

Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Szélessávú Hírközlés és Villamosságtan Tanszék

## Mobil adatgyűjtő egység fejlesztése

DIPLOMATERV

*Készítette:*

Takács Péter

*Konzulens:*

Varga Lajos

2015. december 20.

# Köszönetnyilvánítás

Elsősorban szeretném köszönetet mondani konzulensemnek, Varga Lajosnak a rengeteg segítségért és hogy bármikor fordulhattam hozzá a kérdéseimmel.

Köszönettel tartozom még Vécsi Sándornak a hasznos tanácsokért, illetve a hardvertervezésben és a forrasztásban nyújtott segítségéért.

Továbbá szeretném megköszönni Gondár Flórának a képek készítésében és szerkesztésében, valamint a Diplomaterv nyelvi megformálásában nyújtott segítséget.

## HALLGATÓI NYILATKOZAT

Alulírott *Takács Péter*, szigorló hallgató kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző, cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2015. december 20.

---

*Takács Péter*  
hallgató

## FELADATKIÍRÁS

# Tartalomjegyzék

<b>Ábrák jegyzéke</b>	<b>6</b>
<b>Táblázatok jegyzéke</b>	<b>8</b>
<b>Kivonat</b>	<b>9</b>
<b>Abstract</b>	<b>10</b>
<b>1. Bevezetés</b>	<b>11</b>
1.1. Áttekintés . . . . .	11
1.2. A dolgozat felépítése . . . . .	12
<b>2. Rendszerleírás</b>	<b>13</b>
<b>3. Vezérlőegység</b>	<b>16</b>
3.1. Előzetes követelmények . . . . .	16
3.2. Hardvertervezés . . . . .	17
3.2.1. Energiaellátó rendszer . . . . .	17
3.2.2. Mikrokontroller . . . . .	21
3.2.3. UART jelszintillesztő . . . . .	23
3.2.4. ISM sávú rádiós adó-vevő egység . . . . .	24
3.2.5. DC motor-vezérlő . . . . .	26
3.2.6. Szervomotor-vezérlő . . . . .	31
3.2.7. Léptetőmotor-vezérlő . . . . .	32
3.2.8. Szenzorok . . . . .	34
3.2.9. Nyomtatott áramkör tervezése . . . . .	41
3.3. Beágyazott szoftver . . . . .	45
3.3.1. Mikrokontroller inicializálása . . . . .	45
3.3.2. UART kommunikáció . . . . .	46
3.3.3. ISM sávú rádiós kommunikáció . . . . .	47
3.3.4. Motorok vezérlése . . . . .	49
3.3.5. Telemetria- és szenzoradatok gyűjtése . . . . .	50
3.3.6. Vezérlő parancsok . . . . .	51
3.3.7. Főprogram . . . . .	54

<b>4. Raspberry Pi</b>	<b>56</b>
4.1. A Raspberry Pi kommunikációs csatornái . . . . .	57
4.2. A Raspberry Pi inicializálása . . . . .	58
4.3. Élő kamerakép előállítása . . . . .	59
4.4. Parancsfeldolgozó szoftver . . . . .	61
<b>5. Kliensprogram</b>	<b>66</b>
5.1. A kliensprogram előzetes követelményei . . . . .	66
5.2. A kliensprogram indítása és bezárása . . . . .	67
5.3. Soros kapcsolat kialakítása . . . . .	68
5.4. UDP kapcsolat kialakítása . . . . .	69
5.5. Élő kamerakép megjelenítése . . . . .	69
5.6. Telemetria- és szenzoradatok feldolgozása . . . . .	70
5.7. A mobil adatgyűjtő egység vezérlése, konfigurálása . . . . .	70
<b>6. Tesztek</b>	<b>73</b>
6.1. Funkcionális tesztek . . . . .	73
6.2. Hatótávolság tesztelése . . . . .	75
6.2.1. Beltéri hatótávolság tesztelése . . . . .	75
6.2.2. Kültéri hatótávolság tesztelése . . . . .	75
6.3. Élő kamerakép tesztelése . . . . .	75
6.4. Éjszakai teszt . . . . .	76
6.5. Üzemidő tesztelése . . . . .	76
6.6. Tesztelés során tapasztalt hibák . . . . .	78
<b>7. Összefoglalás</b>	<b>79</b>
7.1. A fejlesztés értékelése . . . . .	79
7.2. Továbbfejlesztési lehetőségek, távlati célok . . . . .	81
<b>Fogalomtár</b>	<b>82</b>
<b>Irodalomjegyzék</b>	<b>83</b>
<b>Függelék</b>	<b>86</b>
F.1. A vezőrlőegység kapcsolási rajza és nyomtatott áramköri terve . . . . .	86
F.2. Az USB-s ISM sávú rádiós adó-vevő egység kapcsolási rajza és nyomtatott áramköri terve . . . . .	96
F.3. A léptetőmotor paraméterei . . . . .	98
F.4. A mobil adatgyűjtő egység prototípusa . . . . .	99

# Ábrák jegyzéke

2.1.	A rendszer blokkvázlata . . . . .	13
3.1.	A lítium-polimer akkumulátor által leadott energia és a terhelt akkumulátor kapocsfeszültségének viszonya . . . . .	18
3.2.	Az energiaellátó rendszer kapcsolási rajza . . . . .	19
3.3.	A mikrokontroller és környezetének kapcsolási rajza . . . . .	22
3.4.	Az UART jelszintillesztő áramkör kapcsolási rajza . . . . .	23
3.5.	Az ISM sávú rádiós adó-vevő áramkör kapcsolási rajza . . . . .	24
3.6.	USB-s ISM sávú rádiós adó-vevő egység . . . . .	26
3.7.	A DC motor-vezérlő kapcsolási rajza . . . . .	27
3.8.	A DC motor négy-negyedes vezérlési módja . . . . .	30
3.9.	A szervomotor hatása a kapcsolóüzemű tápegység kimeneti feszültségére .	32
3.10.	Bipoláris léptetőmotor kapcsoló tranzisztorainak állapota, és fázisainak gerjesztése teljes lépéses üzemmódban . . . . .	33
3.11.	A léptetőmotor-vezérlő kapcsolási rajza . . . . .	34
3.12.	A kétirányú áramerősség mérő áramkör blokkábrája . . . . .	35
3.13.	A sebességmérő kapcsolási rajza . . . . .	37
3.14.	A hőmérő szenzor kapcsolási rajza . . . . .	39
3.15.	Az infravörös távolságérzékelő blokkdiagramja . . . . .	39
3.16.	A transzimpedanciás erősítő kapcsolási rajza . . . . .	40
3.17.	Az akkumulátor celláinak és a tápegységek kimeneti feszültségének mérési elve . . . . .	41
3.18.	Az újragyártott nyomtatott áramkör alső és felső rétege . . . . .	44
3.19.	A GPS modul által küldött NMEA üzenetek . . . . .	47
3.20.	A rádiós adatcsomag felépítése . . . . .	48
3.21.	A telemetria- és szenzoradatokat tartalmazó 60 bájtos adatcsomag felépítése	50
3.22.	A vezérlő parancsok felépítése . . . . .	51
3.23.	A mikrokontrolleren futó beágyazott szoftver folyamatábrája . . . . .	55
4.1.	A Raspberry Pi kommunikációs csatornái . . . . .	57
4.2.	A JPEG tömörítésű képek előállításának és küldésének folyamata . . . . .	60
4.3.	A Raspberry Pi-n futó parancsfeldolgozó Python script folyamatábrája .	62
4.4.	Az UART-on beérkező adatcsomag feldolgozásának folyamata . . . . .	63
4.5.	Az UDP porton beérkező adatcsomag feldolgozásának folyamata . . . . .	64

5.1.	A kliensprogram felhasználói felülete csatlakozás előtt és csatlakozás után	68
6.1.	Élő kamerakép háttérvilágítás nélkül és háttérvilágítással	76
6.2.	Az utasítássorozat idődiagramja	77
F.1.1.	A kapcsolási rajz hierarchikus ábrája	86
F.1.2.	Az energiaellátó rendszer kapcsolási rajza	87
F.1.3.	A mikrokontroller kapcsolási rajza	88
F.1.4.	A DC motor-vezérlő és a DC motor árammérő szenzorának kapcsolási rajza	89
F.1.5.	A léptetőmotor-vezérlő kapcsolási rajza	90
F.1.6.	Az ISM sávú rádiós adó-vevő áramkör kapcsolási rajza	91
F.1.7.	Az UART jelszintillesztő kapcsolási rajza	92
F.1.8.	A szenzorok és a csatlakozók kapcsolási rajzai	93
F.1.9.	A vezérlőegység nyomtatott áramköri tervének felső rétege	94
F.1.10.	A vezérlőegység nyomtatott áramköri tervének alsó rétege	95
F.2.1.	Az USB-s ISM sávú rádiós adó-vevő egység kapcsolási rajza	96
F.2.2.	Az USB-s ISM sávú rádiós adó-vevő egység nyomtatott áramköri tervének felső és alsó rétege	97
F.3.1.	A léptetőmotor paraméterei	98
F.4.1.	A mobil adatgyűjtő egység prototípusa előlnézetből	99
F.4.2.	A mobil adatgyűjtő egység prototípusa hátulnézetből	100
F.4.3.	A mobil adatgyűjtő egység prototípusa az egyik oldalról	101
F.4.4.	A mobil adatgyűjtő egység prototípusa a másik oldalról	102
F.4.5.	A mobil adatgyűjtő egység prototípusa felülnézetből	103

# Táblázatok jegyzéke

3.1.	Fogyasztási adatok . . . . .	19
3.2.	Az energiaellátó rendszer terheléses tesztjének eredményei . . . . .	20
3.3.	Az RS-232-es szabvány feszültségszintjei és a hozzátartozó logikai értékek . .	23
3.4.	Az A3941 gate meghajtó IC és az IRL7833 n-csatornás MOSFET paraméterei .	27
3.5.	A vezérlő parancsok kódja és jelentése . . . . .	54
4.1.	A RaspiCam legfontosabb tulajdonságai . . . . .	59
5.1.	A jelenlegi szoftververziójú kliensprogramban kiadható vezérlő és konfigura- rációs parancsok . . . . .	72
6.1.	A funkcionális tesztek eredményei . . . . .	74

# Kivonat

Napjainkban egyre nagyobb jelentősége van az autonóm működésű, illetve távvezérelhető adatgyűjtő és felderítő célú mobil járműveknek. A mobil adatgyűjtő egység legnagyobb előnye, hogy használata esetén nem szükséges embert küldeni a feltérképezni kívánt területre.

Diplomatervezési feladatom a mobil adatgyűjtő egység helyváltoztatását és a kamera mozgatását lehetővé tevő motorok vezérlő elektronikájának megtervezése és megépítése, valamint a hozzátartozó beágyazott szoftver kifejlesztése. A mobil adatgyűjtő egység vezeték nélküli (Wi-Fi, illetve az irányítás hatótávolságának növelése érdekében 433 MHz-es ISM sávú rádiós) kapcsolaton keresztül távvezérelhető, képes élő kameraképet és egyéb mérési adatokat (hőmérésklet, helyadatok, telemetria adatok, stb.) eljuttatni a vezérlő személyzet számára, miközben könnyű terepen halad.

A mobil adatgyűjtő egység a kamera háttérvilágításának köszönhetően képes éjszakai felderítésre, valamint ütközéselhárító rendszerrel is rendelkezik.

A dolgozatban részletesen bemutatom a hardver tervezésének és megépítésének lépéseit, a mikrokontroller, illetve a Wi-Fi-kapcsolat kialakításáért és az élő kamerakép közvetítéséért felelős Raspberry Pi modul beágyazott szoftvereit, továbbá ismertetem a hordozható eszközön (laptop) futó Matlab kliensprogramot és az elvégzett teszteket.

# Abstract

Nowadays the remote controlled or autonomously working reconnaissance mobile vehicles are becoming more and more important. The biggest advantage of the mobile data collection unit is that it is not necessary to send personnel to the remote mapping area.

The task of my Master's Thesis is to design and build the control electronics responsible for the camera rotation and the movement of the data collection unit as well as to develop the embedded software. The mobile data collection unit can be controlled remotely through a wireless connection (Wi-Fi, and an additional 433 MHz ISM band radio connection to extend the range), furthermore it can provide a live video stream and other measurement data (temperature, location information, telemetry data, etc.) to the operators while it is moving on smooth terrain.

Due to the camera backlight, the mobile data collection unit can be used at night reconnaissance, furthermore it has collision avoidance system.

In this thesis the steps of designing and building the hardware as well as the embedded software of the microcontroller and the Raspberry Pi module what is used for streaming the live video and establish the Wi-Fi connection and the Matlab client program and the tests of the developed mobile data collection unit are presented in detail.

## 1. fejezet

# Bevezetés

### 1.1. Áttekintés

Napjainkban egyre nagyobb jelentősége van az autonóm működésű, illetve távvezérelhető adatgyűjtő és felderítő célú mobil járművek kutatásának és fejlesztésének. Az autonóm működésű, vagy a távvezérlehető egységek legnagyobb előnye, hogy használatuk esetén nem szükséges embert küldeni a feltérképezni kívánt területre. A technológia rohamos fejlődésének következtében a 21. században már számtalan területen alkalmaznak mobil egységeket a legkülönbözőbb célokra. Egyes esetekben, például sugárszennyezett környezetben nem lehetséges a terület felderítése, illetve mérési adatok gyűjtése az emberi élet kockáztatása nélkül. A 2011-ben történt fukushima atomerőmű-baleset során is távvezérelhető robotokat küldtek a sérült reaktorblokkok radioaktív sugárzásának megmérésére.

A mobil felderítő egységek egy további felhasználási területét képezik a rendőrségi, tűzszerészeti, vagy katonai alkalmazások, de az 1960-70-es évektől kezdve a Marsra is számos adatgyűjtő és felderítő szondát (Mars Rover) juttattak a marsi kőzetek vizsgálata, illetve élő organizmusok és víz utáni kutatás céljából.

Diplomatémám meghatározása során a fő célkitűzésem egy összetett rendszer alapjaitól való megtervezése (mind a hardver, mind a szoftver) és kifejlesztése volt. A konzulensemmel közösen egy, az előzőekben bemutatott eszközökhez hasonló működésű és felépítésű, vezeték nélküli hálózaton keresztül távvezérelhető mobil adatgyűjtő egység fejlesztését választottuk feladatomnak. A mobil adatgyűjtő egység hardverét a későbbi továbbfejlesztési lehetőségeket szem előtt tartva úgy terveztem meg, hogy az képes legyen teljesen autonóm működésre.

A mobil adatgyűjtő egység fejlesztését a Diplomatervezés 1 tárgy keretében kezdtem, az egész rendszer és a vezérlőegység hardverének megtervezésével. A hardver építését, a szükséges szoftverek megírását, a rendszer integrálását és az elkészült mobil adatgyűjtő egység tesztelését 2015 nyarán és őszén végeztem.

## **1.2. A dolgozat felépítése**

A bevezető után, a dolgozat második fejezetében bemutatom a mobil adatgyűjtő egység rendszertervét és röviden kifejtem az egyes részegységek működését, illetve a rendszerben betöltött szerepüket.

A harmadik fejezetben ismertetem a mobil adatgyűjtő egység általam tervezett vezérlőegységét, részletesen leírom a hardverelemek tervezésének és működésének folyamatát, végül bemutatom a mikrokontrolleren futó beágyazott szoftver működését.

A negyedik fejezet a Raspberry Pi inicializálási beállításait, a Wi-Fi-kapcsolat kialakítását, a parancsfeldolgozó Python script működését és az élő kamerakép előállításának és átvitelének módjait tartalmazza.

A ötödik fejezet a hordozható egységen futó kliensprogramot mutatja be, amelyen keresztül vezérelhető és konfigurálható a mobil adatgyűjtő egység. A telemetria- és szenzor-adatok feldolgozása és megjelenítése, valamint az élő kamerakép megjelenítése is a kliensprogramban történik.

A hatodik fejezetben ismertetem a rendszerintegráció után elvégzett teszteket és a tesztek során tapasztalt hibákat.

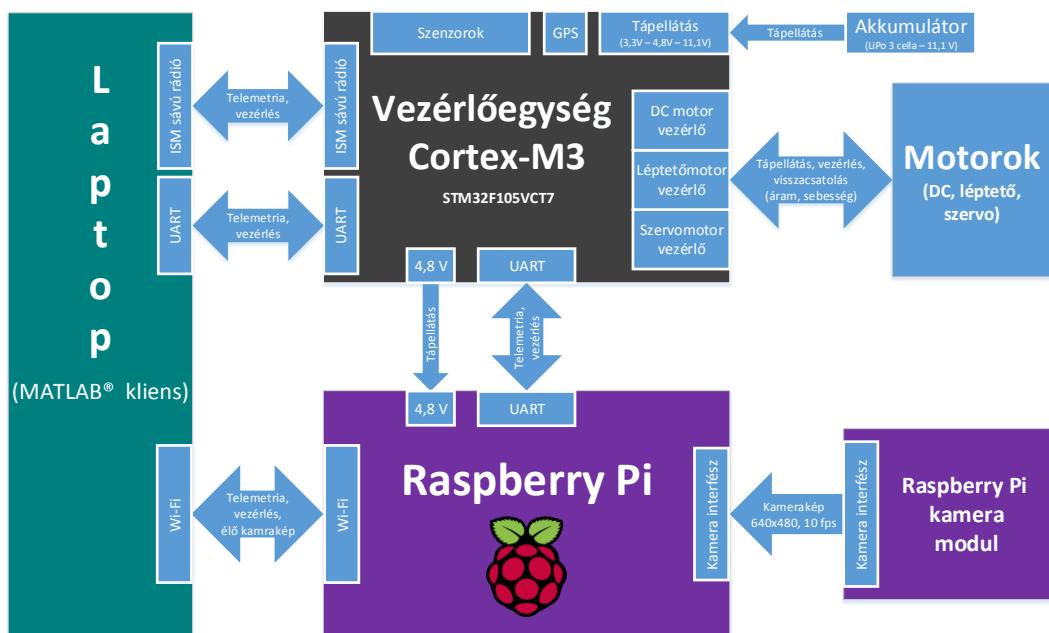
A hetedik fejezetben értékelem az elvégzett fejlesztési munkát és bemutatom a mobil adatgyűjtő egység rendszerében rejlő továbbfejlesztési lehetőségeket.

## 2. fejezet

# Rendszerleírás

Ebben a fejezetben bemutatom a mobil adatgyűjtő egység felépítését, a rendszertervet és a mobil adatgyűjtő egység fő részeinek vázlatos működését.

A mobil adatgyűjtő egység felépítése három fő részre bontható. A vezérlőegységre, melynek feladata az energiaellátás biztosítása, a motorok vezérlése és meghajtása, illetve a telemetria- és szenzoradatok gyűjtése és továbbítása. A Raspberry Pi-re, illetve a hozzá tartozó kamera modulra, amely az élő kamerakép szolgáltatását és a Wi-Fi-kapcsolat kialakítását biztosítja. A harmadik főegység egy hordozható eszköz (laptop, okostelefon) és a rajta futó kliensprogram, amely a telemetria- és szenzoradatok feldolgozását és megjelenítését, illetve az élő kamerakép megjelenítését végzi, valamint a kliensprogramon keresztül vezérelhető és konfigurálható a mobil adatgyűjtő egység. A továbbiakban részletesen kifejtem a 2.1. ábrán látható rendszer működését.



2.1. ábra. A rendszer blokkvázlata

A mobil adatgyűjtő egység mechanikai kialakítását tekintve lényegében egy 1:10 méretarányú távirányítós autó, melyből minden vezérlő elektronikát kiszereltem, így tulajdonképpen csak a négy kerékből, az egyenáramú motorból (DC motor), a szervomotorból és az alvázból áll. Erre a vázra építettem rá a mobil adatgyűjtő egység általam tervezett elemeit.

A vezérlőegység hardverén találhatók a motorvezérlő integrált áramkörök (IC), melyek lehetővé teszik a kamera forgatását és a mobil adatgyűjtő egység irányítását. A mobil adatgyűjtő egységet a DC motor mozgatja, amit egy külső H-híd hajt meg. A H-híd kapcsoló MOSFET-jeit egy gate meghajtó IC vezérli a mikrokontroller által kiadott impulzusszélesség-modulált (PWM) jel kitöltési tényezőjének függvényében.

A mobil adatgyűjtő egység kormányozhatóságát egy szervomotor teszi lehetővé, a szervomotort szintén egy PWM jel impulzusszélességének változtatásával lehet vezélni. A kamera forgatásához léptetőmotort használtam, így pozíció-visszacsatolás nélkül pontosan ismerjük a kamera aktuális helyzetét. A léptetőmotort egy integrált H-hidakat tartalmazó léptetőmotor-meghajtó IC vezérli, a léptetőmotor a belső felépítéséből adódóan egy impulzusra egy egységnyi szöget fordul el.

A vezérlőegységen számos adatgyűjtő szenzor (hőmérséklet, megvilágítás, helymeghatározó modul (GPS), stb.) található, illetve a hardver úgy van kialakítva, hogy igény esetén tüskesoron keresztül további szenzorok csatlakoztathatók. További szenzorok csatlakoztatása esetén a mikrokontroller szoftverét módosítani kell. A szenzoradatokon kívül a mobil adatgyűjtő egység számos telemetriaadatot (akkumulátor celláinak feszültsége, tápfeszültségek, DC motor árama) is szolgáltat magáról.

A laptop és a vezérlőegység közti kommunikációs kapcsolat kialakítására három lehetőség van:

- Közvetlen, kábeles soros (UART) összeköttetés, ezt csak a fejlesztés során használtam, mivel nagyon egyszerű és a másik két opcióval ellentében nem igényel külön hardvert, csupán egy jelszintillesztő IC szükséges hozzá.
- Közvetlen, ISM sávú (433 MHz) rádiós kapcsolat. Az ISM sávú rádiós összeköttetésnek akkor van szerepe, ha a mobil adatgyűjtő egység a Wi-Fi hatótávolságán kívülre kerül. A közvetlen UART és a közvetlen ISM sávú rádiós kapcsolaton keresztül az élő kamerakép továbbítására nincs lehetőség, csak a telemetria- és szenzoradatok, illetve a vezérlő parancsok küldhetők rajta.
- Közvetett, kétirányú kapcsolat kialakítása. A vezérlőegység UART-on keresztül kommunikál a Raspberry Pi-vel, a Raspberry Pi pedig Wi-Fi-n keresztül kapcsolódik a laptophoz. A Wi-Fi-kapcsolaton keresztül mind a vezérlő parancsok, mind a telemetria- és szenzoradatok, mind az élő kamerakép átküldhető.

A mobil adatgyűjtő egység energiaellátását egy háromcellás, 11,1 V névleges feszültségű, 3000 mAh-s lítium-polimer akkumulátor biztosítja. A teljes rendszer működéséhez három különböző tápfeszültség előállítása szükséges. A DC motor és a léptetőmotor közvetlenül a 11,1 V-os akkumulátorról van meghajtva, a szervomotor és a Raspberry Pi 4,8 V-on üzemel, a vezérlőegység többi digitális része pedig 3,3 V-on működik. A 4,8 V-os és a 3,3 V-os

feszültségeket kapcsolóüzemű feszültségcsökkentő áramkörök állítják elő az akkumulátor 11,1 V-os feszültségéből.

A mobil adatgyűjtő egység másik fő része a Raspberry Pi és a Raspberry Pi-hez kap- ható kis méretű digitális kamera modul (a továbbiakban RaspiCam). A RaspiCam szab- ványos interfészen keresztül, szalagkábellel csatlakoztatható a Raspberry Pi-hez. A Ras- piCam további előnye, hogy a könyvtári Python függvényeknek köszönhetően rendkívül egyszerűen és rugalmasan programozható. A RaspiCam tulajdonságait tekintve (4.1. táblázat) maradéktalanul képes a feladatkiírásban megkövetelt 640x480 képpont felbontású és 10 kép/másodperc frissítésű élő kamerakép előállítására. A Raspberry Pi az előző bekez- désben már említett UART kapcsolaton keresztül kommunikál a vezérlőegység mikrokont- rollerével, illetve Wi-Fi-kapcsolaton keresztül a laptoppal. Mivel a Raspberry Pi nem ren- delkezik beépített Wi-Fi modullal, ezért a Wi-Fi-kapcsolat kialakításához egy USB porton csatlakoztatható Wi-Fi adaptort használtam.

A felhasználó a kliensprogramon keresztül irányíthatja és konfigurálhatja a mobil adatgyűjtő egységet, illetve a kliensprogramon keresztül felügyelheti a vezérlőegység által szol- gáltatott telemetria- és szenzoradatokat. Az élő kamerakép megjelenítése szintén a kliens- programban történik.

A dolgozat további fejezeteiben részletesen bemutatom a mobil adatgyűjtő egység három fő részének működését.

### **3. fejezet**

## **Vezérlőegység**

Ebben a fejezetben ismertetem a vezérlőegységgel szemben támasztott előzetes követelményeket, a hardvertervezés és hardverélesztés lépései, részletesen kiterve a vezérlőegység minden elemére, illetve bemutatom a mikrokontrolleren futó beágyazott szoftver működését.

### **3.1. Előzetes követelmények**

A 2. fejezetben bemutatott rendszer fejlesztése során az első lépés a vezérlőegység megtervezése volt. A tervezés megkezdése előtt specifikáltam, hogy a vezérlőegységnek milyen kritériumokat kell teljesítenie. Az előzetes követelmények a következők:

- Energiaellátás biztosítása az egész rendszer számára.
- Kétirányú kommunikációs kapcsolat kialakítása a Raspberry Pi modullal.
- Kétirányú ISM sávú rádiós kapcsolat kialakítása a laptoppal.
- DC motor vezérlése és meghajtása, az előre- és a hátramenet biztosítása.
- Szervomotor vezérlése, a kanyarodás biztosítása.
- Léptetőmotor vezérlése, a kameraforgatás biztosítása.
- Telemetria- és szenzoradatok gyűjtése és továbbítása.
- Csatlakoztatási felületek kialakítása a szenzorok bővíthetőségének céljából.
- A mechanikai kialakítás illeszkedjen a mobil adatgyűjtő egység alvázához.

### 3.2. Hardvertervezés

A hardvertervezés első lépéseként ki kell választani egy, a feladatra alkalmas kapcsolási rajz szerkesztő és nyomtatott áramkör (röviden nyák) tervező programot. A korábbi ismeretek alapján, valamint az előző munkáim során már korábban megrajzolt alkatrészek és alkatrészlenyomatok miatt a választásom az Altium Designer nevű hardvertervező programra esett, de a hardvertervezési feladatra a Protel, az Orcad, az Eagle, stb. is alkalmas lett volna.

Mivel a vezérlőegység elemei logikailag jól elkülöníthetők, ezért a kapcsolási rajzukat is külön dokumentumokban terveztem meg, majd egy hierarchikus ábrán összekötöttem a kapcsolódási pontokat. A hierarchikus ábrát, a teljes kapcsolási rajzot és a nyomtatott áramköri tervet a Függelék F.1. fejezete tartalmazza.

A tervezés során a legfőbb szempontok a minél nagyobb integráltság elérése, ezáltal csökkentve a hardver méretét és a vezetékek számát, valamint a költséghatékonyság voltak. A következő alfejezetekben részletesen bemutatom a vezérlőegység elemeit.

#### 3.2.1. Energiaellátó rendszer

A mobil adatgyűjtő egységet a 2. fejezetben már említett 11,1 V névleges feszültségű és 3000 mAh elektromos kapacitású lítium-polimer akkumulátor látja el energiával. A lítium-polimer akkumulátorok legfőbb előnye, hogy a méretéhez képest nagy az energiasűrűsége, valamint nagy a maximális kisütési árama.<sup>1</sup> Az akkumulátorban tárolt energia a következő képpel számolható:

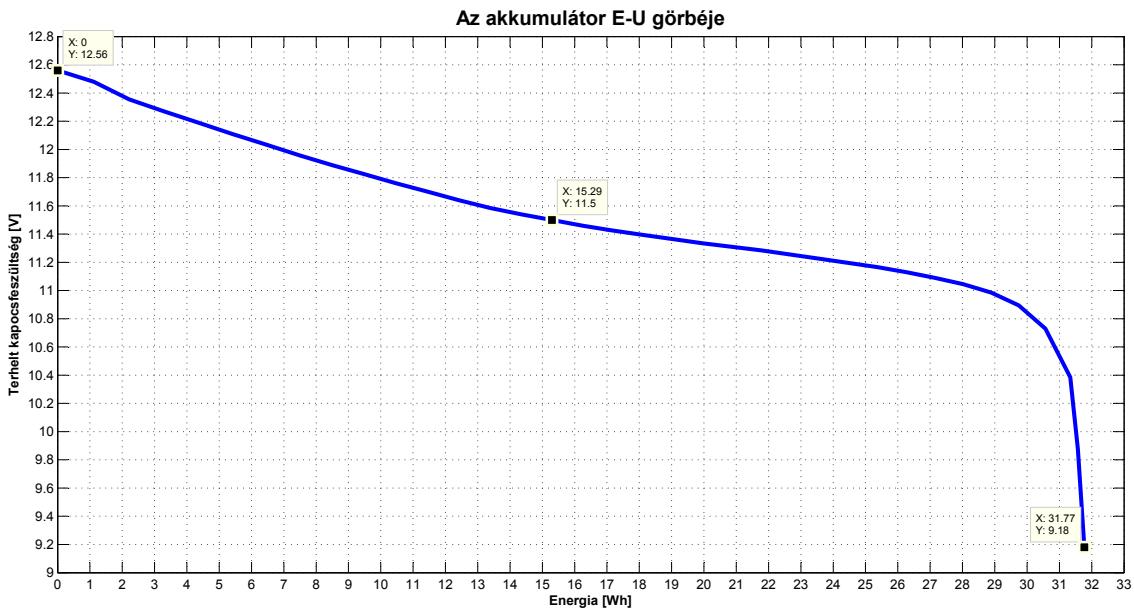
$$E_{akk} = U_{névleges} \cdot C_{elektromos}, \quad (3.1)$$

ahol  $E_{akk}$  az akkumulátorban tárolt energia Wh-ban,  $U_{névleges}$  az akkumulátor névleges feszültsége V-ban,  $C_{elektromos}$  az akkumulátor elektromos kapacitása Ah-ban. Tehát a mobil adatgyűjtő egység akkumulátorában tárolt energia 33,3 Wh.

Teszteléssel megállapítottam az akkumulátor által leadott energia és a terhelt akkumulátor kapocsfeszültségének viszonyát (3.1. ábra). Az akkumulátort először teljesen feltölтtem, majd az akkumulátorral sorosan bekötöttem egy  $7\Omega$ -os, 15 W-os ellenállást és egy árammérőt, valamint az akkumulátor kapcsaira egy feszültségmérőt csatlakoztattam. Az akkumulátor lemerülésig három percenként leolvastam az árammérő és a feszültségmérő értékét. A kapott eredményekből kiszámoltam a mérési pontokban a pillanatnyi leadott teljesítményt, majd Matlab segítségével kirajzoltam a teljesítmény-idő függvényt. Az akkumulátor energiájának kiszámításához az időegységet lenormáltam órára, majd időben integráltam a teljesítmény-idő grafikon görbe alatti területét, ezzel kiküszöböltem, hogy a mérés során a kisütési áram nem konstans a csökkenő kapocsfeszültség miatt.

---

<sup>1</sup>Az általam használt típus maximális kisütési árama 60 A, ez azt jelenti, hogy folyamatos 60 A-es áramfelvételű üzem mellett a teljesen feltöltött akkumulátor 3 perc alatt lemerülne.

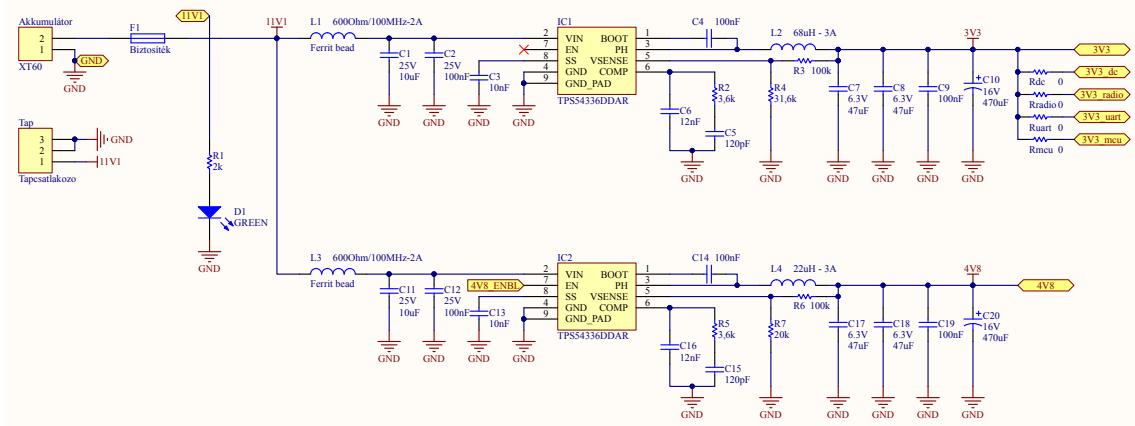


**3.1. ábra.** A lítium-polimer akkumulátor által leadott energia és a terhelt akkumulátor kapocsfeszültségének viszonya

A 3.1. ábráról leolvasható, hogy az akkumulátor energiája 31,77 Wh, ez szinte tökéletesen megegyezik az előzetesen számolt 33,3 Wh-val, ha figyelembe veszzük, hogy az élettartam megóvása céljából a töltőáramkör nem tölti fel 100%-ra, míg a teszt során nem merítettem le 0%-ra az akkumulátort.

Az energiaellátó rendszer tervezésének első lépéseként meg kell vizsgálni a mobil adatgyűjtő egység fő elemeinek működési feszültségeit. Az alkatrészek adatlapjai alapján három feszültségszintet kell előállítani. A DC motort és a léptetőmotort közvetlenül az akkumulátor látja el energiával, a Raspberry Pi, a szervomotor és a távolságérzékelő szenzor működési feszültsége 4,8 V, a mikrokontroller és a többi IC, illetve szenzor pedig 3,3 V-os feszültségszintet használ. A 4,8 V és a 3,3 V kialakításához csökkentenünk kell az akkumulátor feszültségét, erre többféle feszültségcsökkentő módszer létezik.

Egy lehetséges feszültségcsökkentő módszer az úgynevezett lineáris feszültségcsökkentő (LDO) alkalmazása. A lineáris feszültségcsökkentők stabil kimeneti feszültséget állítanak elő, amíg a bemeneti feszültség egy megadott működési tartományon belül van. A lineáris feszültségcsökkentők hátránya az átalakítás hatásfoka, mivel a fölös teljesítményt eldisszipálják, hővé alakítják. Például a 11,1 V 3,3 V-á alakítása során a hatásfok 30%, de a 4,8 V-á alakítás során is csak 43%, ilyen alacsony hatásfok mellett az üzemidő jelentősen lerövidül, ez nem megengedhető. A hatásfok növelése céljából kapcsolóüzemű feszültségcsökkentő áramkörök alkalmaztam. A kapcsolóüzemű tápegységek hátránya a lineáris feszültségszabályozókkal szemben, hogy a kapcsolóüzemből adódó zavarok/zajok zavarhatják a megtáplált áramkörök működését a kimenet nem megfelelő szűrése esetén, illetve több kiegészítő, külső alkatrész is szükséges a megépítésükhez [4]. Az energiaellátó rendszer kapcsolási rajza a 3.2. ábrán látható.



**3.2. ábra.** Az energiaellátó rendszer kapcsolási rajza

A tervezés második lépéseként ki kell választani egy, a feladatra alkalmas kapcsolóüzemű vezérlőáramkört. Az alkatrészválasztás előtt adatlap és mérési adatok alapján meghatároztam a mobil adatgyűjtő egység elméleti fogyasztásának maximumát. A szervomotor fogyasztásáról nincsenek adataim, ezért egy labortáp és egy PWM jelgenerátor segítségével megmértem a konstans áramfelvételét maximális nyomaték kifejtése közben,<sup>2</sup> a pillanatnyi áramfelvétel ennél magasabb is lehet, ezért a 4,8 V-os tápegység kimenetére egy  $470 \mu\text{F}$ -os pufferkondenzátort helyeztem. Az elméleti fogyasztási maximumokat és a hardverépítés során mért tényleges fogyasztási adatokat a 3.1. táblázat tartalmazza.

Alkatrész	Elméleti maximum fogyasztás	Valós fogyasztás	Tápfeszültség
Raspberry Pi B	730 mA	610 mA	4,8 V
Szervomotor	–	2140 mA	4,8 V
Távolságérzékelő szenzor	42 mA	26 mA	4,8 V
Mikrokontroller	70 mA	43 mA	3,3 V
ISM sávú rádió	43 mA	40 mA	3,3 V
GPS + aktív antenna	80 mA	54 mA	3,3 V
Szenzorok, LED-ek	90 mA	74 mA	3,3 V

**3.1. táblázat.** Fogyasztási adatok

A táblázatból kiolvasható, hogy a 4,8 V-on működő eszközök áramfelvétellel elérheti a 2,8 A-t, azonban üzemszervű használat mellett az áramfelvétel csak körülbelül 1,5 A. Gondolva a későbbi felhasználás során esetlegesen csatlakoztatni kívánt 4,8 V-on üzemelő szenzorok fogyasztására is, a választásom a Texas Instruments TPS54336 típuszámú feszültségsökkentő kapcsolóüzemű tápegységére esett. A TPS54336 bemeneti feszültség-tartománya 4,5–28 V, maximális kimeneti árama 3 A. A kimeneti feszültséget a 3.2. ábrán látható  $R_3$ ,  $R_4$ , illetve  $R_6$ ,  $R_7$  feszültségosztókkal lehet beállítani úgy, hogy a VSENSE lábra 0,8 V essen. A TPS54336 kapcsolási frekvenciája fix 340 kHz, a frekvencia, a bemeneti

<sup>2</sup>A maximális nyomaték kifejtését úgy értem el, hogy ütközésgig kivezéreltem a kormányt, ahol az autó mechanikai kialakítása már megakadályozza a szervomotor továbbforgását. A gyakorlati felhasználás során a kormány nem tud ütközésgig elfordulni, ezt szoftveresen megakadályoztam. Rendeltetésszerű használat mellett a fogyasztás 0,8 A alatt marad. A motor indulásakor bekövetkező pillanatnyi áramcsúcs értékét a megfelelő műszerek hiányában nem tudtam lemérni.

feszültség, a kimeneti feszültség és a kimeneti áram ismeretében az adatlapban található képletek alapján meghatározhatóak a bemeneti és a kimeneti szűrők passzív elemeinek értékei. A passzív elemek kiválasztásakor fontos, hogy a kondenzátorok átütési feszültsége és a kimeneti tekercs maximális árama nagyobb legyen, mint az üzemszerű használat során fellépő maximális feszültség- és áramérték. A TPS54336 kompenzáció áramkörének értékei szintén az adatlapban található képletek alapján számolhatók. A TPS54336 áramszabályozással tartja stabilan a kimeneti feszültséget, így a bekapsolási jelenség során a kimeneti kondenzátorok nagy töltési árama miatt a tápegység letiltaná a kimenetet. Ennek elkerülése érdekében a TPS54336 rendelkezik úgynevezett „slow start” funkcióval, melyel beállítható a bekapsolás és a szabályozás kezdete közti holtidő az SS láb és a GND közé kötött kondenzátorral. A 4,8 V-os táp engedélyező (EN) lábat a mikrokontrollerre kötöttem, így az akkumulátor lemerülésekor a tápegység kikapsolásával csökkenthető a fogyasztás [14].

