UNIVERSITATEA DIN CRAIOVA
FACULTATEA DE AUTOMATICĂ, CALCULATOARE ȘI ELECTRONICĂ

DEPARTAMENTUL DE AUTOMATICĂ ȘI ELECTRONICĂ

# PROIECT DE DIPLOMĂ

ALEXANDRU-PARASCHIV CORABIE

COORDONATOR ȘTIINȚIFIC

Șef lucr.dr.ing. PĂTRAȘCU-PANĂ DANIELA-MARIA

IULIE 2023

CRAIOVA

UNIVERSITATEA DIN CRAIOVA
FACULTATEA DE AUTOMATICĂ, CALCULATOARE ȘI ELECTRONICĂ

DEPARTAMENTUL DE AUTOMATICĂ ȘI ELECTRONICĂ

# MAȘINĂ CONTROLATĂ DE LA DISTANȚĂ PRINTR-UN MODUL BLUETOOTH

ALEXANDRU-PARASCHIV CORABIE

COORDONATOR ȘTIINȚIFIC

Șef lucr.dr.ing. PĂTRAȘCU-PANĂ DANIELA-MARIA

IULIE 2023

CRAIOVA

# DECLARAȚIE DE ORIGINALITATE

Subsemnatul(a) ALEXANDRU-PARASCHIV CORABIE, student la specializarea INGINERIA SISTEMELOR MULTIMEDIA din cadrul Facultății de Automatică, Calculatoare și Electronică a Universității din Craiova, certific prin prezenta că am luat la cunoştinţă de cele prezentate mai jos şi că îmi asum, în acest context, originalitatea proiectului meu de licenţă:

- cu titlul MAȘINĂ CONTROLATĂ DE LA DISTANȚĂ PRINTR-UN MODUL BLUETOOTH,
- coordonată de Șef lucr.dr.ing. PĂTRAȘCU-PANĂ DANIELA-MARIA,
- prezentată în sesiunea IULIE 2023.

La elaborarea proiectului de licenţă, se consideră plagiat una dintre următoarele acţiuni:

- reproducerea exactă a cuvintelor unui alt autor, dintr-o altă lucrare, în limba română sau prin traducere dintr-o altă limbă, dacă se omit ghilimele şi referinţa precisă,
- redarea cu alte cuvinte, reformularea prin cuvinte proprii sau rezumarea ideilor din alte lucrări, dacă nu se indică sursa bibliografică,
- prezentarea unor date experimentale obţinute sau a unor aplicaţii realizate de alţi autori fără menţionarea corectă a acestor surse,
- însuşirea totală sau parţială a unei lucrări în care regulile de mai sus sunt respectate, dar care are alt autor.

Pentru evitarea acestor situaţii neplăcute se recomandă:

- plasarea între ghilimele a citatelor directe şi indicarea referinţei într-o listă corespunzătoare la sfărşitul lucrării,
- indicarea în text a reformulării unei idei, opinii sau teorii şi corespunzător în lista de referinţe a sursei originale de la care s-a făcut preluarea,
- precizarea sursei de la care s-au preluat date experimentale, descrieri tehnice, figuri, imagini, statistici, tabele et caetera,
- precizarea referinţelor poate fi omisă dacă se folosesc informaţii sau teorii arhicunoscute, a căror paternitate este unanim cunoscută și acceptată.

Data, Semnătura candidatului(ei),

UNIVERSITATEA DIN CRAIOVA
Facultatea de Automatică, Calculatoare şi Electronică

Departamentul de Automatică şi Electronică

Aprobat la data de
……
Director de
departament,
Prof. dr. ing.
Cosmin IONETE

# PROIECTUL DE DIPLOMĂ

| | |
|---|---|
| Numele și prenumele studentului/-ei: | Corabie Alexandru-Paraschiv |
| Enunțul temei: | MAȘINĂ CONTROLATĂ DE LA DISTANȚĂ PRINTR-UN MODUL BLUETOOTH |
| Datele de pornire: | Formularea temei de proiect. Cunoștințele și abilitățile dobândite la disciplinele specializării Ingineria sistemelor multimedia precum și cunoștințe acumulate prin studiu individual. |
| Conținutul proiectului: | În cadrul lucrării sunt prezentate următoarele aspecte: Introducere, Proiectare, Limite de proiectare și direcții viitoare, Concluzii, Bibliografie, Referințe Web, Codul sursă Arduino, Codul Sursă Android Studio, CD/ DVD |
| Material grafic obligatoriu: | Grafice, capturi de ecran, scheme, diagrame bloc, imagini |
| Consultații: | Periodice |
| Conducătorul științific (titlul, nume și prenume, semnătura): | Șef lucr.dr.ing. PĂTRAȘCU-PANĂ DANIELA-MARIA |
| Data eliberării temei: | 19.01.2023 |
| Termenul estimat de predare a proiectului: | 26.06.2023 |
| Data predării proiectului de către student și semnătura acestuia: | 26.06.2023 |

UNIVERSITATEA DIN CRAIOVA
Facultatea de Automatică, Calculatoare şi Electronică

Departamentul de Automatică şi Electronică

# REFERATUL CONDUCĂTORULUI ȘTIINȚIFIC

| | |
|---|---|
| Numele și prenumele candidatului/-ei: | Corabie Alexandru-Paraschiv |
| Specializarea: | Ingineria Sistemelor Multimedia |
| Titlul proiectului: | MAȘINĂ CONTROLATĂ DE LA DISTANȚĂ PRINTR-UN MODUL BLUETOOTH |
| Locația în care s-a realizat practica de documentare (se bifează una sau mai multe din opțiunile din dreapta): | În facultate ☐<br>În producție ☐<br>În cercetare ☐<br>Altă locație: [*se detaliază*] |

În urma analizei lucrării candidatului au fost constatate următoarele:

| | | | | | |
|---|---|---|---|---|---|
| Nivelul documentării | | Insuficient ☐ | Satisfăcător ☐ | Bine ☐ | Foarte bine ☐ |
| Tipul proiectului | | Cercetare ☐ | Proiectare ☐ | Realizare practică ☐ | Altul [*se detaliază*] |
| Aparatul matematic utilizat | | Simplu ☐ | Mediu ☐ | Complex ☐ | Absent ☐ |
| Utilitate | | Contract de cercetare ☐ | Cercetare internă ☐ | Utilare ☐ | Altul [*se detaliază*] |
| Redactarea lucrării | | Insuficient ☐ | Satisfăcător ☐ | Bine ☐ | Foarte bine ☐ |
| Partea grafică, desene | | Insuficientă ☐ | Satisfăcătoare ☐ | Bună ☐ | Foarte bună ☐ |
| Realizarea practică | Contribuția autorului | Insuficientă ☐ | Satisfăcătoare ☐ | Mare ☐ | Foarte mare ☐ |
| | Complexitatea temei | Simplă ☐ | Medie ☐ | Mare ☐ | Complexă ☐ |
| | Analiza cerințelor | Insuficient ☐ | Satisfăcător ☐ | Bine ☐ | Foarte bine ☐ |
| | Arhitectura | Simplă ☐ | Medie ☐ | Mare ☐ | Complexă ☐ |

| | | | | | |
|---|---|---|---|---|---|
| | Întocmirea specificațiilor funcționale | Insuficientă ☐ | Satisfăcătoare ☐ | Bună ☐ | Foarte bună ☐ |

| | | | | | |
|---|---|---|---|---|---|
| | Implementarea | Insuficientă ☐ | Satisfăcătoare ☐ | Bună ☐ | Foarte bună ☐ |
| | Testarea | Insuficientă ☐ | Satisfăcătoare ☐ | Bună ☐ | Foarte bună ☐ |
| | Funcționarea | Da ☐ | Parțială ☐ | Nu ☐ | |
| Rezultate experimentale | | Experiment propriu ☐ | | Preluare din bibliografie ☐ | |
| Bibliografie | | Cărți | Reviste | Articole | Referințe web |
| Comentarii și observații | | | | | |

În concluzie, se propune:

| ADMITEREA PROIECTULUI | RESPINGEREA PROIECTULUI |
|---|---|
| ☐ | ☐ |

Data, Semnătura conducătorului științific,

# REZUMATUL PROIECTULUI

Lucrarea de față își propune elaborarea unui proiect pe parte de hardware și software în vederea implementării unei mașinute controlată de la distanță, cu ajutorul unui modul bluetooth. În alcătuirea părții hardware s-a folosit o plăcuță de dezvoltare din mediul Arduino (Arduino uno R 3 ce stă la baza microcontrolerului ATmega328P), motoare electrice, diverși senzori, precum și servomotor electirc compatibile cu platforma. Pe parte de software, s-a dezvoltat codul sursă care stă la baza funcționării optime a platformei de dezvoltare ARDUINO în programul "ARDUINO IDE" dezvoltat de aceștia. În ceea ce privește controlul mașini de la distanță, s-a realizat o aplicație pe telefon ce are ca scop conectarea și trimiterea de comenzi, aplicație implementată în "Android Studio". În cadrul aplicației se află diverse opțiuni de control al mașinuței (Controlul mișcării, Urmărește linia, Urmărește lumina, Mișcare automată și Urmărește ultrasonic), precum și funcții utilitare ale unei aplicații pe telefon (de exemplu Caută Dispozitiv, Conectare/ Deconectare, Locația, Acces Bluetooth șamd...).

***Termenii cheie***: arduino, uart, modul bluetooth, aplicație android, senzori, mașinuță.

# CUPRINSUL

# Listă Figuri

# Listă Tabele

# 1 INTRODUCERE

## 1.1 Scopul

Scopul lucrării de față este de a realiza o mașinuță controlată de la distanță, cu ajutorul unei aplicații Android. În sine, se dorește exemplificarea funcționalității sistemelor de asistență, implementate în automotive precum și alte funcții. Industria automotive este recunoscută mai ales pentru inovațiile tehnologice. Scopul acestei industrii este acela de a dezvolta constant performanța autovehiculelor prin adoptarea unor noi tehnologii și soluții, precum sistemele de asistență la conducere, conectivitatea mașinii, vehicule autonome, dar și alte caracteristici avansate. Conceptele folosite se aseamănă cu funcționalitățile din automotive prin următoarele exemple: *adaptive cruise-control* care se aseamănă cu funcția "Urmărește ultrasonic" prin faptul că urmărește și adaptează viteza și controlul mașinii în funcție de obiectul din față; *emergency brake assist* care se aseamănă cu "Mișcare automată" prin faptul că mașina oprește la detectarea unor obstacole, evitându-le.

În primul rând, prototipul realizat a fost testat în diverse medii și a rezultat faptul că este potrivit acestora prin construcția sa. Mașinuța de față poate să fie un bun exemplu de testare pentru implementarea diverselor funcții în industria automotive, pe lângă alte prototipuri existente deja și aflate în testare (Chakraborty et. al., 2016).

În al doilea rând, mașina implementată alături de aplicația Bluetooth poate să reprezinte un interes ridicat pentru copiii din ziua de astăzi. Este foarte bine cunoscut faptul că pasiunea pentru mașinute a copiilor este foarte crescută, dar și interesul acestora pentru dispozitivele mobile. Un alt scop al lucrării de față poate să fie introducerea (sau îmbunătățirea celor existente) pe piață a unor mașinuțe controlate cu ajutorul dispozitivelor mobile.

Pe de altă parte, această lucrare (partea teoretică, mașinuta propriu-zisă, codul sursă, dar și piesele) pot să reprezinte un bun exemplu academic pentru disciplinele realizate în cadrul facultăților de Automatică și Calculatoare. Actualii sau viitorii studenți se pot utiliza de prezentele informații în vederea procesului de învățare, de exemplu disciplina "Sisteme cu microprocesoare" sau "Aplicații multimedia pentru dispozitive mobile" (Dewi et. al., 2023).

## 1.2 Motivația

Motivul pentru care am ales această temă este dorința mea de a dezvolta hardware și software, un proiect bazat pe microcontrolere, dar și implementarea unei aplicații Android. Pe lângă cele menționate anterior, industria automotive este una dintre pasiunile mele. Prin urmare, cu ajutorul documentării și efectuarea aspectelor tehnice am reușit să îmi dezvolt deprinderi în ceea ce privește industria auto și a electrotehnicii.

