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 állítottam be, 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 olatással 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 alkalmaznom, 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 foragatását.

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 foragatását.
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 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, egy megírt kód hardvertől és operációs rendszertől függetlenül fut a Java Virtual Machine-nek (JVM) köszönhetően, mert a fordító bájtkódra fordít. Ezt kihasználva, tudtam a kódot apróbb módosításokkal átültetni x86 alapú PC-ről az ARM processzoros Android operácós 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 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 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.

OpenCV:

Ingyenesen elérhető nyílt gépi látást megvalósító függvénykönyvtár. 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.