A 3,3 V-on működő eszközök összes áramfelvételle egy nagyságrenddel kisebb, mint a 4,8 V-on működőké, így a TPS54336 maximális 3 A-es kimeneti árama túlzásnak tűnhet, azonban a tápegység hatásfoka 100 mA fölött már 90-95% között mozog, ezért a 3,3 V előállításához is ezt a kapcsolóüzemű tápegységet használtam. A 3,3 V-os táp passzív elemeinek értéke szintén az adatlapban szereplő képletek alapján lettek meghatározva [14]. A 3.2. ábrán látható, hogy a 3,3 V-os tápfeszültséget 0 Ω-os ellenállásokon keresztül osztom szét a vezérlőegység részegységeihez. A 0 Ω-os ellenállások beforrasztása előtt a részegységek külön, külső labortáppal is megtáplálhatóak, így az áramfelvételük a táravezetékbe sorosan bekötött multiméterrel közvetlenül megmérhető. Az energiaellátó rendszer ezen kívül tartalmaz egy labortáp-csatlakozót,<sup>3</sup> egy bemeneti feszültség-indikátor fénykibocsátó diódát (LED) és egy olvadóbiztosítékot áramkör védelmi szempontok miatt.

A TPS54336-os IC-ket ingyenes alkatrész mintaként rendeltem a Texas Instruments-tól, ezáltal is csökkentve a fejlesztési költségeket.

Az alkatrészek beforrasztása után elvégeztem az energiaellátó rendszer terheléses tesztjét. A tápegységek kimeneteire különböző értékű 15 W-os teljesítményellenállásokat forrasztottam és megmértem a kimeneti feszültséget különböző terhelések mellett. A terheléses teszt eredményeit a 3.2. táblázat tartalmazza.

Eset	Kimeneti feszültség	Ellenállás	Áramerősség	Teljesítmény
3,3 V-os tápegység				
Üresjárás	3,38 V	szakadás	0,00 A	0,0 W
1. eset	3,37 V	6,0 Ω	0,56 A	1,9 W
2. eset	3,36 V	1,5 Ω	2,24 A	7,5 W
4,8 V-os tápegység				
Üresjárás	4,82 V	szakadás	0,00 A	0,0 W
1. eset	4,81 V	6,0 Ω	0,80 A	3,8 W
2. eset	4,80 V	2,2 Ω	2,18 A	10,5 W
3. eset	4,80 V	1,8 Ω	2,67 A	12,8 W

**3.2. táblázat.** Az energiaellátó rendszer terheléses tesztjének eredményei

<sup>3</sup> A fejlesztés során a vezérlőegységet áramkorlátozott labortápról tápláltam meg, így védve az áramkört egy esetleges rövidzár okozta túlárammal szemben.

### 3.2.2. Mikrokontroller

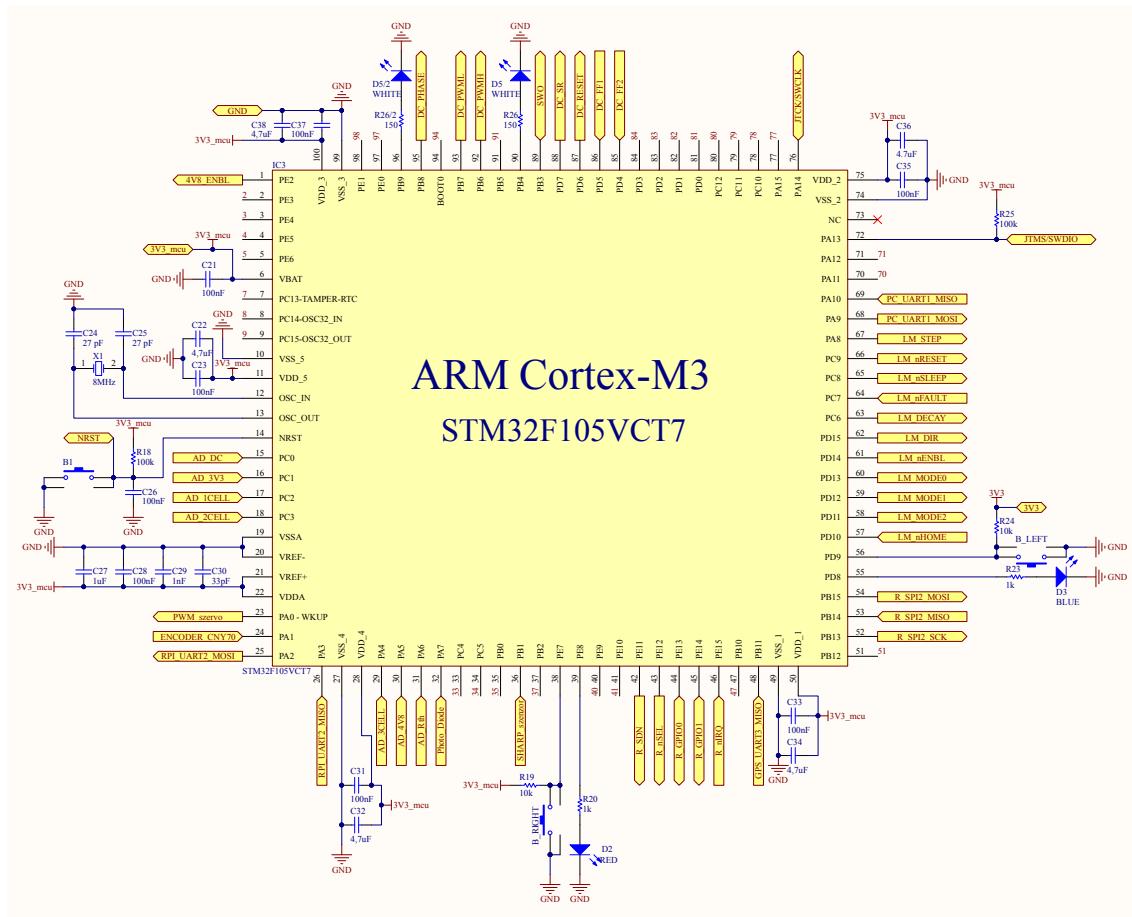
A mikrokontroller kiválasztása előtt specifikáltam, hogy milyen hardveresen támogatott perifériákra van szükség a vezérlési és kommunikációs feladatok ellátásához, illetve a telemetria- és szenzoradatok gyűjtéséhez. A mikrokontrollernek legalább a következő típusú és számú perifériákat kell tartalmaznia:

- 3 számláló a DC motor, a léptetőmotor és a szervomotor vezérléséhez szükséges PWM jelek előállításához.
- 1 számláló a sebességmérő szenzor kimeneti jelfrekvenciájának méréséhez.
- 3 UART periféria a laptopnal, a Raspberry Pi-vel, és a GPS modullal való kommunikációhoz.
- 1 soros periféria-illesztő (SPI) az ISM sávú rádiós IC-vel való kommunikációhoz.
- 9 csatornás analóg-digitális átalakító (ADC) az analóg kimenetű szenzorokhoz.
- 25 általános felhasználású ki- és bemeneti (GPIO) láb a logikai vezérlőjelekhez, nyomógombokhoz, LED-ekhez.

A fenti felsorolás alapján a választásom az STMicroelectronics STM32F105VCT7 típusú 100 lábú LQFP tokozású mikrokontrollerére esett. Az STM32F105VCT7 256 kB flash<sup>4</sup> és 64 kB SRAM memóriát tartalmaz, maximális órajel-frekvenciája 72 MHz. További adatok és tulajdonságok megtalálhatóak a mikrokontroller adatlapjában [36]. A mikrokontroller és környezetének kapcsolási rajza a 3.3. ábrán látható.

---

<sup>4</sup>Nem felejtő memória, elektronikusan törölhető és programozható.



**3.3. ábra.** A mikrokontroller és környezetének kapcsolási rajza

A digitális áramkörök érzékenyek a tápfeszültség ingadozására, ezért egy  $4,7\mu F$ -os és egy  $100\text{ nF}$ -os hidegítő kondenzátort helyeztem el a mikrokontroller tárplábanál a zavarok kiszűréséhez. Az NRST lábra egy nyomógombot kötöttem, a nyomógomb megnyomásával lehetőség van a mikrokontroller hardveres újraindítására. A nyomógombot hardveresen pergésmentesítettem, így a nyomógomb egyszeri megnyomásakor a mikrokontroller csak egyszer kezdi előlről a beágyazott szoftver futtatását.

A kommunikációhoz, vezérléshez és adatgyűjtéshez használt lábak kiosztásánál az egyik fő szempont, hogy a nyáktervezés során a lehető legkevesebb huzal keresztezze egymást. Két további nyomógombot és két LED-et is raktöttem a mikrovezárló lábaira. A nyomógombokra a fejlesztés kezdeti fázisában és a tesztelés során volt szükség, a feltételes utasítások feltételeként a nyomógombok állapotát adtam meg. A LED-ek indikátorként működnek, így nyomon követhető, hogy mely utasítássorozatot végzi a mikrokontroller, valamint a fejlesztés során a szoftverhibák felderítését is segítik. A mikrokontroller a rendszer órajelét egy külső 8 MHz-es kvarckristályból állítja elő belső fáziszárt hurkos (PLL) szorzók segítségével. A mikrokontroller nem használt lábait a későbbi bővítmény lehetőség céljából tüskesoros csatlakozókra vezettem ki.

A mikrokontroller programozása egy STM32F0DISCOVERY típusú fejlesztői kártyán keresztül történik. A STM32F0DISCOVERY-n hardveresen kiválasztható, hogy a fejlesztői kártyán lévő ST-LINK/V2 nevű programozó eszköz az integrált, vagy a hat érintkezős

tüskesoron keresztül csatlakoztatott mikrokontrollert programozza. A programozás három vezetékes (órajel, adat, föld) SWD interfészen keresztül történik, a másik három vezeték a mikrokontroller szoftverhibáinak felderítéséhez szükséges [32].

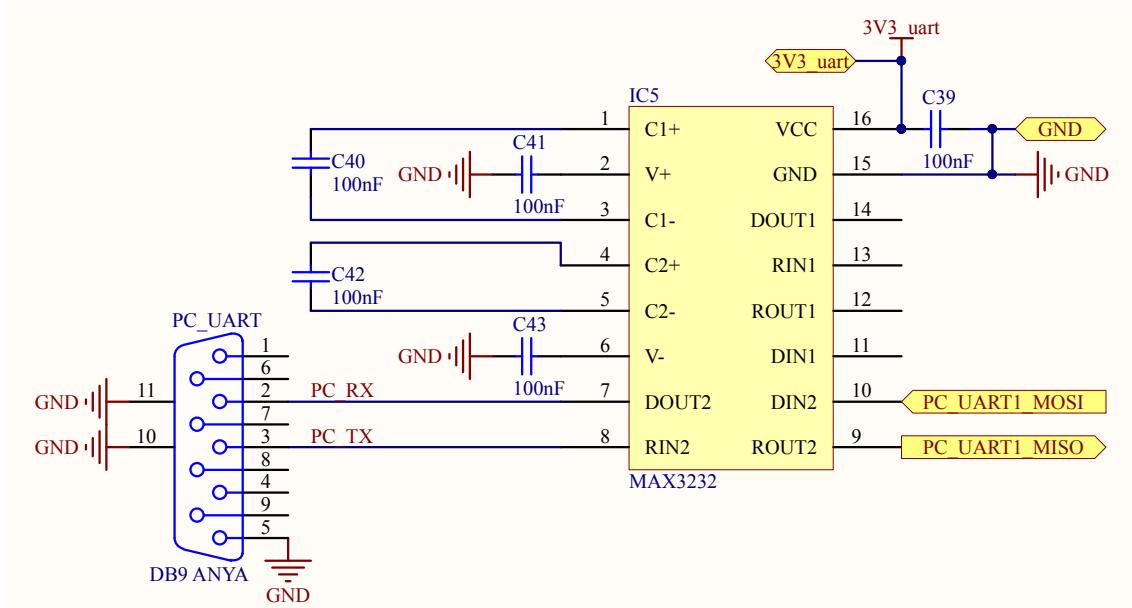
### 3.2.3. UART jelszintillesztő

A laptop és a mikrokontroller közti közvetlen, kábeles kommunikáció soros porti illesztőn keresztül valósul meg. Az UART protokoll full-duplex<sup>5</sup> összeköttetést biztosít az eszközök között. A mikrokontrollerben implementált UART periféria a rendszer alacsony (0 V – logikai 0) és magas (3,3 V – logikai 1) feszültségszintjével üzemel. A laptop UART protokollja a fizika réteget tekintve az RS-232-es szabványnak felel meg, melynek feszültségszintjeit a 3.3. táblázat tartalmazza:

Feszültségtartomány	Logikai érték
3 – 12 V	0
-12 – -3 V	1
-3 – 3 V	nem megengedett

**3.3. táblázat.** Az RS-232-es szabvány feszültségszintjei és a hozzáartozó logikai értékek

A jelszintek közti feszültségkülönbségek eltérését valamilyen, az UART protokoll fizikai rétegeit illesztő áramkörrel lehet megoldani. Erre a célra a Texas Instruments MAX3232-es jelszintillesztő integrált áramkörét alkalmaztam. A MAX3232 a megfelelő feszültséget a töltéspumpa elvén állítja elő. A soros jelszintillesztő áramkör kapcsolási rajza a 3.4. ábrán látható [17].



**3.4. ábra.** Az UART jelszintillesztő áramkör kapcsolási rajza

<sup>5</sup> A kapcsolaton keresztül egyidejűleg minden irányba küldhetők az adatok, külön vezeték van a vevőnek és az adónak.

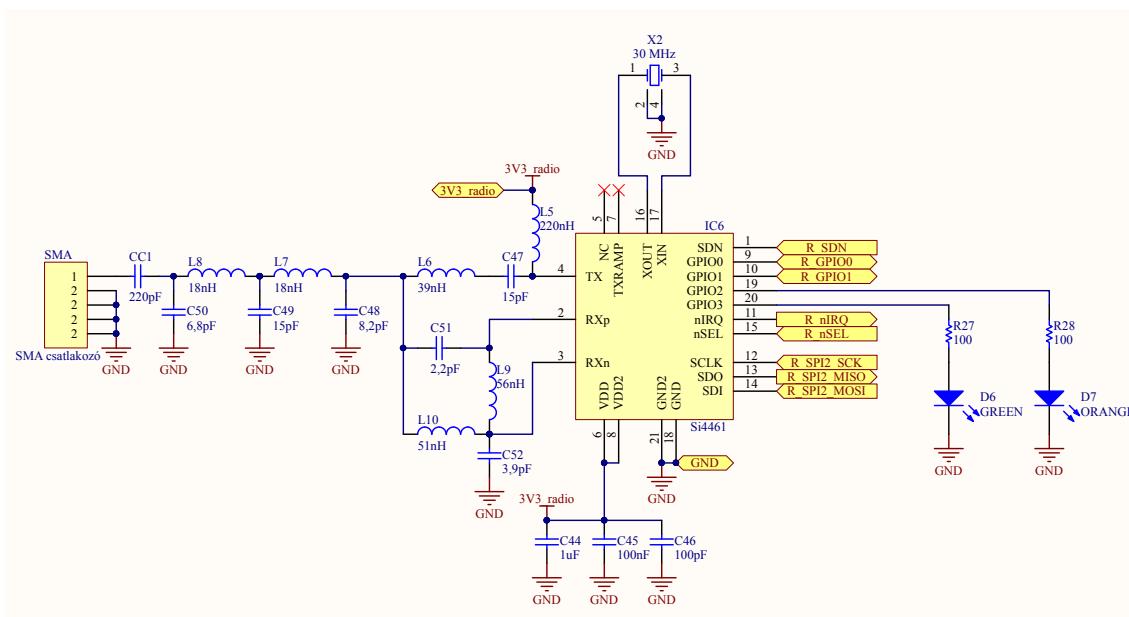
### 3.2.4. ISM sávú rádiós adó-vevő egység

A vezérlőegység és a laptop közti ISM sávú rádiós kapcsolat kialakításához rádiós adó-vevő áramkörre van szükség. A vezérlőegységen elhelyezett ISM sávú rádiós adó-vevő áramkör feladata, hogy a mikrokontrollertől SPI interfészen keresztül kapott telemetria- és szenzoradatokat feldolgozza, a megfelelő fejlécek hozzáadásával felépítse a rádiós adatcsomagot, majd a beállított rádiós paramétereknek megfelelően modulálja, felkeverje a kiválasztott frekvenciára és kisugározza azt. A vételi oldalon a rádiós adó-vevő áramkör a rádiófrekvenciás jeleket lekeveri középfrekvenciára, demodulálja, majd a vett digitális csomagot SPI interfészen keresztül elküldi a vevőegység mikrokontrollerének, ami UART protokollon keresztül továbbítja az adatokat a laptopon futó kliensprogramnak feldolgozásra. A vezérlő parancsok kommunikációs útja ennek a fordítottja.

Az ISM sávú rádiós kapcsolat kialakításához a korábbi munkáim során már megismert Silicon Labs Si4461-es IC-jét használtam. Az Si4461 legfontosabb tulajdonságai a következők [20]:

- Üzemi feszültség: 1,8-3,6 V.
- Frekvenciasáv: 119-1050 MHz.
- Érzékenység: -126 dBm.
- Modulációs típusok: OOK, FSK, 4FSK, GFSK, 4GFSK, GMSK.
- Adatsebesség: 100 bps - 1 Mbps.
- Adóteljesítmény: legfeljebb 16 dBm.
- Automatikus adó (TX) és vevő (RX) csomagkezelő.

Az Si4461 és a hozzá tartozó áramköri elemek kapcsolási rajza a 3.5. ábrán látható.



**3.5. ábra.** Az ISM sávú rádiós adó-vevő áramkör kapcsolási rajza

Az Si4461 regiszterei SPI protokollon keresztül érhetők el. A mikrokontroller a megfelelő parancsok kiküldésével állítja be a rádiós paramétereket, a csomagkezelő tulajdonságait, tölti fel a kimeneti TX FIFO regisztert a rádió kisugározandó adattal, vagy olvassa ki a rádiós interfészen keresztül beérkező adatokat az RX FIFO regiszterből [19]. Az SPI kommunikáció kezdete előtt az nSEL lábbal engedélyezni kell az Si4461 SPI perifériáját.

Az SDN lábbal alacsony fogyasztású módba kapcsolható az IC, azonban ilyenkor törlődnek a regiszterek értékei. Az Si4461 az nIRQ lábon alacsony feszültségszinttel jelzi valamilyen eseményt, például rádiós csomag kiküldését vagy érkezését. A GPIO lábak egyenként konfigurálhatók, a GPIO2 lábra kötött narancssárga LED a rádiós adás állapotát jelzi, a GPIO3 lábon található zöld LED pedig vételi állapotban világít. A további GPIO beállítások megtalálhatóak az Si4461 adatlapjában [20].

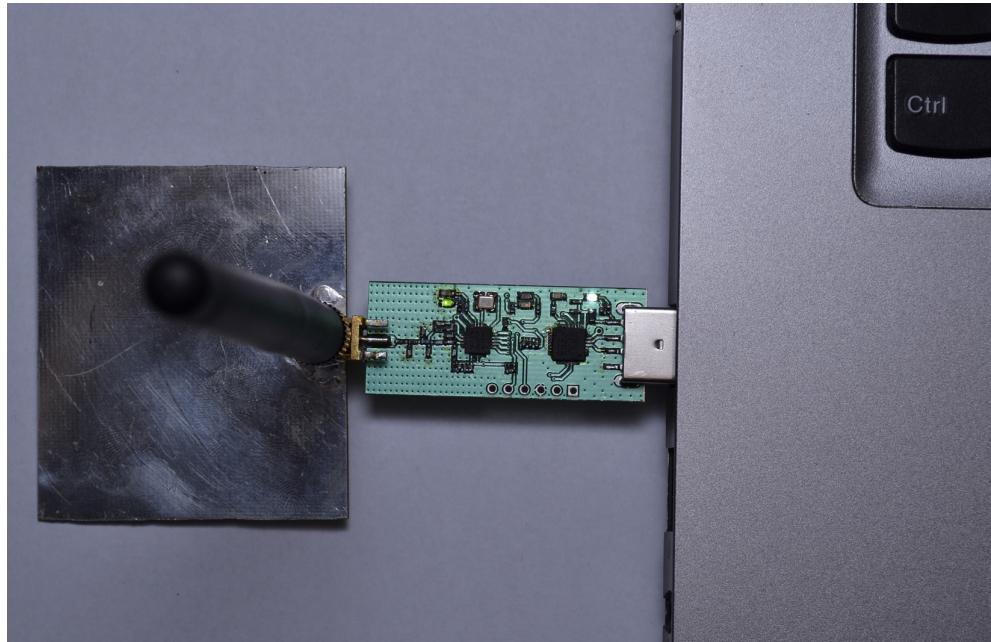
A szükséges 30 MHz-es órajel előállításához kvarckristályt használtam. A kvarckristály lábaihoz tartozó „load” kondenzátorok értékét szoftveresen lehet beállítani.

A kisugárzott rádiós jel egy 433 MHz-re illesztett aluláteresztő szűrőn megy keresztül, a vett jel ugyanezen a szűrőn, illetve egy 433 MHz-re illesztett felüláteresztő szűrőn keresztül érkezik az Si4461 bemenetére. A sorba kötött alul- és felüláteresztő szűrő úgy viselkedik, mint egy sáváteresztő szűrő. Ezenkívül az antenna elé sorasan bekötöttem egy 220 pF-os csatolókondenzátort, illetve különböző kapacitásértékű hidegítő kondenzátorokat helyeztem közvetlenül a tápf-lábak elé. Antennaként egy SMA csatlakozós 433 MHz-es monopol antennát használlok.

**USB-s ISM sávú rádiós adó-vevő egység** Az ISM sávú rádiós kapcsolat kialakításához szükség van egy másik, a laptophoz csatlakoztatható ISM sávú rádiós adó-vevő egységre. A legkézenfekvőbb megoldás egy USB-s eszköz, mely a laptop USB portján keresztül kommunikál a kliensprogrammal és a tápfeszültséget is az USB porton keresztül kapja. Mivel a kliensprogram UART protokollon keresztül kommunikál, ezért szükség van egy USB-UART átalakítóra. Erre a feladatra a Silicon Labs CP2102 IC-jét használtam [22]. Az USB-UART átalakító működéséhez telepítenünk kell a laptopra a szükséges meghajtóprogramot, a meghajtóprogram ingyenesen letölthető a Silicon Labs hivatalos honlapjáról [24]. Az USB port 5 V-os feszültségét a CP2102 belső feszültségszabályozó áramköre alakítja át a mikrokontroller és az Si4461 üzemi (3,3 V) feszültségére. Az USB-s ISM sávú rádiós adó-vevő egységen lévő mikrokontroller szintén ARM Cortex-M3 alapú, így a vezérlőegység mikrokontrollerére megírt szoftver a rendszer órajel, a GPIO portok, és a perifériák inicializálását kivéve teljes egészében átültethető [34]. Az Si4461 rádiós beállításai is azonosak, az egyetlen különbség, hogy az automatikus RX csomagkezelő 60 bajtos (telemetria- és szenzoradatok) csomagokat vár szemben a vezérlőegységen lévő rádiós IC 8 bajtos (vezérlő parancsok) RX csomagkezelőjével, míg a TX csomagkezelő 8 bajtos adatokat vár szemben a vezérlőegységen lévő 60 bajtos TX csomagkezelőjével [21]. Az USB-s ISM sávú rádiós adó-vevő egység kapcsolási rajzát és nyomtatott áramköri tervét a Függelék F.2. fejezete tartalmazza.

Az USB-s ISM sávú rádiós adó-vevő egység tesztelése során azt tapasztaltam, hogy egy csomag rádión való kiküldése után a mikrokontrolleren futó program végrehajtása meg-

áll. Mivel a hiba sem UART kommunikáció közben, sem rádiós vétel során, sem pedig a vezérlőegységen lévő ISM sávú rádióval (ugyanolyan rádiós paraméterek mellett) nem volt reprodukálható, ezért arra a következtetésre jutottam, hogy vagy a kisugárzott rádiófrekvenciás jel reflektálódik,<sup>6</sup> vagy a monopol antennán kisugárzott elektromágneses tér erővonalaiból a nyák kis mérete miatt nem a földrétegen keresztül záródnak, és ez megzavarja a mikrokontroller működését. A hiba kiküszöbölésének érdekében először az adóteljesítményt 16 dBm-ról 0 dBm-re csökkentettem, azonban ezzel párhuzamosan a kommunikáció hatótávja is csökkent. Második megoldásként egy fémlemezről forrasztottam az SMA csatlakozó földpontjához, ebben az esetben a hiba 16 dBm-es adóteljesítmény mellett sem lépett fel, így valószínűsíthető, hogy nem a reflektált hullám zavarta meg a mikrokontroller működését. Az USB-s ISM sávú rádiós adó-vevő egység a 3.6. ábrán látható.



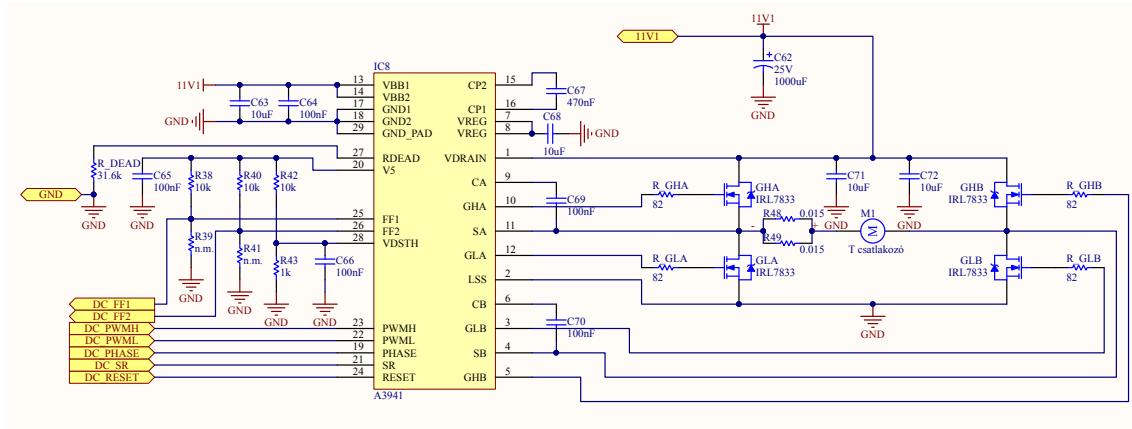
**3.6. ábra.** *USB-s ISM sávú rádiós adó-vevő egység*

### 3.2.5. DC motor-vezérlő

A mobil adatgyűjtő egység kerekeit egy kefés egyenáramú motor hajtja meg. Az előre- és a hátramenet biztosításához teljes vagy másnéven H-hidas motorvezérlést alkalmaztam, így a motoráram minden irányban folyhat. Előzetes mérési eredmények alapján üzemszerű használat közben a DC motor árama a sebbességtől és a motor nyomatékától függően 1-5 A. Az ilyen nagyáramú integrált H-hidas motorvezérlők költségesek, ezért külső H-hidas motorvezérlést terveztem. A H-híd MOSFET-jeit egy Allegro A3941 típusú gate meghajtó IC vezérli. A DC motor-vezérlő kapcsolási rajza a 3.7. ábrán látható.

---

<sup>6</sup>A vezérlőegységen lévő mikrokontroller működése még 100%-os reflexió (szakadás, antenna nélküli), esetén is zavartalan maradt.



**3.7. ábra.** A DC motor-vezérlő kapcsolási rajza

A H-híd kapcsoló MOSFET-jeinek kiválasztásánál a legfontosabb szempont, hogy a drain-source ellenállás ( $r_{ds(on)}$ ) a veszteség minimalizálása érdekében minél kisebb legyen, a maximálisan megengedhető konstans drain áram pedig nagyobb legyen a DC motor maximális áramánál. A választásom az IRL7833-as TO-220AB tokozású n-csatornás MOSFET-re esett, melynek maximális drain árama 150 A,  $r_{ds(on)}$ -ja pedig 3,8 mΩ [28]. A MOSFET-ek kiválasztása után meghatároztam a gate ellenállások ( $R_{Gxx}$ ) értékét, a MOSFET-ek minimális kapcsolási idejét, valamint a tranzisztorok veszteségét és üzemi hőmérsékletét. A gate ellenállások, a kapcsolási idő, és a tranzisztor veszteségének és hőmérsékletének meghatározásához szükséges adatlapi adatokat a 3.4. táblázat tartalmazza.

Név	Jelölés	Érték	Megjegyzés
A3941			
Gate áram	$I_{Gxx}$	150 mA	adatlapban lévő tesztek alapján
Gate-Source feszültség	$U_{GS}$	13 V	–
Felső hídág meghajtó tranzisztorának drain-source ellenállása	$R_{DS(on)UP}$	12 Ω	maximális érték 25 °C-on
Alsó hídág meghajtó tranzisztorának drain-source ellenállása	$R_{DS(on)DN}$	4 Ω	maximális érték 25 °C-on
IRL7833			
Gate kapacitás	$Q_G$	47 nC	maximális érték
Bekapcsolási késleltetési idő	$t_{d(on)}$	18 ns	–
Kikapcsolási késleltetési idő	$t_{d(off)}$	21 ns	–
Felfutási idő	$t_{RISE}$	87 ns	$\frac{Q_{gd}}{I_{Gxx}}$
Lefutási idő	$t_{FALL}$	34 ns	$\frac{Q_{gs2}}{I_{Gxx}}$
Drain-source ellenállás	$r_{ds(on)}$	4,5 mΩ	maximum érték
Hőellenállás	$R_{\theta JA}$	62 °C/W	–

**3.4. táblázat.** Az A3941 gate meghajtó IC és az IRL7833 n-csatornás MOSFET paraméterei [27] [28]

Az adatok alapján az ideális gate ellenállás értéke a következőképpen számolható:

$$R'_{Gxx} = \frac{U_{GS}}{I_{Gxx}} = \frac{13V}{150mA} = 87\Omega, \quad (3.2)$$

ahol  $R'_{Gxx}$  az  $R_{Gxx}$  gate ellenállás és az A3941 gate meghajtó MOSFET-jének drain-source ellenállásának összege,  $U_{GS}$  a gate-source feszültség,  $I_{Gxx}$  a gate áram. Az  $R_{Gxx}$  ellenállások értékét 82  $\Omega$ -nak választottam, így a felső és az alsó hídágak gate ellenállása a következő:

$$R_{GH} = R_{DS(on)UP} + R_{Gxx} = 12\Omega + 82\Omega = 94\Omega, \quad (3.3)$$

$$R_{GL} = R_{DS(on)DN} + R_{Gxx} = 4\Omega + 82\Omega = 86\Omega. \quad (3.4)$$

A gate ellenállások értékéből kiszámolható a tényleges gate áram a felső és az alsó hídágban:

$$I_{GH} = \frac{U_{GS}}{R_{GH}} = \frac{13V}{94\Omega} = 138mA, \quad (3.5)$$

$$I_{GL} = \frac{U_{GS}}{R_{GL}} = \frac{13V}{86\Omega} = 151mA. \quad (3.6)$$

A minimális kapcsolási idő kiszámításához meg kell határozni, hogy ekkora gate áram mellett mennyi ideig tart a MOSFET-ek gate kapacitásainak ( $Q_G$ ) feltöltése és kisütése.

$$t_H = \frac{Q_G}{I_{GH}} = \frac{47nC}{138mA} = 341ns, \quad (3.7)$$

$$t_L = \frac{Q_G}{I_{GL}} = \frac{47nC}{151mA} = 311ns, \quad (3.8)$$

azaz a minimális kapcsolási időt (vagyis a H-híd maximális kapcsolási frekvenciáját) a felső hídág tranzisztorainak kapcsolási ideje határozza meg. A 3.7. ábrán látható  $Q_{GHA}$  és  $Q_{GHB}$  tranzisztorok ki- és bekapsolási ideje a következő:

$$t_{H(on)} = t_{d(on)} + t_{RISE} + t_H = 18ns + 87ns + 341ns = 446ns, \quad (3.9)$$

$$t_{H(off)} = t_{d(off)} + t_{FALL} + t_H = 21ns + 34ns + 341ns = 396ns. \quad (3.10)$$

Az A3941 RDEAD lábára kötött ellenállás értékével beállítható a kapcsolási holtidő. A kapcsolási holtidő az IC által a két hídág kapcsolása közötti késleltetési idő. A kapcsolási holtidő értékének megfelelő megválasztásával elkerülhető, hogy az alsó és a felső hídágak tranzisztorai egyszerre legyenek nyitva, ezzel rövidrezárva az akkumuláltot. Az  $R_{DEAD}$  ellenállás értékét 31,6  $k\Omega$ -ra választottam, ezzel a kapcsolási holtidő nagysága a következő:

$$t_{DEAD} = 50 + \frac{7200}{1,2 + \frac{200}{R_{DEAD}}} = 1\mu s, \quad (3.11)$$

ahol  $t_{DEAD}$  a kapcsolási holtidő ns-ban,  $R_{DEAD}$  az ellenállás értéke  $\text{k}\Omega$ -ban.

A tranzisztorok maximális kapcsolási frekvenciája tehát:

$$f_{SW} = \frac{1}{t_{DEAD} + t_{H(on)}} = \frac{1}{446\text{ ns} + 1000\text{ ns}} = 691,5\text{ kHz}, \quad (3.12)$$

**A tranzisztor vesztesége** A tranzisztor vesztesége a vezetési és a kapcsolási veszteségekből adódik össze. A vezetési és a kapcsolási veszteség a következő képletekkel határozható meg [13]:

$$P_{vezetési} = I^2 \cdot r_{ds(on)}, \quad (3.13)$$

$$P_{kapcsolási} = \frac{1}{2} \cdot V_{táp} \cdot I_d \cdot f_{SW} \cdot (t_{RISE} + t_{FALL}), \quad (3.14)$$

ahol  $P_{vezetési}$  a vezetési veszteség W-ban,  $I_d$  a tranzisztoron átfolyó áram A-ben,  $r_{ds(on)}$  a tranzisztor drain-source ellenállása  $\Omega$ -ban,  $P_{kapcsolási}$  a kapcsolási veszteség W-ban,  $V_{táp}$  a tápfeszültség V-ban,  $f_{SW}$  a kapcsolási frekvencia Hz-ben,  $t_{RISE}$  a tranzisztor felfutási,  $t_{FALL}$  a tranzisztor lefutási ideje másodpercben.

Az értékek behelyettesítésével és a tranzisztor kapcsolási frekvenciájának 40 kHz-re való megválasztásával, 12,6 V-os akkumulátor feszültség és 5 A-es motoráram mellett egy tranzisztor összvesztesége a következő:

$$P_{1tr} = P_{vezetési} + P_{kapcsolási} = 0,113\text{ W} + 0,152\text{ W} = 0,265\text{ W}. \quad (3.15)$$

A tranzisztor hőmérséklete 25 °C-os szobahőmérsékleten ekkora veszteség mellett:

$$T_{1tr} = T_A + (P_{1tr} \cdot R_{θJA}) = 25\text{ °C} + (0,265\text{ W} \cdot 62\text{ °C/W}) = 41,4\text{ °C}. \quad (3.16)$$

Az 41,4 °C bőven a megengedett 175 °C alatt van, ezért nem szükséges hűtőborda használata.

**A H-híd vezérlése** Az A3941 gate meghajtó IC a PWML, PWMH, PHASE és SR lábaival vezérelhető. Az általam használt négy-negyedes vezérlési mód a 3.8. ábrán látható.



Inputs		Outputs		Inputs		Outputs	
		Phase A    B				Phase A    B	
PWMH	1	GHx	H	PWMH	1	GHx	L
PWML	1	GLx	L	PWML	1	GLx	H
PHASE	1			PHASE	0		
SR	1			SR	1		

**3.8. ábra.** A DC motor négy-negyedes vezérlési módja [27]

A MOSFET-eket a PHASE lábra kötött 20 kHz-es PWM jelrel vezérlek.<sup>7</sup> Amikor a PHASE lab logikai 1-ben van a gate meghajtó IC kikapcsolja a GHB és GLA tranzisztorokat, valamint bekapcsolja a GHA és a GLB tranzisztorokat, ekkor a motoráram a 3.8. ábra szerinti balról-jobbra irányban fog folyni. Amikor a PWM jel vált és a PHASE lab logikai 0-ban van, akkor a GHA és GLB tranzisztorok kikapcsolt, míg a GHB és GLA tranzisztorok bekapcsolt állapotban vannak és a DC motoron átfolyó áram ellentétes irányú lesz. 50%-os kitöltési tényezőjű PWM jelnél a két hídág azonos ideig van nyitva, ezért a DC motoron átfolyó áram előjeles összege 0 A. Ha a vezérlő PWM jel kitöltési tényezője 50%-nál kisebb vagy nagyobb, akkor a DC motoron átfolyó áram előjeles összege nem 0 A lesz, ezért a motor nyomatékot fog kifejni és elkezd forogni az egyik vagy a másik irányba. A további vezérlési módok megtalálhatóak az A3941 adatlapjában [27].

Az A3941 rendelkezik túlmelegedés és túláram elleni védelemmel, valamint letiltja a H-híd vezérlését, ha a bemeneti feszültség túl alacsony. A hiba típusára az FF1 és az FF2 labak logikai értéke alapján következtethetünk. Az FF1 és FF2 open drain kimenetű, ezért felhúzó ellenállások használata szükséges.

A CP1, CP2, VREG, CA, SA, CB, és SB labakon lévő kapacitások értékeinek kiszámításához az adatlapban szereplő egyenleteket használtam [27]. Az IC előre egy 1000  $\mu$ F-os, a táp bemenetekhez egy 10  $\mu$ F-os, valamint közvetlenül a két hídág mellé egy-egy 10  $\mu$ F-os puffer kondenzátort helyeztem. A táp-labakat 100 nF-os kondenzátorokkal hidegítettem.

<sup>7</sup>A vezérlés módja miatt ez 40 kHz-es tranzisztor kapcsolási frekvenciát jelent.

