SZAKDOLGOZAT

Megyesi Norbert

Mechatronikai mérnök

Gödöllő

Magyar Agrár- és Élettudományi Egyetem

Szent István Campus

Mechatronikai Mérnök Szak

ELEKTRONIKUS AUTÓ VEZÉRLÉSE MIKROKONTROLLEREKKEL

Belső konzulens: Tóth János
egyetemi tanársegéd
Külső konzulens: Seres István
egyetemi docens
Készítette: Megyesi Norbert
O0QTFS
nappali tagozat

Intézet/Tanszék: Műszaki Intézet
Mechatronika Tanszék

Gödöllő
2022

MAGYAR AGRÁR- ÉS ÉLETTUDOMÁNYI EGYETEM

Szent István Campus, Gödöllő
Cím: 2100 Gödöllő, Páter Károly utca 1.
Tel.: +36-28/522-000
Honlap: https://godollo.uni-mate.hu

MŰSZAKI INTÉZET
MECHATRONIKAI MÉRNÖK ALAPSZAK
Gépipari mechatronika specializáció

SZAKDOLGOZAT

feladatlap

Megyesi Norbert *(O0QTFS)*

részére

A diplomadolgozat címe:

Elektronikus autó vezérlése mikrokontrollerekkel

Feladatkiírás:

Bevezetés, cégbemutatás, szakirodalom feldolgozása, probléma bemutatása, technológiai számítások, programozás, gazdasági számítás, összefoglalás

Közreműködő tanszék: Mechatronika

Külső konzulens: *Dr. Seres István*

Belső konzulens: Dr. Tóth János, *egyetemi tanársegéd,* MATE, Műszaki Intézet

Beadási határidő: 2022. november 02.

Gödöllő, 2022. szeptember 05.

Jóváhagyom		**Átvettem**
(tanszékvezető)	(szakfelelős)	(hallgató)

A dolgozat készítőjének külső konzulense nyilatkozom arról, hogy a hallgató az előre egyeztetett konzultációkon megjelent.

Gödöllő, 2022. 11. hó 07 nap

(külső konzulens)

TARTALOMJEGYZÉK

1 BEVEZETÉS 7
1.1 Témaválasztás 7
1.2 Célok bemutatása 7
2 IRODALOMFELDOLGOZÁS 9
2.1 A Raspberry Pi bemutatása röviden 9
2.1.1 A Raspberry Pi készülékek 9
2.1.2 A Python és a C++ 9
2.1.3 A Raspbian 10
2.2 Az Ardino bemutatása röviden 11
2.2.1 Az Arduino mint hardware 11
2.2.2 Az Arduino mint software 11
2.3 A 3D nyomtatás bemutatása 12
2.3.1 A 3D nyomtatás története 12
2.4 A léptetőmotor 15
2.4.1 A léptetőmotor bemutatása 15
2.4.2 A léptetőmotor működése 15
2.4.3 A léptetőmotorok típusai 16
2.5 Pozícióérzékelés 18
2.5.1 A pozícióérzékelők típusai 18
2.6 Szervomotorok 21
2.6.1 A szervomotorok bemutatása 21
2.6.2 A DC szervomotorok felépítése 21
2.6.3 A szervomotorok vezérlése 22
2.7 Vezeték nélküli Rádiós Adatátvitel 22
2.7.1 Fizikai alapok 22
2.7.2 A rádióhullám terjedése 23

2.7.3 Frekvenciasávok 23
2.7.4 Modulátorok és demodulátorok 24
2.8 Lehetséges eszközök bemutatása 24
2.8.1 Raspberry Pi 25
2.8.2 Arduino 27
2.8.3 Rádiós chip 29
3 MEGVALÓSÍTÁS 30
3.1 Az eszközök kiválasztása 30
3.1.1 A Raspberry Pi, mint vezérlőegység az autón 30
3.1.2 Arduino UNO R3 mint adatokat gyűjtő és küldő egység 31
3.1.3 NRF24L01+ mint rádiókommunikációs chip 31
3.1.4 Léptetőmotor és vezérlése 31
3.1.5 Szervomotor és vezérlése 33
3.1.6 Pozícióérzékelés 33
3.2 A működtető szoftver 34
3.2.1 Raspberry Pi-n futó programok 34
3.2.2 Az Arduino szoftvere 40
3.3 A 3D tervezett váz 42
3.3.1 A tervező szoftver 42
3.3.2 A felhasznált elemek 42
3.3.3 A tervezett váz 45
3.3.4 Az összeszerelési ábra 46
3.3.5 A 3D nyomtató 46
3.3.6 A Filament 46
4 GAZDASÁGI SZÁMÍTÁSOK 47
4.1 Felhasznált alkatrészek és anyagok 47
4.2 Munkadíj 47

5 NYILATKOZTOK 48
6 ÖSSZEFOGLALÁS 51
7 SUMMARY 52
8 IRODALOMJEGYZÉK 53
9 MELLÉKLETEK 55
9.1 A Get_Radio_Data.py 55
9.2 A Data_Handle.py 56
9.3 Stepper_Motor_Class.py 57
9.4 A Servo_Motor_Class.py 58
9.5 main.py 59
9.6 Arduinocode.ino 60

1 BEVEZETÉS

Ebben a fejezetben bemutatom a választott témát és az elérni kívánt célokat.

1.1 Témaválasztás

Felvetődik a kérdés az önvezető autókkal kapcsolatban, hogy teljesen megvalósíthatóak-e? Létrehozható-e egy annyira fejlett mesterséges intelligencia, ami képes minden pillanatban a lehető legjobb döntést meghozni annyira összetett helyzetekben is, mint amik autóvezetés közben adódhatnak, valamint szeretnénk-e egyáltalán egy ennyire kényes és fontos dolgot az életünkben számítógépekre és egyéb, az önvezető autó létrehozásához szükséges eszközökre bízni?

Szakdolgozatomban egy lehetséges alternatíva legelső lépcsőfokának szánt kísérletet szeretnék bemutatni. Megpróbáltam az önvezető autók legfontosabb tulajdonságait átültetni egy olyan rendszerbe, ahol nem vesszük ki a kormányt az ember kezéből, csupán nem létfontosságú, hogy az, aki vezeti az autót, fizikailag is az autóban tartózkodjon. Ezáltal elérhetjük azt, hogy - bár az autóban nem tartózkodik senki - mégis a vezetés során felmerülő nehézségeket egy ember kezeli, valamint a döntéseket sem egy előre programozott számítógép hozza meg. Megtartunk rengeteg a vezetés során fontos emberi tényezőt: előrelátás, morális gondolkodás és ösztönök, de ezekhez társul az az előny, amit egy önvezető autó nyújthat: nem szükséges a fizikai jelenlét az autóban, illetve nem a gépjárműben tartózkodóknak kell vezetni.

1.2 Célok bemutatása

Egy távolról irányítható elektromos autó prototípusának megtervezését választottam dolgozatom témájának, melyet nem távirányítóval, hanem elektromos kormánnyal és pedálokkal lehet vezetni. Ez a gyakorlatban azt jelenti, hogy egy asztalnál ülök, előttem a kormánnyal és a pedálokkal lábamnál. Forgatom a kormányt, lenyomom a pedálokat, és ez alatt mérem a pedál és kormányállásokat jelző számokat egy Arduino-val. Ezt rádióhullámokkal eljuttatom az autón található Raspberry Pi-re, ami a kapott adatok alapján beállítja a kerekek állását és a sebesség értékét és irányát.

Bemutatom a felhasznált eszközöket, kitérek a feladatban felmerülő programozási kihívásokra, és ismertetem a tervezés folyamatát az elejétől a végéig. További célom a lehetséges fejlesztések és bővítések bemutatása.

Nem elhanyagolható egy ilyen feladat esetében a különböző egységek gazdasági oldalról történő megvizsgálása sem, így szót fogok ejteni az eszközöknél a különböző alternatívákról, és hogy miért választottam a felhasznált tárgyakat.

Az alábbiakban bemutatott módszerekkel már működött a vezérlés egy LEGO-ból épült autó vezetésére, az autó LEGO helyett 3D nyomtatott kivitelezése jelenleg is zajlik.

2 IRODALOMFELDOLGOZÁS

Ebben a fejezetben a felhasznált eszközök szakirodalmát mutatom be.

2.1 A Raspberry Pi bemutatása röviden

Ebben a fejezetben a Raspberry Pi-t fogom bemutatni.

2.1.1 A Raspberry Pi készülékek

A málna PC, ahogy Magyarországon sokan emlegetik, egy „egytáblás számítógép”, angol nevén single-board computer (SBC). Lényegében egy áramköri kártyára szerelt számítógép. Az első eszköz a 2012-ben megjelent Raspberry Pi 1 B volt, majd nem sokkal azt követően megjelent a gyengébb, de olcsóbb változat a Raspberry Pi 1 A. Az azóta eltelt 10 évben rengeteg különböző árú készüléket értékesített a gyártó. Most külön nem emelném ki mindet, természetesen az évek során egyre több mindenre képes, egyre erősebb hardverrel rendelkező számítógépeket készítettek. (Upton & Halfacree, 2016)

Magát az alapítványt, ami a Raspberry Pi-k mögött áll, az iskolákban történő számítástechnikai oktatás elősegítése miatt hozták létre, ezért természetesen egy megfizethető, mégis rengeteg lehetőséget rejtő gépet kaptunk.

2.1.2 A Python és a C++

A Raspberry Pi elsődleges programozási nyelvei iparban a Python illetve a C++. Mind a kettőnek vannak előnyei és hátrányai, de a C++ elterjedtebb az iparban. Ott az egyik legfontosabb jellemzője egy jó programkódnak, azt leszámítva, hogy azt csinálja, amit kell, hogy a lehető leggyorsabb legyen. (thinkful.com, 2022)

A python egyik legnagyobb erőssége itt valójában hátrány, az hogy a nyelv igen egyszerű. Gyorsan tanulható, nincsenek komoly megkötések. (Summerfield, 2008)

Csak egy rövid példával élve, ha C-ben, Javaban vagy sok más programozási nyelvben létre kell hoznunk egy változót, akkor azt *deklarálnunk* kell, meg kell mondanunk a számítógépnek, hogy a változó, amit létrehoztunk milyen típusú. Ez segít a számítógépnek mire is fogjuk használni a változót, illetve mekkora helyet foglaljon le a memóriában számára. Legyen az *integer* vagy *char*, rengeteg típusú változó létezik, mindnek más-más funkciója van. Pythonban viszont nincs szükség deklarálásra , egyszerűen létrehozunk egy változót, értéket adunk neki, és a megadott értékből már tudja, hogy mi a változónk típusa.

Ez csak egy, a sok ilyen és hasonló egyszerűsítésből, ami ugyan nagyon befogadhatóvá teszi a Pythont, gyorsan elsajátíthatóvá, cserébe viszont nem lett egy gyors lefutású, optimális nyelv. A lassúságának oka, hogy futásidőben fordít, úgy nevezett értelmezett

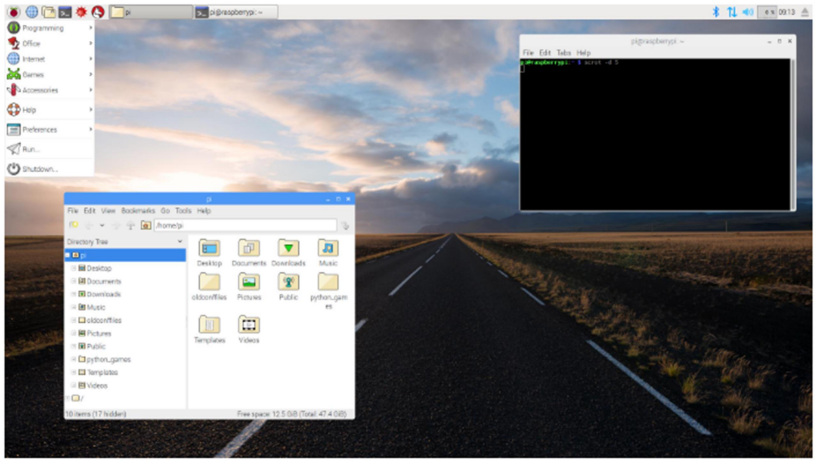
(*interpreted)* nyelv. Ez azt jelenti, hogy amikor elindítjuk a programot, a számítógép akkor fordítja le számára is értelmezhetővé az általunk írt sorokat. Iparban is sokkal inkább a prototípusgyártásnál használják, ha le kell tesztelni magát az algoritmust, hogy úgy működik-e ahogy szeretnénk. (fuzzingbook.org, 2022)

A már kész termékekben vagy robotokban a sokkal bonyolultabb, nehezebben elkészíthető, de a Python- hoz képest villámgyors futású C++ kódokat fogunk találni. A C++ úgy nevezett fordított (*compiled*) nyelv. Még a program indítása előtt át kell futtatnunk a programot egy fordítón (*complier)*, ami a számítógép számára is érthetővé teszi. Ezért amikor a program fut,a fordítás már nem lassítja a végrehajtást.

2.1.3 A Raspbian

A Raspbian egy operációs rendszer kifejezetten Raspberry Pi-re optimalizálva. A kezelőfelület látható a 1. ábrán. Alapját egy Linux disztribúció adja, a Debian. Előre telepített 35000 csomagjában nem csak egy Raspberry Pi elindításához szükséges minimum van, hanem rengeteg, a fejlesztést megkönnyítő egyéb software is. Ezekből is ki fogok emelni néhányat.

Érdekesség hogy nem a Raspberry Pi mögött álló alapítvány hozta létre a Rasbiant.



1. ábra A Rasbian
Forrás: (raspbian.org, 2022)

2.2 Az Ardino bemutatása röviden

Ebben a fejezetben az Arduinot mutatom be.

2.2.1 Az Arduino mint hardware

Az SBC vagyis single-board computer piac egy másik cégóriása az Arduino. Ez a gyártó is széles árspektrumban gyárt különböző felszereltségű mikrokontrollereket, létezik a 2. ábrán is látható Arduino Uno, ebből az évek során többfajta is megjelent, Arduino Pro, a méreteiben kicsit Arduino Mikro, és rengeteg egyéb eszköz.

2. ábra Az Arduino Uno
Forrás: (farnell.com, 2022)

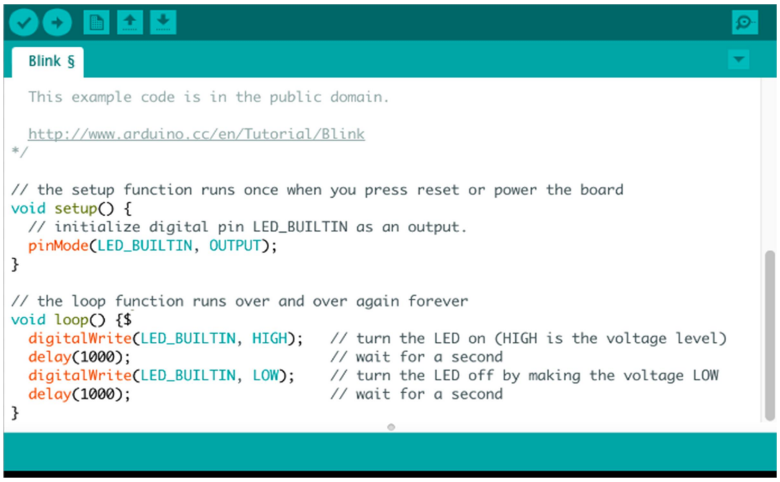
Az egyik lényeges, számunkra is fontos különbséget a Raspberry Pi-hez képest az jelenti, hogy az Arduino rendelkezik olyan portokkal, amik analóg jelek fogadására és küldésére képesek. Erre a Raspberry Pi esetében elméletileg nincs lehetőségünk, pontosabban léteznek különböző bővítőkártyák ennek kiküszöbölésére, de eredendően nem található rajta analóg port. Cserében viszont a Raspberry Pi készülékek nagyobb számítási kapacitással rendelkeznek és bonyolultabb feladatok ellátására is alkalmasak. A tervem a jövőt illetően, hogy tovább fejlesztem ezt a projektet, így a Raspberry Pi használata az autó fedélzeti központi egységeként számomra azért jobb, mert a későbbiekben már nem kell fejlesztenem a hardvert olyan feladatok megoldására, amire esetleg az Arduino már nem megfelelő. (www.interviewbit.com, 2022)

2.2.2 Az Arduino mint software

Az Arduino-nak van saját programozási nyelve, de egy sokkal jobb dologgal is előálltak a fejlesztők, egy saját IDE-t (integrated development enviroment) is létrehoztak, egy saját fejlesztőkörnyezetet amelynek a kezelőfelületét mutatja be az 3. ábra. Ez a Java nyelven íródott, Mac és Windows rendszereken is futtatható applikáció segít az Arduinora szánt programok írásában, fordításában és feltöltésében. Található benne egy szövegszerkesztő,

amiben az olyan egyszerű funkciók mellett, mint a másolás, a kivágás és a beillesztés, egyéb a programkód létrehozását megkönnyítő funkciók is helyet kaptak, mint a brace matching (az egymáshoz tartozó zárójeleket kiemeli), az automatic indenting (szép olvashatóvá teszi a kódot), és a syntax highlight (ami kategóriánként más színnel emeli ki a szövegben a szavakat). (Monk, Programming Arduino, 2013)

2019-ben megjelent az új, fejlesztett Arduino Pro IDE alfa verziója, ami rengeteg újdonságot hozott magával. 2021-re már csak IDE2 néven futó béta verzióban már Git támogatást, és az autocompletion (Az első néhány betű alapján felajánl lehetőségeket) mellett a programozók legjobb barátját hozta, a dark mode-ot.



3. ábra Az Arduino IDE
Forrás: (arduino.cc, 2022)

2.3 A 3D nyomtatás bemutatása

Ebben a fejezetben a 3D nyomtatást fogom bemutatni.

