
UNIVERSITATEA „SAPIENTIA” DIN CLUJ-NAPOCA
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE,
TÎRGU-MUREȘ
SPECIALIZAREA CALCULATOARE

Proiectare și implementarea
unui robot de urmărire a unui
obiect cald
PROIECT DE DIPLOMĂ

Coordonator științific:
Conf. dr. ing. Bakó László

Absolvent:
Bakó József

2023

UNIVERSITATEA „SAPIENTIA” din CLUJ-NAPOCA
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș
Specializarea: Calculatoare

Viza facultății:

LUCRARE DE DIPLOMĂ

Coordonator științific:
conf. dr. ing. Bakó László

Candidat: Bakó József
Anul absolvirii: 2023

a) Tema lucrării de licență:

PROIECTARE ȘI IMPLEMENTAREA UNUI ROBOT DE URMĂRIRE A UNUI OBIECT CALD

b) Problemele principale tratate:

- Studiu bibliografic privind sisteme robotice mobile
- Achiziția de date de la senzorii ultrasonici și camera thermo
- Comanda a două motoare servo sau de curent continuu pentru realizarea deplasării robotului mobil
- Software de comandă a sistemului realizat pe microcontroler (de ex. Arduino)
- Comunicație wireless prin Bluetooth cu un smartphone sau tabletă

c) Desene obligatorii:

- Schema bloc a sistemului hardware-software realizat
- Scheme de conectare a componentelor hardware
- Scheme UML ale programelor realizate

d) Softuri obligatorii:

- Software de comandă sistemului robotic
- Aplicație mobilă pe dispozitiv cu Android pentru comanda și monitorizarea sistemului

e) Bibliografia recomandată:

1. Mikroprocesszorok, mikroszámítógép elemek / Madarász László, Kecskeméti Főiskola Műszaki Főiskolai Kar, 1998, 004.3/MAD.
2. A. Tanenbaum: Számítógép architektúrák. (Arhitectura calculatoarelor) Bp., Panem Könyvkiadó, 2004.
3. Gamma, Erich, Helm, Richard, Johnson, Ralph, Vlissides, John, Programtervezési minták, Kiskapu 2004.

f) Termene obligatorii de consultații: săptămânal

g) Locul și durata practicii: Universitatea „Sapientia” din Cluj-Napoca,
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș

Primit tema la data de: 31.03.2022

Termen de predare: 27.06.2023

Semnătura Director Departament

Semnătura coordonatorului

Semnătura responsabilului
programului de studiu

Semnătura candidatului


Declarație

Subsemnata/ulBakó József....., absolvent(ă) al/a specializării ...Calculatoare....., promoția.....2023... cunoscând prevederile Legii Educației Naționale 1/2011 și a Codului de etică și deontologie profesională a Universității Sapientia cu privire la furt intelectual declar pe propria răspundere că prezenta lucrare de licență/proiect de diplomă/disertație se bazează pe activitatea personală, cercetarea/proiectarea este efectuată de mine, informațiile și datele preluate din literatura de specialitate sunt citate în mod corespunzător.

Localitatea, Târgu Mureș

Data: 2023 06 28

Absolvent

Semnătura..........

Proiectarea și implementarea unui robot de urmărire a unui obiect cald

Extras

Măsurarea temperaturii a jucat un rol important dintotdeauna în viața oamenilor, datorită naturii sale curioase. Mulțumită progresului tehnologic și mondial acum este foarte ușor măsurarea temperaturii unui mediu închis, obiect, sau chiar ființe vii.

Scopul proiectului meu este proiectarea și implementarea unui robot și a unei aplicații care identifică temperaturile extreme într-un spațiu închis ținând o distanță constantă față de ei. În aplicație aș afișa o imagine termică, ca să vede utilizatorul acesteia ce vede și robotul. Aici ar fi implementat și o funcție de control manual. Aceste sisteme ar comunica cu bluetooth.

Cuvinte de cheie: măsurare de temperatură, bluetooth, aplicație, control manual

**SAPIENTIA ERDÉLYI MAGYAR
TUDOMÁNYEGYETEM
MAROSVÁSÁRHELYI KAR
SZÁMÍTÁSTECHNIKA SZAK**

**Hőkövető robot tervezése
és megvalósítása
DIPLOMADOLGOZAT**

Témavezető:

Conf. dr. ing. Bakó László

Végzős hallgató:

Bakó József

2023

Kivonat

A hőmérséklet mérése mindig is egy fontos szerepet játszott az ember életében, kíváncsi természete miatt. A technika és a világ fejlődése miatt mára már játszani könnyedséggel mérünk meg egy térben egy átlaghőmérsékletet, vagy akár konkrét eszközök, vagy akár emberek hőmérsékletét.

Jelen dolgozatom célja egy robot megalkotása mely egy zárt térben képes a legmelegebb pont meghatározására és megközelítésére, illetve ezt egy alkalmazáson keresztül a felhasználójának is megjeleníteni egy kép formájában. Ezt a képet a robot bluetooth segítségével juttatja el a kijelzőre. Ezen keresztül a felhasználó szabadon vezérelheti is a robotot.

Kulcsszavak: hőmérséklet mérés, bluetooth, alkalmazás, szabad vezérlés

Abstract

Measuring temperature has always played an important role in human life due to the curiosity of its nature. With the advance of the world and the temperatures measuring technologies, this task has become a trivial problem, regardless of what we measure the temperature of.

The scope of this project is to create a heat follower robot and a mobile application for controlling it, which identifies the temperature extremes in a closed area and approaches it, as well. As for the app, my aim is to establish a Bluetooth connection between this and the robot and show a picture of its view to the user. Also, I want it to make manual control possible.

Keywords: temperature measurement, bluetooth, application, robot control.

Tartalomjegyzék

1. Bevezető	11
2. Elméleti megalapozás és szakirodalmi tanulmány	12
2.1. Szakirodalmi tanulmány	12
2.2. Elméleti alapok	12
2.2.1. Vonalkövetés	12
2.2.2. PID vezérlők	13
2.3. Ismert hasonló alkalmazások	13
2.4. Felhasznált technológiák	14
2.4.1. Felhasznált alkatrészek	14
2.4.2. Bluetooth és L2CAP protokoll	18
2.4.3. PWM	19
2.4.4. I ² C	20
2.4.5. Verziókövetés	22
3. A rendszer specifikációi és architektúrája	22
3.1. Felhasználói felület	22
3.1.1. Felhasználói követelmények	22
3.1.2. Rendszerkövetelmények	23
3.2. Kommunikációs interfész a robot és a kamera között	24
3.3. Kommunikációs interfész a robot és az alkalmazás között	24
3.4. A rendszer architektúrája	24
3.4.1. Thermal app	25
3.4.2. Kommunikáció	27
4. Részletes tervezés	27
4.1. Thermal app	27
4.1.1. Adatküldés	31
4.1.2. Adatfogadás	32
4.2. Robot tervezés	32
4.2.1. Robot építése	32
Kód felépítése	33
5. Üzembehelyezés és kísérleti eredmények	34
5.1. Üzembehelyezési lépések	34
5.1.1. Robot építés	34
5.1.2. Android kódolás	35
5.1.3. Arduino kódolás	37
5.2. Felmerült problémák és megoldásaik	39
5.3. Kísérleti eredmények, mérések	40
6. A rendszer felhasználása	42
7. Következtetések	43
7.1. Megvalósítások	43
7.2. Továbbfejlesztési lehetőségek	44
8. Köszönetnyilvánítás	44
9. Irodalomjegyzék	45
10. Függelék	46

Ábrák jegyzéke

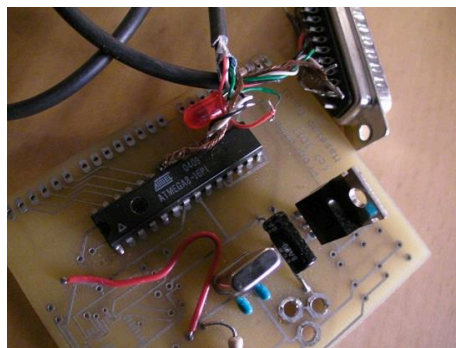
1. Ábra Az első Arduino	11
2. Ábra Arduino Mega 2560	14
3. Ábra HC05 Bluetooth modul	15
4. Ábra AMG8833 hőkamera.....	16
5. Ábra HC-SR04 szenzor	17
6. Ábra L9110S vezérlő egység	17
7. Ábra H-híd áramköri rajz	18
8. Ábra Bluetooth protokoll architektúrája	19
9. Ábra I ² C kommunikáció esetén az adatcsomag szemléltetése	21
10. Ábra Az alkalmazás use-case diagramja.....	23
11. Ábra A teljes rendszer architektúrája	25
12. Ábra UML diagramja az alkalmazásnak	26
13. Ábra Az alkalmazás szekvencia diagrammja	28
14. Ábra Bluetooth nincs bekapcsolva 15. Ábra Alkalmazás főoldal csatlakozás előtt.....	29
16. Ábra Automata mód 17. Ábra Manuális mód	30
18. Ábra A kamera képe.....	31
19. Ábra Modbus: Master-Slave modell esetében az üzenet	31
20. Ábra Automata üzemmód lépései	34
21. Ábra A vezérléshez használt lista.....	36
22. Ábra Kamera kép kérő gomb	36
23. Ábra Hőmérsékleti értékek levágása.....	37
24. Ábra Ultrahangszenzor inicializálás.....	38
25. Ábra Bluetooth modul inicializálása	38
26. Ábra kanyarodásért felelős függvény.....	39
27. Ábra Kamera képe egy emberi kézről 28. Ábra Kamera képe egy pákáról	41
29. Ábra Kamera képe egy szobáról 30. Ábra Kamera képe egy gyújtóról (kevesebb, mint 1 méterről)	42
31. Ábra Kamera képe egy gyújtóról (1 méter határán).....	42

Táblázatok jegyzéke

1. Táblázat Az értékeknek megfelelő mozdulat	32
2. Táblázat Ultrahangszenzor mérési eredmények.....	40
3. Táblázat Forrasztópáka hője kamerával mérve, több pozícióból.....	41

1. Bevezető

Az Arduino története 2003-ban kezdődik, egy olaszországi, Ivrea nevezetű városban, ahol Hernando Barragán, Massimo Banzi és Casey Reas felügyelete alatt létrehozott egy bővíthető integrált áramkört mesterei vizsgájának. Ennek célja egy olyan egyszerű és olcsó eszköz megalkotása, ami nem feltétlen igényel mérnöki tudást kezelési szempontból. A platform maga ekkor még csak egy nyomtatott áramkörtől (ezentúl erre PCB-ként fogok hivatkozni) és egy Atmega128 mikrokontrollerből és egy kezdetleges fejlesztői környezetből állt. Ez a rendszer azonban még Wiring néven létezett. Ebből alakult ki később, 2005-ben az Arduino, ugyanis Banzi két másik kollégájával tovább vitték a Wiring ötletet és a már meglévő mikrokontrollert egy olcsóbb Atmega8ra cserélték. Az 1. Ábra ennek a kezdetleges verzióját szemlélteti.



1. Ábra Az első Arduino

Annak ellenére, hogy az Arduino egyszerű projektek kivitelezésére lett tervezve, mára már elég komplex projektek is születnek, akár a hőmérsékletmérés keretein belül is. Hőmérséklet mérésre régen higanyos hőmérőket használtak, azonban ennek nagy veszélye a nevében is szereplő anyag volt, ami kis golyócskák formájában volt a hőmérőben és ennek törése esetén, amennyiben kézzel lettek felszedve problémákhoz vezettek. Arról nem is beszélve, hogy eszközök hőmérsékletét esélytelen vele megmérni. Ezt később az elektromos hőmérő váltotta fel, ami biztonsági szempontból egy jobb megoldás volt és nemcsak a környezetünk hőmérsékletét tudtuk vele megmérni. Azonban, abban az esetben, ha van egy termem, amiben van 10 számítógép és ezeknek a leadott hőjét szeretném megfigyelni, annak érdekében, hogy van-e köztük hibás darab, minden nap egyesével kéne, kézzel lemérjem és jegyezzem a méréseim. A

gépek számának növelésével pedig a probléma egyre komplexebb lesz és egy ember számára kivitelezhetetlen.

Vagy egy másik probléma, ha van egy hűtőház, amiben szeretném, hogy ne legyen olyan pont, ahol beáramlik a meleg, mert az a portéka megromlásához vezethet. Az előbb említett higanyos hőmérő itt se lesz sok segítségünkre, a digitálissal pedig sokaig tarthat a keresés.

Ezen problémák gondolatmenetén született meg a projektem ötlete. Egy olyan robot megalkotása volt a cél, ami egy zárt térben megkeresi a hőmérsékleti szélső értékeket egy hőkamera segítségével és egy 20 centiméteres távolságra megközelíti azokat, helyváltoztatásuk esetén pedig követi őket. Emellett egy applikáció segítségével Bluetooth-on keresztül elküldi a kamera képét, hogy láthassa a felhasználó a robot nézőpontjából a teret, illetve legyen lehetősége vezérelni is azt.

