



TERMÉSZETTUDOMÁNYI KAR - TTK

# Arduino robotfejlesztés android alkalmazás használatával

**Készítette**

Bozó Tamás Dániel

Programtervező informatikus

**Témavezető**

Balla Tamás

EGER, 2021

# Tartalomjegyzék

<b>1. Arduino és Android bemutatása</b>	<b>6</b>
1.1. Az Arduino . . . . .	6
1.1.1. A platform felépítése . . . . .	6
1.1.2. Elérhető alaplapok . . . . .	6
1.1.3. Memória és Háttértár . . . . .	7
1.1.4. Az IDE felépítése és használata . . . . .	7
1.1.5. A C++ nyelv . . . . .	8
1.2. Az Android . . . . .	8
1.2.1. Az operációs rendszer története . . . . .	8
1.2.2. Az Android Studio felépítése és használata . . . . .	10
1.2.3. A Java nyelv . . . . .	10
<b>2. Követelményspecifikáció</b>	<b>11</b>
2.1. Vágylomrendszer leírása . . . . .	11
2.2. Követelmények . . . . .	12
2.3. Jelenlegi helyzet leírása . . . . .	12
<b>3. Funkcionális specifikáció</b>	<b>13</b>
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 fejlesztés során használt telefonok . . . . .	14
3.3.1. Samsung Galaxy A71 . . . . .	14
3.3.2. Alcatel OneTouch Pop D1 . . . . .	15
<b>4. Belső állapotok és felületterv</b>	<b>16</b>
4.1. A robot belső állapotai . . . . .	16
4.1.1. Kezdőállapot . . . . .	16
4.1.2. Az irányítás folyamata . . . . .	17
4.1.3. Automata követés folyamata . . . . .	18

4.1.4. A naplózási folyamat . . . . .	19
4.2. Android képernyőtervezek . . . . .	19
4.2.1. A kezdtőképernyő . . . . .	19
4.2.2. Súgó képernyő . . . . .	20
4.2.3. Az automata követés képernyője . . . . .	21
4.2.4. Bluetooth irányítás képernyője . . . . .	21
4.2.5. Naplózási képernyő . . . . .	22
<b>5. A robot és az applikáció működése</b>	<b>24</b>
5.1. A telefon és a robot összekapcsolása . . . . .	24
5.2. A fő folyamatok eldöntése . . . . .	26
5.3. Kód küldése a telefonról . . . . .	26
5.4. Motorok kezelése . . . . .	27
5.5. Akadályok érzékelése . . . . .	28
5.6. A pályát jelölő vonal követése . . . . .	29
5.7. Akadály kikerülése . . . . .	29
5.8. A naplófájl feldolgozása és mentése . . . . .	31
<b>6. Fejlesztés közbeni tapasztalatok</b>	<b>33</b>
6.1. Hardverrel kapcsolatos tapasztalatok . . . . .	33
6.1.1. A robot építésekor szerzett tapasztalatok . . . . .	33
6.1.2. A telefon hardverére vonatkozó tapasztalatok . . . . .	35
6.2. Szoftveres észrevételek . . . . .	36
6.2.1. A késleltetéssel kapcsolatos probléma megoldása . . . . .	36
6.2.2. Naplófájl fogadásának megvalósítása . . . . .	37
<b>7. Továbbfejlesztési lehetőségek</b>	<b>38</b>
7.1. Hardverre vonatkozó fejlesztés . . . . .	38
7.2. Android szoftverre vonatkozó fejlesztés . . . . .	38
7.3. A hardver biztosítására vonatkozó fejlesztés . . . . .	39
<b>8. Felhasználói útmutató</b>	<b>40</b>
8.1. A robothoz tartozó feladatok . . . . .	40
8.1.1. Szükséges alkatrészek . . . . .	40
8.1.2. A motorok csatlakoztatása . . . . .	41
8.1.3. Szenzorok felhelyezése . . . . .	41
8.1.4. Akkumulátor csatlakoztatása . . . . .	42
8.1.5. A program telepítése az alaplapra . . . . .	42
8.2. A telefonon elvégzendő lépések . . . . .	42
8.2.1. Az alkalmazás telepítése . . . . .	43

8.2.2. Bluetooth párosítás . . . . .	43
8.2.3. Az alkalmazás első indítása . . . . .	43

# 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.

Programnyelvként 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 az Arduino IDE és Android Studio nevű alkalmazásokat használtam. Ezen alkalmazásokról a 7. és a 10. oldalakon részletesebben olvashatnak.

Jelen dokumentumban kitérek a robot megépítéséhez felhasznált hardverekre, a program megírása során használt fejlesztői környezetekre, valamint programozási nyelvekre, az egyes komponensek/funkciók feladataira, a fejlesztés során szerzett személyes tapasztalataimra, végül egy egyszerű összeszerelési/telepítési útmutatót is csatolok.

# 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 egy szabad szoftveres, nyílt forráskódú elektronikai fejlesztőplatform. Széles tömegek számára elérhető, mivel olcsó, könnyen beszerezhető, egyszerűen programozható és csatlakoztatható más eszközökhöz.

#### 1.1.1. A platform felépítése

A fejlesztői platform az úgynevezett "IDE"-ból 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 vezérelhetünk a segítségével. Az Arduinonak két fő eleme van, ezek a szoftver és a hardver. A szoftvert képezi az IDE, a hardvert pedig a különböző Board-ok és a Shield-ek, melyekről több információt találnak az 1.1.2 fejezetben.

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.

#### 1.1.2. Elérhető alaplapok

Az Arduino Board-ok többféle változatban készülnek, amelyek méretben, a mikrovezérő típusában, a belső memóriában, a be- és kimenetek számában különböznek. Vannak amelyek rendelkeznek beépített Ethernet, Bluetooth, Wi-Fi csatlakozási lehetőséggel. Jelenleg több Board van forgalomban, ezekből néhány:

- Arduino/Genuino UNO
- Arduino/Genuino 101
- Arduino/Genuino Micro

### 1.1.3. Memória és Háttértár

Egy Arduino három memóriatípussal rendelkezik, melyek különböző feladatokat látnak el.

**Flashmemória:** A Flashmemória számít az Arduino-k "fő" memóriájának, ez tárolja a letöltött programot, és kikapcsolás után is megőrzi a tartalmát, azaz a programot elegendő csak egyszer letölteni ide, az minden ismételt bekapcsolás esetén "magától" újraindul. Programozás során nem számolhatunk a teljes memóriával, mivel a letöltőprogram (bootloader) és például a különböző kommunikációk is innen "csípnek le" részeket.

**SRAM:** Az SRAM tárolja a programban definiált belső változókat. Az SRAM - szemben a flash-memóriával - árammentes állapotban nem őrzi meg a tartalmát, ezért minden bekapcsolást követően a program újradefiniálja a változókat és azok az ott meghatározott "alapértelmezett" értékükkel kerülnek az SRAM-ba.

**EEPROM:** Az EEPROM a board-ok nem felejtő változómemóriája. Ez - hasonlóan a Flash-hez - kikapcsolás során is megőrzi a tartalmát, de - és szintúgy, mint a Flash esetén - "csak" 100.000 írásciklusra van hitelesítve, azaz például a ciklikus adatírásra nem igazán alkalmas. Ráadásul egy kicsit lassabb is a kezelése, mint a "normál" SRAM-é. Ezekből a technikai jellemzőkből adódóan az EEPROM az alábbi funkciókra alkalmazható:

- konfiguráció(k) letárolása
- indulási alapbeállítások mentése
- újraindításuktól független számlálók, értékek, gyűjtött értékek mentése

### 1.1.4. Az IDE felépítése és használata

Az Arduino IDE segítségével írt programokat vázlatnak nevezzük. Ezek a vázlatok a szövegszerkesztőben vannak megírva, és az „.ino” kiterjesztéssel kerülnek mentésre. Az IDE teljes felépítése megtekinthető az Arduino hivatalos honlapján[2].

