Budapesti Műszaki és Gazdaságtudományi Egyetem
Önállólaboratórium (VIMIAL01)

Autonóm mérőrobot

Önálló laboratórium beszámoló

Készítette:
Izsó András (UICLAA)

Konzulens:
Sárközy Péter

2019. 05. 26.

Tartalomjegyzék

1 Feladat összefoglalása

A félév során egy autonóm mérőrobot fejlesztésébe kezdtem. A robot hardvere szinte teljesen elkészült, valamint alkalmas távoli vezérlésre a kliens alkalmazásán keresztül. Az autonomitás első lépéseként a kamerakép alapján egy piros labda követését valósítottam meg.

1.1 Részletes leírás

A félév elején hosszú távú célként egy autonóm mérőrobot fejlesztését tűztem ki célul. A kész eszköz képes lesz kamerakép alapján térképet építeni a környezetéről, majd abban tájékozódni SLAM algoritmus segítségével.

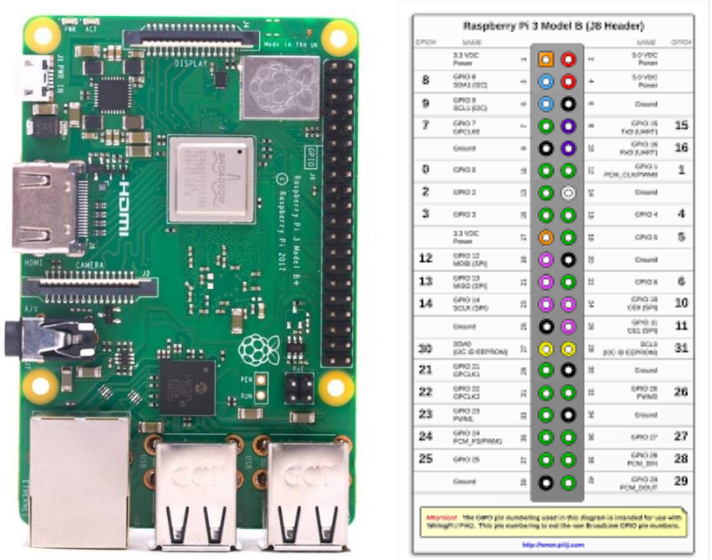
A félévben az elsődleges cél a hardverplatform elkészítése volt, mely biztos alapot nyújt a továbbfejlesztésre. Ennek természetesen alkalmasnak kellett lennie az önálló mozgásra, valamint fizikailag megfelelő méretűnek, hogy az alkatrészek elférjenek rajta.

A hardver összeállítása után a következő lépés a távvezérelhetőség megvalósítása volt. Ezt WebSocket technológiával implementáltam. A roboton fut egy WebSocket server és ehhez kapcsolódik a kliens alkalmazás. A kliens alkalmazáson keresztül nyomon követhetőnek kellett lenni a kameraképnek (minél inkább valós időben). Emellett az alkalmazás debug célból üzenetek megjelenítésére és küldésére is alkalmasnak kellett lennie. Valamint a szenzor adatoknak is láthatónak kellett lennie.

A félév végén első autonóm funkciónak egy piros labda követését valósítottam meg. Ez megfelelő választásnak tűnt, mivel könnyen tesztelhető, a robot szabályzását ki lehet vele tapasztalni, valamint a továbbiakban hasznos tapasztalat a kameraképen történő objektum felismeréshez.

2 Felhasznált eszközök

2.1 Raspberry Pi 3b+



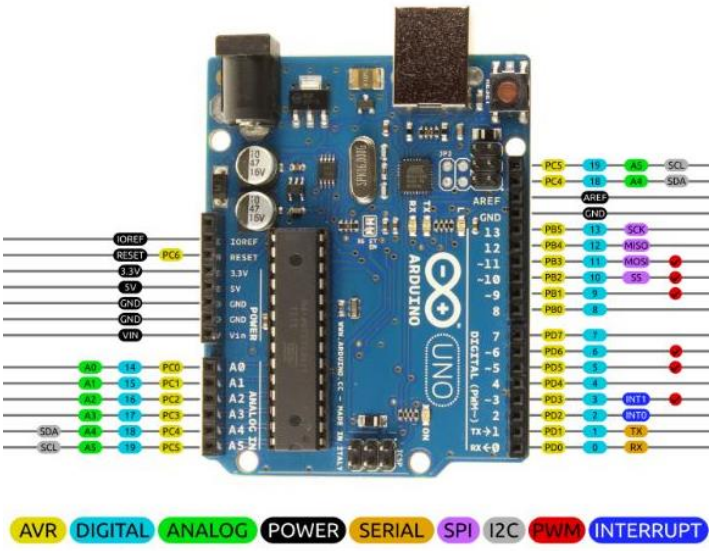
1. ábra: Balra: a Raspberry Pi mini számítógép, jobbra: a GPIO interfész kiosztása

A Raspberry Pi (1. ábra) egy sokak által használt mini számítógép. Előnye, hogy egy kb bankkártya méretű kártyán egy teljes értékű számítógépet valósít meg. Számomra mérete mellett azért jó választás, mivel rendelkezik saját kamera modullal, aminek a képe megfelelő minőségű. Emellett 1,4GHz-es, 64 bites, quad-core processzorral és 1GB RAM-mal kellően nagy számítási kapacitása van a

kamerakép feldolgozásához. Emellett az újabb verziók rendelkeznek beépített WiFi-vel, ami az ac szabványt is támogatja. Az eszköz képes AP módban is működni, így könnyen csatlakozhatunk hozzá a kliensről. A 40 lábas GPIO interfésze pedig a jövőben hasznos lehet különböző perifériák illesztésére.