2.3.1 A 3D nyomtatás története

A 3D nyomtatás elméletben már 1945-ben megjelent, de 1971-ig várni kellett mire a gyakorlatban is valóra váltották az elképzelést. Ekkor még elégé kezdetleges volt a Teletype Corporation első 3D nyomtatója. Ezt még Liquid Metal Recorder-nek nevezték és olvasztott waxszal kísérleteztek Johannes F. Gottwald szabadalma alapján, amely látható a 5. ábrán. Az úgy működött, hogy folyósított fémet helyeztek rétegekben egymásra, ami aztán a kívánt formában megszilárdult. A formát a szerkezet előre meghatározott mozgása adta meg.

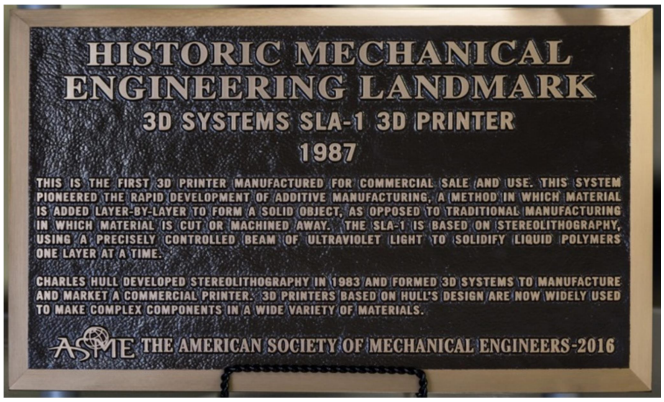
A következő áttörés 1980-ban történt. A japánban található Nagoya város állami kutatóintézetében, ügyvédként dolgozó Dr. Hideo Kodama egy módszert is kidolgozott

Gottwald elképzeléseinek továbbfejlesztéséhez. Hőre keményedő polimereket kellene használni fémek helyett. Bár több helyen is publikálta, és a szabadalmat is benyújtotta, az egész nem jutott sehova érdeklődés hiányában. (autodesk.com, 2022)

William Masters 1984-ben megjelent szabadalmában, a számítógépek által vezérelt gyártási folyamatok és rendszerekben jelent meg először a 3D nyomtatás, mint fogalom.

Szintén 1984-ben Franciaországban is megjelent egy szabadalom az *additív gyártásra* használt sztereolitográfia. A sztereolitográfia olyan háromdimenziós tárgyak gyártására szolgáló folyamat, ahol számítógépek által vezérelt lézersugarakat használnak a szerkezet felépítésére, rétegről rétegre egy folyékony polimerből, ami lézerfény hatására megszilárdul.

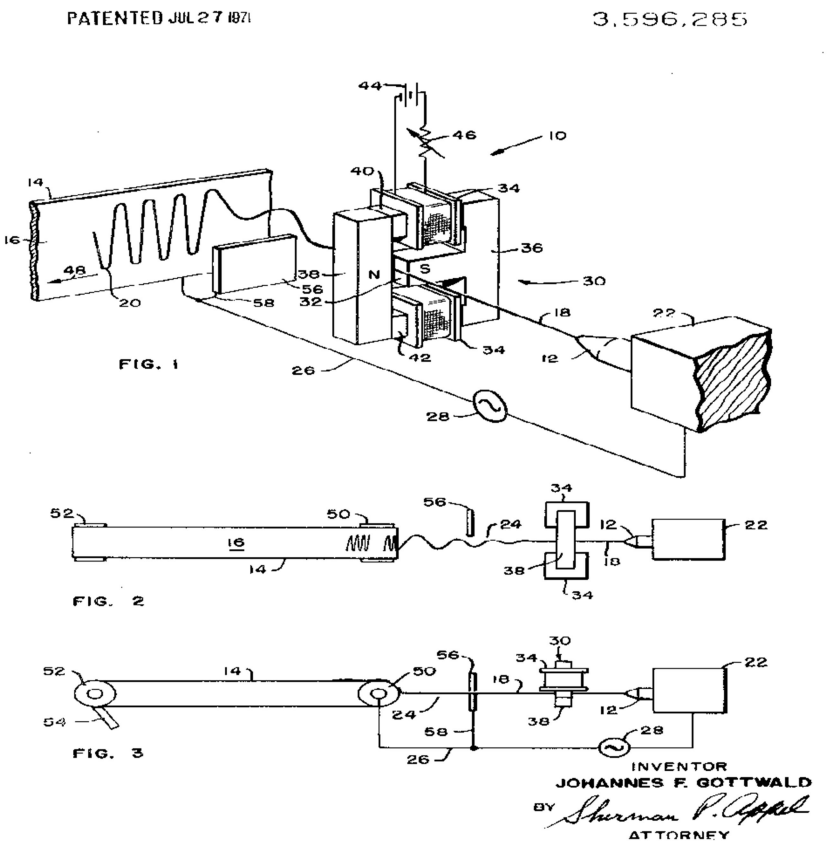
Chuck Hull cége, a 3D Systems Corporation adta ki az első sztereolitográfiás berendezést (stereolithographic apparatus, röviden SLA) 1987-ben. Az SLA-1 lehetővé tette, hogy az addig szükséges idő töredéke alatt lehetett komplex alkatrészeket létrehozni rétegenként. A mai napig használt STL fájlformátumot is ők fejlesztették ki.



4. ábra Az ASME 2016-os táblája az SLA-1 mérnöki mérföldkőnek állít emléket
Forrás: (www.3dsystems.com, 2022)

Ezekben az időkben a 3D nyomtatás még csak egy felemelkedőben lévő technológia volt, a használt anyagokról nem rendelkeztünk akkora tudással, mint ma. A gépek is több ezer dollárba kerültek, kizárólag a gyárakban lehetett őket megtalálni. A mérnöki tudományok egy jelentős eseménye volt az SLA-1 elkészülése. Ezt bizonyítja a 4. ábrán látható 2016-os tábla is amelyet az ASME(Amerikai gépészmérnök társaság) állított ki.

Manapság a 3D nyomtatás az egyik leggyorsabban fejlődő technológia világon. Az egyszerű műanyag alkatrészek nyomtatását réges-rég meghaladták a mai gépek. A nyomtatható anyagok kutatása már odáig eljutott, hogy képesek vagyunk élelmiszereket, egész házakat, sőt akár szerveket is nyomtatni. Rengeteg egyéb kutatás is zajlik jelenleg a 3D nyomtatott autó vagy repülőtől a Marson is életképes robotig. (autodesk.com, 2022)



5. ábra Az eredeti 1971-es szabvány rajza
Forrás: (google.com, 2022)

2.4 A léptetőmotor

Ez a fejezet a léptetőmotorokkal foglalkozik.

2.4.1 A léptetőmotor bemutatása

A léptetőmotor egy egyenáramú, szénkefe nélküli motor, amiben a szerkezete meghatározza, hány lépésre van felosztva egy fordulat. Általában 200 egyenlő részre bontják a 360°-ot, ez 1,8°-ot jelent lépésenként. Ez lehet különböző típusokban eltérő, vannak 2; 2,5; 5; 15 vagy 30°-os lépéssel rendelkező léptetőmotorok. (tme.eu, 2022)

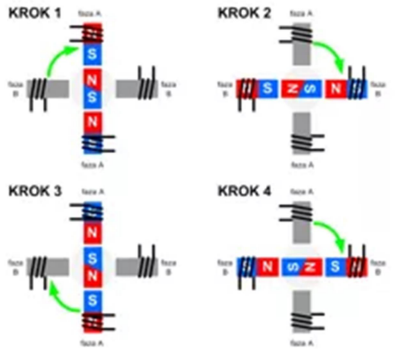
Könnyen megállapítható, hogy ennél a motortípusnál nem egyszerű forgásról beszélünk. A lépések, illetve a köztes állapotok kihagyása jellegzetes hangokkal jár.

A léptetőmotorokat a legtöbb esetben egy külön digitális áramkörön alapuló léptetőmotor vezérlőkkel használjuk együtt. A vezérlő eljárás több dologtól is függ, ezek a visszacsatolás, a fázisok száma és a vezérlés módja. Létezik olyan vezérlő ahol PWM (Pulse-width modulation), „impulzusszélesség moduláció” segítségével állítható be a tekercseken folyó áram. A forgás irányát és a lépéseket pedig négyszögjel hullámokkal vezérlik. Vannak modellek, amelyeket motorvezérlők segítségével lehet vezérelni, egyes modellek pedig egyszerűen a megfelelő kimenetekre megfelelő időben kiadott jelekkel is vezérelhetők.

2.4.2 A léptetőmotor működése

A léptetőmotorok két részből állnak. Az állórészből (stator), valamint a csapágyakkal a tengelyre szerelt forgórészből (rotor). A rotort az állórész körül létrehozott forgó mágneses mező forgatja. Az acélból vagy más fémből készült állórész az elektromágnesek váza, nevezetesen a rotor körül meghatározott helyekre szerelt tekercseké. Mágneses mezőt hoz létre, ha áram alá kerülnek a tekercsek. Az ezeken átáramló áram iránya és áramerőssége adja meg a mágneses fluxusok irányát és erősségét.

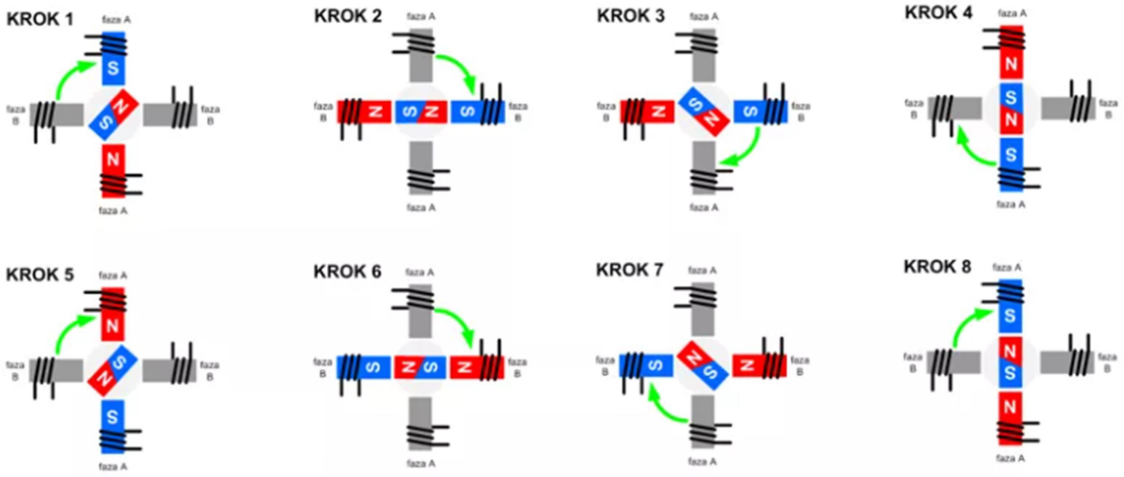
Amikor a tekercs áram alá kerül, az elektromágnes vonzani fogja a rotorra szerelt mágnest, ezáltal a tengely és maga a rotor is elforog olyan szögbe, ahol a mágneses fluxus vagy több fluxus eredője a legkevésbé taszítja. Miután a tengely a megfelelő helyzetbe fordult, másik elektromágnest helyeznek áram alá. amivel újabb elforgást idéznek elő.



6. ábra A teljes léptetés
Forrás: (tme.eu, 2022)

A tekercsek ilyen módon történő kapcsolás, meghatározott sorban idézi elő a motor lépéseit, előre vagy hátra. A forgatást tovább lehet finomítani, ha féllépéses üzemmódban használjuk a motort a 6. ábrán látható teljes léptetés helyett.

A féllépéses üzemmódban működő motor működési elve látható a 7. ábrán. Lehetővé teszi egy lépés kettéosztását, így a névleges szög felével fordul el egy lépésként. Ilyenkor a teljes fordulathoz 400 lépés szükséges.



7. ábra A féllépéses léptetés
Forrás: (tme.eu, 2022)

2.4.3 A léptetőmotorok típusai

A léptetőmotor egy aprólékosan felépített, szálcsiszolt egyenáramú motor. A manapság használt BLDC (Brushless DC) motorok nem sokban különböznek az állandó mágneses léptetőmotoroktól, vezérlésükben is sok a közös vonás. A léptetőmotorok típusait két dolog alapján különböztetjük meg, a tekercsek által igényelt fázisok számával, illetve a felépítésükkel.

2.4.3.1 Állandó mágneses léptetőmotor

Az állandó mágneses léptetőmotorokban lévő rotorokon több pólus és sok mágnes található. Erre azért van szükség, hogy megnövelhessük a lépések számát, illetve növeljük a pozicionálás pontosságát.

Elméletileg működőképes, ha csak kettő elektromágnes van és a forgórész sugárirányban van mágnesezve, de ez hatalmas szerkezetet eredményezne, a lépések száma és pontossága viszont alacsony.

A változtatásokkal elérhető a 7,5-15° lépésenként, ez 48-24 lépést jelent teljes fordulatonként. Mivel a rotor mágnesezett, a mágneses indukció megnő, ez megemelkedett nyomatékot jelent. Az állandó mágneses léptetőmotorok egyszerű felépítése ugyan csökkenti az árukat, és további előnye a nagyobb elérhető nyomaték, az alacsony felbontása miatt mérsékeltek a lehetőségei.

2.4.3.2 Változó reluktanciájú léptetőmotorok

A léptetőmotorok első generációi közé tartozó, változó reluktanciájú léptetőmotorok napjainkban már ritkán használatosak. Ebben a motortípusban a rotor számos, lágyvasból készült fogból áll. Mikor áram alá kerülnek az állórész tekercsei, vonzani kezdik a rotor fogait. Ezen léptetőmotorok felépítése egyszerű, könnyen vezérelhetőek, túlságosan alacsony a nyomatékuk és a felbontásuk.

2.4.3.3 Hibrid léptetőmotorok

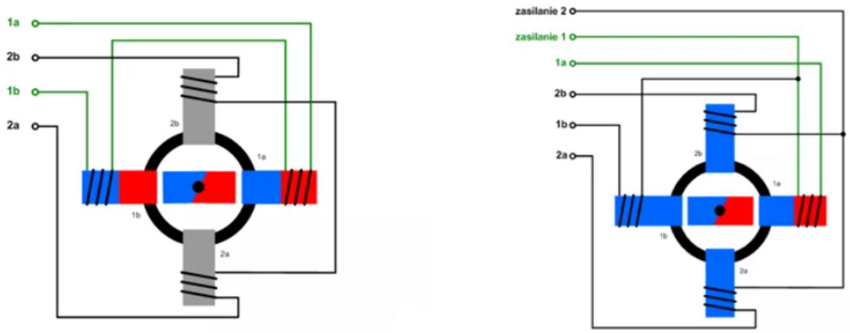
Manapság, ha léptetőmotort szeretnénk használni, nagy valószínűséggel hibrid motort fogunk választani. Ezen típusú motorok nagy felbontás (0,9-3,6° vagy 400-100 lépés) mellett is lehagyja a korábban leírt motortípusokat nyomaték, tartási nyomaték és szögsebesség tekintetében is.

A rotor a hibrid motor esetében is állandó mágnesekből épül fel, a trükk ezek elhelyezésében keresendő. Itt nem sugár, hanem az axiális irányban történik a mágnesezés. Általános esetben a rotoron két, ellentétesen mágnesezett gyűrű található, az ezeken lévő hornyok szolgálnak fogakként. (Scarpino, 2015)

2.4.3.4 Unipoláris és bipoláris motorok

A másik osztályozási mód a tekercselés típusa alapján kategorizálja a motorokat. Léteznek unipoláris (egypólusú) és bipoláris (kétpólusú) motorok. A nevükből adódóan az alapvető különbség a polaritások száma. Ez abban mutatkozik meg, hogy a bipoláris motor tekercseiben az áram iránya változó. Egy további különbség a tekercsek

összekapcsolásában van, bipoláris motor esetében ugyanis a tekercseket úgy alakítjuk ki, hogy az áram átjuthat az egyik tekercs végéről a másik elejére. Ez is látható a 8.ábrán.



8. ábra Bipoláris (balra) és unipoláris (jobbra) bekötés
Forrás: (tme.eu, 2022)

2.5 Pozícióérzékelés

A pozícióérzékelők olyan eszközök, amelyek képesek egy tárgy mozgását, vagy egy referencia ponthoz mért relatív távolságát érzékelni. Ezen eszközök képesek érzékelni egy tárgy jelenlétét vagy annak hiányát is. (Lambert, 2009)

Több szenzortípus van, amelyek hasonló feladatokat látnak el, mint a pozícióérzékelők. A mozgásérzékelők észlelik egy tárgy mozgását és kiváltanak egy válaszreakciót (mint például egy lámpa felkapcsolása). A közelségérzékelők, vagy közelítéskapcsolók azt a célt szolgálják, hogy megállapítsa egy adott tárgy a szenzor megadott hatótávján belül van-e.

Mindkettő esetében elmondható, hogy a pozícióérzékelők egy speciális formája.

2.5.1 A pozícióérzékelők típusai

A pozícióérzékelőket három fő típusba lehet helyezni. Léteznek lineáris helyzetérzékelők, forgásérzékelők és szöghelyzet-érzékelők. Sokféle technológiával megvalósíthatók, ezt mutatja a lejjebb felsorolt szenzorok sokszínűsége.

2.5.1.1 A potenciometrikus helyzetérzékelők

A potenciometrikus, azaz potenciométer segítségével működő helyzetérzékelők ellenálláson alapuló szenzorok. Egy potenciométert rajta egy csúszkával csatlakoztatunk a megfigyelt tárgyhoz, ha a tárgy elmozdul, mozgatja magával a potenciométer csúszkáját, ezzel elállítva a potenciométer ellenállását a vége és a csúszka között. Ezt felhasználhatjuk

a tárgy pozíciójának nyomon követéséhez. Ehhez szükségünk lesz egy feszültségforrásra, hogy egy adott feszültséget helyezünk a potenciométer két végére, a mért feszültség a csúszka és a potenciométer vége között arányos a csúszka pozíciójával.