Az üzenetterület mentés és exportálás közben ad visszajelzést, és megjeleníti a hibákat is. A konzol megjeleníti az Arduino IDE szöveges kimenetét, beleértve a teljes hibaüzeneteket és egyéb információkat. Az ablak jobb alsó sarkában megjelenik a beállított alaplap és soros port.

Az eszköztárban található gombokkal ellenőrizhetjük és feltölthetjük a kódot, új vázlatot hozhatunk létre, valamint megnyithatjuk a soros monitort is.

További parancsok találhatók az öt menüben: **Fájl**, **Szerkesztés**, **Vázlat**, **Eszközök**, **Súgó**. A menük kontextusra érzékenyek, ami azt jelenti, hogy csak a jelenleg elvégzett munka szempontjából releváns elemek állnak rendelkezésre.

### 1.1.5. A C++ nyelv

A C++ egy általános célú, magas szintű programozási nyelv. Támogatja a procedurális, az objektumorientált és a generikus programozást, valamint az adatabsztraktiót. 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 általában 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 körlémények 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. Az Android

Az Android egy Linux kernelt használó mobil operációs rendszer, elsősorban érintőképernyős mobil eszközökre (okostelefon, táblagép) tervezve. Fejlesztését az 'Android, Inc.' kezdte el, melyet a Google 2005-ben felvásárolt, majd az Open Handset Alliance folytatta.

A fejlesztők Java vagy Kotlin nyelven írhatnak rá menedzselt kódot, az eszközök Google által fejlesztett Java programkönyvtárakon keresztül vezérelve.

### 1.2.1. Az operációs rendszer története

Az Android 1.0 platform 2008. október 21-én került kiadásra Apache licenc alatt, mely nem nyerte el igazán az átlagfelhasználók tetszését, ennek oka a használhatóság nehézsége, illetve a kinézete volt az oka.

- 2010-ben a Google bemutatta a Nexus referenciakészülék-termékcshaládját, amit külsős cégek gyártanak. A Google összeállt a HTC-vel, és bemutatták a Nexus One mobiltelefont, amin már az Android 2.1 volt elérhető. Még ebben az évben bejelentették az utódot, amit már a Samsung gyártott, a Nexus S-t, ami már az újabb 2.3-as Androidot futtatja.
- 2011 év végén jött az újabb Galaxy Nexus, szintén a Samsung műhelyéből. Ezen már az Android 4.0 Ice Cream Sandwich futott.
- 2012 júniusában megjelent az első Nexus tablet, a Nexus 7, majd ezt követte a Nexus 10. Mindkét tableten az Android 4.1 volt megtalálható.
- 2012 novemberében jelent meg a Nexus 4, amit az LG gyártott, Android 4.2-vel.
- 2013-ban jött az újabb Nexus 5, szintén LG-gyártmány. Ezen az akkor legújabb, Android 4.4 KitKat futott.
- 2014 októberében mutatták be a jelenleg legújabb Nexus-t, a Nexus 6-ot (Motorola) és a Nexus 9-et (HTC), amikben az Android 5.0 Lollipop található
- 2015 szeptemberében 2 új modell mutatkozott be: az LG által gyártott Nexus 5X és a Huawei által gyártott Nexus 6P, amelyek Android 6.0 Marshmallow-al érkeztek. Ezek után már nem mutattak be újabb Nexus telefonokat, helyüket a Pixel telefonok vették át.
- 2016 októberében jelentette be a Google a Pixel és Pixel XL telefonokat, amelyeket a HTC gyártott.
- 2017-ben szintén októberben mutatta be a Google a Pixel 2-t, amit újra a HTC gyártott és a Pixel 2 XL-t, amelyet az LG készített.
- 2018-ban az eddigiek alapján újra októberben mutatta be a Google a Pixel 3-at és a Pixel 3 XL-t.

## **1.2.2. Az Android Studio felépítése és használata**

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

Az Android Studio első korai szakaszának elérhetősége a 0.1-gyel kezdődött 2013. májusában, majd belépett a béta szakaszba a 2014. júniusában kiadott 0.8 verzióval. Az stabil verziók a 2014. decemberében az 1.0-s verziótól számíthatók.

A JetBrains IntelliJ IDEAján alapuló szoftvert, az Android Studiot kifejezetten androidos fejlesztésre terveztek. Letölthető Windowsra, Mac OS Xre 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.

Az új funkciók folyamatosan jelennek meg az Android Studio új kiadásaiban. minden verzió lehetőséget ad arra, hogy a fejlesztett programot egy virtuális okostelefon segítségével használjuk a számítógépen vagy a csatlakoztatott telefonra telepítsük.

## **1.2.3. A Java nyelv**

A Java 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 forráskóból. 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++ nyelvektől örökölte, viszont sokkal egyszerűbb objektummodellel rendelkezik, mint a C++. A JavaScript szintaxisa és neve hasonló ugyan a Java-hoz, de a két nyelv nem áll olyan szoros rokonságban, mint azt ezekből a hasonlóságokból gondolhatnánk.

Bár a nyelv neve kezdetben Oak<sup>1</sup> volt, később kiderült, hogy ilyen elnevezésű nyelv már létezik, ezért végül Java néven vált ismertté. A Java szó a Oracle védjegye. Ennél fogva engedélye nélkül nem használható mások által kifejlesztett termékek megjelölésére; még például Java-szerű... stb. összetételben sem, mert ez a védjegy jogosult jogaiba ütközik.

<sup>1</sup> James Gosling, a nyelv atya nevezte így az irodája előtt növő tölgfáról

## **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.

Mivel 2020-as Nemzeti Alaptantervben 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 középiskolásoknak 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 felhasználási lehetőségeit mutatom, valamint a 2. fejezetben a 12. oldalon megfogalmazottaknak feleltetem meg a termékemet, majd bemutatom a fejlesztés során használt készüléket.

### 3.1. Használati esetek

A projektnek két felhasználója van, a robot és egy személy. A két felhasználó két különböző szoftverrel érintkezik. Mindkét felhasználó más funkciókat tud elérni a különböző szoftvereknek köszönhetően.

A robot képes a naplófájlhoz új eseményeket hozzáadni, valamint képes a felhasználótól függetlenül egy pályát követni.

A személy el tudja végezni a robot irányítását, a pályakövetési folyamatkor kényszerítheti a robotot a folyamat szüneteltetésére, valamint a robot által készített naplófájl tartalmát lekérheti, mentheti és törölheti.

### 3.2. Követelmény megfeleltetés

A következőkben részletezem, hogy a követelményspecifikációban 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 igyekszem 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 középiskolások számára kellő érdekességgel bír, ugyanis nem csak új ismeretet tudnak vele szerezni, hanem egy interaktív óra lehetőségét is magában hordozza.

Amennyiben két robotot biztosítunk az órán két diádot kiválasztva egy robotok közötti 'verseny' is szimulálható, ahol a diákok képessége határozza meg, hogy melyik robot lesz a gyorsabb, továbbá szemléltetni tudjuk vele, hogy az egyes mobilkommunikációt alapuló eszközök együttműködése nem csak az eszközöktől függ, hanem az eszközök kezelő felhasználóktól is.

### **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 az előzőekben említett több robot közötti 'verseny'.

## **3.3. A fejlesztés során használt telefonok**

A teszteléshez több telefont is használtam, hogy az applikáció erőforrás igényét pontosabban fel tudjam mérni.