2. Elméleti megalapozás és szakirodalmi tanulmány

A projektem alapját a pályakövető robotok vezérlése adta kiegészítve Bluetooth kapcsolattal és hőmérsékletméréssel. Ezen projektekben a felhasználói felület gyakran volt egy Androidos applikáció, mivel a felhasználók többsége rendelkezik egy ilyen eszközzel.

2.1. Szakirodalmi tanulmány

A projektem magját a robotvezérlés képezi. Ebben a témában Márton Lőrinc tanár úr könyvét tanulmányoztam [3], hogy megértssem a PID vezérlők működését, amire azért volt szükség, mert egy konstans távolságot szerettem volna a célpontomtól tartani.

2.2. Elméleti alapok

2.2.1. Vonalkövetés

A vonalkövető robotok alapját az infravörös szenzorok alkotják, melyeket egy adott vonal pozícióját követik a robot pozíciója függvényében [1]. Ha a vonal kanyart vesz, a robot leköveti.

Ezekben a rendszerekben, ha kicseréljük az infravörös szenzort egy hőkamerára és az ütközés elkerülése végett behelyezünk pár ultrahangszenzort el is érkezőnk a projektemhez.

2.2.2. PID vezérlők

A vezérlő egységeket két részre tudjuk lebontani: előre csatolt és visszacsatolt vezérlők. Ahogy nevük is mutatja az előre csatolt vezérlők eredménye a következő lépés előrejelzése alapján kapható meg. Ezzel szemben a visszacsatolt vezérlők eredménye a korábbi kimenettől vagy kimenetektől függ.[2]

A proporcionálisan(P) szabályozás esetében „Az aktuálisan mért hiba függvényében számítjuk ki a beavatkozó jelet“ [3]. Ez annyit tesz, hogy a már meglévő hibánkat visszacsatoljuk a bemenetre, ahol is egy állandóval megszorozva (K_p -proporcionális erősítés) megkapjuk a beavatkozó jelet.

$$U(t) = K_p \cdot e(t), K_p > 0 \text{ [4]}$$

$$e(t) = r(t) - y(t) \text{ [5]}$$

Az alábbi képletekben $u(t)$ a beavatkozó jel idő függvényében, $e(t)$ a hiba idő függvényében, $r(t)$ az alapjel idő függvényében, illetve a K_p a proporcionális erősítés.

A PID szabályozás is ezen a sémán működik, szintén jelen van benne a K_p , viszont pluszba bejön egy integrátor és egy derivátor tag(innen jön a PID név). Sokkal pontosabban meg tudjuk kapni a kimenetet. Ekkor a beavatkozó jelet a következőképpen kapjuk meg:

$$u(t) = K_p \cdot e(t) + \int_0^t K_i \cdot e(t) dt + K_d \cdot de(t)/dt \text{ [6]},$$

ahol K_i az integrátor konstans, K_d pedig a derivátor konstans. Fontos, hogy ezeket a konstansokat mi választjuk meg.

2.3. Ismert hasonló alkalmazások

Arduino Uno alapú vonalkövető robot PID kontrollerral [7]

Ennek a projektnek a lényege, hogy a robot egy fekete, vagy fehér vonalat követ egy fekete, vagy fehér talajon, értelemszerűen a vonal és a talaj színe nem egyezik meg. Ennek a projektnek az alkotója azonban egy fényvisszaverő szenzort használt a projektjében. Ellenben a motorvezérlésre használt PID kontrollere segítségemre volt ezeknek tanulmányozása során.

Vonalkövető robot és PID vezérlővel történő akadálykerülés [8]

Az előző projekthez hasonlóan ennek a projektnek a készítői is egy vonalkövető robotot valósítottak meg, ám ők kibővítették egy ultrahang szenzorral, ami akadály estén leállította a robotot.

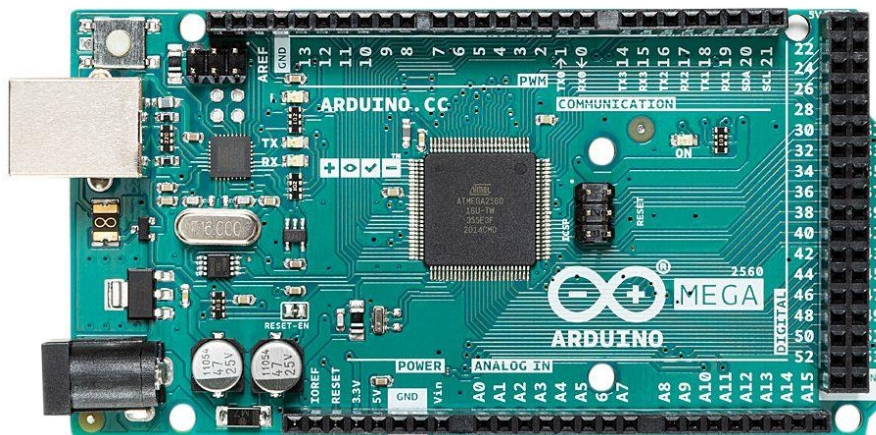
2.4. Felhasznált technológiák

A projektem több szenzor olvasását, illetve egy folytonos kommunikáció fenttartását igényli. Emellett a verziókövetés fontos szerepet kap, hiszen esetleges probléma esetén nem lenne kellemes, ha a munkám elveszne, illetve egy hibás fejlesztői irány kijavításában is jól fog.

2.4.1. Felhasznált alkatrészek

Alkatrészek terén szükségem volt egy Arduinora, ami az egész rendszert vezérli, egy Bluetooth modulra a kommunikációhoz, egy hőkamerára, pár ultrahangszenzorra távolságtartás és akadálykerülés végett, két motorra a mozgatus miatt és ezeknek egy vezérlő egységre.

Arduino terén a szenzorok sokasága miatt egy Arduino Mega 2560-ra esett a választás(2. Ábra). Ez egy Atmega2560 mikrovezérlőn alapuló bővíthető lap, 54 digitális kimenettel, amiből 15 használható PWM kimenetnek, 16 analóg bemenetnek, 4 pedig hardware soros portnak. Memória terén 256 KB Flash memóriával rendelkezik, 8 KB SRAM-mal és 4 KB EEPROM-mal. Több kommunikációs interfészt támogat, köztük a hardveres soros portot, valamint az I2C és SPI protokollokat is [9].



2. Ábra Arduino Mega 2560

Bluetooth modul terén a választás a HC-05 modulra esett, ezt a 2. Ábra szemlélteti. Ez egy gyakran használt Arduino és más mikrokontroller platformokkal is kompatibilis Bluetooth kommunikációs modul. Kényelmes módot biztosít az Arduino és egyéb eszközök, mint például

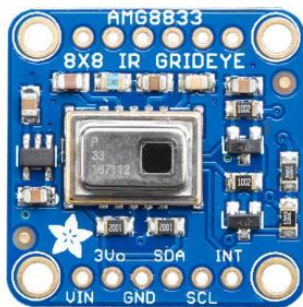
okostelefonok, táblagépek, vagy egyéb Bluetooth eszközök közötti vezeték nélküli kommunikáció létrehozására. A modul Bluetooth 2.0 + EDR(Enhanced Data Rate) specifikációin alapszik. Támogatja a soros port protokollt(SPP), mely lehetővé teszi a soros kommunikációs interfész emulálását Bluetooth-on keresztül. Működését tekintve két üzemmódban működhet: mester és szolga módban. Mester módban más Bluetooth eszközökkel hozhat létre kapcsolatot, míg szolga módban más eszközök kapcsolataira vár. Az eszköz beépített soros-Bluetooth modullal rendelkezik, mely lehetővé teszi, hogy soros kommunikációs interface(általában UART) segítségével kapcsolódjon az Arduino-hoz. Két pinnel rendelkezik a soros kommunikációhoz: RXD(Receive Data) és TXD(Transmit Data). Ezek a pinok az Arduino megfelelő pinjeihez jönnek csatlakoztatva. Ezt az eszközt gyakran hasonlítják össze egyéb Bluetooth modulokkal, mint például a HC-06, vagy a HM-10. Az első nagyon hasonló ehhez az eszközhöz, azonban nem rendelkezik AT-parancs konfigurációs funkcióval, így kevésbé konfigurálható. A másik eszközhöz képest pedig kezelés terén előnyösebb, ugyanis a HC-05 sokkal egyszerűbben integrálható projektekbe és több mikrokontrollerrel is kompatibilis[10].



3. Ábra HC05 Bluetooth modul

Hőkamera terén egy AMG8833as hőkamerára esett a választás. Ez a Panasonic által kifejlesztett hőérzékelő egység. Az érzékelő 8X8, azaz 64 képpontból álló tömbből tevődik össze. Minden egyes pixel egy adott terület hőmérsékletét méri. Ebből következik, hogy a kép amit az érzékelő alkot is 8X8 pixelből fog állni. Látószöge fixált és 60 fokos. Általában -20°C és 80°C közötti tartományban működik, azonban vannak típusok, amik nagyobb hőmérsékleti értékek mérésére is képesek. Ezen mért értékeket I²C protokollt használva kommunikálja le a

mester eszköznek a megfelelő parancsra. Előnye a többi hőkamerával szemben, hogy energiatakarékos(45 mA fogyasztás), tervezése lévén, ami az én projektben egy kulcsfontosságú aspektus. Emellett mérete, software szintű támogatottsága és elég nagy mérési tartománya is előnybe hozta versenytársaival szemben[11].



4. Ábra AMG8833 hőkamera

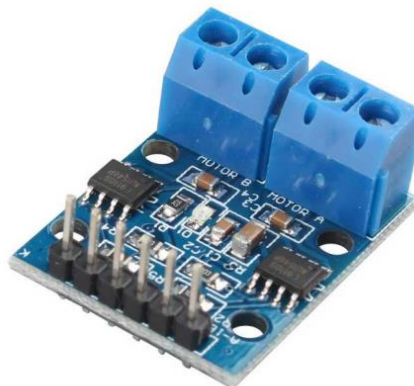
Ultrahangszenzor terén 3 darabra volt szükségem, az egyik a távolságtartás miatt, a másik kettő pedig az akadálykerülés miatt. Modell terén a HCSR04 szenzort választottam. Ez ultrahanghullámok visszhangideje alapján működik. Ultrahangos hanghullámokat bocsát ki és azt méri, hogy a hullámok mennyi idő alatt pattannak vissza, miután egy akadályba ütköznek. Az ezalatt eltelt idő alapján ki lehet számolni a távolságot az akadály és az érzékelő között. Jellemzően 2 cm és 400 cm közötti távolságokat képes mérni, viszont a tényleges hatótávolság függ a környezettől és akár a konkrét megvalósítástól is. A mikrovezérlővel két pin segítségével keresztül kommunikál: a trigger és az echo. A trigger feladata az ultrahangos impulzus kibocsátása, míg az echo piné ennek felfogása. Nem használ speciális kommunikációs protokollt az adatátvitelhez. Ehelyett a kontroller I/O portjaira támaszkodik a jel feldolgozásához. A többi ultrahangszenzorhoz képest előnye az alacsony energiafogyasztás(15 mA), pontosság, gyors reakcióidő(mindössze néhány mikroszekundum) és nem utolsó sorban elérhető ár és széleskörű támogatottság[12].



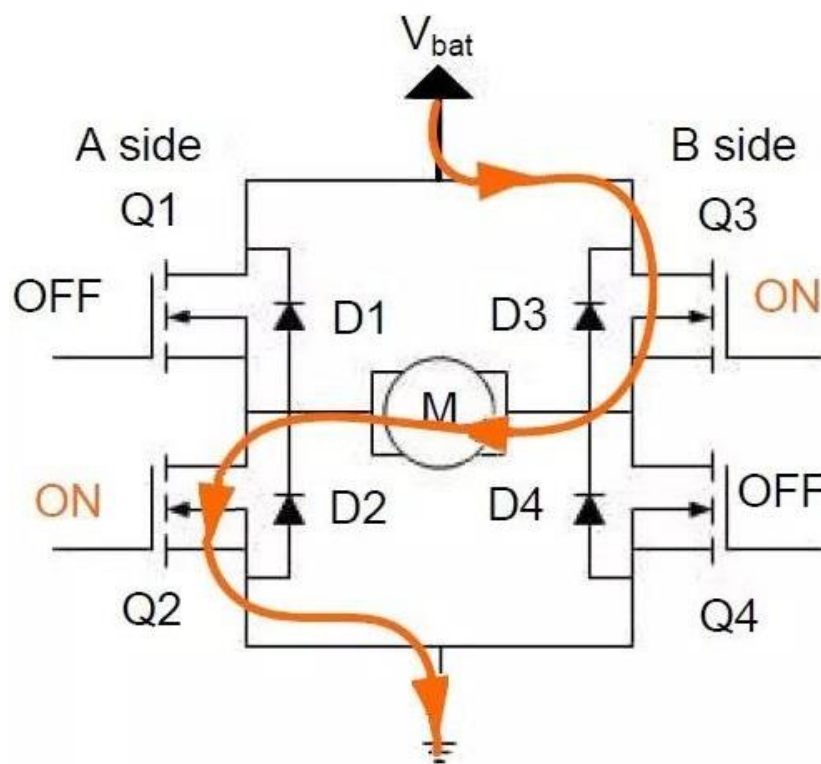
5. Ábra HC-SR04 szenzor

Motorok terén 2 egyenként 12 voltos motort használtam, IG220019X00015R típusúakat, amit a Digilent cég forgalmaz és tartalmaznak 2 beépített HALL effektus szenzort is[13].