Fiind interesat de industria automotive, construcția propriu-zisă a mașinuței a fost unul dintre motivele care au jucat un rol important în alegerea temei, fapt pentru care am ales diverși senzori pentru a dezvolta un prototip în miniatură cu funcționalități asemănătoare adevăratelor autoturisme. Consult în majoritatea timpului articole de specialitate din industria auto și electronicii, fiind mereu prezent și la evenimente care implică aceste teme.

Pe de altă parte, realizarea unei aplicații pe telefon de tip Android a reprezentat pentru mine un aspect important de realizat deoarece majoritatea utilităților din viața de zi cu zi și-au găsit locul în aplicații pe telefon care ne fac lumea și modul de viață mai facil. Consider că tehnologia este într-o continuă dezvoltare și vom ajunge pe viitor într-o lume în care mașinile vor deveni complet autonome, iar autovehiculele nu vor mai avea nevoie de șofer ci doar de pasageri.

Nu în ultimul rând, perioada universitară a reprezentat și ea un factor esențial în alegerea temei. Nivelul de interes pe care îl prezint la momentul actual a fost format cu ajutorul anumitor discipline care au și stat la baza realizării prezentei teme. În timpul anilor de studiu, am încercat să realizez diverse proiecte care implică componentele principale ale acestui studiu.

Pe viitor, îmi doresc să combin aceste domenii de interes care m-au făcut să aleg acest tip de lucrare prin realizarea viitoarei lucrări de disertație, dar și la un viitor loc de muncă care aș vrea să fie în domeniul EMBEDDED.

## 1.3 Generalități proiecte similare

În general, există o serie de proiecte realizate ce prezintă similarități concludente cu prezenta temă. Următoarele proiecte prezentate (ca și cel de față), prezintă aspecte ce pot să ajute la dezvoltarea tehnologiei și la crearea unei vieți de zi cu zi mult mai facilă.

În primul rând, proiectul *SELF-DRIVING CAR: USING OPENCV2 AND MACHINE LEARNING* realizat de Shoeb et. al., urmărește dezvoltarea unui prototip de vehicul autonom cu vizualizare monoculară, ce utilizează cele mai recente tehnologii OpenCV2 și Machine Learning. Modelul de mașină propus în proiectul citat este capabil să detecteze traseul benzii de circulație, semnele de circulație, semafoarele, precum și să răspundă la situațiile din trafic în timp real. Unitatea centrală de procesare este plăcuța de dezvoltare Raspberry Pi și sunt utilizate dispozitive periferice precum: plăcuță de dezvoltare Arduino UNO, L298H- Bridge și cameră Raspberry Pi Cam2 care ajută la controlarea mașinuței în modul dorit de utilizator. În concluziile proiectului, putem identifica faptul că obiectivele stabilite de la bun început au fost respectate, iar ca și obiective viitoare, se dorește implementarea unui sistem de detectare GPS. Acest studiu reprezentat un bun punct de plecare în la construcția proiectului de față, din punct de vedere hardware și software. Putem să identificăm similarități între proiectul menționat și cel de față precum:

1) Traseul benzii de circulație – proiectul de față prezintă funcția "Urmărește linia", ce funcționează pe același principiu. Se vor putea observa în capitolele viitoare ale acestui proiect mai multe detalii.

2) Răspunderea la situații în timp real – în acest proiect, mașinuța este capabilă să evite anumite obstacole cu ajutorul senzorului ultrasonic incorporat. Informații și demonstrații cu privire la acest subiect se vor găsi în capitolele viitoare.

În al doilea rând, proiectul *Smart Parking System for Monitoring Cars and Wrong Parking* propus de Alshehri et. al., (2019) se identifică încă de la început obiectivul proiectului și anume problemele legate de parcarea incorectă a autovehiculelor. Prin urmare, proiectul citat dorește să transforme cunoscuta parcare într-una inteligentă și facilă conducătorilor auto. Ca și în proiectul anterior, și acesta utilizează senzori ultrasonici, care analizează semnalul primit și îl interpretează în cod, ulterior trimițând octeți, utilizând protocolul de comunicare UART printr-un transmițător pe frecvență radio prezent pe cele două plăcuțe Arduino. Placa de dezvoltare mobilă va transmite rezultatul utilizând un ecran tactil Nextion, care va indica starea parcării. În urma testării experimentale, s-a constatat că sistemul implementat este prietenos cu utilizatorii, dar și faptul că a prezentat o acuratețe de 95%. Restul de 5% reprezintă erori cauzate de anumite limite (dacă modelul fizic era mai mare și/sau obiectele detectate erau mai mari, acuratețea putea să fie mai ridicată, undele ultrasonice pot fi

împrăștiate de suprafețele curbate, de exemplu bările mașinilor, fapt pentru care înregistrarea valorilor senzorilor pot fi eronate în unele momente). Ca și similaritate putem identifica faptul că senzorii ultrasonici care în proiectul dezvoltat de Alshehri și colaboratorii săi are rolul de a detecta locurile de parcare disponibile. Este de la sine înțeles faptul că atât în proiectul citat, cât și în cel de față, senzorii ultrasonici au rolul indirect de a detecta obstacolele/ obiectele.

De asemenea, putem să vorbim și despre *Line Follower Robot Arduino (using robot to control Patient bed who was infected with Covid-19 Virus)* realizat de Hadi (2020). Este de menționat faptul că acest model a fost realizat în perioadă pandemică, cu scopul de a evita infectarea cadrelor medicale. Principalul rol pe care acest robot îl are este de a transporta pacienții diagnosticați cu COVID-19 din ambulanțe în camerele specializate din cadrul spitalelor. Robotul (mașinuța) s-a aflat în fază experimentală (de testare), fiind construit în miniatură. Principiul tehnic evidențiat în proiectul citat îl reprezintă urmărirea unui traseu bine-stabilit, cu ajutorul senzorilor cu infraroșu. Acest principiu este asemănător cu cel al proiectului de față, la fel ca și celelalte proiecte menționate. S-a scos în evidență o limită de funcționare care se referă la faptul că senzorii IR sunt afectați de lumina solară. La momentul publicării acestui proiect, atunci când robotul era expus luminii solare, acesta funcționa continuu și non-stop. Aspectul identificat de Hadi reprezintă, de asemenea și o limită a acestui proiect despre care vom vorbi la capitolul “Limite”.

Concluzionând, acest capitol al generalităților proiectelor asemănătoare a evidențiat importanța proiectelor similare și anumite beneficii pe care proiectele menționate le pot aduce sau le-au adus și în alte domenii (spre exemplu domeniul medical în cadrul studiului menționat anterior). De asemenea, există o nevoie constantă de cercetare în domeniu iar acest proiect și partea sa de documentare pot să reprezinte anumite plusuri în ceea ce privește domeniul tehnic. Proiectele menționate anterior au reprezentat doar o mică parte dintre cele ce au fost analizate, în vederea implementării și succesului actualului proiect. În următoarele capitole ale acestei teme, vom identifica și vom vorbi despre o varietate de funcții pe care le-am testat și le-am proiectat în cadrul mașinuței controlate de la distanță cu ajutorul unui modul Bluetooth.

# 2 PROIECTARE

## 2.1 Proiectare mașină

### 2.1.0 Mediu de dezvoltare Arduino

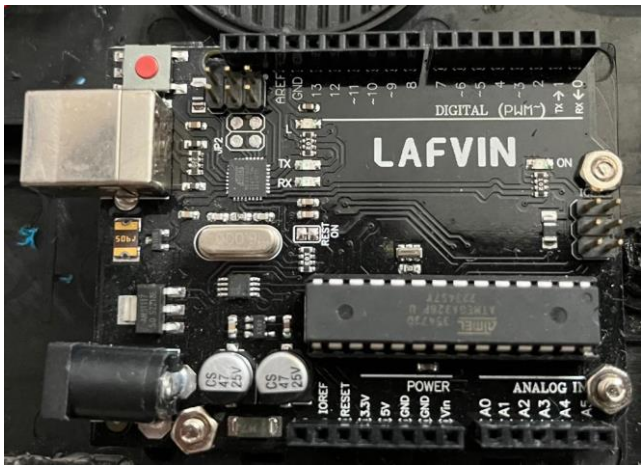
Conform prezentării, Arduino este o companie *open-source hardware și software,* ce are ca scop designul și manufacturarea plăcilor de dezvoltare din seria Arduino și a *microcontrollerelor,* în vederea construirii unor dispozitive digitale. Plăcuțele Arduino folosesc o varietate de microprocesoare și controllere, ce sunt echipate cu pini care pot să fie conectați cu așa-numitele plăcuțe de extensie sau *expansion boards* sau *shields.* Microcontrollerele pot să fie programate cu ajutorul limbajelor de programare C sau C++, utilizând un *API* standard cunoscut ca și "Limbajul de programare ARDUINO".

Mediul de dezvoltare integrat ARDUINO sau *The Arduino integrated development environment (Arduino IDE)* este o platformă specializată ce este scrisă în limbajul de programare Java. Arduino IDE suportă limbaje de programare precum C sau C++, utlizând totuși anumite reguli de structurare a codului. Putem considera faptul că aplicația Arduino IDE este una facilă, întrucât conține o varietate de *input-uri* și *output-uri* comune. [1]

În această parte, s-a realizat o scurtă prezentare a mediului de dezvoltare Arduino. Putem să identificăm utilitatea lui încă de la început și de asemenea, putem observa printr-o scurtă căutare în literatura de specialitate că mediul de dezvoltare Arduino nu mai este o necunoscută și este chiar predat și utilizat în mediul academic și nu numai.

### 2.1.1 Hardware

#### a) Arduino UNO R3




**Figură 1.** Plăcuța Arduino UNO R3

Plăcuța de dezvoltare care inițial a fost lansată în anul 2010, este echipată cu un set de pini de intrare și ieșire analogici și digitali, pini care pot fi conectați la o serie de plăci de extensie și alte circuite.

Plăcuța are 14 intrări/ ieșiri de pini digitali, șase dintre aceștia sunt capabili de PWM sau *pulse-width-modulation* sau modularea lățimii impulsului (pinii 3,5,6,9,10 și 11), șase intrări/ ieșiri analogice. Aceasta poate să fie alimentată cu 5V și programată (în mediul integrat de dezvoltare Arduino IDE) prin intermediul unui cablu tip USB-B, conectat la portul respectiv. În același timp, poate să fie alimentată printr-un conector cilindric, care acceptă voltaj între 7-20V.

Când vine vorba despre anumite particularități, plăcuța prezintă un LED construit pe aceasta, ce este legat de pin-ul 13. Atunci când valoarea pin-ului 13 este *HIGH,* LED-ul este aprins, iar atunci când valoarea este *LOW,* pin-ul este stins.

O altă particularitate sunt ieșirile de 5V și de 3,3V de pe plăcuță, care furnizează voltajul specificat datorită unui regulator de tensiune, ieșire de *ground* sau masă și un buton fizic de reset care resetează funcționarea plăcuței.

Există anumiți pini care prezintă funcții speciale. Comunicare serială UART prin intermediul pinilor D0 (RX sau pin receptor) și D1 (TX sau pin transmițător) pentru a primi și transmite date seriale TTL. Comunicarea serială UART are ca și caracteristică arhitectura *single-master-single-slave*. Acești pini sunt conectați la pini corespunzători unuia dintre microcontrolere, Atmega8U2 pentru conexiune serială USB-TTL. Funcții de întrerupere specifici pinilor 2 și 3, care pot realiza întreruperea, fiind setați

inițial *LOW*, printr-o schimbare a valorii. Funcția PWM de pe pinii menționați mai sus pot furniza o valoare de ieșire pe 8 biți. SPI sau interfață periferică serială, integrată pe pinii 10, 11, 12 și 13, furnizează o comunicare serială sincronă pe distanță scurtă folosită în principal în sistemele *embedded*, ce este o comunicație bazată pe arhitectura *single-master-multi-slave*. Această funcție în cazul utilizării SPI este disponibilă în librăria acestuia. Pe pinii A4 și A5 apare protocolul de comunicare $I^2C$, comunicare sincronă *multi-master-multi-slave* funcție numită și TWI *(two-wire-interface)*. Pinul AREF este folosit pentru a măsura corect referința de voltaj în cazul în care se folosesc senzori pe intrări analogice, pentru a verifica voltajul exact.

