssorba beírt "hosname -I" utasítással kaphatjuk meg. Előbbinél az "IPv4 Address" kezdetű sornál utóbbinál az első számsornál találjuk a szükséges címet. Az IP cím a kliensek számára fontos. Így tudják azonosítani a szervet. A port is egy fontos paraméter, de az fix, a 10000-res számra van beállítva. Ezt a kliensek és a szerver is alapértelmezetten tudja.

A következő lépés az autó üzembe helyezése. Első lépésként a Raspberry-t kell áram alá helyezni. A rendszer automatikusan csatlakozik a wifire, ha ismeri a kapcsolódáshoz szükséges adatokat. Ha ez nem történik meg akkor van lehetőség egy billentyűzet és egy monitor csatlakoztatására és el lehet végezni a folyamatot manuálisan (tesztek során erre nem volt szükség mindig tudott csatlakozni). A Raspberry-nek kétféle módon lehet utasításokat adni. Lehet monitort, egeret és billentyűzetet használni, mert minden ehhez szükséges porttal rendelkezik, vagy lehet rá csatlakozni Putty segítségével. Ilyenkor tudni kell a Raspberry IP-címét, de a teszthez használt router mindig ugyanazokat a címeket osztotta ki az eszközöknek így ez adott volt. Az IP segítségével csatlakozni kell a Raspberry OS-hez meg kell adni a felhasználónevet és a jelszót. I Innentől parancssorral lehet vezérelni az eszközt, így nem szükséges a folytonos fizikai kapcsolat létesítése a beállítások módosításához.

Az autó hasán van egy három állású kapcsoló. A kapcsolót a jobbszélső állásba kell kapcsolni, ezzel áram alá kerül a vezérlő IC (egy piros visszajelző LED világít rajta). Innentől a program képes lesz vezérelni a GPIO lábakon keresztül az eszközt. A kábeleknek a házon belül kell lenniük. Ezt le kell ellenőrizni még egyszer. El kell a parancssor segítségével indítani a programot. Miután a felhasználó válaszol a parancssorban a kliens által feltett konfigurációs kérdésekre az eszköz csatlakozik a szerverre, meg kell adni a felhasználónevet, a cél kliens felhasználónevét és a szerver IP-címét. A sikeres csatlakozást a szerver és a kliens is jelzi. Ha az autó olyan területen halad, ahol nincs wifi kapcsolat akkor csak a QR kódokat tudja követni. A QR kód követése viszont lehetővé teszi, hogy nagyobb területet járjon körbe, mint amit a lefedettsége lehetővé tenne.

Kapcsolódás után a kliens elvégez egy gyors önellenőrzést. Előre-hátra mozog egy keveset majd elforgatja a kerekeket jobbra-balra. Leellenőrzi, hogy van e kamera csatlakoztatva. A felhasználó itt tudja a kerekeket egyenesbe állítani, ha szükséges a "wheel_set" paranccsal. Ez elvégezhető a Putty-n keresztül is.

Ha szükséges lehet csatlakozni a PC klienssel is. Ebben az esetben is a szerver IP-t a felhasználónevet és a másik kliens felhasználóját kell megadni. Ha a PC kliens a szerverrel egy gépen fut akkor a 192.168.56.1-es IP-címmel is lehet csatlakozni.

Az utolsó lépés a telefon szerverre csatlakoztatása. El kell indítani az appot, majd meg kell adni az IP címet. A többi paramétert a program alapértelmezetten ismeri. A sikeres és a sikertelen csatlakozásról is tájékoztatja a felhasznált. Az alapértelmezett felhasználó név az "Android" az alapértelmezett cél pedig a "Raspberry". Innentől a telefonnal lehet az autó mozgását befolyásolni, ha letérne az útról akkor lehet korrigálni. Az autó a QR kódok által kijelölt útvonalat követi alapértelmezetten, de ha szükséges akkor lehet vele kitérőket is tenni a telefon vezérlésével.