A motorok vezérlésére egy HW-016os, vagy L9110S típusú motor vezérlőegységet használtam(5. Ábra). Ez hatékony módot biztosít a kis és viszonylag kis motorok meghajtására. Legtöbb két DC motor meghajtására tervezték, vagy egyetlen léptetőmotor meghajtására. Kis teljesítményű motorokat kezel, melyek feszültsége 2.5 V és 12 V közé esik. Lehetővé teszi ezen eszközök két irányú vezérlését, lehetővé téve így az előre és hátrameneti fordulatot is. Emellett külön vezérlést biztosít minden motorcsatornához, ezáltal lehetővé téve a motorok sebességének és irányának független vezérlését. A vezérlés megvalósításához egy H-híd áramköri konfigurációt valósít meg négy H alakban elhelyezett MOSFET-ből, ahogy azt a 6. Ábra is szemlélteti.



6. Ábra L9110S vezérlő egység



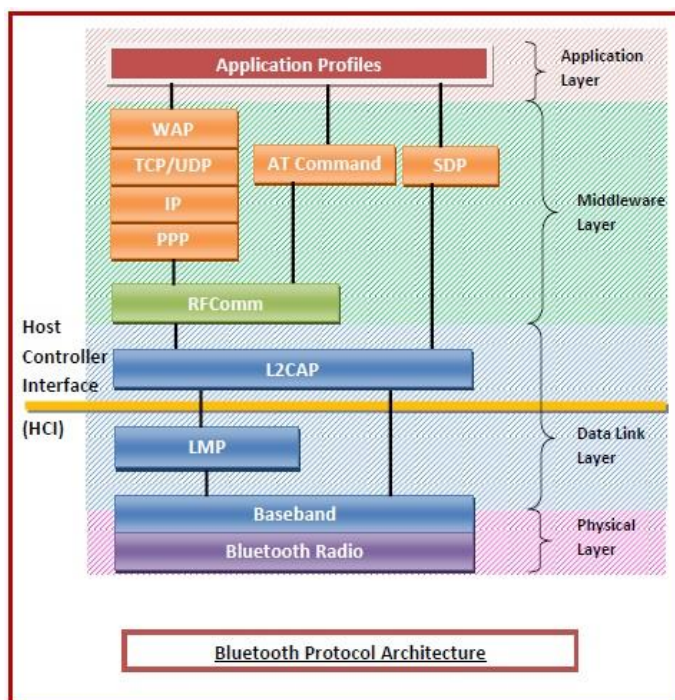
7. Ábra H-híd áramköri rajz

Az L9110S vezérlő egység két üzemmódot kínál: önzáró és zárt üzemmódot. Önzáró üzemmódban a kimenetek egy parancs után megtartják állapotukat és a motorokat pozícióban tartják. Zárt üzemmódban a kimenetek magas impedanciájú állapotba kerülnek, ami hatékonyan leállítja a motorokat. Az eszköz digitális vezérlőjeleket fogad és egyszerű vezérlési logikát alkalmaz[14].

2.4.2. Bluetooth és L2CAP protokoll

Mint már fentebb említettem, a projektben fontos szerepet kap a kommunikáció a felhasználó és a robot között. Ez Bluetooth segítségével valósul meg. Ez egy vezeték nélküli kommunikációs technológia, mely biztosítja az adatcserét rádióhullámok segítségével a 2.4 GHz-es frekvenciasávban két vagy több eszköz között. Ennek a sebessége az eszközök Bluetooth verziójától függ, viszont az eszközök verziószáma meg kell egyezzen, így biztosítva az interoperabilitást és a könnyű használatot. Egyik hatalmas előnye az alacsony energiafogyasztás, ezáltal az eszközök akkumulátorának élettartamát hosszabbítva.

A Bluetooth architektúra saját, független modellel rendelkezik, mely a szabványos OSI-modell, vagy TCP/IP-modell helyett egy protokoll-halmazt tartalmaz. Ezek a protokollok rétegekbe vannak szervezve, melyek mindegyike egy meghatározott célt szolgál a kommunikációs folyamat során. Az adata vitelért felelős protokoll a Logical Link Control and Adaptation Protocol (L2CAP) és a Data Link rétegben található, ahogy azt az 5. Ábra is szemlélteti. Ez kezeli az adatok csomagolását és feldarabolását kisebb, úgynevezett L2CAP csomagokra és ezek kerülnek átadásra.



8. Ábra Bluetooth protokoll architektúrája[15]

2.4.3. PWM

A későbbiekben sokszor fogom a PWM mozaikszót használni, ami a Pulse Width Modulation rövidítése, magyarul impulzus szélesség modulációt jelent. Ez az elektronikában és a digitális rendszerekben általánosan használt technika az analóg jelek vezérlésére egy impulzus hullám aktív ciklusának változtatásával. Működési elve, hogy egy digitális jelet gyorsan váltogat HIGH és LOW állapot között. Az időarányt, amelyben a jel HIGH állapotban van a teljes periódushoz viszonyítva nevezzük aktív ciklusnak. Ennek változtatásával a jel átlagértéke állítható, ezáltal hatékonyan irányítható a kimeneti feszültség, vagy áramerősség.

A PWM jeleket jellemzően időzítőkkal, vagy dedikált PWM modulok segítségével generálják. Ezek az eszközök digitális impulzusok sorozatát állítják elő rögzített frekvencián, ahol az egyes impulzusok időtartama megfelel a kívánt aktív ciklusnak.

Ezen módszer számos előnyt nyújt a hagyományos analóg vezérlési módszerekkel szemben. Egyrészt sokkal hatékonyabb, mivel a teljesítményveszteségek minimalizálódnak. Ez annak köszönhető, hogy a jel teljesen bekapcsolt és teljesen kikapcsolt állapotok között vált, ami jobb energiahatékonyaságot és csökkentett teljesítmény disszipációt eredményez. Ezzel szemben analóg vezérlés esetén a felesleges teljesítmény hőként disszipálódik ellenállásokban, vagy más komponensekben a kellő áram beállítása érdekében.

A PWM vezérlés egy másik nagy előnye a zaj és interferencia csökkentés lehetősége. Mivel ezek a jelek fix frekvencián működnek, hatékonyabban szűrhetők a nagyfrekvenciás zajkomponensek eltávolítása érdekében.

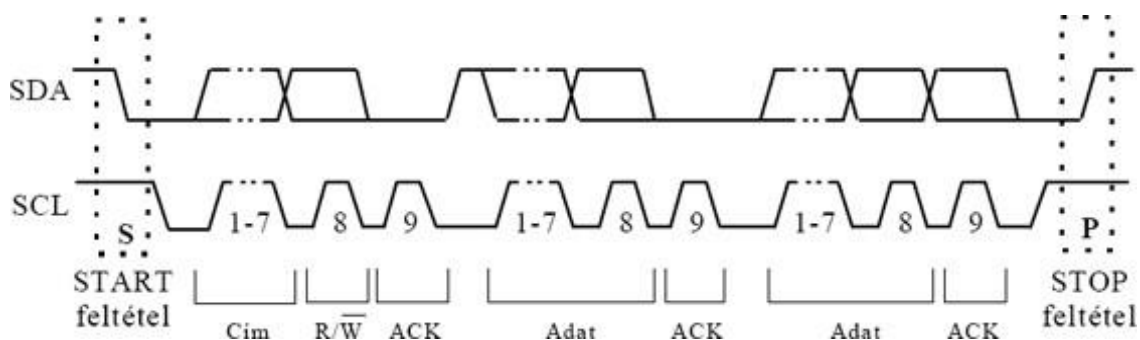
Végül pedig a PWM vezérlés az analóg áramkörök összetettségét, ugyanis ezek sokszor bonyolultak a változó ellenállások, vagy feszültségosztók szükségessége miatt. Ezzel szemben a PWM vezérlés digitális jelek segítségével valósul meg és nem igényel összetett áramköröket, ami egyszerűbb és költséghatékonyabb is.

2.4.4. I²C

Az I²C, vagy Inter-Integrated Circuit széles körben használt soros kommunikációs protokoll, mely lehetővé teszi az integrált áramkörök vagy elektronikai eszközök közti kommunikációt egy közös buszon. Kétvezetékes buszszerkezetet használ, amely egy soros adatvezeték(SDA) és soros órajelvezeték(SCL) áll. Az SDA egy két irányú csatorna, amely az eszközök közötti adatátvitelre és fogadásra használják. Ez hordozza a tényleges információt legyen az parancs, vagy adat. Az SCL egy egyirányú csatorna, amelyet az eszközök közötti adatátvitel szinkronizálására használnak. Ez biztosítja az adatátvitel időzítését és hogy az adatok mintavételezése a megfelelő időpontban történjen. Az egymással kommunikáló eszközök két kategóriába sorolhatóak: mester és szolga eszköz. A mester kezdeményezi és vezérli a kommunikációt a buszon. Ő generálja az órajelet és végzi a szolgál-eszközök címzését, ezáltal több eszköz jelenlétét is támogatva akár a buszon. Az ő parancsaira és kéréseire válaszolnak a szolga eszközök. Utasítástól függően fogadják és továbbítják az adatokat utasítástól függően.

Az adatok továbbításának sebessége a rendszer bitrátájától függ. Ez határozza meg milyen gyorsan jut el az adat a mestertől a szolgákig, vagy fordítva. Ezt az órajelfrekvencia és az egyetlen adatbit továbbításához szükséges órajelciklusok száma határozza meg az alábbi képlet segítségével:

$$\text{bitráta} = \text{órajelfrekvencia} / \text{bittenkénti órajelciklusok száma}$$



9. Ábra I²C kommunikáció esetén az adatcsomag szemléltetése[16]

Ahogy azt a 6.Ábra is szemlélteti I²C kommunikáció esetén az adatok csomagok formájában továbbítódnak. Ezek bitek sorozatából állnak, melyek egy meghatározott struktúrába vannak szervezve. Ez a csomag egy start feltétellel kezdődik. Ez egy speciális szekvencia, mikor az SDA vonal HIGH állapotról LOW állapotba vált, míg az SCL aktív marad. Ezt követi a cím bájtt. Ez 8 darab bitből tevődik össze, melyek közül az első hét a konkrét címet tartalmazza, a maradék egy pedig azt jelzi, hogy a mester írást, vagy olvasást kíván végezni a szolga eszközön. A cím bájtt után a két eszköz a kommunikáció típusától függően(írás vagy olvasás) egy, vagy több adatbájtt cserélnek. Minden adatbájtt 8 bit információból tevődik össze, melyeket 8 órajelciklusonként továbbítanak. Előre kerül a legjelentősebb bit(MSB- most significant bit) és ez lesz leghamarabb továbbítva, majd őt követi a többi 7. Minden egyes adatbájtt fogadása után a fogadó fél egy visszajelzést küld egy bit formájában(ACK- acknowledgment), hogy jelezze a sikeres fogadást. Az ACK bit küldése az SDA vonal LOW állapotba állításával történik a kilencedik órajelciklus alatt. Sikertelen fogadás esetén a SDA vonal HIGH állapotban marad jelezve a sikertelen adatküldést(NACK- non-acknowledgment). Miután az eszközök befejezték a kommunikációt a mester a tranzakció végét jelző stop állapotot generál. Ez egy olyan speciális szekvencia, mikor az SDA vonal LOW állapotról HIGH állapotba vált, míg az SCL vonal HIGH

marad. Ez lehetővé teszi a busz felszabadulását, ezáltal teret engedve más eszközöknek a kommunikációra.

2.4.5. Verziókövetés

Verziókövetésre egy GitHub repositoryt használtam. A GitHub egy online tárhelyszolgáltatás verziókövetéshez, mely Git rendszert használ. Ez a rendszer figyeli, hogy egy repositoryban levő fájlok változtak-e. Különösen hasznosnak bizonyult az esetlegesen felmerülő bugok kijavításában, illetve a korábbi verziók elérésében is[17].

