



INFORMATIKAI KAR – IK

Arduino robotfejlesztés android alkalmazás használatával

Készítette

Bozó Tamás Dániel

Programtervező informatikus BSc

Témavezető

Balla Tamás

Tanársegéd

EGER, 2021

Tartalomjegyzék

1. Arduino és Android bemutatása	5
1.1. Az Arduino	5
1.1.1. A fejlesztéshez választott alaplap	5
1.1.2. Az IDE felépítése és használata	6
1.1.3. A C++ nyelv	7
1.2. A Java nyelv és az Android	8
1.2.1. Miért pont Android?	8
1.2.2. Az Android Studio felépítése és használata	9
2. Követelményspecifikáció	11
2.1. Vágylomrendszer leírása	11
2.2. Követelmények	12
2.3. Jelenlegi helyzet leírása	12
3. Funkcionális specifikáció	13
3.1. Használati esetek	13
3.2. Követelmény megfeleltetés	13
3.2.1. Könnyű módosíthatóság	13
3.2.2. Érdekes projekt	14
3.2.3. Sokoldalú felhasználás	14
3.3. A fejlesztéshez használt hardver	14
4. Belső állapotok és felületterv	15
4.1. A robot belső állapotai	15
4.1.1. Kezdőállapot	15
4.1.2. Az irányítás folyamata	16
4.1.3. Automata követés folyamata	16
4.1.4. A naplázási folyamat	18
4.2. Android képernyőtervezet	18
5. A robot és az applikáció működése	23
5.1. A telefon és a robot összekapcsolása	23

5.2.	A fő folyamatok eldöntése	25
5.3.	Kód küldése a telefonról	25
5.4.	Motorok kezelése	27
5.5.	Akadályok érzékelése	28
5.6.	A pályát jelölő vonal követése	29
5.7.	Akadály kikerülése	30
5.8.	A naplófájl feldolgozása és mentése	31
6.	Fejlesztés közbeni tapasztalatok	34
6.1.	Hardverrel kapcsolatos tapasztalatok	34
6.1.1.	A robot építésekor szerzett tapasztalatok	34
6.1.2.	A telefon hardverére vonatkozó tapasztalatok	36
6.2.	Szoftveres észrevételek	37
6.2.1.	A késleltetéssel kapcsolatos probléma megoldása	38
6.2.2.	Naplófájl fogadásának megvalósítása	38
7.	Felhasználói útmutató	39
7.1.	A robot megépítésének lépései	39
7.1.1.	Szükséges alkatrészek	39
7.1.2.	A motorok és az akkumulátor tartó felhelyezése	40
7.1.3.	Szenzorok felhelyezése	40
7.1.4.	A program telepítése az alaplapra	41
7.2.	A telefonon elvégzendő lépések	42
7.2.1.	Az alkalmazás telepítése	42
7.2.2.	Bluetooth párosítás	42
7.2.3.	Az alkalmazás első indítása	43
8.	Összegzés és továbbfejlesztés	44
8.1.	Összegzés	44
8.2.	Hardverre vonatkozó fejlesztés	44
8.3.	Android szoftverre vonatkozó fejlesztés	45
8.4.	A hardver biztosítására vonatkozó fejlesztés	45

Bevezetés

,There are an endless number of things to discover about robotics. A lot of it is just too fantastic for people to believe.” – Daniel H. Wilson

A mai világban egyre több helyen alkalmaznak robotokat. Akár a munkahelyen a termelés felgyorsítására, az egészségügyben az orvosok segítsére, vagy csak az otthonunkban különböző feladatok elvégzésére. Azért választottam ezt a témát diplomamunkám elkészítéséhez, ugyanis az egyetemi tanulmányaim befejezése utáni általános iskolai tanárként elhelyezkedni lakóhelyemen és a diákok ismereteit az alaptanterven túl bővíteni a robotika világában.

Szakdolgozatom témaja egy olyan robot építése és programozása, mely egy vonallal kirajzolt pályát képes követni, a pályán található esetleges akadályokat érzékelni és kikerülni, valamint egy olyan android rendszerrel rendelkező készülékre telepíthető applikáció fejlesztése melynek segítségével a felhasználó is képes irányítani. A robot megtervezése során igyekeztem egy olyan felépítésre törekedni, amit otthon hobbi felhasználók, de akár általános iskolások is képesek megépíteni, valamint egy olyan programkód írására, amit könnyen meg lehet érteni és esetleg változtatni/bővíteni.

A telefonos applikáció megvalósítására egy olyan kezelőfelületet akartam létrehozni, mely egyaránt biztosítja a lehetőséget a valós idejű irányításra, valamint a robot működésének monitorozására is.

Programozási nyelvként a C++-t és a Java-t választottam. A Java egy széles körben elterjedt nyelv, melyet könnyen lehet értelmezni, a C++ pedig az Arduino által támogatott nyelv, fejlesztői környezetnek pedig az Arduino IDE és Android Studio nevű alkalmazásokat használtam.

A dokumentumban olvashatunk a diplomamunkám elkészítéséhez választott technológiákról és programozási nyelvekről az 1. fejezetben, a téma választása után kitűzött célokáról és követelményekről a 2. fejezetben, a fejlesztés és tesztelés során használt hardverekről a 3. fejezetben, az Android applikáció felületterveiről és a robot belső állapotairól, valamint ezek működéséről a 4. és 5. fejezetekben, a fejlesztés során szerzett személyes tapasztalataimról a 6. fejezetben. A 7. és 8. fejezetekben pedig a projekt összeállításához tartozó útmutatóról és a továbbfejlesztési lehetőségekről.

1. fejezet

Arduino és Android bemutatása

„Programming today is a race between software engineers striving to build bigger and better idiot-proof programs, and the Universe trying to produce bigger and better idiots. So far, the Universe is winning” – Rick Cook, The Wizardry Compiled

1.1. Az Arduino

Az Arduino[1] egy szabad szoftveres, nyílt forráskódú elektronikai fejlesztőplatform. Széles tömegek számára elérhető, mivel olcsó, könnyen beszerezhető, egyszerűen programozható és csatlakoztatható más eszközökhöz. A fejlesztői platform az úgynevezett "IDE"-ből és egy Arduino Board-ból áll. Az előbbi segítségével programokat írhatunk és tesztelhetünk a számítógépen, utóbbi pedig egy hardver eszköz, amelyre az elkészített programokat feltölthetjük a számítógépen keresztül, majd az eszközöket vezérelhetjük a segítségével.

Az Arduino lap kereskedelmi forgalomban kapható, előre összeszerelt, vagy otthon összeszerelhető alkatrészcsomagként. Mivel nyílt forráskódú a hardver is, bárki készíthet magának saját változatot belőle, vagy az eredetivel kompatibilis klónt.

1.1.1. A fejlesztéshez választott alaplap

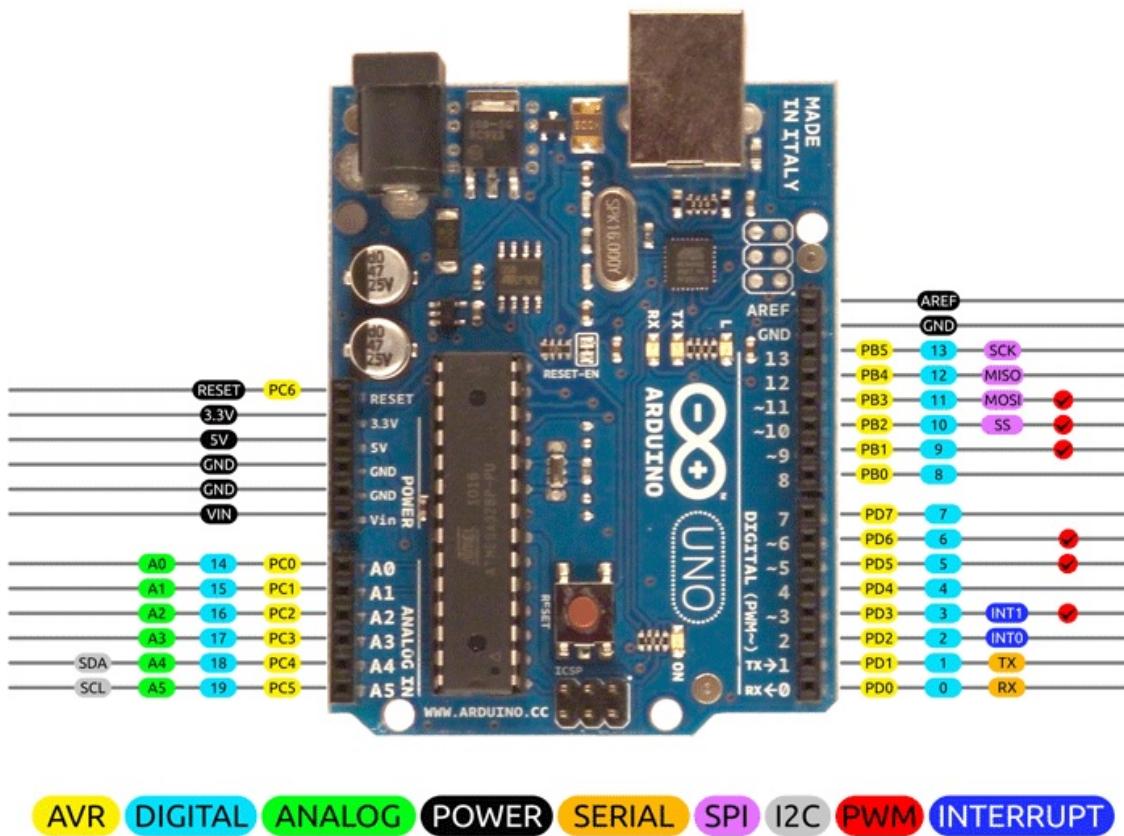
A fejlesztés megkezdése előtt szükséges volt kiválasztanom azon hardverelemeket, melyek a robot mozgását, és vezérlését biztosítják, ezek:

- 4 db villanymotor
- 2 db HW-201 típusú infravörös szenzor
- 1 db HC-SR04 típusú ultrahang szenzor
- 1 db HC-05 típusú Bluetooth modul
- 1 db SG90 típusú szervo motor

- 1 db Adafruit Motorvezérlő Shield

Miután a hardverelemeket kiválasztottam, szükségem volt egy alaplapra is, mely a robot központi része lesz. A választásom az Arduino UNO-ra esett, mivel ez rendelkezik megfelelő számú be- és kimenettel, valamint a motorok vezérléséhez használt shield-el is kompatibilis.

Az UNO egy ATmega328P mikrokontrollerrel van felszerelve, valamint 32 KB Flash memóriával rendelkezik, melyben a feltöltött program tárolódik. Csatlakozók szempontjából 6 db analóg és 14 db digitális tüske, valamint egy USB port található rajta (a kiosztást az 1.1 ábrán láthatjuk).



1.1. ábra. Az UNO tüskekiosztása

1.1.2. Az IDE felépítése és használata

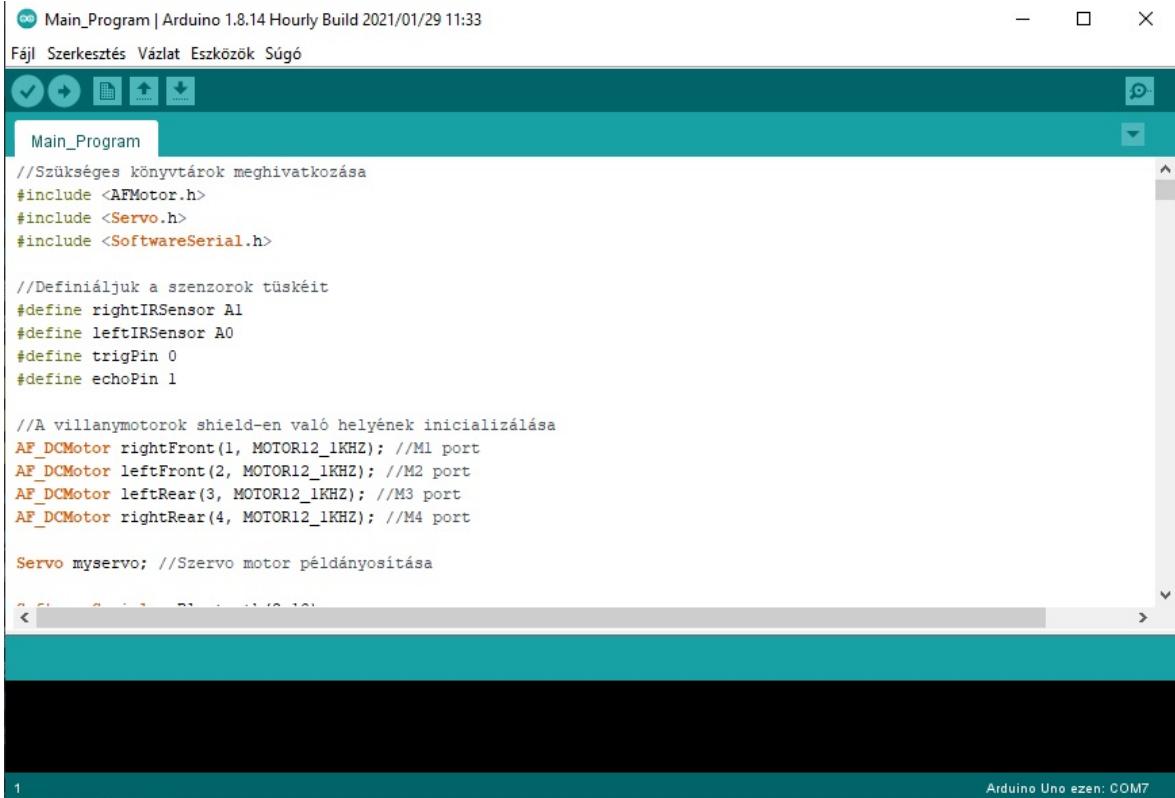
A programok írását és alaplapra való feltöltését az Arduino IDE segítségével tehetjük meg (1.2 ábra), mely egy felhasználóbarát fejlesztőkörnyezet.

Használatakor láthatjuk a használt USB port sorszámát és az ehhez csatlakoztatott alaplat típusát, továbbá a programunk nevét is.

A szerkesztőfelületen írhatjuk a programot és az alatta található üzenetfelületen kap a felhasználó visszajelzést arról, hogy a program feltöltése sikeres volt-e, illetve az

előforduló problémákról is.

A program feltöltésén kívül lehetőségünk van az ellenőrzésre is, továbbá az eszköz-tárban található menüpontok segítségével megnyithatjuk a soros monitort, létrehozhatunk új projektet, de akár a használt alaplap és USB port is megváltoztatható, így egyszerre akár több alaplapot is csatlakoztathatunk a számítógéphez.