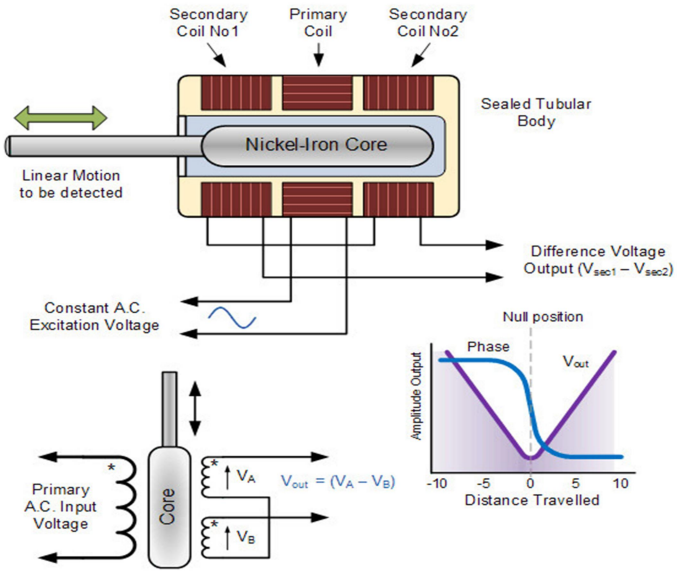
A leírt eljárás a lineáris potenciométerekre vonatkozik, de léteznek, sőt sokkal elterjedtebbek a forgatható potenciométerek. Ebben az esetben a csúszka egy toroid tekercsen csúszik, miközben változik az ellenállás a potenciométer sarka és a csúszka között. Így a forgómozgás lekövetésére van lehetőségünk.

A potenciométerek, amelyeket potenciometrikus pozícióérzékelésre használnak, tartalmaznak egy tekercselt, műanyag (valamilyen polimer) vagy cermet réteget. A cermet az angol ceramic és metal szavak összevonásából született szó, egy porkohászati úton létrehozott anyag. (Van de Voorde & Meetham, 2000)

Ezen eszközök egész olcsón hozzáférhetők, de nem igazán pontosak és a felépítésből adódóan korlátozott a mérete annak a területnek, amelyen mérhető a helyzetváltoztatás.

2.5.1.2 Induktív helyzetérzékelők

Az induktív helyzetérzékelők a tekercseikben indukált mágneses mező karakterisztikájának a megváltozásával képesek érzékelni egy tárgy pozícióját. Egyik típusa a 9. ábrán is látható LVDT (Linear Variable Differential Transformer), ezekben három különálló tekercset találunk egy üreges cső körül. Az elsődleges és a két másodlagos tekercs sorosan van kötve, de közöttük 180°-os fáziseltolás van. Az üreges csőben egy ferromágneses mag vagy tekercs található, ehhez csatlakoztatható a tárgy, amelynek a pozícióját szeretnénk megmérni. Az elsődleges tekercsre feszültséget helyezünk, ez elektromágneses teret gerjeszt az LVDT másodlagos tekercseiben. A feszültségkülönbség megmérésével a két másodlagos tekercs között megállapítható a mag helyzete, és ezzel együtt a hozzá csatlakoztatott tárgyé is. (Scherz & Monk, 2016)



9. ábra Az LVDT induktív helyzetérzékelő működése
Forrás: (thomasnet.com, 2022)

Amikor a mag középen van, az elektromágneses mezők kioltják egymást és nem mérhető feszültség, de ha kitér, megváltozik a feszültség és annak a polaritása is. Így nemcsak arról kapunk információt a megmért feszültség alapján, hogy mekkora a kitérés, hanem annak irányáról is.

Ezen szenzorok nagyon pontosak és érzékenyek, valamint súrlódásmentes működésük közben nagy területet képesek lefedni. Lezárható felépítésüknek köszönhetően alkalmazhatóak szabad téren, vagy bárhol ahol ki vannak téve az elemeknek. (Fraden, 2016)

Ahogyan a potenciometrikus helyzetérzékelők esetében, itt is van lehetőségünk egy tárgy forgómozgásának a lekövetésére. Ezeket RVDT-nek (Rotary Voltage Differential Transformer) nevezzük, és sok hasonlóságot mutatnak az LVDT-kel.

2.5.1.3 Kapacitív helyzetérzékelők

A kapacitív helyzetérzékelők működési elve, hogy a kapacitás megváltozásával érzékeli egy tárgy helyzetét. A kondenzátorok két lemezből állnak, közöttük pedig valamilyen dielektrikum (szigetelőanyag) található. Kétféle módja van a pozícióérzékelésnek, ha kapacitív szenzorokat használunk. (Bánlaki & Lovas, 2012)

Az első, ha a dielektromos állandót változtatjuk. A második, ha a kondenzátor lemezeinek egymást fedő felületét módosítjuk.

Az első esetben a mérni kívánt tárgyat a dielektrikumhoz rögzítjük, ezáltal ha a tárgy mozog, elmozdítja magával a szigetelőanyagot. Ez megváltoztatja a kondenzátor dielektromos állandóját, ahogy levegő kerül az elmozgatott anyag helyére. Ezzel a módszerrel vonal menti mozgást lekövetésére van lehetőség.

A második eset abban különbözik az elsőtől, hogy itt nem a dielektrikumhoz, hanem a kondenzátor egyik lemezéhez rögzítjük a tárgyat. Így az átfedésben lévő felületek nagysága megváltozik, ha mozog a tárgy, ami ugyanúgy a dielektromos állandó megváltozását eredményezi. A kapacitív helyzetérzékelőket is lehet lineáris elmozdulás mellett, szögelfordulást is észlelni.

2.6 Szervomotorok

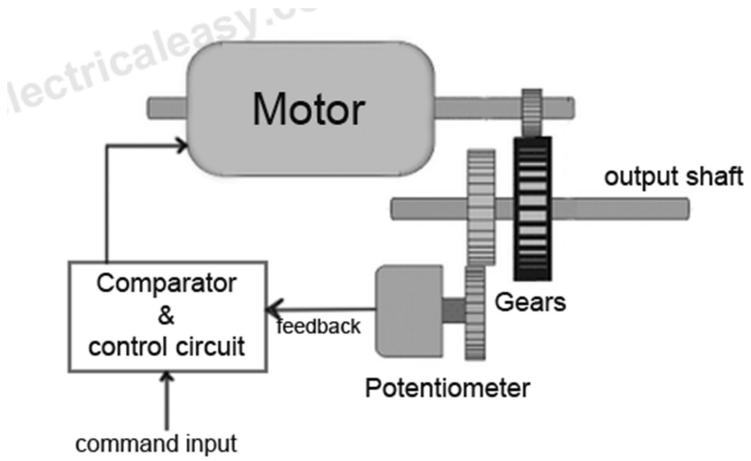
Ebben a fejezetben a szervomotorokat mutatom be.

2.6.1 A szervomotorok bemutatása

A szervomotorok speciálisan robotikára és vezérlésre tervezett motorok. Nagy nyomaték és sebesség mellett is precízen pozícionálhatóak. Léteznek AC és DC szervomotorok is, de az előbbiek inkább az iparban elterjedtebbek, a DC szervomotorok olcsóbbak és a kisebb projektekre jobban megfelelnek, így inkább ezekre koncentrálok. (Scarpino, 2015)

2.6.2 A DC szervomotorok felépítése

A DC szervomotorokon belül található egy kis DC motor, egy potenciométer és egy vezérlő áramkör. A 10. ábrán látható, ahogy motor fogaskerekek segítségével csatlakozik a vezérléshez és a tengelyhez. A tengely pozíciója egy már a fentiekben említett módón, a potenciométer segítségével érzékelve van, ami ad egyfajta visszacsatolást. A modern szervomotorokban nem potenciométert hanem encodereket használnak erre a célra. (Szénássy, 2011)

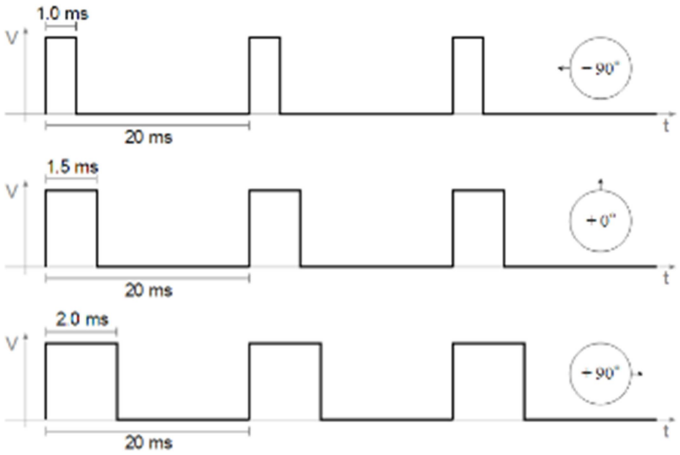


10. ábra A szervomotor felépítése
(webstarsnets.com, 2022)

2.6.3 A szervomotorok vezérlése

Az általános szervomotorok mindkét irányban 90°-ot képes fordulni, ez 180°-os mozgástartományt jelent. Az ennél nagyobb elfordulást egy beépített mechanikus gát akadályozza meg.

A szervomotorhoz a tápellátáshoz szükséges két vezetéken kívül még egy vezérlő vezetéket használok. Egy PWM jelet küldünk a vezérlő vezetéken, 20 ms gyakorisággal. Az impulzus szélessége adja meg, merre és hogyan forogjon a motor. A 11. ábrán látható, hogy például az 1 ms impulzus az óra járásával ellentétes irányú, 90°-os mozgást vált ki a tengelyen.



11. ábra A PWM vezérlés szervomotoroknál
Forrás: (jameco.com, 2022)

2.7 Vezeték nélküli Rádiós Adatátvitel

Ebben a fejezetben a rádiós kommunikációt mutatom be.

2.7.1 Fizikai alapok

Az olyan elektromágneses sugárzások, mint például a fény vagy a rádióhullámok, sok dologban hasonlítanak egymásra. Mind hozzávetőlegesen egyenesen halad körülbelül 3×10^8 m/s sebességgel, az amplitúdójuk ciklikusan változik az idő szerint, (szinuszosan változnak). Azt a számot, hogy ez hányszor történik meg egy másodperc alatt, frekvenciának hívják, (jele f, mértékegysége 1/s vagy a híres német fizikus után hertz [Hz]).

Azt hogy hány másodpercre van szükség egy ciklus teljesítéséhez periódusidőnek hívjuk (jele: T), és a frekvencia reciproka.

1. egyenlet

$$f = \frac{1}{T}$$

Az egymást követő két hullámhegy közötti szakaszt hullámhossznak (jele: λ, a görög lambda) hívjuk, és szintén találunk összefüggést közte és a frekvencia között.

2. egyenlet

$$f = \frac{c}{\lambda}$$

Ahol c a fénysebesség [m/s]

2.7.2 A rádióhullám terjedése

A rádióhullám egy mágneses és egy elektromos mező, egymáshoz képest 90°-ban történő periodikus változásából áll.

Ugyanúgy ahogy a fény esetében, a rádióhullámok intenzitása is csökken, ahogy a forrástól mért távolság nő. Ezt az okozza, hogy növekszik a terület, amin eloszlik a mágneses és elektromos energia, így az elérhető intenzitás (= energia/egységnyi területen) csökken.

A küldő antenna az eszköz, amely a térbe juttatja a transzmitter (adó) által generált rádióhullámokat. Ez megtervezhető úgy is, hogy egy adott irányba koncentrálja az energiát, így megnövelve a hatékonyságot abban az irányban. Például mint mikor fénysugarakat párhuzamosítunk egy parabolával.

2.7.3 Frekvenciasávok

A rádióhullámokat különböző sávokba osztályozzuk, a 30 kHz-től 30 GHz-ig. Minden sávnak kiosztott szerepe van felhasználás terén. Adva van, hogy mely sávokat használjuk televízió és rádiókommunikációra, melyek szolgálnak telemetriára vagy műholdak közötti kommunikációra. Ezeket a 12. ábra mutatja be.

A rádió-frekvenciás sávszélesség alatt azt a tartományt értjük, amit a modulált frekvencia jel lefed. A jel által szállított információhoz társítunk egy adott sávszélességet, és a szállító jelnek legalább akkora csatornaszélességgel kell rendelkeznie, mint az információ sávszélessége.

A szabályos amplitúdó-moduláció (AM) esetén a rádió-frekvenciás sávszélességnek legalább kétszer akkorának kell lennie, mint az információhoz társított sávszélesség.

Table 5: Utilization of Radio-Frequency Spectrum		
very low frequencies (vlf)		time signals, standard frequencies
	30 kHz	
low frequencies (lf)		fixed, maritime mobile, navigational, radio broadcasting
	300 kHz	
medium frequencies (mf)		land, maritime mobile, radio broadcasting
	3 MHz	
high frequencies (hf)		fixed, mobile, maritime and aeronautical mobile, radio broadcasting, amateur
	30 MHz	
very high frequencies (vhf)		fixed, mobile, maritime and aeronautical mobile, amateur, radio and television broadcasting, radio navigation
	300 MHz	
ultrahigh frequencies (uhf)		fixed, mobile, maritime and aeronautical mobile, amateur, television broadcasting, radio location and navigation, meteorological, space communication
	3 GHz	
superhigh frequencies (shf)		fixed, mobile, radio location and navigation, space and satellite communication
	30 GHz	

12. ábra A rádióhullámok spektruma
Forrás: (britannica.com, 2022)

A telefonos beszélgetés nagy érhetőséget igényel, de a természetesség (a forrást a lehető legjobban utánozza a jel) nem kifejezetten fontos. Tesztek kimutatták, hogy egy beszélgetés során körülbelül 300 és 3500 hertz közötti komponensekből áll, ezért a rádió által szállított telefonos csatornák négy kilohertzre vannak korlátozva. (Budó , 1994)

A 12. ábra bemutatja, mely frekvenciákat mire használnak.

2.7.4 Modulátorok és demodulátorok

A hordozó hullám, az a rádió-frekvenciás hullám, amely szállítja az információt. Ezt az információt egy modulációs folyamat során csatolják a hordozó hullámhoz azáltal, hogy megváltoztatják a hordozó egyik karakterisztikáját. Beszélhetünk amplitúdó-modulációról. frekvencia-modulációról vagy fázis-modulációról.

2.8 Lehetséges eszközök bemutatása

Az eddigiekben a felhasznált eszközök működését, felépítését esetlegesen rövid történetét mutattam be. Most ebben az alfejezetben konkrét példákat hozok milyen választási lehetőségeim voltak az eszközöket illetően.

2.8.1 Raspberry Pi

Először a Raspberry Pi-t választom ki, ami minden feladatot ellát az autón. Ahogy az 13. ábrán látszik jelenleg tizenkét darab, különböző modell érhető el, de ezek közül csak öt darab van, ami eléri az általam szükségesnek ítélt 1GB RAM-ot, ezek a Raspberry Pi 2 B,

Raspberry Pi specs

Raspberry Pi Model	Released	USB Ports	CPU	RAM	Bluetooth	WiFi
Raspberry Pi 1 B	Feb 2012	2x USB 2.0	700MHz	512MB	No	No
Raspberry Pi 1 B+	July 2014	4x USB 2.0	700MHz	512MB	No	No
Raspberry Pi 1 A+	Nov 2014	1x USB	700MHz	512MB	No	No
Raspberry Pi 2	Feb 2015	4x USB	900MHz	1GB	No	No
Raspberry Pi Zero	Nov 2015	1x Micro USB	1GHz	512MB	No	No
Raspberry Pi 3	Feb 2016	4x USB	1.2GHz	1GB	4.1 LE	Yes
Raspberry Pi Zero W	Feb 2017	1x Micro USB	1GHz	512MB	4.1	Yes
Raspberry Pi 3 B+	Mar 2018	4x USB 2.0	1.4GHz	1GB	4.2, BLE	Yes
Raspberry Pi 3 A+	Nov 2018	1x USB 2.0	1.4GHz	512MB	4.2, BLE	Yes
Raspberry Pi 4	June 2019	2X USB 3.0, 2X USB 2.0	1.5GHz	1GB, 2GB, 4GB, or 8GB	5.0, BLE	Yes
Raspberry Pi 400	November 2020	2X USB 3.0, 2X USB 2.0	1.8GHz	4GB	5.0, BLE	Yes
Raspberry Pi Pico	January 2021	USB C	133MHz	264KB	No	No

13. ábra Raspberry Pi specifikációk
Forrás: (howchoo.com, 2022)

Raspberry Pi 3, Raspberry Pi 3 Model B+, Raspberry Pi 4 és a Raspberry Pi 400. Ezek közül fogom néhány felszereltségét részletesen bemutatni.

2.8.1.1 Raspberry Pi 2

A Raspberry Pi 2 felszereltsége:

- 900 MHz-es ARM Cortex-A7 négy magos processzor
- 1 GB RAM
- Ethernet port
- 4 db USB 2 port
- 40 GPIO (General –purpose input/output „általános rendeltetésű be és kimenet”) pin
- a kompatibilis képernyőkhöz való csatlakozásra egy Full HDMI (High Definition Multimeda Interface) port
- a hangok lejátszását egy 3.5 mm-es Jack csatlakozó

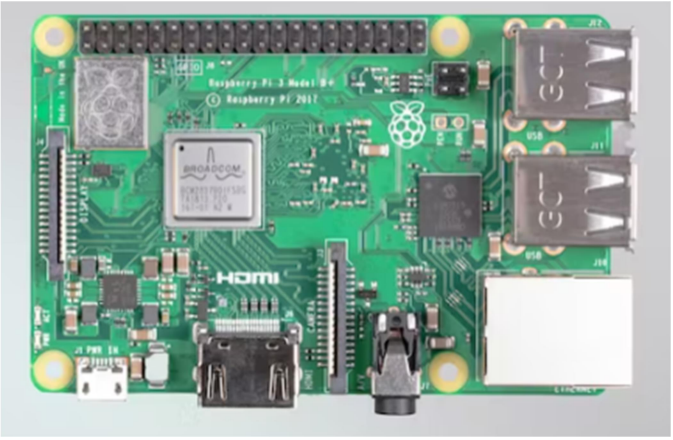
- CSI (Camera Serial Interface)
- DSI (Display Serial Interface)
- az adatok és az operációs rendszer tárolására egy Micro SD card slot.
- A tápellátást egy Micro USB port szolgáltatja.