3. A rendszer specifikációi és architektúrája

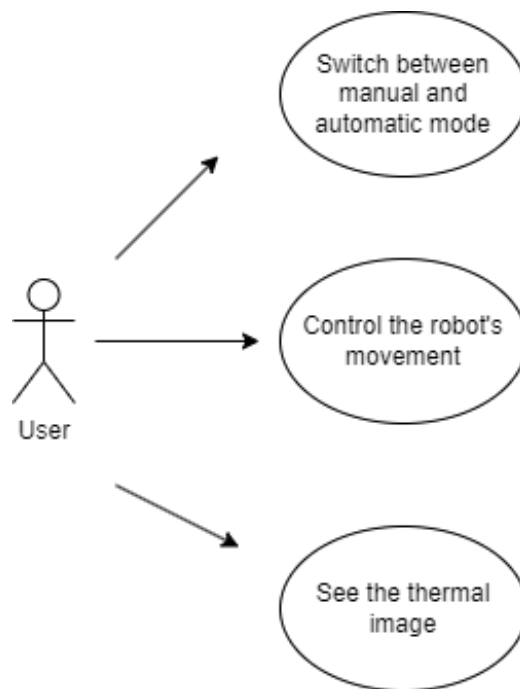
A rendszert specifikáció szempontjából három nagyobb részre lehet osztani: a felhasználói felületre, illetve a kommunikációs interfész a robot és az alkalmazás, illetve a robot és a termál kamera között. A felhasználói felület teszi lehetővé, hogy a felhasználó interakcióba léphessen a robottal. A robot és az alkalmazás közötti kommunikációt a Bluetooth teszi lehetővé, illetve a robot és kamera közti kommunikációt egy buszkapcsolat, az úgynevezett I²C kommunikáció teszi lehetővé. A továbbiakban használt grafikonokat a Draw.IO online szoftver segítségével készítettem [18].

3.1. Felhasználói felület

A felhasználó felület egy Android applikáció formájában valósul meg. Itt a felhasználó eldöntheti, hogy automatizált vagy manuális vezérlést alkalmaz, valamint megnézheti a hőkamera képét.

3.1.1. Felhasználói követelmények

Az applikációt bármely felhasználó használhatja, ha rendelkezésére áll egy okos eszköz, amely rendelkezik Bluetooth integrációval. Amint az az 5. Ábrán is látszik a felhasználó váltani tud a vezérlési lehetőségek között, irányíthatja a robotot, illetve megnézheti a kamera képét.



10. Ábra Az alkalmazás use-case diagramja

3.1.2. Rendszerkövetelmények

- a) Funkcionális követelmények: Az alkalmazás indításakor, amennyiben a Bluetooth nincs bekapcsolva jelez a felhasználónak, hogy kapcsolja ezt be, ugyanis szükséges lesz. Amennyiben be van kapcsolva a felhasználó előtt két gomb jelenik meg a képernyőn. Az egyikkel ki tudja választani, hogy milyen nevű Bluetooth eszközre kíván csatlakozni. Itt a HC-05 nevű eszközt kell kiválasztani, majd a másik gomb segítségével rá tud erre csatlakozni. Amint ezzel megvan egy kapcsoló jelenik meg előtte, ezzel tudja eldönteni, hogy manuális, vagy automatikus módban kívánja működtetni a robotot. Amennyiben a manuális opciót választja megjelenik előtte 4 további gomb, amik az irányításért felelnek. A switch alatt közvetlenül található egy gomb, ami elnavigálja a felhasználót arra képernyőre, ahol a kamera képet nézheti meg. Ezen a képernyőn, vagy egy kör alakú ikon jelzi a felhasználónak, hogy éppen töltődik a kamerakép, vagy maga a kép látható, illetve egy gomb, ami a főoldalra viszi vissza.

b) Nem funkcionális követelmények:

1. Android alapú okos eszköz
2. Android 10.0 +
3. Min API Level: 29
4. Tárhely: 202MB
5. Bluetooth: 5.0, vagy újabb

Az alkalmazást kizárólag Android alapú okos eszközön lehet használni, legalább 10.0-as verziójú eszközön (Min API level: 29). Továbbá az eszköznek rendelkeznie kell legalább egy 5.0-as Bluetooth modullal, illetve 202 MB tárhelyre.

3.2. Kommunikációs interfész a robot és a kamera között

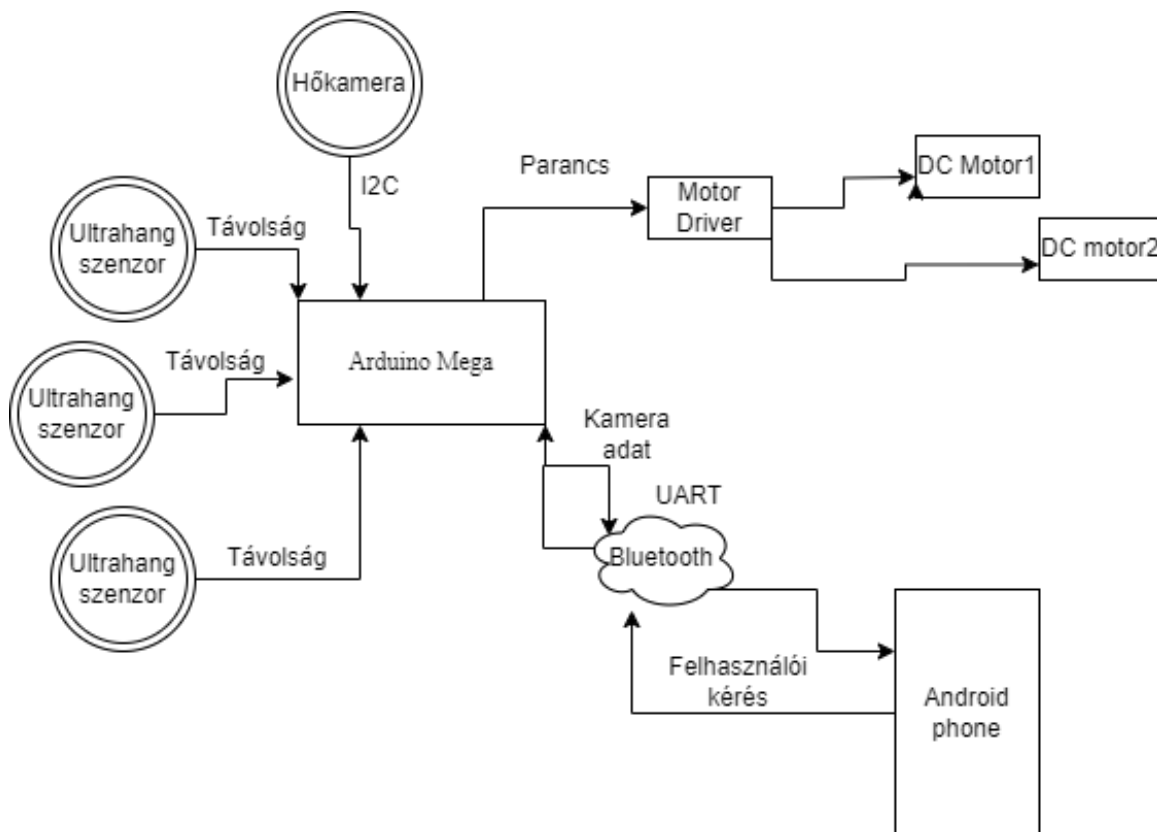
Ez teszi lehetővé, hogy a kamera és a robot kommunikálni tudjanak. Ez lényegében egy I²C kommunikáció, ami mester-szolga szabványt követ és a kamera adatait továbbítja az Arduinonak. A mester szerepet nyilván az Arduino tölti be, a szolga szerepét pedig a kamera.

3.3. Kommunikációs interfész a robot és az alkalmazás között

Ez a Bluetooth kapcsolatot jelenti, ez teszi a kommunikációt lehetővé a robot és az alkalmazás között. Ezen a csatornán keresztül valósul meg az adatküldés is.

3.4. A rendszer architektúrája

A projektben megvalósított rendszer 2 nagyobb részből tevődik össze, melyeket Bluetooth köt össze.



11. Ábra A teljes rendszer architektúrája

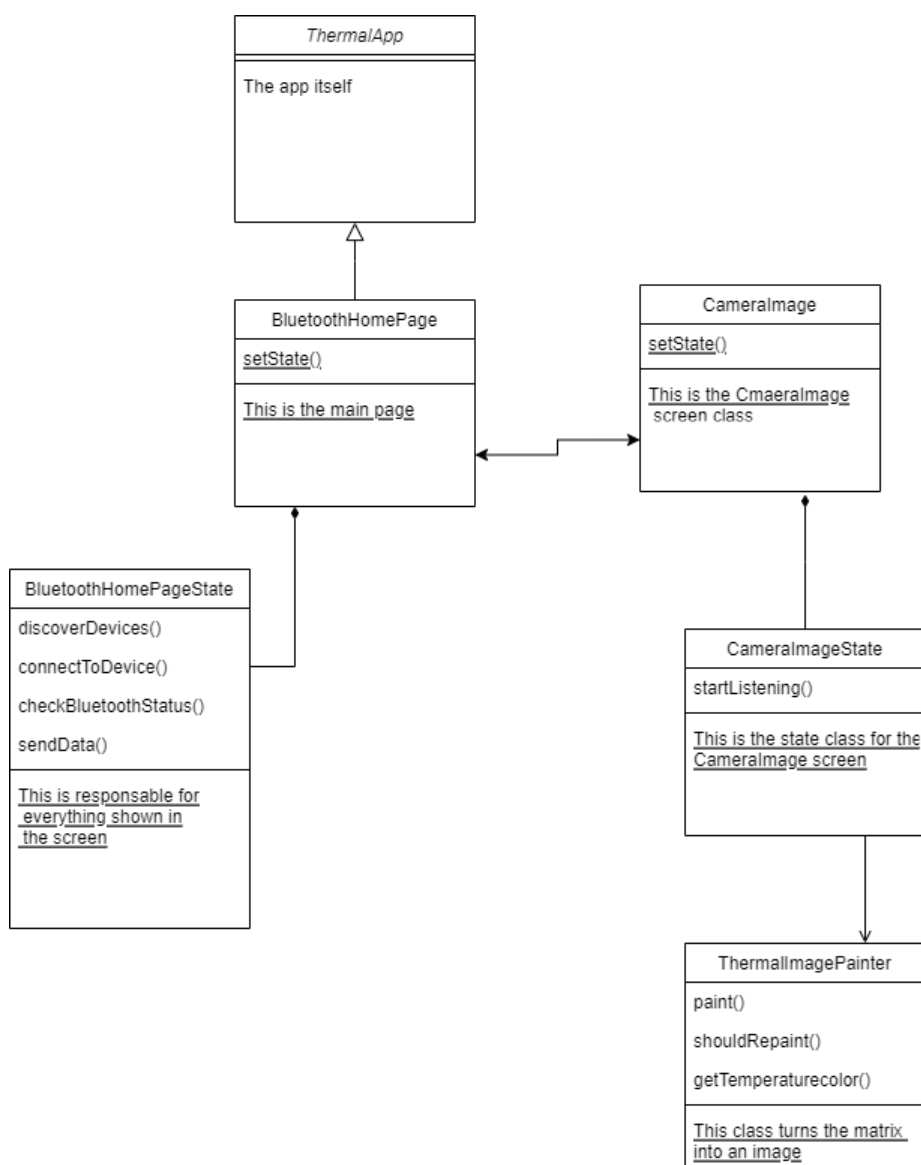
A 6. Ábrán látható a teljes rendszer architektúrája. Ebből az első komponens az Arduino Mega, mely a szenzorok által érzékelt értékeket dolgozza fel és kommunikál Bluetooth-on keresztül az applikációval. Az ultrahang szenzorok sima input bemeneten keresztül küldik a mért adatot a vezérlőegységnek, a termálkamera pedig I²C kommunikációt használva az SDA (sourcedata) adatfolyam segítségével. Ez továbbá szinkronban van a vezérlővel, ez a SCL (Sourceclock) adatfolyamnak köszönhető.

3.4.1. Thermal app

Az applikációt Thermal appnak neveztem el. Ennek átláthatósága érdekében két screen használok, egy a küldésért felel, a másik a fogadásért és képfeldolgozásért, amint ezt a 12. Ábra is mutatja. Ezek állapotát adja meg a BluetoothHomePageState és CameraImageState osztályok, melyek feladata a konkrét adatküldés, illetve adatfogadás. A bluetoothHomePageState osztály fogja a robotnak közölni a felhasználó parancsait. A robot által küldött információkat a

CameraImageState osztály dolgozza fel és jeleníti meg a felhasználónak, illetve menti el az eszközén.

Az alkalmazás indításakor létrejön maga az alkalmazás, melynek neve Thermal App. Ezzel asszociációs viszonyban van a BluetoothHomePage osztály, melynek az állapotát a BluetoothHomePageState osztály adja meg. E két osztály között kompozíció lép fel. Hasonló a helyzet a CameraImage és a CameraImageState osztályok között. Az utóbbi függvény függ a ThermalImagePainter osztálytól, mivel ez szolgáltatja neki a képet, amit meg kell jelenítsen, de ez nincs hatással élettartamára.