Microcontrolerul ce stă la baza plăcuței Arduino UNO R3, ATMega328, vine pre-programat cu un *bootloader* care permite încărcarea unei noi secvențe de cod fără să fie folosit un hardware extern cu scopul de *memory flush* sau eliberarea de memorie. [2]

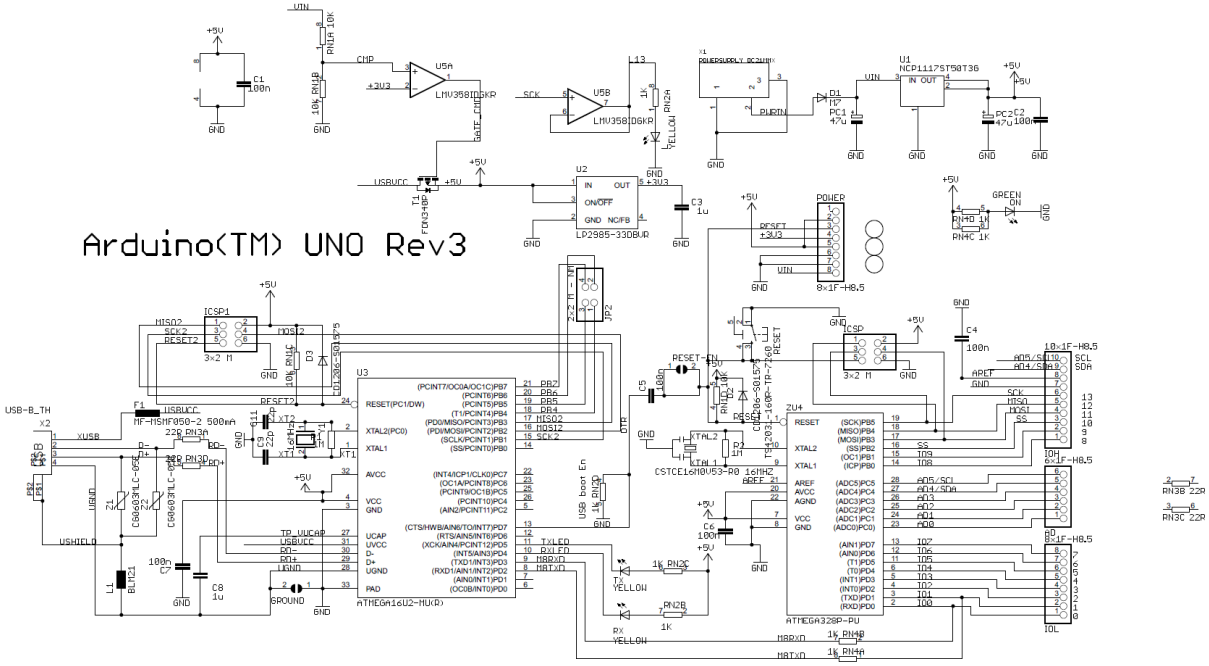
Conform descrierii din fișa tehnică, Arduino UNO R3 este un microcontroler versatil, care este echipat cu procesoarele Atmega328P și ATMega 16U2. Funcționalitățile și descrierea lor se pot observa în tabelele de mai jos (Tabel 1 și Tabel 2) [3] [4]

| Funcționalitate | Descriere |
|---|---|
| 1) Memorie | • CPU până la 16 MHz<br>• 32 KB flash<br>• 2KB SRAM<br>• 1KB EEPROM |
| 2) Securitate | • Resetare la pornire<br>• Detectare uzură |
| 3) Periferice | • 2x Timer/Contor de 8 biți cu un registru dedicat perioadei și compararea canalelor<br>• 1x Timer/Contor pe 16 biți cu un registru dedicat perioadei, captură de intrare și canale de comparare<br>• 1x USART cu generator fracționar de viteză de transmisie și detecție la începutul cadrului de date<br>• 1x controler/periferic *Serial Peripheral Interface* (SPI)<br>• 1x controler cu mod dublu/periferic I2C<br>• 1x comparator analogic (AC) cu o intrare de referință scalabilă<br>• Timer Watchdog cu oscilator separat pe cip<br>• Șase canale PWM<br>• Întrerupere și trezire la schimbarea valorii logice a pinului |

**Tabel 1** Funcționalități ale procesorului ATMega328P

| Funcționalitate | Descriere |
|---|---|
| 1) Memorie | • 16 KB ISP Flash<br>• 512B EEPROM<br>• 512B SRAM<br>• interfață debugWIRE pentru depanare și programare pe cip |
| 2) Putere | • 2.7-5.5 volți |

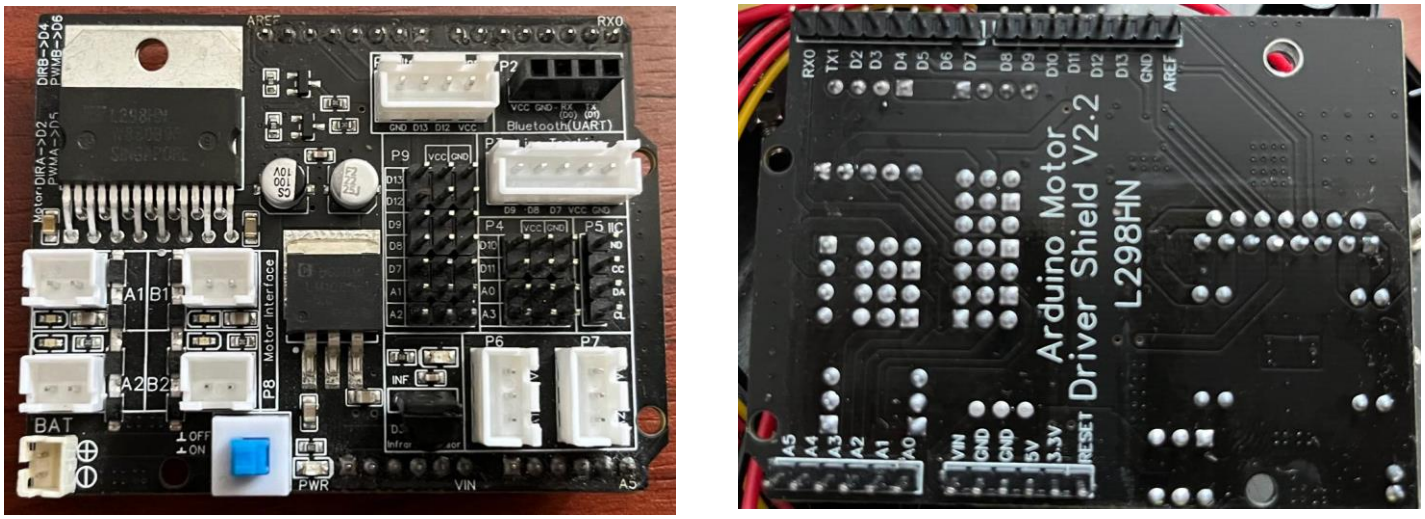
**Tabel 2** Funcționalitățile procesorului ATMega16U2




**Figură 2** Diagramă bloc Arduino UNO R3

### b) Plăcuță de extensie Motor Driver Shield V2.2

Această placă de extensie Motor Driver Shield V2.2 este dedicată plăcuței de dezvoltare Arduino UNO R3 și are un rol important deoarece integrează pe aceasta o punte H. Puntea H folosită de acesta este cipul electronic L298HN Multiwatt-15 (discutat în cadrul secțiunii următoare). Aspectul propriu-zis al acestui *shield* se poate observa în figura 3 . Secțiunile A1, B1, A2 și B2 fac referire la puntea A (motoarele din stânga) și puntea B (motoarele din dreapta). În stânga sus se poate observa și pinout-ul *default* al ieșirilor digitale ale motoarelor. Am decis să aleg acest shield deoarece mi-a validat ideea reducerii numărului de porturi digitale folosite, idee detaliată la punctul următor.

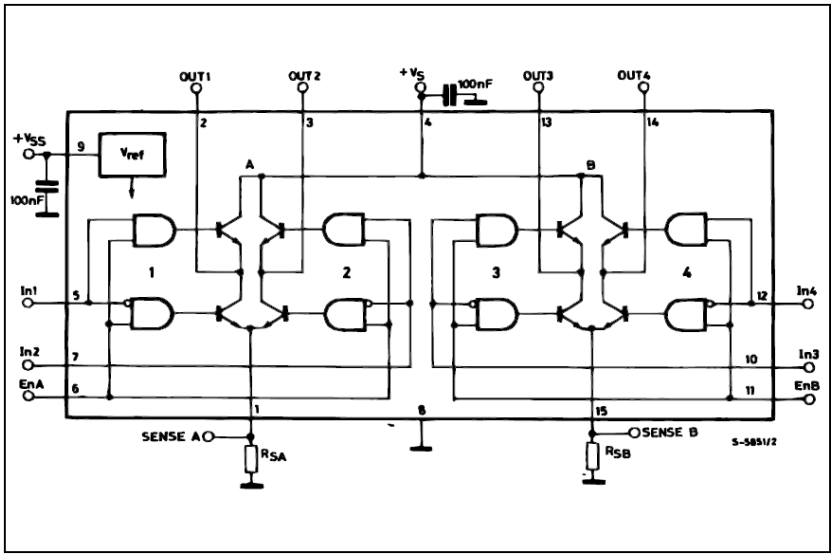



**Figură 3** Plăcuță de extensie față/ spate

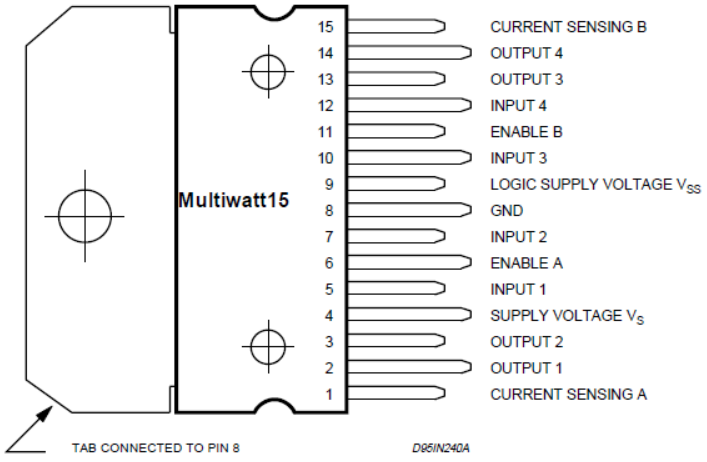
Punctul forte al plăcii de expansiune este acela că face foarte facilă conexiunea componentelor cu mediul de dezvoltare, atât prin puntea H furnizată cât și prin design-ul funcțional. Aceasta integrează conexiuni prin mufe xh (cele albe), cu până la 4 pini, cât și prin fire de salt (fire dupont). O caracteristică de construcție ergonomică este amplasarea alimentării VCC și masei GND paraleli pentru fiecare pini de conexiune analog și digital aflat în rubricile P9 și P4, cât și pentru socket-ul de coneciune UART (P2) unde modulul bluetooth poate fi conectat fără adaptări suplimentare. Butonul albastru capătă funcționalitatea de pornire/ oprire a alimentării.