4.2. Felhasználó által elhárítható hibák

Egy esetleges probléma megoldása sokszor nem igényel semmilyen informatikai alapismeretet vagy műszaki tudást:

- Ha a kamera nem megfelelő irányba néz akkor a felhasználónak csatlakoztatnia kell egy monitort és egy billentyűzetet a Raspberry-hez. Elindítja a programot, de úgy, hogy a kamera kép jelenjen meg. Ezután a kamerát kell úgy mozgatni, hogy a kamera előtt található QR kód 1 méteres távolságban pont a kép közepén helyezkedjen el.
- Ha a Raspberry nem csatlakozik a wifi-re akkor egy monitor és billentyűzet segítségével manuálisan kell megadni a csatlakozáshoz szükséges adatokat.
- Ha a Raspberry nem kapcsol be akkor a tápellátást biztosító kábelt kell ellenőrizni vagy kicserélni.
- Ha az autó sem és a Raspberry sem kapcsol be akkor az akkumulátor vagy az elem merült le, ilyenkor telepcserét kell eszközölni.
- A kerekek beállítása a "wheel_set" parancs megadása utána "j" vagy "b" billentyűk megnyomásával történik.

5. Összegzés

5.1. Felmerült problémák és megoldásaik

A fejlesztés közben felmerül első probléma a socket kapcsolat portjaival volt. Kis kutatás után kiderült, hogy az eredetileg használt portot (1000) egy másik alkalmazás használja így a szerver a 10000-res porton fut. Ezzel a beállítással a portok beállításai megfelelőnek bizonyultak. Egy VPN felhasználásával LAN függetlenné tehető a rendszer.

A kliensek azonosítása, a kliens "megszólítása" után történik. Miután létrejött a kapcsolat a kliens elküldi a felhasználónevét. A szerver indít egy új szálat, ami a "listening" állapotba kerül. A kliens minden üzenetbe belekódolja a cél kliens felhasználónevét, ezek

alapján tudja a szerver ki a cél. Majd törli a tárolt üzenetet, ha ez elmarad akkor addig küldi ki az utolsó kapott üzenetet amíg mást nem kap.

A kommunikáció során előfordult olyan eset, hogy lecsatlakozás után a szerver oldalon a socket és az őt életben tartó szál nem állt le, ezzel értékes erőforrásokat pazarolva. A megoldás az volt, hogy időközönként a szerver küld egy ellenőrző üzenetet amire a kliens válaszol. Ha ezt nem kapja meg akkor leállítja az adott szálat. Újra csatlakozás után az adatok hiba nélkül továbbítódhatnak. A mérések alapján 10 klienst lehet egy átlagos teljesítményű számítógépen egyszerre használni. A hiba javítása után 10 klienssel még nem volt probléma.

A Raspberry-n a Java nyelven nem működött az OpenCV a probléma oka a Raspbian operációs rendszer c++ fordítója volt. Nem felelt meg a verziószám az előírt minimálisnak. 11-es vagy magasabb verziójú fordító szükséges hozzá. A megoldás a Python nyelv használata volt Java helyett, amivel a telepített OpenCV már probléma mentesen futott.

A Python változat nagyjából azonos módon működik, mint a Java. Az osztályokat és a függvényeket át lehetett ültetni. A funkcióik azonosak, a változó és az osztálynevek hasonlóak. Először a socket részek készültek el majd a többszálas futtatás végül a kamera integrálása a rendszerbe végül a QR kódfelismerés.

A Raspberry-re a Python legutolsó verziója (4.5.4) ismeretlen okból nem települ így a 3.4.16-os verzióval készült el a szoftver. Ez a működésre nincs hatással. A Python csomagtelepítője az utolsó lépésnél megakad és hosszas idő után hibaüzenettel leáll a legfrissebb verzió esetén.

A rendszer két kamerát tartalmaz. Az eredeti tervek szerint az egyik a földet nézte volna és ott keres QR kódot. Ha fölé ér akkor végrehajtja az utasítást. A probléma itt az volt, hogy az autónak minden fordulás után pont a kód fölé kell érnie és a kamerának a fókusztávolsága nagyobb volt, mint az elérhető maximális magasság. A következő lehetőség az lett volna, hogy a kamera az autó tetején van és előre néz. A földre elhelyezett kódot kellett volna felismerne. A földön lévő tábla és a 45 fokban elforgatott kamera együttesen okozta, hogy a bejövőkép vertikálisan torzul. Az így módosult objektum nem olvasható be. A megoldás a táblákra és falakra elhelyezett papírlapok és rajtuk a nyomtatott ábrák. A másik kamera jobb minőségű kép rögzítésére alkalmas. Ezzel az eszközzel lehet fényképeket készíteni. Azért nem ez az alapértelmezett kamera, mert a nagyobb felbontás miatt nagyon megnöveli az erőforrásigényt, ami az ARM alapú Raspberry-nél erősen korlátos.