12. Ábra UML diagramja az alkalmazásnak

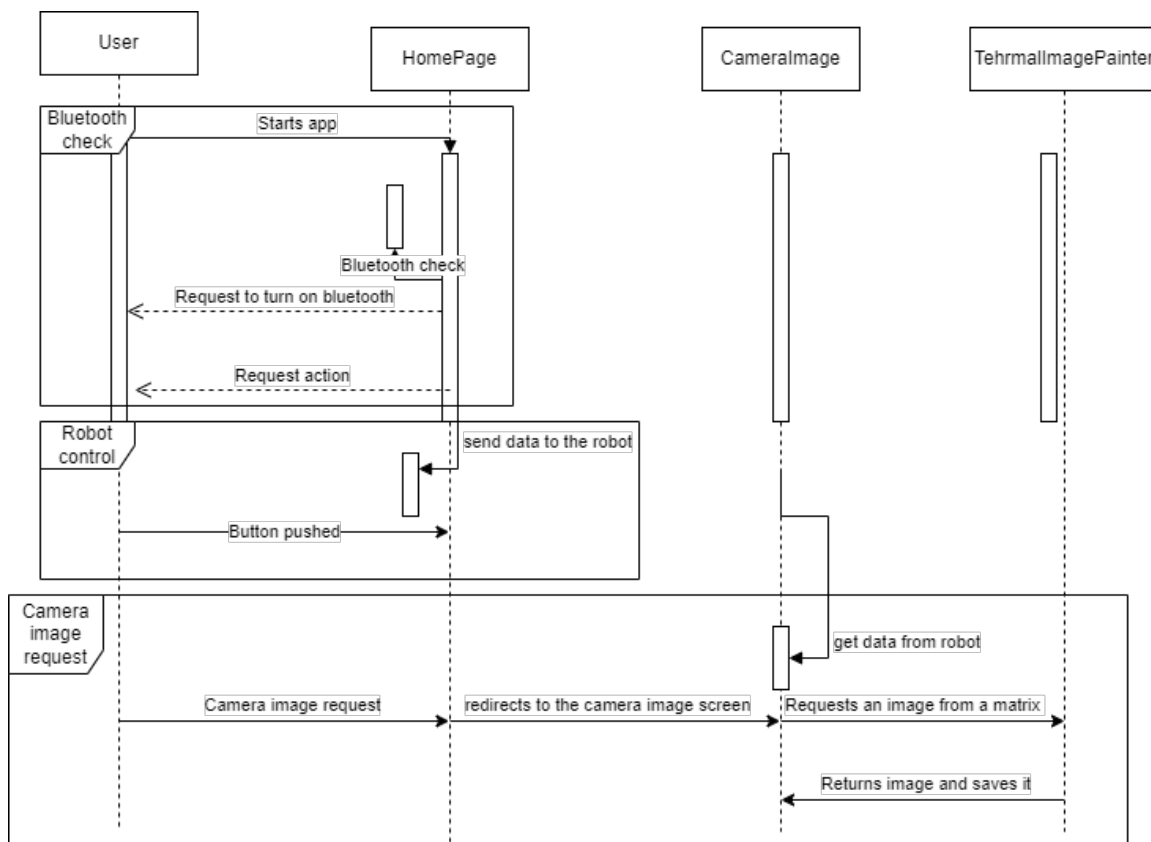
3.4.2. Kommunikáció

Többször is kiemeltem, hogy Bluetooth-ot használok kommunikációra, azonban felmerülhet a kérdés miért ezt használom? Valóban használhattam volna WiFi-t mint kommunikációs interfész, azonban tényleg megérte volna? Való igaz, hogy WiFin keresztül sokkal gyorsabban lehet adatot streamelni és nagyobb mértékben is. Távolság szempontjából is nagyobb a lefedettsége a WiFinak, azonban energiaigényes is emiatt. Az én projektem esetén egy akkumulátorról működik a teljes rendszer, tehát fontos az energiatakarékosság, illetve a WiFi által nyújtott lehetőségeket sem használom ki, mivel kicsi az az adatmennyiség, amit a küld, illetve fogad.

4. Részletes tervezés

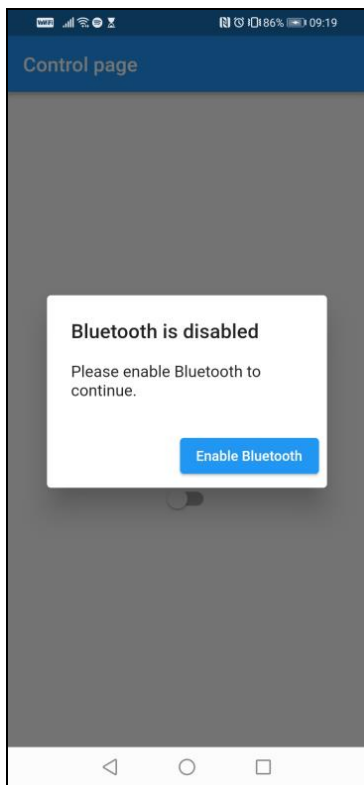
4.1. Thermal app

Az alkalmazás több szekvenciát követ, bizonyos felhasználói interakciók kezelésére, amint az a 13.Ábrán is látható. Az első ilyen szekvencia a Bluetooth kapcsolat ellenőrzése. Fontos, hogy a felhasználó eszközén be legyen kapcsolva, ugyanis enélkül nem tud kommunikálni a robottal. Amennyiben ez nincs bekapcsolva az alkalmazás egy figyelmeztetést küld neki, hogy kapcsolja be. Amint ezt megtette interakciókat végezhet a robottal. Ezt jelenti a diagrammon a robot control szekvencia. Az utolsó szekvencia a Camera Image Request. Ebben a szekvenciában a felhasználó kérést küld, hogy látni óhajtja a hőkamera képét. Ekkor átnavigál az ennek szánt screen-re, ahol is fogadja a robottól a kamera képeit és a ThermalImagePainter osztály segítségével képet alkot belőlük.

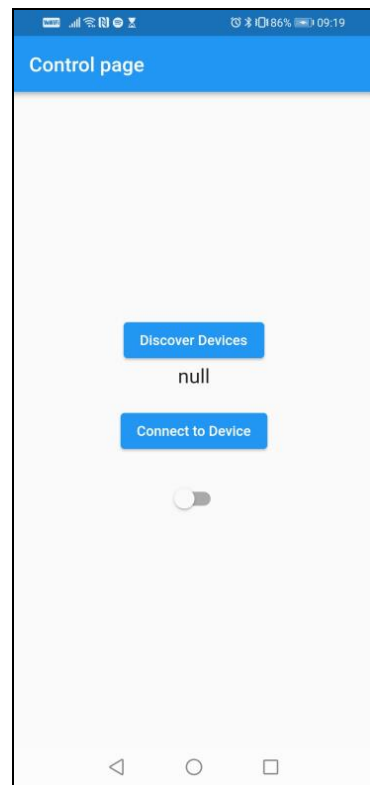


13. Ábra Az alkalmazás szekvencia diagrammja

A 13. Ábrán látható szekvenciadiagramm alapján az alkalmazás megnyitásakor a felhasználó a főoldalon találja magát, ahol, ha nincs bekapcsolva a Bluetooth, ez jelez neki, ahogy az a 14. Ábrán látható, így biztosítva, hogy biztos tudjon az eszközhöz kapcsolódni.



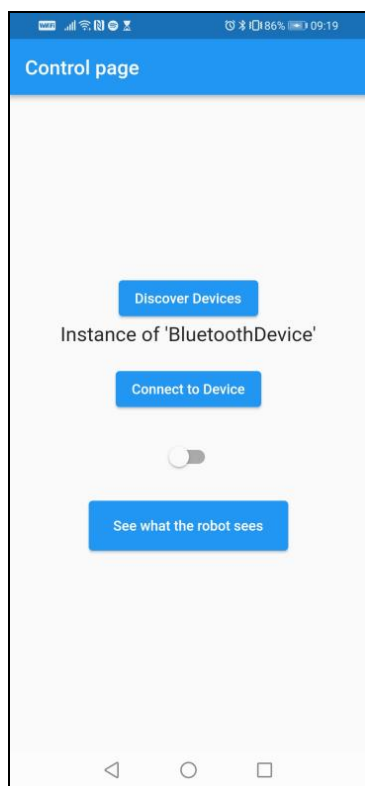
14. Ábra Bluetooth nincs bekapcsolva



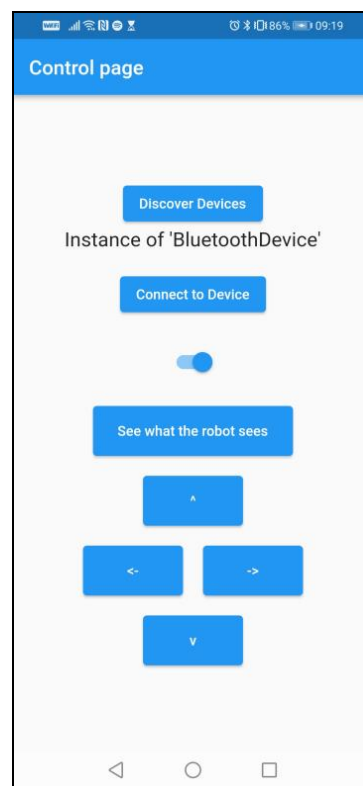
15. Ábra Alkalmazás főoldal csatlakozás előtt

Amennyiben be van kapcsolva a 15. Ábrán látható gombok jönnek vele szembe. Ezek teszik lehetővé, hogy csatlakozzon az eszközhöz. Szintén itt jelenik meg az a bizonyos kapcsoló, amivel eldöntheti, hogy milyen módban óhajtja használni a robotot.

Ezt követően, ha manuális módot választja a 17. Ábrán látható 4 gomb jön szembe vele, amikkel a robot mozgását valósíthatja meg. A 16. Ábrán látható az automata mód is.

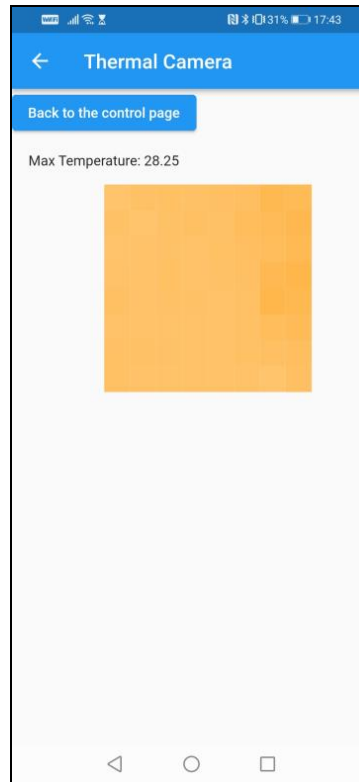


16. Ábra Automata mód



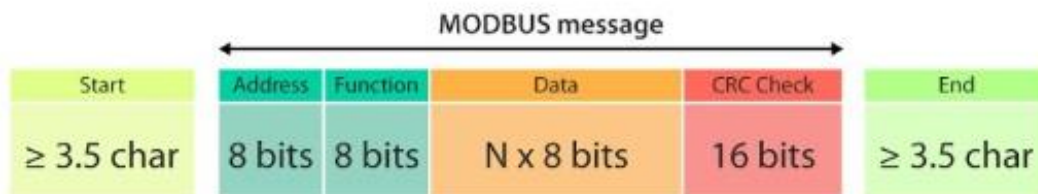
17. Ábra Manuális mód

Végül, úgy a 16. Ábrán, mint a 17. Ábrán szerepel egy See what the camera sees nevű gomb, mely elnavigálja a felhasználót a 18. Ábrán látható oldalra, ahol a kamera kép látható, mely adatérkezés esetén frissül. Ugyanezen az oldalon található a gomb mely visszanavigál a főoldalra, illetve az éppen aktuális hőmérsékleti extrém érték.



18. Ábra A kamera képe

4.1.1. Adatküldés



19. Ábra Modbus: Master-Slave modell esetében az üzenet

Az adatküldés telefonról Arduinora a Modbus: Master-Slave modellt próbálja követni. Ez egy, az ipari automatizálási rendszerekben gyakran használt protokoll, mely lehetővé teszi a kommunikációt egy mester és egy szolga eszköz között. Az előbbi kezdeményezi a kommunikációt és irányítja a folyamatát, az utóbbi pedig válaszol a mester parancsaira. A csomag, amit a mester küldd szemléltetve van a 19. Ábrán. Itt azt láthatjuk, hogy a csomag tartalmaz egy start, illetve egy stop bittet, ami a kommunikáció elejét és végét jelzi. Továbbá tartalmaz egy

címbájtót, aminek a 8-ik bittje határozza meg, hogy írás, vagy olvasás történik, egy parancsbájtót, a konkrét adatot és egy hibaellenőrzőt.

Az én rendszerem esetében az első érték a vezérlés típusát jelenti, ha 0 akkor manuális, ha 1 akkor pedig automatikus. A második érték azt jelzi, hogy igényel-e az alkalmazás képet a kamerától. A harmadik és negyedik érték a mozdulatot írja le, ahogy az alábbi 1. Táblázat bemutatja. Mint látható nem sikerült teljesen implementálni a Modbus modellt, viszont látszanak a hasonlóságok.