### **3.3.1. Samsung Galaxy A71**

- Processzor: Qualcomm Snapdragon 730
- Processzormagok száma: 8
- Processzor sebessége: 2.2 GHz
- RAM mérete: 6 GB
- Bluetooth: v5.00
- Akkumulátor típusa: Li-Pol
- Akkumulátor kapacitás: 4500 mAh
- Android verzió: 10

### **3.3.2. Alcatel OneTouch Pop D1**

- Processzor: ARM Cortex-A7
- Processzormagok száma: 2
- Processzor sebessége: 1 GHz
- RAM mérete: 512 MB
- Bluetooth: v4.00
- Akkumulátor típusa: Li-Ion
- Akkumulátor kapacitás: 1300 mAh
- Android verzió: 4.4.2 KitKat

## 4. fejezet

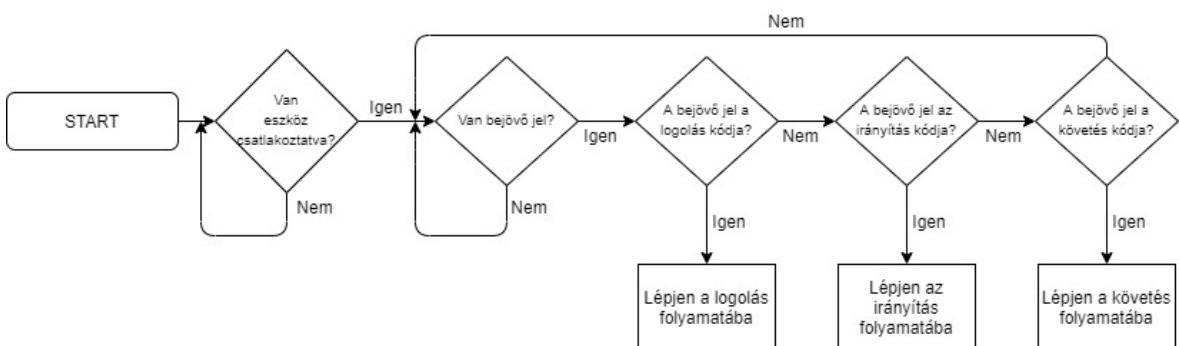
# Belső állapotok és felületterv

„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. Kezdőállapot

A robot indulásakor 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

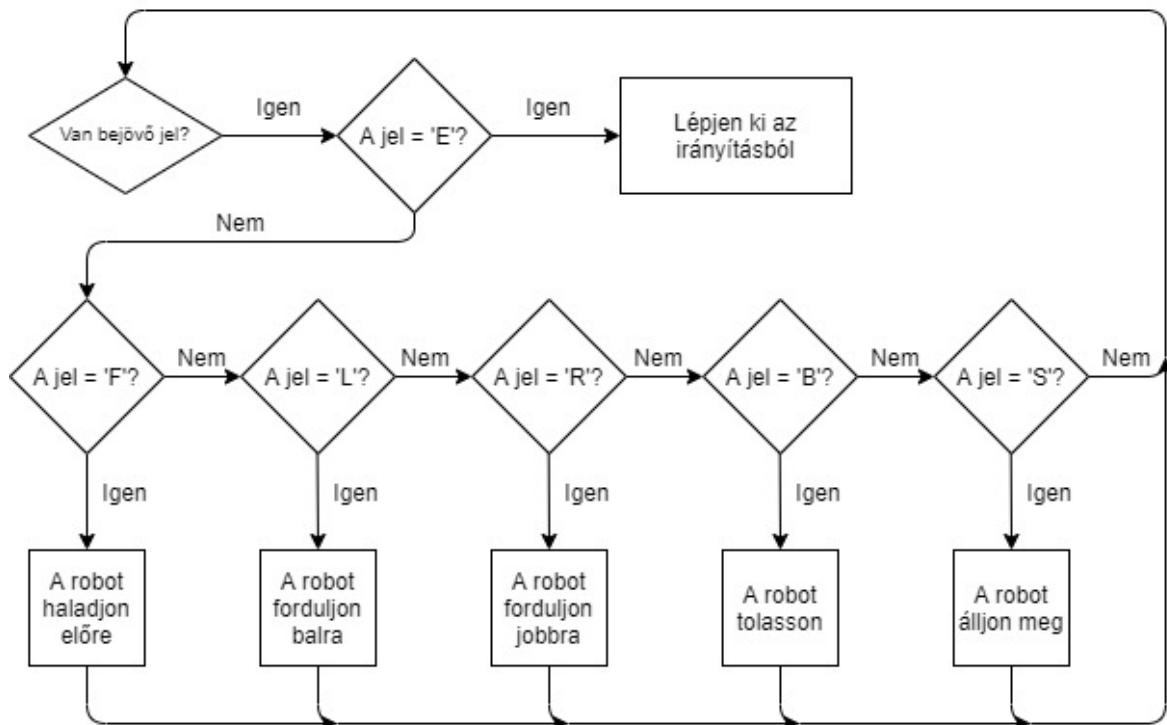
#### 4.1.2. Az irányítás folyamata

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

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.