### 3.2.6. Szervomotor-vezérlő

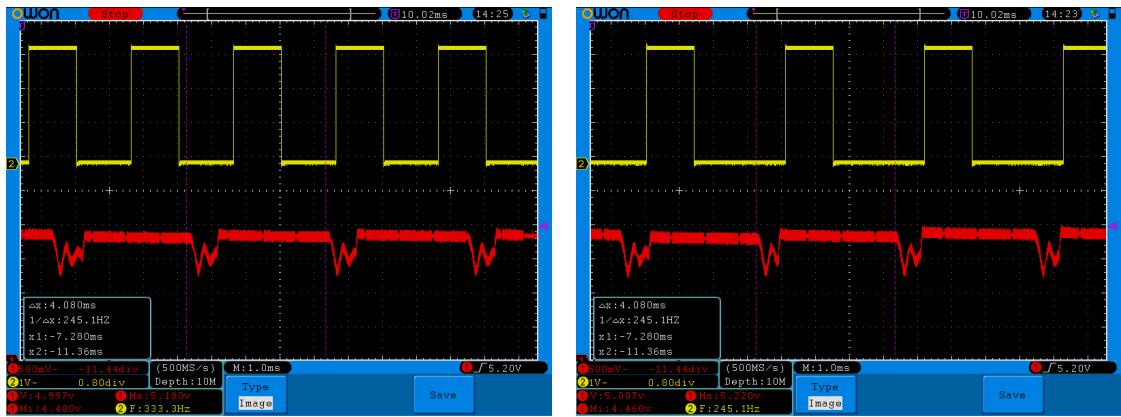
A mobil adatgyűjtő egység kormányozhatóságáért egy szabványos méretű, távirányítós autókhoz kapható, egy dobozba integrált szervomotor és szervomotor-vezérlő felelős. Működés szempontjából két típusú, analóg, illetve digitális szervomotor létezik. Vezérelhetőség szempontjából a két típus azonos, a különbség a szervomotor vezérelhetőségének gyorsaságában van. Az analóg működésű szervomotorok vezérlőjelének frekvenciája 30-50 Hz (tehát a pozíciófrissítés ideje 20-33 ms) típustól és gyártótól függően, míg a digitális szervomotorok vezérlőjelének frekvenciája – szintén gyártó és típus függő – akár 400 Hz is lehet, tehát a pozíciófrissítés ideje egészen 2,5 ms-ig csökkenthető, ami egy nagyságrenddel kevesebb, mint az analóg szervóknál. Mivel a szervomotor vezérelhetőségének sebessége nem kritikus a feladat szempontjából, így minden két típus megfelelő választás lett volna, azonban a későbbi esetleges autonóm felhasználás szempontjából a digitális szervomotor használata a szabályozás gyorsaságának növelése érdekében előnyösebb. A két típus közti árkülönbség sem számottevő, így a választásom a Blue Bird BMS-621DMG+HS típusú digitális szervomotorra esett.

A szervomotor három érintkezős (tápf, föld, vezérlőjel) tüskesoron keresztül csatlakoztatható a vezérlőegységhez. Üzemel feszültsége 4,8-6 V, a nyomaték ennek megfelelően 6,4-7,2 kgcm,<sup>8</sup> forgatási sebessége pedig 0,13-0,10  $\frac{\text{sec}}{60^\circ}$ . A szervomotor egyetlen PWM jelkel vezérelhető, melyet a szervomotor digitális feldolgozóegysége digitalizál és az impulzus szélességének függvényében elforgatja a motort. A vezérlő PWM jel frekvenciájának 333,33 Hz-et választottam, ami 3 ms-os periódusidőnek felel meg. A 3.9. ábrán megfigyelhető, hogy bár a szervomotor digitális feldolgozó egysége képes akár a 333,33 Hz-es PWM jelet is feldolgozni, a szervomotor pozíciófrissítése csak 245 Hz-cel történik. Az egyenes kormányálláshoz általában 1,5 ms szélességű impulzusjel tartozik, ami 50%-os PWM kitöltési tényezőt jelent, azonban ez gyártótól és típustól, valamint az autó mechanikájától függően kicsit eltérhet. Mérési és tesztelési eredmények alapján az egyenes kormányálláshoz 1,4 ms szélességű impulzusjel tartozik, ami ebben az esetben 46,7%-os kitöltési tényezőjű PWM jelnek felel meg. Mérési eredmények alapján az autó maximális kormánykitérése 13° mind jobbra, mind balra. A maximális balra kormányzáshoz 1,75 ms szélességű, míg a maximális jobbra kormányzáshoz 1,05 ms szélességű impulzusjel tartozik, így a kormány egy fokkal való elmozdításához 2,7 ms-mal kell megváltoztatni a vezérlő impulzus hosszát.

A szervomotort a 4,8 V-os kapcsolóüzemű tápegység látja el energiával. A fejlesztés kezdeti fázisában a tápfeszültséget 5,2 V-ra állítottam, hogy a szervomotor nyomatéka nagyobb, míg forgatási sebessége kisebb legyen, azonban a tesztelések során azt tapasztaltam, hogy a szervomotor forgatása után a szervomotor által a tápegység kimeneti kondenzátorába visszatáplált energia átlagosan 0,2 V-tal, csúcsértékben pedig akár 0,3-0,4 V-tal is emeli a kapcsolóüzemű tápegység kimeneti feszültségét (3.9. ábra), amely így kívül esik az ugyanerről a tápegységről működő Raspberry Pi üzemi feszültségtartományán (4,75-5,25 V). A hiba kiküszöbölése érdekében a kapcsolóüzemű tápegység kimeneti feszültségét 4,8 V-ra csökkentettem, így a szervomotorból visszatáplált energia által megnövekedett ki-

<sup>8</sup>Érdekkesség: 6,4 kgcm = 0,63 Nm = 0,000845 lőerő·sec

meneti feszültség csúcsértékben sem éri el a Raspberry Pi maximális megengedett üzemi feszültségét. A 3.9. ábrán továbbá látható, hogy a szervomotor pozíciófrissítésekor a motor nagy indítóárama miatt a tápfeszültség körülbelül 1 ms-ig 4,75 V alá csökken, azonban ez nem okoz gondot a Raspberry Pi működésében, mivel a bemenetén található  $220 \mu\text{F}$ -os pufferkondenzátor elegendő energiát tárol, hogy kisimítsa a tápfeszültség ingadozását.



**3.9. ábra.** A szervomotor hatása a kapcsolóüzemű tápegység kimeneti feszültségére

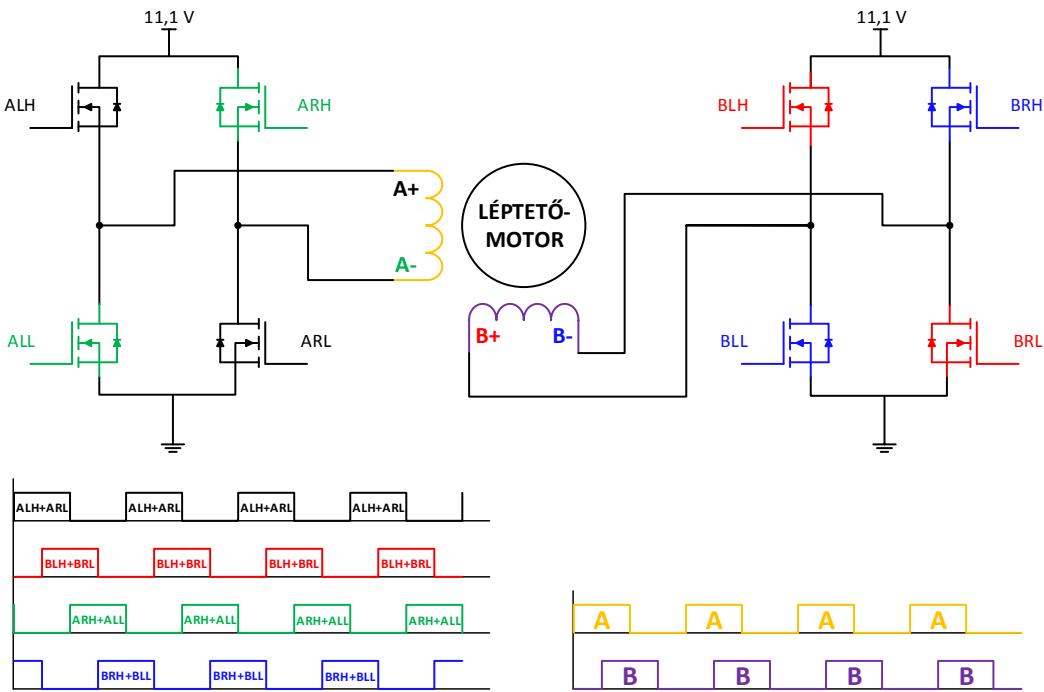
### 3.2.7. Léptetőmotor-vezérlő

A mobil adatgyűjtő egység vezérlő személyzete az élő kamerakép alapján tájékozódhat a felderítendő területről, ezért elengedhetetlen, hogy a kamera  $360^\circ$ -ban körbe tudja pásztázni a mobil adatgyűjtő egység környezetét. A kamera körbefordulása ugyan megoldható lenne, ha fixen rögzítenénk a nyákra, majd magával a mobil adatgyűjtő egységgel tennénk meg egy  $360^\circ$ -os kört, azonban ez szűk területen nem kivitelezhető egyetlen manőverrel, így ezt az ötletet elvettem. Ennél sokkal jobb megoldás, ha magát a kamerát tudjuk elforgatni a mobil adatgyűjtő egységhez képest.

A kamera forgatása megoldható egy kis teljesítményű szervo- vagy léptetőmotorral. A léptetőmotor használata előnyösebb ebben az esetben, mivel nem szükséges pozíció-visszacsatolás a pontos elfordulási szög ismeretéhez a szervomotorral ellentétben. A léptetőmotor mechanikai tulajdonságaiból (fázistekercsek száma, állandó mágnes és lágyvas kialakítása) adódóan a vezérlőjel ismeretében pontosan meg tudjuk mondani az aktuális szögelfordulást. A léptetőmotor kiválasztásánál a legfőbb szempontok az ár, a motor kis mérete, és a minél kisebb lépésszög voltak, ezért a választásom a 42HS48-1504A típusú bipoláris<sup>9</sup> hibrid léptetőmotorra esett, melynek paramétereit a Függelék F.3. fejezete tartalmazza.

A léptetőmotort a Texas Instruments DRV8825-ös bipoláris léptetőmotor-vezérlő IC-je vezérli. A DRV8825 kettő, az IC-be integrált H-híd megfelelő vezérlésével tudja a léptetőmotor fázistekercseit gerjeszteni [15]. A 3.10. ábrán a két H-híd kapcsoló tranzisztorainak állapota és a léptetőmotor fázistekercseinek gerjesztése közti kapcsolat látható teljes lépéses üzemmód esetén.

<sup>9</sup>Bipoláris léptetőmotor esetén csak a fázistekercsek végpontjai, unipoláris léptetőmotoroknál a fázistekercsek középpontjai is ki vannak vezetve a motorból.



**3.10. ábra.** Bipoláris léptetőmotor kapcsoló tranzisztorainak állapota, és fázisainak gerjesztése teljes lépések üzemmódban

A DRV8825 a MODE0, MODE1, MODE2 bemeneti lábaira adott megfelelő logikai értékektől függően hatfélé lépésmódban (teljes, fél,  $\frac{1}{4}$ ,  $\frac{1}{8}$ ,  $\frac{1}{16}$ ,  $\frac{1}{32}$ ) képes üzemelni. A léptetőmotor lépésszöge  $1.8^\circ$ , vagyis az  $\frac{1}{32}$ -es mikrolépések üzemmódban egy lépés  $\frac{1.8^\circ}{32} = 0.05625^\circ$ -os szögelfordulást jelent. A forgás iránya a DIR lábon keresztül állítható be. Vezérelni a STEP lábon keresztül lehet, minden egyes impulzus felfutó élére a léptetőmotor egy lépést tesz meg. A tényleges szögelfordulás a lépésmódtól és a forgási iránytól függ. Az egymást követő felfutó élek sűrűsége, vagyis a léptetési frekvencia maximuma 250 kHz. Az nENBL lábbal lehet engedélyezni és letiltani a H-hidak vezérlését, az nSLEEP lábbal alacsony fogyasztású üzemmódba lehet állítani az IC-t. A DRV8825 rendelkezik túláram és túlmelegedés védelemmel is, ha hiba történt, akkor azt az nFAULT lábon kiadott alacsony jelszinttel jelzi [15].

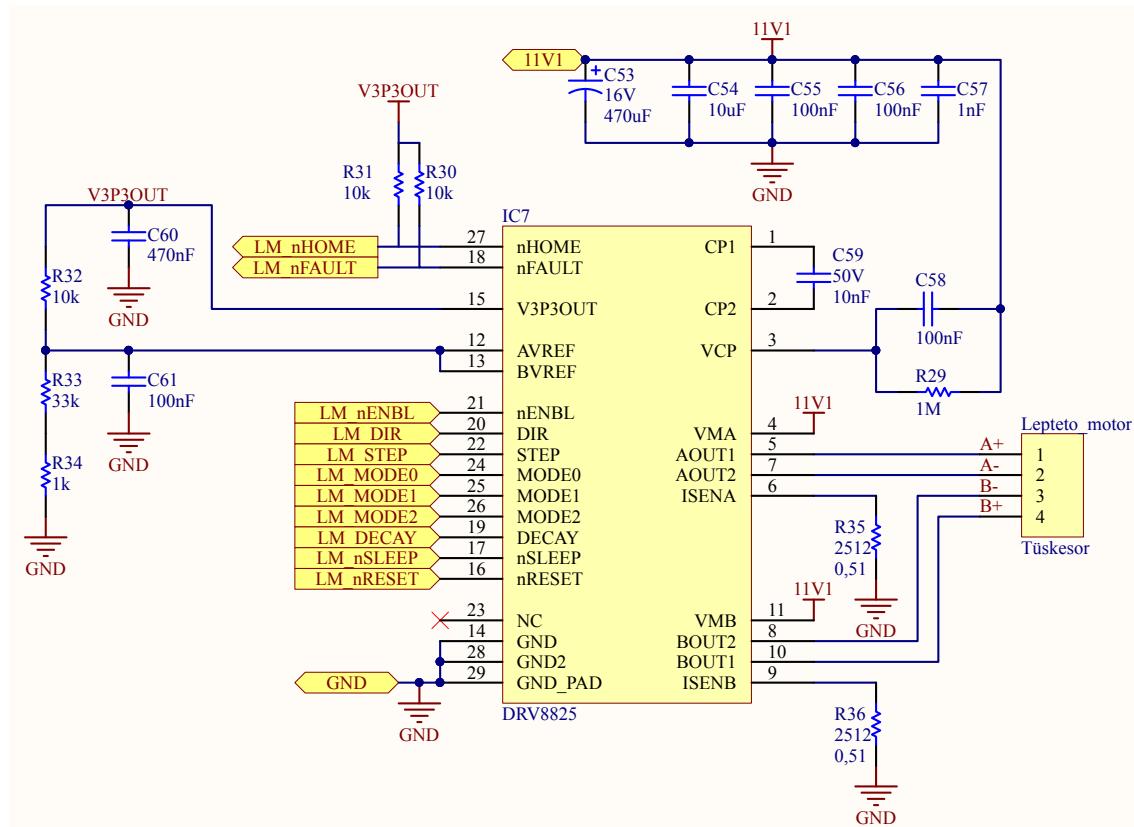
A két H-híd felső ágait (VMA, VMB lábak) az akkumulátorhoz kötöttem, így a léptetőmotort közvetlenül az akkumulátor látja el energiával. A léptetőmotor fázistekercseinek maximális áramát az AVREF és a BVREF lábakra eső feszültség, valamint az ISENA és az ISENB lábakkal sorba kötött sönt ellenállások értéke határozza meg a következő képlet alapján [15]:

$$I_{xCHOP} = \frac{V_xVREF}{5 \cdot RISEN_x}, \quad (3.17)$$

ahol  $I_{xCHOP}$  a fázis árama A-ben,  $V_xVREF$  a referencia feszültség V-ban,  $RISEN_x$  a sönt-ellenállás értéke  $\Omega$ -ban.

A léptetőmotor maximális árama fázisonként 1,5 A lehet, azonban a felhasználás nem

igényli a maximális nyomaték kifejtését a léptetőmotortól, ezért a maximális fázisáramot a fogyasztás csökkentése érdekében 1 A-ben határoztam meg. A fenti képlet alapján az AVREF és a BVREF lábakra eső feszültség értékét 2,55 V-ra állítottam az IC stabilizált, 3,3 V-os kimeneti feszültségének leosztásával, így a söntellenállások értéke  $0,51 \Omega$ -ra<sup>10</sup> adódott. Ha a fázisáram meghaladja az 1 A-t, a DRV8825 a DECAY láb logikai értékétől függően vagy ellenütembe kapcsolja a tranzisztorokat (fast decay), vagyis az áram visszafolyik az akkumulátorba, vagy az alsó két tranzisztor kinyitásával az áram a fázistekercsen keresztül cirkulál, ahol hővé alakul a tekercs soros ellenállásán (slow decay). A léptetőmotor-vezérlő kapcsolási rajza a 3.11. ábrán látható.



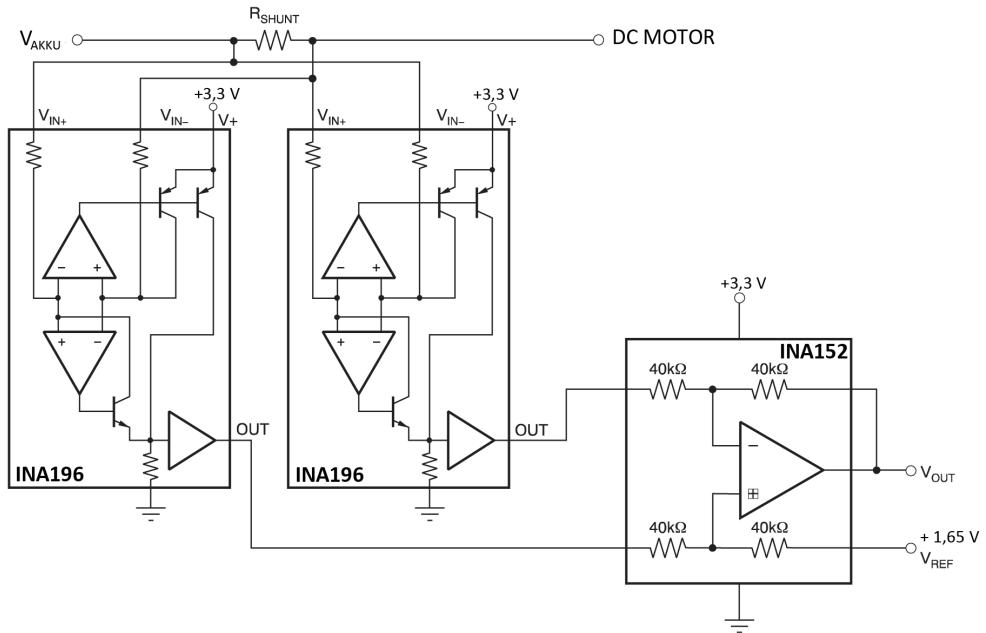
**3.11. ábra.** A léptetőmotor-vezérlő kapcsolási rajza

### 3.2.8. Szenzorok

Ebben az alfejezetben a vezérlőegységre fixen telepített szenzorok működését mutatom be. A fix telepítésű szenzorok mellett további digitális (SPI, IIC, UART, stb. interfésszel keresztül) és analóg (a szenzor kimeneti feszültségének mérése ADC-vel) szenzorokat is csatlakoztathattunk a mikrokontroller vagy akár a Raspberry Pi tüskesorra kivezetett, nem használt lábaihoz. Diagnosztikai célokból az összes szenzor értéke elküldésre kerül a laptopnak, a mérési eredményeket a kliensprogramban értékelem ki.

<sup>10</sup>A söntellenálláson eldisszipálódó teljesítmény  $P = I^2 \cdot R_s = 0,51 W$ , ezt az alkatrész méretezésekor figyelembe kell venni.

**Kétirányú áramérősség mérő** Az egyenáramú motor nyomatéka egyenesen arányos a motor áramával, ezért lejtőn felfelé, vagy kevésbé egyenletes terepen, vagy a mobil adatgyűjtő egység elakadása során a DC motornak nagyobb nyomatékot kell kifejtenie a haladáshoz, ami a motoráram növekedéséhez vezet. A nagy motoráram miatti túlzott fogyasztás és a motor károsodásának elkerülése érdekében áramkorlátozást alkalmaztam. A DC motor árama semmilyen körülmények között nem haladhatja meg a  $\pm 7$  A-t, ellenkező esetben a szoftver fokozatosan leállítja a DC motor meghajtását. A DC motor árama minden irányba folyhat, ezért a motoráram méréséhez két darab INA196-os árammérő és egy INA152-es differenciálerősítő IC-t használtam. A kétirányú áramérősség mérő áramkör blokkábrája a 3.12. ábrán látható.



**3.12. ábra.** A kétirányú áramérősség mérő áramkör blokkábrája [16]

Az  $R_{SHUNT}$  ellenálláson a DC motoron átfolyó áram irányának függvényében pozitív vagy negatív feszültség esik, ezért az ellenállás végpontjai a két INA196-os  $V_{IN+}$  és  $V_{IN-}$  bemenetére ellentétesen vannak kötve. Az akkumulátorból a DC motor felé folyó áram esetén (pozitív irány) csak a 3.12. ábrán látható baloldali INA196-os árammérő IC lesz aktív, a jobboldali kimenete 0 V lesz. Ellenkező áramirány esetén a jobboldali IC lesz aktív, míg a baloldali kimeneti feszültsége 0 V lesz. Az INA196 erősítése 20, vagyis az INA196 bekötéséhez képest pozitív áramirány esetén a sönellenálláson eső feszültség húszszorosa jelenik meg az  $OUT$  kimeneten. Mivel a DC motor árama nem egyenletes, ezért egy 1,6 kHz töréspontú<sup>11</sup> aluláteresztő RC szűrőt terveztem az INA196-osok bemeneteihez [16]. Az  $R_{SHUNT}$  ellenállás értékét 7,5 mΩ-ra választottam, ekkor az ADC 12 bites felbontása miatt az árammérés felbontása 5,4 mA, a maximális mérhető áram pedig  $\pm 11$  A, ami az ADC bemenetén 0-3,3 V feszültsékgörbejelöket jelent. A 7,5 mΩ-os sönellenállást két párhuzamosan kapcsolt 15 mΩ-os ellenállással valósítottam meg, ekkor az egy ellenállásra eső teljesítmény a felére

<sup>11</sup> A törésponti frekvenciát egy dekáddal a 20 kHz-es vezérlő PWM jel alá állítottam.

csökken, és az ellenállás kevésbé fog melegedni.

A két INA196 kimenetét közvetlenül rákötethetnénk a mikrokontroller ADC-jének két küllőnböző csatornájára és az analóg-digitális átalakítás után egyértelműen meghatározható lenne a DC motor áramának nagysága és iránya, azonban erőforrás takarékkossági szempontból az INA196-ok kimeneteit egy INA152-es egyszeres erősítésű differenciálerősítő be-menetére kötöttem. Az INA152-es referencia feszültségét féltápra, 1,65 V-ra állítottam,<sup>12</sup> ekkor a differenciálerősítő kimenete:  $V_{OUT} = V_{REF} + (V_+ - V_-)$  [12]. Az INA152  $V_{OUT}$  kimenetét a mikrokontroller analóg-digitális átalakítójára kötöttem, a DC motor áramának erőssége és iránya a  $V_{OUT}$  feszültségből egyértelműen meghatározható.

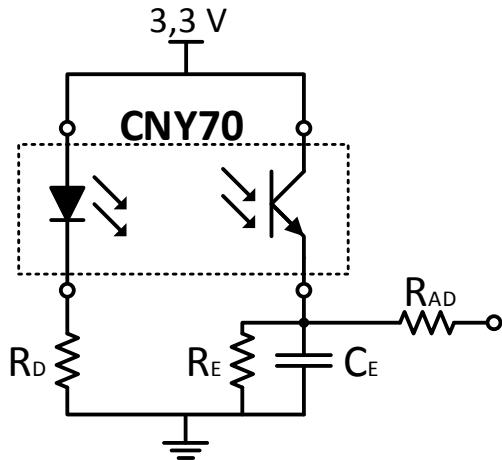
Az árammérő áramkör kalibrálását multiméter segítségével végeztem. A multimétert sorosan bekötöttem a DC motor és a söntellenállás közé, így az INA196-tal és a multiméterrel mért DC motoron átfolyó áram értéke és iránya meg kell, hogy egyezzen. A mérés során azt tapasztaltam, hogy a negatív áramirányt mérő INA196-os 0,4 A-el többet mér. A hiba lineáris, ezért a korrigálás a kliensprogramban elvégezhető, az áram kiszámítása után 0,4 A-t levonok az eredményből.

**Sebességmérő** A mobil adatgyűjtő egység sebességlátozásához szükség van a pontos haladási sebesség ismeretére. A sebesség és a haladási irány meghatározása általában a motor tengelyére kapcsolt enkóderekkel történik, azonban a távirányítós autó alvázának kialakítása miatt sem a DC motor tengelyére, sem a kardántengelyre, sem a kerekek tengelyére nem lehet enkódert helyezni, ezért más megoldás szükséges.

A fekete színű kardántengelyre egy vékony fehér csíkot festettem, mozgás közben ezt a fehér csíkot detektálom egy 3 mm-el a kardántengely fölé helyezett CNY70-es infravörös érzékelővel. A kardántengely fordulata megegyezik a fehér csík detektálásának frekvenciájával. A mobil adatgyűjtő egység sebességének meghatározásához lemértem, hogy 5 m megtétele alatt hányszor fordul körbe a kardántengely. A kapott hányados alapján kiszámolható a sebesség. A haladási irányt egy darab szenzorral nem lehet megállapítani, azonban a vezérlésből vagy a motor áramának méréséből egyértelműen eldönthető. A 3.13. ábrán a CNY70-es kapcsolási rajza látható [31].

---

<sup>12</sup>A DC motor szimmetrikus, vagyis ugyanakkora forgatónyomaték kifejtéséhez ugyanakkora áram szükséges minden irányban, ezért az árammérésnek is szimmetrikusnak kell lennie.



**3.13. ábra.** A sebességmérő kapcsolási rajza

Az  $R_D$  ellenállással beállítható az infravörös LED árama, az  $R_E$  ellenállással pedig a fototranzisztor munkaponti emitterárama. Az  $R_{AD}$  ellenállás áramkorlátozó szerepet tölt be. A fototranzisztor a visszavert infravörös fény erősségtől függően nyit vagy zár. Mivel a világos felületről több fény reflektálódik, ezért az emitter ellenálláson eső feszültségből egyértelműen eldönthető, hogy a szenzor a kardántengely fekete vagy fehér része fölött van. Mérési eredmények alapján a fekete rész fölött a feszültség 0,1 V, míg a fehér csík fölött 2,5 V.

**GPS modul** A mobil adatgyűjtő egység helymeghatározásához GPS modult alkalmaztam. A fejlesztés során két különböző típust teszteltem le, egy méretben kisebb integrált antennás modult, és egy nagyobb méretű külső antennás modult. Az integrált antennás GPS modul tesztelése során azt tapasztaltam, hogy a GPS csak napnyugta után képes venni a GPS műholdak jelét. Erre két magyarázat is lehetséges, az egyik, hogy az antennára rásütő nap megnöveli az antenna zajhőmérsékletét, ezért romlik a vétel érzékenysége – ennek nemileg ellentmond, hogy nappal felhős körülmények között és árnyékban sem volt vétel –, a másik lehetséges magyarázat, hogy a napsütés hatására az ionoszféra rétegei megváltoznak, ezáltal megnő a szakaszcsillapítás. Napnyugta után a helymeghatározás 2-5 m-es hibával működött. A külső antennás modulnál nem tapasztaltam hasonló jelenséget, a helymeghatározás napszaktól függetlenül működött, ezért a külső antennás TYCO A1029-B típusú GPS modul használata mellett döntöttem. Az égboltra való tisztáralátás során a pontatlanság szintén 2-5 m volt. A GPS helymeghatározásának pontosságát a Google Maps-en ellenőriztem.

A TYCO A1029-B GPS modul UART-on keresztül konfigurálható, rendelkezik PPS kiimenettel, antenna detektálással, valamint giroszkóp és odométer bemenettel, azonban számmomra csak a helyadatok kinyerése volt lényeges, ezért a minimális bekötést alkalmaztam, mely a következő [37]:

- A vezérlőegységhoz a GPS modul három (táp, föld, mikrokontroller UART3 RX lába összekötve a GPS modul UART0 TX lábával) érintkezőt tüskesoron keresztül csatla-

kozatható. A csatlakozó elé, a táp és a föld közé egy 100 nF-os hidegitő kondenzátort helyeztem.

- Az aktív antenna táp bemenetét (VANT) egy soros áramkorlátozó (max. 50 mA) ellenálláson keresztül összehuzaloztam a GPS modul tápjával.
- A GPS modul ENABLE lábat a táp-lábhoz kötöttem.
- A GS (Gain Select) lábat szabadon hagytam, így az LNA erősítése a maximális 14 dB.

A GPS modul mikrokontrollere a GPS műholdak jelét feldolgozza, majd a pontos időt, a helyadatokat, a műholdak számát, stb. szabványos NMEA<sup>13</sup> üzenetekben UART protokollen keresztül továbbítja az UART0 TX interfészén keresztül. A vezérlőegység mikrokontrollere a szabványos NMEA üzenetekből kinyeri a feladat szempontjából hasznos információkat és továbbítja a laptopnak. Az NMEA üzenetek kinyerésének algoritmusát a 3.3.2. fejezet tartalmazza.

Az aktív külső antenna koax kábelét közvetlenül a GPS modul antenna-csatlakoztatási pontjához forrasztottam. A GPS modul adatlapjának javaslata alapján olyan aktív antennát érdemes használni, melynek erősítése legalább 20 dB, zajtényezője pedig maximum 1,5 dB. Az általam használt ME450G/3M/SMA(M) típusú aktív antenna paraméterei megfelelnek ezeknek az elvárásoknak.

**Hőmérő** A hőmérséklet méréséhez hőellenállást használtam, amit sorba kötöttem egy ismert értékű ellenállással (3.14. ábra). A hőmérséklet meghatározásához először meg kell határoznunk a hőellenállás pontos értékét a hőellánálláson mért feszültség alapján. A hőellenállás értéke a következő képlettel számolható:

$$3,3 V \cdot \frac{R_{th}}{R_{th} + R} = AD_{fesz}, \quad (3.18)$$

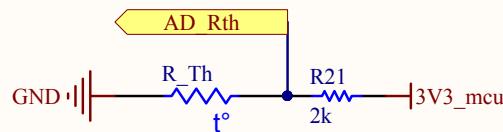
$$R_{th} = \frac{R \cdot AD_{fesz}}{3,3 V - AD_{fesz}}, \quad (3.19)$$

ahol  $R_{th}$  a hőellenállás értéke  $\Omega$ -ban,  $R$  az ismert ellennállás értéke  $\Omega$ -ban,  $AD_{fesz}$  az ADC által mért feszültség értéke V-ban.

A hőellenállás értékének ismeretében az adatlapban található értékek alapján meghatározható a hőmérséklet. A hőellenállás hibája lineáris, értéke 0 °C és 100 °C között ±5-6 °C. A hiba egy hiteles hőmérő segítségével a végeredményhez adott konstans eltérés értékével kompenzálható. A hőellenállás típusa: KTY84/130 [29].

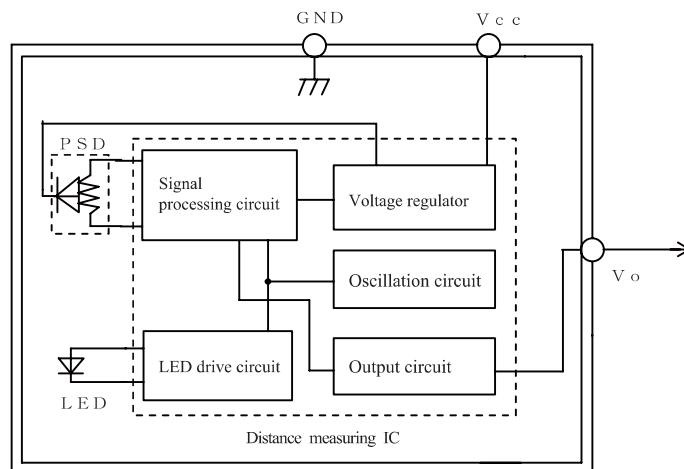
---

<sup>13</sup>NMEA: National Marine Electronics Association - Nemzeti Tengerészeti Elektronikai Szövetség által létrehozott szabványos üzenetforma [10].



**3.14. ábra.** A hőmérő szenzor kapcsolási rajza

**Távolságérzékelő** A mobil adatgyűjtő egység irányítása jelenleg távolról, vezérlő személyzet által történik. A hibás irányítás okozta ütközések és a későbbi autonóm üzemmódra való felkészítés miatt egy SHARP GP2Y0A21YK0F típusú infrás távolságérzékelőt helyeztem el a mobil adatgyűjtő egység elején. Ha a mobil adatgyűjtő egység a beállított távolságon belülre kerül az előtte lévő tárgyhoz képest, akkor a szoftver leállítja a DC motor meghajtását. A távolságérzékelő működésének blokkdiagramja a 3.15. ábrán látható.



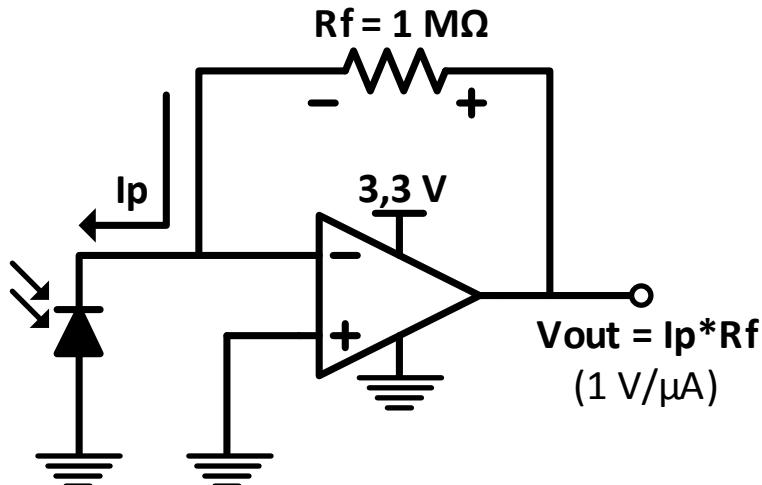
**3.15. ábra.** Az infravörös távolságérzékelő blokkdiagramja [3]

A távolságmérő szenzor körülbelül 40 ms-onként egy infravörös impulzust bocsát ki és méri a tereptárgyakról visszavert impulzus és a kibocsátott impulzus között eltelt időt, majd az időkülönbség alapján meghatározza a tárgy távolságát. A szenzor kimeneti feszültségének értéke a tárgytávolságtól függ, a távolságérzékelő 10-80 cm között üzemel. A GP2Y0A21YK0F tápfeszültsége 4,8 V, azonban a kimeneti feszültségének maximuma 3,3 V, így a kimenet közvetlenül ráköthető a mikrokontroller ADC lábára [3]. A távolságérzékelő tár és föld labai közé, közvetlenül a csatlakozó elé egy  $10 \mu\text{F}$ -os puffer és egy  $100 \text{nF}$ -os hidegítő kondenzátort kötöttem.

**Megvilágításmérő** Ahhoz, hogy a mobil adatgyűjtő egység elő kameraképet sötétben is láthassuk, háttérvilágításra van szükség. A háttérvilágításhoz hideg-fehér teljesítmény LED-eket használtam, melyek közvetlenül a kamera alá, a léptetőmotor tengelyére vannak szerelve, így együtt forgathatók a kamerával. A LED-ek automatikusan be- és kikapcsolnak, ha a környezet megvilágítása egy bizonyos küszöbérték alá csökken, vagy egy bizonyos küszöbérték fölé nő. A környezet megvilágítását egy SFH203 típusú fotodiódával és egy áram-feszültség átalakítóval mérem [11] [30].

A fotodióda előfeszítés nélküli fotovoltaikus módban üzemel.<sup>14</sup> A fotodió p-n átmennetében elnyelt fotonok hatására elektron-lyuk párok keletkeznek, és áram kezd el folyni a katódktól az anód felé. Ez a fotoáram transzimpedanciás erősítővel feszültséggé alakítható, ami a mikrokontroller analóg-digitális átalakítójával megmérhető. A mért feszültségből a fotodióda megvilágítása meghatározható.

A transzimpedanciás erősítő kapcsolási rajza a 3.16. ábrán látható.



**3.16. ábra.** A transzimpedanciás erősítő kapcsolási rajza

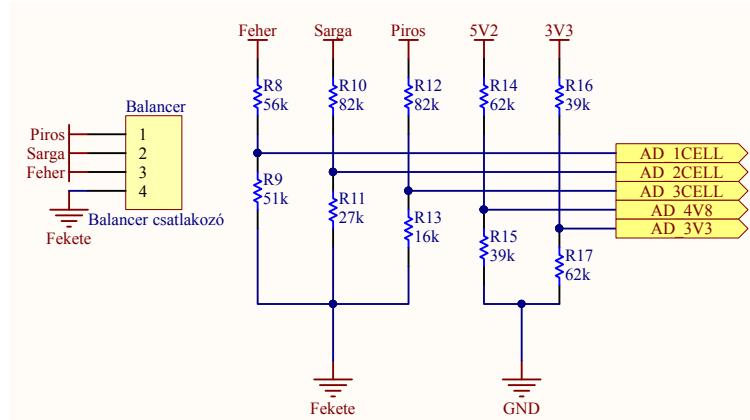
A fotodióda megvilágítása és a transzimpedanciás erősítő kimeneti feszültsége közti függvényt teszteléssel állapítottam meg. Okostelefon kamerájának segítségével különböző megvilágítás mellett 18%-os mattszürke felületen megmértem a megvilágítás értékét, majd ugyanezt megtettem a fotodiódával, miközben mértem a transzimpedanciás erősítő kimeneti feszültségét. A mérési pontokra Matlab segítségével egy egyenest illesztettem. A kamera háttérvilágítása 70 lux-os megvilágítás alatt bekapcsol és 100 lux értékű megvilágítás fölött kikapcsol (lásd: 6.4. fejezet).

**Akkumulátor cella- és tápfeszültség mérő** A lítium-polimer akkumulátorok celláinak névleges feszültsége 3,7 V, azonban teljesen feltöltve a terheletlen cellafeszültség elérheti a 4,4 V-ot, míg a lemerülés során a cellafeszültség 3 V alá is csökkenhet. Az akkumulátor élettartamának meghosszabítása érdekében a töltöttséget célszerű 10-90% között tartani. Az akkumulátor feltöltéséhez külső, mikrokontroller vezérelt töltőt alkalmaztam, amely minden cellát külön (a cellák végpontja ki vannak vezetve az akkumulátorból) tölt fel 4,2 V-ra. A túlmerítés elkerülése érdekében az akkumulátor celláinak feszültségét, valamint a 3,3 V-os és a 4,8 V-os kapcsolóüzemű tápegységek kimeneti feszültségét a vezérlőegység mikrokontrollere felügyeli.

A mikrokontroller analóg-digitális átalakítójának referencia feszültsége 3,3 V, ezért a cella- és a tápfeszültségeket 1%-os ellenállásokkal körülbelül 2 V-ra leosztottam és így kötöttem rá a mikrokontroller ADC bemeneteire (3.17. ábra). Az AD átalakítás után az

<sup>14</sup>A napelemek is ezen hatás alapján működnek.

ellenállások ismeretében kiszámolhatók a pontos feszültségértékek. A szoftver a fogyasztás csökkentésének érdekében leállítja a motorokat és letiltja a 4,8 V-os tápegységet, ha a cellafeszültségek 3,2 V alá csökkennek.