A Python egy interpreteres nyelv, ellentétben a fordított nyelvekkel, mint például a C. A fordítóval rendelkező nyelveket futtatás előtt gép kóddá kell alakítani és csak ezt követően lehet átadni a processzornak. Az interpreteres nyelvek futási időben fordulnak át gépkóddá. Ez is hozzájárul a nagyobb rendszerigényhez.

A megfelelő mozgási sebesség eléréséhez impulzusszélesség-modulációt (PWM) kell alkalmazni, mert a teljes sebességgel való haladás negatívan hat az elérhető üzemidőre és a kamera képét is nehezebb értelmezni. A motorok teljesítményszintjének állításával könnyebben megvalósítható a fokozatos kanyarodás is. Vigyázni kell, mert minimum 15%-os kitöltési tényező kell az előrehaladáshoz és minimum 30% a kerekek forgatásához. Ennél kisebb értékek esetén a motorok nem képesek ellátni a feladatukat.

Az ötletben rejlő lehetőségek a rendelkezésre álló idő és a hardveres lehetőségek miatt nincsenek maximálisan kihasználva.A rendszer működése több helyen is nehézkes, de egy megfelelő szerverrel a problémák nagy része orvosolható. A rendszer megfelelő működéséhez egy erősebb akkumulátorral és jobb kamerával is szükség lenne, de a jobb nagyobb felbontású kamera miatt nehezebb lenne a képek feldolgozása, lassulna a rendszer. Az ideális eszköz megtalálása további kísérleteket igényel. A táviárnyítós autó kopottásga miatt nem használható a teljes rendelkezésre álló teljesítmény.

A hardveres elemek állapota miatt belátható, hogy fejlesztés közben mindig újra kell gondolni a lehetőségeket, hiszen a szoftvernek alkalmazkodnia kell a hardverhez. A programok írása közben mindig az ideális körülményekkel számoltam, de ezek a feltételek sosem voltak adottak, így minden főbb komponens megírása után tesztelni és újratervezni kellett.

6. Irodalomjegyzék

[1] Wikipédia. (2022. február 8). Forrás: Wikipédia: https://hu.wikipedia.org/wiki/Raspberry_Picvb, látogatva: 2022. Március 29.

[2] Raspberry Pi. (2019. június 24). Forrás: Raspberry Pi: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications, látogatva: 2022. Március 22.

[3] Wikipédia. (2021, Szeptember 29). Retrieved from Wikipéda: https://en.wikipedia.org/wiki/General-purpose_input/output, látogatva: 2022. Március 29.

[4] Wikipédia. (2016. december 18). Forrás: Impulzusszélesség-moduláció: https://hu.wikipedia.org/wiki/Impulzussz%C3%A9less%C3%A9g-modul%C3%A1ci%C3%B3, látogatva: 2022. Március 22.

[5] Wikipédia. (2021. június 20). Forrás: Wikipédia: https://hu.wikipedia.org/wiki/Egylapk%C3%A1s_rendszer, látogatva: 2022. Március 29.

[6] Spark Fun. (2000). Forrás: L298: https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf?fbclid=IwAR0BqeOiONFC_-_pYsH-4tw7t9oesdnQVA7x4rLoXpB7_9T3rD5FRkBtuCQ, látogatva: 2022. Március 29.

[7] Metal scrap records . (2021). Retrieved from “Guinness Mini DV” Y2000 Photo Video Camcorder /cam/: https://shop.metalscraprecords.com/guinness-mini-dv-y2000-photo-video-camcorder-cam-677, látogatva: 2022. Április 1.

[8] Makár, P. Á. (2022). Saját készítésű fénykép. Debrecen: Makár Péter Ákos.

[9] Wikipédia. (2019. május 16). Forrás: Egyenáramú gép: https://hu.wikipedia.org/wiki/Egyen%C3%A1ram%C3%BA_g%C3%A9p, látogatva: 2022. Március 22.

[10] kinmoremotor.com. (dátum nélk.). kinmoremotor.com. Forrás: FF/FK-180H: https://kinmoremotor.com/products/ff-fk-180h/, látogatva: 2022. Április 1.

[11] Wikipédia. (2021. Augusztus 11). Forrás: Wikipédia: https://hu.wikipedia.org/wiki/Java_(programoz%C3%A1si_nyelv) , látogatva: 2022. Március 22.