14. ábra Raspberry Pi 2

Forrás: (raspberrypi.com, 2022)

2.8.1.2 Raspberry Pi 3 Model B+

- 1.4 GHz-es Boardcom BCM2837B0 Cortex-A53 négy magos, 64 bites processzor
- 1 GB LPDDR2 SDRAM
- Gigabit Ethernet port
- 2,4 GHz-es és 5GHz-es vezeték nélküli LAN és Bluetooth 4.2-es BLE
- 4 db USB 2 port
- 40 GPIO pin
- Full HDMI csatlakozó
- 3.5 mm-es Jack csatlakozó
- CSI és DSI
- Micro SD card slot
- 5V/2.5A tápellátás



15. ábra Raspberry Pi 3 Model B+
Forrás: (raspberrypi.com, 2022)

2.8.1.3 Raspberry Pi 4

- 1.5 GHz MHz-es Boardcom BCM2711 Cortex-A72 négy magos, 64 bites processzor
- 1 GB, 2GB,4 GB vagy 8 GB LPDDR4-3200 SDRAM modelltől függően
- Gigabit Ethernet port
- 2,4 GHz-es és 5GHz-es vezeték nélküli LAN és Bluetooth 5.0 BLE
- 2 db USB 3.0 port és 1 db USB 2.0 port
- 40 GPIO pin
- 2 db micro HDMI port (4K támogatással)
- 3.5 mm-es Jack csatlakozó
- Micro SD card slot
- 5V USB-C tápegység.

2.8.2 Arduino

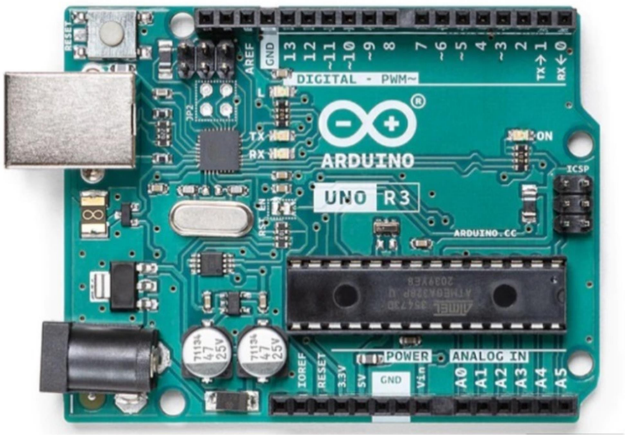
Az Arduino SBC számomra új jelenségként hatott, amikor belekezdtem a munkába. Eredetileg csak Raspberry Pi készülékekkel terveztem dolgozni, de rengeteg előnye van annak, ha Arduinokat is használok. Az én esetemben az a legfontosabb, hogy található rajta analóg bemeneti port, ugyanis ezeknek fontos szerepe van az adatok begyűjtésénél.

A piacon rengeteg különböző árkategóriába tartozó, különböző specifikációkkal rendelkező Arduino modell létezik. Figyelembe véve, hogy az ellátandó feladat kis számítási kapacitást igényel, a legolcsóbb eszközöket vettem figyelembe a kiválasztásnál.

Az Arduino hivatalos weboldalán jelen pillanatban három, körülbelül egy árspektrumba tartozó készüléket találtam. Ezek az Arduino UNO R3, az Arduino Leonardo, és az Arduino Micro. Ezek közül mind a három megfelel annak a követelménynek, hogy rendelkezzen analóg bemeneti porttal, gazdaságosan beszerezhetőek és egyszerűen működtethetőek.

2.8.2.1 Arduino UNO R3

Az Arduino UNO R3 a legelső Arduino SBC, az Arduino UNO egy továbbfejlesztett változata. Elődjéhez méltó módon ez az eszköz is tökéletes azoknak, akik még csak most kezdenek megismerkedni a számítástechnika e részével.

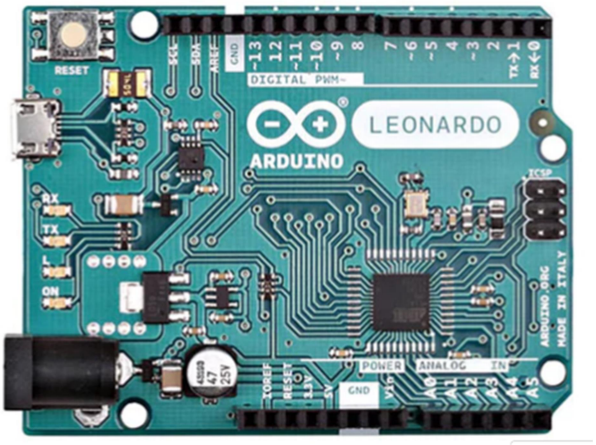


16. ábra Arduino UNO R3
Forrás: (arduino.cc, 2022)

A 16. ábrán látható Arduino UNO R3 eszközön egy ATmega328P mikrochip található, tizennégy digitális ki vagy bemeneti pinjéből hat képes PWM kimenetként működni. Hat analóg input pin található még rajta, illetve egy 16MHz-es CSTCE16M0V53-R0 kerámia rezonátor. A tápellátásról egy USB-B csatlakozó gondoskodik. Az oldalán található ISCP (In-Circuit Serial Programming) *header* az oldalán az SPI (Serial Peripheral Inteface) kommunikációt teszi lehetővé. Egy alternatív tápellátásnak egy jack, és található rajta még egy reset gomb.

2.8.2.2 Arduino Leonardo

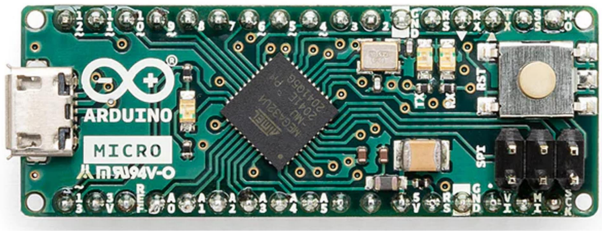
Az Arduino Leonardon egy ATmega32u4 mikrochip kapott helyet. Itt húsz digitális ki- és bementi pin található, de szintén csak 7 használható PWM-re. A tizenkét analóg input mellett itt is megtalálható az ISCP header, a reset gomb és a jack csatlakozó. A táp itt egy Micro USB csatlakozóval lett ellátva, a 16MHz-es órajelről pedig egy kristályoszcillátor gondoskodik.



17. ábra Arduino Leonardo
Forrás: (arduino.cc, 2022)

2.8.2.3 Arduino Micro

Az Ardunio Micro rengeteg mindenben hasonlít Az Arduino Leonardohoz. Ezen a mikrokontrolleren is ATmega32u4 mikrochip található, itt is 20 digitális és 12 analóg input van. Most is ott van a16 MHz-es kristályoszcillátor, a jack, az ISCP header és a reset gomb. A lényegi különbség ezen eszközök között a méretekben és kialakításban van.



18. ábra Arduino Micro
Forrás: (arduino.cc, 2022)

2.8.3 Rádiós chip

Ezen alfejezetben a lehetséges rádiós chipeket mutatom be.

2.8.3.1 NRF24L01+

Az NRF24L01+ egy 2.4 GHz-en működő tranceiver (küldeni és fogadni is képes egység egyben). Az átviteli sebessége 250 kbps-tól egészen 2 Mbps-ig terjed. Üzemeltetéséhez 1,9 és 3,6 V közötti feszültség szükséges, erre van 3.3 V-os táp kimenet mind az Arduino-n és a Pin egyaránt. Szabad téren körülbelül 1000 méter a hatótávolsága, így nem lehet gond a kapcsolattal a korai fázisban.

Az NRF24L01+ kifejezetten mikrokontrollerekkel történő működésre tervezték, tekintve, hogy vezérelhető SPI-on keresztül. (sparkfun.com, 2022)

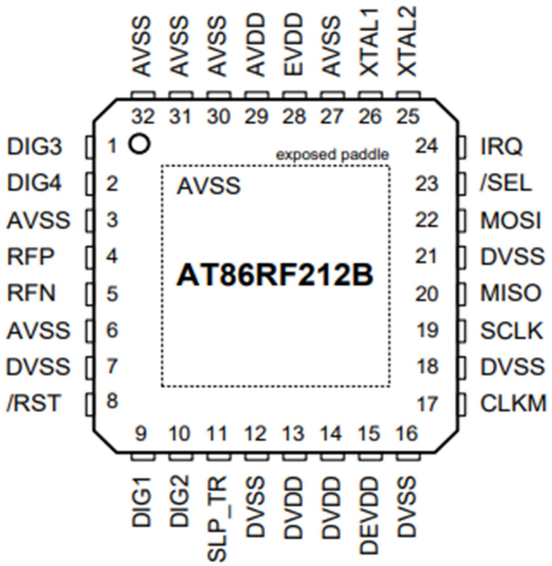
Nagy előnye, hogy nem csak egy mikrochip, a lábak kivezetésével könnyen lehet hozzájuk csatlakozni jumper kábelekkel. Rendelkezik saját antennával is, ezért erről sem nekünk sem gondoskodnunk.

2.8.3.2 MAX2830

A MAX2830 egy szintén 2.4 GHz-en működő tranceiver. Átviteli sebessége 11 Mbps vagy 54 Mbps lehet működtetéstől függően. 2.7 és 3.6 V között tápfeszültségre van szüksége a működéséhez, szóval ez itt is egyszerűen megoldható mind Arduino-val mind Raspberry Pi-vel. Az elérhető magas átviteli sebesség miatt ez egy drágább megoldás. Az NRF24L01+-tól eltérően nincs saját beépített antennája.

2.8.3.3 AT86RF212B

A 19. ábrán látható AT86RF212B egy alacsony feszültségen működő, 100kbps-tól 1 Mbps-ig terjedő átviteli sebességgel rendelkező tranceiver. Beépített 16 Mhz kristályoszcillátora biztosítja az órajelet a működéséhez. Sávszélessége 700 MHz-től 900 MHz-ig terjed. SPi-on keresztül megfelelő rádiós tranceiver interfész az antenna és a mikrokontroller között, de saját beépített antennája nincs.



19. ábra Az AT86RF212B tranceiver
Forrás: (microchip.com, 2022)

3 MEGVALÓSÍTÁS

Ezen fejezetben a bemutatom a kiválasztott eszközöket, a működtető programokat, valamint a 3D tervezés folyamatát.

3.1 Az eszközök kiválasztása

Ezen alfejezet a kiválasztott eszközöket hivatott bemutatni.

3.1.1 A Raspberry Pi, mint vezérlőegység az autón

A fent felsoroltak közül a Raspberry Pi 2-es modell éppen megfelel, mindenre képes, amire szükségünk van, és rendelkezik a kívánt portokkal és csatlakozókkal. A választás mégis a Raspberry Pi 3 Model B+ számítógép lett, ugyanis a Raspberry Pi 2 kifutó modell, ezért nehéz beszerezni. Továbbá, mivel a célom, hogy a jövőben fejlesztem a projektet így nem célszerű egy gyengébb készüléket választani. A Raspberry Pi 3 Model B+ megbízható

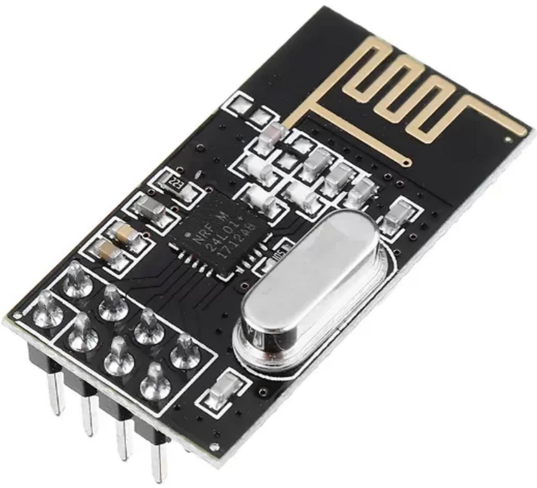
és gyors, nem véletlenül közkedvelt az iparban és az általános célú felhasználók között. A Pi 400, illetve a Pi 4 jelentős többletköltséget vontak volna maguk után.

3.1.2 Arduino UNO R3 mint adatokat gyűjtő és küldő egység

Az Arduino UNO R3-at választottam erre a célra, mert egyszerű kezelhetősége, rengeteg forrásanyaga és gazdaságos ára kedvezővé teszi számomra. Rengeteg könyv és cikk foglalkozik az Arduinoval és a legtöbb kezdőknek szóló részben ezt az eszközt javasolják, vagy egy korábbi Arduino UNO modellt. (Monk, Programming Arduino: Getting Started with Sketches, 2016) (Shiloh & Banzi, 2015)

3.1.3 NRF24L01+ mint rádiókommunikációs chip

A 20. ábrán látható NRF24L01+ nagyon jól dokumentált, sok forrás érhető el hozzá. rengeteg példaprogram megtalálható az interneten. Megbízható működésének köszönhetően



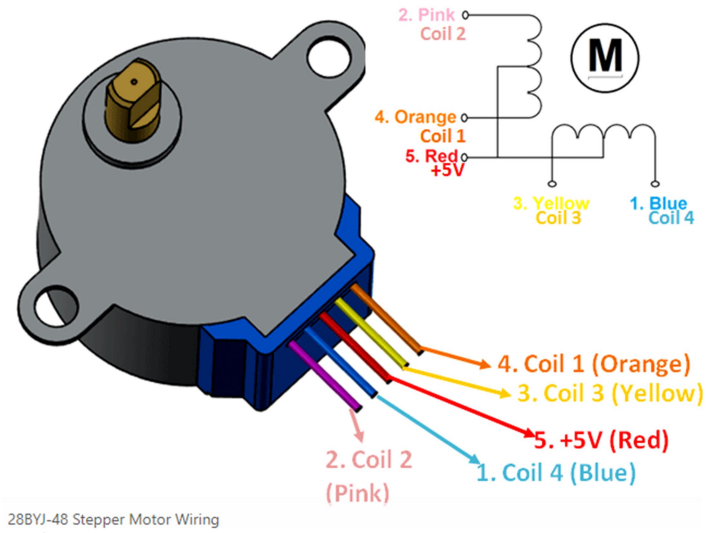
20. ábra NRF24L01+ tranceiver
Forrás: (howtomechatronics.com, 2022)

a világon mindenhol szívesen használják. Számunka nem annyira fontos a magas átviteli sebesség, és nagyon előnyös, hogy a tervezés elején is egyszerűen lehet dolgozni a chippel. Csak csatlakoztatjuk a megfelelő kimenetekre és ma megfelelő lib_nrf24 könyvtár telepítése után már használható is. Az általunk végzett teszteken stabil jelátvitelt biztosított 100m távolságból is.

3.1.4 Léptetőmotor és vezérlése

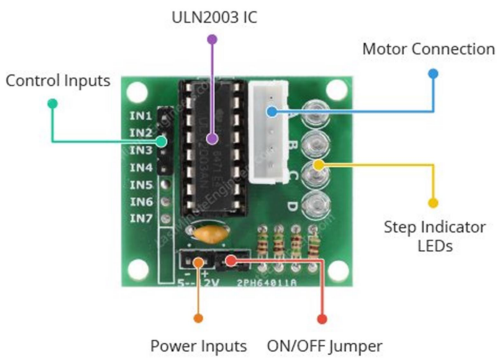
Az általam választott léptetőmotor a 28BYJ-48 látható a 21. ábrán. Ez egy nagyon elterjedt léptetőmotor. Találkozhatunk vele DVD meghajtókban, és más hasonló

eszközökben. A motor négy, unipoláris elrendezésű tekerccsel rendelkezik, tekercsenként 5V névleges feszültsége miatt egyszerűen vezérelhetők mikrokontrollerekkel.



21. ábra 28BYJ-48 Léptetőmotor bekötése
Forrás: (components101.com, 2022)

Ehhez a léptetőmotorhoz megfelelő választás a 22-ábrán látható ULN2003 motorvezérlő. A motor csatlakozó egységét egyszerűen csatlakoztatjuk az ULN2003 erre kialakított portjához. Ezek után már csak a vezérlőt kell csatlakoztatni a Raspberry Pi-hez, és a megfelelő bemenetekre jelet küldve megvalósítható a motor léptetése. A lépéseket a vezérlőn elhelyezett LED-ek segítségével nyomon is követhetjük. Elvégeztem egy tesztet, mely során vizsgáltam a léptetőmotor lépéseinek pontosságát. Megjelöltem a ráérősített fogaskerék egy pontját, majd a léptetőmotor megtett ezer teljes fordulatot mindkét irányba. A teszt végén a megjelölt pontot összevetettem a viszonyítási ponttal. A léptetőmotor milliméter pontosan visszaállt a kiindulási pontjára.