The screenshot shows the Arduino IDE interface. The title bar reads "Main_Program | Arduino 1.8.14 Hourly Build 2021/01/29 11:33". The menu bar includes "Fájl", "Szerkesztés", "Vázlat", "Eszközök", and "Súgó". Below the menu is a toolbar with icons for file operations like Open, Save, and Print. The main workspace displays the following C++ code:

```
//Szükséges könyvtárok meghivatkozása
#include <AFMotor.h>
#include <Servo.h>
#include <SoftwareSerial.h>

//Definiáljuk a szenzorok tüskéit
#define rightIRSensor A1
#define leftIRSensor A0
#define trigPin 0
#define echoPin 1

//A villanymotorok shield-en való helyének inicializálása
AF_DCMotor rightFront(1, MOTOR12_1KHZ); //M1 port
AF_DCMotor leftFront(2, MOTOR12_1KHZ); //M2 port
AF_DCMotor leftRear(3, MOTOR12_1KHZ); //M3 port
AF_DCMotor rightRear(4, MOTOR12_1KHZ); //M4 port

Servo myservo; //Szervo motor példányosítása
```

The status bar at the bottom indicates "Arduino Uno ezen: COM7".

1.2. ábra. Az IDE felülete

Az IDE teljes felépítése megtekinthető az Arduino hivatalos honlapján[5].

1.1.3. A C++ nyelv

A C++[2] egy általános célú, magas szintű programozási nyelv. Támogatja a procedurális, az objektumorientált és a generikus programozást, valamint az adatabsztraktciót. Napjainkban szinte minden operációs rendszer alá létezik C++ fordító.

A nyelv a C programozási nyelv hatékonyságának megőrzése mellett törekszik a könnyebben megírható, karbantartható és újrahasznosítható kód írására, ez azonban sok kompromisszummal jár. Erre utal, hogy általánosan elterjedt a mid-level minősítése is, bár szigorú értelemben véve egyértelműen magas szintű.

Bjarne Stroustrup kezdte el a C++ programozási nyelv fejlesztését a C programozási nyelv kiterjesztéseként, más nyelvekből véve át megoldásokat (Simula67, Algol68), ötleteket (ADA).

A nyelv első, nem kísérleti körülmények között való használatára 1983-ban került sor, 1987-ben pedig nyilvánvalóvá vált, hogy a C++ szabványosítása elkerülhetetlen. Ez a folyamat 1991 júniusában kezdődött el, amikor az ISO szabványosítási kezdeményezés részévé vált.

A C++ programozási nyelv szabványát 1998-ban hagyták jóvá ISO/IEC 14882:1998 néven, az aktuális, 2017-es változat kódjelzése ISO/IEC 14882:2017.

1.2. A Java nyelv és az Android

A Java[3] általános célú, objektumorientált programozási nyelv, amelyet a Sun Microsystems fejlesztett a '90-es évek elejétől kezdve egészen 2009-ig, amikor a céget felvásárolta az Oracle. 2011-ben a Java 1.7-es verzióját az új tulajdonos gondozásában adták ki.

A Java alkalmazásokat jellemzően bájtkód formátumra alakítják, de közvetlenül natív (gépi) kód is készíthető Java forráskódóból. A bájtkód futtatása a Java virtuális géppel történik, ami vagy interpretálja a bájtkódot, vagy natív gépi kódot készít belőle, és azt futtatja az adott operációs rendszeren. Létezik közvetlenül Java bájtkódot futtató hardver is, az úgynevezett Java processzor.

A Java nyelv a szintaxisát főleg a C és a C++ nyelvectől örökölte, viszont sokkal egyszerűbb objektummodellel rendelkezik, mint a C++.

1.2.1. Miért pont Android?

A modern világban szinte minden ember rendelkezik, vagy találkozott már olyan telefonnal, mely valamilyen mobil OS-t¹ futtat, melyek közül a legjobban elterjedt az Android[4] és az iOS.

Az Android-ot kezdetben az Android Inc. fejlesztette, azonban 2005-ben a Google felvásárolta és a fejlesztést az Open Handset Alliance folytatta, melyet 82 cég alkot, köztük az Intel, a HTC, a Samsung és az eBay is.

Az Android-ot futtató telefonok a felhasználók számára nagy rendszer-testreszabási lehetőségeket nyújtanak, valamint széleskörű kompatibilitást különféle kiegészítőkkel, például vezeték nélküli eszközök, okos óra/tv.

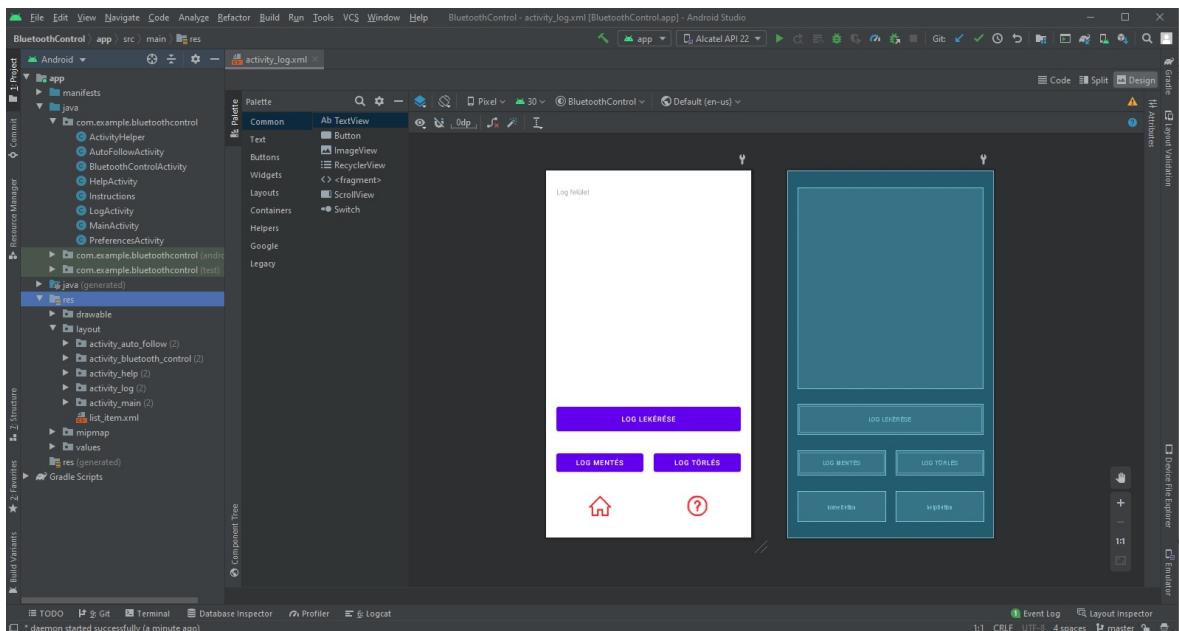
Android-ra a fejlesztők Java vagy Kotlin programozási nyelv segítségével tudnak programokat írni, fejlesztőkörnyezetnek pedig legelterjedtebb az Android Studio, melyről a következő oldalon részletesen is olvashatunk.

¹ mobil operációs rendszer

1.2.2. Az Android Studio felépítése és használata

Az Android Studio[4] egy integrált fejlesztőkörnyezet (IDE) az Android platformra való fejlesztéshez. 2013. május 16-án jelentette be a Google I/O konferencián a Google termékmenedzsere Katherine Chou. Az Android Studio szabadon elérhető Apache License 2.0 alatt.

A JetBrains IntelliJ IDEA-ján alapuló szoftvert, az Android Studiot kifejezetten androidos fejlesztésre terveztek. Letölthető Windowsra, Mac OS X-re és Linuxra is egyaránt, és teljes egészében helyettesíti az Eclipse Android Development Tools (ADT)-ját. A Google elsődleges IDE-jének választott a natív Android alkalmazás fejlesztésre.



1.3. ábra. Az Android Studio

Az IDE használata hamar megtanulható, ugyanis felhasználóbarát felépítéssel rendelkezik(1.3 ábra). Az alkalmazás dizájnjának megalkotására egy beépített felületet biztosít az Android Studio, ami WYSIWYG² alapon működik.

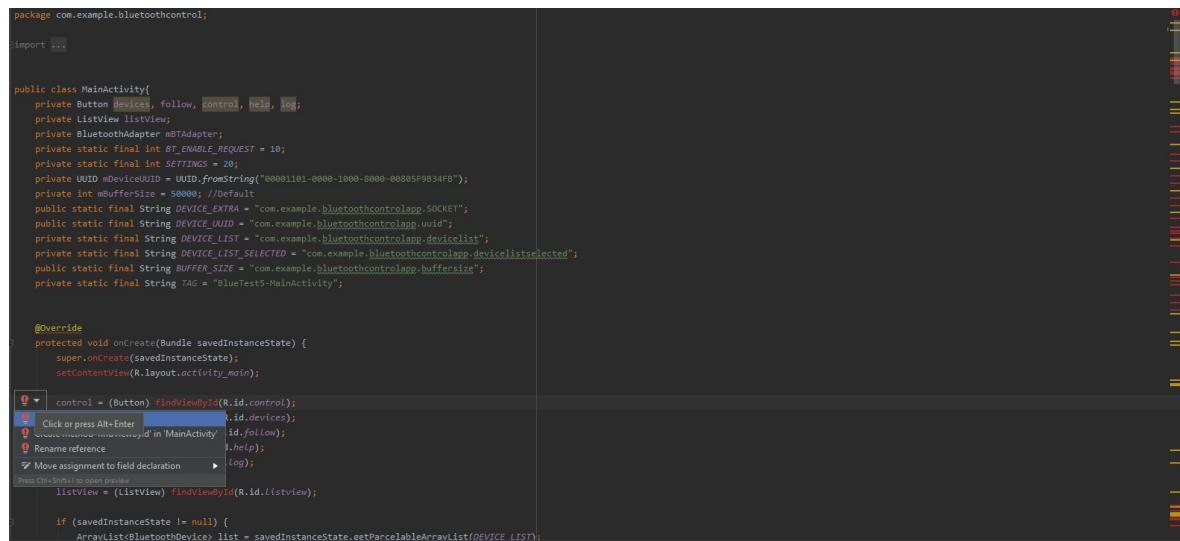
Az IDE tartalmaz továbbá egy beépített navigációs felületet, ahol a felhasználó a megnyitott projekt elemeit találja meg. A projektben megtalálható fájlok mappaszerkezetét látja a felhasználó, így a dizájnért felelős .xml kiterjesztésű fájlok a 'res/layout' mappában találhatóak és az ugyanazon képernyőhöz tartozó fekvő és álló elrendezés pedig a képernyő mappájában.

A navigációs sávon kívül a felhasználó több, a projekt állapotához tartozó információt is kap, mint például a projekt futtatásakor az ablak alján megjelenő Event Log, ami a programmal kapcsolatban ad visszajelzést (pl: sikeres volt-e a build), vagy az

² What you see is what you get/Azt kapod, amit látsz

eszköztár alatt látható a készülék, vagy AVD³ neve, amelyen a program futtatása fog történni.

További hasznos tulajdonság, hogy amennyiben a fejlesztő hibát követ el a forráskód írása közben az IDE figyelmeztet erről, valamint az ebből eredő összes hibát megjelöli és a hiba megoldására is tanácsot ad a felhasználónak (1.4 ábra), valamint a projektben történő refaktorálást is könnyen megtehetjük a Shift+F6 billentyűkombinációval, aminek hatására az IDE nem csak az adott sorban, hanem minden használati esetnél is módosítja az általunk kijelölt objektumot.



A screenshot of the Android Studio code editor showing a warning tooltip. The code is for a Main Activity class. A tooltip is displayed over the line 'control = (Button) findViewById(R.id.control);'. The tooltip contains the following options:

- Click or press Alt+Enter
- Create instance variable 'id' in 'MainActivity'
- Rename reference
- Move assignment to field declaration
- Press Ctrl+Shift+O to open project

The code snippet is as follows:

```
package com.example.bluetoothcontrol;

import ...

public class MainActivity {
    private Button devices, follow, control, help, log;
    private ListView listView;
    private BluetoothAdapter mBTAdapter;
    private static final int BT_ENABLE_REQUEST = 10;
    private static final int SETTINGS = 20;
    private UUID mDeviceUUID = UUID.fromString("00001101-0000-1000-8000-00805F9834FB");
    private int mBufferSize = 50000; //Default
    public static final String DEVICE_EXTRA = "com.example.bluetoothcontrolapp.SOCKET";
    public static final String DEVICE_UUID = "com.example.bluetoothcontrolapp.uuid";
    private static final String DEVICE_LIST = "com.example.bluetoothcontrolapp.deviceList";
    private static final String DEVICE_SELECTED = "com.example.bluetoothcontrolapp.deviceSelected";
    public static final String BUFFER_SIZE = "com.example.bluetoothcontrolapp.bufferSize";
    private static final String TAG = "BlueTest5MainActivity";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        control = (Button) findViewById(R.id.control);
        Click or press Alt+Enter
        Create instance variable 'id' in 'MainActivity'
        Rename reference
        Move assignment to field declaration
        Press Ctrl+Shift+O to open project
        listView = (ListView) findViewById(R.id.listView);
    }

    if (savedInstanceState != null) {
        ArrayList<BluetoothDevice> list = savedInstanceState.getParcelableArrayList(DEVICE_LIST);
    }
}
```

1.4. ábra. Hiba jelzése az IDE-ben

³ Android Virtual Device

2. fejezet

Követelményspecifikáció

Ebben a fejezetben arról fogok beszámolni, hogy a projekt kigondolásakor, milyen célt tűztem ki magamnak, valamint ennek megvalósítására milyen komponenseket használtam.

2.1. Vágylomrendszer leírása

A projekt megalkotásakor a célom egy olyan robot kifejlesztése volt, amely képes egy megadott pályát követni, a követés közben észlelt bármekkora méretű akadályt kikerül és ezután a követendő pályát újra megtalálni, valamint egy olyan telefonos alkalmazást kifejleszteni, mellyel a robotot irányítani tudjuk, a robot által vezetett napló tartalmának megtekintését lehetővé teszi, valamint a naplót a készülékre lehessen menteni.

Szerettem volna ezeket úgy megvalósítani, hogy a végtermékkel különböző életkorú diákok oktatását lehessen segíteni, de otthoni felhasználáshoz is szórakoztató időtöltés legyen akár egyszerre több embernek is.

A 2020-as Nemzeti Alaptantervben[6] az általános iskolás diákok számára a Digitális Kultúra tárgy magába foglalja a robotika oktatását 5. és 6. osztályban 11 óra alkalmával, valamint 7-8. osztályosoknak 8 órában. Az órák alatt a diákok számára a javasolt tevékenység különböző programok megírása blokkprogramozással, ezért nekik lehetőséget akarok biztosítani egy másik programozási módszer megismerésére.

Középiskolások részére a tárgyban megtalálható a Mobiltechnológiai ismeretek téma, melyet 9. és 11. osztályban 4-4 órában tanulnak a diákok. Az Ő részükre a telefonra fejlesztett applikációval szeretnék egy példát mutatni a mobiltechnológiai eszközök segítségével megvalósított együttműködésre.

Hobbifelhasználóknak pedig egy olyan kikapcsolódási lehetőséget akartam biztosítani, mellyel maguk építhetnek meg egy olyan robotot, amit utána irányítani is tudnak.

2.2. Követelmények

Miután kitűztem a célt, amit el szeretnék érni meg kellett fogalmaznom néhány elvárást is.