**L298HN Multiwatt-15 – punte H**

Conform descrierilor din fișa tehnică, L298 este un chip electronic integrat într-un singur circuit, având 15 pini și pachete de tip Multiwatt și PowerSO20. Este proiectat să funcționeze la tensiuni mari și să gestioneze curenți mari. Este un driver special conceput pentru a primi semnale logice de nivel TTL și pentru a controla dispozitive inductive, cum ar fi releele, solenoizii și motoarele de curent continuu. Are două intrări de activare care permit pornirea sau oprirea dispozitivului independent de semnalele de intrare. De asemenea, permite conectarea unui rezistor de detecție extern pentru a măsura și monitoriza curentul. De asemenea, are o intrare suplimentară pentru a alimenta componentele logice cu o tensiune mai mică. Vom putea observa și diagrama bloc (figura 2). [5]




**Figură 4** Schemă bloc punte H




**Figură 5** Schemă pini L298HN

| Multiwatt 15 | Nume Pini | Funcție |
|---|---|---|
| 1; 15 | Sens A; Sens B (*Sense A; Sense B)* | Între acest pin și masă este conectat rezistorul de detecție pentru a controla curentul sarcinii. |
| 2;3 | Ieșire 1; Ieșire 2 *(Out 1; Out 2)* | Ieșirile punții A; curentul care trece prin sarcina conectată între acești doi pini este monitorizat la pinul 1. |
| 4 | VS | Tensiunea de alimentare pentru etapele de ieșire de putere.<br>Un condensator non-inductiv de 100nF trebuie conectat între acest pin și masă (ground). |
| 5;7 | Intrare 1; Intrare 2 *(Input 1; Input 2)* | Intrări compatibile cu nivelurile logice TTL ale punții A. |
| 6;11 | Activare A; Activare B *(Enable A; Enable B)* | Intrarea de activare compatibilă cu nivelurile logice TTL: starea L dezactivează puntea A (activare A) și/sau puntea B (activare B). |
| 8 | GND | Masă |
| 9 | VSS | Tensiunea de alimentare pentru blocurile logice. Un condensator de 100nF trebuie conectat între acest pin și masă (ground). |
| 10;12 | Intrare 3; Intrare 4 *(Input 3; Input 4)* | Intrări compatibile cu nivelurile logice TTL ale punții B. |
| 13;14 | Ieșire 3; Ieșire 4 *(Out 3; Out 4)* | Ieșirile punții B. Curentul care trece prin sarcina conectată între acești doi pini este monitorizat la pinul 15. |

**Tabel 3 Descriere pini Multiwatt 15**

L298HN Multiwatt-15 este puntea H folosită în proiectul curent cu scopul de a utiliza și controla patru motoare cu alimentare de curent continuu. Rolul punții H este acela de a oferii directia de rotație a motoarelor și controlul precis al vitezei rezultat prin utilizarea modulării în lățime de impuls *(PWM)* oferit de placa de dezvoltare Arduino. Potrivit fișei tehnice pentru a stabilii viteza se folosesc pinii Activare A/B de pe L298HN , iar pentru direcția motoarelor se folosesc pinii descrisi tabelul 4. [6]

| Intrare1/ Intrare3 | Intrare2/ Intrare4 | Starea motorului A/B |
|---|---|---|
| 0 | 0 | Oprire |
| 0 | 1 | Rotație în sensul acelor de ceasornic |
| 1 | 0 | Rotație în sens invers acelor de ceasornic |
| 1 | 1 | Oprire |

**Tabel 4 Nivel logic pentru controlul direcției motoarelor**

Luând în cosiderare că prezentul proiect va avea în componență o varientate și un numar mai mare de senzori, iar numarul porturilor de intrare/iesire al Arduino Uno R3 este limitat am luat decizia să folosesc o abordare mai diferită și rezervată în ceea ce privește utilizarea porturilor.

Într-un scenariu favorabil în care numărul porturilor este suficient pentru a conecta 4 motoare ( două pe Activare A și Intrare 1/2 , respectiv două pe Activare B și Intrare 3/4 ) și nu am dorii să avem îmbunatățiri pe viitor ( păstrând un număr mai mare de pini disponibili ) putem folosi 6 intrări/ieșiri digitale luând în considerare că cel puțin 2 dintre acestea trebuie să aibă caracteristica modulării în lățime de impuls *(PWM)*. [7]

În prezentul proiect am luat decizia de a reduce numărul porturilor utilizate la doar 4 porturi digitale în loc de 6. Pinii ce vor servi la controlul vitezei motoarelor vor rămâne tot în număr de 2 și aceștia vor fi : pinul digital 5 pentru motorul stâng al mașinuței, respectiv pinul digital 6 pentru motorul drept al mașinuței. Vor fi setați ca și pini de ieșire și bineînțeles va fi folosită caracteristica lor specială de *PWM* ( ce cuprinde valori între 0 si 255, 0 fiind oprit , iar 255 fiid viteza maximă de rotație a motoarelor). Optând pentru *PWM* egal 0 pe ambele punți (A si B) putem executa oprirea motoarelor fără a fi nevoie ca acestea să fie oprite la nivel logic ( ambele intrări de pe puntea A si B să fie în același timp 0 sau 1). Astfel mai gestionăm doar situația în care nivelele logice de pe puntea A, respectiv B sunt diferite pe fiecare intrare. Pentru acest lucru avem nevoie de două ieșiri digitale din Arduino, pinul digital 2 și pinul digital 4.

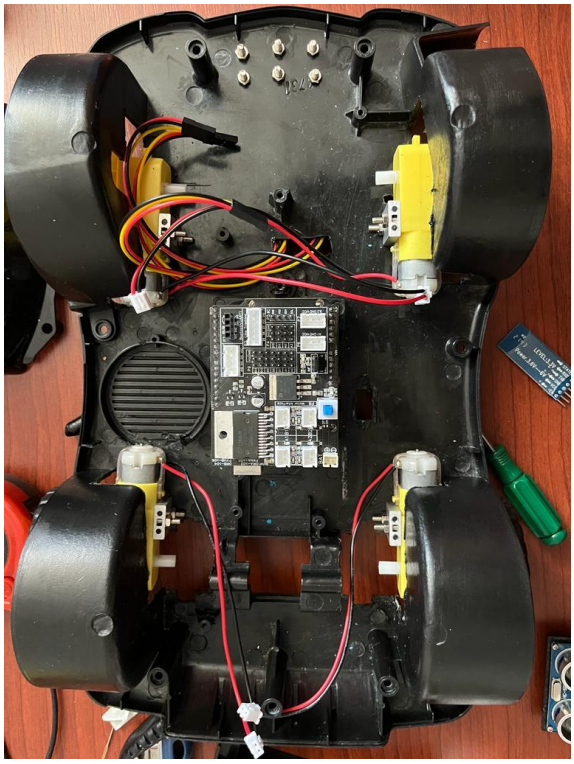
Pinii vor fii legați la puntea H în felul urmator: pinul digital 2 ( D2 ) la Intrarea 2 ( a punții A ) și Intrarea 4 ( a puții B ), iar pinul digital 4 ( D4 ) la Intrarea 1 ( a punții A ) și Intrarea 3 ( a puții B ).

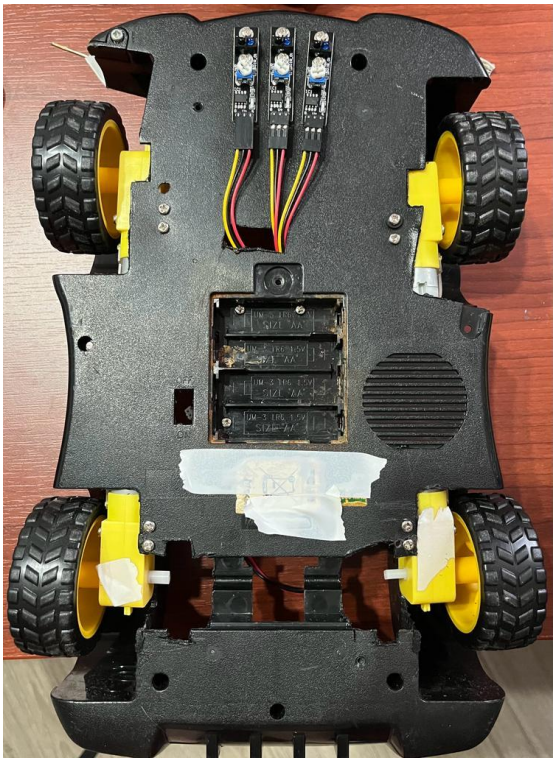
### c) Motor TT (DC Gearbox Motor) și roți

**Figură 6** Motor TT și roți

Motorul ales pentru proiectul prezent este compus dintr-un motor DC care este conectat la un sistem mecanic de roți zimțate asemănător conceptului de cutie de viteze, care are rolul de a transforma viteza motorului în cuplu. Rația produsului prin adăugarea roților zimțate este de 1:48, adică motorul DC din interior va avea nevoie de 48 de rotații pentru a rotii o singură dată o roată. Acest lucru este util în aplicația prezentă deoarece creșterea în cuplu ajută la deplasarea mașinii în situația în care aceasta pleacă de pe loc. Motorul se alimentează de la 3V (voltaj minim)-6V (voltaj maxim) și atinge viteza de 200 de rotații pe minut la voltaj maxim. [8]

Acest model a fost ales deoarece necesită o tensiune de alimentare relativ mică și prezintă o putere de deplasare potrivită în raport cu cerințele mașinuței (cerințele de greutate și mărime). De asemenea, amplasarea și poziționarea lor este recomandat să fie făcută cât mai mult înspre extremitățile suportului inferior pe care sunt așezate pentru a facilita o eficiență cât mai bună la deplasare și întoarcere (de văzut figura 7). Tot în ceea ce privește poziționarea motoarelor, acest model prezintă un montaj prietenos deoarece în componența sa identificăm orificii pentru șuruburi în vederea plasării pe macheta specifică a mașinuței.





**Figură 7** Poziționare motoare

| Detalii tehnice | |
|---|---|
| 1. Curent continuu fără sarcină | 150mA +/- 10% |
| 2. Viteză minimă de funcționare | (3V): 90+/- 10% RPM |
| 3. Viteză minimă de funcționare | (6V): 200+/- 10% RPM |
| 4. Cuplu | 0.15Nm ~0.60Nm |
| 5. Cuplu maxim | (6V): 0.8kg.cm |
| 6. Raport de transmisie | 1:48 |
| 7. Dimensiuni corp | 70 x 22 x 18mm |
| 8. Greutate | 30.6g |

**Tabel 5 Specificații tehnice Motor TT**

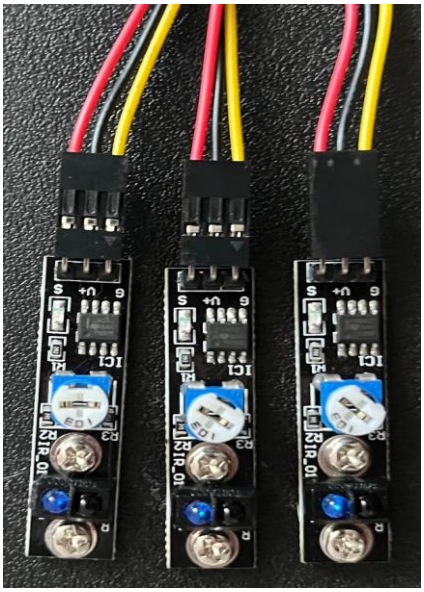
### d) Senzori infraroșu KY-033

Senzorii cu infraroșu sunt folosiți în industrie cu o varietate de scopuri ca de exemplu controlul de la distanță al televizoarelor care cu ajutorul senzorilor cu infraroșu decodează semnalul transmis de telecomenzi. Senzorii cu infraroșu pot detecta prezența, mișcarea, cât și temperatura unui obiect.

Principiul de funcționare al senzorilor cu infraroșu este următorul: senzorul prezintă un transmițător și un receptor infraroșu, transmițătorul este un LED infraroșu și receptorul este o fotodiodă cu infraroșu. Fotodioda este activată de lumina ledului cu infraroșu, iar rezistența fotodiodei se schimbă împreună cu ieșirea de voltaj de pe aceasta care sunt proporționale cu cantitatea de lumină cu infraroșu captată de fotodiodă. Odată ce dioda transmite emisii cu infraroșu și acestea ajung la suprafața unui obiect, o parte dintre acestea se reflectă înapoi către fotorezistor. Astfel, ieșirea de semnal a fotorezistorului oscilează în funcție de prezența și distanța obiectului față de care se află. [9] [10]

KY-033 este senzorul cu infraroșu ales pentru proiectul de față, acesta încadrându-se în tipul de senzori activi, adică acei senzori care trimit și recepționează semnal în spectru infraroșu. Senzorul poate măsura pe ieșire semnal analogic sau digital.