**3.17. ábra.** Az akkumulátor celláinak és a tápegységek kimeneti feszültségének mérési elve

### 3.2.9. Nyomtatott áramkör tervezése

A nyomtatott áramkör tervezésének első lépéseként ki kell választanunk az általunk használni kívánt (angolszász vagy metrikus) mértékegységrendszeret. Az Altium Designer minden mértékegységet támogatja, váltani köztük a Board Options menüben lehet. A hardvertervezés során végig metrikus mértékegységeket használtam. A két mértékegységrendszer közti átváltást a következő egyenlet írja le:

$$1 \text{ mil} = \frac{1}{1000} \text{ inch} = 0,0254 \text{ mm.} \quad (3.20)$$

Második lépésként ki kell választanunk az alkatrészekhez tartozó alkatrészlenyomatokat (footprint). Az elemekhez tartozó alkatrészlenyomatok méretei a vonatkozó alkatrészek adatlapjaiban mil-ben és mm-ben is megtalálhatók. Használhatjuk a beépített könyvtárakban lévő alkatrészlenyomatokat – azonban ezek gépi hullám vagy újraömlesztéses forrasztáshoz vannak méretezve –, vagy meg is rajzolhatjuk őket az Altium Designerben. A kézi forrasztás megkönyítése érdekében a beépített könyvtárakból használt alkatrészlenyomatok méretét megöveltem.

A következő lépés a nyomtatott áramkör panel méretének meghatározása és a gyártathatóra vonatkozó szabályok beállítása. A nyomtatott áramkör méretének és formájának kialakítása során a távirányítós autó alvázának méretéhez és az alvázon található rögzítési pontok helyzetéhez igazodtam. A fejlesztés során két nyomtatott áramkör került legyártásra. Az első nyákot az Elektronikai Technológia Tanszék gyártotta le, így a gyártathatóra szabályoknál az ETT gyártósorának technikai korlátait, valamint a költséghatékonysságot (bizonyos méretek alatt a gyártási költség feláras) vettet alapul. Az Altium Designerben beállított gyártathatóra szabályok a következők:

- Minimális szigetelővastagság: 0,254 mm.
- Minimális vezetékvastagság: 0,254 mm.
- Legkisebb furatátmérő: 0,7 mm.
- Minimális alkatrésztávolság: 0,254 mm.
- Rézfólia vastagsága:  $35 \mu\text{m}$ .
- Hőcsapdák alkalmazása csak a forrasztandó alkatrészlenyomatok körül.

A nyomtatott áramkör tervezése során a nagyáramú táp- és földvezetékek vezetőszélességeinek kialakításánál a gyakorlatban használatos  $1 \frac{\text{mm}}{\text{A}}$  ökölszabályt alkalmaztam. Az Allegro A3941 adatlapja alapján két, logikailag különböző, a nyákon egyetlen ponton összekötött földkitöltést alkalmaztam, így a DC motor nagy áramai a teljesítményföldön keresztül folyanak vissza az akkumulátorba. A másik földkitöltés pedig a kisáramú, digitális eszközökhez csatlakozik.

A nyomtatott áramkör kialakítása során zavarvédelmi szempontok miatt a nagyfrekvenciás kapcsolóüzemű és a nagyáramú alkatrészeket (kapcsolóüzemű tápegységek, léptetőmotor-vezérlő, DC motor-vezérlő) a nyák felső rétegének egyik oldalán, míg a kisáramú digitális alkatrészeket (mikrokontroller, ISM sávú rádió, GPS, UART jelszintillesztő) a nyák felső rétegének másik oldalán, a lehető legtávolabb helyeztem el egymástól. A nyáktér kialakításánál a további szabályokat is figyelembe vettetem:

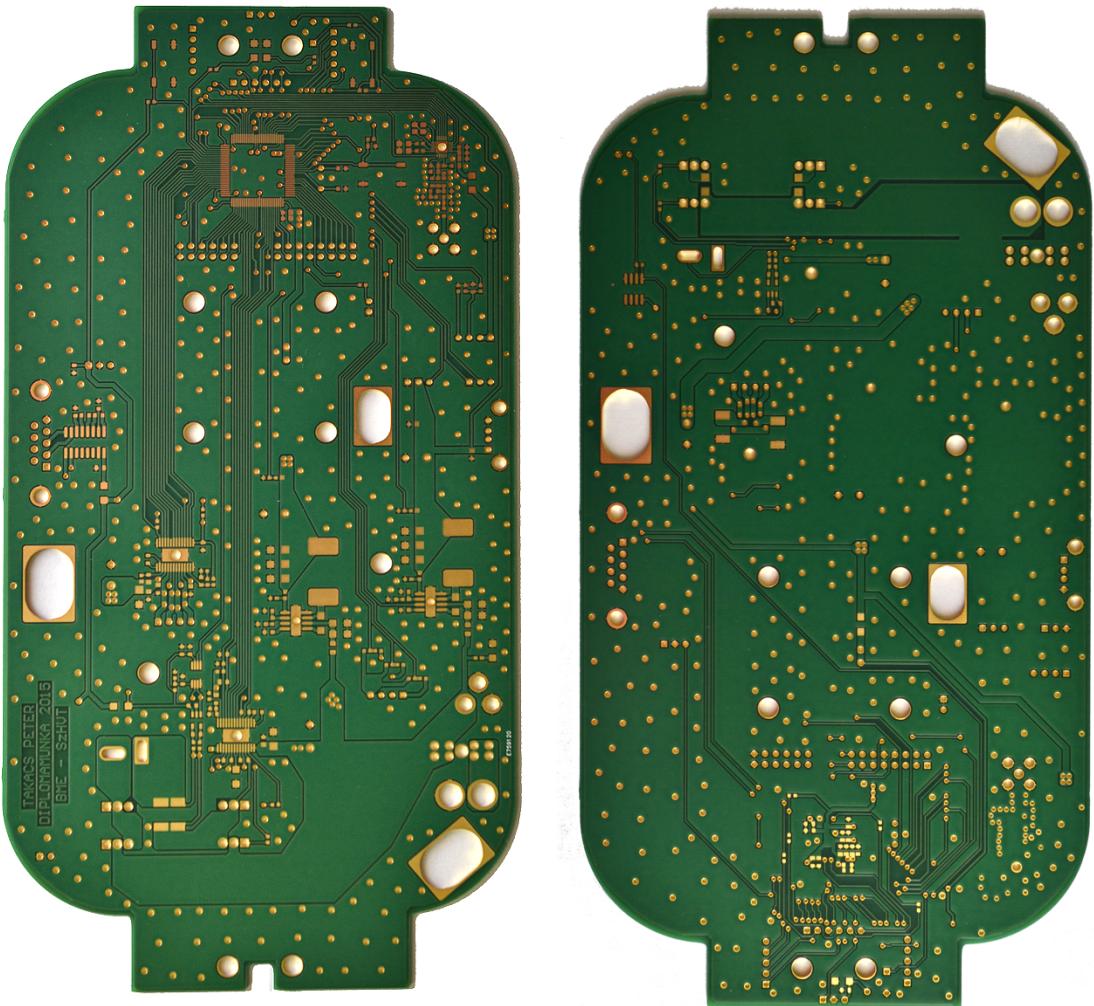
- A vezeték törésénél kerüljük a derék és hegyesszögeket!
- A furatokba és átvezetésekbe (viákba) minden merőlegesen kössük a vezetéket!
- Az alkatrészek lábaihoz (padekhez) minden derékszögben csatlakozzunk!
- Kerüljük a föld- és táphurkokat!
- Az analóg vezetékek mellett alkalmazzunk földkitöltést!
- Az ekvipotenciális föld kialakításának érdekében sűrűn viázzuk át az alsó és a felső földréteget!

A hardverépítés során először az energiaellátó rendszer került beforrasztásra és tesztelésre, majd egyesével minden további részegységet beforrasztottam és külön letesztem.

Egy feltehetőleg elektrosztatikus kisüléses (ESD) hiba következtében a mikrokontroller SWD interfésze a fejlesztés későbbi szakaszában működésképtelenné vált, így a mikrokontrollert nem lehetett újraprogramozni (az aktuális program futása zavartalanul működött). A hibás mikrokontrollert hőlégfúvó segítségével eltávolítottam a nyákról, azonban a horodzó és a gyártás rossz minősége miatt a mikrokontroller lábaihoz csatlakozó vékony rézvezetékek megsérültek, leváltak. A nyák így alkalmatlanná vált egy másik mikrokontroller beforrasztására, ezért a konzulensemmel közösen az újragyártás mellett döntöttünk. Az új nyomtatott áramköri terven kijavítottam az első prototípuson elkövetett kisebb hibákat és néhány módosítást is elvégeztem, melyek a következők:

- A minimális vezeték- és szigetelővastagságot 0,3 mm-re növelte.
- A felső réteg földkitöltése miatt a DC motor teljesítményföldje több helyen is csatlakozott a digitális alkatrészek földjéhez, ezeket az összekötéseket eltávolítottam.
- Az olvadóbiztosítékot az akkumulátor vezetékébe való soros bekötés helyett ráterveztem a nyákra.
- Az első nyákon a DC motor áramát csak egyetlen INA196-os árammérő szenzorral mértem, így az áramot csak pozitív áramirány mellett tudtam mérni.
- A 4,8 V-os tápegység engedélyező lábat raktötöttem a mikrokontrollerre.
- Az Si4461 GPIO2 és GPIO3 lába eredetileg a mikrokontrollerre volt kötve, az új nyákon indikátor LED-eket kötöttem ezekre a lábakra.
- Az USB csatlakozót vízszintesről függőlegesre cseréltem.
- A mikrokontroller tárvezetékébe sorasan bekötöttem egy  $0\Omega$ -os ellenállást, hogy a mikrokontroller fogyasztását pontosan meg lehessen mérni.
- A sebességmérő kimenetét az ADC 4-es csatornája helyett a Timer2 egyik input capture bemenetére kötöttem, így a sebességmérő szenzor által érzékelt fehér és fekete felületekhez tartozó feszültségek logikai értéke eldönthető a számláló bemenetén található komparátorral, nincs szükség analóg-digitális átalakításra.
- A szervomotor és a DC motor vezérlőjelét különböző számlálók PWM kimenetéhez kötöttem, így egymástól függetlenül beállítható a két vezérlő PWM jel frekvenciája és kitöltési tényezője.
- A megvilágításmérő, a hőmérő és a távolságmérő szenzorokat, valamint a kamera háttérvilágításához használt hideg-fehér teljesítmény LED-eket ráterveztem a nyákra, így az eredeti elgondoláshoz képest ezeket az eszközöket nem kell külön csatlakoztatni a nyákhöz.

Az új nyomtatott áramkört, melynek alsó és felső rétege a 3.18. ábrán látható a Eurocircuits Kft. gyártotta.



**3.18. ábra.** Az újragyártott nyomtatott áramkör alsó és felső rétege

### 3.3. Beágyazott szoftver

A szoftverfejlesztés első lépéseként ki kell választani egy, a feladatra alkalmas fejlesztői környezetet. Korábbi tanulmányaim során már megismerkedtem a CooCox és az Eclipse fejlesztői környezetekkel, számomra az Eclipse átláthatóbb, illetve a beágyazott könyvtáraknak és függvényeknek köszönhetően a fejlesztés egyszerűbb, így ezt választottam. Az Eclipse moduláris felépítésű fejlesztői környezet, ezért a mikrokontrollerre való fejlesztés előtt néhány bővítményt le kell tölteni és integrálni kell az Eclipse-be. A mikrokontrolleren futó beágyazott szoftvert C nyelven írtam [1]. A programkód fordításához az ARM által kiadott hivatalos GNU Tools for ARM Embedded Processors fordítót használtam. A projekt létrehozásakor meg kell adnunk a fordító elérési útját. A programkód fordítása után az Eclipse egy .hex fájlt generál, ezt a .hex fájlt az ST-LINK Utility nevű programmal töltöm a mikrokontroller flash memoriájába a 3.2.2. fejezetben említett STM32F0DISCOVERY kártyán keresztül [9].

Ebben az alfejezetben bemutatom a mikrokontrolleren futó program működését külön kitérve a különböző kommunikációs módok felépítésére és működésére, a telemetria- és szenzoradatok, illetve a vezérlő parancsok felépítésére, valamint a motorok vezérlésére.

#### 3.3.1. Mikrokontroller inicializálása

A főprogram végrehajtása előtt be kell állítani a rendszer órajelét, a megszakításvezérlőt, inicializálni kell a különböző GPIO portokat, valamint engedélyezni kell a használni kívánt perifériákat.

A mikrokontroller a rendszer órajelét belső osztó és szorzó áramkörökkel egy külső 8 MHz-es kvarckristályból állítja elő. A mikrokontroller perifériái három különböző buszra vannak felfűzve, a buszok órajelét külön-külön a mag órajelét leosztva lehet beállítani. A mag órajelének frekvenciáját a maximális 72 MHz-nek választottam, az AHB és APB2 buszok órajele szintén 72 MHz, az APB1 busz órajelét is a maximumra, 36 MHz-re állítottam. Az órajelek pontossága az MCO<sup>15</sup> kimenet megfelelő beállítása után oszcilloszkóppal ellenőrizhető [35].

Az órajel beállítása után a szoftver inicializálja a mikrokontroller megszakításvezérlőjét (NVIC). A megszakításvezérlő beállításaitól függően különböző számú megszakítás-csoportok hozhatók létre, a csoportok közti és a csoporton belüli megszakítások kiszolgálásának prioritása külön-külön konfigurálható. A megszakításvezérlőt úgy inicializáltam, hogy a maximális 16 megszakítás-csoportot lehessen létrehozni, ekkor a megszakítás-csoportokon belül az alprioritások azonosak, nem testreszabhatóak [33].

A fogyasztás minimalizálása érdekében a mikrokontroller perifériai alapértelmezetten le vannak tiltva, tehát egy periféria használatához először engedélyezni kell a periféria órajelét, majd magát a perifériát [35].

Az inicializálás utolsó lépése a mikrokontroller lábainak beállítása. Ehhez először engedélyezni kell a megfelelő lábhoz tartozó megfelelő GPIO perifériát, majd a nyolcféle mód

<sup>15</sup>A GPIO lábak maximális frekvenciája 50 MHz, ezért a rendszer, valamint az AHB, APB2 buszok 72 MHz-es órajele csak leosztva portolható ki.

közül ki kell választani az aktuális láb állapotát. A perifériákhoz tartozó alternatív funkciójú GPIO lábakat is inicializálni kell, ezt a periféria vezérlő nem teszi meg helyettünk [35].

### 3.3.2. UART kommunikáció

A mikrokontroller három különböző UART periférián keresztül is kommunikál. Az UART1-en a mikrokontroller és a laptop, az UART2-n a mikrokontroller és a Raspberry Pi, míg az UART3-on a mikrokontroller és a GPS modul közti kommunikáció zajlik. A továbbiakban kifejtem e három UART periféria beállításait, és a kommunikáció menetét.

**Mikrokontroller - laptop** A mikrokontroller és a laptop közti UART kommunikáció háromvezetékes (TX, RX, GND), full-duplex kialakítású. Az UART adatcsomag felépítése 8N1, vagyis 1 START bit, 8 adatbit, 1 STOP bit, és 0 paritás bit. A kommunikáció sebessége  $115200 \frac{\text{baud}}{\text{sec}}$ .

A mikrokontroller 60 bájtos csomagokat<sup>16</sup> küld a laptop felé, a laptopon futó kliensprogram egyesével olvassa ki a bájtokat a laptop UART perifériájából, ha értelmezhető csomag érkezett, akkor a kliensprogram feldolgozza és megjeleníti az adatokat.

A laptop 8 bájtos csomagokat<sup>17</sup> küld a mikrokontroller felé, a 8 bájtos csomagot a mikrokontroller tehermentesítése érdekében a DMA<sup>18</sup> vezérlő olvassa ki az UART1 periféria RX pufferéből és betölti egy tömbbe, majd a művelet végén a DMA vezérlő megszakítást küld a mikrokontrollernek. A csomag feldolgozása a megszakítást kezelő rutinban történik.

A mikrokontroller és laptop közti UART kommunikációt csak a fejlesztés során használtam.

**Mikrokontroller - Raspberry Pi** A mikrokontroller és a Raspberry Pi közti UART kommunikáció háromvezetékes (TX, RX, GND), full-duplex kialakítású. Az UART adatcsomag felépítése 8N1, vagyis 1 START bit, 8 adatbit, 1 STOP bit, és 0 paritás bit. A kommunikáció sebessége  $115200 \frac{\text{baud}}{\text{sec}}$ .

A mikrokontroller 60 vagy 8 bájtos csomagokat küld a Raspberry Pi felé, a Raspberry Pi a 60 bájtos csomagokat Wi-Fi-n keresztül továbbküldi a laptopnak a 8 bájtos csomagokat pedig feldolgozza.

A Raspberry Pi a Wi-Fi-n beérkező 8 bájtos csomag tartalmának függvényében vagy végrehajtja a parancsot, vagy továbbküldi a csomagot a mikrokontroller felé. A 8 bájtos csomagokat a mikrokontroller tehermentesítése érdekében a DMA vezérlő olvassa ki az UART2 periféria RX pufferéből és betölti egy tömbbe, majd a művelet végén a DMA vezérlő megszakítást küld a mikrokontrollernek. A csomag feldolgozása a megszakítást kezelő rutinban történik.

---

<sup>16</sup>Telemetria- és szenzoradatok, lásd: 3.3.5. fejezet.

<sup>17</sup>Vezérlő parancsok, lásd: 3.3.6. fejezet.

<sup>18</sup>DMA: Direct Memory Access - közvetlen memória-hozzáférés

**Mikrokontroller - GPS modul** A mikrokontroller és a GPS modul közti UART kommunikáció kétvezetékes (mikrokontroller UART3 RX lába összekötve a GPS modul UART0 TX lábával, GND) kialakítású, csak a GPS modul küld adatokat a mikrokontrollernek. A GPS modul által küldött UART adatcsomag felépítése 8N1, vagyis 1 START bit, 8 adatbit, 1 STOP bit, és 0 paritás bit. A kommunikáció sebessége 4800  $\frac{\text{baud}}{\text{sec}}$ .

A mikrokontroller bájtonként olvassa ki és dolgozza fel a GPS modul által küldött maximum 80 bájt csomagmegérű NMEA üzeneteket. Az NMEA üzenetek tartalmazzák a GPS műholdak által szolgáltatott idő, koordináta és egyéb adatokat. A GPS modul által küldött NMEA üzenetek a 3.19. ábrán láthatók.

\$GPGSS,A,3,30,05,20,15,13,17,28,,,,,,2.4,1.6,1.9\*35  
\$GPGSS,3,1,09,05,11,205,27,13,74,167,41,15,61,286,42,17,31,127,39\*72  
\$GPGSS,3,2,09,18,14,319,09,20,20,229,35,24,22,280,27,28,54,056,38\*7A  
\$GPGSS,3,3,09,30,22,084,38,,,,,,,,,\*44  
\$GPAGA,085036.000,4728.2747,N,01902.8990,E,1,07,1.6,0124.9,M,41.1,M,,\*,63  
\$GPGRMM,085036.000,A,4728.2747,N,01902.8990,E,0.1,0.0,0.040615,0.0,W,A\*15  
\$GPGSS,A,3,30,05,20,15,13,17,28,,,,,,2.4,1.6,1.9\*35  
\$GPGSS,3,1,09,05,11,205,28,13,74,167,43,15,61,286,42,17,31,127,41\*70  
\$GPGSS,3,2,09,18,14,319,00,20,20,229,37,24,22,280,28,28,54,056,37\*71  
\$GPGSS,3,3,09,30,22,084,37,,,,,,,,,\*4B  
\$GPAGA,085037.000,4728.2747,N,01902.8990,E,1,07,1.6,0124.8,M,41.1,M,,\*,63  
\$GPGRMM,085037.000,A,4728.2747,N,01902.8990,E,0.1,0.0,0.040615,0.0,W,A\*14

**3.19. ábra.** A GPS modul által küldött NMEA üzenetek

Az NMEA üzenetek \$ karakterrel kezdődnek és CR/LF (Carriage Return / Line Feed, Kocsi vissza / Soremelés) karakterekkel végeződnek. Az egy üzeneten belüli különböző adatok vesszővel vannak elválasztva, a program ezeket a vesszőket keresi meg a hasznos adatok kinyeréséhez. Az üzenetek a \$ karakter utáni 5 bájt hosszúságú betűkód<sup>19</sup> alapján különböztethetők meg. Az összes, számomra hasznos adat<sup>20</sup> megtalálható a 'GGA' típusú üzenetekben, a többi üzenetet a program eldobja [10].

### 3.3.3. ISM sávú rádiós kommunikáció

Az ISM sávú rádiós kapcsolat kialakításához a 3.2.4. fejezetben bemutatott Si4461-es rádiós IC-t használtam. A mikrokontroller SPI protokollon keresztül kommunikál az Si4461-gyel. Az SPI kommunikáció három (SCLK, MOSI, MISO) plusz egy (nSEL) vezetéken keresztül történik. A master eszköz a mikrokontroller, így ő biztosítja az órajelet (SCLK), valamint az nSEL láb logikai 0-ra állításával engedélyezi a slave eszköz (Si4461) SPI perifériáját.

Az Si4461 SPI periférijának maximális órajel frekvenciája 10 MHz, ezért a mikrokontroller SPI2 periférijának órajel frekvenciáját az APB2 busz 36 MHz-es frekvenciájának 4-gyel való leosztásával 9 MHz-re állítottam. Az Si4461 az SPI periféria engedélyezése után az SCLK első felfutó élre beolvassa a bemeneti pufferbe a MOSI vonal logikai értékét, valamint időben ezzel párhuzamosan kiadja a kimeneti pufferben lévő adatbit logikai értékének megfelelő feszültségszintet a MISO vonalra. A mikrokontrollerben tehát úgy kell beállítani az órajelet, hogy az felfutó éssel kezdődjön, ezt a CPHA és CPOL bitek 0-ra állításával érhetjük el. Az Si4461 SPI perifériája 8 bites, MSB először adatstruktúrát használ [20].

<sup>19</sup> A 'GP' a készülék (GPS), a másik három karakter az üzenet típusára utal.

<sup>20</sup> A GPS modulból kiolvasható hasznos adatokat a 3.3.5. fejezet tartalmazza.

Az SPI periféria beállítása után inicializálnunk kell az Si4461-es rádiós IC-t. Az Si4461 rádiós paramétereinek beállítását, a rádiós ki- és bemeneti FIFO regisztereinek írását és olvasását, a bekövetkezett megszakítások kiolvasását, illetve további műveletek végrehajtását különböző parancsok kiadásával érhetjük el. minden parancs után meg kell várni, amíg az Si4461 a CTS (Clear To Send) állapotba való belépéssel jelzi, hogy a parancs végrehajtása befejeződött.<sup>21</sup>

A rádiós paraméterekhez valamint az automatikus csomagkezelőhöz tartozó regiszterek értékeinek meghatározásához a Silicon Labs WDS nevű szoftverét használtam. A WDS a paraméterek megadása után egy .h fájlt generál, ami az összes regiszter értékét #define-ként tartalmazza [23]. A WDS-ben nem találtam olyan opciót, ahol a rádiós ki- és bemeneti csomagkezelő paramétereit külön lehet beállítani,<sup>22</sup> ezért ezeknek a regisztereknek az értékét az adatlap segítségével határoztam meg [19]. A legfontosabb rádiós paraméterek a következők:

- Frekvencia: 433 MHz.
- Moduláció: FSK.
- Frekvencialöket: 100 kHz.
- Bemeneti szűrő sávszélessége: 250 kHz.
- Adóteljesítmény: 10 dBm.
- Adatsebesség: 10 kbps.

A rádiós adatcsomag felépítése a 3.20. ábrán látható.



- Az automatikus csomagkezelő tölti ki      - A vezérlő szoftver tölti be és olvassa ki a TX és RX FIFO-ból

**3.20. ábra.** A rádiós adatcsomag felépítése

Rádiós adatcsomag küldéséhez először az adatot a megfelelő parancs kiadása után be kell írni a TX FIFO-ba, majd az Si4461-et adás (TX) állapotba kell kapcsolni. A rádiós adatcsomag kiküldésének végét az Si4461 a megszakítás regiszterek megfelelő bitjének bebillentésével, és az nIRQ lab logikai 0-ra állításával jelzi. Rádiós adatcsomag vételéhez az Si4461-et vétel (RX) állapotba kell kapcsolni, a rádiós adatcsomag beérkezését az Si4461 a megszakítás regiszterek megfelelő bitjének bebillentésével, és az nIRQ lab logikai 0-ra állításával jelzi. A beérkező rádiós adatcsomagot a megfelelő parancs kiadásával lehet kiolvasni az RX FIFO-ból. Az automatikus csomagkezelő minden rádiós csomag elküldése előtt

<sup>21</sup>Az Si4461 konfigurálásának és vezérlésének pontos menete, valamint az egyes parancsok jelentése megtalálhatóak a vonatkozó dokumentumokban [19] [20].

<sup>22</sup>A laptortól a vezérlőegység felé 8 bájtos, a vezérlőegységtől a laptop felé pedig 60 bájtos adatokat továbbít.

kiszámolja a CRC ellenőrző összeget. A vevő a csomag beérkezése után szintén kiszámolja a CRC ellenőrző összeget, ha a kapott és számított eredmény nem azonos, a vevő eldobja a csomagot, majd a megszakítás regiszterek megfelelő bitjének bebillentésével, és az nIRQ láb logikai 0-ba állításával jelzi a hibás adatcsomag beérkezését [21].

### 3.3.4. Motorok vezérlése

**Léptetőmotor** A léptetőmotor vezérlése impulzusokkal történik, a léptetőmotor az impulzus felfutó élre a léptetőmotor-vezérlő beállításainak függvényében valamelykorra szöggel elfordul. Az impulzust a mikrokontroller egyik számlálójával állítom elő. Az impulzus hossza és az egymást követő impulzusok gyakorisága a számláló beállításaitól függ. A léptetőmotor vezérlését úgy állítottam be, hogy a vezérlési módtól függetlenül egy teljes  $360^\circ$ -os kör megtétele 10 másodpercig tartson, tehát  $\frac{1}{32}$ -es mikrolépéses üzemmódban az impulzusok frekvenciája a következő képpel számolható:

$$f_{impulzus} = \frac{360^\circ}{\frac{\text{lépésszög}}{32}} \cdot \frac{1}{10 \text{ sec}} = \frac{360^\circ}{\frac{1,8^\circ}{32}} \cdot \frac{1}{10 \text{ sec}} = 640 \text{ Hz}. \quad (3.21)$$

A RaspiCam szalagkábellel csatlakozik a Raspberry Pi-hez, ezért a vezérlést a szalagkábel megtörésének elkerülése érdekében úgy oldottam meg, hogy a léptetőmotor az alaphelyzetéhez képest csak  $\pm 180^\circ$ -ot fordulhasson el. A léptetőmotor vezérlési módját a megfelelő vezérlő parancsokkal lehet módosítani,<sup>23</sup> az alapbeállítás az  $\frac{1}{32}$ -es mikrolépéses üzemmód.

A léptetőmotor szögfelfordulását a kliensprogramban kiadott parancsokkal lehet vezérelni, a szög nagyságának és a forgatás irányának megadása után a szoftver a vezérlési módtól függően a parancsfeldolgozó megszakítás rutinban kiszámolja, hogy milyen frekvenciával és hány impulzust kell küldeni a léptetőmotor-vezérlő IC-nek, majd visszatér a főprogramba, és a főprogramban számolja a hátralévő impulzusok számát, ezáltal nem blokkolja a program futását azzal, hogy a megszakítás rutinban marad az adott szög elfordulásához szükséges ideig. A szoftver egy globális változóban tárolja a szögfelfordulás mértékét az alapállapothoz képest, a kommunikációs kapcsolat megszakítása esetén a léptetőmotor visszatér a kiindulási állapotba.

**DC motor** A DC motor vezérlése a 3.2.5. fejezetben bemutatott módon történik. A 20 kHz-es vezérlő PWM jelet számláló segítségével állítom elő. A PWM jel kitöltési tényezője a kliensprogramban kiadott parancssal módosítható.

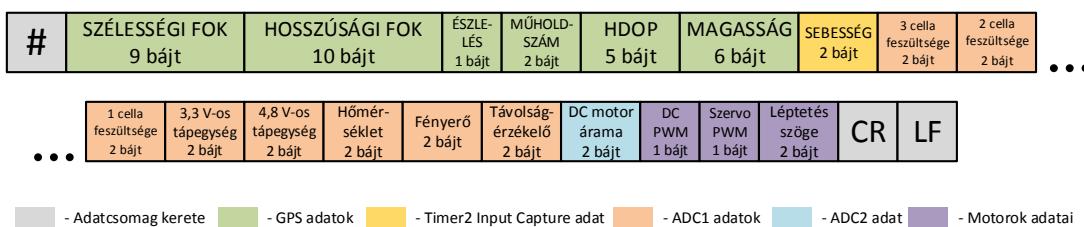
**Szervomotor** A szervomotor vezérlése a 3.2.6. fejezetben bemutatott módon történik. A szervomotor vezérléséhez szükséges adott hosszúságú impulzust számláló segítségével állítom elő. Az impulzus szélessége a kliensprogramban kiadott parancssal módosítható.

---

<sup>23</sup>A kliensprogram legújabb szoftververziójában nem lehetséges kiadni a léptetőmotor vezérlését módosító parancsokat.

### 3.3.5. Telemetria- és szenzoradatok gyűjtése

A telemetria- és szenzoradatokat a szoftver egy 60 bájt hosszúságú tömbben tárolja. A tömb megfelelő elemei a telemetria- és szenzoradatok beolvasása után folyamatosan frissülnek és másodpercenként kiküldésre kerülnek, amennyiben a telemetria- és szenzoradatok küldése engedélyezve van. A telemetria- és szenzoradatokat tartalmazó adatcsomag felépítése a 3.21. ábrán látható.



**3.21. ábra.** A telemetria- és szenzoradatokat tartalmazó 60 bájtos adatcsomag felépítése

Az adatcsomag # karakterrel kezdődik és CR/LF karakterekkel végződik, így a kliensprogram az adatcsomag hosszából, valamint a kezdő- és végkarakterekből egyértelműen el tudja döntenı, hogy telemetria- és szenzoradatokat tartalmazó csomag érkezett-e. Az adatcsomag első 33 bájtja GPS információkat tartalmaz, a GPS információk kinyerésének módját a 3.3.2. fejezet tartalmazza.

A következı két bájt a sebességmérés adatait tartalmazza. A sebesség mérése fehér és fekete felületek váltakozási frekvenciájának mérésén alapszik (lásd: 3.2.8. fejezet). A sebességmérő szenzor kimenetét egy számláló input capture bemenetére kötöttem, és mivel az input capture bemenetén lévő komparátor a fehér felülethez tartozó 2,5 V-os feszültséget logikai 1-nek, a fekete felülethez tartozó 0,1 V-os feszültséget pedig logikai 0-nak értelmezi, ezért a feladat egy impulzus frekvenciájának mérésére redukálódik. A számláló értéke két impulzus felfutó éle közti eltelt időt adja meg. Az adatcsomag a számláló értékét tartalmazza,<sup>24</sup> a kiértékelés a kliensprogramban történik.

Az akkumulátor cella- és tápfeszültség mérők, a hőmérő, a megvilágításmérő és a távolságérzékelő szenzor analóg kimeneteit az ADC1 különbözı csatornáira kötöttem. Az analóg-digitális átalakító értékét a mikrokontroller tehermentesítése érdekében a DMA vezérlő olvassa ki. A DMA egy számláló által 100 ms-onként előállított időzítőjel hatására egymás után kiolvassa és betölти egy tömbbe az ADC1 felkonfigurált csatornáinak értékét, végül megszakítást küld a mikrokontrollernek. Az analóg vonalakon megjelenő zavarok kiszűréséhez 10-es mozgóátlagolást alkalmazok. A feszültségedatok átlagolása utáni értékeket betöltöm az adatcsomag megfelelő bájtjaiba.<sup>25</sup> Az adatok kiértékelése a kliensprogramban történik.

<sup>24</sup>A számláló 16 bites, az adatcsomag viszont 8 bites szervezésű. A számláló értékének felső 8 bitjét az adatcsomag kisebb indexő, míg az alsó 8 bitjét a nagyobb indexő bájtja tartalmazza.

<sup>25</sup>Az ADC 12 bites, az adatcsomag viszont 8 bites szervezésű. Az analóg-digitális átalakítás és az átlagolás utáni 12 bit felső 4 bitjét az adatcsomag kisebb indexő, míg az alsó 8 bitjét a nagyobb indexő bájtja tartalmazza.

A DC motor árama a MOSFET-ek nagyfrekenciás kapcsolójéle miatt nem stabil, ezért a DC motorral sorba kötött sönellenálláson eső feszültség értéke sem stabil. A feszültség DC komponensének megállapításához hardveres és szoftveres átlagolást is alkalmazok. A hardveres átlagolást a sönellenállás és az árammérő INA196-osok közé helyezett 1,6 kHz törésponti frekvenciájú RC szűrő végzi. A szoftver pedig 100-as mozgóátlagot számol az analóg-digitális átalakítás után és az eredményt betölti a telemetria- és szenzoradatokat tartalmazó adatesomag megfelelő bajtjaiba. A DC árammérő szenzor kimeneti feszültségét az ADC2-vel 10 ms-onként mintavételezem.

A telemetria- és szenzoradatok utolsó 3 bitje a DC motor vezérlő PWM jelének aktuális kitöltési tényezőjét, a szervomotor vezérlő PWM jelének impulzusának hosszát, és a léptetőmotor aktuális helyzetét tartalmazza.

### 3.3.6. Vezérlő parancsok

A mobil adatgyűjtő egység vezérlése a kliensprogramban kiadott előre meghatározott hosszúságú és felépítésű vezérlő parancsokkal történik. A vezérlő parancsok felépítése a 3.22. ábrán látható.

<b>#</b>	<b>BETŰKÓD 2 bajt</b>	<b>SZÁMKÓD 3 bajt</b>	<b>CR</b>	<b>LF</b>
----------	---------------------------	---------------------------	-----------	-----------

**3.22. ábra.** A vezérlő parancsok felépítése

Minden vezérlő parancs # karakterrel kezdődik és CR/LF karakterekkel végződik. A kétbajtos betűkód meghatározza, hogy a vezérlő parancs a vezérlőegység mely részegységehez szól, a 3 bajtos számkód pedig a részegységhoz tartozó vezérlő parancs kódját, vagy értékét jelenti. Tehát minden a  $26 \cdot 26 = 676$  betűkombinációhoz<sup>26</sup>  $10^3 = 1000$  különböző számkód tartozik, ez  $676 \cdot 1000 = 676000$  lehetséges vezérlő parancsot jelent. A jelenlegi szoftververzióban a mikrokontroller által értelmezett vezérlő parancsok kódját és jelentését a 3.5. táblázat tartalmazza.

Vezérlő parancs kódja	Vezérlő parancs jelentése	Megjegyzés
#DM050\r\n	DC motor vezérlése 50%-os kitöltési tényezőjű PWM jellel	A számkód értéke 0-100 lehet. 0-49 az előre-, 51-100 a hátramenetet jelenti
#DF000\r\n	DC motor 1 egységgel gyorsít	Aktuális PWM jel kitöltési tényezőjének csökkentése 1%-kal
#DB000\r\n	DC motor 1 egységgel lassít	Aktuális PWM jel kitöltési tényezőjének növelés 1%-kal

<sup>26</sup>Csak az angol ABC betűi használhatók.