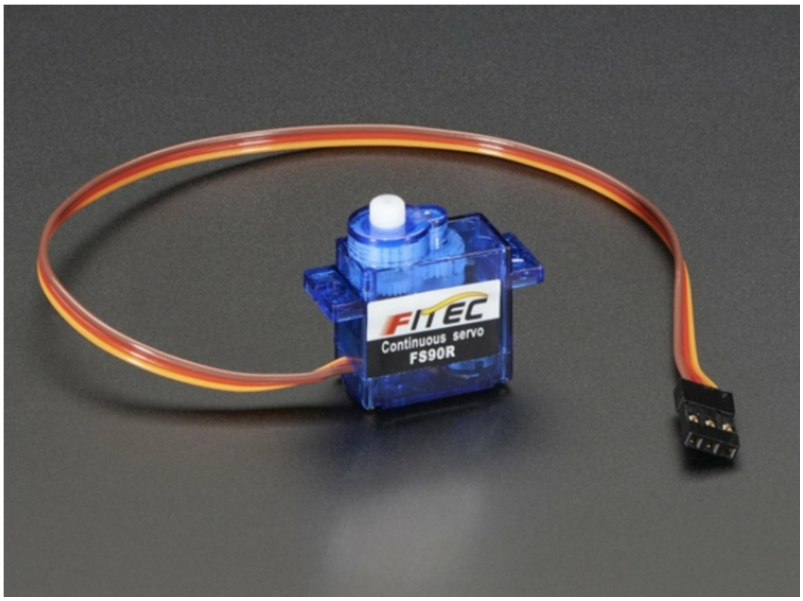


22. ábra Az ULN2003 léptetőmotor vezérlő
Forrás: (lastminuteengineers.com, 2022)

3.1.5 Szervomotor és vezérlése

Szervomotornak a 23. ábrán bemutatott A2442 Continuous Micro Servo Motort választottam. Az FS90R a legtöbb szervomotortól eltérően nem 180, hanem 360°-os mozgástartománnyal rendelkezik. Ez elengedhetetlen számunkra, hiszen nekünk ez fogja biztosítani a hajtást.

Ezen szervomotor vezérléséhez nincs szükség külön vezérlőre, egyszerűen csatlakoztatjuk a vezérlő vezetéket a Raspberry Pi egy PWM-re képes kimenetére, és egy egyszerű beállítás után már lehet is vezérelni a motort.



23. ábra Az FS90R Continuos servo motor
Forrás: (malnapc.hu, 2022)

3.1.6 Pozícióérzékelés

Ebben a projektben az pozícióérzékelés az adatok begyűjtésénél játszik szerepet. Mind a 24. ábrán látható kormányban, ami egy Wingman Formula GP, mind pedig a hozzá tartozó pedálokban egy 100 kΩ-os potenciométer található. A potenciométer feszültségosztóként viselkedik. Működés során az Arduino a potenciométer egyik végét megtáplálja 5V-al, a másik végét pedig csatlakoztattam a 0-hoz (GND-föld). Ezáltal a harmadik vezetéken mérhető az ellenállással arányos pozíciója a csúszkának. Amikor eltekerem a kormányt, az két rugó segítségével elforgatja a potenciométert, amennyiben az erőhatás megszűnik, a rugók visszatérítik a kormányt eredeti állapotába, a potenciométer eltekerése megváltoztatja a csúszka helyét, ami a feszültségjel megváltozását okozza.

A középső jelvezetéket csatlakoztattam az Arduino A0-ás analóg portjához, és az azon érkező feszültségjelekkel figyelhető meg a kormány forgási pozíciója.

A pedálok esetében is ugyanígy járok el. Mind a két pedál egy-egy potenciométert forgat el, egy rugó ellenében. Ugyanúgy megtápláljuk őket az Arduino segítségével, és a másik végüket a GND-hez (Ground-Föld) csatlakoztatom, tehát a jelvezetéken itt is a csúszka pozíciójával arányos a feszültségjelek keletkeznek a jelvezetékeken, ezek az Arduino A1-es és A2-es analóg portjaihoz lettek csatlakoztatva.

24. ábra A Wingman Formula GP és a potenciométert forgató rugók
Forrás saját fotók

3.2 A működtető szoftver

A szakdolgozat ezen szakaszában bemutatom azokat a programkódokat, amelyek működtetik az eszközöket, mint az Arduino UNO R3-at és a Raspberry Pi 3 Model B+-t.

3.2.1 Raspberry Pi-n futó programok

A feladatokat felosztottam öt különböző részre. Minden részt egy külön Python fájl valósít meg. Ezek az adatok fogadása, a beérkezett adatok feldolgozása és ezek alapján a szervo és léptetőmotorok vezérlése. Az ötödik az úgy nevezett main.py ami összefogja az előzőeket egy közös programba. Ezt futtatjuk a Raspberry Pi-n amikor működik. Minden programot egy külön folyamatábrában mutatom be.

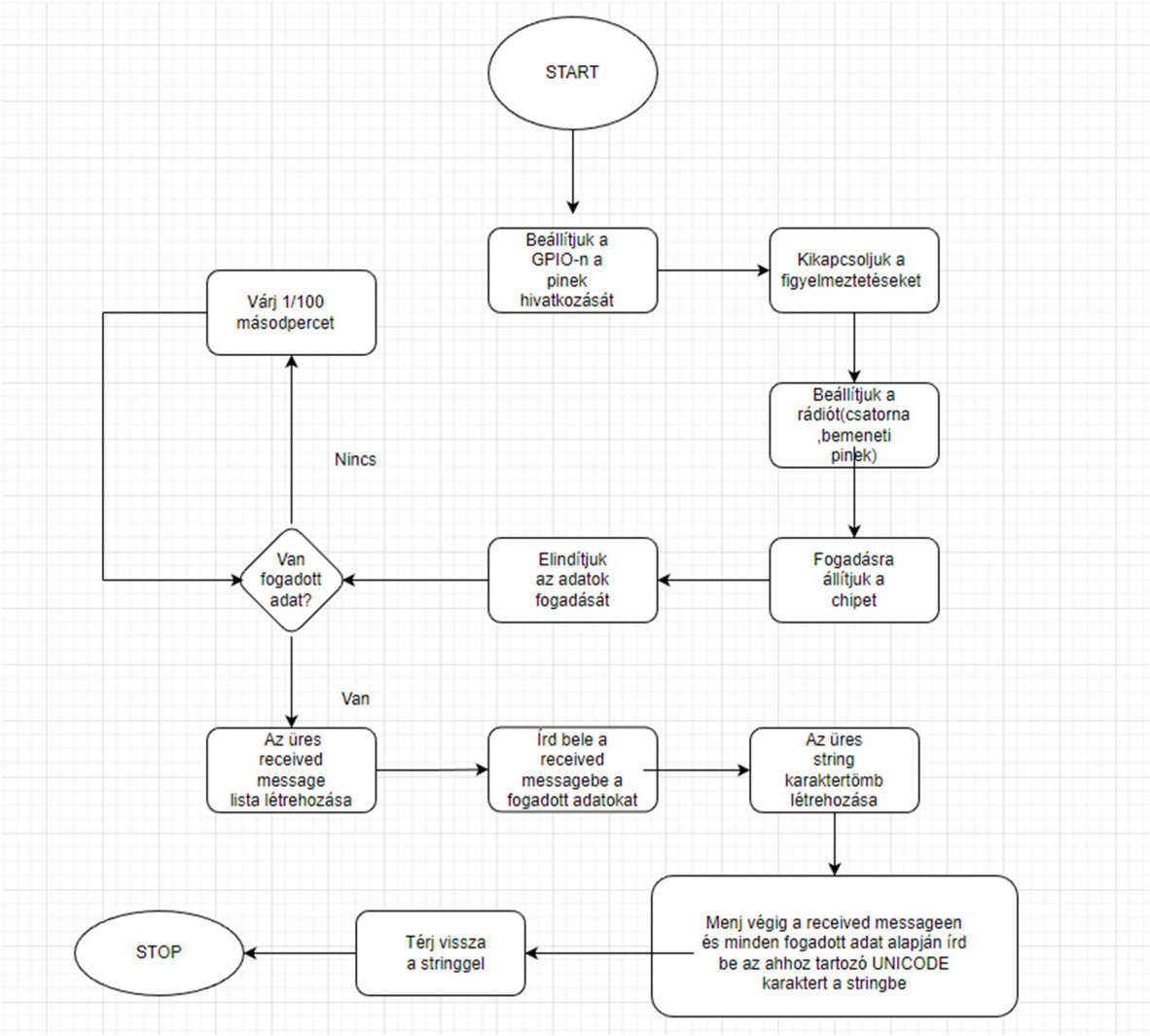
3.2.1.1 Get_Radio_Data.py

Az első lépés az adatok fogadása az Arduinoról. Ehhez a fent említett NRF24L01+ chip szoftveres működtetésére van szükségünk. Ehhez le kell töltenünk az internetről két python könyvtárat. Az első a Spidev, ami az SPI kommunikációt teszi lehetővé a chip és a Raspberry Pi között. Ez előre letölthető formában megtalálható a GitHubon.

A másik könyvtár a lib-nrf24, ami egy előre megírt program kifejezetten az NRF24L01+ chip üzemeltetéséhez. Ezt a könyvtárat is a GitHubon lehet megtalálni, ingyenesen letölteni és használni. A programozás kezdete előtt mindkét könyvtárat fel kell telepíteni a Raspberry Pi-re.

A szükséges fájlokat megtalálhatjuk a https://github.com/doceme/py-spidev valamint a https://github.com/BLavery/lib_nrf24 oldalakon.

A 25. ábra bemutatja a program folyamatábráját:



25. ábra A Get_Radio_Data folyamatábrája
Forrás: Draw.io saját munka

A GPIO könyvtár két lehetőséggel rendelkezik a pinek hivatkozását illetően. Ez a különböző ki vagy bemenetekre való utaláskor segít nekünk egyértelműen megadni, melyik pint akarjuk vezérelni, amikor kiadunk egy utasítást. Létezik a GPIO.BOARD, ilyenkor egy-egy pin fizikai helyét adjuk meg, hogy hányadik is a sorban, a bal felső pin az első, a jobb felső a második és így tovább. Ha ilyen módban adjuk meg például, hogy a huszonharmadik pin legyen kimenet, akkor a jobb oldali tizenkettedik pin-t állítjuk be.

Ha a GPIO.BCM lehetőséget választjuk akkor a Raspberry Pi egy előre beállított pin kiosztását használjuk. Ez a kiosztás megtalálható Raspberry Pi weboldalán. Ha így hivatkozunk a huszonharmadik pinre, tehát a GPIO23-as kimenetre, akkor a jobb oldali nyolcadik pint állíthatjuk be. Látható, hogy fontos ezt helyesen megadni.

A rádió beállítását a lib_nrf könyvtár különböző parancsai segítségével tehetjük meg. Beállítható a csatorna száma, az adatsebesség, az energiafogyasztás és rengeteg minden más is. Minden megtalálható az említett könyvtárban.

A python kód teljes egészében megtalálható az 1-es számú mellékletben.

3.2.1.2 Data_Handle.py

A következő lépés az adatok fogadása után, hogy az Arduino által küldött számokat értelmezzük, és az autó mozgatásához megfelelő adatokká konvertáljuk.

Az Arduino analóg bementén az 5V-os feszültséget egy tízbites számmá alakítja, azaz egy 0-1023 közötti számot küld. Elméletben tehát ezt fel lehetne osztani egyenlő tartományokra, és ha a küldött szám a tartományon belül van, akkor az ahhoz rendelt számot adná vissza a program. Gyakorlatban is e logika mentén haladok, de a fogadott értékek nem teljesen követik ezt le. Elméletben a teljesen balra tekert kormány 0-át a teljesen jobbra tekert pedig 1023-at adna vissza, ám a potenciométer kopása és egyéb technikai okok miatt nem kaptam 90-nél kisebb és 950-nél nagyobb értéket.

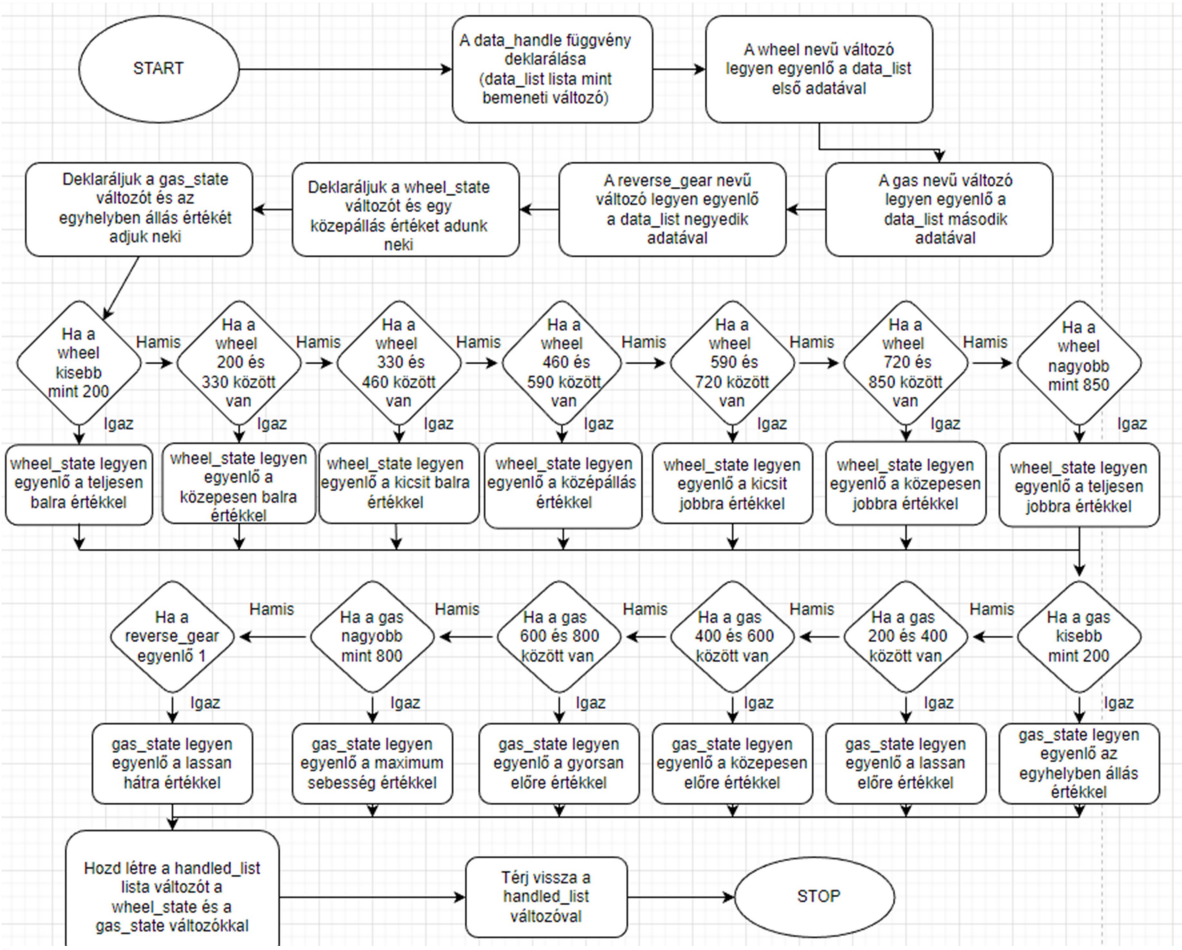
Ezt a hibát kiküszöböltem azáltal, hogy a két szélső régiót nagyobbnak választottam, és a köztes tartományokat osztottam fel egyenlő részekre. Mivel a kerekek három különböző állást vehetnek fel oldalanként, ez az egyenesen előre állással kiegészülve hét lehetséges állapotot jelent. A két szélső határt 200-ra és 850-re választottam. Az öt köztes régió méretét pedig a következő egyenlettel állapítható meg:

$$850 - 200 = 650 \quad \frac{650}{5} = 130$$

Tehát a köztes régiók 130 egységenként lettek felosztva.

A pedálokról kapott jel esetében is hasonlóan jártam el. Itt kevesebb tartományra kell felosztani a kapott adatod, mivel csak öt különböző sebességet állapítottam meg. A hátramenetet egy külön érkező szám, a reverse_gear adja meg. A folyamatábra megtalálható a 26. ábrán.

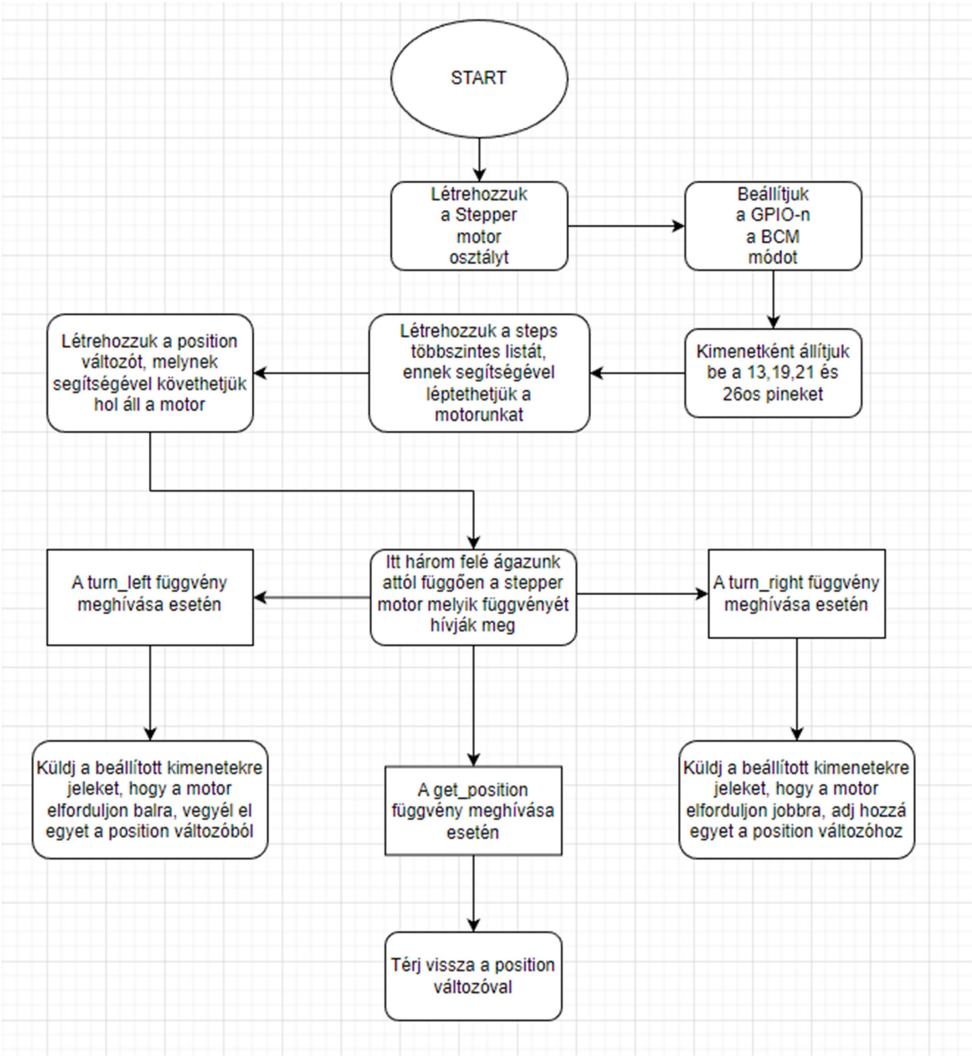
A teljes program a 2-es számú mellékletben megtalálható.