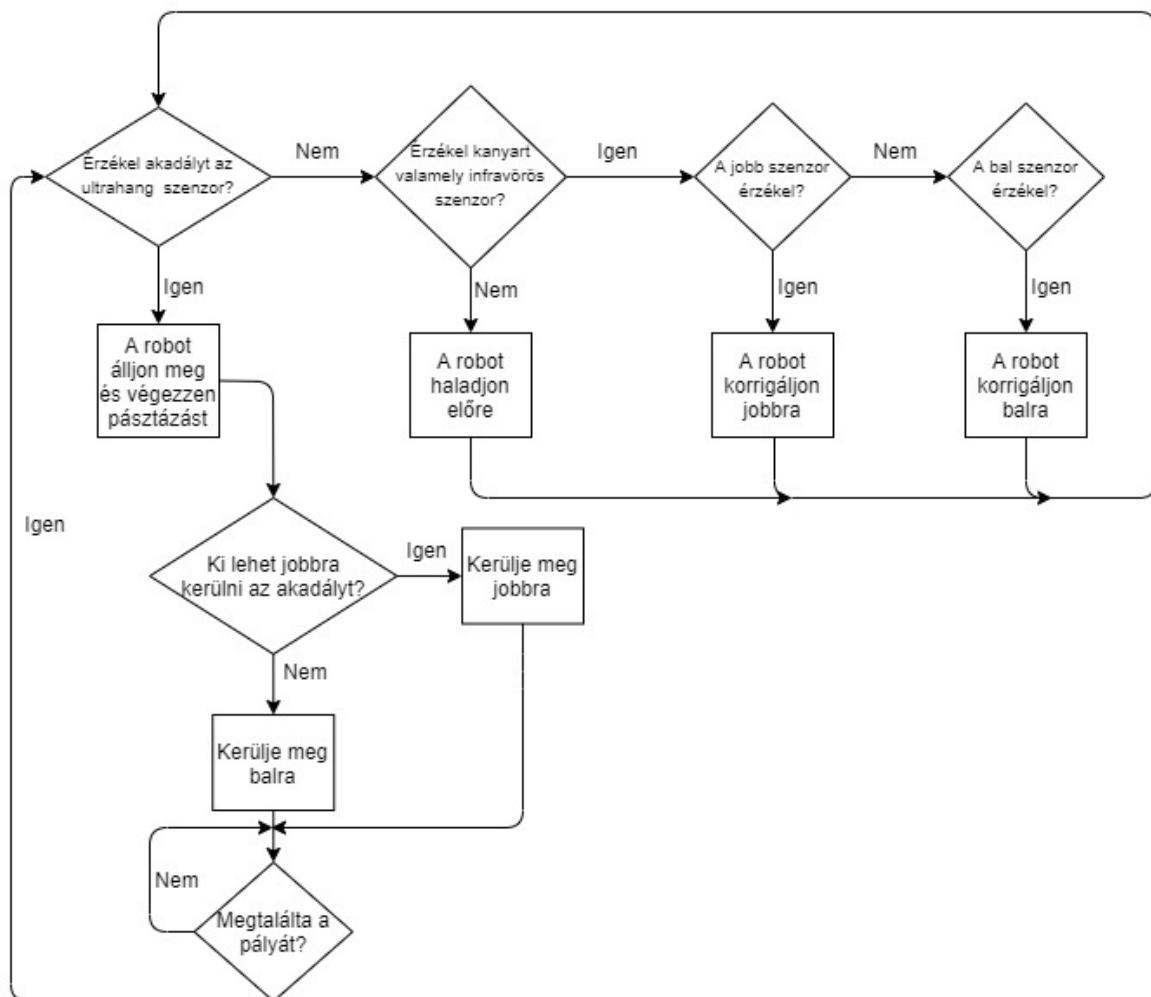


4.2. ábra. Bluetooth irányítás

### 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
- Az 'autó' elején található UH szenzor akadályt érzékel



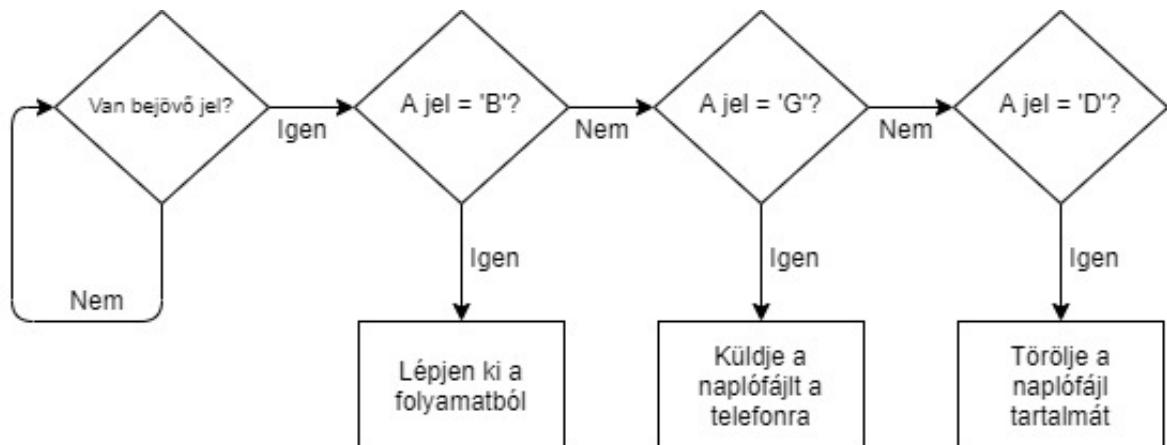
4.3. ábra. A követés folyamata

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.

Amennyiben 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.

#### 4.1.4. A naplózási folyamat

A naplózási folyamat során a robot csak kommunikációs folyamatokat hajt végre a Bluetooth modul segítségével.



4.4. ábra. Naplózási állapotok

Amennyiben a telefonról kapott jel 'B' a robot kilép a folyamatból és a 4.1 állapotba lép.

Ha a bejövő jel 'G' a robot elküldi a telefon részére a Naplófájl tartalmát, ha pedig a jel 'D' törli azt.

## 4.2. Android képernyőtervez

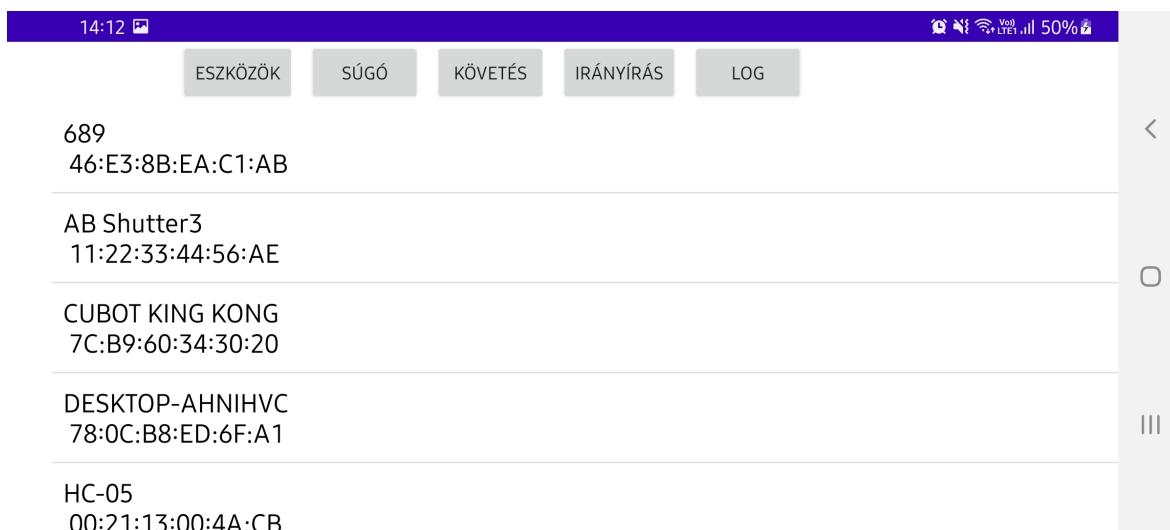
Az applikációnak négy képernyője van.

#### 4.2.1. A 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 képernyón megjelenik a párosított Bluetooth-eszközök listája, melyből ki tudjuk választani a számunkra megfelelőt.
- **Súgó:** Az applikációba beépített 'Súgó' képernyő (4.2.2) válik aktívvá ahol a felhasználó minden funkciót megismerhet.

- **Követés:** Átirányítja a felhasználót az 'Automata követés' képernyőjére (4.2.3).
- **Irányítás:** A felhasználó a 'Bluetooth irányítás' képernyőre kerül (4.2.4), ahol a robotot tudjuk vezérelni.
- **Log:** A naplófájl megtekintésére és mentésére szolgáló képernyőre kerül a felhasználó. (4.9)



4.5. ábra. Kezdőképernyő

## 4.2.2. Súgó képernyő

A képernyő tartalmazza a funkciók rövid leírását.



4.6. ábra. Súgó képernyő

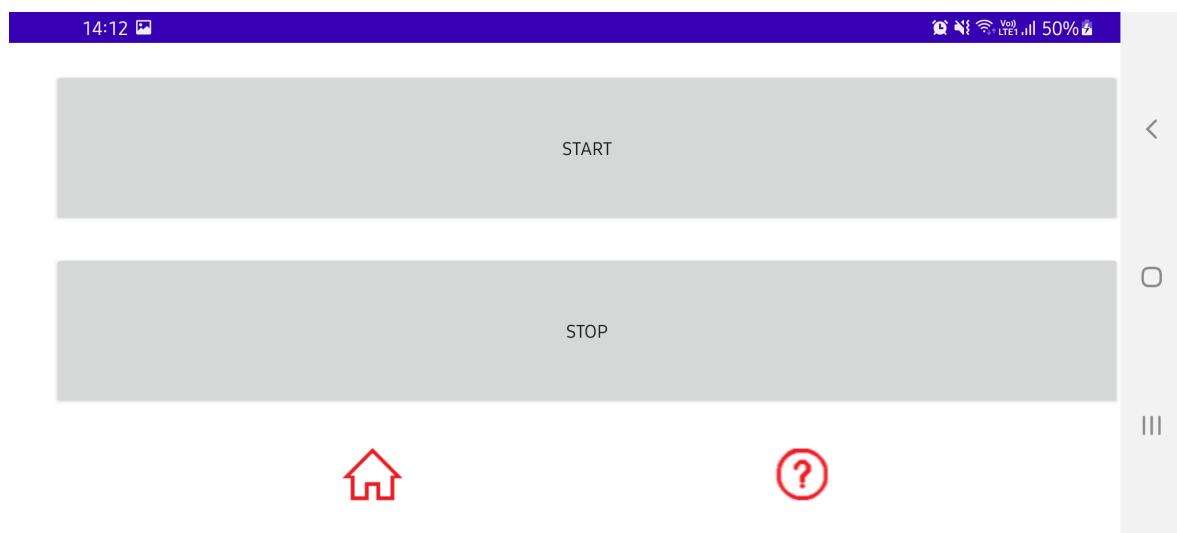
- **Home ikon:** A kezdőlapra navigálja a felhasználót.