A kód könnyen módosítható legyen. Ahhoz, hogy az általános iskolai diákok számára ne legyen bonyolult a projekt, szükséges, hogy az alapoktól kezdve tudjuk megmutatni nekik hogyan is épül fel.

A projekt legyen érdekes. A diákok számára egy olyan példát kell biztosítani, mely felkelti az érdeklődésüket.

Sokoldalú felhasználás. Mivel nem csak oktatási célra szánom a projektemet, szükségesnek tartottam, hogy a hétköznapi felhasználók is szórakoztatónak találják.

2.3. Jelenlegi helyzet leírása

A projekt jelenlegi állapotában a 2.1. fejezetben megfogalmazottaknak sikerült eleget tennem. A robot az alváz elején található infravörös szenzorok segítségével képes követni egy előre megadott pályát, ezen a kanyarokat be tudja venni (az éles, 45-90°-os kanyarok néha problémát okozhatnak), majd ugyanezen szenzoroknak köszönhetően egy akadály kikerülése után a követendő pályát képes ismételten megtalálni.

A felhasznált ultrahang szenzor biztosítja azt, hogy a robot nem fog akadálynak ütközni a pálya követése közben, a Bluetooth modulnak köszönhetően pedig a telefonos applikációval rá tudunk csatlakozni a robotra.

A telefonos applikációban a robot vezérlésének több részét is el tudjuk végezni. Először is el tudjuk dönten, hogy a robot kövesse-e a pályát vagy mi magunk szeretnékn irányítani. Másodszor a robot irányításakor mi tudjuk megmondani az irányt, amerre szeretnénk, ha a robot mozogna. Harmadszor pedig a robot által létrehozott naplófájlt meg tudjuk a telefonon tekinteni, a telefonra le tudjuk menteni, vagy a robot memóriájából ki tudjuk törölni.

3. fejezet

Funkcionális specifikáció

A fejezetben a projekt használati lehetőségeit mutatom, valamint a 2. fejezetben megfogalmazottaknak feleltetem meg a termékemet, majd ismertetem a fejlesztéshez és teszteléshez használt hardverelemek néhány tulajdonságát.

3.1. Használati esetek

A projektnek egy felhasználója van, egy személy. A felhasználó két különböző szoftverrel érintkezik és ezek más-más funkciókkal rendelkeznek.

A roboton futó szoftver képes a naplóbólhoz új eseményeket hozzáadni, eseményeket törölni, a naplóból a telefonra elküldeni, valamint képes a pálya követésére.

Az okostelefonon futó szoftver el tudja végezni a robot irányítását, a vonal követését szüneteltetheti/indíthatja el és a robot által készített naplóból tartalmát lekérheti és mentheti.

3.2. Követelmény megfeleltetés

A következőkben részletezem, hogy a 2.2 pontban megfogalmazottaknak hogyan tettem eleget.

3.2.1. Könnyű módosíthatóság

Az Arduino alaplapon futó kódban minden olyan egységet, mely egy szenzorról beérkező értéket használ fel egy feltétel vizsgálatához, vagy változó értékének módosításához igyekeztem külön metódusba építeni, így a kód gyorsan és biztonságosan módosítható.

A módosíthatóságnak köszönhetően a diákok számára létre lehet hozni egy olyan struktúrát a projektben, melyben kezdetben csak az infravörös szenzorokat használjuk LED-el, majd lépésről-lépéstre a LED-et lecseréljük a motorokra, valamint hozzáadjuk az ultrahang szenzort és végül ezeknek a működését hangoljuk össze.

3.2.2. Érdekes projekt

A projekt véleményem szerint a diákok számára kellő érdekességgel bír, ugyanis nem csak új ismeretet tudnak vele szerezni, hanem egy interaktív óra lehetőségét is magában hordozza.

3.2.3. Sokoldalú felhasználás

A projekt nem csak az irányítás és a pályakövetés lehetőségét biztosítja. Ha szeretnénk a robotnak egyre összetettebb és nehezebb pályákat tudunk építeni, emelkedőkkel, nagyobb akadályokkal összetettebb kanyarokkal, ezzel tesztelve, hogy mi az amire a robot képes és mire nem. Más lehetőség, hogy az otthoni felhasználás során a roboton található modulok, vagy a telefonos alkalmazás átalakításával a projekt továbbfejlesztésére is van lehetőség.

3.3. A fejlesztéshez használt hardver

Az 1.1.1 fejezetben olvashattunk az alaplap adatairól, most bemutatom a további hardverelemeket, illetve a teszteléshez felhasznált telefonokat.

A robothoz tartozó hardverelemek

Megnevezés	Csatlakozás	Működési feszültség
Villanymotor	2 db digitális port/motor	3-6 Volt
SG90 szervo motor	1 db analóg port	4.8-6 Volt
HC-SR04 szenzor	2 db digitális port	5 Volt
HW-201 szenzor	1 db analóg port	5 Volt
HC-05 Bluetooth modul	2 db digitális port	3.3-5 Volt

A telefonok adatai

	Samsung Galaxy A71	Alcatel Pop D1
Processzor	Qualcomm Snapdragon 730	ARM Cortex-A7
Processzormagok száma	8	2
Processzor sebessége	2.2 GHz	1 GHz
RAM mérete	6 GB	512 MB
Akkumulátor típusa	Lithium-Polymer	Lithium-Ion
Akkumulátor kapacitás	4500 mAh	1300 mAh
Android verzió	10	4.4.2 KitKat
Bluetooth verzió	v5.00	v4.00

4. fejezet

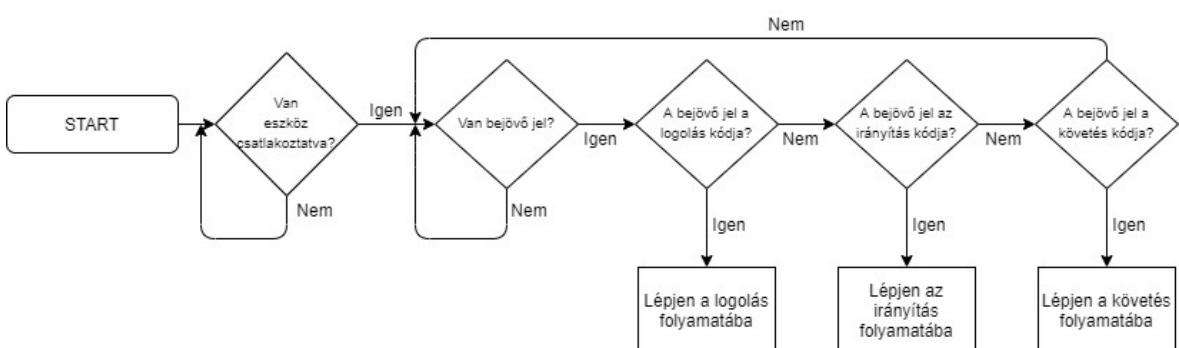
Belső állapotok és felületterv

„Any fool can write code that a computer can understand. Good programmers write code that humans can understand.” – Martin Fowler

4.1. A robot belső állapotai

A robot működése során többször is változik a belső állapota, a változást a telefonos applikációból kapott adat, valamint a roboton található szenzorok idézik elő.

4.1.1. Kezdőállapot



4.1. ábra. Fő állapotok közötti átmenet

A robot indulását követően a kezdőállapotból minden alkalommal a telefonról kapott jel fogja kimozdítani. Amint van csatlakoztatott eszköz a robot elkezdi várni az utasítást. A telefonról érkező utasítás 3 számkód lehet. minden számkód egy-egy folyamatért felel, mely lehet:

- Irányítás folyamata = 111-es kód
- Pályakövetés folyamata = 222-es kód
- Naplázás folyamata = 333-as kód

4.1.2. Az irányítás folyamata

Abban az esetben, ha az applikáció főmenüjében az 'irányítás' lehetőséget választjuk a robot elkezd várni a bejövő utasításra. Amennyiben ez az utasítás megegyezik a folyamatot megszakító kóddal a robot visszatér a kezdőállapotába.

Ha a kapott utasításból származó kód az 'F', 'L', 'R', 'B', 'S' betűk valamelyike a robot az ehhez tartozó mozgást végzi el.

– F = Előrehaladás

– L = Balkanyar

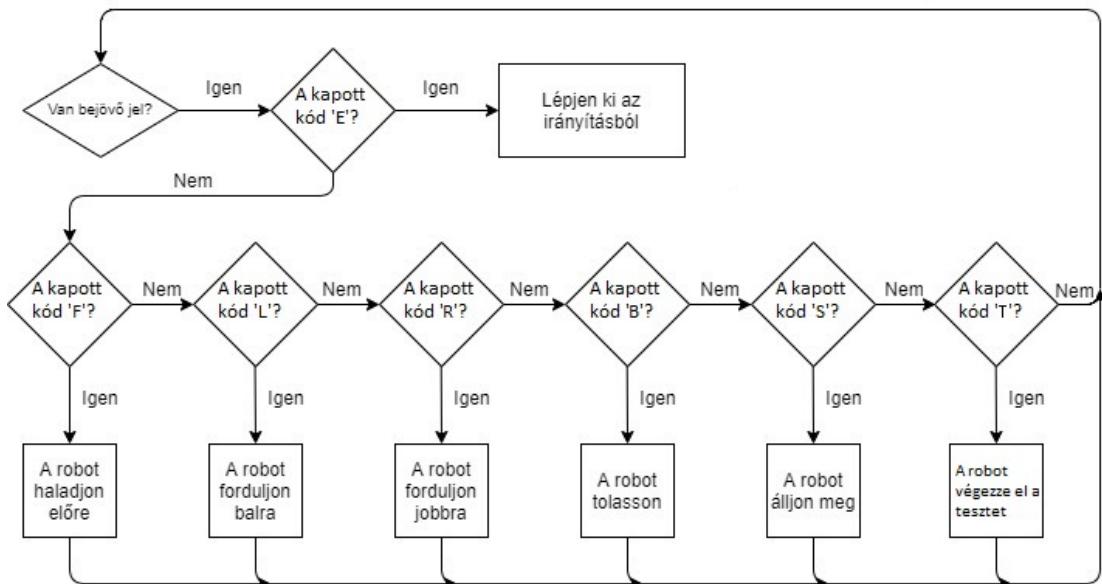
– R = Jobb kanyar

– B = Tolatás

– S = Állj

– T = Teszt folyamat

A telefon a kódot mindaddig küldi ki a robotnak, amíg az adott gomb lenyomva van. Amint a gombot felengedjük a telefon elküldi a robot megállítására szolgáló 'S' kódot.



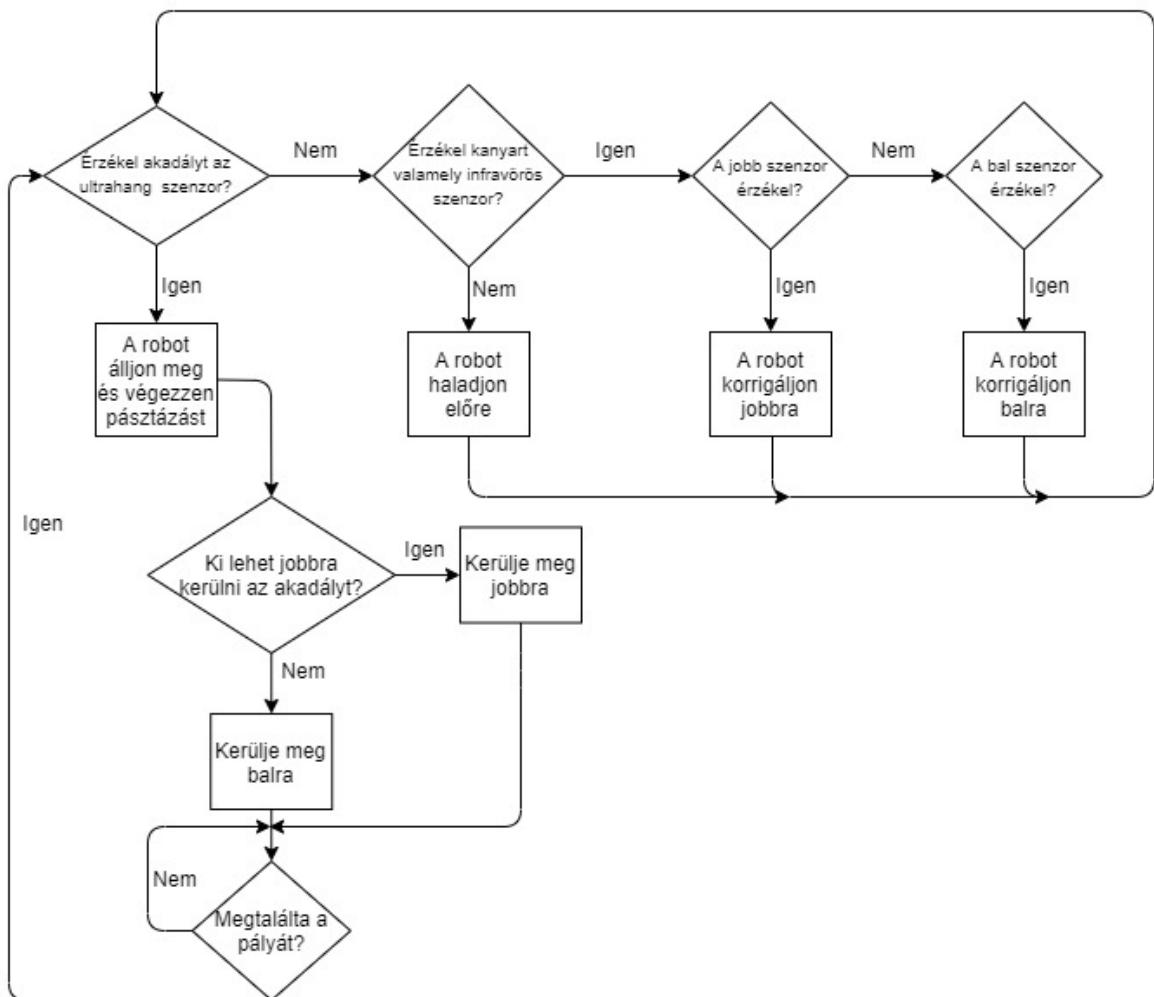
4.2. ábra. Bluetooth irányítás

4.1.3. Automata követés folyamata

Amennyiben a kezdőképernyőn a 'követés' gombot nyomja meg a felhasználó a roboton található szenzorok végzik el a belső állapotok változtatását. Ekkor a robot a váz elején

található Infravörös szenzorok által érzékelt vonal követését hajtja végre és ebből az állapotából két esetben lép ki:

- Valamelyik IR szenzor jelzést ad a vonal kanyarodásáról
- A robot elején található UH szenzor akadályt érzékel



4.3. ábra. A követés folyamata

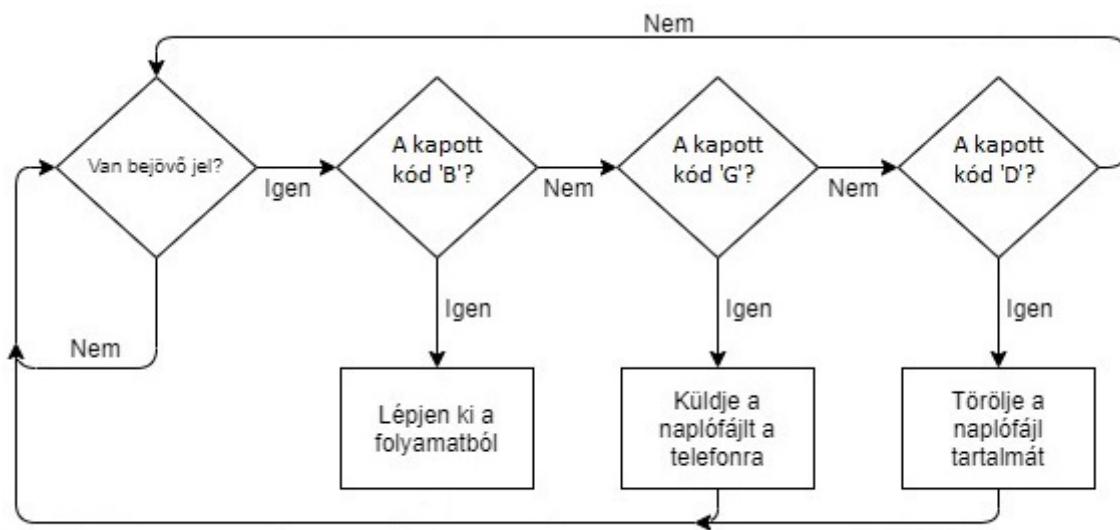
Mindaddig, míg a robot nem érzékel akadályt vagy kanyart a motorok egyforma sebességgel forognak előre. Amennyiben kanyarról ad jelzést valamely IR szenzor a robot korrigálást hajt végre a megfelelő motorok paramétereinek megváltoztatásával (nyomaték, irány). Amint a korrigálás megtörtént és a jelzést leadó szenzor visszaáll eredeti értékére, a robot visszatér a követő állapotba.