În proiectul actual, vom utiliza capacitatea lui de a emite semnal digital. Scopul lui este pentru funcția de *line-tracking* sau urmărire a liniei, acesta întorcând valoarea logică 0 când linia este detectată și 1 atunci când acesta nu se află deasupra liniei. Pentru o exemplificare eficientă, se va utiliza o linie neagră pe o suprafață cât mai deschisă la culoare deoarece negru nu reflectă la fel de mult lumina. Valoarea ieșirii digitale este rezultată din comparația nivelelor de lumină și poate să fie ajustată tensiunea de prag sau *threshold voltage* prin rotirea potențiometrului aflat pe senzor. Acest lucru se va regăsi pe larg în secțiunea de funcționalitate a mașinii.



**Figură 8** Senzori infraroșu KY-033

| Detalii tehnice | |
|---|---|
| 1. Tensiune de lucru | DC 3.3V-5V |
| 2. Curent de lucru | ≥ 20mA |
| 3. Temperatura de operare | -10℃~+50℃ |
| 4. Distanță de detectare | 2-40cm |
| 5. Interfață IO | Interfețe cu 4 fire (-/+/S/EN) |
| 6. Semnal de ieșire | Nivel TTL (nivel scăzut în caz de obstacol, nivel înalt în absența unui obstacol) |
| 7. Ajustare | rezistență cu mai multe înfășurări reglabilă |
| 8. Unghi eficient | 35° |
| 9. Dimensiuni | 28mm×23mm |

**Tabel 6 Specificații tehnice KY-033**

### e) Senzor infraroșu REES52 IR

Senzorul cu infraroșu REES52 IR a fost ales în prezentul proiect cu scopul de a detecta prezența sau absența obiectelor, fiind utilizat în funcția de mișcare automată și de urmărire. Ca și principii de funcționare, este similar cu senzorul anterior dar prin construcția sa este mai facilă folosirea lui pentru funcțile menționate. La fel ca KY-033, este prezent pe corpul acestuia un potențiometru prin care se poate seta voltajul de prag. Este echipat cu un LED verde pe carcasă care oferă un răspuns cu privire la detecția obiectelor. Când ledul se aprinde, senzorul își schimbă valoarea logică. [11]

**Figură 9** Senzor REES52 IR

| Detalii tehnice | |
|---|---|
| 1. Tensiune de lucru | DC 3.3V-5V |
| 2. Comparator | LM393 |
| 3. Distanță de detectare | 2-30cm |
| 4. Interfață IO | Interfețe cu 3 fire (-/+/S) |
| 5. Semnal de ieșire | Nivel TTL (nivel scăzut în caz de obstacol, nivel înalt în absența unui obstacol) |

| | |
|---|---|
| 6. Ajustare | rezistență reglabilă cu mai multe înfășurări |
| 7. Unghi eficient | 35° |
| 8. Dimensiuni | 32mm×14mm |

**Tabel 7** Specificații tehnice REES52 IR

### f) Senzor ultrasonic HC-SR04

Senzorul ultrasonic HC-SR04 este un senzor care în proiectul de față are ca scop măsurarea distanței obiectelor din fața mașinuței, acesta fiind amplasat pe capotă. Principiul lui de funcționare este unul similar liliecilor sau delfinilor, senzorul HC-SR04 utilizând într-un fel principiul de ecolocație pe care animalele menționate anterior îl au în sistemul biologic. Senzorul transmite o undă de sunet cu ajutorul pinului său logic *echo*, care trebuie setat ca 1 logic cel puțin 10 microsecunde, iar apoi setat 0 logic. În momentul în care se trimite unda de sunet, celălalt pin, pinul *trigger* așteaptă ca unda de sunet propagată să se lovească de un obiect și să se întoarcă înapoi, fiind interceptată de acesta. Măsurarea distanței de către senzor se face în microsecunde. Trebuie luat în considerare că unda de sunet parcurge mereu dublul distanței deoarece pleacă și se întoarce înapoi la senzor. Dacă dorim să transformăm în centimetrii, vom respecta următoarea formulă: Distanța (în cm) = (Timpul de propagare x Viteza sunetului) / (2 x 100). Formula simplificată pentru calcularea distanței ar trebui să fie valoarea măsurată în microsecunde împărțită la 58. [12]


**Figură 10** Senzor ultrasonic HC-SR04

| Detalii tehnice | |
|---|---|
| 1. Tensiune de lucru | DC 5V |
| 2. Curent de lucru | 15mA |
| 3. Frecvență de lucru | 40Hz |
| 4. Distanță de detectare | 2cm-4m |
| 5. Unghi de măsurare | 15° |
| 6. Semnal de intrare *trigger* | 10uS puls TTL |
| 7. Semnal de ieșire *echo* | Semnal de intrare TTL și raport de scală |
| 8. Dimensiuni | 45mm×20mm×15mm |

**Tabel 8** Specificații tehnice HC-SR04

### g) Micro Servo SG90

Micro servoul SG90 este un servo de dimensiuni mici care dezvoltă o putere de ieșire mare. Acest servo se poate rotii la maxim 180°, 90° în fiecare direcție, cu toate că nu este recomandat să îl rotim la unghiurile sale maxime (0° sau 180°) deoarece se poate deteriora în timp. În interiorul său se află un motor și cu ajutorul unui mecanism de roți, la fel ca și la motoarele TT (vezi mai sus), acesta câștigă cuplu. Pinout-ul său conține alimentarea, masa și o intrare pentru semnal PWM, oferită de plăcuța de dezvoltare Arduino. [13]

În acest proiect este utlizat împreună cu senzorul HC-SR04 (vezi detalii mai sus), pentru a-l rotii pe axa y. Acest lucru facilitează măsurarea distanței împrejurimilor prin schimbarea direcției de măsurare oferită de direcția de rotație. Datorită montajului, adică senzorul HC-SR04 este poziționat pe micro servo, aceste două componente se rotesc împreună. La finalizarea procesului de rotație, se măsoară distanța cu senzorul ultrasonic.

Arduino IDE suportă o librărie pentru controlul precis al servoului, numită și" <Servo.h>" prin care se stabilește valoarea unghiului de rotație (în acest caz de la 0° până la 180°). [14]

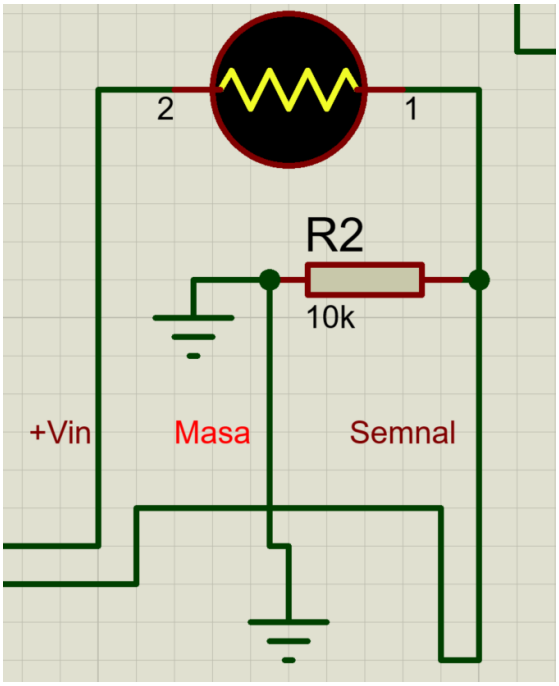

**Figură 11** Micro Servo SG90

| Detalii tehnice | |
|---|---|
| 1. Tensiune de lucru | DC 4.8-5V |
| 2. Temperatură de lucru | 0°-55°C |
| 3. Lățimea benzii moarte | 10 µs |
| 4. Viteză de operare | 0.1s/ 60 de grade de rotație |
| 5. Cuplu de blocaj | 1,8 kgf·cm |
| 6. Greutate | 9g |
| 7. Dimensiuni | 22.2mm×11.8mm×31mm |

**Tabel 9** Specificații tehnice SG90

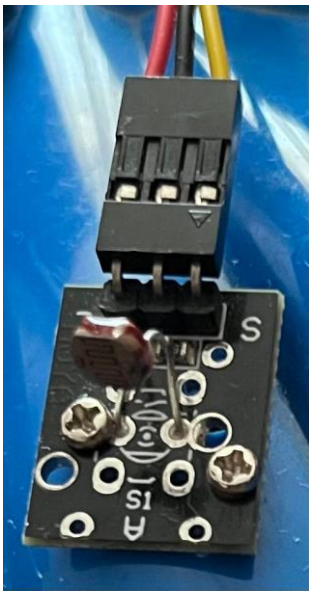
### h) Senzor fotorezistiv KY-018

Senzorul KY-018 are în componența sa un rezistor LDR (*Light dependent resistor)* sau o rezistență dependentă de lumină care își modifică valoarea în funcție de intensitatea luminoasă aplicată pe aceasta. Principiul de funcționare al acestui senzor este că la rezistența dependentă de lumină, se leagă în circuit un divizor de tensiune de 10kOhm. Putem observa schema bloc în cele ce urmează (figura 12) realizat în Proteus 8 Professional. [15]




**Figură 12** Schemă electrică KY-018

Senzorul prezintă 3 pini, semnal, masă și alimentare (Vin). Pinul de semnal are capacitatea de a oferi semnal analogic la conectarea sa cu plăcuța de dezvoltare Arduino (pe una dintre intrările analogice). Pinul de masă trebuie să fie comun cu cel al plăcuței, iar alimentarea este furnizată de acesta. [16]




**Figură 13** Senzor KY-018

| Detalii tehnice | |
|---|---|
| 1. Semnal de ieșire | Analog |
| 2. Temperatură de lucru | -30°până la 70+°C |
| 3. Tensiune de operare | 3.3V-5V |
| 4. Rezistența în întuneric sau *Dark resistence* | 500kΩ |
| 5. Timp de răspuns la detectarea luminii | 30 μs |
| 6. Greutate | 2g |
| 7. Dimensiuni | 30mm×15mm×6mm |

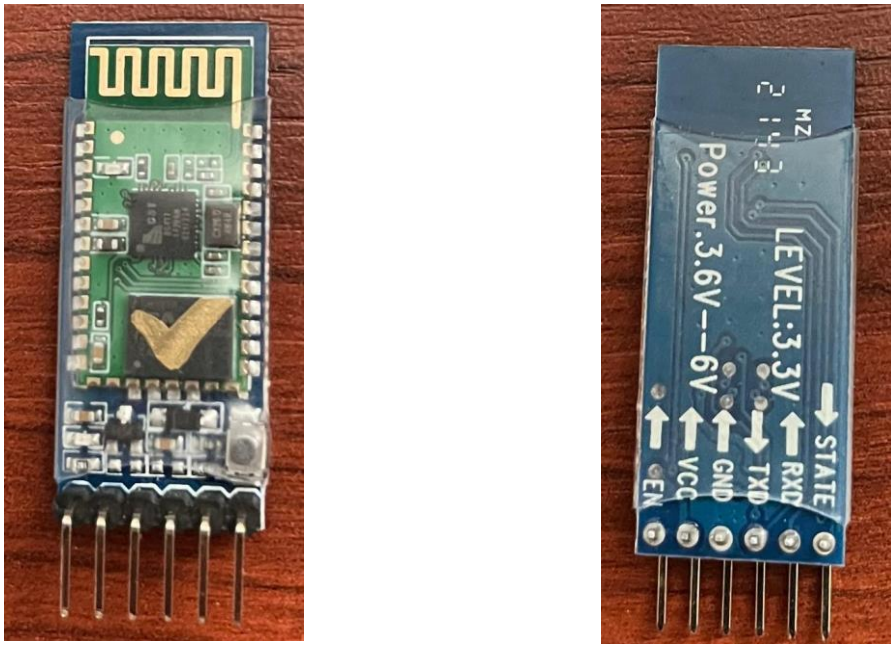
**Tabel 10** Specificații tehnice KY-018

### i) Modul bluetooth clasic HC-05

Modulul bluetooth HC-05 ales în vederea realizării proiectului este un modul de bluetooth clasic ce are ca și caracteristică principală capabilitatea de a fi atât *master* cât și *slave*, comparat cu modulul bluetooth HC-06 care poate opera decât ca slave. Modulul HC-05 vine setat din fabrică în stadiul de master. Acesta are caracteristica de UART (*Universal Asyncron Reciver Transmiter*), ceea ce înseamnă că realizează o comunicare serială când este legat la Arduino [17]. Se leagă la plăcuța de dezvoltare Arduino prin pinii digitali 0 și 1 (TX și RX), la alimentare și la masă comună. Procesul de împerechere este simplu, de exemplu cu telefonul. Pe telefon se pornește funcția de bluetooth și se scanează dispozitivele din apropiere. În momentul în care acesta este detectat, se poate realiza împerecherea după introducerea parolei.