26. ábra A Data_Handle folyamatábrája
Forrás: Draw.io saját munka

3.2.1.3 A Stepper_Motor_Class.py

A következő lépések során a léptetőmotor és a szervomotor szoftveres vezérlése. Ezeket objektum orientál programozási (OOP) módszerrel oldottam meg. Az OOP leegyszerűsítve lehetővé teszi számunkra bizonyos objektumok létrehozását, ezekhez társíthatunk különböző attribútumokat vagy tulajdonságokat, illetve metódusokat. Az objektumban



27. ábra A Stepper_Motor_Class folyamatábrája
Forrás: Draw.io, saját munka

található adatokkal az objektum metódusai végeznek műveleteket. A Python nem csak a procedurális (függvényekkel, vagy szubrutinokkal történő programozás) programozást támogatja, hanem az OOP-t is. Itt úgynevezett Class-ok létrehozására van lehetőségünk. (Phillips, 2010)

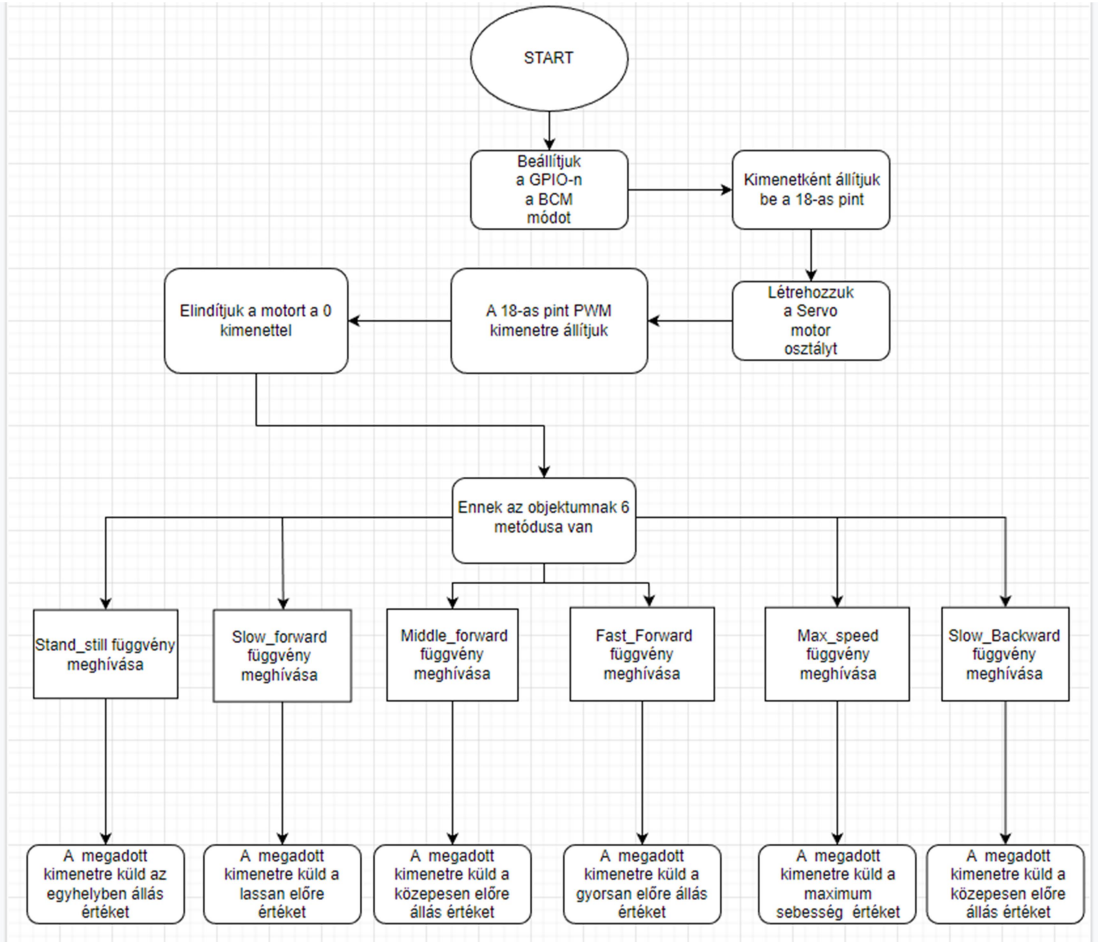
Ez abban nyújt segítséget, hogy a main.py soraiban létrehozhatunk majd egy stepper_motor és servo_motor objektumokat, amennyiben importáljuk ezeket. Ezen objektumok segítségével pedig az előre létrehozott metódusaikat hívhatjuk meg, amik vezérelni fogják a motorokat. A stepper_motor esetében ezek a balra kanyarodás, jobbra kanyarodás, valamint a pozíció lekérése. Az objektum legelején az úgy nevezett konstruktor (__init__) azokat a sorokat tartalmazza, amelyek egy objektum létrehozásakor futnak le. A program folyamatábrája a 27. ábrán látható.

A teljes kód megtalálható a 3-as számú Mellékletben.

3.2.1.4 A Servo_Motor_Class.py

Az előzőekben már említettem az OOP-t és annak előnyeit. A szervomotort vezérlő program egy servo_motor objektumot hoz létre. A metódusok a különböző sebességekhez tartoznak, amelyekkel a szervomotor foroghat. A folyamatábra a 28. ábrán megtalálható.

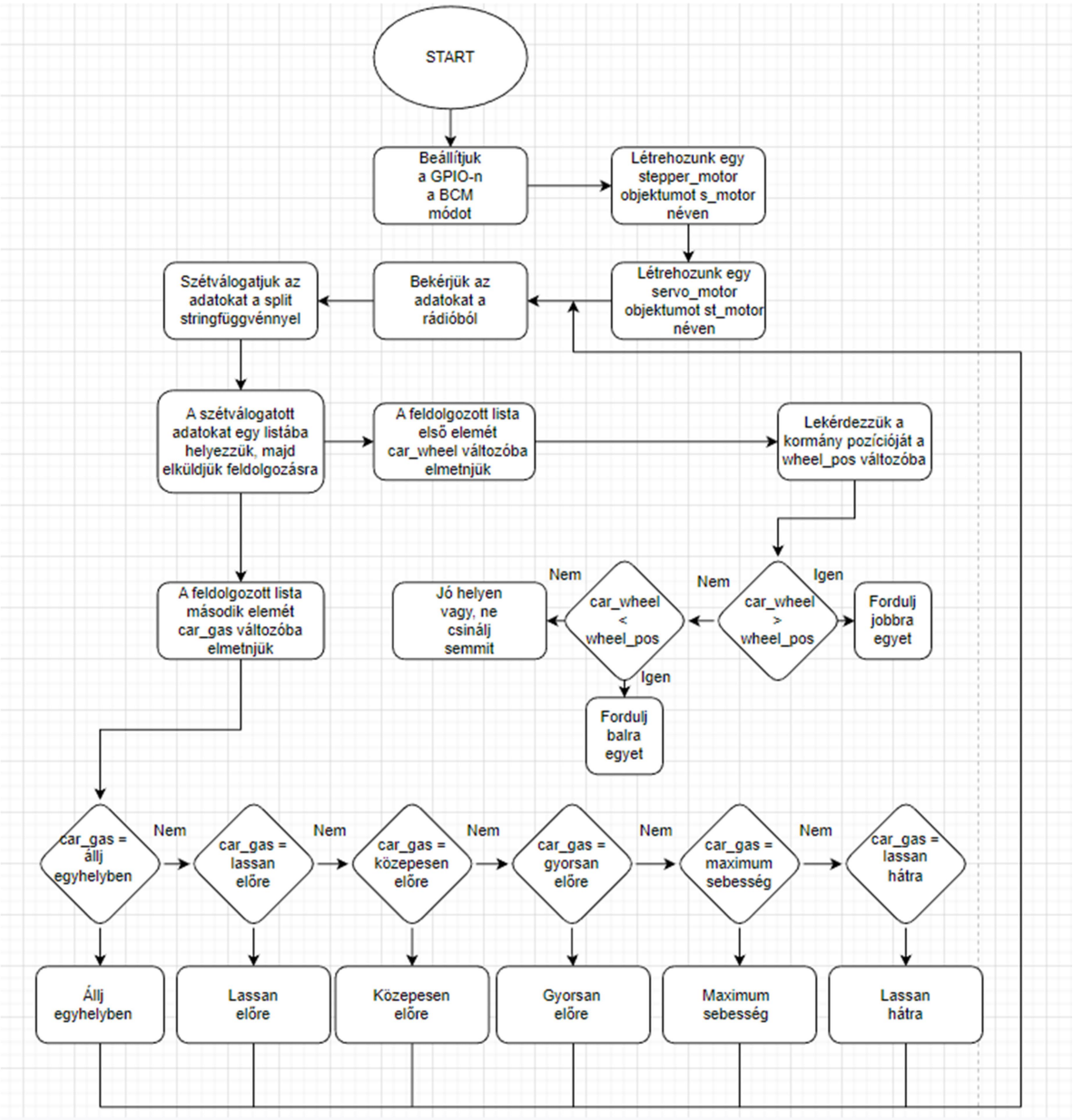
A 4-es számú Melléklet tartalmazza ezt a kódot.



28. ábra A Stepper_Motor_Class folyamatábrája
Forrás: Draw.io, saját munka

3.2.1.5 A main.py

A main.py az a program, amit a Raspberry Pi futtat. Folyamatábrája a 29. ábrán megtekinthető. Ahhoz, hogy minden eddig leírt függvényt és objektumot használhassam, az összes eddig fájlból importálnunk kell a szükséges részeket. Mindezek mellett két, a program olvasását megkönnyítő, konstans Dictionary lett létrehozva. Ez az adattípus elempárokat tartalmaz, egy kulcsot és egy értéket. Az utóbbit úgy kapjuk meg, ha a hozzá tartozó kulccsal hívjuk meg a Dictionary-t. A Dictionary nincs nagy hatással a futásidőre,



29. ábra A main.py folyamatábrája
Forrás Draw.io, saját munka

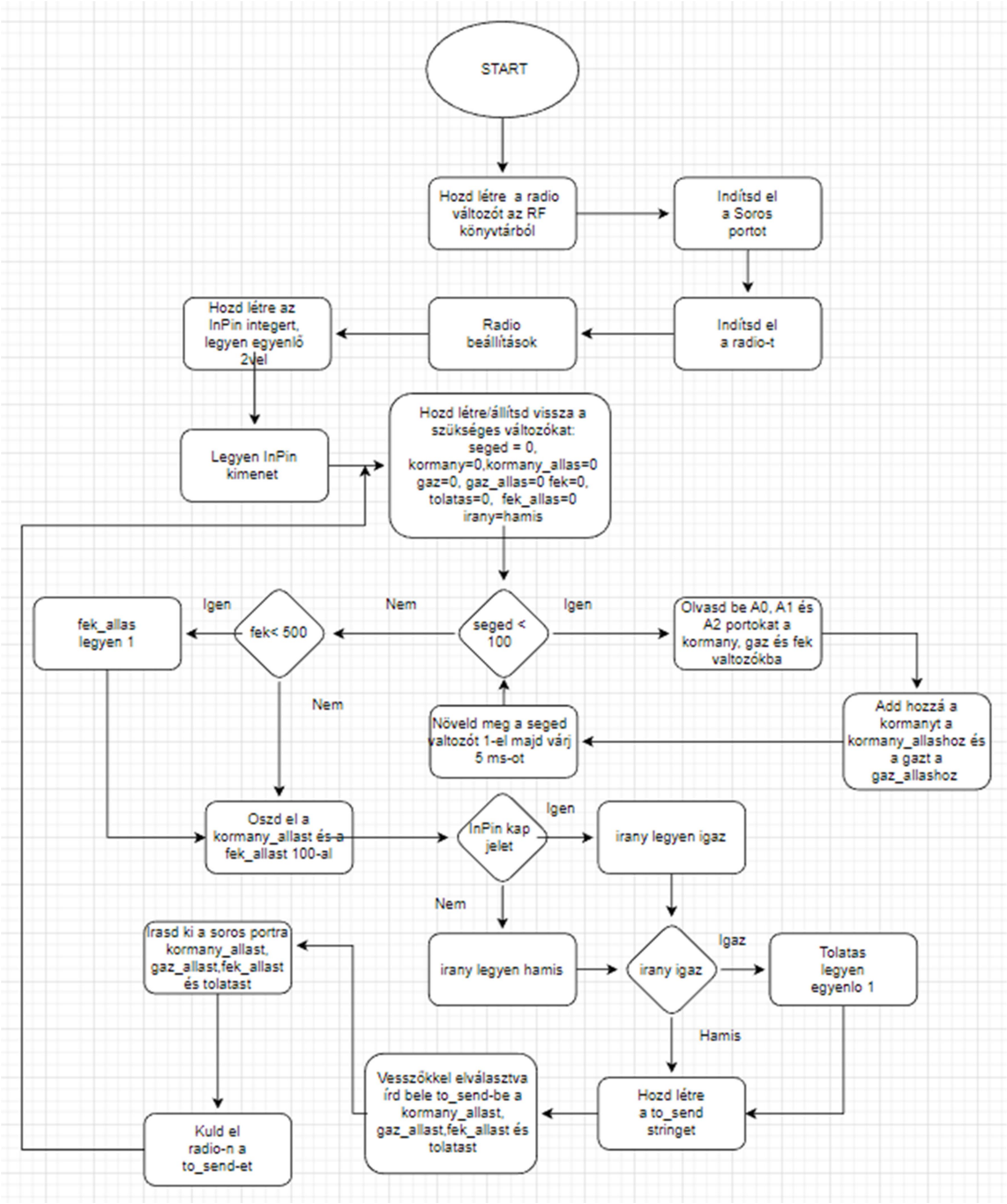
de egyszerűbbé teszi az értelmezést, ha nem csak egy integer van a programba, hanem egy szöveg, ami a kulcs az integerhez és érthetővé teszi a programrészletet.

3.2.2 Az Arduino szoftvere

Az Arduinora szánt programok megírása előtt a már említett Arduino IDE-t kellett telepítenem a számítógépemre. Ezt követően, csakúgy, mint a Raspberry Pi esetében, itt is először telepítenem kell egy RF 24 nevű könyvtárat. Ez fog segíteni az NRF24L01+ vezérlésében. A telepítéshez csak követnem kellett az ezen az oldalon található utasításokat, https://www.arduino.cc/reference/en/libraries/rf24/

A telepítés után már csak a program volt hátra. Az Arduino IDE alapértelmezett nyelvét, egy C++ alapjaira épített programozási nyelvet használtam. Ennek a programnak a feladata, hogy begyűjtse az adatokat, majd elküldje őket a Raspberry Pi-nek.

A most következő 30. ábrán látható a folyamatábrán látható, hogy a porgram végén nem teljesen az elejére térünk vissza.



30. ábra Az Arduinocode.ino folyamatábrája
Forrás: Draw.io, saját munka

Minden Arduino-n futó program két részből áll, ez jól látható a 4. ábrán. Az első rész az úgynevezett beállítások(setup), a második pedig a hurok (loop). A setup-ban található parancsok csak egyszer futnak le, míg a loop az utolsó sorának lefutása után egyből visszaugrik a loop első sorára. Ez a funkció megjelenik a Raspberry Pi-n futó main.py-ban is, ott egy amíg igaz (while True:) végtelen hurok szolgál erre a célra. Az Arduino-n futó Arduinocode.ino kódot a 6-os számú melléklet tartalmazza.

3.3 A 3D tervezett váz

Ebben az alfejezetben a 3D tervezett vázat mutatom be.

3.3.1 A tervező szoftver

Az autó vázát Solid Edge 2022-ben terveztem meg. A Solid Edge egy Windowson futó 3D-s CAD (computer-aided design), elsősorban a gépiparban használják. Az első verziója a Solid Edge 1 1995-ben jelent meg, akkor még az Intergraph kiadásában. 2007-óta a Siemens Digital Industries Software fejleszti. A Solid Edge kétféle modellezési megoldást kínál:

3.3.1.1 Hagyományos modellezés

A Hagyományos modellezés már a szoftver legelső verzióiban is megtalálható volt. Ez a technológia modelltörténeten alapuló parametrikus modellezés alaksajátosságokkal. Az alaksajátosságok, ilyen például a kihúzás vagy a kivágás, létrehozásához először egy vázlatot kell létrehozni, ez fog az alaksajátosság alapjaként szolgálni. Ha módosítjuk a vázlatot, módosul vele a test is, amit létrehoz.

A modelltörténeten alapuló azt jelenti, hogy ha az alaksajátosságok sorában elől szerkesztünk vázlatot, akkor az összes, utána következő elemet újraszámolja.

3.3.1.2 Szinkron modellezés

Ez a technológia csak 2008 óta létezik. Itt parametrikus alakelem alapú modellezésről beszélhetünk. Ezzel a módszerrel közvetlenül a modellt változtathatjuk, bár itt is szükség van a vázlatra az nem teszi kötötté a modellt. Nincs újraszámolásra szükség, ez pedig megszünteti a modelltörténeti kötöttségeket.