Amenyiben akadály érzékelése történik minden motor leállításra kerül, majd a robot elvégez egy pásztázást a váz elején található szervó motor és UH szenzor segítségével. A pásztázás befejezését követően az akadály megkerülése történik jobb vagy bal irányba, majd a pálya megtalálását követően a robot ismét követi azt (erről bővebben olvashatunk a 5.9. fejezetben).

4.1.4. A naplózási folyamat

A naplózási folyamat során a robot nem végez mozgást, csak a telefonnal kommunikál és az onnan kapott kód alapján három feladatot tud elvégezni:

- A folyamatból kilép, ha a bejövő kód 'B'
- A telefonra küldi a naplófájl jelenlegi tartalmát, ha a kód 'G'
- A naplófájl tartalmát üríti a memoriájából, ha a kód 'D'



4.4. ábra. Naplózási állapotok

4.2. Android képernyőtervez

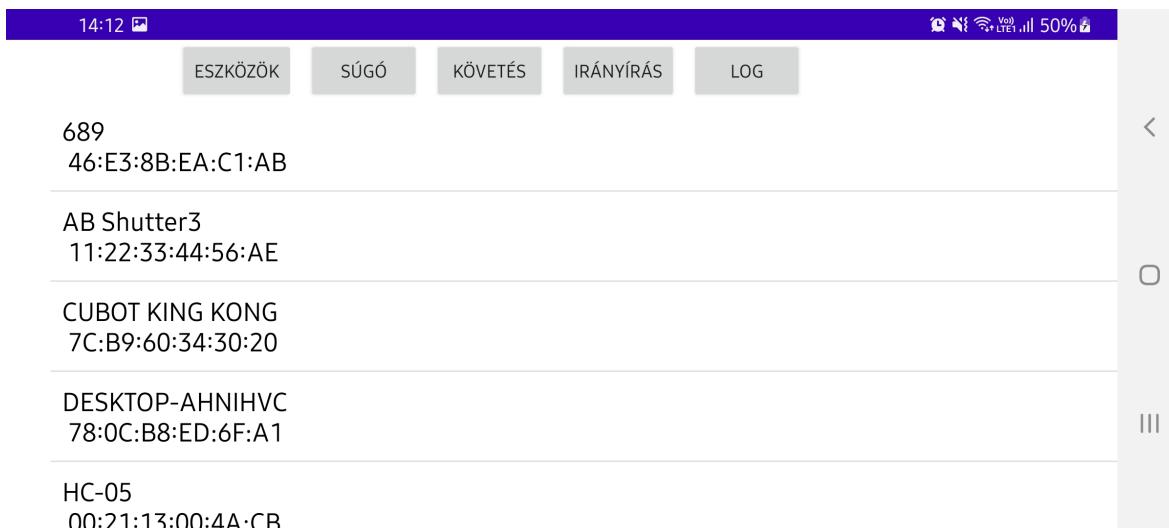
Az applikációnak négy képernyője van, ezek:

- Kezdőképernyő
- Súgó képernyő
- Automata követés képernyő
- Bluetooth irányítás képernyő
- Naplózási képernyő

A képernyők között az alkalmazásban megtalálható gombok segítségével navigálhatunk.

Kezdőképernyő: Az alkalmazás elindításakor a felhasználót ez a képernyő üdvözli, ahol öt gomb közül választhatunk, melyek különböző funkciókért felelnek, ezek:

- Eszközök: A megnyomása után a képernyőn megjelenik a párosított Bluetooth-eszközök listája, melyből ki tudjuk választani a számunkra megfelelőt.
- Súgó: Az applikációba beépített 'Súgó' képernyő válik aktívvá ahol a felhasználó minden funkciót megismerhet.
- Követés: Átirányítja a felhasználót az 'Automata követés' képernyőjére.
- Irányítás: A felhasználó a 'Bluetooth irányítás' képernyőre kerül, ahol a robotot tudjuk vezérelni.
- Log: A naplófájl megtekintésére és mentésére szolgáló képernyőre kerül a felhasználó.



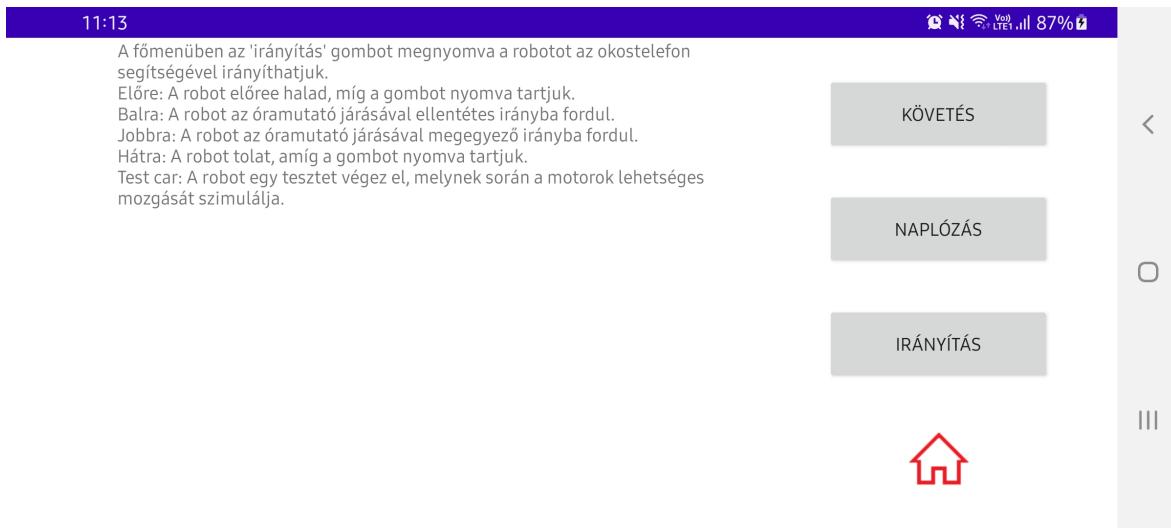
4.5. ábra. Kezdőképernyő

Súgó képernyő: A képernyő tartalmazza a funkciók rövid leírását.

- Home ikon: A kezdőlapra navigálja a felhasználót.
- Szöveg felület: A kiválasztott gomb által tartalmazott leírást jeleníti meg.
- Követés/irányítás gombok: Az applikációban található funkciók leírását jeleníti meg a szöveg felületen.

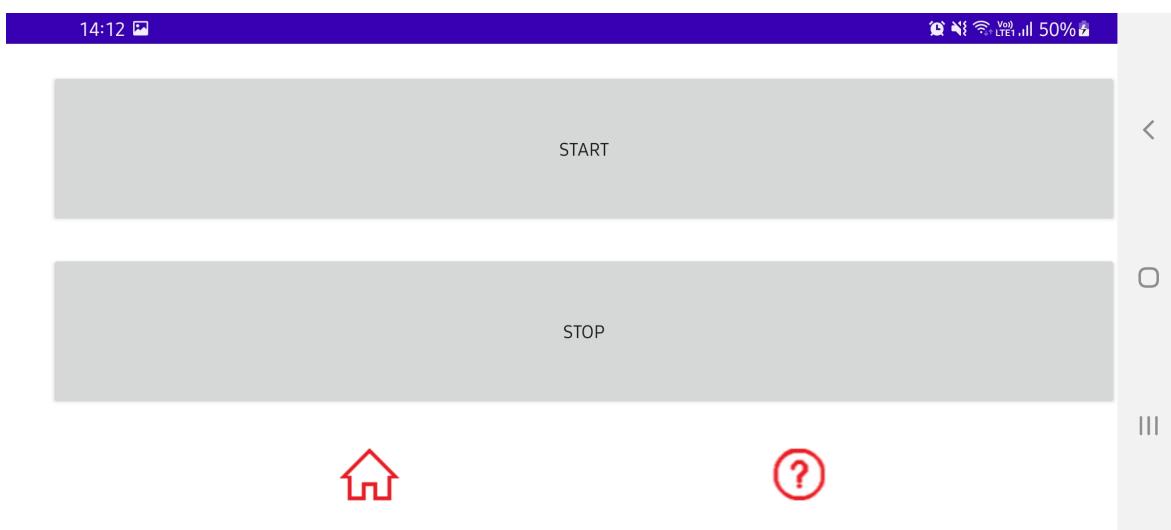
Automata követés képernyő: Amint a felhasználó erre a képernyőre kerül, a robot megkezdi a pálya követését, valamint elérhetővé válnak a 4.7 ábrán látható elemek.

- 'START' gomb: Ennek megnyomására a robot elkezdi követni a pályát.
- 'STOP' gomb: A robot megállítására szolgál.



4.6. ábra. Súgó képernyő

- HOME gomb: A kezdőlapra navigálja a felhasználót, a robot befejezi a pálya követését.
- SÚGÓ gomb: Megjelenik a Súgó oldal képernyője.

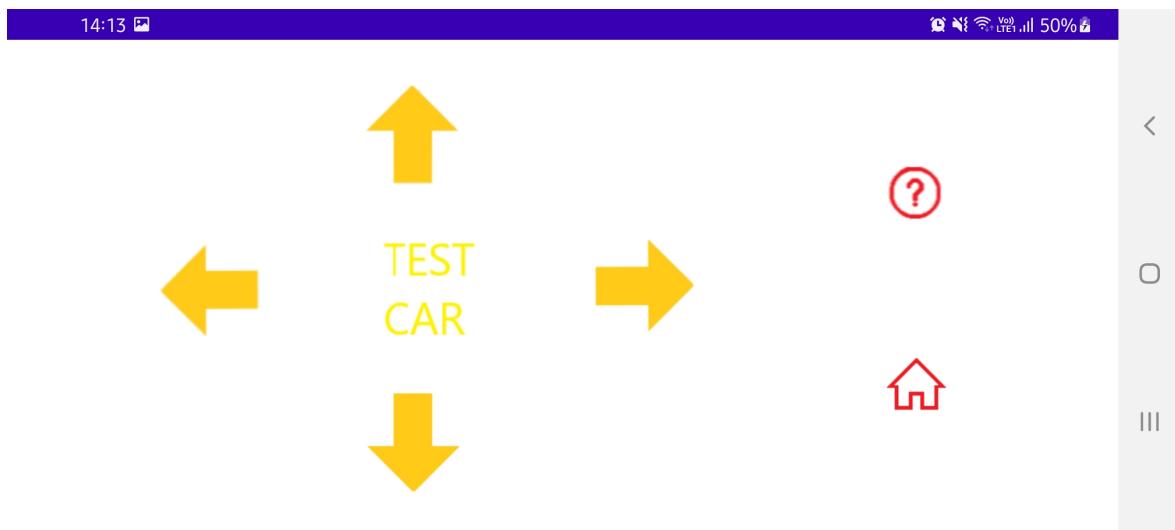


4.7. ábra. Követés képernyő

Bluetooth irányítás képernyő: Az ablakon a 4.8 ábrán látható gombok szerepelnek.

- Előre: A motorok mindegyike előre halad, megegyező nyomatékkal.
- Balra: A roboton található motorok megegyező nyomatékkal, a bal oldaliak hátra, a jobb oldaliak előre forognak, így a robot az óramutató járásával ellentétes irányba fordul.

- Jobbra: A roboton található motorok megegyező nyomatékkal, a jobb oldaliak hátra, a bal oldaliak előre forognak, így a robot az óramutató járásával megegyező irányba fordul.
- Hátra: A motorok hátra forognak megegyező nyomatékkal.
- Test car: A roboton egy előre megírt teszt mozgást futtat le, melynek segítségével a felhasználó ellenőrizni tudja, hogy a motorok működnek-e. A tesztmozgás során a robot minden motorokat érintő mozgást elvégez, melyek:
 - Előrehaladás
 - Tolatás
 - Jobb kanyar
 - Bal kanyar
 - Pásztázás
- Home: A kezdőlapra navigálja a felhasználót.
- Súgó: A felhasználó a SÚGÓ oldalra kerül.

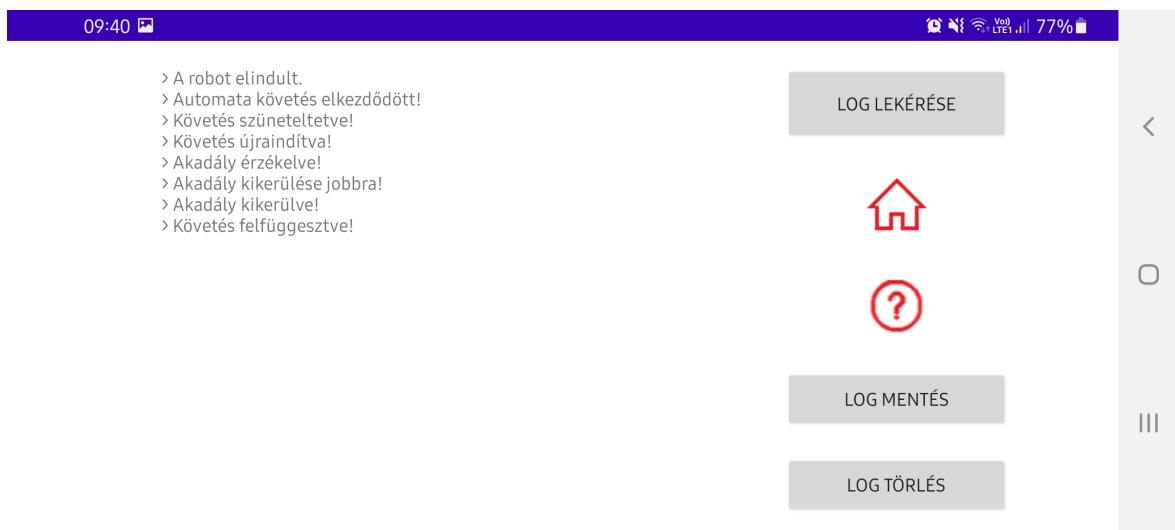


4.8. ábra. Irányítás képernyő

Naplózási képernyő: Ezen a képernyőn tudjuk lekérni a robot által rögzített naplót, valamint ezt tudjuk menteni és törölni is.