2.2 Arduino UNO

Az Arduino UNO (2. ábra) ATmeg328P alapú nyílt forráskódú fejlesztőkártya. 14 digitális és 6 analóg I/O pinnel rendelkezik, a digitálisak közül 6 pulzus szélesség modulációval (PWM) is vezérelhető. A fejlesztés első fázisában Wemos D1 mini kártyát használtam. A félév közepén a Wemos kártya, és a saját motorvezérlője meghibásodott. Mivel az Arduino UNO rendelkezésre állt, valamint a Wemos is Arduino környezetben programozható, ezért ésszerű választásnak tűnt a folytatáshoz. Így egyszerű volt a már elkészült kódot migrálni. Az is mellette szól, hogy az újonnan beszerzett motorvezérlő 5V-os, a



2. ábra: Az Arduino UNO fejlesztőkártya, lábkiosztással

Raspberry viszont 3,3V-on működteti a GPIO csatlakozóit.

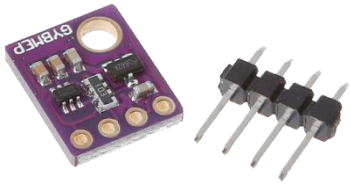
2.3 LSM303c

Az LSM303c (3. ábra) egy 3 tengelyes magnetométer és gyorsulásmérő. A szenzor I2C és SPI támogatással rendelkezik, határérték túllépés esetén interruptot küld. A választás azért erre a típusra esett, mivel mindkét szenzort tartalmazza, amely a robot autonóm irányításához szükséges, illetve kalibrálás után megfelelő pontosságú eredményt szolgáltat. Emellett az ára is alacsony.

3. ábra: LSM303c Magnetométer és gyorsulásmérő szenzor

2.4 BME280

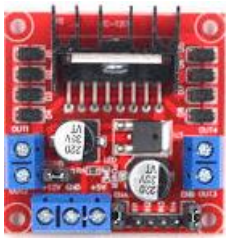
A BME-280 (4. ábra) egy 3-az-egyben szenzor, egymaga képes hőmérséklet, páratartalom, valamint légnyomás mérésére. A szenzor beépítve támogatja az I2C valamint az SPI buszokat, ami rendkívül kényelmessé teszi a használatát. További előnye a széles konfigurálhatósága.



4. ábra: BME280 Hőmérséklet, légnyomás, páratartalom szenzor

2.5 L298N

Az L298N (5. ábra) egy két csatornás, H-hidas DC motor vezérlő. A Wemos D1 motor shieldjének meghibásodását követően váltottam erre a típusra. Amellett, hogy ez gyorsan rendelkezésre állt az szól mellette, hogy sokkal robosztusabb kialakítású, akár 40V feletti motor vezérlő feszültséggel is tud operálni. Hátránya, hogy 5V logikai szint felett minimum 2,5V-al kell lennie a motor tápfeszültségnek, így az üzemidő nem használható ki teljesen (8,4V a jelenlegi akkumulátor maximális feszültsége).



5. ábra: L298N 2 csatornás H-díd

2.6 Vázszerkezet

A strukturális alapot egy lánctalpas platform nyújtja (6. ábra). A mechanikai stabilitás érdekében egy alumínium vázat választottam, amire egy műanyag lapot erősítettem szerelő felületnek.

6. ábra A felhasznált lánctalpas platform

2.7 Egyéb felhasznált eszközök

Akkumulátor: a rendszer energiaellátását egy két cellás LiPo (Lítium-Polimer) akkumulátor biztosítja, 8,4V maximális feszültséggel