3.3.2 A felhasznált elemek

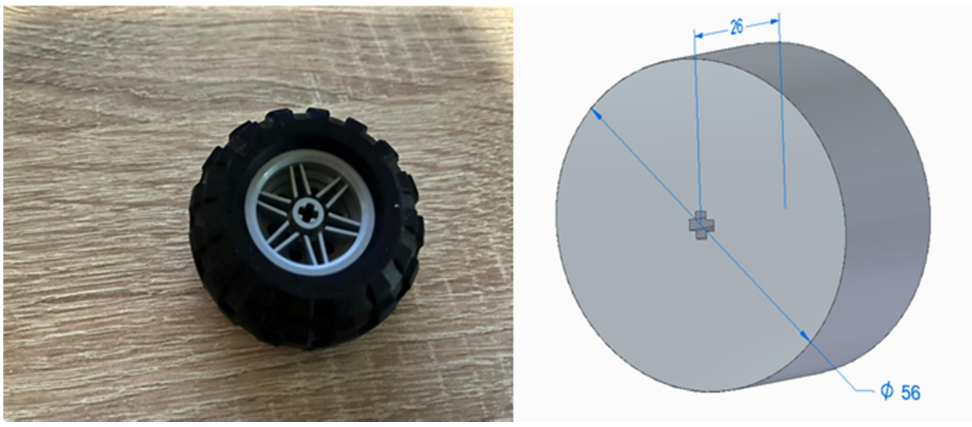
Nem az összes az autóban lévő elem lett tervezve és nyomtatva. A léptetőmotor, a szervomotor, a kerekek, a fogaskerekek valamint maga a Raspberry Pi is megvolt már.

Nekem ezek összefogására képes autó vázat kellett terveznem. A már létező elemek alapján egyszerűsített alkatrészeket készítettem, és ezek segítségével terveztem meg a vázat.

3.3.2.1 A kerekek

Az autó kerekeinek az 56mm átmérőjű, 26mm magasságú abroncsot és a hozzá tartozó 30,4mm átmérőjű és 20 mm magasságú műanyag felnit választottam.

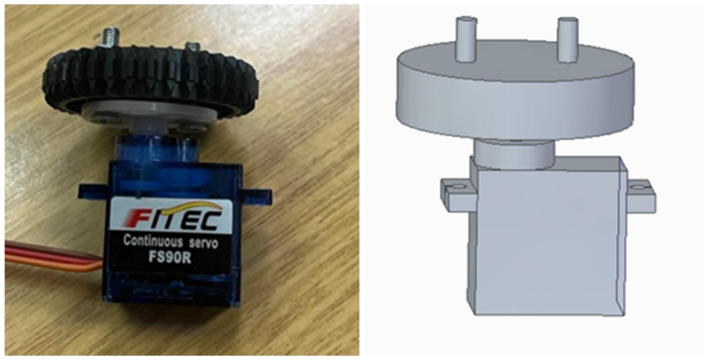
Az egyszerűsített Solid Edge alkatrész egy 56 mm átmérőjű henger.



31. ábra A kerék és az egyszerűsített alkatrész
Forrás: Solid Edge, saját munka

3.3.2.2 A Szervomotor a ráerősített fogaskerékkel.

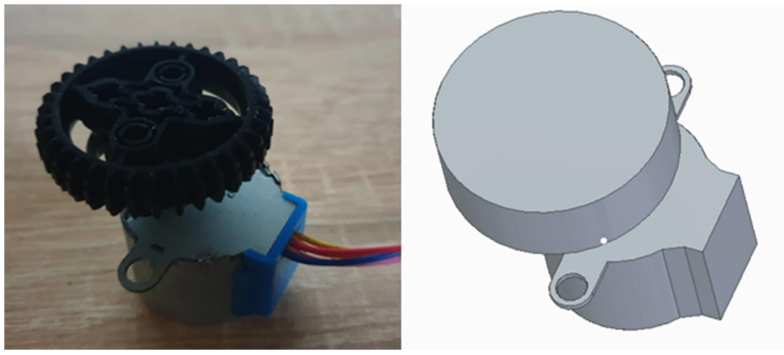
A szervomotor alkatrészét már a csavarokkal ráerősített fogaskerékkel együtt készíttettem. A fogaskerék adatai: fogszám: z = 36, modul: m = 1.



32. ábra A szervomotor a fogaskerékkel és az alkatrész
Forrás: Solid Edge, saját munka

3.3.2.3 A léptetőmotor a ráerősített fogaskerékkel

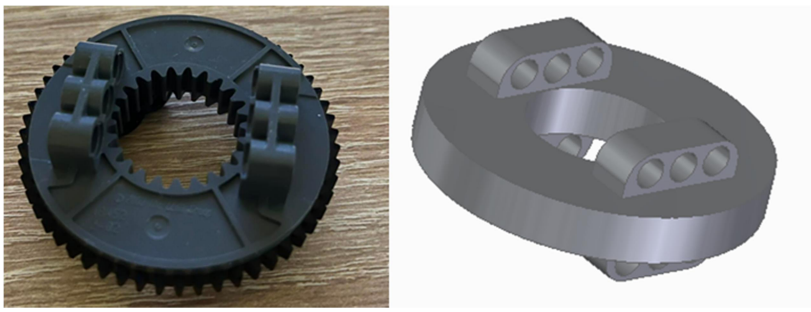
A léptetőmotor alkatrésze szintén a ráerősített fogaskerékkel együtt készült.A fogaskerék ugyanaz, mint a szervomotor esetében.



33. ábra A léptetőmotor a fogaskerékkel és az alkatrész
Forrás: Solid Edge, saját munka

3.3.2.4 A csapágyas fogaskerék

A kerekek forgatását egy elfordulásra képes fogaskerékkel oldottam meg. Az egyik oldalát a vázhoz erősítettem, a másikon található fogaskereket pedig csatlakoztattam a léptetőmotorra erősített fogaskerékhez. A csapágyas fogaskerék adatai: fogszám: z =56, modul: m=1



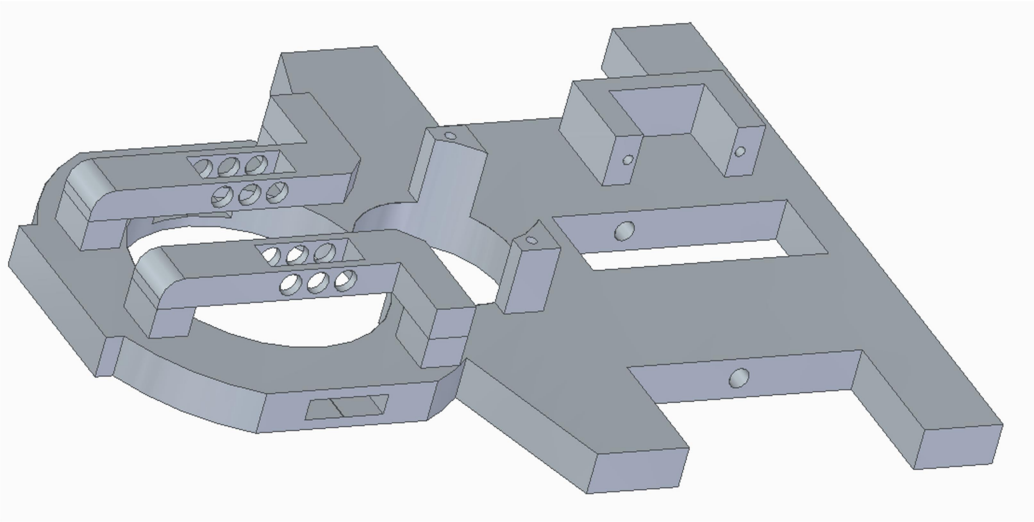
34. ábra A csapágyas fogaskerék

Forrás: Solid Edge, saját munka

3.3.3 A tervezett váz

Az eddig bemutatott elemek segítségével, megterveztem a vázat. Először az alváz készült el. Ehhez lehet rögzíteni a motorokat, a kerekeket és a kerekek forgatására használt csapágyas fogaskereket. Egy 150mm széles, 200mm hosszú 10 mm vastag téglatest volt a kiindulási alap. Ebből ki kellett vágni a kerekek helyét, ami főleg az első kerekek esetében volt problémás, tekintettel arra, hogy kanyarodás közben a tengely is elfordul. Szükséges volt elég helyet biztosítani a tengelynek és az első kerekek számára is mindkét irányba a maximális elfordulás lehetővé tételéhez.

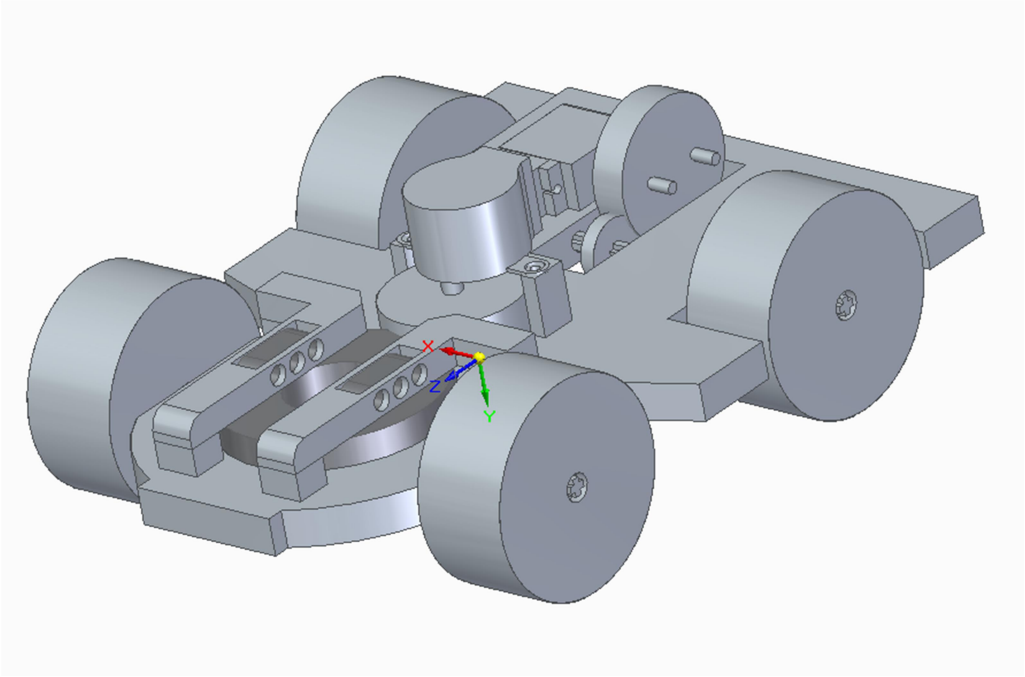
A léptető és szervo motoroknak és a kialakítottam egy-egy csatlakozót, amelyekhez csavarral lehet erősíteni őket. A tervezett vázat a 35. ábra mutatja be



35. ábra A tervezett váz
Forrás Solid Edge, saját munka

3.3.4 Az összeszerelési ábra

A 36. ábrán látható az összeszerelés ábra, amelyen minden eddig bemutatatott alkatrész látható.



36. ábra Az összeszerelési ábra
Forrás Solid Edge, saját munka

3.3.5 A 3D nyomtató

A váz nyomtatására egy ENDER 5 Plus 3D nyomtatót használtam. Az Creality által fejlesztet ENDER 5 Plus egy FFF nyomtatási technológiát támogató, nagy (350mm x 350mm x 400mm) nyomtatótérrel rendelkező, könnyen kezelhető 3D nyomtató.

3.3.6 A Filament

Azt az anyagot, amelyet egy FFF (Fused Filament Fabrication) technológiás 3D nyomtató megolvaszt, majd egy fúvókán keresztül a platformra juttat filamentnek nevezzük. A legfontosabb filament típusok közé az ABS, a PETG és az általunk is használt PLA tartoznak.

4 GAZDASÁGI SZÁMÍTÁSOK

Ezen fejezet foglalkozik a gazdasági számítások bemutatásával.

4.1 Felhasznált alkatrészek és anyagok

Az autó építéséhez használt alkatrészek és anyagok gazdasági vizsgálatát az alábbi táblázat tartalmazza.

1. táblázat Gazdasági számítások

Megnevezés	Egységár	Darabszám	Összesen	Megjegyzés
Raspberry Pi	25000	1	25000	Raspberry Pi Model 3 B+
Arduino	10080	1	10080	Arduino UNO R3
FS90R	3500	1	3500	Szervomotor
SM-28BYJ-48	925	1	925	Léptetőmotor
SM-ULN2003	510	1	510	Léptetőmotor vezérlő
CLF-ECOPLA-BLACK-1000-175	10170	1	10170	EcoPLA fekete filament 1,75mm1000g
Lego elem: 32498	400	2	800	36 fogú Fogaskerék
Lego elem 4018	60	1	60	16 fogú fogaskerék
Lego elem: 48452cx1	1200	1	1200	Csapágyas fogaskerék
Lego elem: 70695	576	4	2304	Kerék felni
Lego elem: 56145	133	4	532	Kerékabroncs
Lego elem : 50450	1700	2	3400	Tengely
Jumper Kábelek	30	10	300	Anya-anya jumper kábel
Jumper kábelek	30	20	600	Anya-Apa jumper kábel
Összesen			59381	

Az összes anyagköltség tehát 59.381 Ft.

4.2 Munkadíj

Az elvégzendő feladatokat egy mérnök, körülbelül 2 hét, azaz 10 munkanap alatt gond nélkül abszolválhatja. Napi 8 órával számolva ez 80 óra munkát jelent. Egy kezdő gépészmérnök 500.000 Ft-os havi jövedelmével számolva ez 250.000 Ft munkadíjat jelent. (fizetesek.hu, 2022)

A teljes költség $59.381 + 250.000 = 309381\ Ft$

5 NYILATKOZTOK

MAGYAR AGRÁR- ÉS ÉLETTUDOMÁNYI EGYETEM

Szent István Campus, Gödöllő
Cím: 2100 Gödöllő, Páter Károly utca 1.
Tel.: +36-28/522-000
Honlap: https://godollo.uni-mate.hu

NYILATKOZAT

Alulírott Meczegyi Norbert, a Magyar Agrár- és Élettudományi Egyetem, Szent István Campus, mechatronikai mérnök szak nappali/levelező* tagozat végzős hallgatója nyilatkozom, hogy a dolgozat saját munkám, melynek elkészítése során a felhasznált irodalmat korrekt módon, a jogi és etikai szabályok betartásával kezeltem. Hozzájárulok ahhoz, hogy Záródolgozatom/Szakdolgozatom/Diplomadolgozatom egyoldalas összefoglalója felkerüljön az Egyetem honlapjára és hogy a digitális verzióban (pdf formátumban) leadott dolgozatom elérhető legyen a témát vezető Tanszéken/Intézetben, illetve az Egyetem központi nyilvántartásában, a jogi és etikai szabályok teljes körű betartása mellett.

A dolgozat állam- vagy szolgálati titkot tartalmaz: igen nem*

Kelt: 2022 év 11 hó 7 nap

Meczegyi Norbert
Hallgató

NYILATKOZAT

A dolgozat készítőjének konzulense nyilatkozom arról, hogy a Záródolgozatot/Szakdolgozatot/Diplomadolgozatot áttekintettem, a hallgatót az irodalmi források korrekt kezelésének követelményeiről, jogi és etikai szabályairól tájékoztattam.

A Záródolgozatot/Szakdolgozatot/Diplomadolgozatot záróvizsgán történő védésre javaslom / nem javaslom*.

A dolgozat állam- vagy szolgálati titkot tartalmaz: igen nem*

Kelt: 2022 év 11 hó 7 nap

Belső konzulens

***Kérjük a megfelelőt aláhúzni!**

NYILATKOZAT

a szakdolgozat[1] nyilvános hozzáféréséről és eredetiségéről

A hallgató neve: Meggesi Norbert

A Hallgató Neptun kódja: OLQTFS

A dolgozat címe: Elektronikus autó vezérlése mikrokontrollerekkel

A megjelenés éve: 2022

A tanszék neve: Műszaki Intézet, Mechatronikai Tanszék

Kijelentem, hogy az általam benyújtott szakdolgozat[2] egyéni, eredeti jellegű, saját szellemi alkotásom. Azon részeket, melyeket más szerzők munkájából vettem át, egyértelműen megjelöltem, s az irodalomjegyzékben szerepeltettem.

Ha a fenti nyilatkozattal valótlant állítottam, tudomásul veszem, hogy a Záróvizsga-bizottság a záróvizsgából kizár és a záróvizsgát csak új dolgozat készítése után tehetek.

A leadott dolgozat, mely PDF dokumentum, szerkesztését nem, megtekintését és nyomtatását engedélyezem.

Tudomásul veszem, hogy az általam készített dolgozatra, mint szellemi alkotás felhasználására, hasznosítására a Magyar Agrár- és Élettudományi Egyetem mindenkori szellemitulajdon-kezelési szabályzatában megfogalmazottak érvényesek.

Tudomásul veszem, hogy dolgozatom elektronikus változata feltöltésre kerül a Magyar Agrár- és Élettudományi Egyetem könyvtári repozitori rendszerébe.

Kelt: 2022 év 11 hó 7 nap

Meggesi Norbert

Hallgató aláírása

[1] A megfelelő dolgozattípus meghagyása mellett a többi típus törlendő.
[2] A megfelelő dolgozattípus meghagyása mellett a többi típus törlendő.

KONZULTÁCIÓS
NYILATKOZAT

A Ökrösi Norbert (név) (hallgató Neptun azonosítója: OØQTFS) konzulenseként nyilatkozom arról, hogy a szakdolgozatot[1] áttekintettem, a hallgatót az irodalmi források korrekt kezelésének követelményeiről, jogi és etikai szabályairól tájékoztattam.