Mozdulat	3as érték	4es érték
előre	0	0
jobbra	0	1
balra	1	0
hátra	1	1

1. Táblázat Az értékeknek megfelelő mozdulat

4.1.2. Adatfogadás

Fogadás terén az alkalmazás egy 64 elemű tömböt kell fogadjon, ami duplapontos értékekkel van megtöltve, melyek egy kép színkódolását adják meg. Amennyiben az alkalmazás kérést küld a mátrixért, elkezdi a streamet figyelni és amint adatot kap elmenti és feldolgozza ezt.

4.2. Robot tervezés

A robot tervezése két részre osztható: a fizikai, azaz kézzel fogható elemek tervezésére, illetve Software tervezésére.

4.2.1. Robot építése

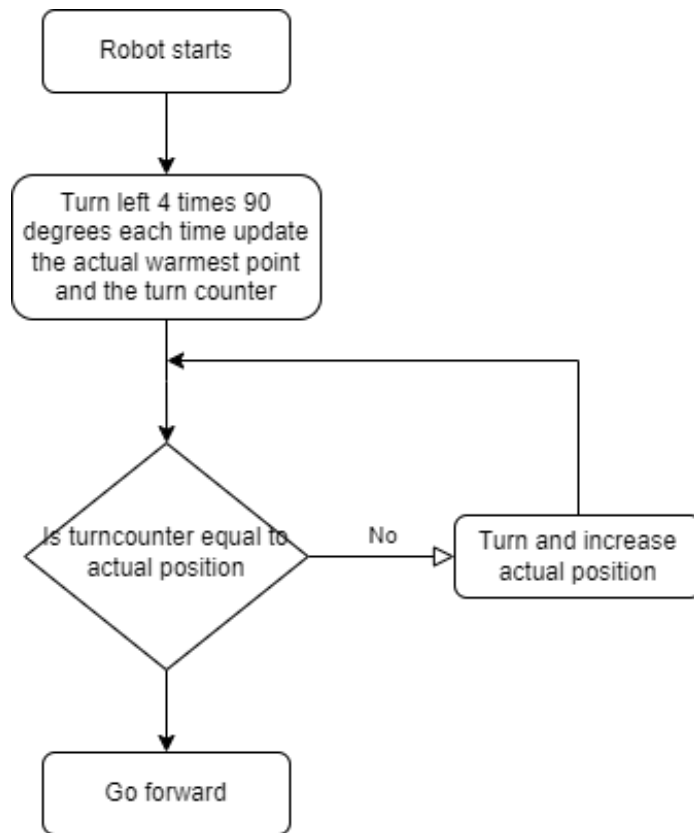
A robot építése egy alap beszerzésével kezdődött. Ennek egy tartós, de ugyanakkor enyhén képlékeny anyagnak kellett lennie. Így jutott a döntés a plexire, ami minden feltételemet teljesítette. Ennek a mérete nagyobb kellett legyen, mint az Arduino mérete, illetve a tápé, mivel nem kellene lelógjanak róla működés közben. Hely spórolás szempontjából szükségem volt egy Shieldre is, hogy az áramköröket meg tudjam valósítani, illetve 2 darab dc dc konverterre is,

mivel a motorok 12 volton működnek, viszont az Arduino kimeneti feszültsége csupán 5 volt. Valamint az is fontos volt, hogy a kamera magasítva legyen, illetőleg vertikális pozícióban legyen. Az is fontos volt, hogy a Bluetooth modul RXD pinje 3.3 V-on működik, így létre kellett hoznom egy feszültségosztó áramkört.

Kód felépítése

Kód szempontjából figyelembe kellett vennem, hogy az Arduino magának mely kimenetei használhatóak interrupció kimenetként, illetve mely kimenetei használhatóak kommunikációs portként. Ez azért volt fontos, mert, ha például a 0 és 1-es PWM bemenetbe kötöm a Bluetooth modulom, akkor az alap Soros portját zavarom.

Az automata irányítás azzal kezdődik, hogy a robot egy adott irányba fordul négyet, ezzel egy 360 fokot megtéve és közben jegyzi hol látta a legmelegebb pontot és ezt hány forgásból találta meg. Amint befejezte a forgást fordul még annyit amennyi fordulatból a legmelegebb pontot megtalálta. Miután megtalálta ismét a legmelegebb pontot elindul felé előre. Ezt szemlélteti a 20. Ábra.



20. Ábra Automata üzemmód lépései

5. Üzembehelyezés és kísérleti eredmények

Ez a fejezet a rendszer üzembe helyezésének folyamatát, tesztelését és a működő rendszerrel végzett kísérleteket, méréseket kell, tartalmazza.

5.1. Üzembehelyezési lépések

5.1.1. Robot építés

A robot építése azzal kezdődött, hogy a plexilapra kimértem, hogy a kerek hova fognak kerülni. Itt figyeltem arra, hogy nagyjából egyenletes távolságra legyenek. Miután kimértem hova lesznek felfogva a plexi kifúrásával fel is tudtam erősíteni őket süllyesztett fejű 6os csavarok segítségével. A motorok felfogásához készítettem alumíniumból két foglalatot, melyek ezeket a lap aljára erősítették 2-2 csavar segítségével.

Ezután következett az Arduino és a kamera plexije. Az Arduinot 3 csavarral fogtam fel a laphoz, amihez 3 belülről is menetelt csavart használtam, illetve anya helyett műanyagot is

használtam, hogy az Arduino magasságát tudjam állítani, ha erre szükség lenne. A kamera plexijét egy csavar és anya segítségével rögzítettem a lapra, itt is használva egy műanyagot magasztás érdekében. A maradék rendelkezésemre álló helyre elhelyeztem ragasztó segítségével a tápot, illetve a két DC-DC konvertert, illetve szintén ragasztó segítségével a robot elejére erősítettem a 3 ultrahang szenzort.

Ezután következett a shield-en az áramkörök kialakítása. Első lépésben felforrasztottam a Bluetooth modulomat, illetve a motor vezérlő egységem. Ezután kialakítottam az ezeknek szükséges áramköröket, illetve a többi alkatrész kábelezését is megcsináltam, némelyet szintén forrasztás segítségével.

5.1.2. Android kódolás

Az applikáció elkészítéséhez Flutter-t használtam és Android Studio keretein belül dolgoztam.

Az alkalmazás indításakor, amennyiben nincs Bluetooth aktiválva, egy párbeszédablak ugrik fel, ami értesíti a felhasználót, hogy kapcsolja ezt be és erre egy gomb is segítségére van. Ez úgy történik, hogy a FlutterBluetoothSerial könyvtárnak az isEnabled függvénye segítségével ellenőrzöm az aktuális állapotát a Bluetooth-nak és ha false feldobom az előbb is említett párbeszédablakot egy gombbal, ami segítségével be lehet a Bluetooth-ot kapcsolni. A bekapcsolást is ugyanannak a könyvtárnak egyik függvénye végzi, pontosabban a requestEnable függvény. Ezt szemlélteti a 1.Függelék ábra.

Amikor a 15. Ábrán látható Connect to device gombra nyom a felhasználó létrejön egy üres lista, amibe a már párosított Bluetooth eszközök listája tárolódik el és egy párbeszédablakban a felhasználónak is megjelenik. A flutter_bluetooth_serial plugin által nyújtott függvény segítségével kérem le őket, ahogy az a 2.Függelék ábrán látható. Ezeket behelyezem egy listanézetbe és egy text widgetbe helyezem a nevüket. Amennyiben rányomunk egyre annak az eszköznek a nevét átadom egy _device változónak, ami ennek a nevét tárolja el.

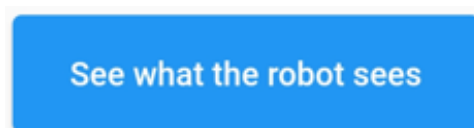
Amint kiválasztotta a felhasználó melyik eszközhöz óhajt csatlakozni és megnyomja a Connect gombot szintén ebből a könyvtárból származó kapcsolódás függvény, melyet a 3.Függelék ábra szemléltet segítségével létrejön a kapcsolat, ellenkező esetben a hibaüzenetet kiírom terminálra. Ekkor lesznek elérhetőek a robot vezérlésére használandó gombok is. Ezek ElevatedButton típusú gombok. Bármelyik vezérlő gomb lenyomása után elküldésre kerül az

aktuális vezérlőlista, azaz a `_dataList` lista, amely `UInt8List` típusú és 4 értéket tartalmaz. Alap inicializálása a következőképpen történik:

```
UInt8List _dataList= UInt8List.fromList([0,0,0,0]);
```

21. Ábra A vezérléshez használt lista

Az első elem a vezérlés típusát írja le, a második egy kérés, hogy igényli-e az alkalmazás a kamera mátrixát és az utolsó kettő pedig az 1. Táblázatban bemutatott értékkombinációk valamelyikét tartalmazzák. Amennyiben a 22. Ábrán látható gomb megnyomódik elnavigál az alkalmazás a kamera képét mutató screenre.



22. Ábra Kamera kép kérő gomb

A fentebb említett screenen lesz a kamera mátrixából kép. Ezt a mátrixot a `_startListening()` függvény segítségével kérem le (4. Függelék ábra), ami a `flutter_bluetooth_serial` könyvtár `listen` függvénye segítségével hallgatja az adatfolyamot és amint van bejövő adat beleteszi egy karakterláncba addig, amíg tartalmazni fog egy sor vég karaktert (`\n`). Miután fogadta a mátrix összes elemét feldarabolja a karakterláncot a `String.split()` függvény segítségével `,` karakter szerint. A feldarabolt elemek egy `String` típusú listába kerülnek (`_dataList`), aminek minden elemét átkonvertálom dupla pontos elemmé és egy dupla pontos listába helyezem át, feltéve, ha sikeresen át lehet konvertálni. Ellenekező esetben 0.0-ás érték kerül a listába annak az elemnek a helyére. Itt van továbbá a kép mentésének folyamata is lekezelve. Itt létrehozok egy vásznat, amire a képet helyezi a függvény és kimentí az alkalmazás mappájába a felhasználó eszközére.

Lévé, hogy a mátrix 64 elemet tartalmaz, tehát 8X8-as nem lesz egy nagy és részletes kép, de a hőértékek látszanak így is. A hőmérsékleti értékek 3 szín különböző árnyalatait kaphatják: kék, amennyiben az érték kisebb, mint 30, narancssárga, ha hőmérséklet 30 és 40 között van, illetve piros, ha 40 fok felett van az érték. Az értékek festését a `ThermalImagePainter` osztály végzi, ami a `CustomPainter` beépített osztályból lett származtatva és paraméterként kapja meg az `_imageData` listát. Az 5. Függelék ábrában bemutatott függvény segítségével végzem az

értékek festését. Mindenik pixelértéknek kiszámolom a neki megfelelő színt és annak árnyalatát a `_getColorFromTemperature()` függvény() segítségével. Ebben a függvényben egy skála szerint meghatározom milyen színű lesz az adott pixel(kék, narancssárga, piros), majd a `_getTemperatureColors()` függvény segítségével meghatározom az árnyalatát is és ezt visszaküldöm a `paint()` függvénynek. A színárnyalat váltást úgy érem el, hogy a szín világosságából, ami normál körülmények esetén 1.00 kivonom a hőmérséklet/50.0-át, ahogy ezt a 9. Függelék ábra mutatja. Azért 50, mert a jól láthatóság miatt levágom a kamera által küldött hőmérsékleti értékeket és a maximális értékem ami kaphatok az 50, a minimális pedig 20. Ezt a következőképpen oldottam meg a `_getColorFromTemperature()` függvényben:

```
if(temperature<20.0) {  
    return Colors.blue.shade900;  
}  
if(temperature>50.0) {  
    return Colors.red.shade900;  
}  
}
```

23. Ábra Hőmérsékleti értékek levágása

Amint új kép érkezik a kameráról a `shouldRepaint()` függvény újra színezi a frissült pixeleket. A 24. Ábrán látható a kép, amit így generálok 26 pixeles hosszúsággal és szélességgel.

5.1.3. Arduino kódolás

Az Arduinora készült kódokat a fejlesztő laphoz járó nyílt forráskódú Arduino IDE környezetben fejlesztettem.