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

#### 4.2.3. Az automata követés képernyője

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



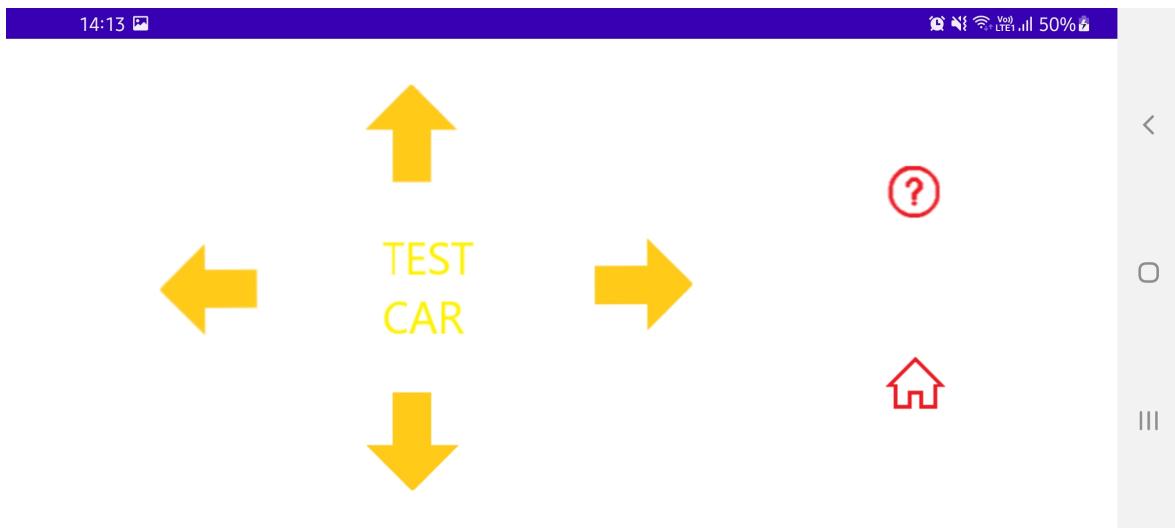
4.7. ábra. Követés képernyő

#### 4.2.4. Bluetooth irányítás képernyője

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.

- **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
- **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.
- **Home:** A kezdőlapra navigálja a felhasználót.
- **Súgó:** A felhasználó a SÚGÓ oldalra kerül.



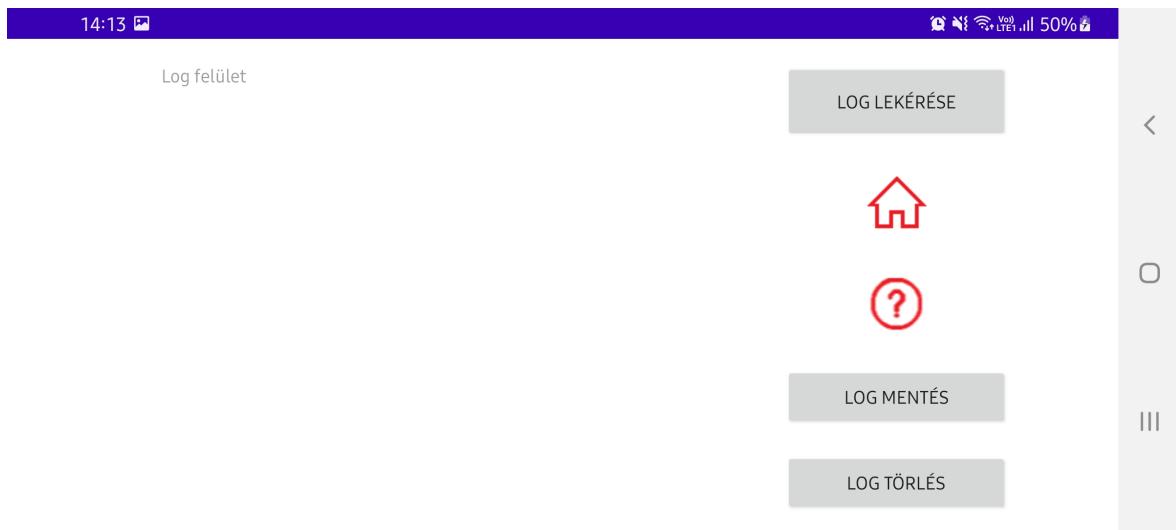
4.8. ábra. Irányítás képernyő

#### 4.2.5. 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 memoriájából 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

*„In some ways, programming is like painting. You start with a blank canvas and certain basic raw materials. You use a combination of science, art, and craft to determine what to do with them.” – Andrew Hunt*

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);
3
4 void setup() {
5     myBluetooth.begin(9600);
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
8             if (mBTAdapter == null) {
9                 Toast.makeText(getApplicationContext(), "A
10                    ↪ bluetooth nincs bekapcsolva", Toast
11                    ↪ .LENGTH_SHORT).show();
12             } else if (!mBTAdapter.isEnabled()) {
13                 Intent enableBT = new Intent(
14                     ↪ BluetoothAdapter.
15                     ↪ ACTION_REQUEST_ENABLE); //Meghívjuk
16                     ↪ a bekapcsolási kérelmet
17                 startActivityForResult(enableBT,
18                     ↪ BT_ENABLE_REQUEST);
19             } else {
20                 new SearchDevices().execute();
21             }
22         }
23     });
24
25     follow.setOnClickListener(new View.OnClickListener() {
26         @Override
27         public void onClick(View arg0) {
28             BluetoothDevice device = ((MyAdapter) (
29                 ↪ listView.getAdapter())).getSelectedItem
30                 ↪ (); //Eltároljuk a listából kiválasztott
31                 ↪ eszközt
32             Intent intent = new Intent(
33                 ↪ getApplicationContext(),
34                 ↪ AutoFollowActivity.class);
35             intent.putExtra(DEVICE_EXTRA, device);
36
37             intent.putExtra(DEVICE_UUID, mDeviceUUID.
38                 ↪ toString());
39             intent.putExtra(BUFFER_SIZE, mBufferSize); //
40                 ↪ Az új activity indításakor a választott
41                 ↪ eszköz paramétereit is átadjuk
42             startActivity(intent);
43         }
44     });
45 }
```

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 végbe megy 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

Az alkalmazásban két módon kell kódot küldeni a robot számára.

- Adott gomb lenyomásakor egy alkalommal
- Folyamatosan, míg az iránygombok valamelyikét nyomva tartjuk

A kód egyszeri kiküldését egyszerűen egy 'onClickListener' használatával el lehet érni, ahol a gomb lenyomásakor az mBTSocket névvel példányosított BluetoothSocket 'getOutputStream().write()' metódusának segítségével elküldjük a kódot.

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). Amíg a gomb nyomva van az 'mBTSocket.getOutputStream().write()' metódus segítségével az iránykódot küldjük a robotnak, majd mikor felengedésre kerül a gomb ugyanezen metódussal egyszer elküldjük a motorok leállítására szolgáló 'stopAction()' kódját.

## 5.4. Motorok kezelése

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

### 5.3. kód. Motorok kezelése

```
1 #include <AFMotor.h>
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 villanymotorok példányosításakor meg kell adni, hogy a panelen melyik csatlakozóhoz vannak kötve, valamint a típushoz.