Motor: 350 fordulat per percre képes 12V-on

Kapcsoló üzemű tápegység: 5V előállítására képes, akár 15V-ból is

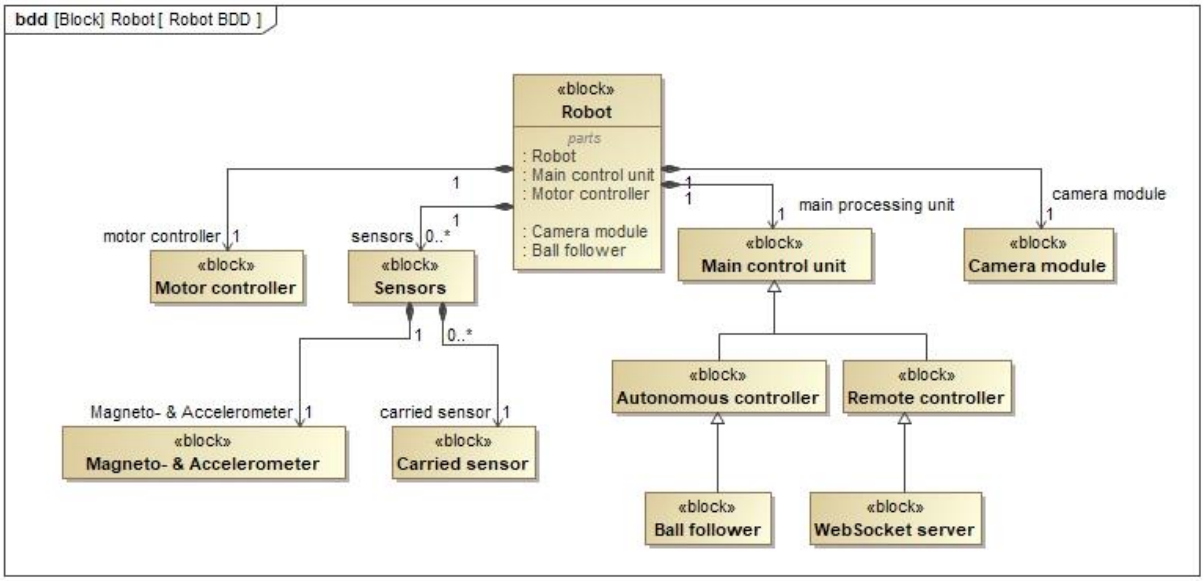
3 Feladat megvalósítása

3.1 Funkcionális felépítés

A robotnak szüksége van egy központi vezérlő egységre, ami a kamera képét, a szenzorok adatait és a felhasználótól érkező üzeneteket dolgozza fel. Mivel távolról is szeretnénk irányítani és autonóm működést is tudnia kell majd, ezért a központi egységet két részre lehet tovább osztani. Jelenleg távolról WebSocketen keresztül lehet irányítani a berendezést, autonóm működésként pedig a labda követés van megvalósítva. A leírtak alapján ezek a konkrét irányító komponensek.

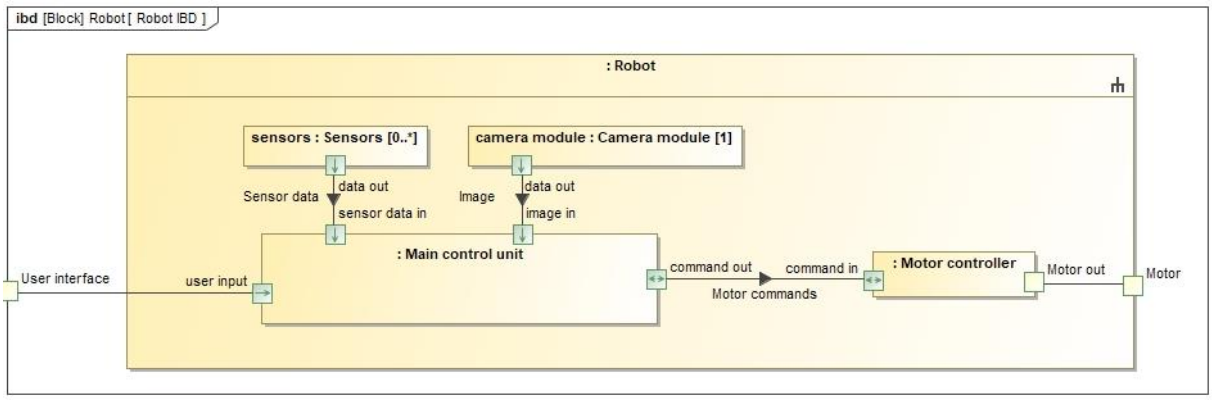
Szükség van egy kamera modulra is, aminek a képét feldolgozzuk, valamint szenzorokra is. A szenzorok közül a robot mindenképpen fel van szerelve egy magnetométerrel és gyorsulásmérővel, ami a tájékozódásban segíti a robotot. Ezen kívül lesznek szállított szenzorok is, amik a robot működésébe közvetlen nem szólnak bele, csak adatgyűjtést végeznek.

Ezen kívül szükség van egy motor vezérlő komponensre, ami a közvetlen motorvezérlést végzi. A teljes funkcionális felépítés a 7. ábrán látható.



7. ábra: A robot funkcionális egységeinek felépítése

A központi egység, tehát a felhasználói bemenet, a kamera kép valamint a szenzor adatok alapján küld parancsokat a motor vezérlőnek. A motor vezérlő ezek alapján végzi a közvetlen beavatkozást. A komponensek közötti kommunikációt a 8. ábra szemlélteti.



8. ábra: A funkcionális egységek kommunikációja

3.2 Hardver összeállítás

A központi egység funkcióját a Raspberry Pi tölti be. Hozzá kapcsolódik WebSocketen keresztül a kliens alkalmazás. A kamera a Raspberry saját MIPI CSI portján csatlakozik.

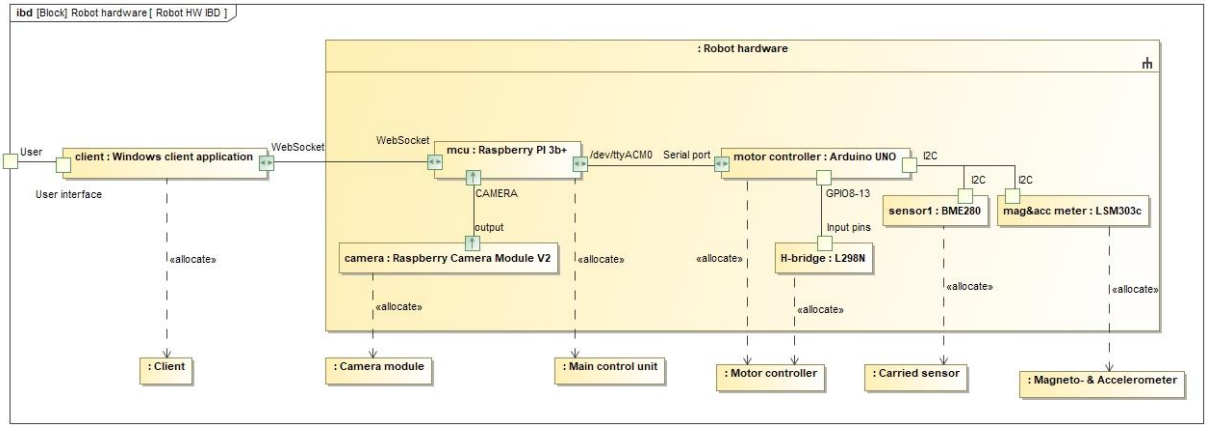
A motor vezérlő funkciót az Arduino UNO és az L298N valósítja meg. Az Arduino az ESP8266 helyére került be. Eredetileg kisebb mérete és fogyasztása miatt választottuk az ESP-t, valamint mivel rendelkezésre állt hozzá a saját motor shield-je, amivel egyszerűbb megvalósítani a motor vezérlést. Az Arduino jelenleg kiváltható lenne a Raspberry-vel, viszont az idő szűkössége miatt nem akartam már módosítani a platformon. Emellett a szenzorok könnyebb kezelhetősége miatt döntöttem a plusz mikrokontroller bevezetése mellett.