- Log lekérése: A robottól bekérjük a log fájl jelenlegi tartalmát, majd ezt megjelenítjük a 'Log felület'-en.
- Log mentés: A megjelenített napló fájlt a telefonra mentjük a jelenlegi dátumot és időt felhasználva fájlnévként.

- Log törlés: A log tartalma törlésre kerül a 'Log felület'-ről, valamint a robot memóriájából is.



4.9. ábra. Naplózási képernyő

A képernyőn megtalálható **Home** és **Súgó** gombok megnyomásakor egy üzenet jelenik meg, mely azt biztosítja, hogy a felhasználó ne felejtse el menteni a log fájlt.

Minden képernyő elérhető horizontális és vertikális elrendezésben, ezzel biztosítva a felhasználó számára a lehetőséget, hogy a neki legkényelmesebb módon tudja tartani készülékét az applikáció használata közben.

5. fejezet

A robot és az applikáció működése

Ebben a fejezetben bemutatom a robot és a telefonos applikáció működését.

5.1. A telefon és a robot összekapcsolása

Mivel a robot a telefonról kapott kód alapján kezdi el valamely folyamatot, először szükséges a kettő közötti kapcsolatot biztosítani. A robot kódjában ehhez nem kell mást tenni, csak az 5.1 kódban látható módon mindenkor, míg a HC-05-ös modulon nincs csatlakozás érzékelve a robotot váratjuk egy while() ciklus segítségével.

5.1. kód. Bluetooth csatlakozás

```
1 #include <SoftwareSerial.h>
2 SoftwareSerial myBluetooth(2,13); //Új soros portot
  ↪ példányosítunk a HC-05-ös modul számára az Rx és
  ↪ Tx-ként használt tüskék megadásával
3
4 void setup() {
5   myBluetooth.begin(9600); //Beállítjuk a modul
  ↪ sávszélességét
6 }
7
8 void loop() {
9   while(myBluetooth.available() == 0);
10 }
```

A telefonos applikációban ennél összetettebb kódra van szükség. Először biztosítani kell azt, hogy az készüléken be legyen kapcsolva a Bluetooth modul, ezt úgy tudtam elérni, hogy egy erre beépített metódust használtam[8] (5.2 kód 9. sor), és amennyiben ez igaz értéket adott vissza jeleztem a felhasználónak, majd meghívtam a 'BluetoothAdapter.ACTION_REQUEST_ENABLE' metódust[9], mellyel a felhasználó engedélyezni tudja a bekapcsolást.

5.2. kód. Eszköz kiválasztása

```

1 devices.setOnClickListener(new View.OnClickListener() {
2     ↪ {
3         @Override
4         public void onClick(View arg0) {
5             mBTAdapter = BluetoothAdapter.
6                 ↪ getDefaultAdapter();
7             if (mBTAdapter == null) {
8                 Toast.makeText(getApplicationContext(), "
9                     ↪ Nincs Bluetooth modul!", Toast.
10                    ↪ LENGTH_SHORT).show();
11             } else if (!mBTAdapter.isEnabled()) {
12                 Intent enableBT = new Intent(
13                     ↪ BluetoothAdapter.
14                     ↪ ACTION_REQUEST_ENABLE); //Meghívjuk
15                     ↪ a bekapcsolási kérelmet
16                 startActivityForResult(enableBT,
17                     ↪ BT_ENABLE_REQUEST);
18             } else {
19                 new SearchDevices().execute();
20             }
21         }
22     });
23     follow.setOnClickListener(new View.OnClickListener() {
24         @Override
25         public void onClick(View arg0) {
26             BluetoothDevice device = ((MyAdapter) (
27                 ↪ listView.getAdapter())).getSelectedItem
28                 ↪ (); //A listából kiválasztott eszköz
29             Intent intent = new Intent(
30                 ↪ getApplicationContext(),
31                 ↪ AutoFollowActivity.class);
32             intent.putExtra(DEVICE_EXTRA, device);
33             ↪
34             intent.putExtra(DEVICE_UUID, mDeviceUUID.
35                 ↪ toString());
36             intent.putExtra(BUFFER_SIZE, mBufferSize); //
37                 ↪ Az új activity indításakor a választott
38                 ↪ eszköz paramétereit is átadjuk
39             startActivity(intent);
40         }
41     });
42 });

```

Ahhoz, hogy a Bluetooth bekapcsolásához szükséges engedélyt az applikációból meg tudjuk adni az AndroidManifest fájlban használni kell a „<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>” parancsot.

Amennyiben a Bluetooth be van kapcsolva a felhasználó egyszerűen meg tudja jeleníteni a párosított eszközök listáját, majd ebből ki tudja választani azt, melyhez csatlakozni szeretne. Ezek után a 4.5 ábrán látható gombok valamelyikét kiválasztva egy új ablakba navigálhat.

Amint a felhasználó megnyomja a gombot az applikáció eltárolja a kiválasztott készülék adatait, majd az új képernyő indulásakor a csatlakozás végbemegy az eltárolt adatok segítségével[10]. A folyamat alatt a felhasználó egy általános állapotjelzőt lát, majd a csatlakozás végén visszajelzést adunk a végkimenetéről is.

5.2. A fő folyamatok eldöntése

Ahogy a 4.1 ábrán is látható a robot az indulás után három fő állapotba léphet át, melyek az applikációban való képernyőváltással idézhetők elő. minden alkalommal, mikor a kezdőképernyőt elhagyjuk és a Bluetooth csatlakozás végbe ment az applikáció automatikusan kiküldi az aktív képernyőhöz tartozó aktiváló kódot a robot részére (lásd 4.1.1 fejezet), valamint a képernyő elhagyásakor a megszakító kódot is.

Ahhoz, hogy a telefonról képesek legyünk adatok küldésére az 'AndroidManifest.xml' fájljában engedélyeznünk kell a Bluetooth modul használatát melyet a „<uses-permission android:name="android.permission.BLUETOOTH" />” paranccsal tudunk megtenni.

5.3. Kód küldése a telefonról

Ahhoz, hogy a telefonról képesek legyünk bármilyen kódot küldeni, először is létre kell hoznunk egy új BluetoothDevice példányt, mely az a kezdőképernyőn kiválasztott eszköz, melyhez csatlakozni szeretnénk és egy BluetoothSocket példányt, mellyel a telefon és a BluetoothDevice példány közötti folyamatokat tudjuk kezelni (5.3 kódrészlet).

5.3. kód. Kapcsolathoz szükséges példányosítás

```
1 private UUID mDeviceUUID ;
2 private BluetoothDevice mDevice ;
3 private BluetoothSocket mBTSocket ;
4
5 mDevice = b.getParcelable(MainActivity.DEVICE_EXTRA) ;
6 mDeviceUUID = UUID.fromString(b.getString(MainActivity
    ↪ .DEVICE_UUID)) ;
7
8 mBTSocket = mDevice.
    ↪ createInsecureRfcommSocketToServiceRecord(
    ↪ mDeviceUUID) ;
9 mBTSocket.connect() ;
```

A kód egyszeri kiküldését a lenyomott gombhoz tartozó 'onClickListener' használatával el lehet érni, ahol a gomb lenyomásakor a példányosított BluetoothSocket 'getOutputStream().write()' metódusát hívjuk meg a kód elküldéséhez.

Az iránykód folyamatos küldéséhez az 'onClickListener' helyett 'onTouchListener' kell használni. Az utóbbinál a gombtól kapott 'MotionEvent' alapján tudjuk eldönten, hogy az adott gomb jelenleg le van-e nyomva (ACTION_DOWN) vagy sem (ACTION_UP). Ameddig a gomb nyomva van az 'mBTSocket.getOutputStream().write()' metódus segítségével az iránykódot küldjük a robotnak, majd mikor felengedésre kerül a gomb ugyanezen metódussal egyszeri alkalommal küldünk adatot.

5.4. kód. Kód küldése telefonról

```

1  forward . setOnTouchListener ( new View . OnTouchListener () {
2      ↪ {
3          @Override
4          public boolean onTouch ( View v , MotionEvent event ) {
5              ↪ {
6                  switch ( event . getAction () ) {
7                      case MotionEvent . ACTION_DOWN:
8                          try {
9                              mBTSocket . getOutputStream () . write (
10                                 ↪ forwardAction );
11                         } catch ( IOException e ) {
12                             e . printStackTrace ();
13                         }
14                         return true ;
15                     case MotionEvent . ACTION_UP: //Az
16                         ↪ onClickListener () szintaxisa
17                         ↪ megegyezik ezzel
18                         try {
19                             mBTSocket . getOutputStream () . write (
20                                 ↪ stopAction );
21                         } catch ( IOException e ) {
22                             e . printStackTrace ();
23                         }
24                         return false ;
25                 }
26             }
27         });
28     });
29 }
```

5.4. Motorok kezelése

A roboton található 4 db villanymotor és 1 db szervo motor kezelésére egy motorvezérlő panelt használtam, mely L293D típusú motorvezérlőt tartalmaz. A kódban minden motort külön kell példányosítani.

5.5. kód. Motorok kezelése

```
1 #include <AFMotor.h> //A shield könyvtára
2 #include <Servo.h>
3
4 AF_DCMotor rightFront(1, MOTOR12_1KHZ); //M1 port
5 AF_DCMotor leftFront(2, MOTOR12_1KHZ); //M2 port
6 AF_DCMotor leftRear(3, MOTOR12_1KHZ); //M3 port
7 AF_DCMotor rightRear(4, MOTOR12_1KHZ); //M4 port
8
9 Servo myservo;
10
11 void setup() {
12     myservo.attach(10);
13 }
14
15 void loop() {
16     forwardAction(100);
17 }
```

A kódban láthatóan a 4 db villanymotor példányosításakor meg kell adni, hogy a panelen melyik csatlakozóhoz vannak kötve, valamint a típust is.

Ezen felül a panelhez szükséges behivatkozni az 'AFMotor.h' könyvtárat, illetve az 1 db szervo motor kezeléséhez szükséges a 'Servo.h' könyvtár is.

Miután a 4 db villanymotor példányosítása megtörtént, megtesszük ezt az 1 db szervo motorral is. A program 'setup()' részében a 'myservo.attach()' utasítással tudjuk megadni, hogy melyik port felel ezért a motorért. – Jelen esetben ez a digitális 10-es port lesz.

Amennyiben ezek megtörténtek a szervo motort a 'myservo.write()' utasításban paraméterként megadott értékkel tudjuk forgatni 0-180° között, a 4 db villanymotort pedig az 5.6 kódban látható módon tudjuk kezelni itt paraméterként a motorok forgásának sebességét kell megadnunk. – 'FORWARD' helyett használhatunk 'BACKWARD' parancsot, hogy hátra forgassuk a motort, illetve 'RELEASE' parancsot a motorok leállítására.

Az egyes irányokhoz írt metódusok csak a 4 db villanymotor forgásának irányában különböznek, illetve kanyarodáskor két paramétere van szükség, melyek azt konkretizálják, hogy a motorok milyen sebességgel forognak előre és hátra, megálláskor pedig nincs szükség paramétere, ugyanis ekkor a motorok sebessége 0.

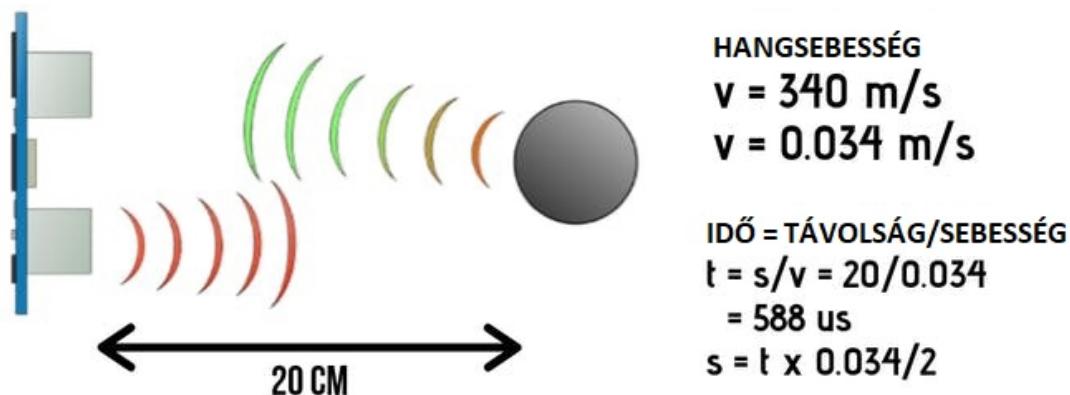
5.6. kód. Előrehaladás

```

1 void forwardAction( int sebesseg ) {
2     leftFront . run (FORWARD) ;
3     leftFront . setSpeed ( sebesseg ) ;
4     rightFront . run (FORWARD) ;
5     rightFront . setSpeed ( sebesseg ) ;
6     rightRear . run (FORWARD) ;
7     rightRear . setSpeed ( sebesseg ) ;
8     leftRear . run (FORWARD) ;
9     leftRear . setSpeed ( sebesseg ) ;
10 }
```

5.5. Akadályok érzékelése

Annak érdekében, hogy a robot ne ütközzön akadálynak a kijelölt pálya követése közben a szervo motorra egy tartóelem segítségével felhelyeztem egy HC-04 típusú ultrahang szenzort, mely két részből áll, egy bemenetből és egy kimenetből. A bemenet olvasása az Echo tüskén, a kimenet írása pedig a Trig tüskén megy végbe. Alábbiakban ennek működését fogom ismertetni.



5.1. ábra. HC-SR04

Ahogy az 5.1 ábra is mutatja a roboton található ultrahang szenzor a jel kiküldése és annak visszaérkezése között eltelt idő alapján határozza meg a távolságot[7]. Mivel a hang levegőben való terjedési sebessége $0.034 \text{ cm}/\mu\text{s}^1$, a távolságot meg tudjuk határozni úgy, hogy a mért időt megszorozzuk ezzel a sebességgel, majd az így kapott értéket elosztjuk kettővel. A kettővel való osztásra azért van szükség, mert a kiküldött jelnek az akadály és a szenzor között oda-vissza meg kell tennie az utat.