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

Miután a DC motorok példányosítása megtörtént, megtesszük ezt a szervo motorral is. A program 'setup()' részében a 'myservo.attach()' utasítással tudjuk megadni, hogy melyik port felel a szervo 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 DC motort pedig az 5.4 kódban látható módon tudjuk kezelní. – 'FORWARD' helyett használhatunk 'BACKWARD' parancsot, hogy hátra forgassuk a motort, illetve 'RELEASE' parancsot a motorok leállítására.

A kódban található 'rightAction()', valamint 'leftAction()' metódusok meghívásakor nem egy, hanem két értéket kell megadnunk, melyek az előre és a hátra forgó motorok sebességeiért felelnek.

Másrészről a 'forwardAction()' és 'backwardAction()' metódusoknál 1-1, míg a 'stopAction()' metódusban nincs szükség érték megadására, ugyanis utóbbi az összes DC motort leállítja, vagyis a sebesség minden esetben 0 lesz.

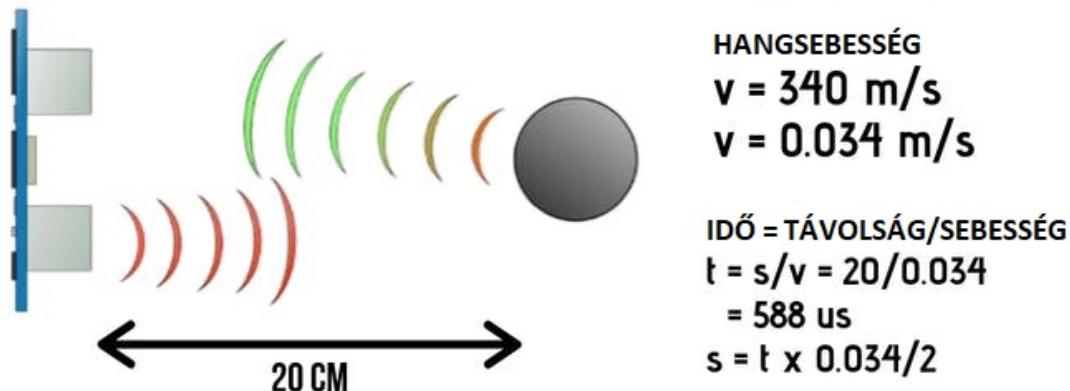
#### 5.4. kód. Előrehaladás

```

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. 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 határozní úgy, hogy a mért időt megsorozzuk 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.

A programban ezt a 12. oldalon megfogalmazottaknak eleget téve a 2.2 kódrészletben látható módon egy metódusba építettem.

<sup>1</sup> centiméter/mikroszekundum

### 5.5. 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

### 5.6. 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 5.6 kódrészleten látható.

A kódban látható 'forwardAction()', 'backwardAction()', 'leftAction()', 'rightAction()' és 'stopAction()' metódusokban megy végbe a motorok irányítása.

## 5.7. Akadály kikerülése

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

Erre az 5.7 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.

### 5.7. 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(); }
```

Először a robot hátról, hogy véletlenül se ütközzön az akadálynak kikerülés közben. Ezután megvizsgáljuk a bal és jobb oldalakat és eltároljuk a szenzorral kapott értékeket, amik alapján el lehet döntenи melyik irányba kell kikerülni az akadályt. Az 'if' és 'else' ágakban található metódusok felelnek ezért.

Az akadály kikerülésekor fontos, hogy a robot el tudja döntenи, 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

A robot működés közben az adott eseményeket egy 'string' típusú változóban tárolja el a 5.8 kód részletben látható módon. Az adat átküldése a telefonra a 'println()' metódussal történik az applikáció pedig a 'getInputStream()' metódussal fogadja ezt és ezek után a while() cikluson belül feldolgozza azt.

A feldolgozás megvalósításának szükségességről részletesebben if olvashatnak a 6.2.2. fejezetben.

5.8. kód. Naplófájl feldolgozása

```
1 — Arduino kód —
2 void logEvent( String event){ logFile += event; }
3 myBluetooth . println(logFile); //Az adat küldése a
4 → telefonra
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 = 0;
12         for ( i = 0; i < buffer . length && buffer [ i ] !=
13             → 0; i++) {}
14         final String strInput = new String (buffer , 0 ,
15             → i );
16         displayedLog = strInput ;
17     }
18     Thread . sleep (500);
19 } //A bejövő adat fogadása
20 displayedLog = displayedLog . replaceAll ("_ , "\n" + ">_
→ "); //A napló formázása és megejelenítése
21 logTextView . setText (displayedLog);
```

Miután az adatot fogadtuk, feldolgoztuk és megjelenítettük a felhasználónak lehetősége van menteni is ezt. A mentést a 4.9 ábrán látható 'Log mentése' gombbal tudja megtenni a felhasználó. A naplófájl neve az adott dátum lesz másodperc pontossággal, amelynek megvalósítását a 5.9 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.

### 5.9. kód. Napló mentése

```
1 SimpleDateFormat sdf = new SimpleDateFormat("yyyy.MM.
    ↪ dd.HH.mm.ss", Locale.getDefault());
2 File_name = "Naplófájl:" + sdf.format(new Date()) + "
    ↪ .txt"; //A naplófájl nevének megadása
3
4 FileOutputStream myStream = null;
5
6 myStream = openFileOutput(File_name, MODE_PRIVATE); //
    ↪ Fájl létrehozása
7 OutputStreamWriter myWriter = new OutputStreamWriter(
    ↪ myStream);
8 myWriter.write(logViewContent); //Fájl tartalmának
    ↪ módosítása
9 myWriter.flush();
10 myWriter.close();
11 msg("Naplómentve:" + File_name + "banéven!", 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 hozzáérhessen a telefon tárhelyéhez. Az engedély megadásához először ellenőriznünk kell, hogy az alkalmazás rendelkezik-e az engedéllyel. Ezt a 5.10 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.

### 5.10. kód. Engedély kérés

