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.

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

1. **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.
2. **RIGHT: FALSE** : Ez egy elkészített, de sosem használt funkció. Abbahagyja a kerekek balra forgatását.
3. **PICTURE\_SEND:** Fejlesztés alatt. Készít egy képet az erre a célra felszerelt kamerával és elküldi a kliensnek.

A "robot" (#Nevezhetem robotnak?#) 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 módosíthatóak pl.: a készített képeket a szerveren is le lehet tárolni. A beérkező képet továbbítja a célként kiválasztott kliensre.

**Programozási nyelvek**:

Java**:**

A Java egy objektumorientált programozási nyelv, amelyet a Sun Microsystems a '90-es években fejlesztett egészen 2009-ig amikor az Oracle tulajdonába került a cég. 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. 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 objektumorientáltság 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, Raspbian )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. (#Wikipédia használva#)

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.

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 prototípuséval, 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 Raspbian 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. (#Wikipédia használva#)

OpenCV**:**

Az OpenCV egy C++ nyelven írt ingyenes gép látásért felelős függvénykönyvtár. A C++ alapok ellenére elérhető Python, Java és Matlab nyelven is. Ezen interfészek API-ja az online dokumentációkban érhető el.

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 kamerát tud 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. (#Wikipédia használva#)

**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.

Socket elemei:

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

Socketek típusai:

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.
3. Datagram socket: kapcsolatmentes adattovábbításra valóak, az User Datagram Protocol-t (UDP) alkalmazzák. Az adatcsomagok nem sorrendhelyesen továbbítódnak.

Szerver:

Egy szerveroldali socket kommunikációt valósít meg. Fogadja a kliensek kéréseit és továbbítja a megfelelő címre / portra.

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

**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. 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 ( pl.: max bal + jobb = egyenes ) és még egy utasítás szükséges a másik szélsőérték.

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 beolvasás nagyjából 230 cm ezt a kamera sajátosságai befolyásolják, de szoftveresen is csökkenthető az érték. 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 2 órás üzemidő is elérhető (#sosem mértem csak saccolom!#). Használható akumulátor is, mert az alapot képező távirányítós autó elemmel is és aksival is működtethető.

A korai tesztekben a Raspberry tápját egy hordozható 5V 2A 10 000mAh teljesítményű akumulátor 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 vertikálisan a rázkódás pedig horizontálisan 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.

**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 portbeállítást érintő hibák megszűntek. 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. Újracsatlakozás után az adatok probléma mentesen 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.