Ezen téren először minden szenzort és alkatrészt külön-külön teszteltem és ezután készítettem el a végső kódot. Az ultrahang szenzorok olvasásához a `newPing` könyvtárat használtam, annak is a `ping()` függvényét, ami miatt a szenzor kibocsát egy ultrahangot az echo pin hatására, ami visszaverődik az első eltalált akadályról és ezt felfogja a trigger egység. Az eltelt időből ki lehet számolni a megtett útját a hangnak ismerve a hangsebességet.

$$d = v_s \cdot t,$$

ahol d a megtett út, v_s a hangsebesség, ami konstans érték(330 m/s), illetve a t az eltelt idő. Amikor a szenzorokat inicializáltam megadtam először is, hogy melyik pinekhez vannak csatlakoztatva, majd egy NewPing típusú objektumot hoztam létre meindeniknek a 24. Ábrán látható módon.

```
NewPing sonar1(trigPin1, echoPin1);  
NewPing sonar2(trigPin2, echoPin2);  
NewPing sonar3(trigPin3, echoPin3);
```

24. Ábra Ultrahangszenzor inicializálás

A hőkamera olvasásához az Adafruit_AMGxx.h könyvtárat használtam, azon belül a readPixels() függvényt, ami egy 64 méretű tömbbe olvas be lebegőpontos értékeket. Ezt a tömböt küldöm Bluetooth-on keresztül az alkalmazásnak. Ez úgy történik, hogy létrehozok egy soros kimenetet a softwareSerial könyvtár segítségével a 25.Ábrán látható módon.

```
SoftwareSerial bluetooth(10, 11); // RX, TX
```

25. Ábra Bluetooth modul inicializálása

Ezt követően inicializálom a megfelelő baudrate-el, ami az én esetemben 9600 volt és már használhatom is küldésre, illetve fogadásra. Fogadás terén a könyvtár available() függvénye mondja meg, hogy van-e érkező adat. Adatfogadás után feldolgozásra kerül a kapott 4 byte érték, ahogy a 7. Függelék ábra is mutatja. Az elsővel kiválasztom a vezérlés típusát(kézi, automatikus), majd meghívom a megfelelő függvényeket. A manuális vezérlés a harmadik, illetve a negyedik byte értékeinek függvényében valósul meg, ahogy az 1. Táblázatban is vázoltam. Ezt követően küldök ki egy HIGH jelet és a megfelelő irányt a megfelelő motornak 700 másodpercig mivel ennyi idő, amíg a terhelt kerekek végrehajtanak egy rotációt. Ezt szemlélteti a 11. Függelék ábra. A második érték a mondja meg, hogy van-e igény a kamera képére és amennyiben van elküldésre kerül. Ahogy a 8. Függelék ábrán is látható, minden érték bekerül egy Stringbe és kerül utána egy ',' karakter szeparáció végett. A küldési puffer végét a '\n', azaz az endline karakter jelzi.

Manuális mód esetében első lépésben a robot megtesz 4 bal kanyart, ezzel szimulálva a felhasználó általi mozgatót. A kanyarodásért felelős függvény a 26 . Ábrán látható.

```

void startTurningLeft() {
    digitalWrite(B1A, LOW);
    digitalWrite(B1B, HIGH);
    digitalWrite(A1A, LOW);
    digitalWrite(A1B, LOW);
    delay(700);
    stopMoving();
}

```

26. Ábra kanyarodásért felelős függvény

Minden kanyar után a kamera aktuális mátrixában megnézi melyik a legmelegebb pont és elmenti a temp változóba. Ezt minden kanyar után frissíti és jegyzi, hogy hány fordulásból találta meg a poz változóba. Miután mind a 4 kanyart megtette még fordul annyit amennyi a poz változó értéke és elindul előre, megközelítve a pontot. Előremenetel előtt kiíratom mindkét változó tartalmát a Soros kimenetre, hogy meg tudjam figyelni a helyes működést. A kód a 12. Függelék Ábrán látható.

5.2. Felmerült problémák és megoldásaik

Az első probléma, amivel találkoztam a kivitelezés során az a motorok működésre bírása volt. Amint már említettem ezek 12 volton működnek és az Arduino 5 voltot képes leadni. Erre kaptam a vezető tanáromtól az ötletet, hogy alkalmazzunk Dc-Dc konvertereket. Ezekkel a probléma azonban, hogy a rajtuk lévő potméterek nem épp a legminőségibbek a piacon és elég hamar elromlanak. Erre a megoldás ellenállások alkalmazása lett, így ugyan nem lehet állítani a konverter kimeneti feszültségét, cserébe viszont a motornak megfelelő értéken marad.

Szintén egy probléma volt, hogy nem találtam kerekeket a motorokhoz, ugyanis a foglalatjuk annyira speciális, hogy semmi nem ment rájuk fel. Erre a megoldást végül közösen édesapámmal találtuk meg, ugyanis az ő segítségével készítettünk két plexi kereket és elláttuk őket gumival a tapadás elérése érdekében.

Egy további probléma volt, hogy miután a shield-en megcsináltam a kellő áramköröket a kábelek takarták a kamerát. Ezt úgy sikerült kiküszöbölni, hogy a kamera taróját egy műanyag segítségével megemelttem.

Problémákkal kódolás közben is találkoztam főleg az alkalmazás fejlesztésekor. Nagyon nagy problémám volt a kapcsolat létrehozása. A baj az volt, hogy az alkalmazás induláskor

megnézte, hogy található HC-05 nevű Bluetooth eszköz a párosított eszközök listájában és próbált csatlakozni rá. Itt a megoldás az volt, hogy párosítani kellett egy kóddal a két eszközt. Ez egy biztonsági intézkedés a gyártó részéről, hogy biztonságos legyen a kapcsolat, nekem azonban elég nagy fennakadást okozott.

5.3. Kísérleti eredmények, mérések

Több mérést kellett végezni a szenzorokkal külön-külön, hogy tanulmányozzam a működésüket. Az ultrahang szenzorok pontosságát egy 20 cm-es vonalzó segítségével teszteltem. Elhelyeztem egy tárgyat 5, 10, 15 és 20 centis távolságokra. A 2. Táblázat mutatja be a szenzorok válaszát külön-külön.

Távolság	Szenzor 1 eredmény	Szenzor 2 eredmény	Szenzor 3 eredmény
5cm	6,1 cm	5,7 cm	5,8 cm
10cm	10,7 cm	10,7 cm	10,4 cm
15cm	15,5 cm	15,6 cm	15,5 cm
20cm	21 cm	20,7cm	20,6 cm

2. Táblázat Ultrahangszenzor mérési eredmények

Látható, hogy a mérési eredmények nagyobbak, mint a távolság. Ez annak tudható be, hogy döntött pozícióban vannak a szenzorok egyrészt, másrészt pedig maximum másfél centit tévedhetnek. A helyes működés érdekében korlátoztam a szenzorok felső korlátját 200 cm-re. Ez azért volt szükséges, mert egyrészt, ha a robot túl messze van egy akadálytól 0 lesz a szenzor visszatérítési értéke, ami a későbbi méréseim és a helyes működést is akadályozza.

A további méréseket a kamerával végeztem. Tesztelésekor különböző hőmérsékletű eszközöket, illetve embereket helyeztem elé és jegyeztem le a mért értékeket. A 3. Táblázatban egy forrasztópáka fejének hőmérsékletét mértem le különböző szögekből, hogy megfigyeljem a kamera pontosságát és látási távolságát.

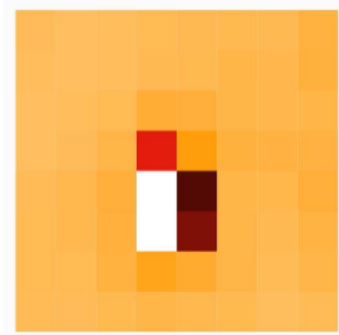
Tárgy szöge a kamerához képest	Legmelegebb pont indexe	Legmelegebb pont hője [C°]
0°	3	105
15°	5	105
30°	0	105

3. Táblázat Forrasztópáka hője kamerával mérve, több pozícióból

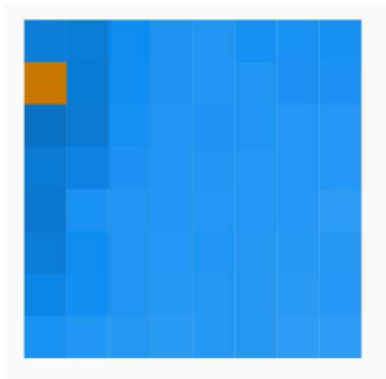
A táblázatból kivehető, hogy pozíciótól függetlenül a kamera ugyanazt az értéket adja vissza, addig a pontig amíg a látószögén belül van a tárgy. A mérés során arra is rájöttem, hogy a kamera nagyjából a tőle 1 méterre levő dolgokat érzékeli és érdekességképp jegyezném meg, hogy például egy embert nem érzékel, ha ruha fedi. Az előbb említett dolgok láthatóak az alábbi képeken, amiket a kamera mátrixából jelenítettünk meg. Például az előbb említett pákáról alkotott kép a 28.Ábrán látható. A 27.Ábrán a kezemről alkotott hőkép látható. A 29. Ábrán egy szobának a hője látható, rögtön utána pedig két kép egy gyújtóról, úgy, hogy az első a gyújtó közel van a kamerához, a másikon pedig majdnem 1 méterre tőle. Ezekből az következik, hogy a kamera bár pontos, kicsi a hatótávolsága(alig 1 méter).



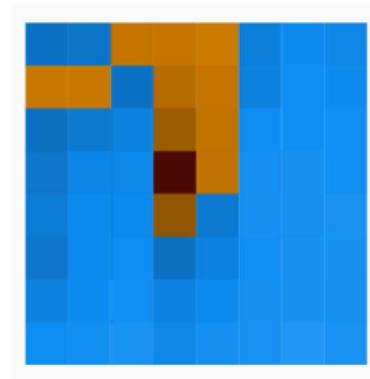
27. Ábra Kamera képe egy emberi kézzől



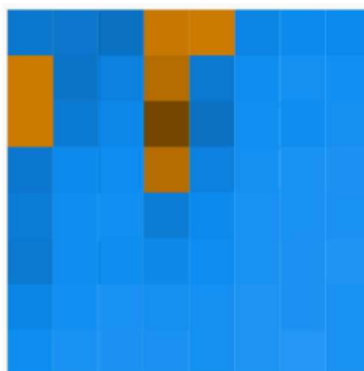
28. Ábra Kamera képe egy pákáról



29. Ábra Kamera képe egy szobáról



30. Ábra Kamera képe egy gyűjtőről
(kevesebb, mint 1 méterről)



31. Ábra Kamera képe egy gyűjtőről (1 méter határán)

6. A rendszer felhasználása

A rendszerem nagy segítséget nyújthat osztályokban, irodákban, laboratóriumokban, ahol sok az elektronikai berendezés és ezek állapotát kívánják felügyelni. Tehát az oktatási rendszerben, de akár az iparban is segítséget nyújthat.

Az építőiparban is hasznos lehet, ugyanis segíthet azonosítani a rossz hőmérséklet-szabályozással, vagy légszivárgással rendelkező területeket az épületekben, lehetővé téve a fűtési, szellőzési és légkondicionálási rendszerek célzott javítását és fejlesztését. Ugyanakkor a potenciális tűzveszélyes, vagy túlmelegedő berendezések azonosításában ipari környezetben,

vagy háztartásokban, ezzel lehetővé téve a korai beavatkozást és tűz megelőzési intézkedéseket. A háztartásoknak azonban nem csak ilyen formában lehet hasznos a rendszer. Segíthet a hővesztéssel, vagy a nem megfelelő szigeteléssel rendelkező területek azonosításában az épületen belül. Ezzel energiafogyasztásuk optimálisabb lesz és fűtési költségeik is csökkennek.

Egészségügyi környezetben a robot segítséget nyújthat a betegek testhőmérsékletének ellenőrzésére azonosítva a fertőzéseket, gyulladásokat, vagy más gyanús forró pontot a betegen, ami problémás lehet. Emellett az orvosi eszközök állapotának ellenőrzésében, illetve hűtőtárolók hőjének ellenőrzésében is.

Vészhelyzetek esetén, mint például földrengések, ami épületek összeomlásával jár sokszor, a robot áldozatok felkutatásában nagy segítség lehet, mivel a romok alatt érzékeli a testük hőjét. Ez felgyorsítaná a mentési művelet menetét és növelné a túlélők számát.

Az üvegházi vagy mezőgazdasági környezetekben a robot segíthet a hőmérséklet ingadozások azonosításában és a növények növekedési körülményeinek optimalizálásában. Segítséget nyújthat az öntözőrendszerek vagy egyéb kertészethez használt berendezések hibás működése által okozott forró pontok forró pontok felderítésében is.

A robot szokatlan hőjelek, vagy betolakodók észlelésével korlátozott területeken fokozhatja a biztonsági rendszereket és a megfigyelési erőfeszítéseket.

Hűtőházak esetében, melyekben húsokat, vagy egyéb romlandó termékeket tárolnak a robot képes lenne megtalálni az esetleges szivárgási pontokat ezáltal az áru épségét garantálva.

7. Következtetések

A projekt megvalósítása során sok tapasztalatot szereztem, úgy hardware, mint software fejlesztés terén. Sikerült mélyebb belátást nyerni az Arduino mikrovezérlők világába, de ugyanakkor a mobilfejlesztési készségeim is fejlődtek. Valamint sikerült jobban megértenem a PWM és a Bluetooth fogalmakat, amikre később még valószínűleg szükségem lesz.