Pentru o mai bună înțelegere a modului de funcționare, se vor descrie pinii și funcționalitatea lor. Pinul *key*/ *enable* este folosit pentru a aduce modulul bluetooth în modul de comenzi AT, deși îl putem duce în acest mod conectând pinii modulului bluetooth la alte porturi digitale. În acest mod în care modulul răspunde la comenzile AT, ii se pot modifica toate setările acestuia (exemplu nume, parolă, trecerea din master în slave, *bit-rate* sau *baud-rate* în care ii se modifică viteza de transfer șamd...). Pinul VCC este pin de alimentare al modulului. GND este pinul de masă și se leagă la masă comună cu plăcuța de dezvoltare. Pinul TX al modulului bluetooth se leagă la pinul RX al

microcontrolerului, iar pinul RX al modulului bluetooth se leagă la pinul TX al microcontrolerului. Pinul STATE poate fi conectat sau nu și servește ca și indicator de status în vederea verificării conexiunii dintre modul și un dispozitiv. [18]



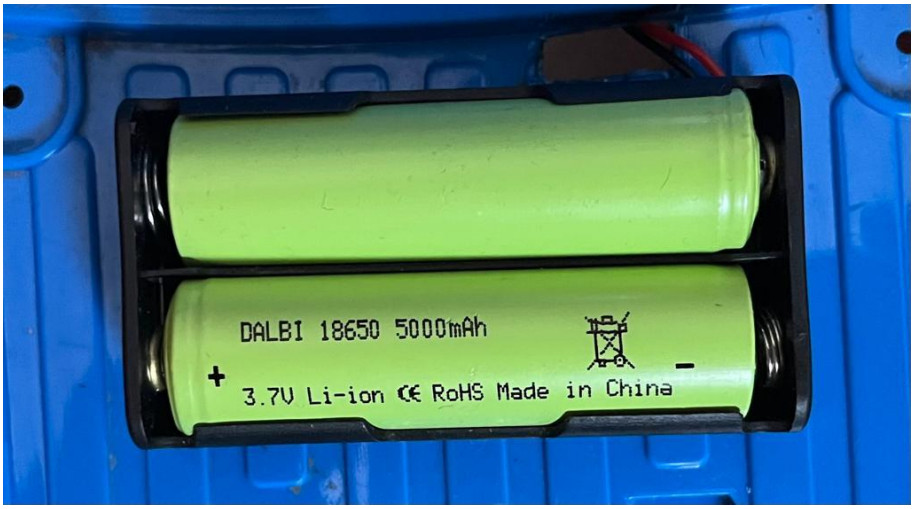

**Figură 14** Modul bluetooth HC-05 față/ spate

| Detalii tehnice | |
|---|---|
| 1. Protocol Bluetooth | v2.0+EDR (*enhanced data rate*) |
| 2. Frecvență | 2.4GHz |
| 3. Tensiune de alimentare | 3.6V-6V |
| 4. *Baud rate* sau frecvență de transmitere biți/s | 9600 |
| 5. Putere de transmisie a semnalului radiofrecvență | +4dBm |
| 6. Interfață de conectare | UART |
| 7. Dimensiuni | 12.7mm×27mm |

**Tabel 11** Specificații tehnice HC-05

### j) Baterii litium-ion 18650

În cadrul proiectului se folosesc două baterii 18650 de 3.6V, legate în serie ca sursă de alimentare. Numele bateriilor revine din specificațiile lor tehnice de mărime 18mm×65mm. S-au ales aceste baterii deoarece oferă alimentarea suficientă a întregului ansamblu de senzori și componente electronice.

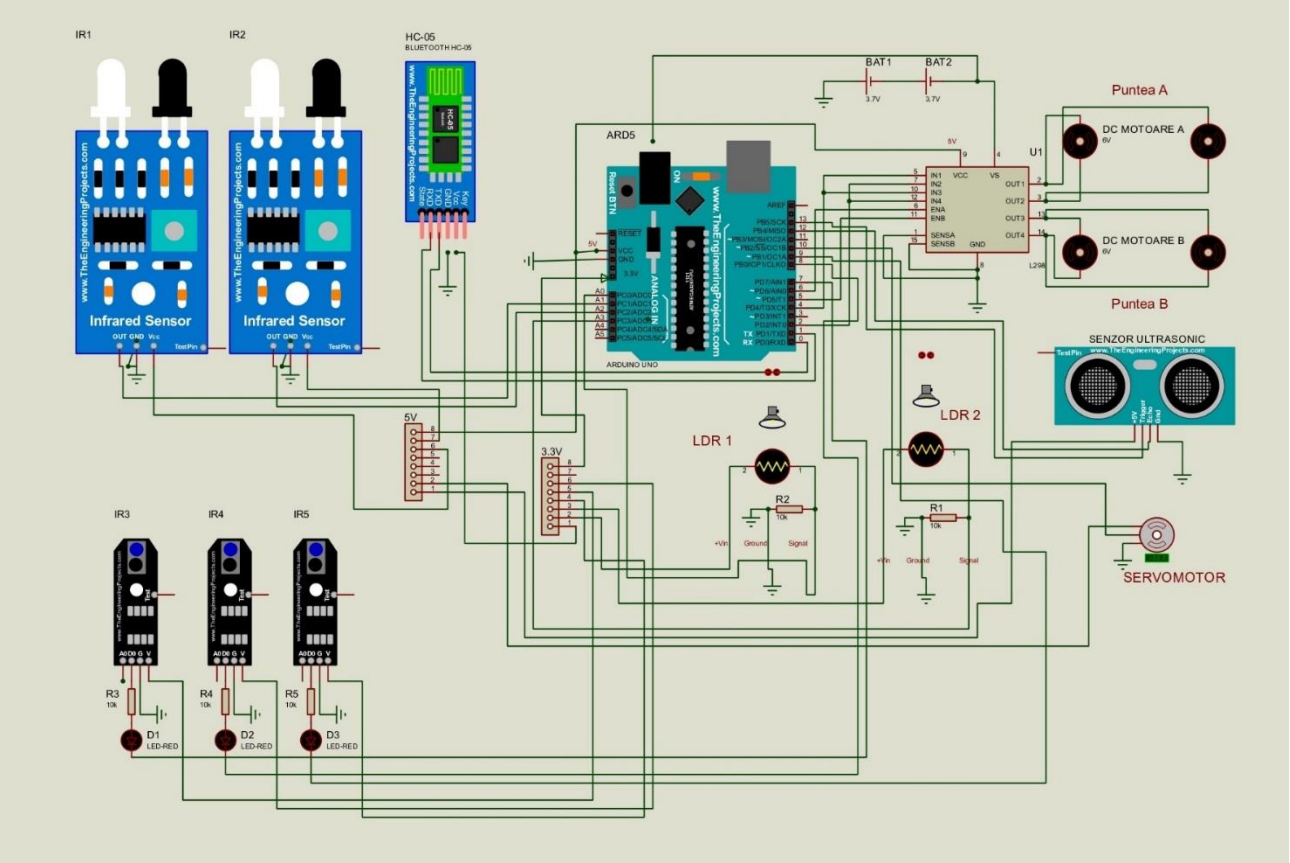



**Figură 15** Baterii litium-ion 18650

## 2.1.2 Schema circuitului electric, imagini și funcționalități

În realizarea schemei circuitului electric, s-a utilizat Proteus Professional 8. Proteus este un program specializat pentru a realiza capturi schematice, simulări și PCB (*printed circuit board* sau Placă de Circuit Imprimată *)*. Capturarea schematică pentru simularea proiectelor, cât și faza de proiectare a unei plăci PCB este o componentă de bază și este inclusă în toate configurațiile produselor, pieselor și integratelor electronice.

În figura de mai jos, vom vedea schema circuitului electric al proiectului de față. Este urmat de asemenea și de explicații, în vederea înțelegerii principiului de funcționare.




**Figură 16** Schema circuit realizată în Proteus

În schema din figura 16 s-au utilizat biblioteci ce includ senzorii utilizați în proiect de pe website-ul www.theengineeringprojects.com. Această pagină web este o platformă pentru ingineri, studenți, dar și oameni profesioniști sau pasionați cu scopul de a-și împărtășii proiectele, experiența și modul de lucru.

În cadrul proiectului întâlnim mai multe funcționalități, ce sunt strâns legate cu modul de funcționare al senzorilor prezentați anterior. În partea din dreapta a schemei electrice, putem observa modul de implementare al punții H LN298HN cu placa de dezvoltare Arduino, având ca ieșire cele patru motoare legate la Puntea A și Puntea B. Acest ansamblu stă la baza mobilității mașinii. În partea din stânga a plăcuței de dezvoltare Arduino se poate observa modulul bluetooth HC-05, legat la interfața serială de comunicare UART. Acest protocol de comunicare este unul simplu, deoarece necesită doar două fire. Pachetele de date trimise de acesta au următorul format: un bit de start ce este reprezentat de 0 logic, ulterior urmează până la 8 biți de date, un bit de paritate și în final, un bit de stop reprezentat de 1 logic [19]. Modulul bluetooth HC-05 acționează ca *master* al plăcii de dezvoltare, ceea ce înseamnă că poate să inițieze și să controleze comunicația între plăcuță și alte dispozitive externe prin intermediul

conexiunii Bluetooth. Prin urmare, atât partea de conectare cu puntea H, cât și conexiunea cu modulul Bluetooth stau la baza tuturor funcționalităților mașinii.

Prima funcționalitate a mașinii o reprezintă funcția de *line-tracking* sau “Urmărește linia”. Această funcție utilizează trei senzori cu infraroșu (de văzut partea de jos din stânga, figura 16), IR 3, IR 4 și IR5, care împreună favorizează un bun demers al urmăririi unei linii pre-stabilite. Linia pre-stabilită de noi poate să fie comparată cu acțiunea de păstrare corectă a benzii de circulație. În figura 7 (poziționare motoare) se poate observa și amplasarea celor trei senzori pe partea inferioară a mașinii, în exterior, în fața motoarelor. Acest amplasament ajută deplasarea în cadrul funcției menționate. În figura de mai jos este ilustrată reprezentativ funcția.

**Figură 17** Funcția "Urmărește Linia"

O a doua funcție a mașinii o reprezintă *light-siking* sau “Urmărește Lumina”. Această funcție utilizează doi senzori fotorezistivi, ce sunt reprezentați în figura 16 sub plăcuța de dezvoltare (LDR 1 și LDR 2) și LN298HN (puntea H). Senzorii sunt amplasați pe partea din față a mașinii (pe partea superioară cu orientarea în față) și deservesc la controlul acesteia în funcție de poziția sursei de lumină și distanța la care aceasta se află. Prezenta funcție din cadrul proiectului oferă un răspuns satisfăcător și în medii iluminate ambiental mai puternice. Totuși, un mediu iluminat foarte puternic poate să reprezinte un aspect limitant, deoarece duce la configurarea unui algoritm mai complex care este variabil în funcție de nivelul de iluminare al încăperii.

Mai jos se poate observa o figură reprezentativă acesti funcții.