Az Arduino a parancsokat soros porton kapja a Raspberry-től, a szenzorokkal pedig I2C buszon kommunikál. Az állandó szenzorokat az LSM303c valósítja meg, szállított szenzorként jelenleg egyedül a BME280 funkcionál.

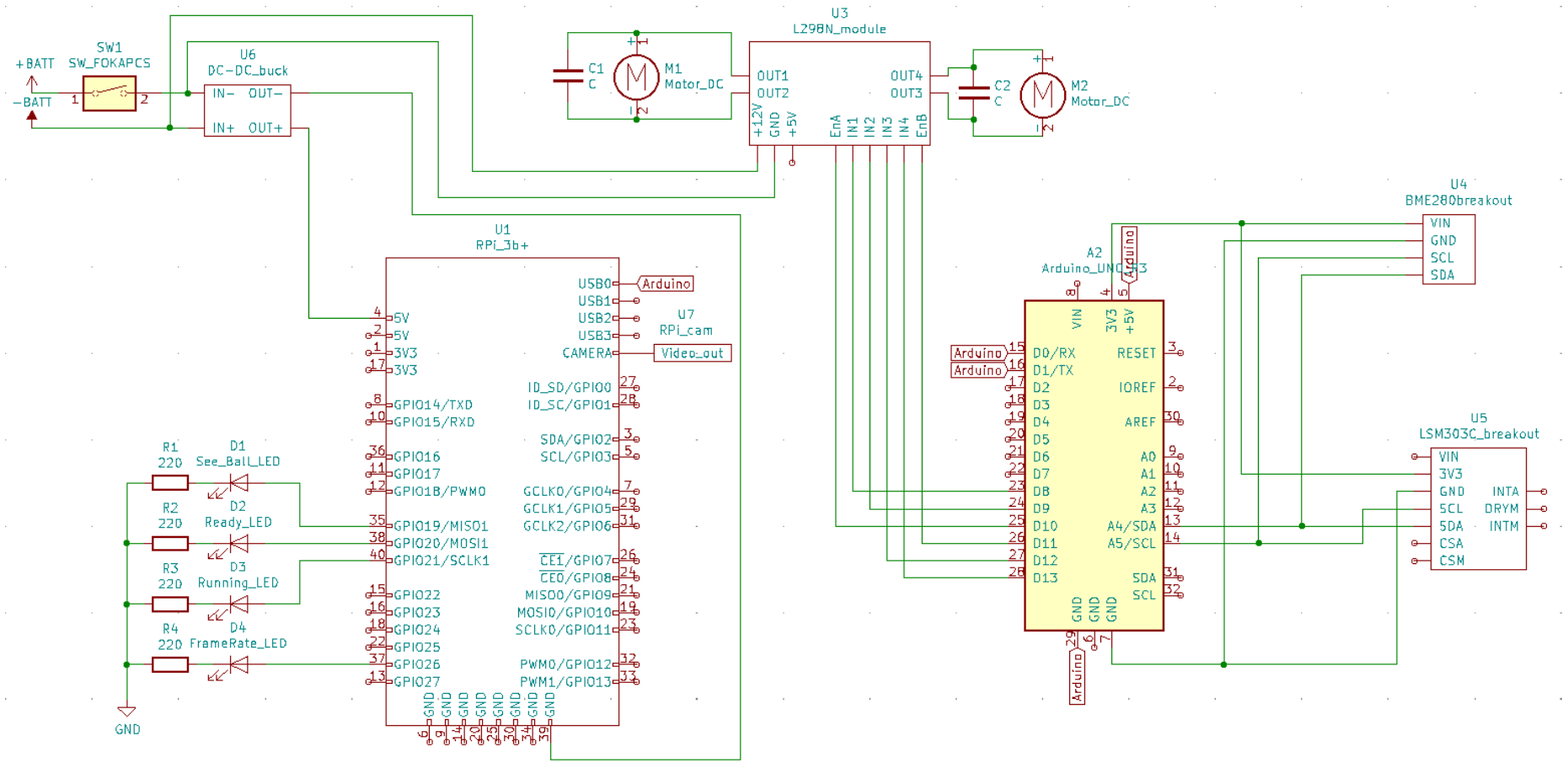
A motor percenként 350 fordulatra képesek, 12V-on. Sajnos a jelenlegi 2 cellás LiPo akkumulátor csak 8,4V-ot tud biztosítani. Ez elegendő a működéshez, viszont nem marad elegendő tartalék, hogy a gyorsan mozgó labdát is követni tudja. Emiatt következő fejlesztési lépésként az akkumulátort egy 6 cellás, 2-esével párhuzamosan kötött LiPo csomagra fogom lecserélni, mely 12,6V maximális feszültséget tud biztosítani. Emellett hosszabb üzemidőre lesz elegendő. Jelenleg terheléstől függően kb 1 órát képes akkumulátorról üzemelni a robot.