¹ centiméter/mikroszekundum

A programban ezt a 2.2. fejezetben megfogalmazottaknak eleget téve az 5.7 kódrészletben látható módon egy metódusba építettem, ahol a szenzor Trig tüskéjén kiküldök egy jelet $10\mu\text{s}^2$ ideig, majd az Echo tüskén olvasott értéket eltárolom egy változóban, ami a jel kiküldése és visszaérkezése között eltelt időt jelöli. Ahhoz, hogy a távolságot megkapjuk ezt az értéket a 5.1 ábrán láthatóan meg kell szorozni 0.034-el, majd osztani 2-vel.

5.7. kód. Akadályérzékelés

```

1 long measureDistance () {
2     digitalWrite ( trigPin , LOW ) ;
3     delayMicroseconds ( 2 ) ;
4     digitalWrite ( trigPin , HIGH ) ;
5     delayMicroseconds ( 10 ) ;
6     digitalWrite ( trigPin , LOW ) ;
7
8     duration = pulseIn ( echoPin , HIGH ) ;
9
10    distance = duration * 0.034 / 2 ;
11
12    return distance ;
13 }
```

5.6. A pályát jelölő vonal követése

5.8. kód. Pálya követése

```

1 if ( digitalRead ( leftIRSensor )==0 && digitalRead (
2     ↪ rightIRSensor )==0){ forwardAction ( 100 ) ; }
3 else if ( digitalRead ( leftIRSensor )==0 && ! digitalRead (
4     ↪ rightIRSensor )==0){ rightAction ( 100 , 130 ) ; }
5 else if ( ! digitalRead ( leftIRSensor )==0 && digitalRead (
6     ↪ rightIRSensor )==0){ leftAction ( 100 , 130 ) ; }
7 else if ( ! digitalRead ( leftIRSensor )==0 && ! digitalRead (
8     ↪ rightIRSensor )==0){ stopAction ( ) ; }
```

Ahhoz, hogy a robot követni tudja a pályát jelölő fekete vonalat kettő HW-201 típusú infravörös szenzort használtam. Abban az esetben, ha a szenzorok valamelyike jelez a vonal kanyarodásáról a robot a megfelelő irányba történő korrigálással reagál, melynek kódja az 5.8 kódrészleten látható.

Amennyiben egyik szenzor se ad le jelzést a robot előre halad, amikor az egyik infravörös szenzor jelzést ad le, akkor a szenzorral egy oldalon (bal vagy jobb) található

² mikroszekundum

motorok hátra, az ellentétes oldalon lévők pedig előre forognak, így a robot korrigálást hajt végre, ha pedig minden szenzor jelez, a motorokat megállítjuk.

5.7. Akadály kikerülése

A megfogalmazott követelmények alapján szükséges volt, hogy a robot mérettől függetlenül tudjon akadályt kikerülni.

Erre az 5.9 kódrészlet tartalmazza a megoldást. A kódrészletben látható, hogy amint egy akadályt érzékelünk a motorok megállításra kerülnek, majd meghívásra kerül az 'ObstacleAvoidance()' metódus, melyben az alábbi módon történik az akadály kikerülése.

5.9. kód. Pásztázás

```
1 distance = measureDistance();  
2 if (distance < 10){  
3     stopAction();  
4     delay(100);  
5     ObstacleAvoidance();  
6 }  
7  
8 void ObstacleAvoidance(){  
9     //Az akadály érzékelése után egy kicsit hátrál a  
10    ↪ robot  
11    backwardAction(100);  
12    delay(250);  
13    stopAction();  
14    //Megvizsgáljuk, a jobb oldalt  
15    myservo.write(0);  
16    delay(100);  
17    rightObstacle = measureDistance();  
18    delay(100);  
19    //Megvizsgáljuk a bal oldalt is  
20    myservo.write(180);  
21    delay(100);  
22    leftObstacle = measureDistance();  
23    delay(100);  
24  
25    if (leftObstacle < rightObstacle){  
26        ↪ avoidObstacleToRight(); }  
27    else{ avoidObstacleToLeft(); }  
28 }
```

Először a robot hátrál, hogy véletlenül se ütközzön az akadálynak kikerülés közben. Ezután az ultrahang szenzor segítségével megvizsgáljuk a bal és jobb oldalakat, hogy merre tudjuk kikerülni az akadályt. Amikor a szenzort elfordítjuk valamelyik oldalra

egy mérést hajtunk végre és a kapott értéket eltároljuk, majd amelyik irányból nagyobb értéket kaptunk vissza, arra kerüljük meg az akadályt. Az 'if' és 'else' ágakban található metódusok felelnek ezért.

Az akadály kikerülésekor fontos, hogy a robot el tudja dönten, hogy mikor érte el az akadály szélét. Ennek elérésére egy egyszerű do-while ciklust használtam, melyben a robot addig halad előre, míg az akadály aktuális távolsága maximum 5cm-el haladja meg az eredeti mért távolságot.

Ezek után ismételten meg kellett találni az akadály szélét, hogy széltében is meg tudjuk kerülni azt.

Az akadály kikerülése után a robotnak ismételten meg kell találnia a követendő vonalat. Ezt úgy tudtam elérni, hogy a robot egyenesen halad mindaddig, míg valamely infravörös szenzor jelzést nem ad a vonal érzékeléséről és a robot a jelzést leadó szenzorral ellentétes irányba fordul.

5.8. A naplófájl feldolgozása és mentése

5.10. kód. Naplófájl feldolgozása

```

1 — Arduino kód —
2 void logEvent(String event){ logfile += event; }
3 myBluetooth.println(logfile); //Az adat küldése a
   ↪ telefonra
4
5 — Android kód —
6 inputStream = mBTSocket.getInputStream();
7 while (!bStop) {
8     byte[] buffer = new byte[256];
9     if (inputStream.available() > 0) {
10         inputStream.read(buffer);
11         int i;
12         for (i = 0; i < buffer.length && buffer[i] !=
           ↪ 0; i++) {
13             displayedLog = new String(buffer, 0, i);
14         }
15         displayedLog = displayedLog.replaceAll("_", "\n"
           ↪ + ">_");
16         logTextView.setText(displayedLog); //A fogadott
           ↪ adat formázása és megjelenítése
17     }
18     t.sleep(500);
19 } //A bejövő adat fogadása

```

A robot működés közben az adott eseményeket egy 'string' típusú változóban tárolja el a 5.10 kódrészletben látható módon. Az adat átküldése a telefonra a 'println()' funkcióval.

metódussal történik az applikáció pedig a 'getInputStream()' metódussal fogadja ezt és ezek után a while() cikluson belül feldolgozza azt.

A feldolgozás megvalósításának szükségességről részletesebben is olvashatunk a 6.2.2. fejezetben.

Miután az adatot fogadtuk, feldolgoztuk és megjelenítettük a felhasználónak lehetsége van menteni is ezt. A mentést a 4.9 ábrán látható 'Log mentése' gombbal tudja megtenni a felhasználó. A naplóbájl neve az adott dátum lesz másodperc pontossággal, amelynek megvalósítását a 5.11 kódban láthatják.

A fájlnév eltárolása után példányosítunk egy új 'FileOutputStream' egyedet, majd ennek paraméterben megadjuk a mentendő fájl nevét és mentési módját.

Ezek után szükség van egy 'OutputStreamWriter' példányra, hogy a létrehozott fájl tartalmát módosítani tudjuk. Ha létrehoztuk a saját egyedünket a beépített 'write()' metódusát használva a fájl tartalmát módosítjuk a LogFelület tartalmával.

5.11. kód. Napló mentése

```
1 <uses-permission android:name="android.permission.  
    ↪ WRITE_EXTERNAL_STORAGE" /> //Az AndroidManifest-  
    ↪ hez hozzáadandó kód  
2  
3 if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M &&  
    ↪ checkSelfPermission(Manifest.permission.  
    ↪ WRITE_EXTERNAL_STORAGE) != PackageManager.  
    ↪ PERMISSION_GRANTED) //Megvizsgáljuk, hogy az  
    ↪ alkalmazás rendelkezik-e a szükséges engedélyel  
4     requestPermissions(new String []{ Manifest.  
    ↪ permission.WRITE_EXTERNAL_STORAGE} ,1000);  
5  
6 SimpleDateFormat sdf = new SimpleDateFormat("yyyy.MM.  
    ↪ dd.HH.mm.ss" , Locale.getDefault());  
7 File_name = "Naplóbájl:" + sdf.format(new Date()) + "  
    ↪ .txt"; //A naplóbájl nevének megadása  
8  
9 FileOutputStream myStream = null;  
10  
11 myStream = openFileOutput(File_name , MODE_PRIVATE); //  
    ↪ Létrehozunk a telefonon egy új fájlt az eltárolt  
    ↪ névvel  
12 OutputStreamWriter myWriter = new OutputStreamWriter(  
    ↪ myStream);  
13 myWriter.write(logViewContent); //Fájl tartalmának  
    ↪ módosítása  
14 myWriter.flush();  
15 myWriter.close();  
16 msg("Napló_mentve:" + File_name + "néven!" , 1);
```

Ahhoz, hogy a fájlt létre lehessen hozni és módosítani tudjuk a tartalmát a megfelelő engedély biztosítása szükséges.

Ehhez engedélyt kell kérni a felhasználótól, hogy az alkalmazás hozzáférhessen a telefon tárhelyéhez. Az engedély megadásához először ellenőriznünk kell, hogy az alkalmazás rendelkezik-e az engedéllyel. Ezt az ?? kódrészletben látható beépített metódussal lehet megtenni, majd a kérés után fel is kell dolgozni a felhasználó által hozott döntést, amire az 'onRequestPermissionsResult' metódust kell használni.

Amennyiben a felhasználó nem adja meg az engedélyt a naplófájl mentése nem megy végbe, helyette egy üzenet jelenik meg a képernyőn, mely figyelmeztetést ad az engedély hiányáról, majd a felhasználó számára egy útmutatást ad arról, hogy a hiányzó engedélyt miképp tudja megadni.

6. fejezet

Fejlesztés közbeni tapasztalatok

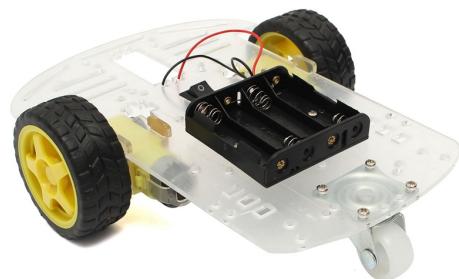
A fejlesztés közben több hardver és szoftverszintű új tapasztalatot is szereztem. Hardver szinten előfordultak problémák is elsősorban a robotnál az egyes elemek rögzítése kapcsán.

6.1. Hardverrel kapcsolatos tapasztalatok

Nem csak a robot hardvere jelentett néha problémát, hanem a használt telefoné is. A 6.1.2 fejezetben bővebben olvashatunk arról, hogyan befolyásolja a telefon hardvere a projekt teljesének működését.

6.1.1. A robot építésekor szerzett tapasztalatok

A robot fejlesztése közben a legnagyobb problémát a megfelelő alváz megtalálása jelentette. Amikor eldöntöttem, hogy milyen robotot szeretnék építeni meglátogattam különböző fórumokat, hogy nagyjából megtudjam milyen mennyiségű szenzorra, illetve motorra lesz szükségem.



6.1. ábra. Két kerekű Arduino alváz

Sok már létező projektben a 6.1 képen látható alvázat használták 'autó' építésére, azonban felmerült bennem a kérdés, hogy hogyan tudom rögzíteni a szükséges infravörös szenzorokat, valamint lesz-e elegendő hely minden szenzornak és az alaplapnak is, ezért úgy döntöttem, hogy ehelyett a megoldás helyett egy saját alvázat használok 4 villanymotorral.

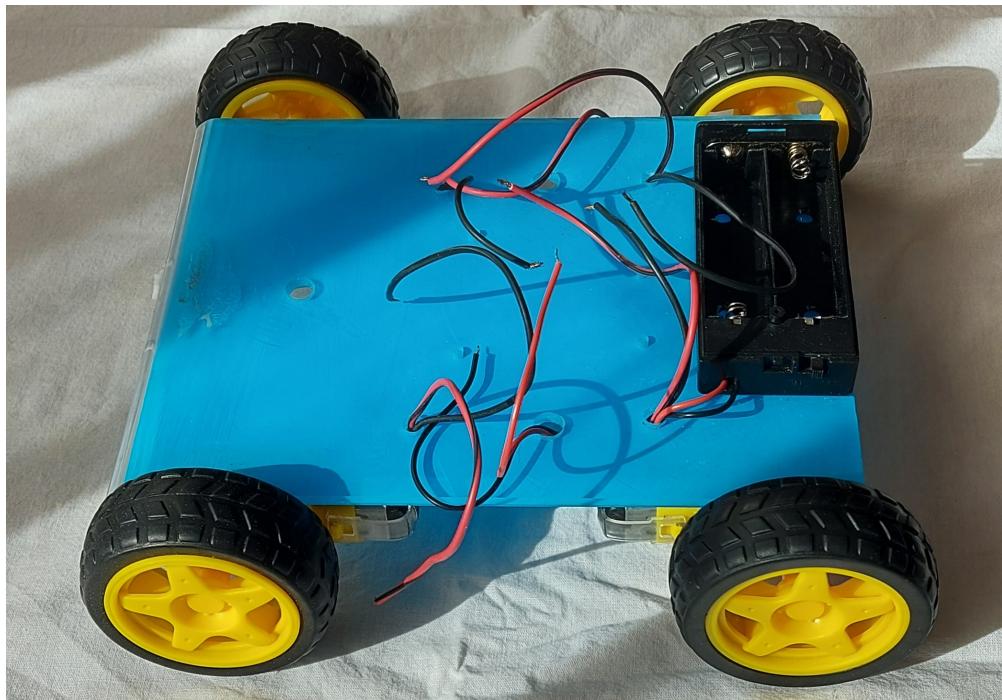
Az alvának elég nagynak kellett lennie ahhoz, hogy a motorok elférjenek rajta, viszont túl nagy se lehetett, ugyanis egy nagy alváz magában hordozhatja azt a hiba-lehetőséget, hogy a motorokat nagy nyomatékkal kell használni, hogy a robot mozogni tudjon és ez hamar merítené le az akkumulátort.

Továbbá olyan alváz használatát részesítettem előnyben, mely nem vezeti az elektromosságot, mivel a használt Arduino UNO alaplap alján a forrasztási pontok szabadon vannak felmerült bennem az a kérdés, hogy biztonságos-e egy olyan alváz használata, mely vezető vagy félvezető, vagy esetleg zárlatot fog-e ez okozni.

Miután nem találtam egy fórumon se konkrét választ erre a kérdésemre úgy döntöttem, hogy műanyag alvázat használok.

A robot első verziójának megépítése után azonban rájöttem arra, hogy az alvának merevnek is kell lennie. Az első tesztek során arra lettem figyelmes, hogy vannak alkalmak amikor a robot minden probléma nélkül képes bevenni egy kanyart, majd a következő alkalommal amikor ehhez a kanyarhoz ér már nem.