**Figură 18** Funcția "Urmărește Lumina"

O a treia funcție prezentă în cadrul proiectului o reprezintă funcția “Mișcare automată”. Această funcție utilizează doi senzori cu infraroșu (IR1 și IR2), un senzor ultrasonic și un microservo (senzorul ultrasonic este poziționat în partea dreaptă-mijloc, iar microservo-ul se află sub acesta, fiind observabile în schema electrică din cadrul figurii 16). Microservo-ul va ajuta senzorul ultrasonic să se rotească pe axa Y, fiind atașat pe acesta. În timpul acestei funcții, mașina se deplasează singură, evitând și oprindu-se în momentul în care în fața acesteia se află un obiect. În momentul în care un obiect se află poziționat în aria de acoperire a senzorului ultrasonic la o anume distanță, mașina se oprește și își va schimba direcția de deplasare în funcție de distanța din stânga și din dreapta a acesteia. În momentul în care se oprește, servoul se rotește către stânga și dreapta și va prelua informațiile necesare cu privire la distanță. După ce sunt preluate informațiile, acesta va alege cea mai potrivită direcție de deplasare. Senzorul ultrasonic având un unghi de măsurare limitat, acesta nu poate să capteze prezența obiectelor din lateral. Prin urmare, s-au utilizat cei doi senzori cu infraroșu (IR1 și IR2) ca și efect compensator pentru limita anterior menționată. Când aceștia sunt activi, mașina ocolește rapid obstacolul. În figura de mai jos, putem observa funcționalitatea anterior descrisă.



**Figură 19** Funcție "Mișcare automată"

A patra funcție din cadrul proiectului se numește *ultrasonic follow* sau "Urmărește ultrasonic". Această funcție se bazează pe capacitatea mașinii de a urmării un obiect aflat în mișcare și amplasat în fața acesteia. Se folosește de o parte din senzorii utilizați la funcția anterioară, adică senzorii cu infraroșu de pe mașină (IR1 și IR2), senzorul ultrasonic și capacitatea de mobilitate dată de puntea H și de motoare (de văzut figura 16). În cadrul acestei funcții se poate utiliza palma sau un obiect aflat în mișcare, care este plasat în fața mașinii, acesta începând să-l urmărească. Dacă obiectul este plasat doar în fața senzorului ultrasonic, mașina execută mersul înainte. Dacă obiectul este interceptat și de senzorul ultrasonic dar și de unul dintre cei doi senzori cu infraroșu, atunci mașinuța va executa o mișcare la stânga sau la dreapta după obiectul respectiv. Odată ce mașina ajunge la o distanță dată, distanță măsurată cu ajutorul senzorului ultrasonic, aceasta se opreste. Dacă distanța continuă să scadă până la un prag minim setat (explicat la partea de cod), aceasta execută mersul cu spatele pentru a evita o eventuală atingere cu obiectul. Mersul cu spatele continua până când se atinge distanța de siguranță normală.

**Figură 20** Funcție "Urmărește ultrasonic"

### 2.1.3 Software

În proiectul de față, ca mediu software, s-a utilizat Arduino IDE pentru programarea plăcii de dezvoltare Arduino UNO R3. Arduino IDE conține un editor de text pentru scrierea codului, o zonă unde sunt trimise mesaje către utilizator (de exemplu un mesaj de informare cu privire la încărcarea codului), o consolă text, bara de instrumente ce conține o serie de meniuri și funcții. De asemenea, Arduino IDE este un instrument accesibil și convenabil pentru studenți și persoane care lucrează în domeniu sau pasionate, deoarece plăcile de dezvoltare Arduino prezintă un preț convenabil și sunt disponibile pe o scară largă (fie de la producător, fie de la surse terțe). [1]

În această parte a lucrării, se vor analiza din punct de vedere software bibliotecile folosite, funcții și secvențe de cod importante din cadrul celor patru funcții menționate în capitolul anterior.

Pentru început, Arduino IDE conține două funcții esențiale și predefinite „setup()” și “loop()”. Funcția “void setup()” este apelată o singură dată în momentul în care placa de dezvoltare este pornită și aceasta conține de regulă inițializarea pinilor ca intrări sau ieșiri, setarea *baud-rate*, inițializarea comunicării seriale și stabilirea vitezei acesteia. Funcția “void loop()” este executată într-un ciclu continuu, imediat după terminarea funcției setup. În cadrul funcției “void loop()” se pot citii date de la senzori sau se pot îndeplinii anumite sarcini dorite. Această funcție rulează până când placa de

dezvoltare este oprită. Inițializarea variabilelor și definirea funcțiilor se fac în afara acestor două funcții. [20]

Singura bibliotecă folosită la nivel de cod este biblioteca “Servo”. Această bibliotecă a fost realizată de Michael Margolis (2009) și are ca scop inițializarea pinului digital, funcția “attach()” la care microservoul se atașează. Funcția “write()” are ca scop setarea unghiului la care servomotorul se rotește, luând ca parametru un număr întreg (de la 0-180, 180 fiind capacitatea maximă de rotație a microservoului utilizat în acest proiect). Inițializarea variabilei servo se face înainte de „setup()” prin linia de cod “Servo servo;” [21].

Înainte de funcția “setup()” s-au inițializat variabilele necesare pentru definirea pinilor, ceea ce oferă o lizibilitate crescută la nivel de cod. Pe lângă acestea s-au mai definit și variabile necesare pentru funcțiile mașinuței.

Variabila “dateSeriale” are ca scop citirea și salvarea datelor seriale de pe monitorul serial ce urmează să fie interpretate. Acest lucru se face în linia de cod “dateSeriale = Serial.read();” din interiorul funcției “loop()”.

În interiorul lui “void setup()” se va seta viteza comunicației seriale prin linia de cod “Serial.begin(9600);” , se vor inițializa pinii folosiți la nivelul proiectului ca și *INPUT/ OUTPUT* sau intrare/ ieșire și se va seta unghiul la care se rotește servo-ul la 120°. Din pricina amplasării hardware a servomotorului pe mașină și a orientării acestuia, acesta este rotit cu o abatere de 30° și unghiul de mijloc, adică poziția în care senzorul ultrasonic este îndreptat în față este 120° și nu 90° cum este în condițiile ideale de amplasare.

În interiorul lui “void loop()” este definită structura programului ce rulează într-o buclă infinită, în care sunt apelate funcții principale ale mașinii.

În momentul în care conexiunea este stabilită între telefon și modulul Bluetooth, ultimul menționat trimite datele reprezentate de octeți  pe monitorul serial. Funcția “void loop()” începe cu condiția logică în cadrul căreia se stabilește dacă există sau nu date pe monitorul serial prin linia de cod “`if(Serial.available() > 0)`”.  După ce această condiție este îndeplinită, variabila “dateSeriale” primește ultimul octet afișat pe monitorul serial. Acest fapt se realizează prin linia de cod “`dateSeriale = Serial.read();`”. În cele ce urmează, există o serie de condiții care verifică valoarea octetului salvată în variabila “dateSeriale”. [22]

| Condiții logice | Caracter ASCII | Valoare zecimală | Funcție |
|---|---|---|---|
| Condiție logică 1 | 1 | 49 | inFata() |
| Condiție logică 2 | 2 | 50 | inSpate() |
| Condiție logică 3 | 3 | 51 | inStanga() |
| Condiție logică 4 | 4 | 52 | inDreapta() |
| Condiție logică 5 | 5 | 53 | inDreapta() continuu |
| Condiție logică 6 | 6 | 54 | inStanga() continuu |
| Condiție logică 7 | 7 | 55 | inFata() continuu |
| Condiție logică 8 | 8 | 56 | inSpate() continuu |
| Condiție logică 9 | # | 35 | miscareAutonoma() |
| Condiție logică 10 | $ | 36 | urmarireBandaInfrosu() |
| Condiție logică 11 | % | 37 | urmaresteLumina() |
| Condiție logică 12 | & | 38 | urmaresteUltrasonic() |
| Condiție logică 13 | @ | 64 | opresteMiscare() |

**Tabel 12** Instrucțiuni logice în funcția loop()

Conform tabelului 12, în perechile de condiții logice 4-5, 3-6, 1-7 și 2-8 sunt executate aceleași funcții principale, dar completarea "continuu" indică faptul că la nivel de cod acestea se execută repetat atunci când utilizatorul aplicației de pe telefon apasă lung pe butoanele ce trimit caracterele ASCII corespunzătoare condițiilor logice 5,6,7 și 8. Condiția logică 1 arată în felul următor la nivel de cod:

```
if(dateSeriale == 49){
  inFata();
  dateSeriale = "\n";
  dateSeriale = "\0";
  delay(d3);
  opresteMiscare();
}
```

În interiorul condiției logice se verifică dacă variabila “dateSeriale” este egală cu 49 (caracterul 1 în ASCII). Se apelează prima dată funcția “inFata()”,variabila “dateSeriale” primește valoarea \n, ce reprezintă *new line* sau trecere la linie nouă, lucru folosit la partea de vizualizare în consolă. Ulterior, aceasta primește valoarea \0, valoare ce este reprezentată de bitul *null*, ceea ce setează valoarea variabilei la valoarea precizată. Apoi sunt apelate funcțiile “delay()”, funcție ce este bazată pe întrerupere si intârzie execuția liniilor de cod, primește un argument de tip întreg și este reprezentat de valoarea în milisecunde (variabila “d3” având valoarea 1000 ) si “opresteMiscare()” care setează viteza exprimată în valoare *PWM* pe puntea A și B la 0. [23]

Condiția 7 logică, perechea condiției 1, prezintă următoarele linii de cod:

```
if(dateSeriale == 55){
  inFata();
  dateSeriale = "\n";
  dateSeriale = "\0";
}
```

Această condiție rulează continuu până când variabila “dateSeriale” își schimbă valoarea, adică până când utilizatorul aplicației ridică degetul de pe butonul respectiv ce are setat ca eveniment de intrare o apăsare mai îndelungată.

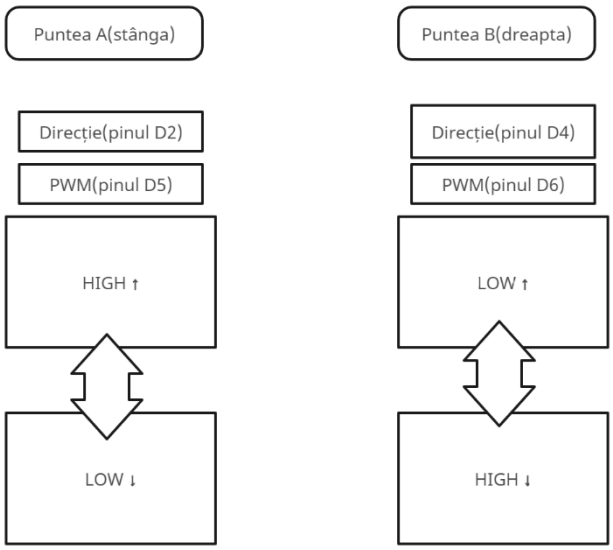
Acest format se aplică la toate perechile de condiții logice enunțate mai sus, funcționalitatea lor fiind una similară, alcătuind funțiile de control ale utilizatorului aplicației asupra mișcării mașinii.

Condițiile logice 9,10,11,12 și 13 sunt reprezentate de funcții mai complexe îndeplinite de mașină, care doar se apelează în funcția “loop()”, acestea fiind create în afara funcției menționate anterior. Aceste funcții respectă formatul următorului exemplu:

```
if(dateSeriale == 36){
  urmarireBandaInfrosu();
  dateSeriale = "\0";
}
```

Până în acest punct, s-a descris modul de funcționare al aplicației. Pentru început, au fost descrise mai multe condiții logice și utilitatea lor. Întrucât nivelul de complexitate crește odată cu înaintarea în cod, în cele ce urmează se vor explica funcțiile logice.

În condițiile în care toate cele 4 opțiuni pe care mașinuța le prezintă sunt bazate pe mișcare, se va începe cu descrierea funcțiilor de mișcare la nivel de cod. În aceste funcții se definesc sensul și viteza motoarelor legate la puntea A și B. În figura de mai jos se poate observa modalitatea de funcționare a motoarelor fiecărei punți în funcție de valorile date.




**Figură 21** Diagramă bloc punte H și pinii folosiți

Pe baza diagramei din figura 21, s-au definit toate funcțiile de mișcare ale mașinii. Un exemplu îl reprezintă funcția de mișcare “inFata()”, ce setează modul de deplasare pe direcția înainte. Asemenea funcții prezentate și celelalte sunt realizate într-o modalitate similară.

```
void inFata(){
    digitalWrite(pinDirSt,HIGH);
    analogWrite(pinPWMSt,100);
    digitalWrite(pinDirDr,LOW);
    analogWrite(pinPWMDr,100);
}
```

Funcțiile legate de mișcare se numesc “dreaptaEvitare”, “stangaEvitare”, “inSpate”, “inFataIncet”, “inSpateDreapta”, “inSpateStanga”, “inFataStanga”, “inFataDreapta”, “inFata”, “inStanga”, “inStangaRapid”, “inDreapta”, “inDreaptaRapid”, “opresteMiscare”. Acestea sunt apelate în interiorul funcțiilor mai complexe pentru a controla mașinuța în funcție de datele furnizate de către senzori, pe scurt în funcție de situația de mediu în care se află mașinuța.