A hardver további fejlesztéseként a fogaskerekekre enkódert fogok készíteni, mellyel távolságot és sebességet fogok mérni. Ezen kívül miután próbapanelen már kialakult a végleges kapcsolás, szeretném direkt erre a célra tervezett NYÁK lemezre lecserélni, mivel ez biztonságosabb működést jelentene (nincsenek kicsúszó, összeérintődő röpzsinórok), valamint az alkatrészek fizikai elrendezésén is javítana.

A hardver allokáció a 9. ábrán, a teljes kapcsolási rajz a 10. ábrán látható.



9. ábra: A funkcionális egységek hardveres megvalósítása, és azok kommunikációja

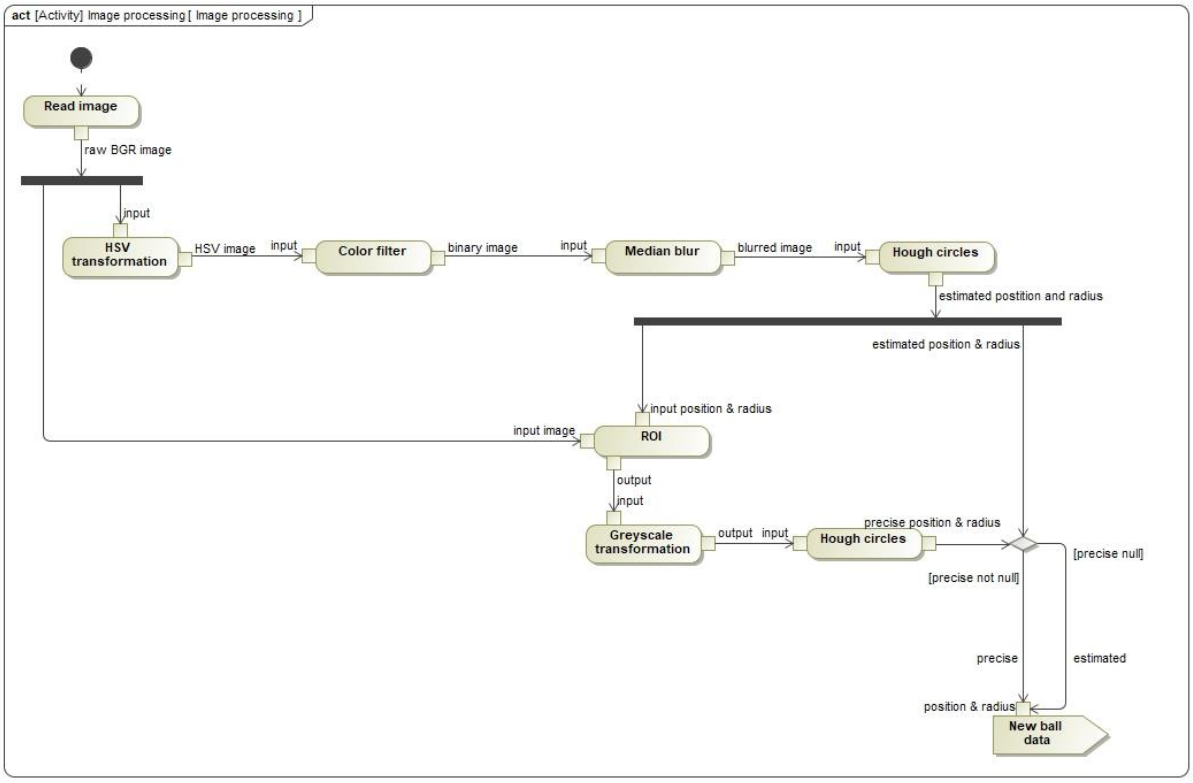


10. ábra: A teljes kapcsolási rajz

3.3 Szoftveres megoldások

3.3.1 Képfeldolgozás

A képfeldolgozást az OpenCV függvénykönyvtárral valósítottam meg. Ezzel már korábban is foglalkoztam, valamint kifejezetten Raspberry Pi-ra fordítható verziója is van. A képfeldolgozás folyamata a 11. ábrán követhető.



11. ábra: A képfeldolgozás lépsei

A kamerából a képet BGR (Blue, Green, Red) színtérben lehet kiolvasni. Ezt először HSV (Hue, Saturation, Value) tartományba transzformálom. A HSV tartomány szín szűréshez előnyösebb, mivel itt csak 1 érték reprezentálja a pixel színét. A másik kettő csak a világosságát és színességét állítja. Ezzel szemben az BGR színtérben mind a 3 értéktől függ a pixel színe.

A HSV színtérbe transzformált képen ezután piros szín szűrést végzek a [0,100,100] – [10,255,255], valamint [170,100,100] - [179,100,100] tartomány között. OpenCV-ben a Hue érték 0-360°helyett 0-179 értékek között van ábrázolva. A két tartományra azért van szükség, mert piros szín a tartomány elején és végén is van. A kimeneti képen a tartományokba eső pixelek fehérek, az azon kívüliek feketék lesznek.

A következő lépés a medián szűrés. A művelethez 3x3-as kernelt használok, mert ez még nem terheli le túlságosan a Raspberry-t, de kellően erőteljes hatása van. A medián szűrés eredményeként a kontúrok sokkal élesebbek lesznek, ami segíti a kör keresést. Próbálkoztam az erózió-dilatáció műveletével is, hogy minél pontosabb legyen a felismerés, viszont ezekkel már túlságosan lelassult a feldolgozás folyamata.