```
1 if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M &&
    ↪ checkSelfPermission(Manifest.permission.
    ↪ WRITE_EXTERNAL_STORAGE) != PackageManager.
    ↪ PERMISSION_GRANTED)
2     requestPermissions(new String[]{Manifest.
        ↪ permission.WRITE_EXTERNAL_STORAGE}, 1000);
```

## 6. fejezet

# Fejlesztés közbeni tapasztalatok

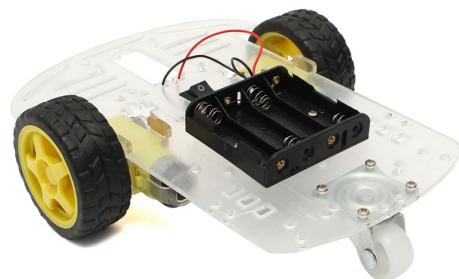
A fejlesztés közben több hardver és szoftverszintű új tapasztalatot is szereztem. Hardver szinten előfordultak problémák is elsősorban a robotnál az egyes elemek rögzítése kapcsán.

### 6.1. Hardverrel kapcsolatos tapasztalatok

Nem csak a robot hardvere jelentett néha problémát, hanem a használt telefoné is. A 6.1.2 fejezetben bővebben olvashatnak 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.

Mivel nincs birtokomban 3D nyomtató mellyel egy olyan alvázat tudnék nyomtatni, mely merevsége és mérete is megfelelő úgy döntöttem, hogy egy plexi lemez fogok használni (másik alternatíva lehet a plexi lemeznél olcsóbb SAN<sup>1</sup> 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.

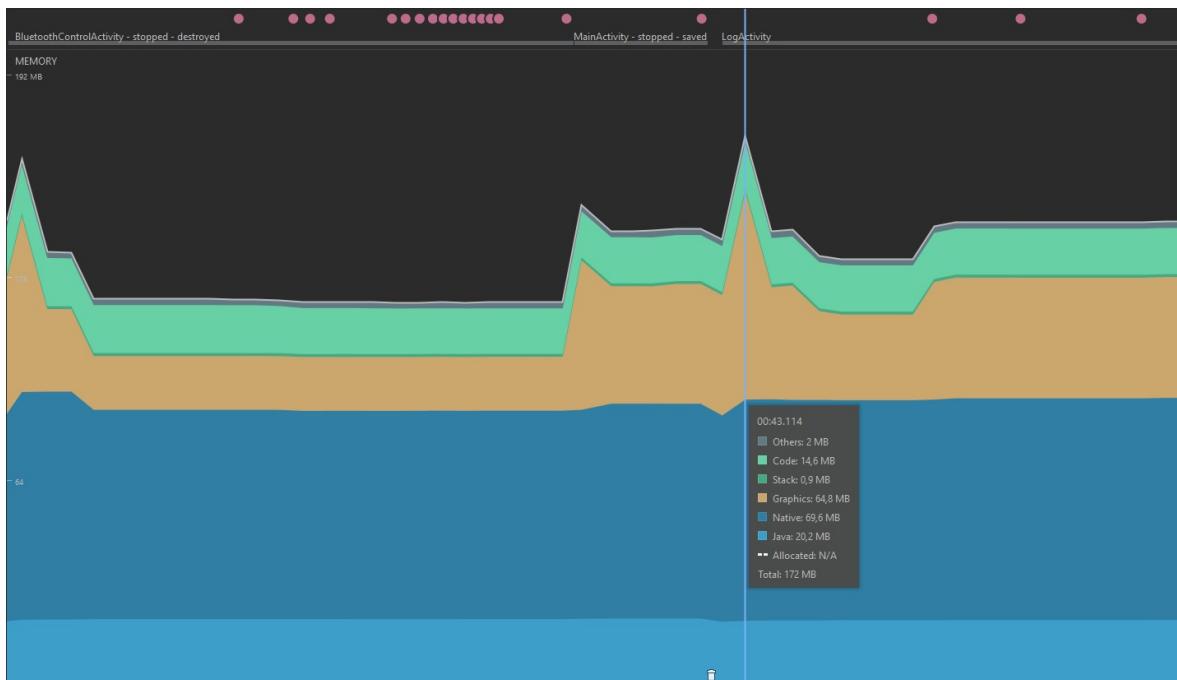
Ennek megoldására az alvázra több lyukat is fúrtam, melyek segítségével a kábelek az alváz alján tudom elvezetni a szenzortól/motortól az alaplapig, továbbá nem használtam hosszabb kábelt, mint amire szükségem volt.

<sup>1</sup> sztirol-akril-nitril

### 6.1.2. A telefon hardverére vonatkozó tapasztalatok

A fejlesztés során a 14 oldalon 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.

A memóriafelhasználás mértéke nem a felhasznált memória mennyiségében, hanem a teljes memóriához viszonyított százalékában emelkedett. Amíg a teszteléshez használt A71 készüléken a program a 6.2 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.



6.2. ábra. A71 memóriafelhasználása

További tapasztalom a telefonban található hardverre nézve a Bluetooth modul típusához köthető. A modernebb A71 támogatja a Bluetooth 5-ös specifikációit, azonban a Pop D1 csak a 4.0-t.

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 észrevételem 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 36. oldalon bővebben olvashatnak, 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 észrevétem 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 észrevétem 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 megoldása 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 használták 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 naplózási funkciót, ezért utánakerestem, hogy egy csatlakoztatott eszközről hogyan tudunk bejövő adatot fogadni[11].

Miután egy olyan kódöt találtam, amelyet értelmezni is tudtam úgy döntöttem, hogy megpróbálom implementálni is, azonban ez is kihívást jelentett. A mintaként talált kódon módosításokat kellett elvégeznem. Első sorban azt terveztem, hogy a naplózási eseményeket a megtörténés után rögtön meg is jelenítem a telefonon, azonban ez az applikáció összeomlását okozta, melynek okára nem tudtam rájönni se kísérletezéssel, se fórumon való kereséssel.

Ezek fényében úgy döntöttem, hogy a napló feldolgozását egy külön képernyőn végzem el ezzel azt akartam biztosítani, hogy a naplózás 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 naplózási 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.

## 7. fejezet

# Továbbfejlesztési lehetőségek

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

### 7.1. Hardverre vonatkozó fejlesztés

Ahogy a 2.3 fejezetben is olvasható a robot jelenleg nem képes az éles kanyarok precíz bevételére. Ennek megoldását nyújthatja, ha a vonal érzékelésére másfajta szenzort használunk (pl.: Digitális Vonalkövető 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 kerülni 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.

### 7.2. 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 autómatá követéskor a kanyarodások közötti időeltérés és a kanyarok szögének ismeretében a pálya vizualizálása is lehetséges válhat.

Továbbá amennyiben a felhasználó nincs birtokában a felhasználói útmutatónak (8. fejezet) a robot összeszerelésekor nem tudhatja, hogy az adott szenzort/motort hova kell csatlakoztatnia. Erre megoldást nyújt az említett dokumentum integrálása az applikáció 'Súgó' felületére.

### **7.3. A hardver biztosítására vonatkozó fejlesztés**

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

Ennek kijavítása lehetséges egy olyan váz megtervezése és 3D nyomtatása, mely védelmet biztosít a hardver elemeinek, illetve így egy olyan kinézetet lehet adni a robotnak, ami ténylegesen egy autó vázára hasonlít.