A záródolgozatot/szakdolgozatot/diplomadolgozatot/portfóliót a záróvizsgán történő védésre javaslom / nem javaslom[2].

A dolgozat állam- vagy szolgálati titkot tartalmaz: igen nem*[3]

Kelt: 2022 év 11 hó 7 nap

Belső konzulens

[1] A megfelelő dolgozattípus meghagyása mellett a többi típus törlendő.
[2] A megfelelő aláhúzandó.
[3] A megfelelő aláhúzandó.

6 ÖSSZEFOGLALÁS

A dolgozatom témája egy elektronikus, 3D tervezett autó mikrokontrollerekkel történő vezérlése volt, egy kormány és két pedál segítségével. Legelső lépésként kiválasztottam a mikrokontrollereket, mind a kormány és a pedál, mind pedig az autó mellé. Miután felfedeztem, hogyan küldenek jelet saját pozíciójukról a pedálok és a kormány, átirányítottam azokat a melléjük választott Arduino UNO R3-ba.

Ezután a szükséges, vezeték nélküli kapcsolatot kellett biztosítanom a két mikrokontroller között. Ezt két NRF24L01+ rádiós kommunikációt használó chip segítségével oldottam meg.

A következő lépés a beavatkozók kiválasztása volt, ezek végül a hajtás számára egy FS90R szervomotor és egy 28BYJ-48 léptetőmotor a kormányzáshoz lettek. Mind a két motor egyszerűen és pontosan vezérelhető az autóra választott Raspberry Pi 3 Model B+-al.

Miután az alkatrészek megvoltak, megterveztem köréjük Solid Egde alkalmazásával az autó vázát.

Ezzel a hardver készen volt, elkezdhettem a működtető szoftver elkészítését. Az Arduino esetében saját programnyelvében, a Raspberry Pi-t viszont Pythonban programoztam.

Az autó ugyan vezethető, a projekt a célját elérte, de a jövő szempontjából sok fejlesztés lehetséges még. Az első, hogy bár a chip képes rá, nem küld visszajelzéseket a megérkezett adatokról. A hardveres fejlesztés egy fékrendszer beépítése lehetne, ami a fékpedál lenyomására megállítaná az autót.

7 SUMMARY

The topic of my thesis was control an electronic, 3D designed car with microcontrollers using a steering wheel and pedals. First I choosed the microconrollers for the steering wheel and pedals, and for the car. After I dicoverd how the pedals and the steering wheel send signals abuot their own position, I redirected them to the choosen Arduino UNO R3.

After that I had to provide a mandatory wireless connection between the two microcontrollers. Fort that I used two NRF24L01+ radiocommunication chips.

The selection of motors was the next step. The selected motors are one FS90R servomotor for the power to move forward and one 28BYJ-48 stepper motor for driving. Both motors are easy to control precisely with Raspberry Pi 3 Model B+.

After i had all parts, I could design around them the framework.

With that, I could start to create the operating software.I I developed the software for Arduino in it's own programming language, on the other hand I programmed Raspbery Pi in Python.

The car is driveable, the project was sucessful, although much development is available in the future. Firstly, there is no feedback on becoming data, despite the chip is capale of doing that. The upgeade of the hardware could be installing a break system, witch stops the car upon pushing down the break pedal.

8 IRODALOMJEGYZÉK

arduino.cc. (2022). *https://store.arduino.cc/products/arduino-uno-rev3*.

autodesk.com. (2022). *https://redshift.autodesk.com/articles/history-of-3d-printing*.

Bánlaki, P., & Lovas, A. (2012). *Szenzorika és anyagai.* Budapesi Műszaki és Gazdaságtudományi Egyetem Közlekedésmérnöki és Járműmérnöki kar egyetemi tananyaga.

britannica.com. (2022). *https://www.britannica.com/technology/radio-technology*.

Budó , Á. (1994). *Kísérleti Fizika II.* Nemzeti Tankönyvkiadó.

components101.com. (2022). *https://components101.com/motors/28byj-48-stepper-motor*.

farnell.com. (2022). *https://hu.farnell.com/arduino-org/k010007/starter-kit-arduino-uno-board/dp/2356913*.

fizetesek.hu. (2022). *https://www.fizetesek.hu/fizetesek/gepipar/gepeszmernok?search=1*.

Fraden, J. (2016). *Handbook of Modern Sensors.* Springer.

fuzzingbook.org. (2022). *https://www.fuzzingbook.org/beta/html/PrototypingWithPython.html*.

google.com. (2022). *https://patents.google.com/patent/US3596285A/en*.

howchoo.com. (2022). *https://howchoo.com/pi/raspberry-pi-models*.

howtomechatronics.com. (2022). *https://howtomechatronics.com/tutorials/arduino/arduino-wireless-communication-nrf24l01-tutorial/*.

Lambert, M. (2009). *Szenzorok - Elmélet és Gyakorlat.* IM Kiadó és Mérnöki Iroda Kft.

lastminuteengineers.com. (2022). *https://lastminuteengineers.com/28byj48-stepper-motor-arduino-tutorial/*.

malnapc.hu. (2022). *https://malnapc.hu/a2442-continuous-rotation-micro-servo*.

Monk, S. (2013). *Programming Arduino.* McGraw-Hill Education.

Monk, S. (2016). *Programming Arduino: Getting Started with Sketches*.

Phillips, D. (2010). *Python 3 Object Oriented Programming.* Packt Publishing.

raspberrypi.com. (2022). *https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/*.

raspbian.org. (2022). *https://www.raspbian.org/*.

Scarpino, M. (2015). *Motors for Makers: A Guide to Steppers, Servos, and Other Electrical Machines.* Que Publishing.

Scherz, P., & Monk, S. (2016). *Practical Electronics for Inventors, Fourth Edition.* McGraw-Hill Education.

Shiloh, M., & Banzi, M. (2015). *Getting Started with Arduino: The Open Source Electronics Prototyping Platform*.

sparkfun.com. (2022). *https://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss_Preliminary_Product_Specification_v1_0.pdf.*

Summerfield, M. (2008). *Programming in Python 3: A Complete Introduction to the Python Language.* Addison-Wesley Professional.

Szénássy, I. (2011). *Szenzorok és Aktuátorok.* Széchényi Isván egyetem jegyzete.

thinkful.com. (2022). *https://www.thinkful.com/blog/top-coding-languages-by-industry/*.

thomasnet.com. (2022). *https://www.thomasnet.com/articles/instruments-controls/all-about-position-sensors*.

tme.eu. (2022). *https://www.tme.eu/hu/news/library-articles/page/41861/Leptetomotorok-tipusok-es-alkalmazasok/*.

Upton, E., & Halfacree, G. (2016). *Raspberry Pi User Guide.* Wiley.

webstarsnets.com. (2022). *Forrás: https://webstarsnets.com/en/44-how-does-a-servo-motor-work.html*.

www.3dsystems.com. (2022). *https://www.3dsystems.com/blog/2016/05/sla-1-designated-historic-mechanical-engineering-landmark-asme*.

www.interviewbit.com. (2022). *https://www.interviewbit.com/blog/arduino-vs-raspberry-pi/*.

9 MELLÉKLETEK

A mellékletek című fejezet a programkódokat és az autó vázának műszaki rajzát tartalmazza.

9.1 A Get_Radio_Data.py

```
import RPi.GPIO as GPIO                     # import gpio
import time                                 #import time library
import spidev                               #import spidev library
from lib_nrf24 import NRF24                 #import NRF24 library


GPIO.setmode(GPIO.BCM)                      # set the gpio mode to BCM
GPIO.setwarnings(False)

                                            # set the pipe address. this
                                            address shoeld be entered on
                                            the other transiver also
pipes = [[0xE0, 0xE0, 0xF1, 0xF1, 0xE0], [0xF1, 0xF1, 0xF0, 0xF0, 0xE0]]
radio = NRF24(GPIO, spidev.SpiDev())        # use the gpio pins
radio.begin(0, 17)                          # start the radio and set
                                            the ce,csn pins ce=GPIO08,
                                            csn= GPIO17
radio.setPayloadSize(32)                    #set the payload size as 32
                                            bytes
radio.setChannel(0x76)                      # set the channel as 76 hex
radio.setDataRate(NRF24.BR_1MBPS)           # set radio data rate
radio.setPALevel(NRF24.PA_MIN)              # set PA level

radio.setAutoAck(True)                      # set acknowledgement as
                                            true
radio.enableDynamicPayloads()
radio.enableAckPayload()

radio.openReadingPipe(1,pipes[0])           # open the defined pipe for
                                            reading
radio.startListening()                      # start the radio


def get_radio_data():

    while not radio.available(0):           # If there is no collected
                                            signal
        time.sleep(1/100)                   # sleep for 1/100 seconds


    received_message = []                   # create a list
                                            # put the received items
                                            into the list
    radio.read(received_message,radio.getDynamicPayloadSize())


    string = ""                             # create a string

    for n in received_message:              # If the received number is
                                            a UNICODE character convert
                                            to it and add to the string
        if n >= 32 and n <= 126:
            string += chr(n)
    return string                           # Return with the string
```

9.2 A Data_Handle.py

```
WHEEL_STATEMANT_CONST_DICT = {"Left_max"  : 0,
                              "Left_mid"  : 1,
                              "Left_min"  : 2,
                              "Mid"       : 3,
                              "Right_min" : 4,
                              "Right_mid" : 5,
                              "Right_max" : 6
                              }
# Use constant dictionaries to make the code more readable
GAS_STATEMANT_CONST_DICT = {"Stand_still"    : 0,
                            "Slow_forward"   : 1,
                            "Middle_forward" : 2,
                            "Fast_forward"   : 3,
                            "Max_speed"      : 4,
                            "Slow_backward"  : 5
    }
def data_handle(data_list):

     wheel = int(data_list[0])                  # get received data into
                                                variables
     gas  = int(data_list[1])
     reverse_gear = data_list[3]
     wheel_state = WHEEL_STATEMANT_CONST_DICT["Mid"]
     gas_state = GAS_STATEMANT_CONST_DICT["Stand_still"]
      #  The received data is between 0 and 1024
      #  We split that to 7 pieces, each with 130. The regions at the start
         and end are bigger because we rarely get data
      #  from 0-100 or from 900-1024

     if wheel <= 200:
         wheel_state = WHEEL_STATEMANT_CONST_DICT["Left_max"]
     if wheel >200 and wheel<= 330:
         wheel_state = WHEEL_STATEMANT_CONST_DICT["Left_mid"]
     if wheel >330 and wheel<= 460:
         wheel_state = WHEEL_STATEMANT_CONST_DICT["Left_min"]
     if wheel >460 and wheel<= 590:
         wheel_state = WHEEL_STATEMANT_CONST_DICT["Mid"]
     if wheel >590 and wheel<= 720:
         wheel_state = WHEEL_STATEMANT_CONST_DICT["Right_min"]
     if wheel >720 and wheel<= 850:
         wheel_state = WHEEL_STATEMANT_CONST_DICT["Right_mid"]
     if wheel >850:
         wheel_state = WHEEL_STATEMANT_CONST_DICT["Right_max"]


     if gas < 200:
         gas_state = GAS_STATEMANT_CONST_DICT["Stand_still"]
     if gas > 200 and gas <= 400:
         gas_state = GAS_STATEMANT_CONST_DICT["Slow_forward"]
     if gas > 400 and gas <= 600:
         gas_state = GAS_STATEMANT_CONST_DICT["Middle_forward"]
     if gas > 600 and gas <= 800:
         gas_state = GAS_STATEMANT_CONST_DICT["Fast_forward"]
     if gas > 800 and gas <= 1000:
         gas_state = GAS_STATEMANT_CONST_DICT["Max_speed"]
     if reverse_gear == 1:
         gas_state = GAS_STATEMANT_CONST_DICT["Slow_backward"]


     handled_list = [wheel_state,gas_state]
     return handled_list
```

9.3 Stepper_Motor_Class.py

```
import time
import RPi.GPIO as GPIO
from Data_Handle import WHEEL_STATEMANT_CONST_DICT

GPIO.setwarnings(False)


class stepper_motor:

    def __init__(self):
        self.in1 = 13
        self.in2 = 19
        self.in3 = 26
        self.in4 = 21
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(self.in1, GPIO.OUT)
        GPIO.setup(self.in2, GPIO.OUT)
        GPIO.setup(self.in3, GPIO.OUT)
        GPIO.setup(self.in4, GPIO.OUT)
        self.step_sleep = 1/5000
        self.steps = [[1,0,0,0],
                      [1,1,0,0],
                      [0,1,0,0],
                      [0,1,1,0],
                      [0,0,1,0],
                      [0,0,1,1],
                      [0,0,0,1],
                      [1,0,0,1]]
        self.pins = [self.in1,self.in2,self.in3,self.in4]
        self.position = WHEEL_STATEMANT_CONST_DICT["Mid"]


    def get_position(self):
            return self.position

    def turn_right(self):
            for j in range(16):
                for i in range(len(self.steps)):
                    for pin in range(0, len(self.pins)):
                        GPIO.output(self.pins[pin],self.steps[i][pin])
                        time.sleep(self.step_sleep)
            self.position += 1

    def turn_left(self):
            for j in range(16):
                for i in reversed(range(len(self.steps))):
                    for pin in range(0, len(self.pins)):
                        GPIO.output(self.pins[pin],self.steps[i][pin])
                        time.sleep(self.step_sleep)
            self.position -= 1
```

9.4 A Servo_Motor_Class.py

```
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

GPIO.setup(18, GPIO.OUT)


class servo_motor:

    def __init__(self):
        self.motor = GPIO.PWM(18,50)
        self.motor.start(0)


    def Stand_still(self):
        self.motor.ChangeDutyCycle(0)
    def Slow_forward(self):
        self.motor.ChangeDutyCycle(7.5)
    def Middle_forward(self):
        self.motor.ChangeDutyCycle(8)
    def Fast_forward(self):
        self.motor.ChangeDutyCycle(13)
    def Max_speed(self):
        self.motor.ChangeDutyCycle(17)
    def Slow_backward(self):
        self.motor.ChangeDutyCycle(6)
```

9.5 main.py

```
import RPi.GPIO as GPIO                     # import gpio
import time                                 #import time library
import spidev
from lib_nrf24 import NRF24
from Get_Radio_Data import get_radio_data
from Data_Handle import data_handle, GAS_STATEMANT_CONST_DICT
from servo_motor_class import servo_motor
from stepper_motor_class import stepper_motor



GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

s_motor = servo_motor()
st_motor = stepper_motor()

while True:
    data = get_radio_data()
    data_list = data.split(",")
    handled_list = data_handle(data_list)
    print(data)
    print(data_list)
    print(handled_list)

    car_wheel = handled_list[0]
    wheel_pos = st_motor.get_position()

    if car_wheel < wheel_pos:
        st_motor.turn_left()
    if car_wheel > wheel_pos:
        st_motor.turn_right()

    car_gas = handled_list[1]

    if car_gas == GAS_STATEMANT_CONST_DICT["Stand_still"]:
        s_motor.Stand_still()
    if car_gas == GAS_STATEMANT_CONST_DICT["Slow_forward"]:
        s_motor.Slow_forward()
    if car_gas == GAS_STATEMANT_CONST_DICT["Middle_forward"]:
        s_motor.Middle_forward()
    if car_gas == GAS_STATEMANT_CONST_DICT["Fast_forward"]:
        s_motor.Fast_forward()
    if car_gas == GAS_STATEMANT_CONST_DICT["Max_speed"]:
        s_motor.Max_speed()
    if car_gas == GAS_STATEMANT_CONST_DICT["Slow_backward"]:
        s_motor.Slow_backward()
```

9.6 Arduinocode.ino

```
#include <SPI.h>
#include <RF24.h>


RF24 radio(9, 10);

void setup() {
  Serial.begin(9600);
  radio.begin();
  radio.setPALevel(RF24_PA_MAX);
  radio.setChannel(0x76);
  //radio.openWritingPipe(0xF0F0F0F0E1);
  radio.openWritingPipe(0xE0E0F1F1E0);
  radio.enableDynamicPayloads();
  radio.powerUp();

  int InPin = 2;
  pinMode(InPin, INPUT);
}
void loop() {
  int InPin = 2;
  int seged = 0;
  long kormany_allas = 0;
  int kormany = 0;
  long gaz_allas = 0;
  int gaz = 0;
  int fek = 0;
  int fek_allas = 0;

  int tolatas = 0;
  bool irany;
  while (seged < 100){
    int kormany = analogRead(A0);
```

```
  fek = analogRead(A1);
  gaz = analogRead(A2);
  kormany_allas = kormany_allas + kormany;
  gaz_allas = gaz_allas + gaz;
  seged = seged + 1;
  delay (5);
  }
if (fek<500){
  fek_allas = 1;
  }
kormany_allas = kormany_allas / 100;
gaz_allas     = gaz_allas / 100;
irany = digitalRead(InPin);

if (irany == true)
  {
    tolatas = 1;
  }
else
  {
    tolatas = 0;
  }
String to_send = "";
String kormany_str = String(kormany_allas);
to_send += kormany_str;
to_send +=",";
String gaz_str = String(gaz_allas);
to_send += gaz_str;
to_send +=",";
String fek_str = String(fek_allas);
to_send += fek_str;
to_send +=",";
String tolatas_string = String(tolatas);
to_send += tolatas_string;
```

```
char char_to_send[16];
to_send.toCharArray(char_to_send,16);
//char kormany_char[5];
//itoa(kormany_char,kormany_allas,10);
//char gaz_char[5];
//itoa(gaz_char,gaz_allas,10);
//char fek_char;
//itoa(fek_char,fek_allas,10);

//char char_to_send[16] = strcat(kormany_char,gaz_char);
Serial.print("kormány   ");
Serial.print(kormany_allas);
Serial.print(",  gáz   ");
Serial.print(gaz);
Serial.print(",   fék  ");
Serial.print(fek);
Serial.print(",   tolat  ");
Serial.println(tolatas);
radio.write(char_to_send, sizeof(char_to_send));
```