A képen a kört Hough transzformáció segítségével keresem meg. Ez alapesetben minden kört megtalálna a képen, ezért azzal a feltételezéssel élve, hogy a színszűrés elegendően pontos, hogy a képen csak a labda legyen fehér (eltekintve a kisebb zajoktól), azt a kontúrt tekintem a labdának, amit a transzformáció legmagabiztosabban ítél körnek.

A szűrt képen a kört nagy zajjal találja meg az eddigi algoritmus, mivel a labda egyik része hol fényesebb, a másik része sötétebb, így nem mindig ugyan úgy látszik a szűrt képen. A legnagyobb problémát a sugár ingadozása okozta, mivel a robot ez alapján tart tőle távolságot, és ha bemenet ennyire zajos nem lehet pontosan behangolni a szabályozót.

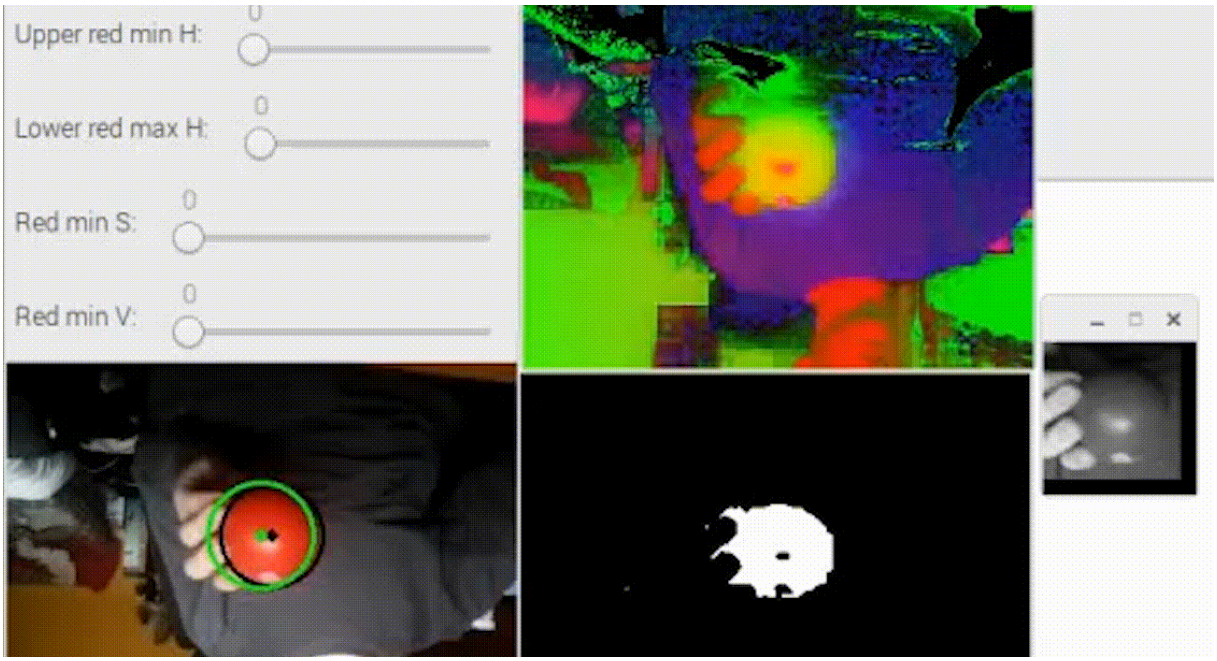
A problémát az alábbi módon orvosoltam: az eddigi eredményt csak becslésként használva, az eredeti képből kivágom a vélt labda-középpont körüli részt, a becsült sugár 1,5-szörösével (csak, hogy biztosan benne legyen a képben a labda). Ez egy olcsó művelet. A kapott kisebb képet áttranszformálom szürke-árnyalatossá (az OpenCV szürke-árnyalatos képeket kezel), hogy végrehajthassak rajta egy második kör keresést. Mivel ez a kép jellemzően nem túl nagy, és nagy részét a labda foglalja el, könnyebb dolga van az algoritmusnak, jut rá számítási kapacitás.

Szürke-árnyalatos képen a kör kereső algoritmus sokkal pontosabban találja meg a labdát (mivel itt biztosan nem vágtunk ki belőle részleteket). Ez szemmel is látható, a paraméter hangoláshoz készített egyszerű grafikus felületen. Olyan esetekben, amikor a szűrt képes felismerés 1,5-2-szer akkorának gondolja a labdát, mint valójában, a pontosított felismerés hozzávetőlegesen 10-20%-kal látja nagyobbnak. Ami még fontosabb, hogy az ingadozása is jelentősen kisebb. Az extra műveletekre jut elég erőforrás a Raspberry-n, a feldolgozó ciklus stabil 15-20 képkocka/másodperccel tud futni.

A kapott eredmények alapján ezután egy PID szabályzó állítja elő a szükséges motor meghajtás paramétereit, hogy a robot a labda felé nézzen, körülbelül 50cm távolságot tartva tőle.

A paraméterek hangolásához készítettem egy egyszerű grafikus felületet (12. ábra).

A csúszkákkal lehet a szín szűrés paraméterit hangolni. Alatta a kamerakép, a fekete kör a becsült, a zöld a pontosabb kör illesztés. Jobboldalt felül a kép HSV térben történő megjelenítése látható. Alatta a szín szűrés és medán szűrés eredménye. Jobb szélen a pontosabb kör kereséshez kivágott képrészlet látható.