[12] Wikipédia. (2022. február 14.). Forrás: Wikipédia: https://hu.wikipedia.org/wiki/Python_(programoz%C3%A1si_nyelv) , látogatva: 2022. Április 1.

[13] Wikipédia. (2021. március 7). Forrás: Wikipédia: https://hu.wikipedia.org/wiki/OpenCV , látogatva: 2022. Március 20.

[14] Wikipédia. (2020. Május 5). Forrás: Socket: https://hu.wikipedia.org/wiki/Socket , látogatva: 2022. Március 20.

[15] Wikipédia. (2021. Augusztus 10). Wikipéda, QR-kód. Forrás: QR-kód: https://hu.wikipedia.org/wiki/QR-k%C3%B3d, látogatva: 2022. Március 22.
[16] Pi4J. (2012). Retrieved from The Pi4J Project: https://pi4j.com/1.2/install.html , látogatva: 2022. Április 1.
[17] Pi4J. (2012). Forrás: The Pi4J Project: https://pi4j.com/1.2/example/control.html , látogatva: 2022. Március 20.
[18] Tutorials Point. (2021). Forrás: Python - Multithreaded Programming: https://www.tutorialspoint.com/python/python_multithreading.htm, látogatva: 2022. Február 20.
[19] GeegksForGeeks. (2021, December 15). Retrieved from Multithreading in Python | Set 1: https://www.geeksforgeeks.org/multithreading-python-set-1, látogatva: 2022. Február 20.
[20] Python.org. (2021). Forrás: threading — Thread-based parallelism: https://docs.python.org/3/library/threading.html#module-threading, látogatva: 2022. Február 20.
[21] GeeksforGeeks. (2021, Március 28). Retrieved from OpenCV Python Tutorial: https://www.geeksforgeeks.org/opencv-python-tutorial/, látogatva: 2022. Március 17.
[22] GeegksforGeeks. (2021, November 10). Retrieved from Socket Programming in Python: https://www.geeksforgeeks.org/socket-programming-python/, látogatva: 2022. Február 20.
[23] Real Pyhton. (2021). Forrás: Socket Programming in Python: https://realpython.com/python-sockets/, látogatva: 2022. Február 20.
[24] Towards Data Science. (2020, Október 12). Retrieved from Building a Barcode/QR code Reader using Python: https://towardsdatascience.com/building-a-barcode-qr-code-reader-using-python-360e22dfb6e5, látogatva: 2022. Március 17.
[25] GeeksforGeeks. (2021, Októbert 20). Retrieved from Reading and Generating QR codes in Python using QRtools: https://www.geeksforgeeks.org/reading-generating-qr-codes-python-using-qrtools/, látogatva: 2022. Március 17.
[26] TheCleverProgrammer. (2020. Október 23). Forrás: Barcode and QR code Reader with Python: https://thecleverprogrammer.com/2020/10/23/barcode-and-qr-code-reader-with-python/, látogatva: 2022. Március 17.
[27] Stack overflow. (2010. December). Forrás: Capture single picture with opencv: https://stackoverflow.com/questions/4179220/capture-single-picture-with-opencv, látogatva: 2022. Március 21.
[28] Wikipédia. (2019, november 12). Retrieved from Audio Video Interleave: Audio Video Interleave, látogatva: 2022. Március 21.
[29] Wikipédia. (2021. Október 18). Forrás: JPEG: https://hu.wikipedia.org/wiki/JPEG, látogatva: 2022. Március 25.
[30] Wikipédia. (2021. Március 30). Forrás: Watchdog: https://hu.wikipedia.org/wiki/Watchdog_timer, látogatva: 2022. Március 25.
[31] Wikipédia. (2022, Február 9). Retrieved from Swing(Java): https://en.wikipedia.org/wiki/Swing_(Java) , látogatva: 2022. Március 21.

[32] Stat counter. (2021. December). Forrás: Operating System Market Share Worldwide: https://gs.statcounter.com/os-market-share/all/worldwide/2021, látogatva: 2022. Január 11.
[33] Wikipédia. (2021. december 21). Forrás: Android (operációs rendszer): https://hu.wikipedia.org/wiki/Android_(oper%C3%A1ci%C3%B3s_rendszer) , látogatva: 2022. Január 11.
[34] GeeksforGeeks. (2021. November 8). Forrás: GeeksforGeeks: https://www.geeksforgeeks.org/socket-programming-in-java/, látogatva: 2022. Január 11.