#SM140\r\n	Szervomotor középre forgatása	A szervomotor vezérlő-impulzusának hossza: számkód·10 µs. A számkód értéke 105-175 lehet. 105-139 a jobbra, 141-175 a balra fordulást jelenti
#SJ000\r\n	A szervomotor jobbra forgatása 1,86°-kal	A vezérlő impulzus hosszának csökkentése 5 ms-mal
#SB000\r\n	A szervomotor balra forgatása 1,86°-kal	A vezérlő impulzus hosszának növelése 5 ms-mal
#LJ010\r\n	A léptetőmotor forgatása 10°-kal jobbra	A maximális érték 180° lehet
#LB010\r\n	A léptetőmotor forgatása 10°-kal balra	A maximális érték 180° lehet
#LC014\r\n	A léptetőmotor visszaforgatása a kiindulási állapotba	0-13 számkódú parancsok a léptetőmotor beállításait módosítják, a szoftver futása közben ezek a paraméterek nem módosíthatók, ezért a parancsok felsorolásától eltekintek
#AK4.2\r\n	DC motor áramkorlátjának beállítása ±4,2 A-ra	A számkód az áramkorlát értéke A-ben
#SK010\r\n	Sebességkorlát beállítása 10 km/h-ra	A számkód a sebességkorlát értéke km/h-ban
#UE042\r\n	Ütközéselhárítás, ha akadályt észlelt 42 cm-en belül	A számkód az akadály távolságát jelenti cm-ben
#KT000\r\n	4,8 V-os kapcsolóüzemű tápegység letiltása	–
#KT001\r\n	4,8 V-os kapcsolóüzemű tápegység engedélyezése	–
#SS000\r\n	Letiltja minden motor vezérlését, alaphelyzetbe állítja a kamerát és a szervomotort	Az akkumulátor lemerülése, és kommunikációs csatorna váltása után automatikusan végre hajtódik
#TS000\r\n	Telemetria- és szenzoradatok küldésének letiltása	–
#TS001\r\n	Telemetria- és szenzoradatok küldésének engedélyezése	–

#KV000\r\n	Kamera háttérvilágításának manuális kikapcsolása	A háttérvilágítás ekkor átvált automatikus be- és kikapcsolásra
#KV001\r\n	Kamera háttérvilágításának manuális bekapcsolása	–
#RE000\r\n	ISM sávú rádiós kommunikáció letiltása, telemetria- és szenzoradatok továbbítása a Raspberry Pi felé	A végső szoftververzióban az #RP001\r\n parancs váltja ki
#RE001\r\n	ISM sávú rádiós kommunikáció engedélyezése, telemetria- és szenzoradatok továbbítása ISM sávú rádión keresztül	A végső szoftververzióban az #RP000\r\n parancs váltja ki
#RP111\r\n	A Raspberry Pi inicializálása befejeződött	Ezt a vezérlő parancsot a Raspberry Pi küldi, nem adható ki a kliensprogramban
#RP000\r\n	A Raspberry Pi bezárja az UDP kommunikációs portokat.	A parancs csak ISM sávú rádión keresztül adható ki. A vezérlőegység leállítja a motorokat és átáll ISM sávú rádiós kommunikációra, a Raspberry Pi továbbá leállítja az élő kamerakép küldését
#RP001\r\n	A Raspberry Pi megnyitja az UDP kommunikációs portokat	A parancs csak ISM sávú rádión keresztül adható ki. A vezérlőegység leállítja a motorokat és átáll Wi-Fi-s kommunikációra
#RP002\r\n	A Raspberry Pi TCP kapcsolódás után elkezdi a JPEG tömörítésű képek küldését	–
#RP003\r\n	A Raspberry Pi elkezdi a H.264 kódolású videó-adatfolyam küldését	–
#RP004\r\n	A Raspberry Pi leállítja a JPEG tömörítésű képek küldését, lezárja a TCP kapcsolatot	–

#RP005\r\n	A Raspberry Pi leállítja a H.264 kódolású videó-adatfolyam küldését	-
#RP006\r\n	A Raspberry Pi újraindítása	A vezérlőegység átvált ISM sávú rádiós kommunikációra
#SR000\r\n	Vezérlőegység és Raspberry Pi újraindítása	A kliensprogram is alapállapotba kerül

**3.5. táblázat.** A vezérlő parancsok kódja és jelentése

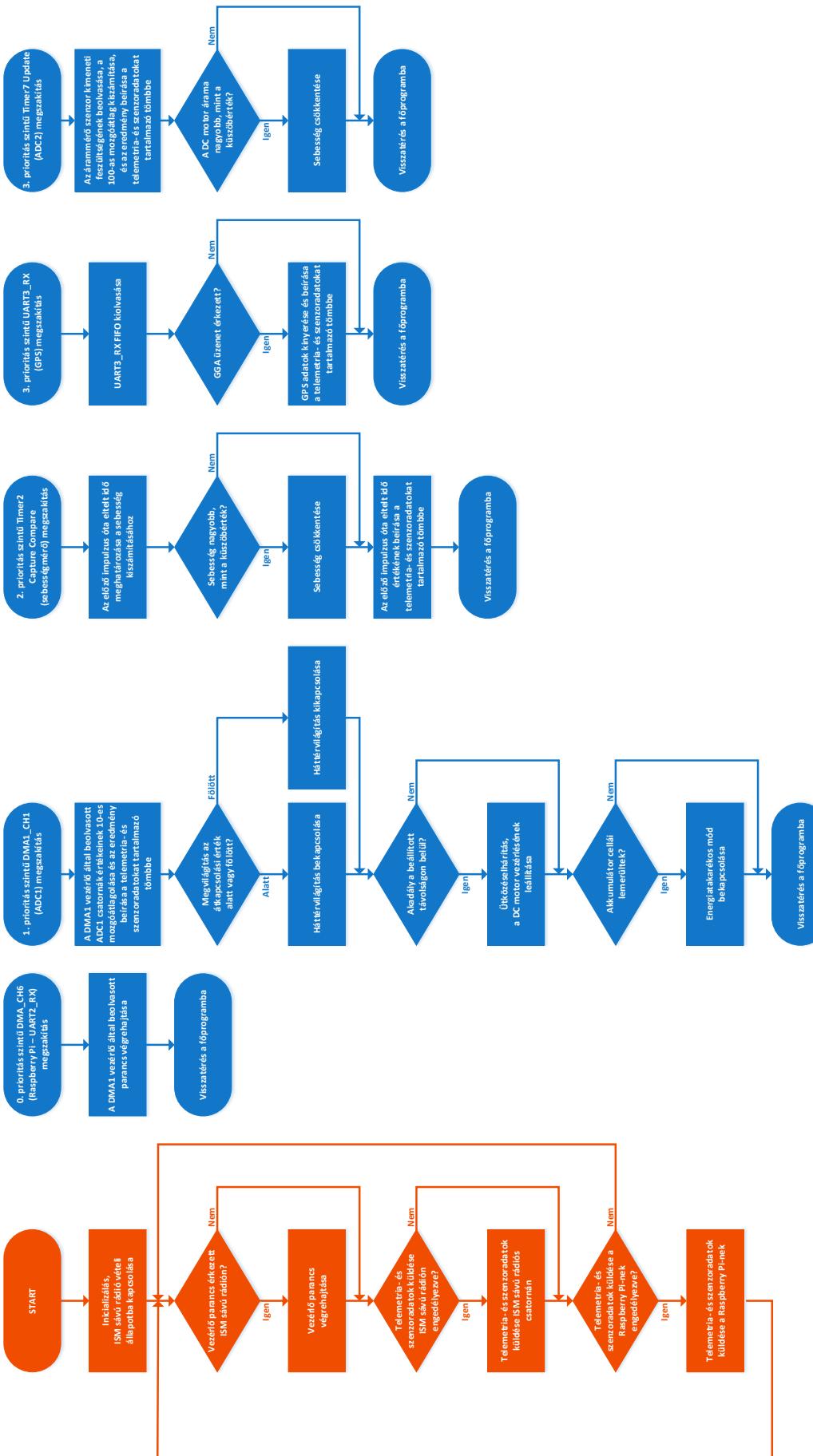
### 3.3.7. Főprogram

A mikrokontroller a bekapcsolás után inicializálja az órajelet, a perifériákat, a megszakításvezérlőt, majd vételi állapotba kapcsolja az ISM sávú rádiót, végül belép a főprogram végtelen ciklusába. A program minden ciklus elején ellenőrzi, hogy érkezett-e vezérlő parancs az ISM sávú rádiós csatornán, amennyiben igen, úgy végrehajtja a parancsot. Ezután az ISM sávú rádió engedélyezése és a Raspberry Pi engedélyezése jelzőbitek állása alapján eldönti, hogy mely kommunikációs csatornán keresztül küldi a telemetria- és szenzoradatokat, amennyiben a telemetria- és szenzoradatok küldése engedélyezve van. A főprogram futását különböző prioritású megszakítások szakíthatják félbe. A megszakítás rutinokon belüli utasítások végrehajtása után a szoftver visszatér a főprogramba.

A mikrokontrolleren futó beágyazott szoftver működési folyamata a 3.23. ábrán látható, a főprogramot narancssárgával, a megszakításokat pedig kékkel ábrázoltam. A magasabb prioritású megszakításhoz kisebb szám tartozik, vagyis a legnagyobb prioritású megszakítás a 0. szintű a legalacsonyabb prioritású pedig a 3. szintű.<sup>27</sup>

---

<sup>27</sup> A mikrokontroller az egy időben érkező megszakítás kérések közül a nagyobb prioritású hajtja végre először, az alacsonyabb prioritásúkat pedig a prioritás szintje alapján berakja a megszakítási sorba, így elköpzelhető olyan eset, hogy az alacsonyabb prioritású megszakítás kiszolgálására nem kerül sor a magasabb prioritású megszakítás kiszolgálások sűrűsége miatt. Az azonos prioritási megszakítások kiszolgálása a kérés beérkezésének időrendi sorrendjében történik. Egy magasabb prioritású megszakítás félbeszakíthatja az alacsonyabb prioritású megszakítás kiszolgálását.



**3.23. ábra.** A mikrokontrolleren futó beágyazott szoftver folyamatábrája

## 4. fejezet

# Raspberry Pi

A Raspberry Pi egy olcsó, bankkártyaméretű mini számítógép. Eredetileg oktatási célokra fejlesztették ki az Egyesült Királyságban. A Raspberry Pi egyetlen áramköri lapra szerelt Broadcom BCM2835 típusú SoC<sup>1</sup>-ból és a köré épített perifériákból áll. Háttértárolóként egy 8 GB-os SD kártyát használunk. A Raspberry Pi-re különböző operációs rendszerek telepíthetők, a legelterjettebb a Linux Debian alapú Raspberry Pi-re optimalizált Raspbian nevezetű operációs rendszer, így én is ezt használtam. Az élő kameraképet a Raspberry Pi-hez kapható szabványos soros interfészen keresztül (4.1. ábra) csatlakoztatható kamera modul szolgáltatja. A Wi-Fi-kapcsolat kialakításához egy USB-s Wi-Fi adaptort használunk. A Raspberry Pi-n futó szoftvert az előre implementált soros port és kamerakezelő könyvtárak miatt Python nyelven írtam [2] [38].

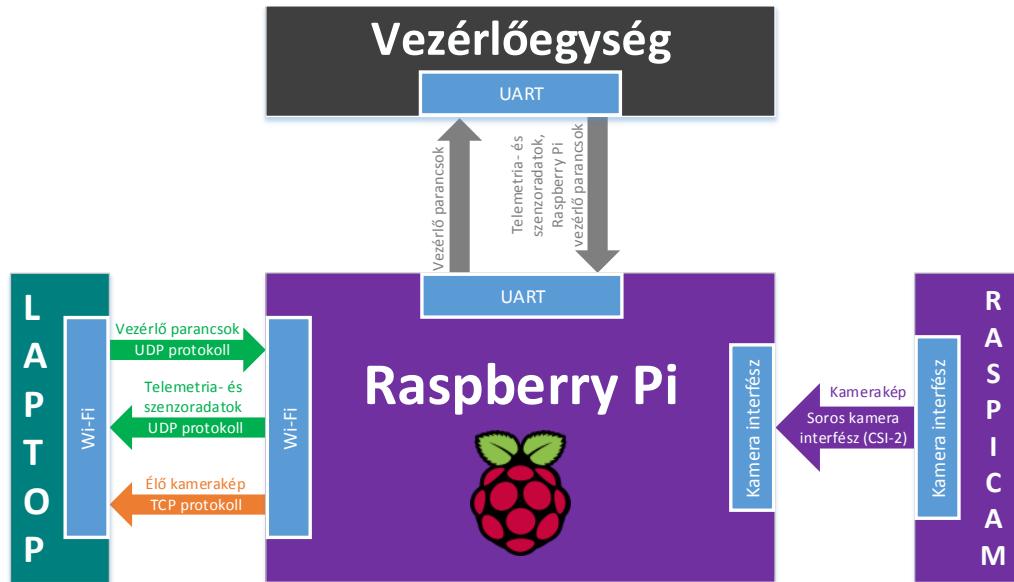
Ebben a fejezetben bemutatom a Raspberry Pi kommunikációs csatornáit, a Raspberry Pi inicializálását, az élő kamerakép előállításának módjait, és a parancsfeldolgozó Python script működését.

---

<sup>1</sup>SoC: System on Chip - Egy csipbe van integrálva a CPU, GPU, DSP, és az SDRAM.

#### 4.1. A Raspberry Pi kommunikációs csatornái

A 4.1. ábrán a Raspberry Pi különböző kommunikációs csatornái, a kommunikációs csatornák irányai, és az üzenettípusok láthatók.



4.1. ábra. A Raspberry Pi kommunikációs csatornái

A Raspberry Pi és a vezérlőegység UART protokollon keresztül kommunikál. A Raspberry Pi a Wi-Fi-n beérkező 8 bájtos vezérlő parancsokat tartalmazó UDP csomagokat feldolgozza, és amennyiben a vezérlő parancs a vezérlőegységnek van címezve, úgy UART-on továbbküldi.

A vezérlőegység vagy a 60 bájtos telemetria- és szenzoradatokat tartalmazó csomagot, vagy az ISM sávú rádión beérkező Raspberry Pi-nek címzett 8 bájtos vezérlő parancsot küldi UART-on a Raspberry Pi felé. A Raspberry Pi a vezérlő parancsot végrehajtja, a telemetria- és szenzoradatokat tartalmazó csomagot pedig UDP protokollon keresztül továbbküldi a laptopnak.

A RaspiCam soros kamera interfészen (CSI-2) keresztül küldi el a nyers kameraképet a Raspberry Pi-nek. A Raspberry Pi GPU-ja elvégzi a képfeldolgozást, majd TCP<sup>2</sup> protokollon keresztül továbbküldi az élő kameraképet a laptopnak.

<sup>2</sup>A TCP egy kapcsolat-orientált, megbízható protokoll. A TCP protokoll a fizikai kapcsolat megszakadásáig garantálja a csomagok hibamentes átvitelét. Az élő kamerakép pontos visszaállításához hibamentes adatátvitel szükséges.

## 4.2. A Raspberry Pi inicializálása

A Raspberry Pi bekapcsolása előtt az operációs rendszert telepíteni kell az SD kártyára. A Raspbian Wheezy operációs rendszert tartalmazó képfájl és a telepítés menete megtalálható a Raspberry Pi hivatalos oldalán [8].

A bekapcsolás után célszerű engedélyezni az SSH protokollon való hozzáférést, így a Raspberry Pi távolról, helyi hálózaton keresztül is elérhető, nem szükséges külön monitor és billentyűzet használata. Az SSH hozzáférést a *sudo raspi-config* parancs beírása után felugró menüben kapcsolhatjuk be. Az SSH kapcsolat kialakításához ismerni kell a Raspberry Pi IP címét. Mivel a helyi hálózatokhoz kapcsolódó eszközök IP címét általában a hálózati útvonalválasztó eszköz (router) DHCP-je osztja ki, ezért az IP cím minden kapcsolódáskor váltohat, ennek elkerülése érdekében statikus IP címet állítottam be a Raspberry Pi-n. A statikus IP cím beállításához a *sudo nano /etc/network/interfaces* parancs kiadása után megfelelően módosítanunk kell az interfaces fájlt.<sup>3</sup>

A vezetéknélküli kapcsolat felépítéséhez a *sudo nano /etc/wpa\_supplicant/wpa\_supplicant.conf* parancs kiadása után a wpa\_supplicant.conf fájlból meg kell adnunk a vezetéknélküli hálózat nevét, a titkosítás kódját, és a titkosítás típusát.

Az SSH kapcsolat kialakítása után a *sudo apt-get update* parancssal frissítettem a Linux csomagkezelőjét (Advanced Packaging Tool), így a csomagkezelő a telepítési kívánt programok, modulok legfrissebb verzióit tartalmazza. A *sudo apt-get upgrade* parancssal frissítettem a már feltelepített programokat, modulokat.

A programozáshoz először telepítenünk kell a Python legfrissebb verzióját és a soros port, illetve a kamera kezeléséhez szükséges Python könyvtárakat. A telepítést a következő parancsok kiadásával tehetjük meg:

```
sudo apt-get install python3
sudo apt-get install python3-pip
pip install pyserial
sudo apt-get install python3-picamera
```

A kamera és a soros port használatához először a *sudo raspi-config* menüben engedélyeznünk kell azokat. A soros port használatához továbbá le kell tiltanunk a soros porti bejelentkezés lehetőségét, ezt a *sudo nano /etc/inittab* parancs kiadása után az inittab fájlból lévő *T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100* sor törlésével érhetjük el, valamint le kell tiltani a rendszer indításakor a kerneladatok soros portra való kiírását. A kerneladatok soros portra való kiírásának tiltásához először a *sudo nano /boot/cmdline.txt* parancs kiadása után a cmdline.txt fájlból ki kell törölni a *console=ttyAMA0,115200 kgdboc=ttyAMA0,115200* részeket, majd a boot mapában lévő tömörített kernel.img fájlt ki kell csomagolni, különben a rendszer indításakor a kernel.img fájl kicsomagolásának állapota kiírásra kerül a soros portra.<sup>4</sup>

<sup>3</sup> A nano a Linux szövegszerkesztője, amennyiben az operációs rendszer nem tartalmazza, úgy a *sudo apt-get install nano* parancssal feltelepíthető.

<sup>4</sup> A kerneladatok soros portra való kiírásának letiltása azért szükséges, mert a vezérlőegységen lévő mikrokontroller DMA vezérlője 8 bájtonként olvassa ki az UART2 RX pufferét, így ha a kerneladatok mérete nem 8 bájt egész számú többszöröse, akkor a DMA a Raspberry Pi által küldött 8 bájtos soros csomagokat két különböző olvasási ciklusban fogja beolvasni és a „kettévágott” vezérlő parancsok értelmezhetetlenek

A Raspberry Pi-n futó szoftver indításához először be jelentkeznünk egy felhasználói fiókba, majd futtatnunk kell a parancskezelő Python script-et. A Raspberry Pi-t ezért úgy állítottam be, hogy a rendszer indítása után automatikusan bejelentkezzen és elindítsa a parancskezelő Python script-et. A Raspberry Pi soros porton jelzi a vezérlőegységnek, ha az inicializálási folyamat befejeződött. A felhasználói fiókba való automatikus belépéshez a `sudo nano /etc/inittab` parancs kiadása után az inittab fájlban a

```
1:2345:respawn:/sbin/getty --noclear 38400 tty1 sort
1:2345:respawn:/bin/login -f pi tty1 </dev/tty1 >/dev/tty1 2>&1
```

sorra kell módosítani. A parancskezelő Python script bejelentkezés utáni automatikus futtásához a `sudo crontab -e` parancs kiadása után a fájl végére be kell írni a `@reboot python3 /home/pi/diploma/parancskezelo.py &` parancsot. A parancs végi `&` jel a Python script háttérben történő futtatását eredményezi.

### 4.3. Élő kamerakép előállítása

Az élő kamerakép előállításához a 2. fejezetben említett RaspiCam modult használtam. A RaspiCam szalagkábellel csatlakoztatható a Raspberry Pi-n lévő szabványos 15 érintkezős csatlakozóhoz. A RaspiCam a kamera beállításaitól függő nyers adatokat CSI-2 interfészen keresztül küldi el a GPU-nak. A nyers adatok feldolgozását, a képtömörítést vagy videó kódolást a GPU végzi el. A RaspiCam legfontosabb tulajdonságait a 4.1. táblázat foglalja össze.

RaspiCam tulajdonságai	
Felbontás	2592x1944 (5 Mp)
Videó módok	1080p30, 720p60, 640x480p60/90
Képarány	16:9, 4:3
Képformátumok	JPEG, JPEG + RAW, GIF, BMP, PNG, YUV420, RGB888
Videó formátumok	RAW, H.264, MJPEG
Különböző effektek	
Automatikus vagy egyéb expozíciós módok	
Automatikus vagy állítható fehéregyenúsúly	

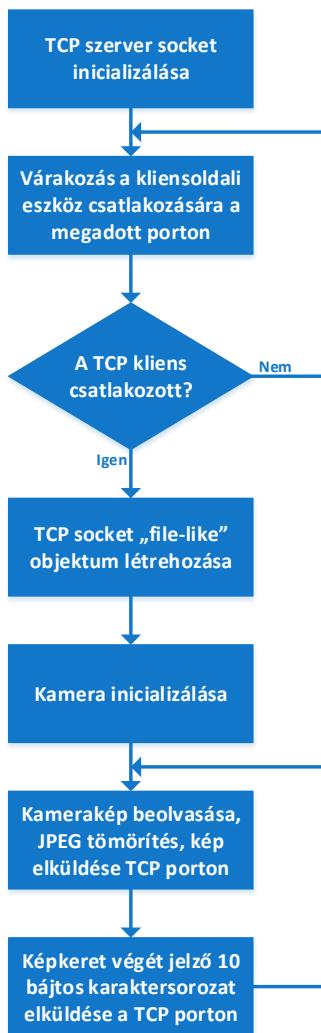
4.1. táblázat. A RaspiCam legfontosabb tulajdonságai [6]

Mivel a feladatkiírásban szereplő legalább 640x480 képpont felbontású, 10 kép/másodperces frissítésű élő kamerakép formátuma nincs előre definiálva, ezért az élő kamerakép átvitelére több lehetséges módot is találtam, ezek közül az alábbi kettő került megvalósításra.

**JPEG tömörítésű képek sorozata** Első megoldásként JPEG tömörítésű képeket különdök TCP protokollon keresztül a kliensprogramnak. A kliensprogram az egymás után beérkező képeket folyamatosan megjeleníti, így kialakítva az élő kameraképet. A 4.2. ábrán a JPEG tömörítésű kép előállítását és TCP protokollon való küldését megvalósító Python script folyamatábrája látható.

---

lesznek.



**4.2. ábra.** A JPEG tömörítésű képek előállításának és küldésének folyamata

A Python script először létrehoz egy TCP kapcsolódási pontot (TCP socket), amely egy kiválasztott porton keresztül várja egyetlen, bármilyen IP című kliensoldali eszköz csatlakozását, tehát jelen esetben a Raspberry Pi a TCP szerver és a laptop a TCP kliens. A kliens eszköz sikeres csatlakozása után a szoftver létrehoz egy binárisan írható „file-like” objektumot. A „file-like” objektumba írása gyakorlatilag megfelel a TCP porton való adatküldésnek. A program ezután inicializálja a kamerát, vagyis beállítja a 640x480 képpontos felbontást, a 10 kép/másodperces frissítési sebességet, a képtömörítés formátumát, a használni kívánt kameraportot,<sup>5</sup> illetve a már tömörített képet összekapcsolja a TCP socket „file-like” objektumával, így nem szükséges külön művelet a kép TCP porton való elküldéséhez. Az inicializálás után a program egy végtelen cikluson belül folyamatosan

<sup>5</sup>A kép készítése során a RaspiCam videó- és a képportja közül választhatunk. A videó- és képport csak logikai portokat jelölnek, az adatok ugyanazon a fizikai csatornán mennek keresztül. A képport a kép készítésekor minden a szenzor teljes felületét használja (a kisebb felbontású képet utólag méretezi át) és erős zajszűrő algoritmusokat alkalmaz, míg a videóport az előre beállított felbontást alkalmazza és a zajszűrő algoritmusá is gyengébb. Az élő kamerakép előállításához a gyorsabb videóportot használtam.

– a megadott képfrissítés gyakoriságától, a képfeldolgozás, illetve tömörítés sebességétől és a Wi-Fi-kapcsolat minőségétől függően – küldi a képeket, továbbá minden kép elküldése után egy 10 bájtos karaktersorozat<sup>6</sup> is elküldésre kerül, a kliensprogram a karaktersorozat alapján tudja szétválasztani az egyes képkockákat [18].

**H.264 kódolású videó-adatfolyam** A másik megoldás H.264 kódolású videó-adatfolyam létrehozása és küldése a megadott porton. A kliensoldalon a port és a Raspberry Pi IP címe alapján lehet fogadni a videó-adatfolyamot, majd továbbítani a megjelenítő programnak.

A H.264-es videó-adatfolyam előállításához a *raspivid -w 640 -h 480 -t 0 -fps 20 -b 5000000 -o -* parancsot használtam. A *-w -h -fps* paraméterekkel beállítható a videó mérete és a képfrissítés gyakorisága, a *-b* az adatsebességet jelenti bps-ban, a *-t* a videó hossza másodpercben (0 másodperc a végtelent jelenti), a *-o*-val pedig beállítható, hogy a videó-adatfolyam megjelenítésre kerüljön-e a Raspberry Pi-hez csatlakoztatott monitoron [7]. A videó-adatfolyam elküldéséhez a netcat nevű programot használtam. A netcat-et a *sudo apt-get install netcat* parancssal telepíthetjük a Raspberry Pi-re, majd az *nc -l -p 5001* parancssal indíthatjuk el. A netcat alapértelmezetten TCP protokollt használ, a hibamentes videóátvitel megvalósítása érdekében ezen nem is változtattam. A *-l* paraméter a „listener” módot jelöli, vagyis ebben az esetben is a Raspberry Pi a szerveroldal, és a videó-adatfolyam küldése csak a kliensoldali eszköz csatlakozása után kezdődik. A *-p 5001* a videó-adatfolyam elküldéséhez használt portot jelöli. Tehát a Raspberry Pi oldalon a H.264-es adatfolyam előállításának és küldésének teljes parancsa a következő:

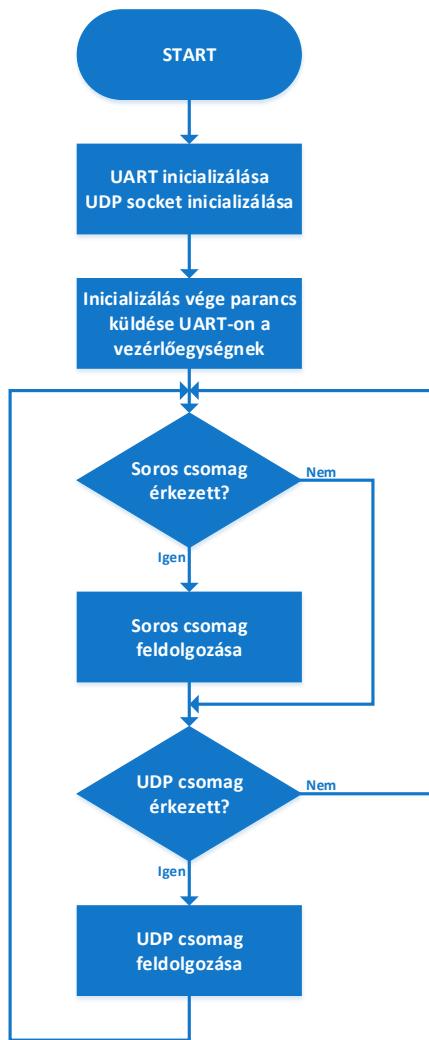
```
raspivid -w 640 -h 480 -t 0 -fps 20 -b 5000000 -o - / nc -l -p 5001 &
```

#### 4.4. Parancsfeldolgozó szoftver

A Raspberry Pi a rendszer inicializálása után statikus IP címen automatikusan csatlakozik a laptopon létrehozott vezetéknélküli hozzáférési ponthoz, majd automatikusan belép egy felhasználói fiókba és elindítja a parancsfeldolgozó Python scriptet. A parancsfeldolgozó Python script folyamatábrája a 4.3. ábrán látható.

---

<sup>6</sup>Elméletileg ez a 10 bájtos karaktersorozat a JPEG tömörítésű képben is előfordulhat, azonban ennek esélye elhanyagolható.



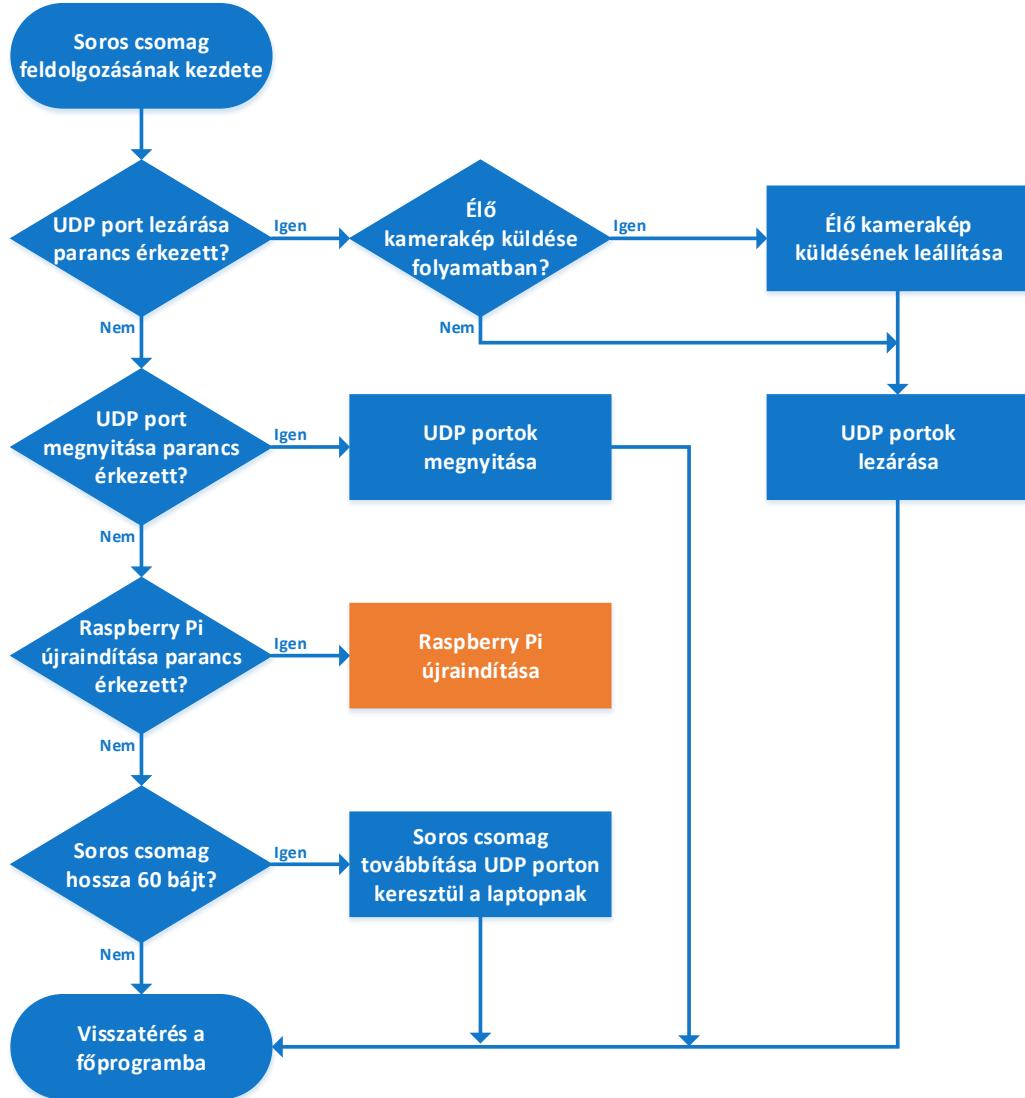
**4.3. ábra.** A *Raspberry Pi-n* futó parancsfeldolgozó Python script folyamatábrája

A program futása során először inicializálja az UART-ot, majd létrehoz egy UDP kapcsolódási pontot, amely a megadott portokon keresztül kommunikál a laptopnal. Az UDP kommunikáció kialakításához ismerni kell a laptop IP címét, ezért a laptop IP címét a Raspberry Pi-hez hasonlóan statikusra állítottam. Az UART beállításai megegyeznek a 3.3.2. fejezetben ismertetett beállításokkal, azaz a kapcsolat full-duplex kialakítású, az UART adatcsomag felépítése 8N1, vagyis 1 START bit, 8 adatbit, 1 STOP bit, és 0 paritás bit. Az UART kommunikáció sebessége  $115200 \frac{\text{baud}}{\text{sec}}$ . Az inicializálás után a Raspberry Pi UART-on, a megfelelő vezérlő parancs kiadásával jelzi a vezérlőegységnek, hogy az inicializálás befejeződött [5] [25].

A parancsfeldolgozó Python script ezután figyeli, hogy érkezett-e adat az UART<sup>7</sup> vagy

<sup>7</sup>A program a sorvégi LF karakter beérkezésekor olvassa ki az UART puffer tartalmát. Mivel minden a vezérlő parancsokat, minden a telemetria- és szenzoradatokat tartalmazó csomagok LF karakterre végződnek, ezért a beolvasott adat 8 vagy 60 bájt hosszú, így a csomag mérete alapján egyértelműen eldönthető, hogy milyen típusú csomag érkezett.

az UDP<sup>8</sup> porton keresztül, ha igen, akkor az adat beolvasása után a program belép az UART vagy UDP parancsfeldolgozó függvénybe. Az UART-on vagy UDP porton beérkező adatok feldolgozásának folyamata a 4.4. és a 4.5. ábrákon látható.



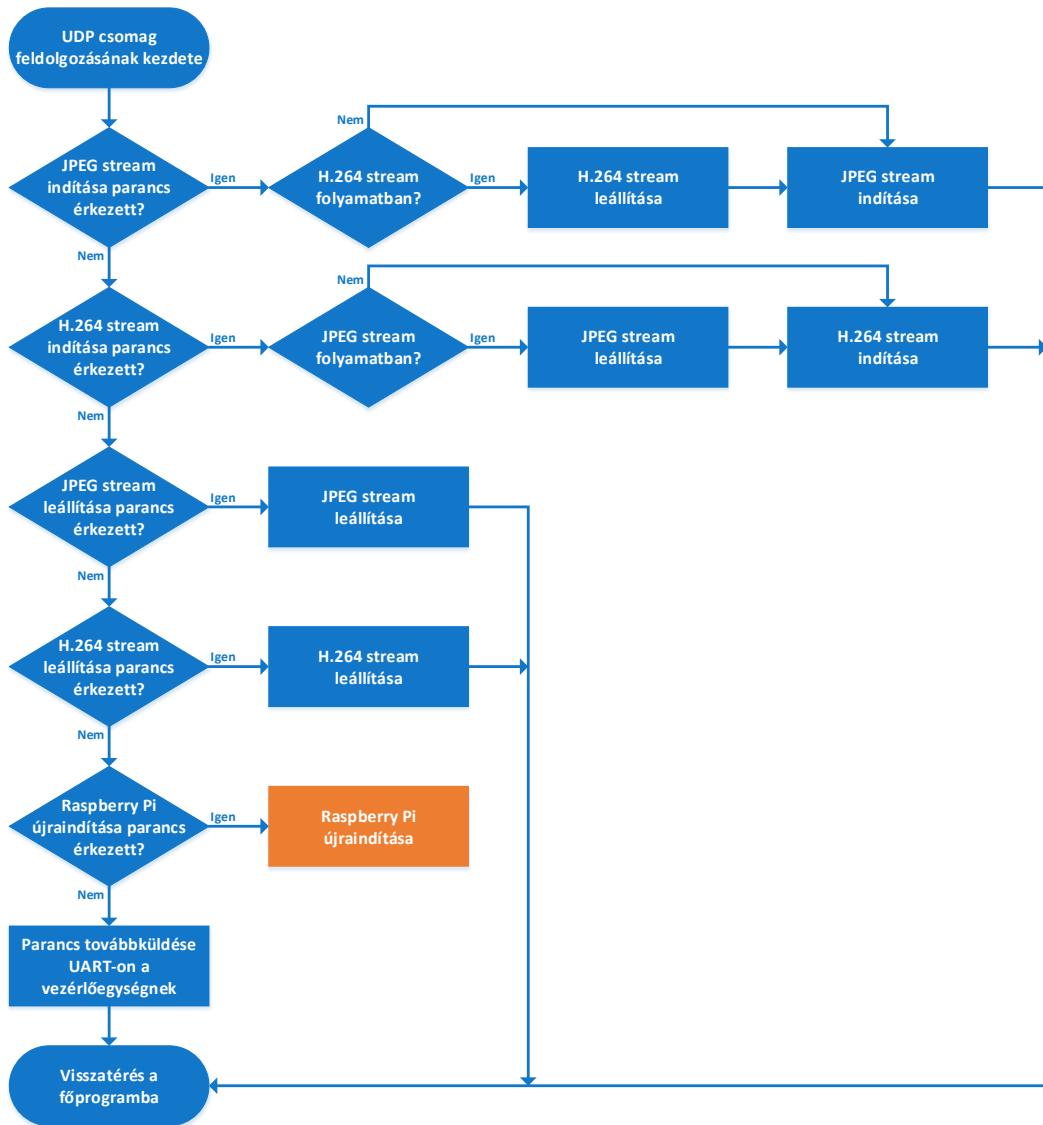
4.4. ábra. Az UART-on beérkező adatcsomag feldolgozásának folyamata

A soros adatcsomag feldolgozása során a program a csomag hossza és tartalma alapján eldönti, hogy továbbküldje-e a csomagot a laptopnak UDP porton keresztül, ha a Wi-Fi adapter csatlakozik a hálózathoz, vagy végrehajtsa-e a csomag tartalmában lévő utasítást. A Raspberry Pi soros adatcsomag-feldolgozója által értelmezett vezérlő parancsok a következők:

<sup>8</sup>Mivel UDP porton csak vezérlő parancsok érkezhetnek, ezért a program a vezérlő parancsok 8 bájtos hosszához igazodva 8 bájtonként olvassa ki az UDP port bemeneti pufferét.

- UDP kommunikációs portok lezárása, amennyiben fut, az élő kamerakép küldésének leállítása.
- UDP kommunikációs portok megnyitása.
- Raspberry Pi újraindítása.

Az első két parancsot csak a vezérlőegység adhatja ki azután, hogy a felhasználó a kliensprogramban kezdeményezi a kommunikációs csatorna váltását Wi-Fi-ről ISM sávú rádióra vagy ISM sávú rádióról Wi-Fi-re.



**4.5. ábra.** Az UDP porton beérkező adatcsomag feldolgozásának folyamata

Az UDP porton beérkező 8 bájtos adatcsomag feldolgozása során a program a csomag tartalma alapján eldönti, hogy továbbküldje-e a csomagot a vezérlőegységnak vagy

végre hajtsa-e a csomag tartalmában lévő utasítást. A Raspberry Pi UDP adatcsomag-feldolgozója által értelmezett vezérlő parancsok a következők:

- JPEG tömörítésű képek küldésének elindítása, amennyiben fut, a H.264 kódolású videó-adatfolyam küldésének leállítása.
- H.264 kódolású videó-adatfolyam küldésének elindítása, amennyiben fut, a JPEG tömörítésű képek küldésének leállítása.
- JPEG tömörítésű képek küldésének leállítása.
- H.264 kódolású videó-adatfolyam küldésének leállítása.
- Raspberry Pi újraindítása.

Ha egyéb vezérlő parancs érkezett, a Raspberry Pi UART-on továbbküldi a vezérlőegységeknek.

## 5. fejezet

# Kliensprogram

A mobil adatgyűjtő egység irányítása, konfigurálása, a telemetria- és szenzoradatok feldolgozása, és az élő kamerakép megjelenítése egy távoli hordozható eszközön történik. A hordozható eszközön egy interaktív felhasználói felület létrehozása célszerű, amely a különböző események (gombnyomás, billentyű lenyomás, stb.) hatására a kiválasztott kommunikációs csatornán keresztül elküldi a vezérlő parancsokat, valamint a beépített ablakokban feldolgozás után képes megjeleníteni a beérkező telemetria- és szenzoradatokat, illetve az élő kameraképet.

A kliensprogramot Matlabban írtam, majd egy .exe fájlt generáltam belőle, így a Matlab fordítóprogram (compiler) feltelepítése után a kliensprogram bármilyen számítógépen futtatható. A felhasználói felület kialakításához a Matlab grafikus felhasználói interfészét<sup>1</sup> (GUI) használtam [26].

Ebben a fejezetben először felsorolom a kliensprogrammal szemben támasztott előzetes követelményeket, majd bemutatom a kliensprogram működését, részletesen kitérve a különböző kommunikációs csatornák felépítésére, a telemetria- és szenzoradatok feldolgozására, az élő kamerakép megjelenítésére, és a mobil adatgyűjtő egység irányítására.

### 5.1. A kliensprogram előzetes követelményei

A kliensprogram fejlesztése előtt specifikáltam, hogy a kliensprogramnak milyen feladatokat kell ellátnia. A kliensprogram által ellátandó feladatok a következők:

- Vezetéknélküli hozzáférési pont létrehozása a hordozható eszközön.
- UDP kapcsolat kialakítása, bezárása.
- TCP kapcsolat kialakítása, bezárása.
- Soros kapcsolat kialakítása, bezárása.
- Kétirányú kommunikáció kezelése UDP porton keresztül.

<sup>1</sup>A Matlab GUI-ban beépített objektumokat illeszthetünk a felhasználói felületre. A kiválasztott objektum adott esemény (például gombnyomás) bekövetkezéskor meghívja a felhasználói felülethez tartozó .m fájlból lévő megfelelő függvényt és végrehajtja a függvényben lévő utasításokat (például egy vezérlő parancs elküldését soros porton).

- Kétirányú kommunikáció kezelése soros porton keresztül.
- Telemetria- és szenzoradatok feldolgozása, megjelenítése.
- Élő kamerakép megjelenítése.
- Kommunikációs csatornák közti váltás menedzselése.
- A mobil adatgyűjtő egység irányítása, konfigurálása.