12. ábra: A képfelismerés hangolásához készített grafikus felület

3.3.2 WebSocket server

Ahogy már korábban említettem, a robotot WebSocketen keresztül lehet távvezérelni. Ehhez szükség volt egy WebSocket szerverre. Ehhez a Pythonban elérhető Tornado csomagot használtam.

A szerver egy egyszerű üzenetkezelő ciklust futtat. Amennyiben érkezett üzenet azt értelmezi, és a megfelelő üzenetet küldi tovább a motorvezérlőnek. Elsősorban billentyűzet üzeneteket vár. A kliens alkalmazás küldi el neki, hogy a felhasználó milyen billentyűt nyomott le, vagy engedett fel. Ezeket a szerveralkalmazás tárolja, és mindig az aktuális billentyű állásoknak megfelelő parancsokat küldi a motor vezérlőnek. Emellett lehetőséget biztosít a vezérlőnek történő közvetlen üzenetküldésre is.

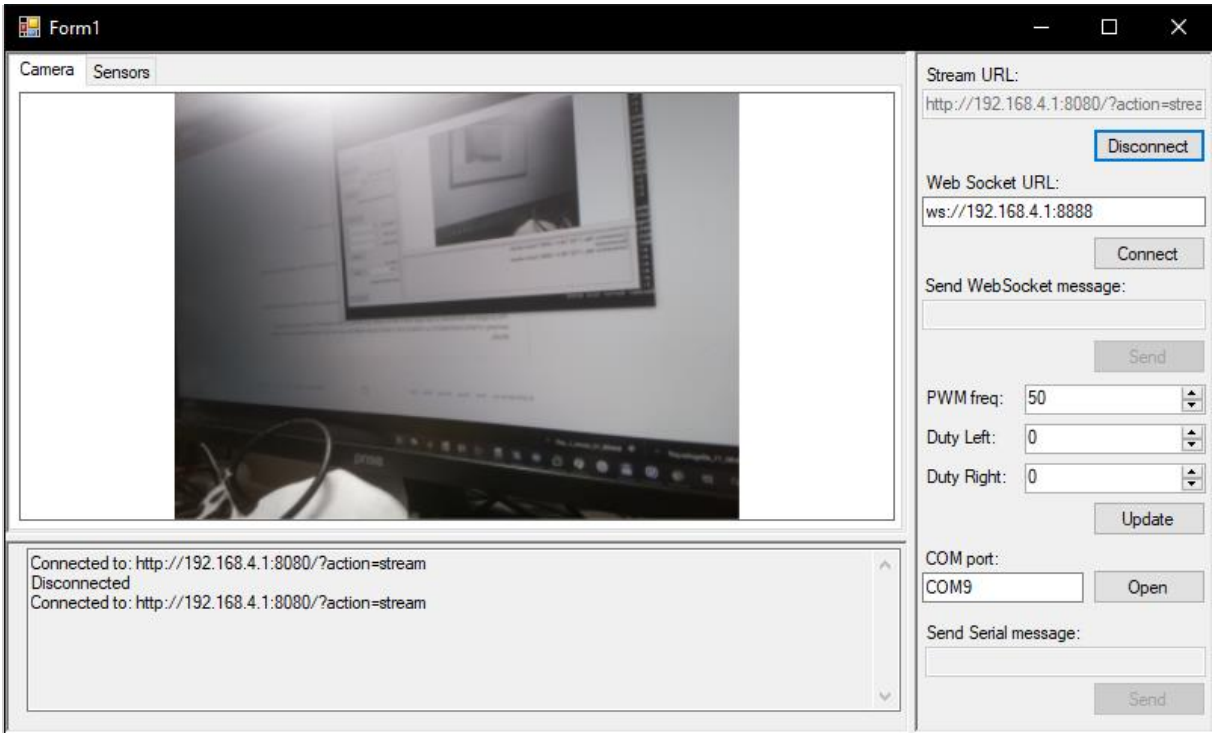
A szerver alkalmazás debug célból az összes soros porton küldött, illetve fogadott üzenetet továbbítja a kliens alkalmazásnak.

3.3.3 Kliens alkalmazás

A távoli irányításhoz, valamint a fejlesztési folyamat könnyítéséhez készítettem egy Windows Forms alapú applikációt (13. és 14. ábra). Ez a platform igaz, hogy elavultnak számít, de gyorsan lehet benne ilyen egyszerű alkalmazásokat készíteni, ezért ezt használtam.

Az alkalmazásban lehetséges követni a kamera képét, megközelítőleg élőben. Ehhez a Raspberry Pi-ra telepítettem egy MJPEG Streamer alkalmazást, ami weben keresztül közvetíti a kameraképet.