## 8. fejezet

# Felhasználói útmutató

### 8.1. A robothoz tartozó feladatok

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!

#### 8.1.1. Szükséges alkatrészek

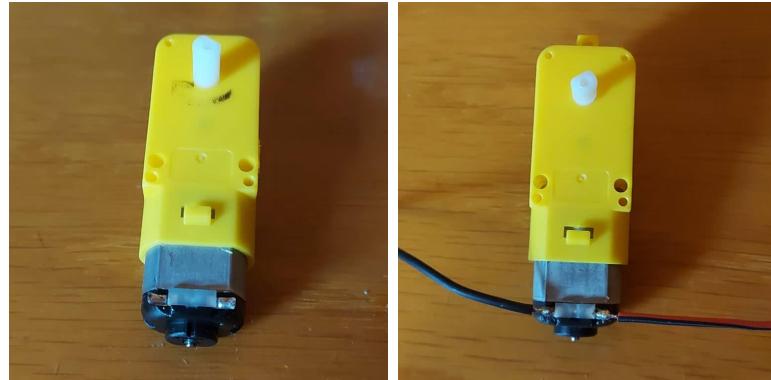
A robothoz szükséges elemek.

Megnevezés	Mennyiség
Arduino Uno mikrokontroller	1db
Alaplappal kompatibilis L293D-t tartalmazó shield	1db
HC-SR04 ultrahang szenzor	1db
HW-201 infravörös szenzor	2db
SG90 szervo motor	1db
Bluetooth modul	1db
Villanymotor	4db
Kapcsolóval ellátott akkumulátor tartó	1db
18650-es akkumulátor	2db
Anya-anya jumper kábel	15db
1mm átmérőjű sodort rézvezeték 12-17 cm hosszú	8db

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

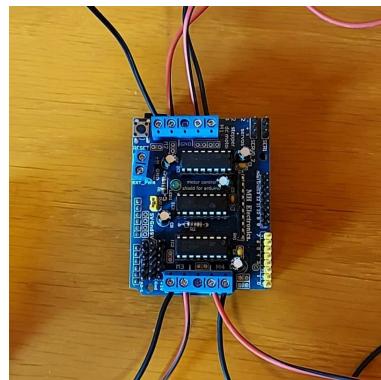
### 8.1.2. A motorok csatlakoztatása

Helyezzünk a 4db 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. (lehetőség szerint a bal oldali csatlakozóhoz fekete, a jobb oldalhoz piros vezeték kerüljön).



8.1. ábra. A kábelek felhelyezése

Ezek után a motorokat felhelyezhetjük az alvázra, majd a vezetékeket a 8.2 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./



8.2. ábra. A shield-re való bekötés

### 8.1.3. Szenzorok felhelyezése

Helyezzük fel az 'autó' elejére a 2db 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

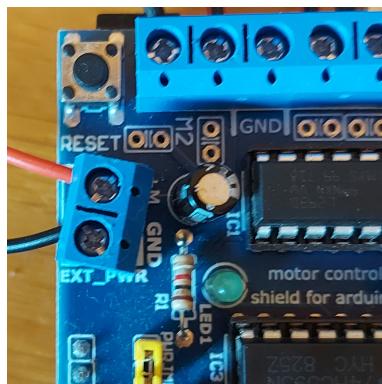
ellátott tüskéit csatlakoztassuk a digitális 0 és 1 portokhoz, valamint GND és VCC tüskéit egy-egy GND és VCC porthoz. A szervó motor vezetékét csatlakoztassuk a shield-en található SERVO\_2 jelzésű tüskesorhoz.

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

#### 8.1.4. Akkumulátor csatlakoztatása

Helyezzük fel és rögzítsük az alvázon az akkumulátor tartót, majd ebbe helyezzük bele az akkumulátorokat, ügyelve a pólusok helyességére.

Amennyiben ez megtörtént a shield-en található +M csatlakozóhoz kerüljön az akkumulátor piros vezetéke, a GND csatlakozóhoz pedig a fekete. (lásd 8.3 ábra)



8.3. ábra. Bekötés

#### 8.1.5. 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. Utolsó lépésként helyezzük el a mikrokontrollert a roboton, majd csatlakoztassuk hozzá a shield-et, ezek után a robot működőképes.

## 8.2. A telefonon elvégzendő lépések

Mielőtt a robotot vezérelni tudnánk el kell végeznünk néhány lépést.

### **8.2.1. Az alkalmazás telepítése**

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

- A számítógépről másoljuk a készülékre az 'app-debug.apk' nevű fájlt
- Engedélyezzük a Beállításokban az ismeretlen forrásból származó alkalmazások telepítését
- Nyissuk meg a telefon fájlkezelő programját, majd keressük ki és futassuk a fájlt.
- A megjelenő üzenetetnél válasszuk a 'Telepítés' lehetőséget
- Az installálás közben felugró 'PlayProtect' üzenetkor válasszuk a 'Telepítés mégis' opciót
- 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

### **8.2.2. Bluetooth párosítás**

Ahhoz, hogy a robothoz csatlakozni tudjunk 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.

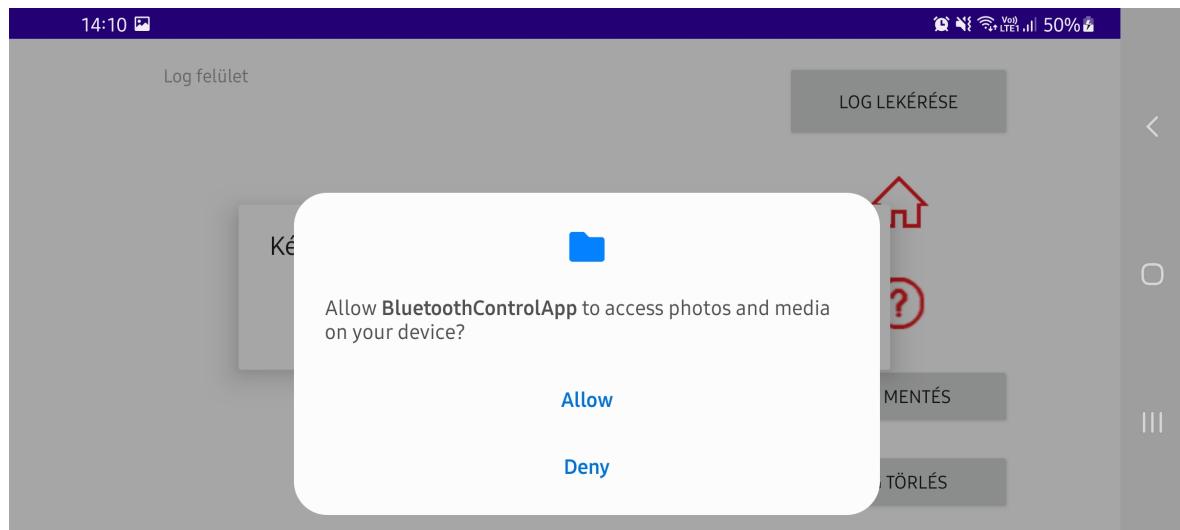
A párosításhoz ki kell választanunk az eszközt és a felugró párbeszédablakban jelzónak az '1234' számsort kell beírnunk. Amint a párosítás megtörtént az applikációt indíthatjuk.

### **8.2.3. 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.



8.4. ábra. Engedélyadás

Ezek után az applikáció minden funkciója használható.

# Irodalomjegyzék

- [1] [HTTPS://HU.WIKIPEDIA.ORG/WIKI/ARDUINO](https://hu.wikipedia.org/wiki/Arduino)
- [2] [HTTPS://WWW.ARDUINO.CC/EN/GUIDE/ENVIRONMENT](https://www.arduino.cc/en/Guide/Environment)
- [3] [HTTPS://HU.WIKIPEDIA.ORG/WIKI/C%2B%2B](https://hu.wikipedia.org/wiki/C%2B%2B)
- [4] [HTTPS://HU.WIKIPEDIA.ORG/WIKI/ANDROID\\_\(OPERÁCIÓS\\_RENSZER\)](https://hu.wikipedia.org/wiki/Android_(operációs_rendszer))
- [5] [HTTPS://HU.WIKIPEDIA.ORG/WIKI/ANDROID\\_STUDIO](https://hu.wikipedia.org/wiki/Android_Studio)
- [6] [HTTPS://HU.WIKIPEDIA.ORG/WIKI/JAVA\\_\(PROGRAMOZÁSI\\_NYELV\)](https://hu.wikipedia.org/wiki/JAVA_(programozási_nyelv))
- [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)
- [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)
- [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)
- [10] [HTTPS://STACKOVERFLOW.COM/QUESTIONS/5171248/PROGRAMMATICALLY-CONNECT-TO-PAIRED-BLUETOOTH-DEVICE](https://stackoverflow.com/questions/5171248/programmatically-connect-to-paired-bluetooth-device)
- [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)