## 5.2. A kliensprogram indítása és bezárása

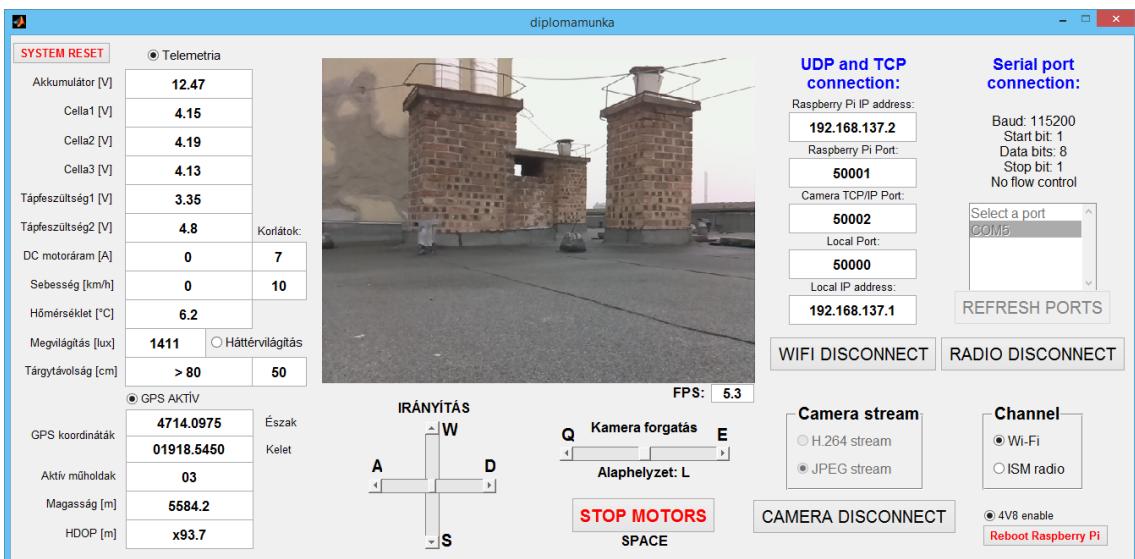
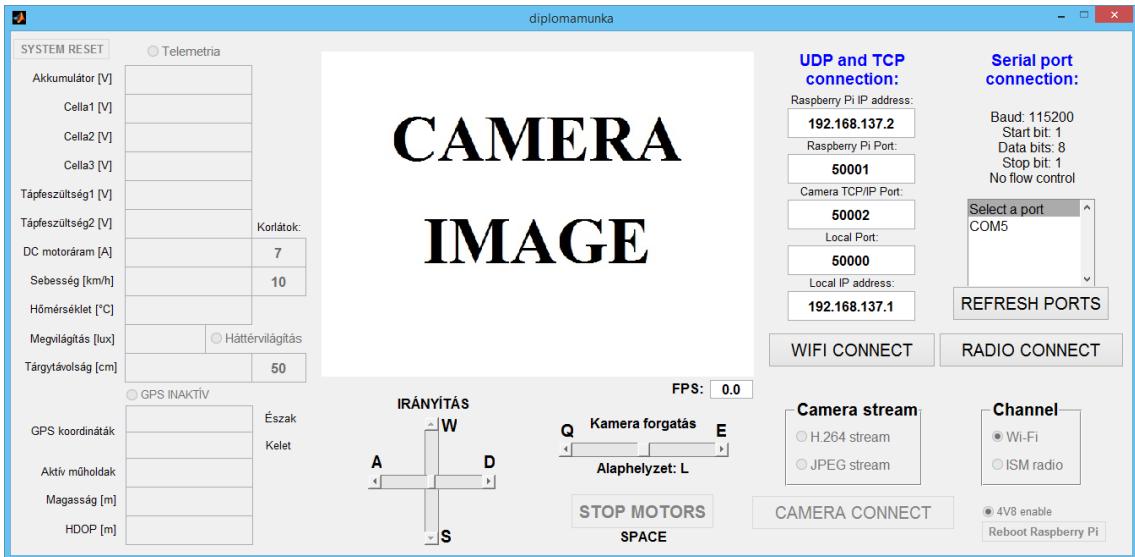
A kliensprogram az indítás után létrehoz egy RPi nevű vezetéknélküli hozzáférési pontot<sup>2</sup> és kiszámolja a hőmérőhöz, a távolságérzékelőhöz, és a megvilágításmérő szenzorhoz tartozó poligonok értékét, így e három szenzorhoz tartozó digitalizált kimeneti feszültségi adatok nem csak a mérési vagy adatlapi adatok diszkrét pontjaiban értelmezhetők, hanem a kapott függvény alapján bárhol. A vezetéknélküli hozzáférési pont létrehozásához a program tulajdonképpen lefuttat egy batch fájlt, ami a hozzáférési pont létrehozásához szükséges Windows parancsokat tartalmazza. A Raspberry Pi ehhez a vezetéknélküli hozzáférési ponthoz csatlakozik az indítás után.

A soros, az UDP, és a TCP portokat a kliensprogram bezárása előtt minden esetben kell zárni a megfelelő gomb megnyomásával, különben a kliensprogram újraindítása és a portok újranyitása során hibaüzenetet fogunk kapni egészen addig, amíg a portokat manuálisan le nem zárjuk, vagy újra nem indítjuk a számítógépet. A kliensprogram felhasználóbarátabbá tétele érdekében létrehoztam egy függvényt, mely a kliensprogram bezárása előtt automatikusan lezárja az aktív portokat, valamint megszünteti a vezetéknélküli hozzáférési pontot a megfelelő Windows parancsot tartalmazó batch fájl futtatásával.

A kliensprogram felhasználói felülete az 5.1. ábrán látható. A különböző interaktív objektumok engedélyezését úgy állítottam be, hogy a felhasználó ne küldhessen értelmetlen parancsokat a vezérlőegyégnak és a Raspberry Pi-nek. Például H.264 videó-adatfolyam küldésének indítása nem lehetséges, ha az aktív kommunikációs csatorna az ISM sávú rádió.

---

<sup>2</sup>Csak Windows-alapú számítógépeken működik.



**5.1. ábra.** A kliensprogram felhasználói felülete csatlakozás előtt és csatlakozás után

### 5.3. Soros kapcsolat kialakítása

A közvetlen, kábeles és az ISM sávú rádiós<sup>3</sup> összeköttetéshez soros porti kapcsolat ki-alakítása szükséges. A felhasználói felület jobb oldalán lévő REFRESH PORTS gombbal frissíthető listában kiválaszthatjuk, hogy melyik a laptophoz csatlakoztatott soros porti eszközökhöz szeretnénk csatlakozni, majd a RADIO CONNECT gomb megnyomása után a Matlab létrehoz és megnyit egy soros porti objektumot a képen feltüntetett soros porti beállításokkal. A soros port ezután egyszerű I/O műveletekkel írható és olvasható. A soros porton beérkező adatokat egyesével olvasom ki a soros port bemeneti pufferéből. A soros porti kapcsolódás a RADIO DISCONNECT gombbal szüntethető meg.

<sup>3</sup>Az USB-s ISM sávú rádió UART-on fogadja a kliensprogram által küldött vezérlő parancsokat. Lásd: 3.2.4. és 3.3.3. fejezet.

A soros port megnyitásakor, ha nincs felépített Wi-Fi-kapcsolat, akkor a kliensprogram az ISM sávú rádión keresztül automatikusan utasítja a vezérlőegységet a csatornaváltásra. A Wi-Fi-kapcsolat fennállása esetén a csatornaváltást manuálisan is megtehetjük a felhasználói felület jobb alsó felében lévő csatornakiválasztó gombokkal. Csatornaváltáskor a mobil adatgyűjtő egység megáll és alaphelyzetbe állítja a motorokat.

#### 5.4. UDP kapcsolat kialakítása

A Raspberry Pi-n futó parancsfeldolgozó Python script UDP protokollon keresztül kommunikál a kliensprogrammal (4.1. ábra). Az UDP kapcsolat létrehozásához meg kell adnunk az UDP kommunikációs portokat és a Raspberry Pi IP címét, majd a WIFI CONNECT gomb megnyomásakor a Matlab létrehoz és megnyit egy UDP típusú objektumot. Az UDP objektum szintén I/O műveletekkel írható és olvasható. Az UDP porton beérkező csomagot a CR/LF karakterek beérkezése után olvasom ki az UDP port bemeneti pufferéből, majd ellenőrzöm, hogy érvényes csomag érkezett-e. Az UDP kapcsolat a WIFI DISCONNECT gomb megnyomásával zárható le. Ha lezárjuk az UDP kapcsolatot, de az ISM sávú rádiós csatorna csatlakoztatva van, akkor a kliensprogram automatikusan utasítja a vezérlőegységet a csatornaváltásra.

#### 5.5. Élő kamerakép megjelenítése

A 4.3. fejezetben ismertettem az élő kamerakép küldésének két lehetséges módját. A két lehetőség közül a Wi-Fi-kapcsolat kialakítása után az 5.1. ábrán látható H.264 stream és JPEG stream gombokkal választhatunk, majd a CAMERA CONNECT gomb megnyomásával elindíthatjuk az élő kamerakép fogadását és megjelenítését. Az élő kamerakép fogadása és megjelenítése két különböző módon történik.

**JPEG tömörítésű képek fogadása és megjelenítése** A CAMERA CONNECT gomb megnyomásakor a Matlab létrehoz és megnyit egy TCP objektumot az előre beállított porttal, IP címmel és 5 MB-os TCP puffer mérettel. A szoftver ezután figyeli, hogy a TCP porton beérkezett-e a minden JPEG tömörítésű kép után küldött 10 bájt, ha igen beolvassa a TCP bemeneti pufferének tartalmát, majd dekódolja és megjeleníti a JPEG-képet a felhasználói felületen. A tesztelés során előfordult olyan eset, hogy a TCP puffer – a Matlab viszonylag lassú TCP puffer olvasása és JPEG-kép dekódolása, illetve megjelenítése miatt – több JPEG-képet is tartalmazott. Ekkor a megjelenítő függvény az összes JPEG-képet kiolvassa a TCP pufferból, de csak az utolsó képkockát jeleníti meg (a többit eldobja), így elkerülhető, hogy a TCP puffer megteljen és a képek elvesszenek. A szoftver továbbá folyamatosan számolja a képfrissítés gyorsaságát (FPS).

**H.264 kódolású videó-adatfolyam fogadása és megjelenítése** A Raspberry Pi által előállított és a netcat programmal TCP porton keresztül küldött H.264-es videó-adatfolyam fogadásához a laptopon szintén a netcat-et használom. A H.264-es videó-adatfolyam dekódolása és megjelenítése mplayer lejátszával történik. A CAMERA CON-

NECT gomb megnyomása után a Matlab lefuttat egy batch fájlt, ami a következő parancsot tartalmazza:

```
nc64.exe 192.168.137.2 5001 | mplayer.exe -fps 40 -demuxer h264es -
```

A parancs hatására a netcat a megadott IP című eszköztől a megadott porton keresztül fogadja a beérkező H.264-es videó-adatfolyamot, majd továbbítja az mplayernek dekódolásra és megjelenítésre. Az mplayer kétszer akkora frekvenciával jeleníti meg a képkockákat, mint amivel a Raspberry Pi küldi, mert az mplayer csak pár másodperccel később indul el, mint a netcat, viszont a netcat az mplayer indítása előtti képkockákat is továbbküldi az mplayernek megjelenítésre, így a netcat és az mplayer indítása között eltelt idő az élő kamerakép késleltetését okozná. Az mplayer jelenleg nincs a felhasználói felületre integrálva, így az élő kamerakép különálló ablakban jelenik meg.

## 5.6. Telemetria- és szenzoradatok feldolgozása

A telemetria- és szenzoradatok küldése és fogadása mind Wi-Fi, mind ISM sávú rádiós csatornán lehetséges. A mobil adatgyűjtő egység a telemetria- és szenzoradatokat csak a telemetria engedélyezése jelzőbit bebillentése után kezdi küldeni a kiválasztott csatornán. A telemetria- és szenzoradatok küldésének indítása vagy leállítása az 5.1. ábra bal felső sarkában található Telemetria gomb állapotától függ.

A vezérlőegység a telemetria- és szenzoradatokat feldolgozás nélkül küldi a kliensprogramnak. A beérkező telemetria- és szenzoradatokat tartalmazó csomagot a kliensprogram dolgozza fel és jeleníti meg a felhasználói felület bal oldalán lévő ablakokban.

## 5.7. A mobil adatgyűjtő egység vezérlése, konfigurálása

A mobil adatgyűjtő egység vezérlése és konfigurálása a csatornaválasztó gombok állapotától függően Wi-Fi-n vagy ISM sávú rádión történik. A jelenlegi szoftververzióban a vezérlő személyzet által kiadható vezérlő és konfigurációs parancsokat az 5.1. táblázat tartalmazza.

Parancs kiadásának módja	Parancs kódja	Parancs jelentése	Megjegyzés
W billentyű lenyomása	#DF000\r\n	Előremenet sebességének növelése	Vagy hátramenet sebességének csökkentése
S billentyű lenyomása	#DB000\r\n	Hátramenet sebességének növelése	Vagy előremenet sebességének csökkentése
A billentyű lenyomása	#SB000\r\n	Kormányszervó forgatása balra 1,86°-kal	Vezérlő impulzus hosszának növelése 5 ms-mal

D billentyű lenyomása	#SJ000\r\n	Kormányszervó forgatása jobbra 1,86°-kal	Vezérlő impulzus hosszának csökkentése 5 ms-mal
Q billentyű lenyomása	#LB010\r\n	Kamera forgatása 10°-kal balra	A léptetőmotor maximum 180°-ot fordul balra
E billentyű lenyomása	#LJ010\r\n	Kamera forgatása 10°-kal jobbra	A léptetőmotor maximum 180°-ot fordul jobbra
L billentyű lenyomása	#LC014\r\n	Léptetőmotor alaphelyzetbe állítása	–
Space billentyű, vagy STOP MOTORS gomb lenyomása	#SS000\r\n	Motorok leállítása és alaphelyzetbe állítása	Csatornaváltáskor automatikusan elküldi a parancsot
Áramkorlát beírása és enter lenyomása	#AK7.8\r\n	Áramkorlát változtatása ±7,8 A-re	Mértékegység: Amper
Sebességkorlát beírása és enter lenyomása	#SK013\r\n	Sebességkorlát változtatása 13 km/h-ra	Mértékegység: km/h
Tárgytávolság beírása és enter lenyomása	#UE036\r\n	DC motor leállítása, ha az akadály 36 cm-nél közelebb van	Mértékegység: cm
Telemetria gomb lenyomása	#TS001\r\n #TS000\r\n	Telemetria- és szenzoradatok küldésének engedélyezése, letiltása	–
Háttérvilágítás gomb lenyomása	#KV001\r\n #KV000\r\n	Háttérvilágítás manuális bekapcsolása, kikapcsolása	Ha a háttérvilágítás manuálisan be van kapcsolva, akkor az automatikus háttérvilágítás be- és kikapcsolása le van tiltva
4V8 enable gomb lenyomása	#KT001\r\n #KT000\r\n	4,8 V-os tápegység engedélyezése, letiltása	–

Wi-Fi gomb lenyomása	#RP001\r\n	Manuális csatornaváltás Wi-Fi-re	Raspberry Pi UDP portjainak megnyitása, vezérlőegység utasítása csatornaváltásra
ISM radio gomb lenyomása	#RP000\r\n	Manuális csatornaváltás ISM sávú rádióra	Raspberry Pi UDP portjainak lezárása, vezérlőegység utasítása csatornaváltásra
H.264 stream gomb lenyomása	#RP003\r\n	H.264 kódolású videó-adatfolyam küldésének indítása	Amennyiben fut, a JPEG tömörítésű képek küldésének leállítása
JPEG stream gomb lenyomása	#RP002\r\n	JPEG tömörítésű képek küldésének indítása	Amennyiben fut, a H.264 kódolású videó-adatfolyam küldésének leállítása
CAMERA DISCONNECT gomb lenyomása	#RP004\r\n #RP005\r\n	A kiválasztott JPEG tömörítésű vagy H.264 kódolású élő kamerakép küldésének leállítása	Elő kamerakép megjelenítésének leállítása
Reboot Raspberry Pi gomb lenyomása	#RP006\r\n	Raspberry Pi újraindítása	Automatikusan átvált ISM sávú rádiós csatornára
SYSTEM RESET gomb lenyomása	#SR000\r\n	Vezérlőegység és Raspberry Pi újraindítása	A kliensprogram is alapállapotba kerül

**5.1. táblázat.** A jelenlegi szoftververziójú kliensprogramban kiadható vezérlő és konfigurációs parancsok

## 6. fejezet

# Tesztek

Ebben a fejezetben ismertetem a mobil adatgyűjtő egység összeszerelése és a különböző rendszerelemek integrálása után elvégzett teszteket.

### 6.1. Funkcionális tesztek

A mobil adatgyűjtő egység fejlesztése során minden új beépítésre kerülő hardver- és szoftverelemet leteszteltem. A rendszer integrációja után ezeket a funkciókat újra leteszteltem. A funkcionális tesztek eredményeit a 6.1. táblázat tartalmazza.

Tesztelés	Teszt eredménye
3,3 V-os kimeneti feszültség ellenőrzése	Sikeres
4,8 V-os kimeneti feszültség ellenőrzése	Sikeres
Mikrokontroller programozhatóságának ellenőrzése	Sikeres
Mikrokontroller órajel, GPIO, ADC, DMA, Timer, UART, SPI, és NVIC funkcióinak ellenőrzése	Sikeres
Vezérlőegység és laptop közti UART kapcsolat működésének ellenőrzése	Sikeres
Vezérlőegység és laptop közti ISM sávú rádiós kapcsolat működésének ellenőrzése	Sikeres
Vezérlőegység és Raspberry Pi közti UART kapcsolat működésének ellenőrzése	Sikeres
Motorok ellenőrzése	Sikeres
DC motor-vezérlő hardverének és szoftverének ellenőrzése	Sikeres
Léptetőmotor-vezérlő hardverének és szoftverének ellenőrzése	Tesztelés alatt. A léptetőmotor bizonyos esetekben tovább fordul, mint $\pm 180^\circ$

Szervomotor-vezérlő hardverének és szoftverének ellenőrzése	Sikeress
GPS modul ellenőrzése	Sikeress
Kétirányú árammérő szenzor ellenőrzése	Sikeress
Sebességmérő szenzor ellenőrzése	Sikeress
Távolságmérő szenzor és ütközéselhárító szoftver ellenőrzése	Sikeress
Telemetria feszültségek mérésének ellenőrzése	Sikeress
Hőmérő szenzor ellenőrzése	Sikeress
Megvilágításmérő szenzor ellenőrzése	Sikeress
Háttérvilágítás automatikus be- és kikapcsolása a megvilágítás függvényében.	Sikeress
Áramkorlátozás betartásának ellenőrzése	Sikeress
Sebességkorlátozás betartásának ellenőrzése	Tesztelel alatt. A sebességkorlátozó algoritmus a sebességkorlát elérése előtt is csökkenti a sebességet
Raspberry Pi inicializálásának ellenőrzése	Sikeress
Wi-Fi adapter automatikus csatlakozása a hálózathoz statikus IP címmel	Sikeress
Parancskezelő Python script automatikus indításának ellenőrzése	Sikeress
RaspiCam működésének ellenőrzése	Sikeress
Élő kamerakép átvitelének ellenőrzése	Sikeress
A Raspberry Pi parancs végrehajtásainak ellenőrzése	Sikeress
A vezérlőegység parancs végrehajtásainak ellenőrzése	Sikeress
Telemetria- és szenzoradatok küldésének és megjelenítésének ellenőrzése	Sikeress
Kommunikációs csatorna váltásának ellenőrzése	Sikeress

**6.1. táblázat.** *A funkcionális tesztek eredményei*

## **6.2. Hatótávolság tesztelése**

A feladatkiírásban megkövetelt legalább 50 m-es hatótávolság elérésének tesztelését Wi-Fi, és ISM sávú rádió kommunikációs csatornán is elvégeztem kültéri és beltéri körülmenyek között. A kültéri teszteket a Goldmann György téri parkolóban, a beltéri teszteket a V1 épületben végeztem.

### **6.2.1. Beltéri hatótávolság tesztelése**

A beltéri hatótávolság tesztelését először a V1 épület 2. emeleti folyosóján a laptop és a mobil adatgyűjtő egység közti közvetlen rálátás biztosítása mellett végeztem. A tesztelés alatt mind a Wi-Fi, mind az ISM sávú rádiós kapcsolat stabil maradt, a mobil adatgyűjtő egység végig irányítható volt, és a telemetria- és szenzoradatok küldése, valamint az élő kamerakép küldése Wi-Fi-n sem akadozott.

A második beltéri teszt során a közvetlen rálátás nem volt biztosítva, a laptop a V1 épület 108-as laborjában, a mobil adatgyűjtő egység pedig a Wi-Fi hatótávolság mérése során az 1. emeleti folyosón az ISM sávú rádió hatótávolságának mérése során az 5. emeleti folyosón volt. A Wi-Fi-kapcsolat minden esetben körülbelül a folyosón található üvegajtó után 6-7 m-rel szakadt meg, ez körülbelül 20 m-t jelent légvonalban (két falon keresztül). Az ISM sávú rádiós kapcsolat csak abban az esetben szakadt meg, ha a mobil adatgyűjtő egység a lift vonalán túlra került.

### **6.2.2. Kültéri hatótávolság tesztelése**

A kültéri teszt során először a Wi-Fi, majd az ISM sávú rádió hatótávolságát teszteltem közvetlen rálátás mellett. A Wi-Fi-kapcsolat körülbelül 55-60 m-es távolság után szakadt meg, ekkor a mobil adatgyűjtő egység átállt ISM sávú rádiós kommunikációra. Az ISM sávú rádiós kapcsolatot a parkoló méreteiből adódóan 100 m-es távolságig tudtam letesztelni közvetlen rálátás mellett.

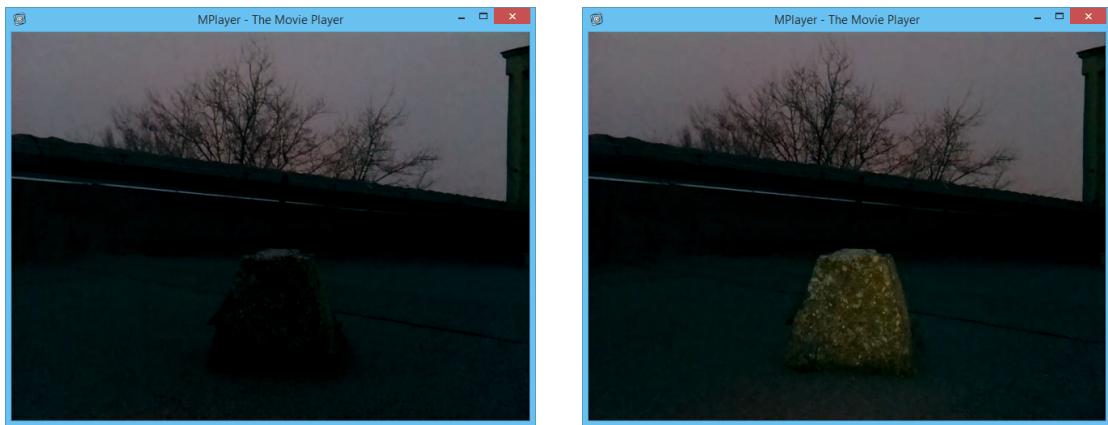
A második kültéri teszt során az ISM sávú rádiós összeköttetést teszteltem, ha a közvetlen rálátás a Goldmann György téri épület takarása miatt nincs biztosítva. Az ISM sávú rádiós kapcsolat az E épület előtt szakadt meg, ez körülbelül 150 m légvonalban.

## **6.3. Élő kamerakép tesztelése**

A teszt során a feladatkiírásban megkövetelt legalább 640x480 képpont felbontású, 10 kép/másodperces frissítésű élő kamerakép átvitelének biztosítását ellenőriztem 50 m-es távolság mellett. A H.264 kódolású élő kamerakép maradéktalanul teljesítette az elvárásokat, beleértve a képfrissítés gyorsaságát. A JPEG tömörítésű élő kamerakép képfrissítésének gyorsasága csupán 3-5 kép/másodperc volt 50 m-es távolságnál. Azonban 3-5 kép/másodperces frissítésű élő kamerakép is alkalmas a mobil adatgyűjtő egység távoli irányításához a mobil adatgyűjtő egység lassú haladása mellett.

## 6.4. Éjszakai teszt

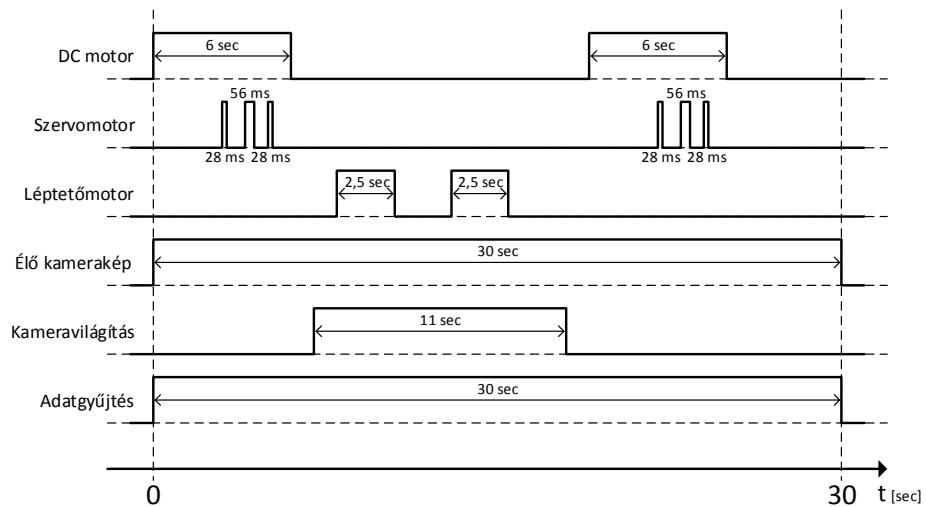
Az éjszakai teszt során a kamera háttérvilágításának működését teszteltem. A tesztet szabadtéren, sötétedéskor végeztem. A teszt alatt folyamatosan figyeltem az élő kameraképet és feljegyeztem a megvilágításmérő szenzor kimeneti feszültségét. Egy bizonyos ponton túl az élő kamerakép már túl sötétnek bizonyult a tájékozódáshoz, ekkor a megvilágításmérő szenzor kimeneti feszültsége 0,086 V, ami 70 luxnak felel meg. A háttérvilágítás bekapcsolásának küszöbértékét tehát 70 luxra, a háttérvilágítás kikapcsolásának küszöbértékét pedig 100 luxra állítottam. A 6.1. ábrán egy háttérvilágítás nélküli és egy háttérvilágítással készített kép látható 70 lux megvilágítású környezetben.



6.1. ábra. Élő kamerakép háttérvilágítás nélküli és háttérvilágítással

## 6.5. Üzemidő tesztelése

Az üzemidő becsléséhez a kliensprogram egy előre beprogramozott, valós felhasználási körülményeket szimuláló szekvenciának megfelelő utasítássorozatot küld a mobil adatgyűjtő egységeknek. A mobil adatgyűjtő egység a tesztelés alatt ezt az utasítássorozatot hajtotta végig periodikusan. Az utasítások és az utasítások elküldésének idődiagramja a 6.2. ábrán látható, a magas érték az adott funkció bekapcsolt állapotát jelzi.



**6.2. ábra.** Az utasítássorozat idődiagramja

Egy teljes periódus 30 másodpercig tart. Az élő kamerakép, illetve a telemetria- és szenzoradatok küldése az egész teszt alatt be volt kapcsolva. A periódus elején a mobil adatgyűjtő egység 6 másodpercig halad előre (körülbelül 2 km/h-s sebességgel), miközben a kormányt a középpállásból teljesen jobbra, majd jobbról teljesen balra, majd balról középre forgatja. 6 másodperc után a mobil adatgyűjtő egység megáll, bekapcsolja a kameravilágítást, majd elforgatja a kamerát 90°-kal jobbra, majd balra, végül kikapcsolja a kameravilágítást. A periódus végén a mobil adatgyűjtő egység hátramenetbe kapcsol (körülbelül 2 km/h-s sebességgel), miközben a kormányt a középpállásból teljesen balra, majd balról teljesen jobbra, majd jobbról középre forgatja, végül leállítja és alaphelyzetbe állítja a motorokat és vár 5 másodpercet. A valós körülmények szimulálásakor tehát az alábbi feltevéseket alkalmaztam:

- A DC motor az idő 40%-ában meg van hajtva.<sup>1</sup>
- A szervomotor az idő körülbelül 0,75%-ában van meghajtva.<sup>2</sup>
- A léptetőmotor az idő 16,7%-ában meg van hajtva (kamera forgatása a környezet felméréshöz).
- Az élő kamerakép az idő 100%-ában elérhető.
- A kamera háttérvilágítása az idő 36,7%-ában be van kapcsolva.
- A telemetria- és szenzoradatok gyűjtése és küldése az idő 100%-ában történik.

A tesztet eredetileg úgy terveztem, hogy addig járatom a mobil adatgyűjtő egységet, amíg az akkumulátor kapocsfeszültsége 9,6 V alá nem csökken, azonban 2 óra után a kapocsfeszültség még mindig 11,5 V volt, ezért a tesztet megszakítottam és a 3.2.1. fejezetben

<sup>1</sup>Fogyasztás szempontjából mindegy, hogy a DC motor előre, vagy hátra hajtja a mobil adatgyűjtő egységet.

<sup>2</sup>A szervomotor forgatási sebesség  $0,13 \frac{\text{sec}}{60^\circ}$ .

ismertetett akkumulátor által leadott energia-terhelt kapocsfeszültség görbéből (3.1. ábra) meghatároztam a 11,5 V-hoz tartozó töltöttségi állapotot, majd ebből megbecsültem a mobil adatgyűjtő egység üzemidejét. A 11,5 V-hoz 52%-os töltöttség tartozik, vagyis az üzemidő a fent említett felhasználási körülmények között körülbelül 4 óra.<sup>3</sup>

## 6.6. Tesztelés során tapasztalt hibák

A funkcionális tesztek elvégzése során két hibát tapasztaltam. Az egyik, hogy a léptetőmotor bizonyos esetekben az alaphelyzetbe állító parancs hatására többször is körbefordul. A léptetőmotor-vezérlő szoftverének módosítása után a hibát nem tapasztaltam, azonban a hiba okának egyértelmű azonosításához további tesztelés szükséges. A másik hiba, hogy a sebességkorlátozó algoritmus már a sebességkorlát elérése előtt csökkenti a sebességet. A hiba azonosításához és elhárításához a sebességmérő szenzor, illetve a sebességkorlátozó algoritmus további tesztelése szükséges.

A hatótávolság tesztelése során azt tapasztaltam, hogy ha a mobil adatgyűjtő egység a Wi-Fi hatótávolságán kívülre került, és a Wi-Fi-kapcsolat megszakadt, akkor a Wi-Fi hatótávolságon belülre kerülés után a Raspberry Pi Wi-Fi adaptere nem csatlakozott újra a laptop vezetéknélküli hozzáférési pontjához. Ennek oka, hogy a kapcsolat megszakadása után a Raspberry Pi wlan0 interfész letiltásra kerül. A hiba kiküszöbölése érdekében a parancsfeldolgozó Python scripthez hasonlóan a Raspberry Pi inicializálása után automatikusan elindul egy shell script, mely 30 másodpercenként lekérdezi a Wi-Fi-kapcsolat állapotát, ha a wlan0 interfész letiltott állapotba kerül, akkor a *sudo ifup --force wlan0* parancs kiadásával engedélyezi a wlan0 interfészt. A wlan0 interfész engedélyezése után, ha a mobil adatgyűjtő egység ismét a Wi-Fi hálózat hatósugarán belülre kerül, akkor a Wi-Fi adapter automatikusan újracsatlakozik. A Wi-Fi-kapcsolat megszakadása után a mobil adatgyűjtő egység kommunikációs csatornáját manuálisan kell átállítani az ISM sávú rádiós kapcsolatra.

Az üzemidő tesztelése egyben megbízhatósági tesztnek is tekinthető. A teszt 2 órás időtartama alatt a mobil adatgyűjtő egység két alkalommal megállt és csak a teljes rendszer újraindítása után indult újra. A piros indikátor LED villogása jelezte, hogy a vezérlőegység megkapta a vezérlő parancsokat, azonban a parancsok végrehajtása nem történt meg. A hiba azonosításához és elhárításához további tesztelése szükséges.

---

<sup>3</sup>Gyorsabb haladási sebességnél, vagy nem egyenletes terepen a DC motor nyomatéka, és így árama is nagyobb, ezt az üzemidő számításakor figyelembe kell venni.

## 7. fejezet

# Összefoglalás

### 7.1. A fejlesztés értékelése

A fejlesztés eredményeként elkészült a Függelék F.4. fejezetében látható mobil adatgyűjtő egység prototípusa. A mobil adatgyűjtő egységnek a feladatkiírásban szereplő következő kritériumokat kell teljesítenie:

- Legalább 50 m-es hatótávolságú vezetéknélküli kommunikációs összeköttetés kialakítása a mobil adatgyűjtő egység és egy hordozható eszköz között 2,4 GHz-es Wi-Fi-alapokon.
- Mérési adatok küldése, majd megjelenítése hordozható eszközön.
- Legalább 640x480 képpont felbontású, 10 kép/másodperces frissítésű élő kamerakép küldése, majd megjelenítése hordozható eszközön.
- A kamera forgatásának lehetővé tétele.
- A mobil adatgyűjtő egység helyváltoztatásának lehetővé tétele.
- Szenzorok csatlakoztatásának lehetővé tétele.

A mobil adatgyűjtő egység rendszerének teljes felépítése három logikailag elkülöníthető részegységből áll.<sup>1</sup> A mobil adatgyűjtő egység helyváltoztatását, illetve a kamera forgatását lehetővé tevő motorok vezérlése, valamint a telemetria- és szenzoradatok gyűjtése a vezérlőegység feladata, a telemetria- és szenzoradatok, illetve az élő kamerakép küldése és a vezérlő parancsok fogadása Wi-Fi-n keresztül a Raspberry Pi feladata, a mobil adatgyűjtő egység irányítása, a telemetria- és szenzoradatok feldolgozása és az élő kamerakép megjelenítése pedig a hordozható eszközön történik.

A mobil adatgyűjtő egység fejlesztésének első lépése a vezérlőegység hardverének megtervezése volt. A hardvertervezés első lépéseként kiválasztottam a feladatra alkalmas áramköri elemeket, majd Altium Designerben megterveztem a kapcsolási rajzot és a nyomtatott

---

<sup>1</sup>Fizikailag csak két részre különíthető el, a vezérlőegység és a Raspberry Pi egy távirányítós autó alvázára vannak rögzítve, a hordozható eszköz pedig egy laptop.

áramköri tervet. Bár a feladatkiírás nem indokolja, a motorvezérlők mellett egy UART-RS232 jelszintillesztő és egy ISM sávú rádiós adó-vevő áramkört is terveztem a vezérlőegységre. A UART-RS232 jelszintillesztővel kábeles összeköttetés valósítható meg a laptop és a vezérlőegység között, ennek a fejlesztés kezdeti szakaszában, a szoftverhibák felderítésénél volt jelentősége. Az ISM sávú rádiós kapcsolat kialakításának célja a hatótávolság növelése. Az ISM sávú rádiós kapcsolaton keresztül az élő kamerakép nem továbbítható. Az ISM sávú rádiós kapcsolat kialakításához egy pendrive méretű USB-s ISM sávú rádiós adó-vevő egységet is fejlesztettem, mely közvetlenül a laptopba dugható és UART protokollon keresztül kommunikál a kliensprogrammal. A vezérlőegységre ráterveztem egy távolságérzékelő szenzort ütközéselhárítás céljából, a mobil adatgyűjtő egység haladási sebességét és a DC motor kétirányú áramát mérő szenzort sebesség- és áramkorlátozás céljából (a későbbi autonóm felhasználás során a haladási sebesség és a DC motor áramának pontos ismerete sebesség- és áramszabályozásra is felhasználható), továbbá terveztem egy megvilágításmérő szenzort, mely alapján a kamera háttérvilágítása automatikusan be- és kikapcsol, egy hőmérő szenzort, és elhelyeztem egy helymeghatározó modult is a vezérlőegységen.

A hardver élesztése és tesztelése után Eclipse fejlesztői környezetben C nyelven megírtam a mikrokontrolleren futó beágyazott szoftvert. A szoftver az inicializálás után a kiválasztott kommunikációs csatornán fogadja, majd végrehajtja a felhasználó által a kliensprogramban kiadott vezérlő parancsokat. A mikrokontroller folyamatosan beolvassa a szenzorok adatait és továbbküldi a kiválasztott kommunikációs csatornán, ha a telemetria- és szenzoradatok küldése engedélyezve van.

A fejlesztés második lépése a Raspberry Pi beállítása és a Raspberry Pi-n futó parancsfeldolgozó Python script megírása volt. A Raspberry Pi a bekapcsolás után statikus IP címmel automatikusan csatlakozik a laptopon létrehozott vezetéknélküli hálózathoz, majd automatikusan elindítja a parancsfeldolgozó szoftvert. A Raspberry Pi UDP porton keresztül kommunikál a laptoppal és UART porton keresztül a vezérlőegységgel. A Raspberry Pi-nek kiadott parancsokkal elindíthatjuk, vagy leállíthatjuk a TCP porton küldött H.264 kódolású vagy JPEG tömörítésű élő kamerakép átvitelét.

A fejlesztés utolsó lépése a kliensprogram megírása volt. A kliensprogramot Matlabban, a Matlab grafikus felhasználói interfészének lehetőségeit kihasználva fejlesztettem. A kliensprogramban először csatlakoznunk kell a kiválasztott csatornán keresztül a mobil adatgyűjtő egységhez, majd különböző billentyűparancsokkal irányíthatjuk, illetve interaktív gombok megnyomásával konfigurálhatjuk a mobil adatgyűjtő egységet. A telemetria- és szenzoradatok feldolgozása és megjelenítése, valamint az élő kamerakép megjelenítése a kliensprogram felhasználói felületén történik.

A mobil adatgyűjtő egység részegységeinek integrálása után leteszteltem a komplett rendszert. Elsőként egy funkcionális tesztet végeztem, majd lemértem a mobil adatgyűjtő egység hatótávolságát kültéri és beltéri körülmények között mind Wi-Fi, mind ISM sávú rádiós csatorna használata mellett. Ezután leteszteltem, hogy az élő kamerakép szolgáltatásának minősége megfelel-e a feladatkiírásban megkövetelt paramétereknek, illetve éjszakai körülmények között a háttérvilágítás bekapcsolása mellett is leellenőriztem az élő kamerakép minőségét. Végül elvégeztem az üzemiidő becsléséhez szükséges tesztet. A tesztek során fény

derült néhány szoftveres hibára, ezek kijavítása után a mobil adatgyűjtő egység további tesztelését szükségesnek érzem a megbízhatóság növelésének érdekében.

A fejlesztés összeségében sikeresnek tekinthető, a mobil adatgyűjtő egység maradéktalanul teljesíti a feladatkiírásban kitűzött célokat, sőt a későbbi továbbfejlesztési lehetőségeket figyelembe véve a hardver képességei túl is teljesíti azokat.