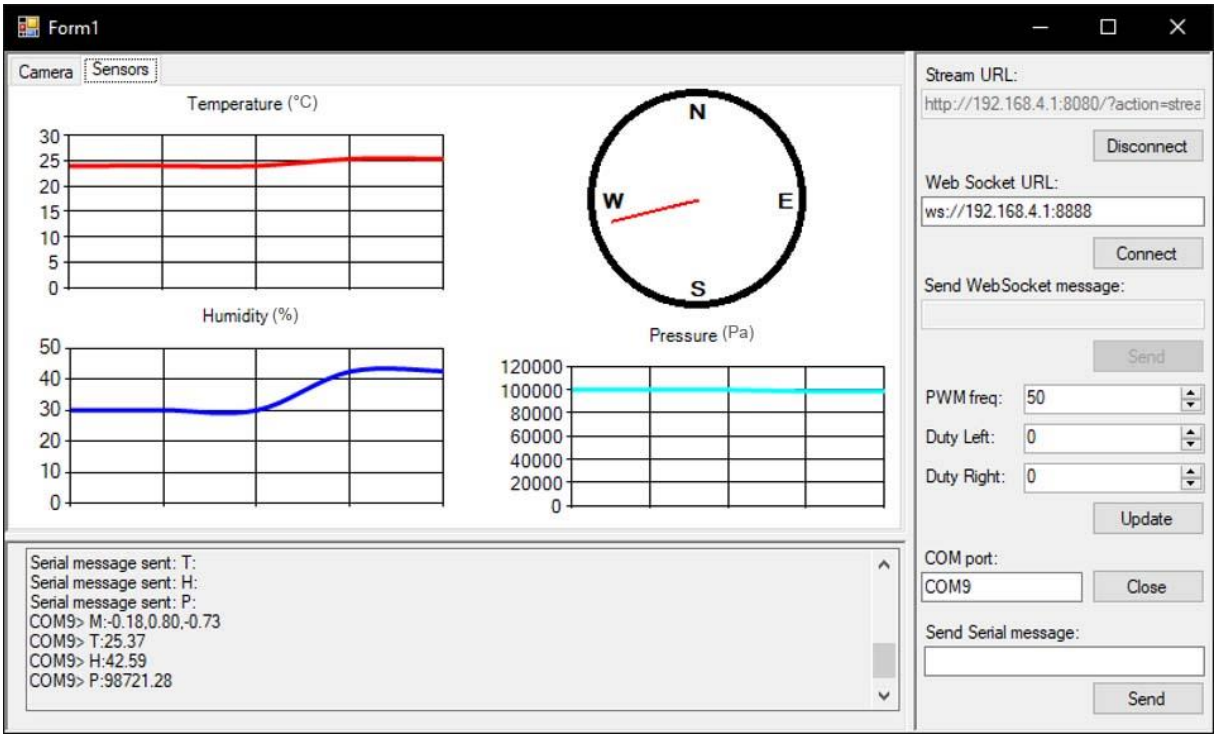
Az ablak alsó részében követhetjük nyomon az üzenetváltásokat a szerverrel, illetve itt jeleníthetők meg egyéb debug üzenetek. Jobb oldalt állíthatjuk be, hogy milyen címen érhető el a stream szolgáltatás, illetve a WebSocket szerver. Emellett ezen a panelen van lehetőségünk csatlakozás után üzenetet küldeni a szervernek. A felső TextBox-ot használva tetszőleges üzenet küldhető. Az alsóbb mezőkkel lehetőség van közvetlen PWM kitöltési tényezőt, illetve frekvenciát állítani. Ez rendkívül hasznosnak bizonyult a megfelelő paraméterek megtalálásához. Az előző motorvezérlővel a 2kHz-es PWM frekvencia bizonyult megfelelőnek, mivel itt elég tág dinamikatartományban lehet állítani a motor sebességét. Sajnos Arduino UNO-n nem sikerült megoldani a PWM frekvencia állítását, így az alapértelmezett 500Hz-et használom, amivel kissé szűkebb tartományban, de még használhatóan lehet a motorokat vezérelni.



3. ábra: A felhasználói felület. Nyomon követhető a kamera képe

A panel alján lehetőség van közvetlen a motorvezérlőhöz csatlakozni soros porton keresztül. A fejlesztést nagyban megkönnyítette, hogy nem kell feltétlenül a Raspberry-n keresztül kommunikálni az Arduino-val. Itt is lehetőség van tetszőleges üzenet küldésére, valamint a PWM tulajdonságok közvetlen megadására.

A középső panel másik fülére váltva megtekinthetőek a hőmérséklet, páratartalom, légnyomás szenzorok által mért adatok, 5mp-es frissítéssel. Emellett a magnetométer mérései alapján egy iránytű megjeleníti, hogy a robot milyen irányba áll jelenleg Északhoz képest.



4. ábra: A szenzor adatokat megjelenítő felület

4 Összefoglalás

A félév során egy hosszabb távú projekt fejlesztését sikerült elkezdenem. A fejlesztés során rengeteg új dolgot ismertem meg, többek között tanultam a DC motorok vezérléséről, a LiPo akkumulátor tulajdonságairól, és tapasztalatot szereztem hardverépítés gyakorlatában. Emellett a Raspberry Pi platform teljesen új területnek bizonyult, Linux rendszerrel is csak az egyetemi órák kapcsán foglalkoztam. A Raspberry-n futó kódot Python nyelven írtam. Tapasztalataim alapján erre a platformra ez a standard, ezzel lehet legkönnyeben megvalósítani. Korábban viszont sosem foglalkoztam ezzel a nyelvvel, így ez is teljesen új volt számomra.

A félév végére a sok nehézség (új környezet, hardver meghibásodás) sikerült eljutni a hardver majdnem 100%-os véglegesítéséig, a távoli irányíthatóság megvalósításáig, valamint egy kezdetleges autonóm funkciót is sikerült implementálni.

5 Források

https://coderwall.com/p/gvknya/python-websocket-server

https://www.raspberrypi.org/documentation/configuration/wireless/access-point.md

https://www.deciphertechnic.com/install-opencv-python-on-raspberry-pi/

https://learn.adafruit.com/lsm303-accelerometer-slash-compass-breakout/calibration

https://docs.opencv.org/3.4/db/df6/tutorial_erosion_dilatation.html

https://docs.opencv.org/3.1.0/d4/d13/tutorial_py_filtering.html

https://www.pyimagesearch.com/2015/12/28/increasing-raspberry-pi-fps-with-python-and-opencv/

https://ebldc.com/?p=86

A linkek ellenőrizve: 2019. 05. 25. 23:50