Ennek oka az volt, hogy a műanyag alváz rugalmassága és a középen elhelyezkedő akkumulátor súlya miatt a kerekek nem teljes felületükkel érintkeztek a talajjal és ezért a tapadásuk is csökkent.



6.2. ábra. A robot alváza

Mivel nincs birtokomban 3D nyomtató mellyel egy olyan alvázat tudnék nyomtatni, mely merevsége és mérete is megfelelő úgy döntöttem, hogy egy plexi lemezt fogok használni (másik alternatíva lehet a plexi lemeznél olcsóbb SAN¹ lemez is). A plexi használatát több okból is jó választásnak tartom.

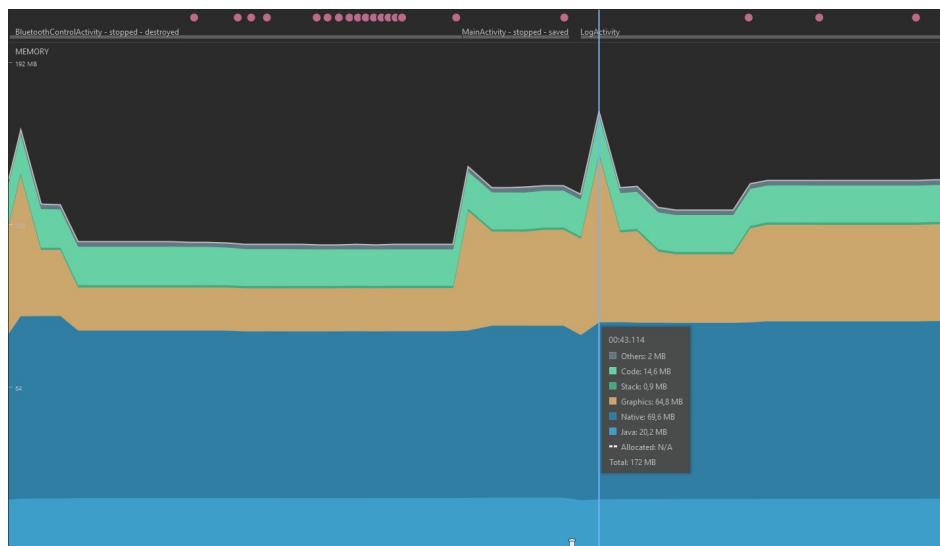
- A merevsége miatt nagy terhelés alatt sem deformálódik, így a kerekek minden esetben teljes felületükkel fognak a talajjal érintkezni
- Amennyiben módosítani kell a kinézetét rövid melegítés után lehet hajtani/formázni, miközben nem kell attól tartanunk, hogy eltörök

Miután az alvázat kiválasztottam a használt kábelek elvezetésére/védelemre kellett megoldást találnom. Ezt azért tartottam fontosnak, mert ha szabadon vannak a kábelek előfordulhat, hogy a robot olyan helyen halad át, ahol valami ezekbe beleakadhat, így kimozdítva a helyéről, vagy a tüskét, melyhez csatlakozik elhajlíthatja.

Ennek megoldására az alvázra több lyukat is fúrtam, melyek segítségével a kábeleket az alváz alján tudom elvezetni a szenzortól/motortól az alaplapig, továbbá nem használtam hosszabb kábelt, mint amire szükségem volt.

6.1.2. A telefon hardverére vonatkozó tapasztalatok

A fejlesztés során a 3.3 fejezetben található specifikációkkal rendelkező telefonokat használtam. A modernebb és erősebb hardverrel ellátott Smasung Galaxy A71 esetében nem volt probléma, azonban egy régebbi készüléken az alkalmazás memória és energiafogyasztása is jelentősen megnőtt. A megnövekedett energiafogyasztás a készülék gyorsabb merülését eredményezni.



6.3. ábra. A71 memóriafelhasználása

¹ sztirol-akril-nitril

A memóriafelhasználás mértéke nem a felhasznált memória mennyiségében, hanem a teljes memóriához viszonyított százalékában emelkedett. Amíg a teszteléshez használt A71 készüléken a program a 6.3 ábrán látható módon a telefon memóriájának megközelítőleg 3%-át használta, addig ez a szám a régebbi típusú 512 MB memóriával rendelkező Alcatel OneTouch Pop D1 telefon esetében már 30-40% közötti érték volt.

További tapasztalom a telefonban található hardverre nézve a Bluetooth modul típusához köthető. A legjelentősebb különbséget a távoli irányításban tapasztaltam. Az A71 esetében akár egy teljes emelet távolság is lehetett a telefon és a robot között a csatlakozás és az irányítás is minden probléma nélkül végbement, azonban a Pop D1 esetében ilyen távolságról a csatlakozás sikertelen volt.

A Bluetooth modulhoz köthető másik észrevételelem a kezdeti irányítás tesztelésekor a parancs telefonról való kiküldése és a robot mozdulása között eltelt idő mértékében volt. Az A71 esetében ez az érték nagyjából 300ms volt egy szoftveres hiba következtében melyről a 6.2.1. fejezetben bővebben olvashatunk, a Pop D1 esetében pedig ez az érték közel kétszeresre nőtt.

A várakozási idő szoftveres okának kijavítása után az A71-nél már nem volt probléma a parancs kiadása és a végrehajtás között minimális idő telt el, azonban a Pop D1-nél egy megközelítőleg 40-50ms késleltetést tapasztaltam, mely a készülékben megtalálható Bluetooth modul miatt következett be.

További hardverhez köthető észrevételelem volt, hogy az A71-ben található több maggal és magasabb órajellel rendelkező processzornak köszönhetően a program futása közben a képernyők közötti váltás, valamint az adott képernyő orientációjában betörtént változáskor is a program gyors és sima átmenettel működött, a kevesebb magszámú és alacsonyabb órajelű Pop D1-nél pedig akadásokra, esetenként a program összeomlására lettem figyelmes.

A másik észrevételelem a telefonok melegedésében volt. Az A71 esetében a processzor kihasználtsága 2% alatt volt és az orientáció váltásakor emelkedett 4-5%-ra, viszont a Pop D1-nél az érték 30-50% között mozgott, valamint az orientáció váltásakor a 60-70%-ot is elérte.

6.2. Szoftveres észrevételek

A fejlesztés közben a szoftverrel kapcsolatban két olyan esettel találkoztam, ahol a probléma megoldásához hosszabb keresést kellett végrehajtanom. Ezek:

- A robot irányításakor fellépő késleltetés
- A naplófájl fogadása

6.2.1. A késleltetéssel kapcsolatos probléma megoldása

A fejlesztés elején amint a telefon és a robot közötti kapcsolódást megoldottam az adatokat először a telefonról String változóban eltárolt számsorozatokként küldtem a robot részére, majd a roboton futó kódban a fogadáskor egy parseInt() függvényt használtam arra, hogy ezt tényleges számkóddá alakítsam, azonban ezek után jelentős késleltetést tapasztaltam.

Első gondolatként azt hittem, hogy a használt HC-05 modul okozza a problémát, azonban fórumokon talált projekteken is ilyen modult használtak és gyorsan működött, ezért elkezdtem kísérletezni az átküldött adatok méretével és típusával, majd észrevettem, hogy amikor egy-egy karaktert küldtem át és fogadtam read() metódussal a késleltetés megszűnt.

6.2.2. Naplófájl fogadásának megvalósítása

A fejlesztés során a naplófájl fogadása jelentette a legnagyobb kihívást, ugyanis a legtöbb már létező projektben nem használtak naplázási funkciót, ezért utánakerestem, hogy egy csatlakoztatott eszközről hogyan tudunk bejövő adatot fogadni[11].

A mintaként talált kódon módosításokat kellett elvégeznem. Első sorban azt terveztem, hogy a naplázási eseményeket a megtörténés után rögtön meg is jelenítem a telefonon, azonban ez az applikáció összeomlását okozta, melynek okára nem tudtam rájönni se hibakereséssel, se fórumon nem találtam megoldást.

Ezek fényében úgy döntöttem, hogy a napló feldolgozását egy külön képernyőn végzem el ezzel azt akartam biztosítani, hogy a naplázás alatt ne legyen más folyamat. Miután ezt megettettem szerettem volna biztosítani azt is, hogy csak akkor menjen végbe a beolvasás, amikor van bejövő jel, amit el is tudtam érni egy egyszerű 'if' segítségével. Ezek után néhány esetben az adat egyes részei felülírták egymás, aminek javítását egy 'for' ciklussal tudtam megoldani.

Miután az adatot fogadtam nem voltam kész, ugyanis ezt még formáznom is kellett, hogy az általam elképzelt módon jelenjen meg a naplázási felületen. Ennek eléréséhez nem kellett mást tennem, mint a String típusú változókhöz beépített replaceAll() metódust meghívni és az eredeti szövegben a jelölő karaktereket, sortörésre és '>' jelre cserélnem.

```
_Naplóesemény 1_Naplóesemény 2_Naplóesemény 3..._Naplóesemény N  
> Naplóesemény 1  
> Naplóesemény 2  
> Naplóesemény 3  
...  
> Naplóesemény N
```

6.4. ábra. Naplófájl átalakítása

7. fejezet

Felhasználói útmutató

7.1. A robot megépítésének lépései

A következő rész tartalmazza a robot megépítéséhez szükséges alkatrészek listáját, ezek bekötési módját, valamint a program telepítésének lépéseit. Fontos megjegyezni, hogy amennyiben a felhasználó hibásan köti be valamely alkatrészt, úgy a robot nem megfelelően fog működni!

7.1.1. Szükséges alkatrészek

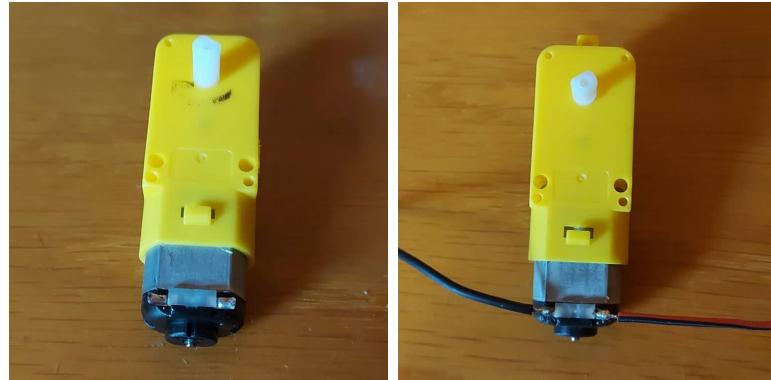
A robothoz szükséges elemek.

Megnevezés	Mennyiség
Arduino Uno mikrokontroller	1 db
Motorvezérlő shield	1 db
HC-SR04 ultrahang szenzor	1 db
HW-201 infravörös szenzor	2 db
SG90 szervo motor	1 db
Bluetooth modul	1 db
Villanymotor	4 db
Kapcsolóval ellátott akkumulátor tartó	1 db
18650-es akkumulátor	2 db
Anya-anyá jumper kábel	15 db
1mm átmérőjű sodort rézvezeték 12-17 cm hosszú	8 db

Szükséges továbbá 1 db tartóelem az ultrahang szenzor szervó motorra való rögzítéséhez, 1 db minimum 15x20cm méretű merev műanyag lap az alváznak, valamint a forrasztáshoz szükséges eszközök.

7.1.2. A motorok és az akkumulátor tartó felhelyezése

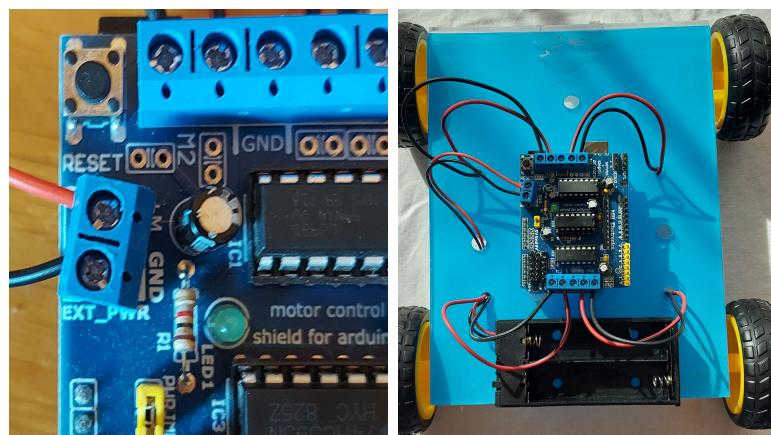
Helyezzünk a 4 db villanymotor közül egyet magunk elé úgy, hogy a csatlakozási pontok felfele nézzenek, majd ezekhez forraszunk hozzá egy-egy vezetéket. Ha végeztünk ismételjük meg a lépést a többi motorral is. (a pozitív pólushoz piros, a negatív pólushoz fekete vezeték kerüljön).



7.1. ábra. A kábelek felhelyezése

Ezek után a motorokat rögzítsük az alváz aljához, majd az akkumulátort az alváz tetejére és a hozzájuk tartozó vezetékeket a 7.2 ábrán látható módon csatlakoztassuk a motorvezérlő shield-hez. /Shield-en jelzett M1 – Jobb első, M2 – Bal első, M3 – Bal hátsó, M4 – Jobb hátsó kerékért felelős motor./

Miután a motorokat és az akkumulátor tartót rögzítettük a motorokra tegyük fel a kerekeket.



7.2. ábra. A shield-re való bekötés

7.1.3. Szenzorok felhelyezése

Helyezzük fel a robot elejére a 2 db infravörös szenzort úgy, hogy a szenzorok között legalább 2-4 cm távolság legyen (a szenzorok közötti távolság méretét a követendő vonal vastagsága határozza meg), majd jumper kábel segítségével csatlakoztassuk ezeket

a shield-en található A0 és A1-es, valamint az ehhez tartozó GND és VCC jelölésű portokhoz. (szemből nézve jobb oldali szenzor A0, bal oldali A1).

Ezek után helyezzük fel a szervó motort a szenzorok közé, majd erre a tartóelemet, a hozzá rögzített ultrahang szenzorral. Az ultrahang szenzor 'Trig' és 'Echo' jelzéssel ellátott tüskéit csatlakoztassuk a digitális 0 és 1 portokhoz, valamint GND és VCC tüskéit egy-egy GND és VCC porthoz. A szervó motor vezetékét csatlakoztassuk a shield-en található SERVO_2 jelzésű tüskesorhoz.