## 7.2. Továbbfejlesztési lehetőségek, távlati célok

A feladatkiírásban szereplő kritériumok teljesítése után az alábbi fejlesztések megvalósítását tűztem ki célomul:

- A Raspberry Pi és a kliensprogram szoftverének módosítása, hogy ne statikus IP címeket használjanak, hanem az indulás után az ISM sávú rádión keresztül küldjék el, a DHCP által kiosztott IP címüket, ezáltal a mobil adatgyűjtő egység és a laptop közti Wi-Fi-kapcsolat bármilyen vezetéknélküli hozzáférési ponton keresztül kialakítható lenne.
- A vezérlőegységhez további szenzorok (giroszkóp, iránytű, távolságmérő szenzorok a mobil adatgyűjtő egység hátuljára és oldalára is ütközéselhárítás céljából, stb.) csatlakoztatása.
- Audió adatok átvitelének megvalósítása, így az élő kamerakép mellett a feltérképezni kívánt területen élő hangfelvételt is készíthetünk.
- Napelemes töltőrendszer fejlesztése, mely működés közben és a mobil adatgyűjtő egység kikapcsolt állapotában is képes az akkumulátor töltésére, így nem szükséges külön töltésvezérlő áramkör és hálózati konnektor az akkumulátor töltéséhez. Napelemes töltéssel a mobil adatgyűjtő egység üzemidejének hossza is tovább növelhető.
- Kamerakép-alapú nyomkövető algoritmus fejlesztése autonóm helyváltoztatás céljából.
- GPS koordináta-alapú autonóm navigáció megvalósítása (földi drón).
- A Wi-Fi és az ISM sávú rádiós kapcsolat helyett egy legalább 5 Mbps adatsebességű, legalább 200 m hatótávolságú rádiós összeköttetés megvalósítása a telemetria- és szenzoradatok, a vezérlő parancsok és az élő kamerakép átviteléhez, valamint egy olyan kameramodul használata, amely digitális interfészen keresztül, közvetlenül összeköthető a mikrokontrollerrel, ezáltal növelve a rendszer integráltságát (a Raspberry Pi kihagyható a kommunikációs útból, vagyis további használata nem szükséges).

# Fogalomtár

CRC	Cyclic Redundancy Check / Körkörös redundancia ellenőrzés
CSI-2	Camera Serial Interface / Soros kamera-illesztő
DC	Direct Current / Egyenáram
DMA	Direct Memory Access / Közvetlen memóriahezáférés
FIFO	First In-First Out / Érkezési sorrendnek megfelelő feldolgozás
FPS	Frame Per Second / Másodpercenkénti képfelvétel
FSK	Frequency Shift Keying / Frekvenciabillentyűzés
GPIO	General Purpose Input Output / Általános felhasználású be- és kimenet
GPS	Global Positioning System / Globális helymeghatározó rendszer
GPU	Graphics Processing Unit / Grafikus feldolgozóegység
GUI	Graphical User Interface / Grafikus felhasználói felület
IC	Integrated Circuit / Integrált áramkör
IP	Internet Protocol / Internet protokoll, az eszköz hálózati címe
ISM	Industrial, Scientific, Medical radio bands / Ipari, tudományos, orvosi rádiósávok
JPEG	Joint Photographic Experts Group / Veszeséges képtömörítési formátum
LED	Light Emitting Diode / Fénykibocsátó dióda
MOSFET	Metal–Oxide Semiconductor Field-Effect Transistor / Fém-oxid-alapú félvezető tervezérlésű tranzisztor
NMEA	National Marine Electronics Association / Nemzeti Tengerészeti Elektronikai Szövetség által kifejlesztett szabványos üzenetformátum
NVIC	Nested Vector Interrupt Controller / Beagyazott-vektoros megszakításvezérlő
PWM	Pulse Width Modulation / Impulzusszélesség-moduláció
SPI	Serial Peripheral Interface / Soros periféria-illesztő
SSH	Secure Shell / Egy helyi és egy távoli eszköz közötti biztonságos csatorna kiépítéséhez használt protokoll
SWD	Serial Wire Debug / Soros programozófelület
TCP	Transmission Control Protocol / Megbízható üzenetküldési protokoll, a fizikai kapcsolat megszakadásáig garantálja a csomagok átvitelét
UART	Universal Asynchronous Receiver-Transmitter / Univerzális Soros adó-vevő
UDP	User Datagram Protocol / Gyors üzenetek küldéséhez használt protokoll, amikor a sebesség fontosabb a megbízhatóságnál
USB	Universal Serial Bus / Univerzális Soros busz

# Irodalomjegyzék

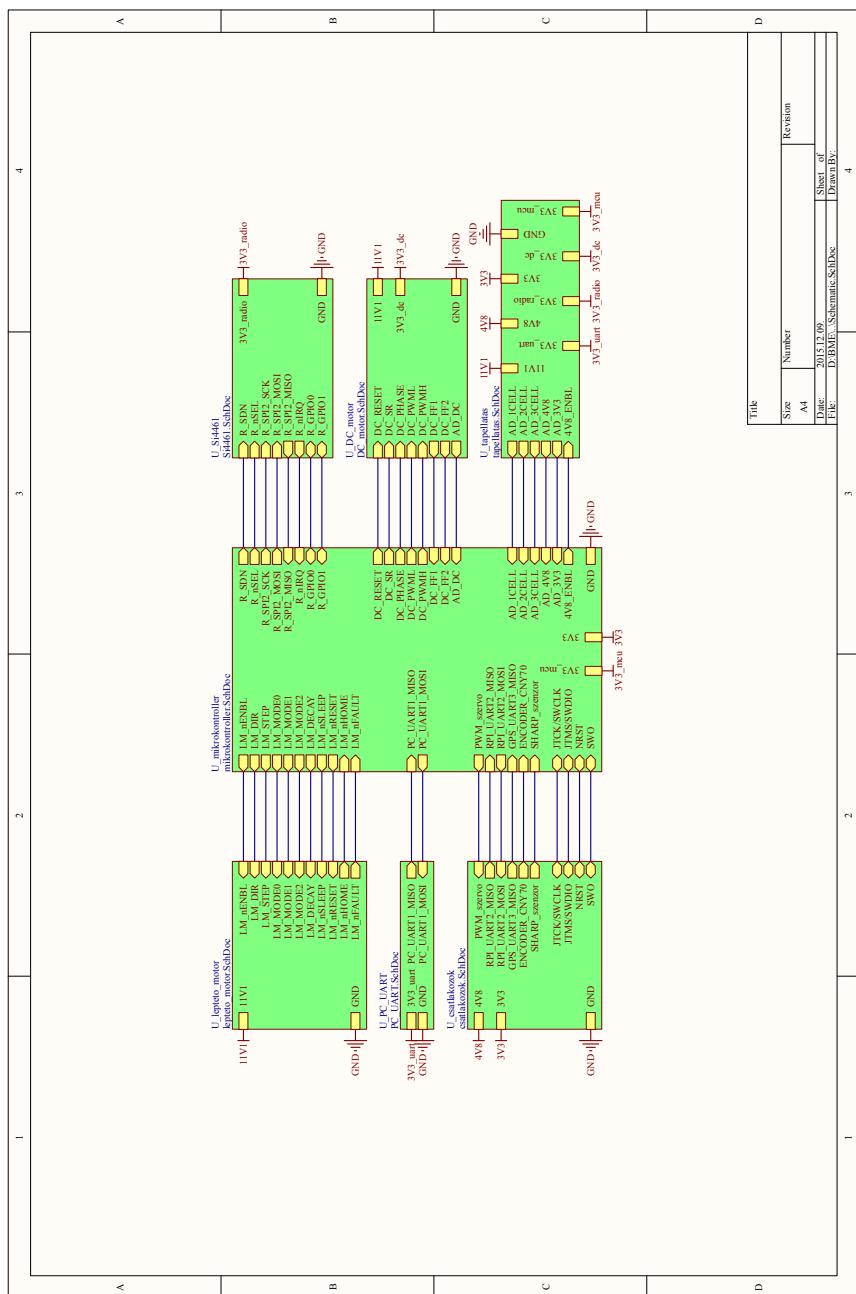
- [1] Benkő Tiborné-Benkő László-Tóth Bertalan. *Programozzunk C nyelven!* Computer-Books, Budapest, 2009.
- [2] Alex Bradbury and Ben Everard. *Learning Python with Raspberry Pi*. WILEY, 2014.
- [3] SHARP Corporation. *GP2Y0A21YK0F Distance Measuring Sensor Unit*, December 2006.
- [4] Robert W. Erickson and Dragan Maksimović. *Fundamentals of Power Electronics*. Kluwer Academic Publishers, University of Colorado, Boulder, Colorado, second edition, 2004.
- [5] Python Software Foundation. *Low-level networking interface*, December 2015. <https://docs.python.org/3.4/library/socket.html>.
- [6] RASPBERRY PI FOUNDATION. *CAMERA HARDWARE AND SOFTWARE FEATURES*, December 2015. <https://www.raspberrypi.org/documentation/hardware/camera.md>.
- [7] RASPBERRY PI FOUNDATION. *RASPBERRY PI CAMERA MODULE*, December 2015. <https://www.raspberrypi.org/documentation/raspbian/applications/camera.md>.
- [8] RASPBERRY PI FOUNDATION. *Raspbian Wheezy*, December 2015. <https://www.raspberrypi.org/downloads/raspbian/>.
- [9] Szabó Zoltán-Kovács Viktor-Csorvási Gábor. *Eclipse alapú fejlesztőkörnyezet összeállítása STM32F4Discovery fejlesztőkártyához*. Budapesti Műszaki és Gazdaság tudományi Egyetem, Automatizálási és Alkalmazott Informatikai Tanszék, October 2014.
- [10] GPSINFORMATION.ORG. *NMEA data*, December 2015. <http://www.gpsinformation.org/dale/nmea.htm#nmea>.
- [11] Texas Instruments. *LMx58-N Low-Power, Dual-Operational Amplifiers*, January 2000.
- [12] Texas Instruments. *INA152, Single-Supply DIFFERENCE AMPLIFIER*, January 2001.
- [13] Texas Instruments. *Calculating Motor Driver Power Dissipation*, February 2012.

- [14] Texas Instruments. *TPS54335, TPS54336, 4.5V to 28V Input, 3A Output, Synchronous Step Down SWIFT Converter*, May 2013.
- [15] Texas Instruments. *DRV8825 Stepper Motor Controller IC*, July 2014.
- [16] Texas Instruments. *INA19x Current Shunt Monitor ?16 V to +80 V Common-Mode Range*, January 2014.
- [17] Texas Instruments. *MAX3232E 3-V to 5.5-V Multichannel RS-232 Line Driver/Receiver With +-15-kV IEC ESD Protection*, January 2014.
- [18] Dave Jones. *Picamera*, December 2015. <http://picamera.readthedocs.org/en/release-1.10/index.html>.
- [19] Silicon Labs. *AN625, Si446X API DESCRIPTIONS*, February 2012.
- [20] Silicon Labs. *Si4464/63/61/60, HIGH-PERFORMANCE, LOW-CURRENT TRANSCEIVER*, December 2012.
- [21] Silicon Labs. *AN626, PACKET HANDLER OPERATION FOR Si446X RFICS*, July 2013.
- [22] Silicon Labs. *CP2102/9, SINGLE-CHIP USB TO UART BRIDGE*, December 2013.
- [23] Silicon Labs. *AN632, WDS USERS GUIDE FOR EZRADIOOPRO DEVICES*, 2014.
- [24] Silicon Labs. *CP210x USB to UART Bridge VCP Drivers*, December 2015. <https://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx>.
- [25] Chris Liechti. *pySerial*, December 2015. <https://pyserial.readthedocs.org/en/latest/pyserial.html>.
- [26] MathWorks. *GUI Building*, December 2015. <http://www.mathworks.com/help/matlab/gui-development.html>.
- [27] Allegro MicroSystems. *A3941, Automotive Full Bridge MOSFET Driver*, August 2011.
- [28] International Rectifier. *IRL7833PbF, IRL7833SPbF, IRL7833LPbF HEXFET Power MOSFET*, May 2004.
- [29] NXP Semiconductors. *KTY84 series, Silicon temperature sensors*, May 2008.
- [30] OSRAM Opto Semiconductors. *SFH 203, SFH 203FA Silicon PIN Photodiode*, October 2011.
- [31] Vishay Semiconductors. *CNY70, Reflective Optical Sensor with Transistor Output*, June 2006.
- [32] STMicroelectronics. *UM1525 User manual, STM32F0DISCOVERY Discovery kit for STM32 F0 microcontroller*, May 2012.

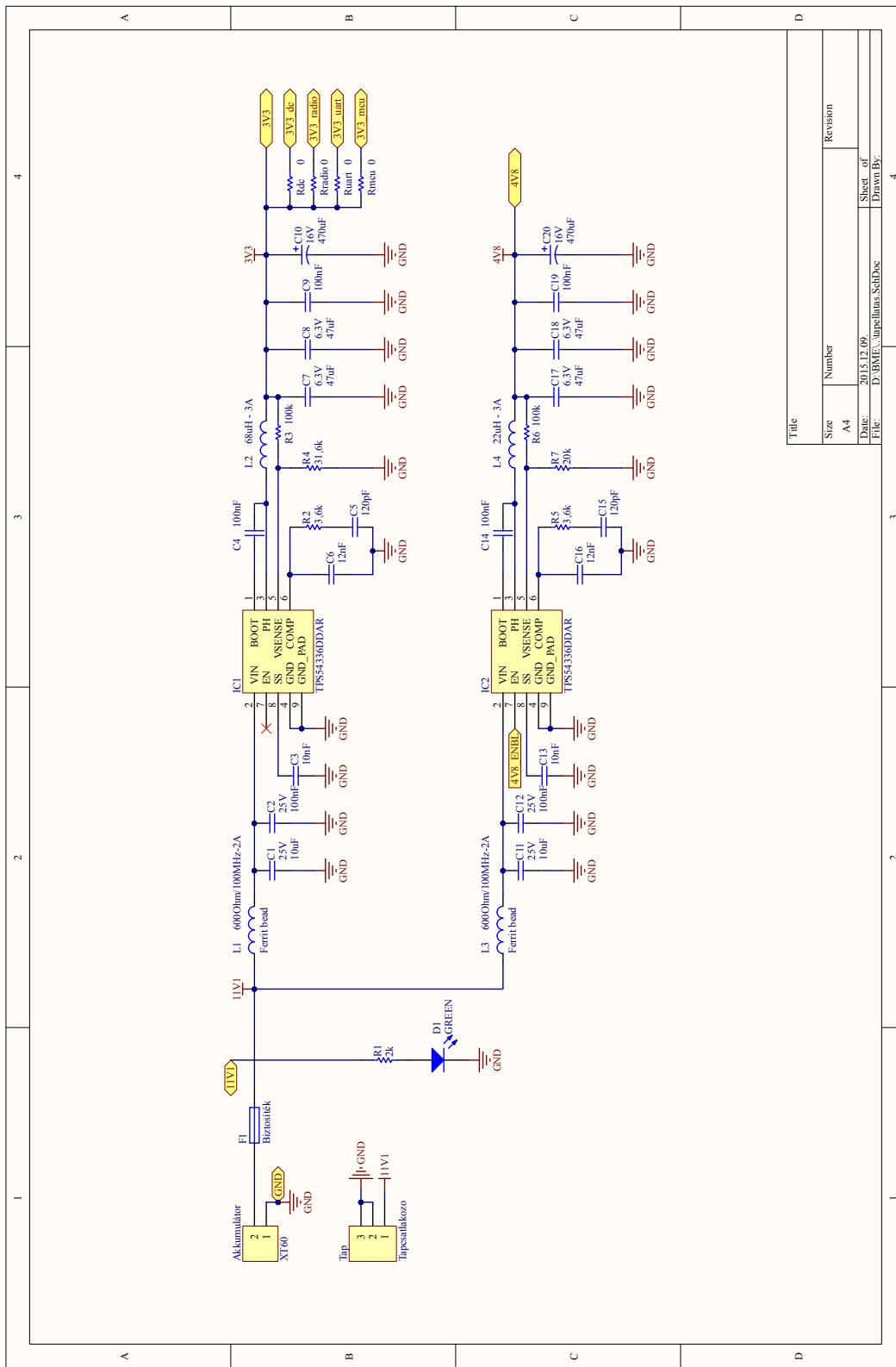
- [33] STMicroelectronics. *PM0056 Programming manual, STM32F10xxx/20xxx/21xxx/L1xxxx Cortex-M3 programming manual*, May 2013.
- [34] STMicroelectronics. *STM32F103x4, STM32F103x6, Low-density performance line, ARM-based 32-bit MCU*, May 2013.
- [35] STMicroelectronics. *RM0008 Reference manual, STM32F101xx, STM32F102xx, STM32F103xx, STM32F105xx and STM32F107xx advanced ARM-based 32-bit MCUs*, June 2014.
- [36] STMicroelectronics. *STM32F105xx, STM32F107xx, Connectivity line, ARM-based 32-bit MCU*, March 2014.
- [37] tyco/Electronics. *GPS Receivers A1029 Users Manual*, February 2006.
- [38] Eben Upton and Gareth Halfacree. *Raspberry Pi User Guide*. WILEY, second edition, 2014.

# Függelék

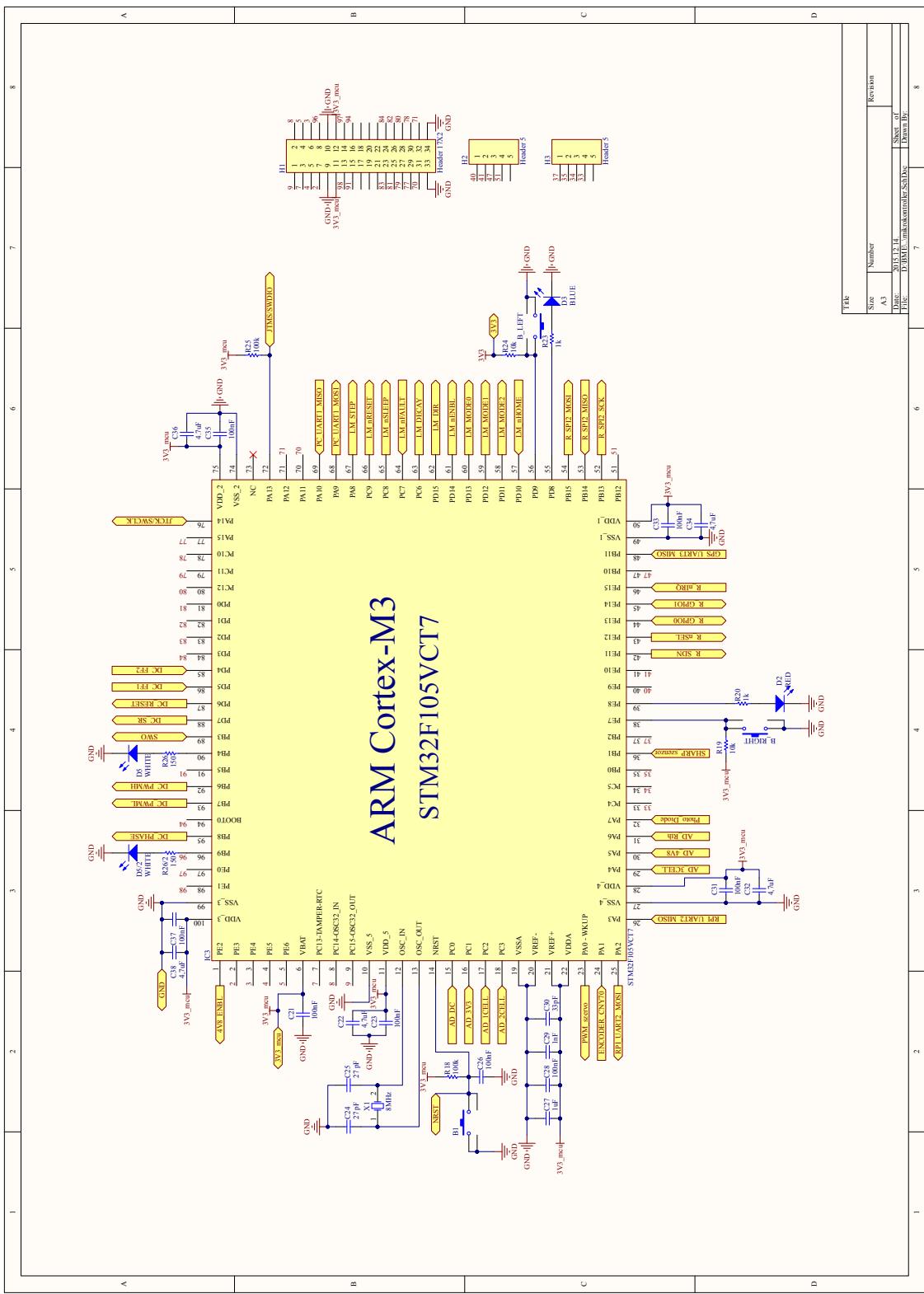
#### F.1. A vezőrlőegység kapcsolási rajza és nyomtatott áramköri terve



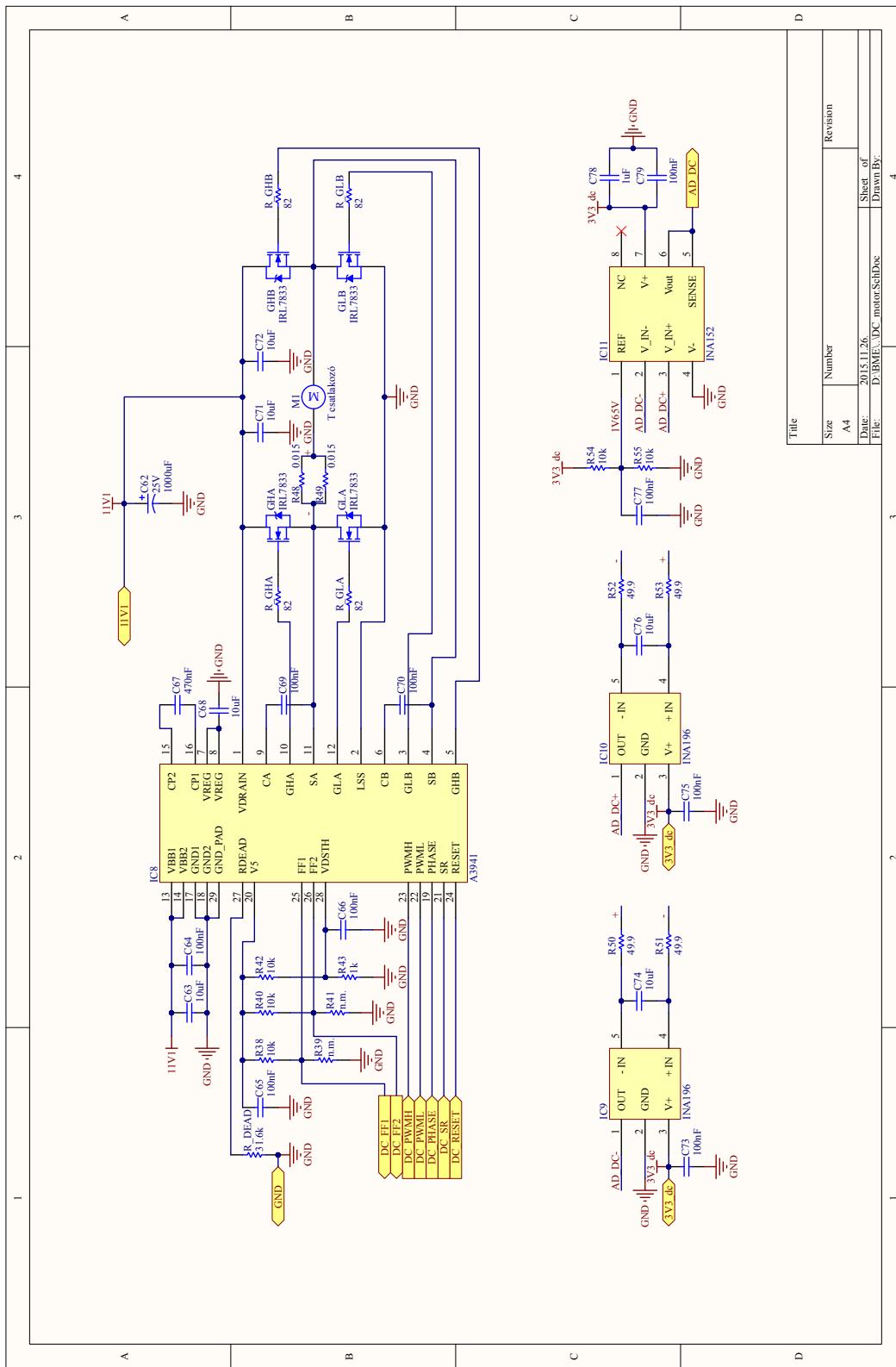
**F.1.1. ábra.** A kapcsolási rajz hierarchikus ábrája



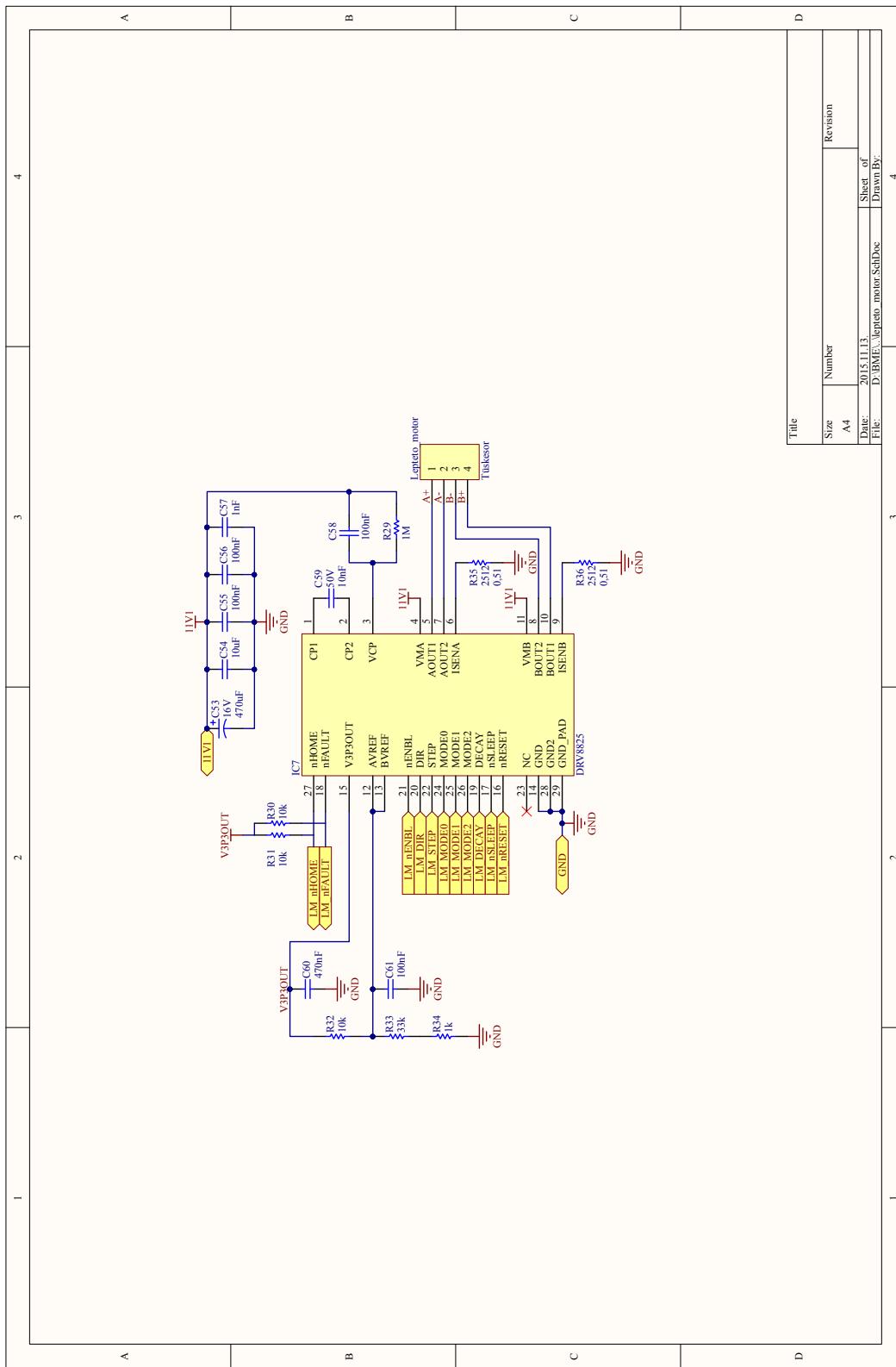
**F.1.2. ábra.** Az energiaellátó rendszer kapcsolási rajza



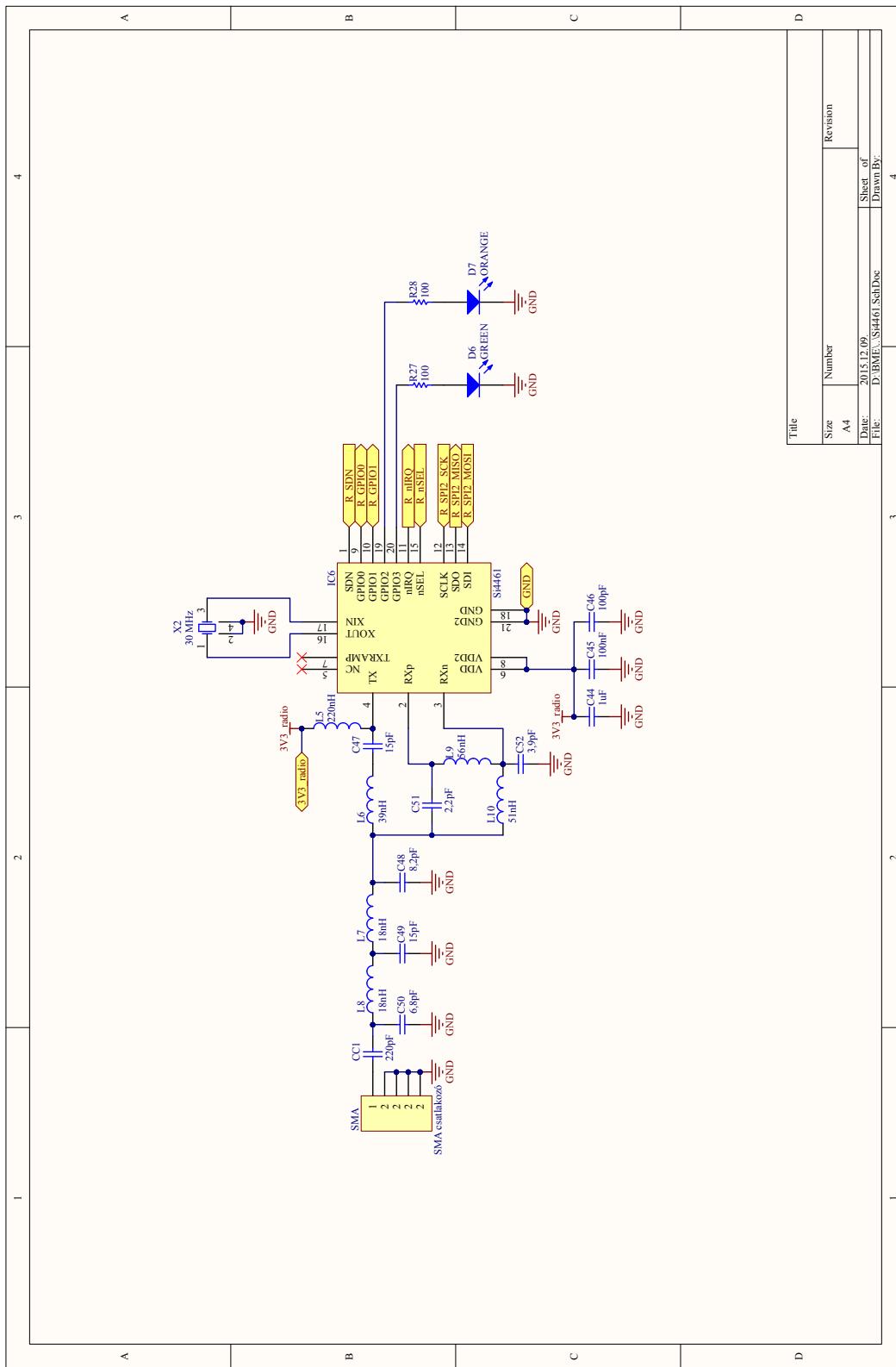
F.1.3. ábra. A mikrokontroller kapcsolási rajza



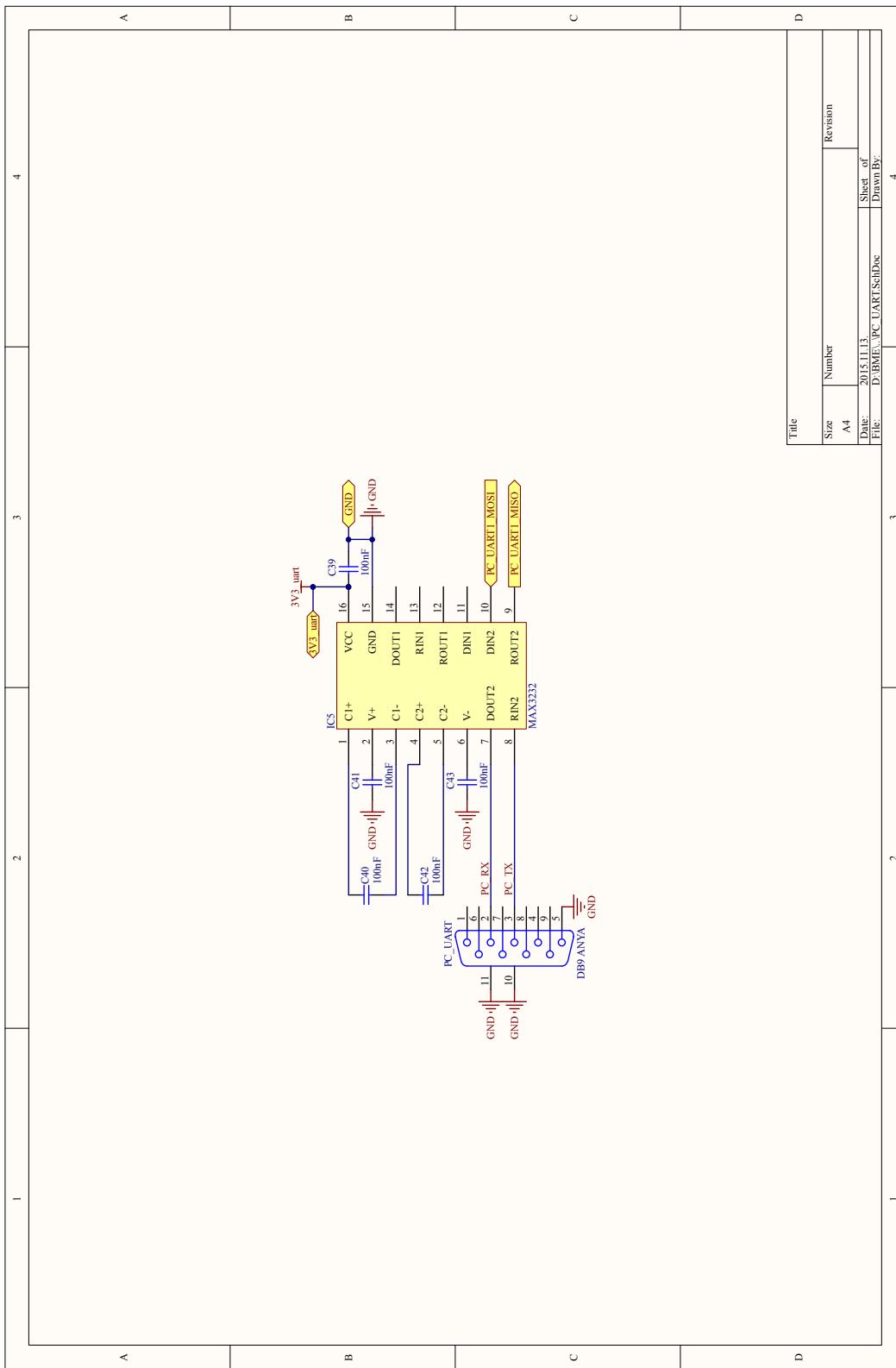
**F.1.4. ábra.** A DC motor-vezérlő és a DC motor árammérő szenzorának kapcsolási rajza



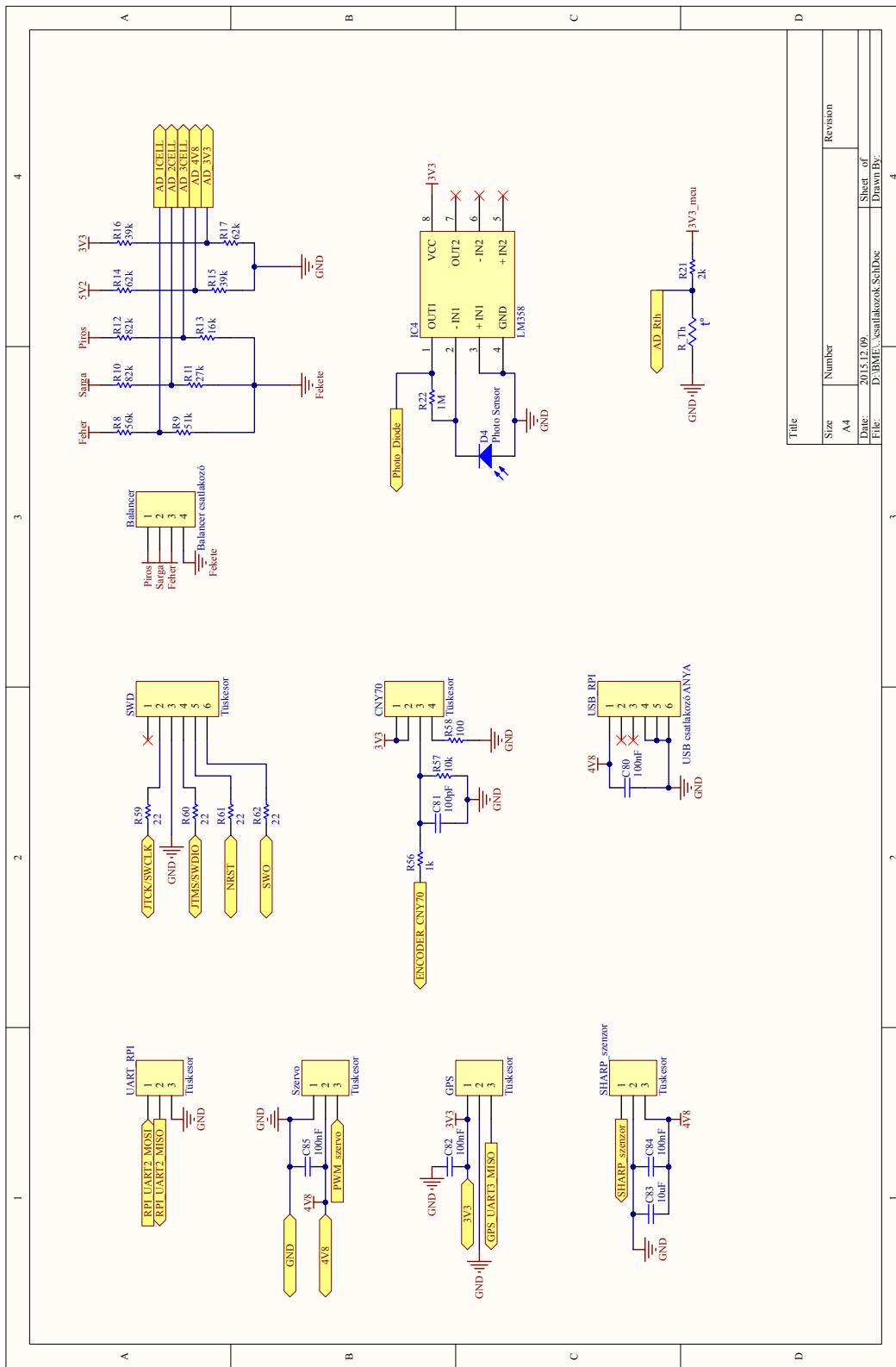
**F.1.5. ábra.** A léptetőmotor-vezérlő kapcsolási rajza