7.1. Megvalósítások

A projekt célkitűzéseim közül sikerült megvalósítanom egy alkalmazást, mely képes Bluetooth-on keresztül csatlakozni a robothoz, ennek képes adatot küldeni és képes adatot fogadni is tőle. A fogadott adatból képes egy képet létre hozni és ezt elmenteni a telefonra.

Robot terén sikerült a távvezérlő funkciókat implementálni, valamint a szenzorok olvasása és adatainak feldolgozása is sikeresen megvalósult.

7.2. Továbbfejlesztési lehetőségek

Továbbfejlesztési lehetőségként említeném a telefonos alkalmazásba építhető haptic feedback funkciót, aminek lényege, hogy ha például neki vezetjük a robotot a falnak akkor elkezd a telefon rezegni jelezve a felhasználónak, hogy akadályba ütközött.

Az autómata vezérlés sem tökéletes, nem veszi ugyanis a legmelegebb pontot középre, így továbbfejlesztésként ezt is komplexebb síkra lehet helyezni. A kamera mátrixának pozíciója függvényében kell oszlop száma $\cdot 7.5$ fokot forduljon valamelyik irányba a robot, hogy középre kerüljön a célpontja.

Egy másik továbbfejlesztési lehetőségként jegyezném meg, hogy lehetne két kapcsolót szerelni a robot oldalára, ezzel elkerülve az olyan akadályokat melyeket az ultrahang szenzorok nem láttak, de akadályozzák a robot haladását.

8. Köszönetnyilvánítás

Ezúton is köszönetet szeretnék mondani témavezetőmnek, dr. Bakó Lászlónak a belém fektetett bizalmáért, türelméért és a szakmai tanácsadásáért.

9. Irodalomjegyzék

- [1] Line Follower Using Arduino And Its Applications 3as oldal 4es fejezet első mondat.
- [2] Understanding and Design of an Arduino-based PID Controller Dinesh Bista 17/72 első bekezdés.
- [3] Márton Lőrinc és Fehér Áron Irányítástechnika laboratóriumi útmutató 4.2es fejezet első mondat.
- [4] P szabályozók beavatkozó jelének képlete
- [5] P szabályozók hibaszámítási képlete
- [6] PID szabályozók beavatkozó jelének képlete
- [7] Line Follower Robot(with PID controller)
<https://projecthub.arduino.cc/anova9347/line-follower-robot-with-pid-controller-01813f>
- [8] Line Follower Robot & Obstacle Detection Using PID Controller D.Vijendra Babu, D.C.Jenniffer,R.Karthikeyanhttps://www.researchgate.net/publication/345499688_Line_Follower_Robot_Obstacle_Detection_Using_PID_Controller
- [9] Arduino mega specifikációk: <https://store.arduino.cc/products/arduino-mega-2560-rev3>
- [10] HC05 bluetooth modul specifikációk:
https://components101.com/sites/default/files/component_datasheet/HC-05%20Datasheet.pdf
- [11] AMG8833 specifikációk: <https://cdn-learn.adafruit.com/downloads/pdf/adafruit-amg8833-8x8-thermal-camera-sensor.pdf>
- [12] HCSR04 specifikációk:
- [13] Motrok adatlapja https://digilent.com/reference/_media/motor_gearbox/290-006_ig220019x00015r_ds.pdf
- [14] HW-016-specifikációk:
https://www.ti.com/lit/ds/symlink/drv8833.pdf?ts=1686905600685&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FDRV8833%252Fpart-details%252FDRV8833RTYR

- [15] [https://www.tutorialspoint.com/the-bluetooth-protocol-architecture#:~:text=The%20commonly%20adopted%20protocols%20used,Wireless%20Application%20Protocol%20\(WAP\).](https://www.tutorialspoint.com/the-bluetooth-protocol-architecture#:~:text=The%20commonly%20adopted%20protocols%20used,Wireless%20Application%20Protocol%20(WAP).)
- [16] https://www.hobbielektronika.hu/cikkek/kommunikacio_alapjai_-_soros_adatvitel.html?pg=5
- [17] Github repo: https://github.com/JBLogistic/Licence_Exam.git
- [18] Draw.IO: <https://app.diagrams.net>

10. Függelék

```
Future<void> _checkBluetoothStatus() async {  
  bool? isEnabled = await FlutterBluetoothSerial.instance.isEnabled;  
  if (!isEnabled!) {  
    showDialog(  
      context: context,  
      builder: (BuildContext context) {  
        return AlertDialog(  
          title: Text('Bluetooth is disabled'),  
          content: Text('Please enable Bluetooth to continue.'),  
          actions: [  
            ElevatedButton(  
              child: Text('Enable Bluetooth'),  
              onPressed: () async {  
                Navigator.of(context).pop();  
                await FlutterBluetoothSerial.instance.requestEnable();  
              },  
            ),  
          ],  
        );  
      },  
    );  
  }  
}
```

1. Függelék ábra Bluetooth állapot ellenőrző függvény

```

Future<void> _discoverDevices() async {
  List<BluetoothDevice> devices = [];
  try {
    devices = await FlutterBluetoothSerial.instance.getBondedDevices();
  } catch (e) {
    print(e);
  }

  showDialog(
    context: context,
    builder: (BuildContext context) {
      return AlertDialog(
        title: Text('Select a device'),
        content: ListView.builder(
          itemCount: devices.length,
          itemBuilder: (BuildContext context, int index) {
            return ListTile(
              title: Text(devices[index].name.toString()),
              subtitle: Text(devices[index].address),
              onTap: () {
                Navigator.pop(context, devices[index]);
              },
            );
          },
        ),
      );
    },
  );
}.then((value) {
  if (value != null) {
    setState(() {
      _device = value;
    });
  }
});
}

```

2. Függelék ábra Eszközkiválasztás függvény

```

Future<void> _connectToDevice() async {
  if (_device == null) {
    return;
  }

  try {
    _connection = await BluetoothConnection.toAddress(_device?.address);
    print('Connected to device');
    // Perform any further operations with the connection
  } catch (e) {
    print(e);
  }
}

```

3. Függelék ábra Connect függvény


```

String _buffer = '';
void _startListening() {
  _localConnection!.input!.listen((value) async {
    String data = utf8.decode(value); // Convert the received bytes to a string
    _buffer += data; // Add the received data to the buffer
    if (_buffer.contains('\n')) {
      // A complete message has been received
      List<String> dataString = _buffer.split(','); // Split the string by commas
      // Parse the received data
      List<double> newData = [];
      newData.clear();
      newData.addAll(dataString
        .map((stringValue) => double.tryParse(stringValue) ?? 0.0)
        .toList());
      setState(() {
        for (int i = 0; i < newData.length; i++) {
          if (i < _imageData.length) {
            _imageData[i] = newData[i];
            _maxTemperature = max(_maxTemperature, newData[i]);
          }
        }
      });
      _buffer = ''; // Clear the buffer

      final recorder = PictureRecorder();
      final canvas = Canvas(recorder);
      final size = Size(208, 208); // 8 * 26
      final painter = ThermalImagePainter(_imageData);
      painter.paint(canvas, size);
      final picture = recorder.endRecording();
      final img =
        await picture.toImage(size.width.toInt(), size.height.toInt());
      final pngBytes = await img.toByteData(format: ImageByteFormat.png);

      final directory = await getApplicationDocumentsDirectory();
      final file = File('${directory.path}/my_image.png');
      await file.writeAsBytes(pngBytes!.buffer.asUint8List());
    }
  });
}

```

4. Függelék ábra Fogadó függvény

```

void paint(Canvas canvas, Size size) {
    double pixelWidth = 26;
    double pixelHeight = 26;

    for (int row = 0; row < 8; row++) {
        for (int col = 0; col < 8; col++) {
            double temperature = pixels[row * 8 + col];
            Color? color = _getColorFromTemperature(temperature);

            canvas.drawRect(
                Rect.fromLTWH(
                    col * pixelWidth,
                    row * pixelHeight,
                    pixelWidth,
                    pixelHeight,
                ),
                Paint()..color = color!,
            );
        }
    }
}

```

5. Függelék ábra paint() függvény

```

Map<double, Color> getTemperatureColors(Color color, double temperature) {
    final hslColor = HSLColor.fromColor(color);

    Map<double, Color> temperatureColors = {};

    // Calculate the lightness value based on the temperature,
    // with lower temperatures resulting in lighter colors
    final lightness = 1.0 - (temperature / 50.0);

    final shade = hslColor.withLightness(lightness).toColor();
    temperatureColors[temperature] = shade;

    return temperatureColors;
}

```

6. Függelék ábra getTemperatureColors() függvény

```

Color? _getColorFromTemperature(double temperature) {
    if(temperature<20.0) {
        return Colors.blue.shade900;
    }
    if(temperature>50.0) {
        return Colors.red.shade900;
    }
    if(temperature<30.0) {
        Map<double, Color> temperatureShades = getTemperatureColors(
            Colors.blue, temperature);
        Color? temperatureColor = temperatureShades[temperature];
        return temperatureColor;
    }
    if(temperature<40.0) {
        Map<double, Color> temperatureShades = getTemperatureColors(
            Colors.orange, temperature);
        Color? temperatureColor = temperatureShades[temperature];
        return temperatureColor;
    }
    if(temperature<50.0) {
        Map<double, Color> temperatureShades = getTemperatureColors(
            Colors.red, temperature);
        Color? temperatureColor = temperatureShades[temperature];
        return temperatureColor;
    }
    return Colors.white;
}

```

7. Függelék ábra *_getColorFromTemperature()* függvény

```

#include <SoftwareSerial.h>
#include <Adafruit_AMG88xx.h>
#include <NewPing.h>
Adafruit_AMG88xx amg;
SoftwareSerial bluetooth(10, 11); // RX, TX

```

8. Függelék ábra *Arduino kódban használt könyvtárak*

```

byte val1, val2, val3, val4;
if(isConnected){
  for (int i = 0; i < 4; i++) {
    byte receivedByte = bluetooth.read();
    Serial.print(receivedByte, DEC);
    Serial.print(' ');
    if(i == 2){
      val3 = receivedByte;
    }
    if(i == 3){
      val4 = receivedByte;
    }
    if(i == 0){
      val1 = receivedByte;
    }
    if(i == 1){
      val2 = receivedByte;
    }
  }
}

```

9. Függelék ábra Arduino oldalon a fogadás

```

amg.readPixels(sensorArray);
if(val2 == 1){
  message_to_send = "";
  for (int i = 0; i < ARRAYSIZE; i++) {
    message_to_send += String(sensorArray[i]) + String(",");
  }
  message_to_send += String("\n");
  bluetooth.print(message_to_send);
}

```

10. Függelék ábra Arduino oldalon a küldés

```

if(val1 == 0){
    if (val3 == 0 && val4 == 0) {
        analogWrite(A1A,255);
        analogWrite(A1B,0);
        digitalWrite(B1A, LOW);
        digitalWrite(B1B, HIGH);
        delay(7000);
        digitalWrite(A1A, LOW);
        digitalWrite(A1B, LOW);
        digitalWrite(B1A, LOW);
        digitalWrite(B1B, LOW);
    }
    if (val3 == 1 && val4 == 1) {
        digitalWrite(A1A,LOW);
        digitalWrite(A1B,HIGH);
        digitalWrite(B1A, HIGH);
        digitalWrite(B1B, LOW);
        delay(700);
        digitalWrite(A1A, LOW);
        digitalWrite(A1B, LOW);
        digitalWrite(B1A, LOW);
        digitalWrite(B1B, LOW);
    }
    if (val3 == 0 && val4 == 1) {
        digitalWrite(A1A,LOW);
        digitalWrite(A1B,HIGH);
        digitalWrite(B1A, LOW);
        digitalWrite(B1B, LOW);
        delay(700);
        digitalWrite(A1A, LOW);
        digitalWrite(A1B, LOW);
        digitalWrite(B1A, LOW);
        digitalWrite(B1B, LOW);
    }
    if (val3 == 1 && val4 == 0) {
        digitalWrite(B1A, LOW);
        digitalWrite(B1B, HIGH);
        digitalWrite(A1A, LOW);
        digitalWrite(A1B, LOW);
        delay(700);
        digitalWrite(A1A, LOW);
        digitalWrite(A1B, LOW);
        digitalWrite(B1A, LOW);
        digitalWrite(B1B, LOW);
    }
}
}

```

11. Függelék ábra Manuális vezérlés

```

float temp = 0.1;
int poz = 0;
if(val1 == 1){
    float j;
    for(int i = 0; i < 4; i++){
        amg.readPixels(sensorArray);
        j = getTargetTemp();
        if(temp < j){
            temp = j;
            poz++;
        }
        startTurningLeft();
        delay(200);
    }
    for(int i = 0; i < poz; i++){
        startTurningLeft();
    }
    Serial.println(temp);
    Serial.println(poz);
    moveForward();
}
}

```

12. Függelék ábra Automata mód kódja

UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE, TÎRGU-MUREȘ
SPECIALIZAREA CALCULATOARE

Vizat decan
Conf. dr. ing. Domokos József



Vizat director departament
Ș.l. dr. ing. Szabó László Zsolt