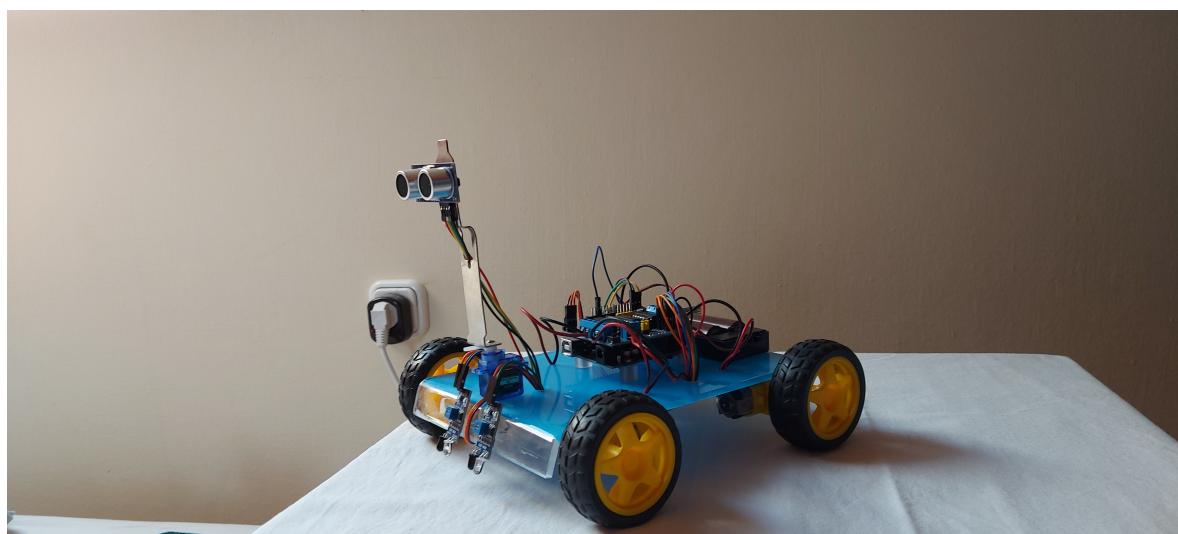
A korábbi lépések elvégzése után helyezzük fel a Bluetooth modult, majd csatlakoztassuk a TX – Transmitter és RX – Receiver tüskéket a digitális 3 és 13 portokra, valamint a VCC és GND tüskéket a szenzorokhoz hasonlóan a shield VCC és GND tüskéihez.

7.1.4. A program telepítése az alaplapra

Csatlakoztassuk a mikrokontrollert a számítógéphez. Nyissuk meg a letöltött mappában található '.ino' kiterjesztésű fájlt az Arduino IDE segítségével. Az Eszközök -> Alaplap menüpontban válasszuk ki a használt alaplap típusát, valamint a portot amelyen keresztül csatlakoztattuk a számítógéphez. Kattintsunk a feltöltés gombra.

Amint elkészül a feltöltés az IDE jelzi, ezek után az alaplapot leválaszthatjuk. A telepítés után helyezzük el a mikrokontrollert az alvázon és csatlakoztassuk hozzá a shield-et.

Utolsó lépésként a felhelyezett akkumulátor tartóba helyezzük el a 2 db 18650-es akkumulátort ügyelve a pólusok helyességére, majd a kapcsoló kapcsolásával lássuk el energiával az alaplapot, így elindítva a robotot.



7.3. ábra. A kész robot

7.2. A telefonon elvégzendő lépések

Mielőtt a robotot vezérelni tudnánk el kell végeznünk néhány lépést.

7.2.1. Az alkalmazás telepítése

Mivel az alkalmazás még nem érhető el a PlayStore-on, ezért a telepítést manuálisan kell elvégeznünk. Ennek menete:

1. A számítógépről másoljuk a készülékre az 'app-debug.apk' nevű fájlt
2. Engedélyezzük a Beállításokban az ismeretlen forrásból származó alkalmazások telepítését
3. Nyissuk meg a telefon fájlkezelő programját, majd keressük ki és futtassuk a fájlt.
4. A megjelenő üzenetetnél válasszuk a 'Telepítés' lehetőséget
5. Az installálás közben felugró 'PlayProtect' üzenetkor válasszuk a 'Telepítés mégis' opciót
6. Amennyiben megkapjuk az 'Alkalmazás telepítve' üzenetet a telepítés sikeres volt, használhatjuk az alkalmazást

Fontos megjegyezni, hogy a sikertelen telepítés főbb okai a következők lehetnek:

- A telefon nem rendelkezik elegendő szabad tárhellyel
- Az alkalmazás nem kompatibilis a készülék Android verziójával
- Nem engedélyezett az ismeretlen forrásból származó alkalmazások telepítése

7.2.2. Bluetooth párosítás

Ahhoz, hogy a robothoz csatlakozni tudunk először párosítani kell a készülékünket a roboton található HC-05 modullal. Ezt úgy tudjuk megtenni, hogy a telefonon elnavigálunk a Beállítások → Kapcsolatok → Bluetooth menüpontra, majd bekapcsoljuk a telefonban található Bluetooth egységet.

Ezek után a roboton bekapcsoljuk az akkumulátort, ezzel energiával látjuk el a HC-05-ös modult, ami ezt követően megjelenik az elérhető eszközök listájában.

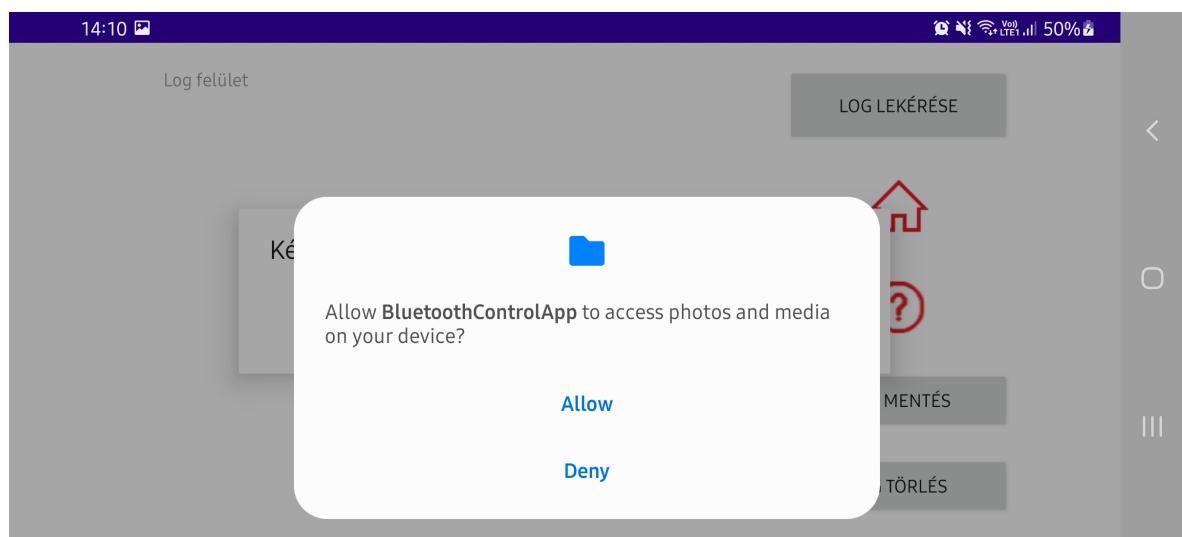
A párosításhoz ki kell választanunk az eszközt és a felugró párbeszédablakban jelzónak az '1234' számsort kell beírnunk. Amint a párosítás megtörtént az applikációt indíthatjuk.

7.2.3. Az alkalmazás első indítása

Az alkalmazás indítását követően a kezdőképernyő fogad minket (lásd 4.5 ábra). Itt az „Eszközök” gomb megnyomása után láthatóvá válik a párosított eszközök listája.

Miután kiválasztottuk a HC-05-ös szenzort a listából a „Követés”, „Irányítás” és „Log” gombok valamelyikét megnyomva a telefon csatlakozik a robothoz.

Amennyiben a „Log” gombot választjuk az applikáció a Naplázási képernyőre navigál minket, ahol az első futtatás alkalmával engedélyt kell adnunk az applikációnak a tárhely eléréséhez. Ha ezt nem tesszük meg a Naplófájlt nem fogjuk tudni menteni mindaddig, amíg az engedélyt meg nem adjuk. Az engedély megadása után az applikáció összes funkciója használható.



7.4. ábra. Engedélyadás

8. fejezet

Összegzés és továbbfejlesztés

„Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live” – John Woods

8.1. Összegzés

A projekt jelenlegi állapotában a 2.1 fejezetben megfogalmazottaknak eleget tettem, vagyis a robot képes egy előre megadott pálya követésére, a követés közben észlelt akadály(ok) kikerülésére és a pálya újbóli megtalálására. A robot továbbá irányítható egy telefonnal, valamint a telefonon keresztül el tudjuk érni a robot által vezetett naplófájlt és ezt a telefonra menthetjük.

A végtermékkel sikerült egy olyan könnyen módosítható projektet megvalósítanom, mely különböző életkorú és programozási tapasztalattal rendelkező diákok oktatására és ismereteik bővítésére is alkalmas, valamint hobbi felhasználók részére sem bonyolult.

Bár a végtermék teljes, szoftveres és hardveres fejlesztés is lehetséges, a továbbiakban ezekről olvashatunk.

8.2. Hardverre vonatkozó fejlesztés

Ahogy a 2.3 fejezetben is olvasható az éles kanyarok alkalmanként hibásan kerülnek bevételre. Ennek megoldását nyújthatja, ha a vonal érzékelésére másfajta szenzort használunk (pl.: digitális vonalkötő szenzor), mely pontosabban képes érzékelni a pálya vonalában végbemenő változást. Lehetséges megoldási módszer a motorok más formában való kezelése az IR szenzorról kapott jelzéskor (nagyobb nyomaték azokra a motorokra melyeknek előre kell mozdulni).

További hardver fejlesztés lehet egy kamera beépítése a robot elejére és ezt felhasználva az irányításért felelős okostelefonon előtérképet kaphatunk arról, hogy a robot éppen merre jár. Ennek segítségével a telefonnal való irányításkor nincs szükség arra,

hogy a robotot ténylegesen lássuk, hanem a kamera képére hivatkozva ki tudjuk ke-rülni az akadályokat, valamint egy jobb felhasználói élményt biztosít. Ehhez szükséges egy olyan telefon, amely kellő hardverrel rendelkezik egy ilyen program futtatásához, valamint a külső akkumulátor is gyorsabban fog merülni.

8.3. Android szoftverre vonatkozó fejlesztés

A szoftverben két nagy fejlesztési lehetőségre lettem figyelmes:

- Logolás fejlesztése (hardver fejlesztése is szükséges)
- Súgó felület bővítése

Amennyiben beépítésre kerül a robotra egy giroszkóp meg lehet mondani, hogy az egyes kanyarok elvégzésekor a robot milyen szögben fordult jobbra/balra, illetve az is, hogy a felületben milyen változás történik (a robot emelkedőn halad felfele, vagy lejtőn lefelé).

Ezeket az adatokat felhasználva el lehet készíteni egy statisztikát, valamint automata követéskor a kanyarodások közötti időeltérés és a kanyarok szögének ismeretében a pálya vizualizálása is lehetséges válhat.

További fejlesztési lehetőség a felhasználói útmutató implementálása az alkalmazás Súgó felületére, ugyanis ha a felhasználó nincs birtokában az említett dokumentumnak (7. fejezet) a robot összeszerelésekor nem tudhatja, hogy az adott szenzort/motort hova kell csatlakoztatnia..

8.4. A hardver biztosítására vonatkozó fejlesztés

A projekt jelenlegi állapotában fedetlenül található meg az összes szenzor, kábel, motor és a mikrokontroller is, ezért fennáll az esélye annak, hogy valamely komponens megsérül, ezzel meggátolva a robot megfelelő működését.

A komponensek biztosítására lehetséges egy olyan váz megtervezése és 3D nyomta-tása, mely védelmet biztosít a hardver elemeinek, illetve így egy olyan kinézetet lehet adni a robotnak, amilyet a felhasználó elképzel.

Irodalomjegyzék

- [1] HARSÁNYI RÉKA – JUHÁSZ MÁRTON ANDRÁS: FIZIKAI SZÁMÍTÁSTECHNIKA : ELEKTRONIKAI ALAPOK ÉS ARDUINO PROGRAMOZÁS
- [2] LAPTEVA NATALIA – TÓTH BERTALAN: PROGRAMOZZUNK C++ NYELVEN
- [3] NAGY GUSZTÁV – JAVA PROGRAMOZÁS
- [4] FEHÉR KRISZTIÁN: ALKALMAZÁSFEJLESZTÉS ANDROID STÚDIÓ RENDSZERBEN
- [5] [HTTPS://WWW.ARDUINO.CC/EN/GUIDE/ENVIRONMENT](https://www.arduino.cc/en/Guide/Environment)
Megtekintés: 2021.04.10.
- [6] [HTTPS://WWW.OKTATAS.HU/KOZNEVELES/KERETTANTERVEK/2020_NAT](https://www.oktatas.hu/kozneveles/kerettantervek/2020_nat)
Megtekintés: 2021.04.11.
- [7] [HTTPS://CREATE.ARDUINO.CC/PROJECTHUB/ABDULARBI17/ULTRASONIC-SENSOR-HC-SR04-WITH-ARDUINO-TUTORIAL-327FF6](https://create.arduino.cc/projecthub/abdularbi17/ultrasonic-sensor-hc-sr04-with-arduino-tutorial-327ff6)
Megtekintés: 2020.10.22.
- [8] [HTTPS://STACKOVERFLOW.COM/QUESTIONS/7672334/HOW-TO-CHECK-IF-BLUETOOTH-IS-ENABLED-PROGRAMMATICALLY](https://stackoverflow.com/questions/7672334/how-to-check-if-bluetooth-is-enabled-programmatically)
Megtekintés: 2021.01.28.
- [9] [HTTPS://STACKOVERFLOW.COM/QUESTIONS/3806536/HOW-TO-ENABLE-DISABLE-BLUETOOTH-PROGRAMMATICALLY-IN-ANDROID](https://stackoverflow.com/questions/3806536/how-to-enable-disable-bluetooth-programmatically-in-android)
Megtekintés: 2021.01.28
- [10] [HTTPS://STACKOVERFLOW.COM/QUESTIONS/5171248/PROGRAMMATICALLY-CONNECT-TO-PAIRED-BLUETOOTH-DEVICE](https://stackoverflow.com/questions/5171248/programmatically-connect-to-paired-bluetooth-device)
Megtekintés: 2021.01.28.
- [11] [HTTPS://STACKOVERFLOW.COM/QUESTIONS/45140098/HOW-TO-SEND-RECEIVE-MESSAGES-VIA-BLUETOOTH-ANDROID-STUDIO](https://stackoverflow.com/questions/45140098/how-to-send-receive-messages-via-bluetooth-android-studio)
Megtekintés: 2021.03.11.

NYILATKOZAT

Alulírott Bozó Tamás Dániel, buntetőjogi felelősségem tudatában kijelentem, hogy az általam benyújtott, Arduino robotfejlesztés android alkalmazás használatával című szakdolgozat (diplomamunka) önálló szellemi termékem. Amennyiben mások munkáját felhasználtam, azokra megfelelően hivatkozom, beleértve a nyomtatott és az internetes forrásokat is.

Tudomásul veszem, hogy a szakdolgozat elektronikus példánya a védés után az Eszterházy Károly Egyetem könyvtárába kerül elhelyezésre, ahol a könyvtár olvasói hozzájuthatnak.

Kelt: Kisböröe ..., 2021. év április hó 27 ... nap.

Bozó Tamás Dániel.....
aláírás