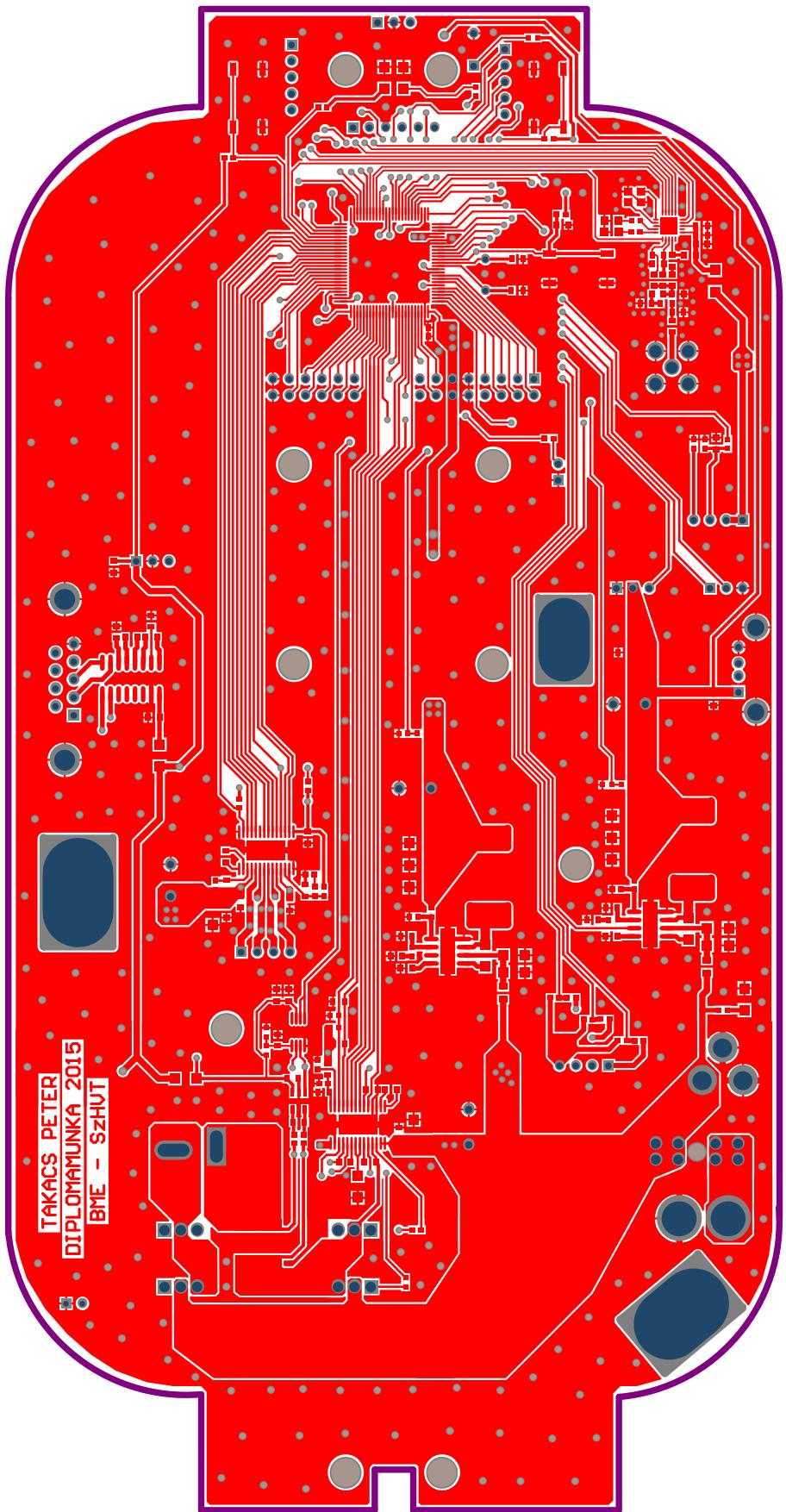
**F.1.6. ábra.** Az ISM sávú rádiós adó-vevő áramkör kapcsolási rajza



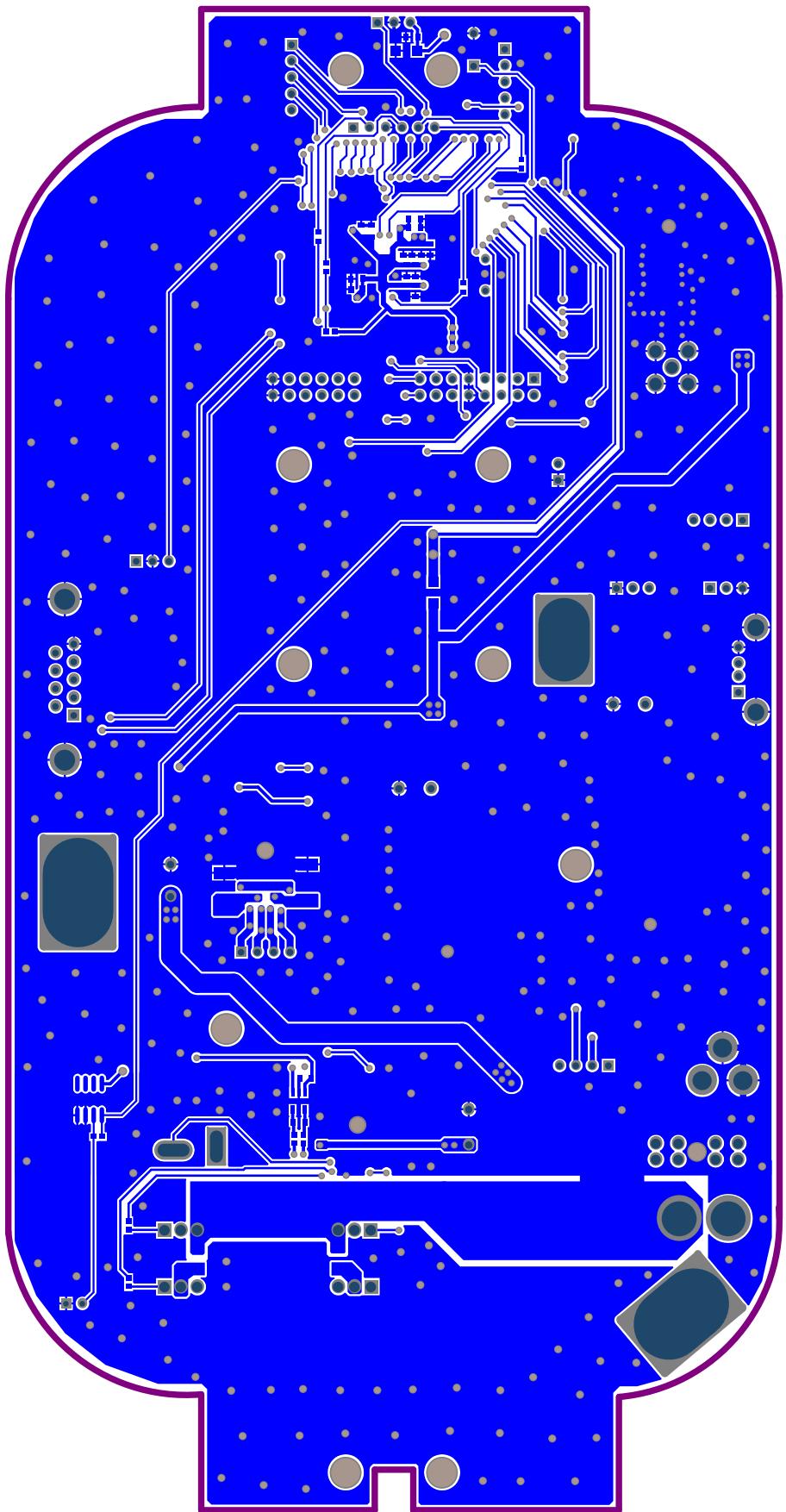
**F.1.7. ábra.** Az *UART* jelszintillesztő kapcsolási rajza



F.1.8. ábra. A szenzorok és a csatlakozók kapcsolási rajzai

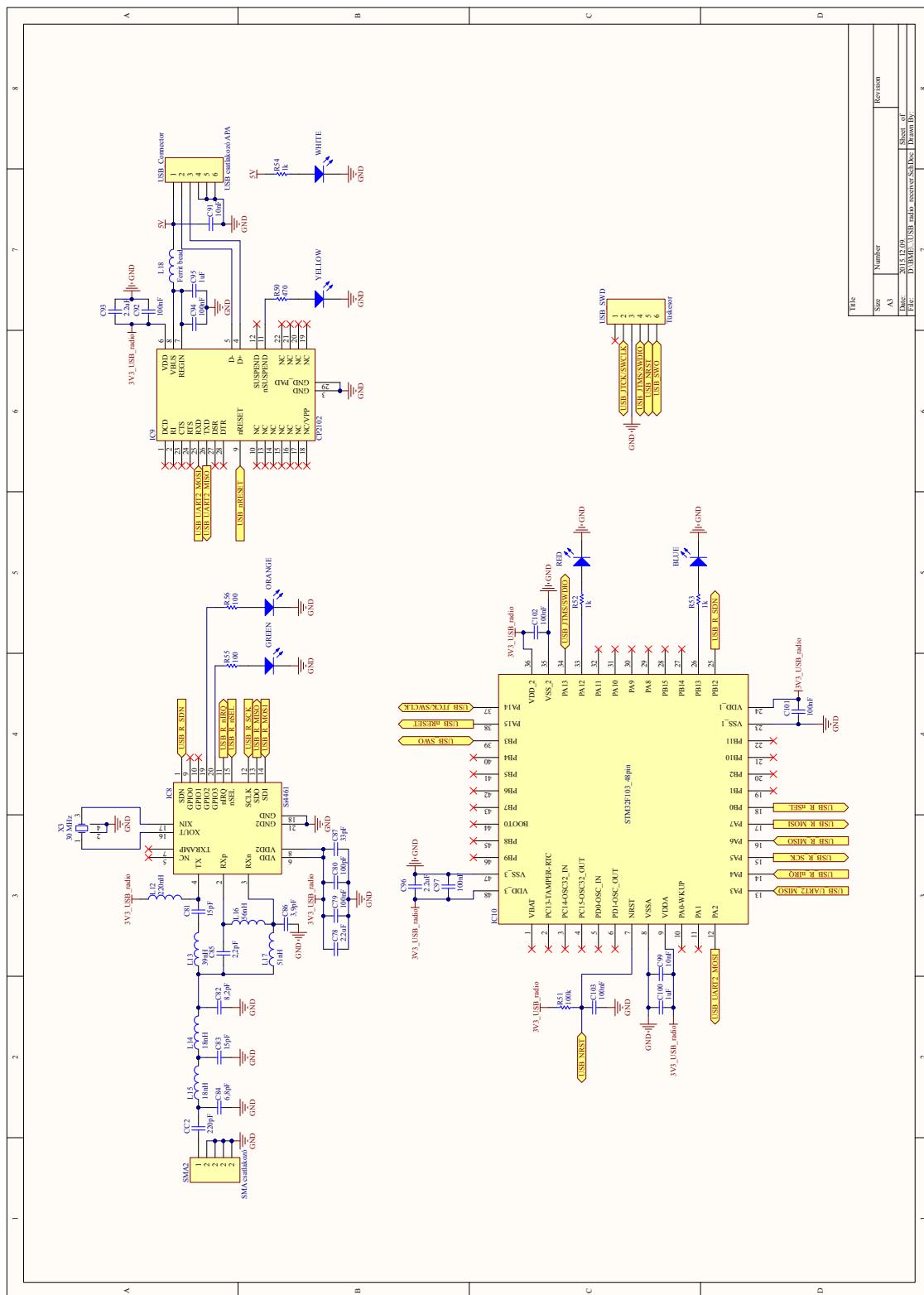


F.1.9. ábra. A vezérlőegység nyomtatott áramköri tervének felső rétege

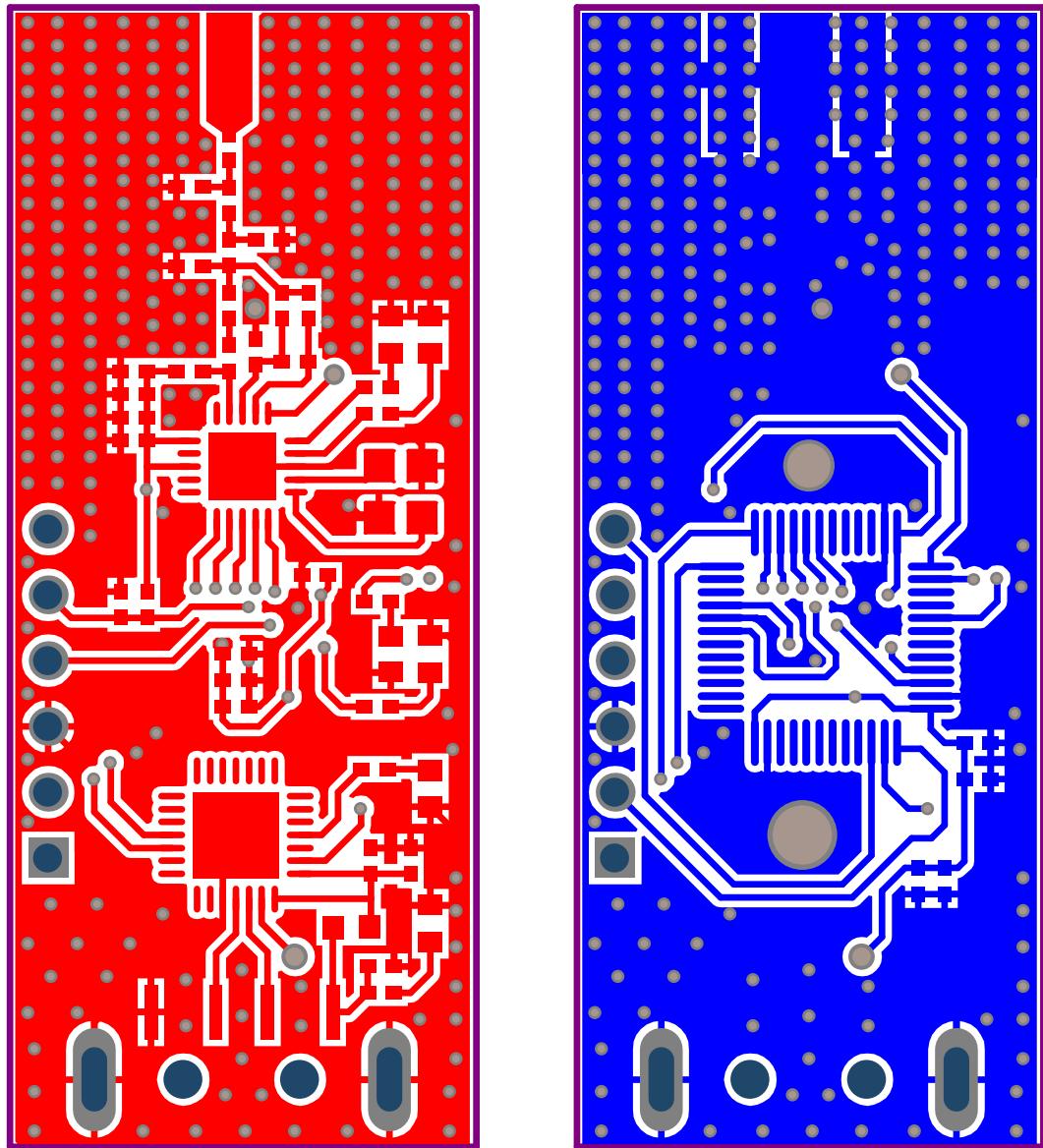


F.1.10. ábra. A vezérlőegység nyomtatott áramköri tervének alsó rétege

**F.2. Az USB-s ISM sávú rádiós adó-vevő egység kapcsolási rajza és nyomtatott áramkörű terve**



**F.2.1. ábra.** Az USB-s ISM sávú rádiós adó-vevő egység kapcsolási rajza

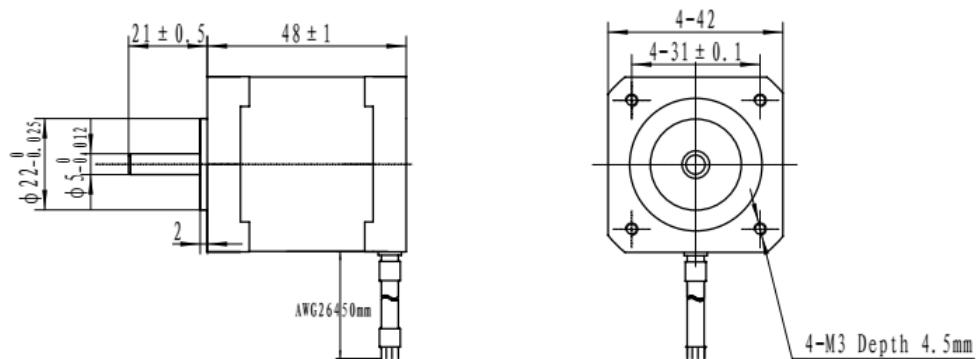


**F.2.2. ábra.** Az USB-s ISM sávú rádiós adó-vevő egység nyomtatott áramköri tervének felső és alsó rétege

F.3. A léptetőmotor paraméterei

# 42HS48-1504A

## MÉRETEK



## MOTORPARAMÉTEREK

FÁZIS	LÉPÉS SZÖG	FESZÜLTSÉG	ÁRAM/FÁZIS	ELLENÁLLÁS/FÁZIS	INDUKCIÓ/FÁZIS	NYOMATÉK	ROTOR INERCIA	SÚLY
		V	A	Ω	mH	Nm	gcm <sup>2</sup>	Kg
2	1,8°	4,2	1,5	2,8	4,8	0,56	68	0,35

## A MOTOR ÉS VEZÉRLŐ BEKÖTÉSE

### BIPOLÁRIS

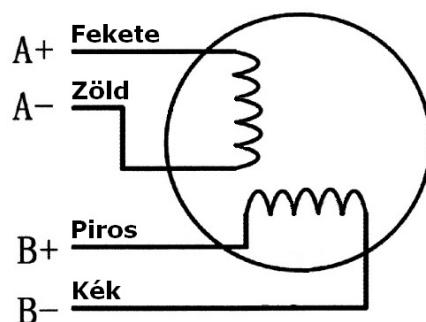
FÁZIS ÁRAM = 1,5A  
NYOMATÉK = 0,56Nm

1. TEKERCS  
 • FEKETE  
 • ZÖLD

A+  
A-

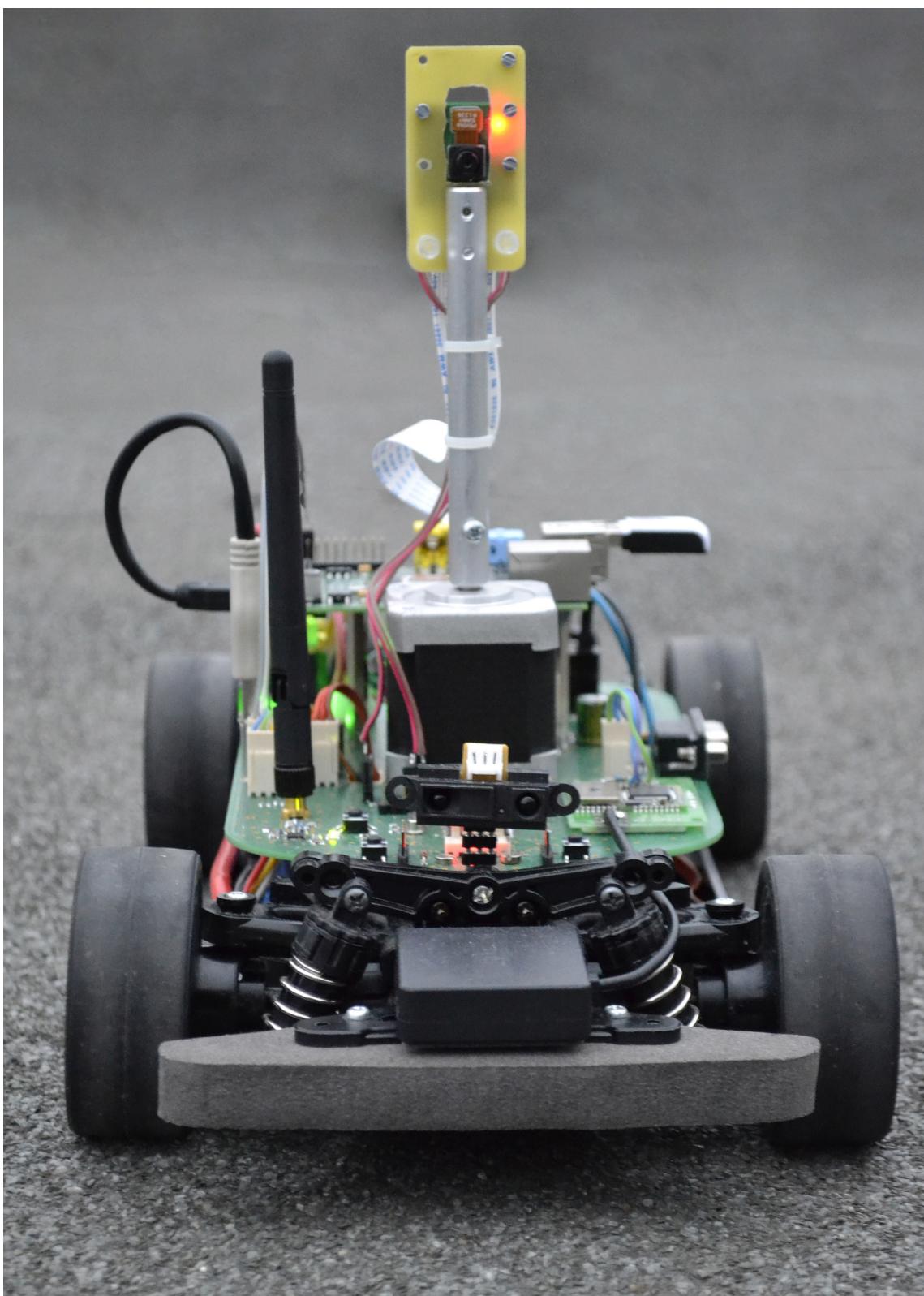
2. TEKERCS  
 • PIROS  
 • KÉK

B+  
B-

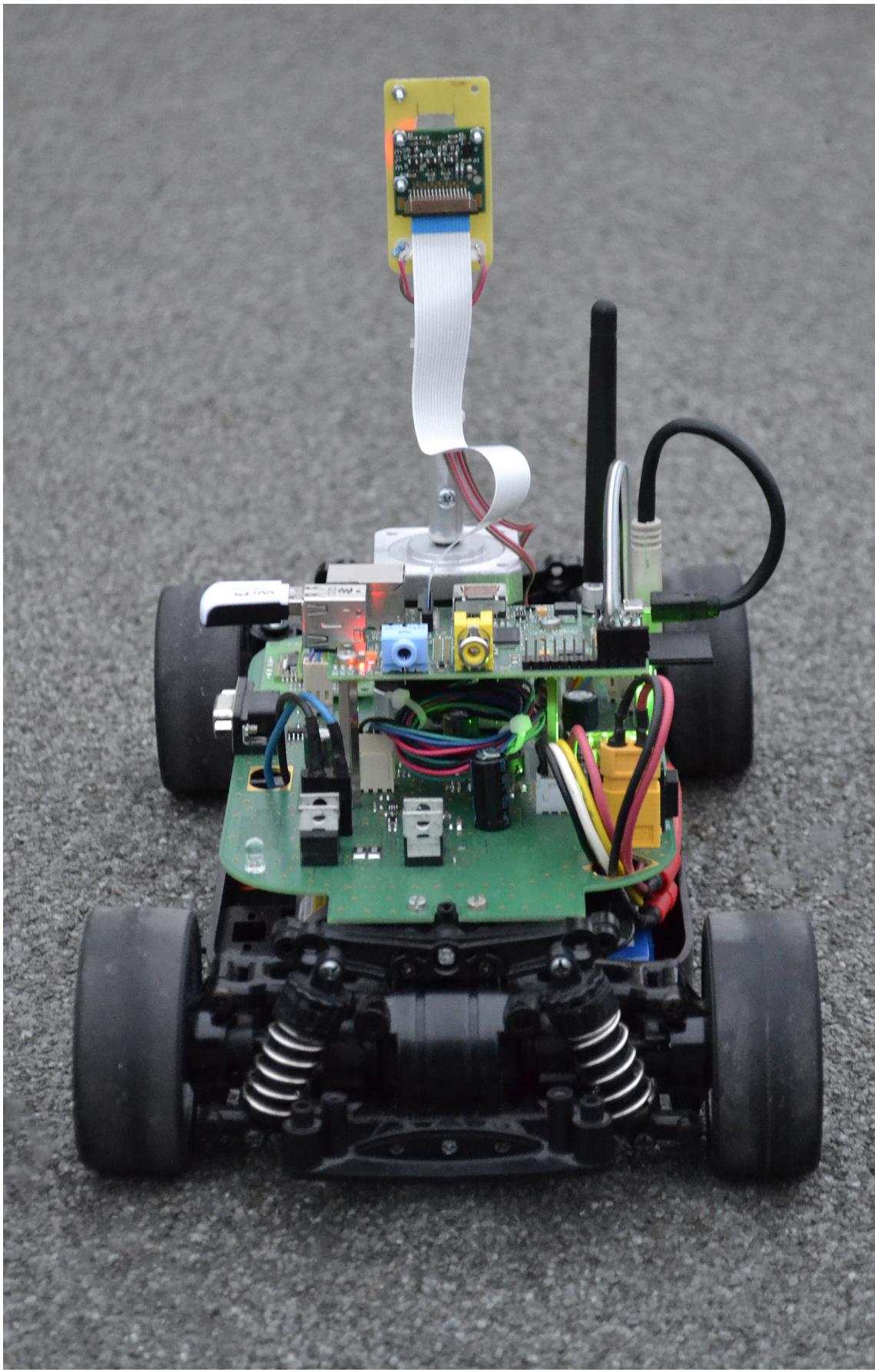


F.3.1. ábra. A léptetőmotor paraméterei

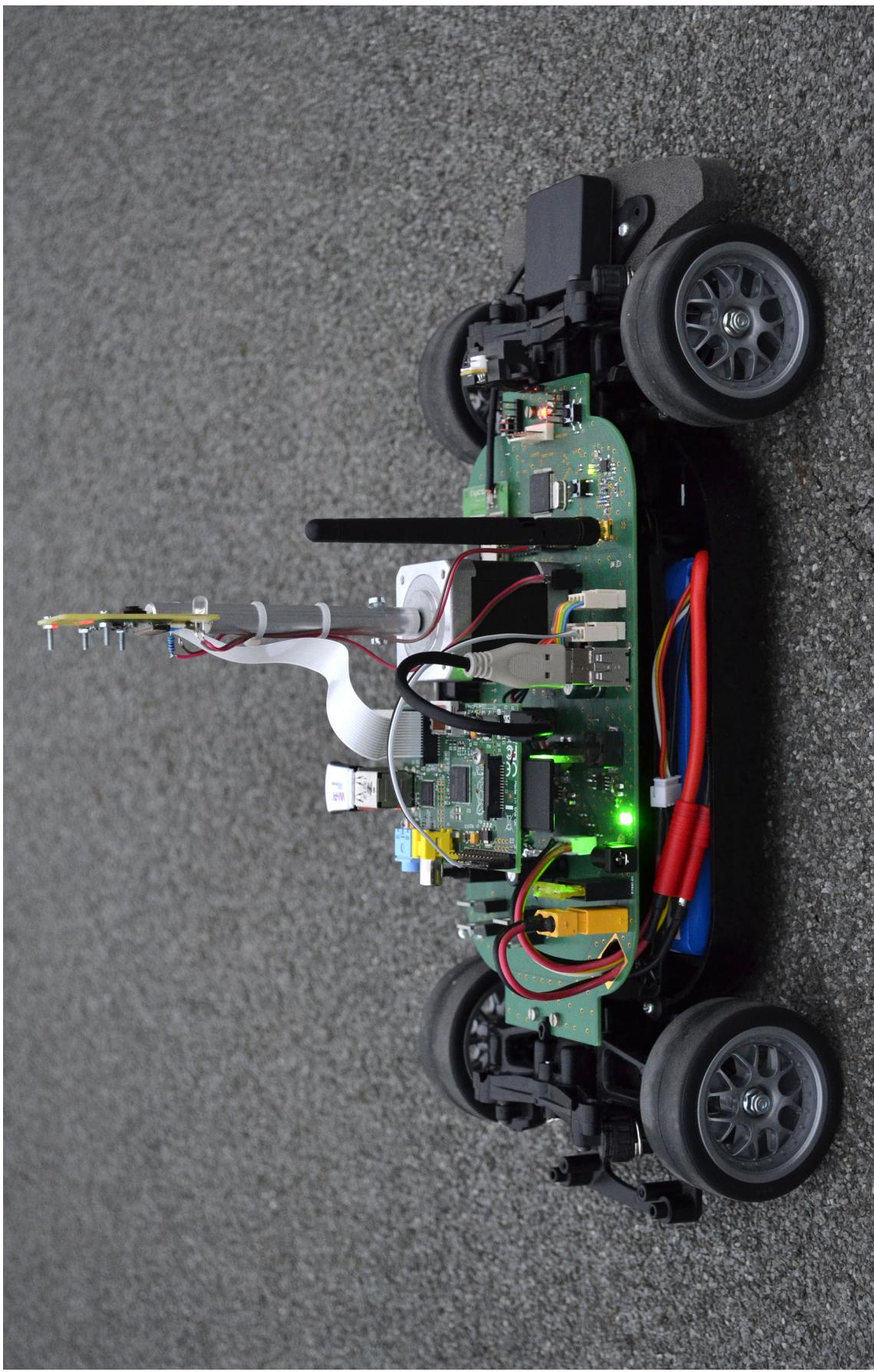
F.4. A mobil adatgyűjtő egység prototípusa



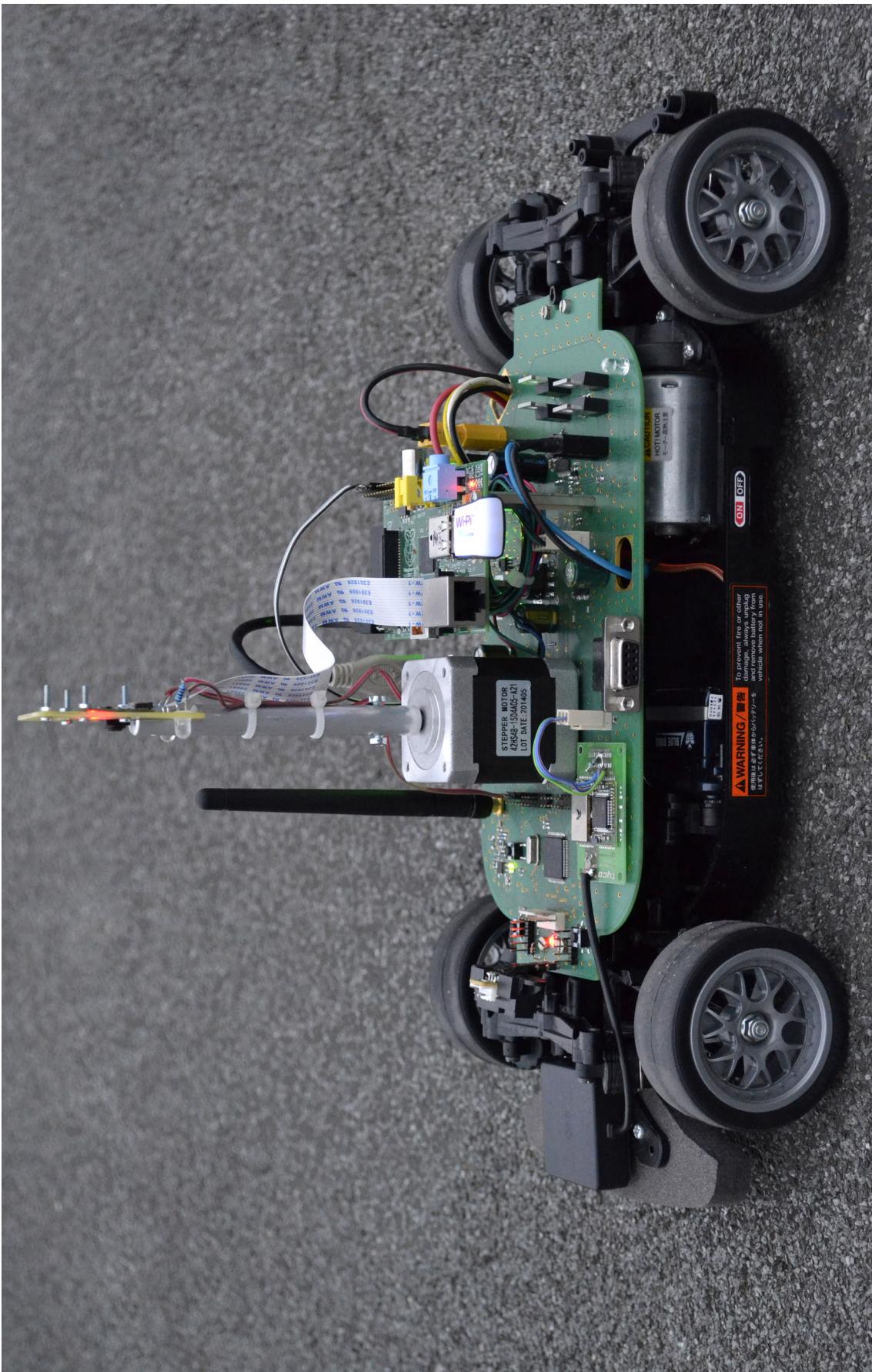
F.4.1. ábra. A mobil adatgyűjtő egység prototípusa előlnézetből



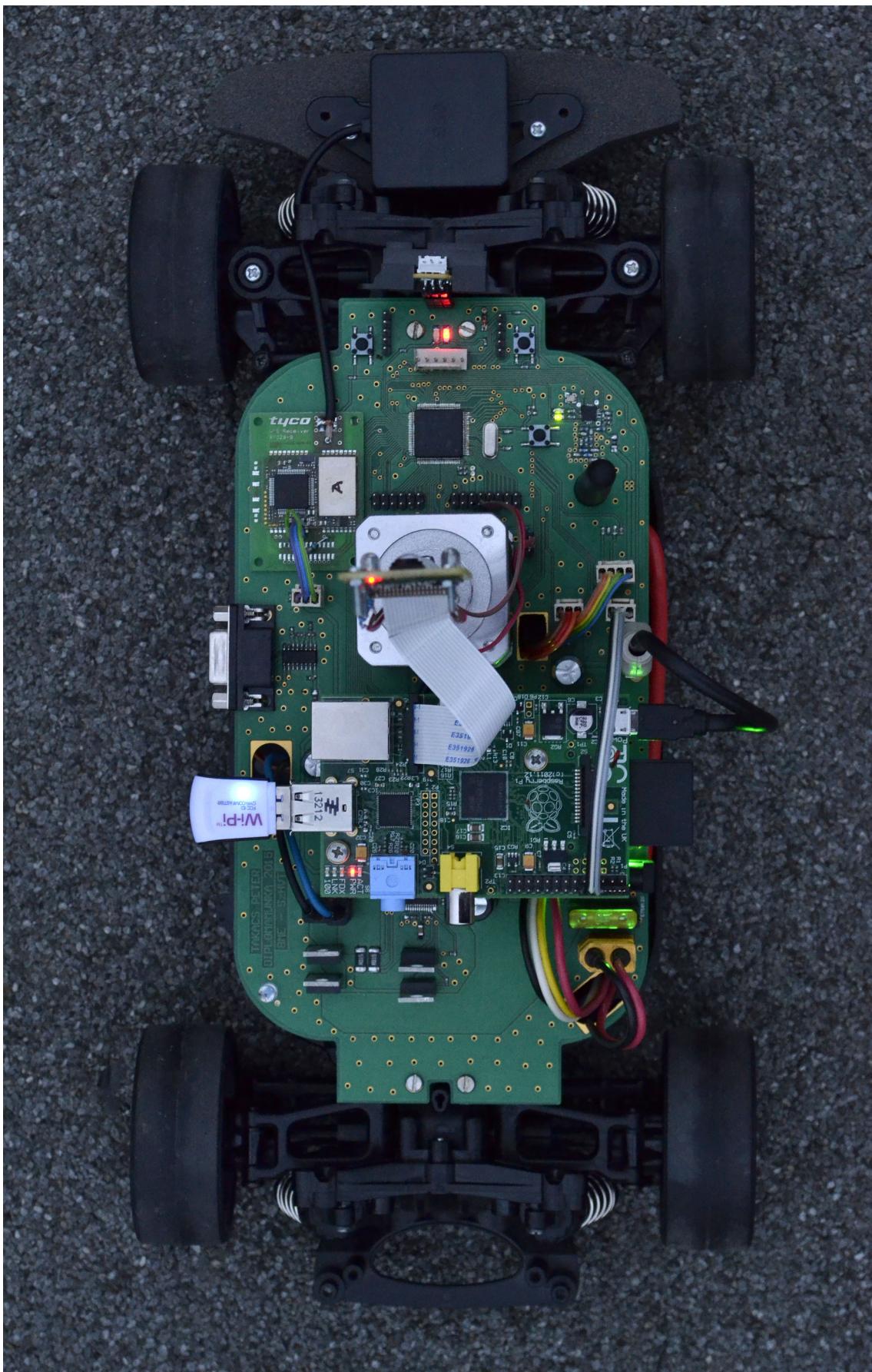
F.4.2. ábra. A mobil adatgyűjtő egység prototípusa hátulnézetből



**F.4.3. ábra.** A mobil adatgyűjtő egység prototípusa az egyik oldalról



F.4.4. ábra. A mobil adatgyűjtő egység prototípusa a másik oldalról



F.4.5. ábra. A mobil adatgyűjtő egység prototípusa felülnézetből