Prima funcție cu un nivel mai ridicat de complexitate o reprezintă funcția “urmarireBandaInfrosu”, ce are ca și rol urmărirea unei benzi sau *line-tracking*, aspect prezentat în capitolele anterioare. Senzorii cu infraroșu sunt inițializați pe pinii digitali 7,8 și 9, valorile piniilor find salvate în variabile de tip *boolean*. Datele înregistrate de aceștia atunci când sunt activi și trec de pragul

de *threshold*, aceștia indică valoarea 0, fapt pentru care s-a ales negarea valorii întoarsă de aceștia (din motive de logică și simplitate). În funcție de valorile înregistrate de cei trei senzori, mașina execută o anumită funcție de mișcare. Datorită faptului că există trei senzori, vor rezulta opt situații diferite în care aceștia vor prezenta valori logice. [24]

S-au omis cazurile în care cei trei indică valoarea 1 logic sau 0 logic.

În primul rând, atunci când indică valoarea 0 logic, rezultă că nu se află deasupra liniei ce trebuie urmărită.

În al doilea rând, în momentul în care mașinuța ajunge în necesitatea unei virări pe linia urmărită, toți cei trei senzori înregistrează valoarea 1 logic. Din pricina acestui fapt, s-a ales să nu se definească o funcție de mișcare în acest caz. În situația în care s-ar fi definit o funcție de mișcare, mașinuța ar fi existat posibilitatea ca mașinuța să părăsească (să se abată) linia urmărită.

Cea de a doua funcție este reprezentată de "miscareAutonoma()" în care sunt salvate datele transmise de cei doi senzori cu infraroșu amplasați pe mașină (detalii în capitolele anterioare) în variabile, date ce sunt negate la fel ca în funcția anterior discutată. Într-o altă variabilă se va salva valoarea distanței din fața mașinii în centimetrii (cm), date preluate de senzorul ultrasonic, atribuind variabilei funcția "verificaDistanta()", unde sunt prelucrate datele captate de senzorul anterior. După cele trei inițializări, se vor apela funcțiile "evitareInfrarosuLateral()" și "evitareUltrasonica()", în cadrul cărora se vor utiliza variabilele inițializate anterior.

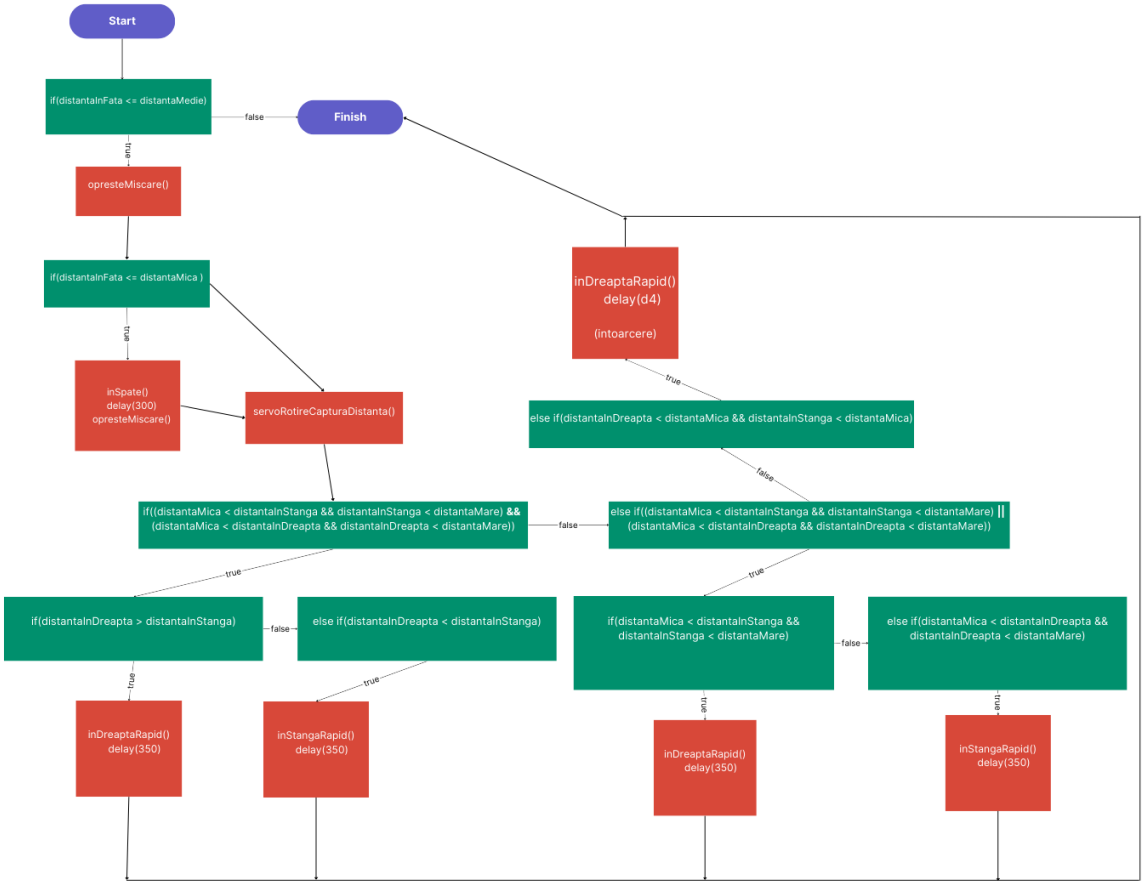
În funcția "evitareInfrarosuLateral()", prezența celor doi senzori cu infraroșu indică faptul că avem patru cazuri în care aceștia sunt în stări logice diferite. În funcție de contextul situațional, mașina execută funcțiile de mișcare corespunzătoare pentru a evita obstacolele.

Pentru îndeplinirea funcționalității funcției "miscareAutonoma()" s-au folosit două funcții de măsurare a distanței "verificaDistanta()" și "servoRotireCapturaDistanta()". Funcția "verificaDistanta()" incorporează principiul de funcționare al senzorului ultrasonic descris în fișa tehnică a acestuia (vezi în capitolul Hardware). Pinul echo (legat pe pinul digital 12 și setat ca *OUTPUT*) i se atribuie valorile de stare *LOW-HIGH-LOW* la un interval de 10 microsecunde pentru a trimite și a aștepta răspunsul recepționat de pinul *trigger* (legat pe pinul digital 13 și setat ca *INPUT*) [12]. În variabila "pingTimpRaspuns", este atribuită valoarea recepționată de pinul *trigger* în microsecunde, cu ajutorul funcției "pulseIn(pinTrigger, HIGH)", funcție oferită de mediul de dezvoltare Arduino IDE(returnează distanța in uS). Pentru a afla distanța în centimetrii (cm), este necesară folosirea formula calculului distanței ($distanța = timp \times viteză$). Aproximând viteza sunetului la 340 m/s [25], putem utiliza următoarea formula pentru a afla distanța în centimetrii: $distanța\ [cm] = timp\ [uS] \times$

$viteza\ [\frac{m}{s}]$ [26]. După transformarea unităților de măsură și simplificarea ecuației, ajungem la următoarea linie de cod pentru calculul în centimetrii: „pingDistantaRaspuns = (pingTimpRaspuns * 0.0350);”. Această distanță trebuie împărțită la doi deoarece unda de sunet trimisă de pinul *echo* călătorește către obiect, se lovește de acesta și apoi se întoarce. Prin urmare, rezultă următoarea linie de cod: “distantaObstacol = pingDistantaRaspuns/2;”. Totodată, “distantaObstacol” reprezintă valoarea returnată de funcție (tip *float*).

O altă funcție de calcul a distanței menționată anterior, este reprezentată de funcția “servoRotireCapturaDistanta()”, funcție ce are rolul de a calcula distanța în stânga și în dreapta mașinuței. Această funcție este realizată cu ajutorul microservoului (legat pe pinul digital 10). Servoul este îndreptat pe poziția de mijloc (în cazul în care acesta era îndreptat în altă direcție), după fiecare mișcare a servoului este necesară aplicarea unui *delay* deoarece se așteaptă mișcarea mecanică a acestuia. Apoi, servoul este îndreptat înspre fiecare direcție, stânga și dreapta, moment în care cele două variabile “distantaInStanga” și “distantaInDreapta” li se atribuie funcția “verificaDistanta()”.

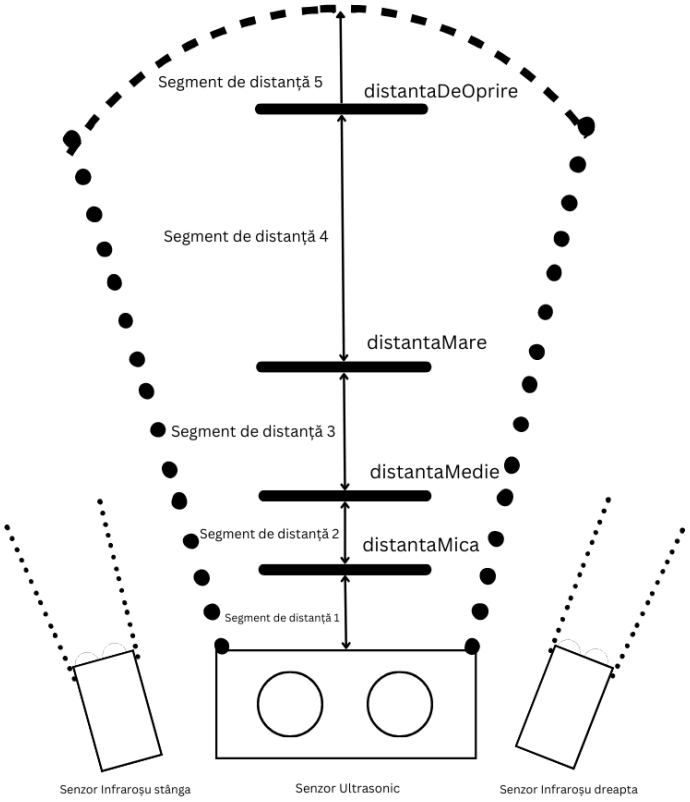
În funcția “evitareUltrasonica()” se verifică dacă distanța în față este mai mică sau egală cu o distanță de siguranță sau la nivel de cod numită “distantaMedie” ce are valoarea de 19 centimetrii. Dacă




**Figură 22** Schemă bloc "Evitare ultrasonică"

această condiție este îndeplinită, se oprește mișcarea mașinii și se verifică mai multe condiții logice conform schemei bloc de mai jos.

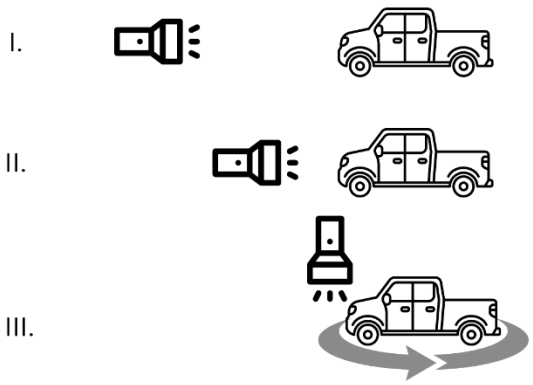
Cea de a treia funcție importantă din proiect o constituie funcția “urmaresteUltrasonic()”. În cadrul acestei funcții s-au folosit senzorii din funcția precedentă, dar nu și microservoul. La apelul funcției, două variabile “InfrarosuSt” și “InfrarosuDr” li se atribuie valorile negate, înregistrate de senzorii cu infraroșu din stânga și din dreapta. De această data, se va utiliza variabila “distantaInFata” care ii se atribuie funcția “verificaDistanta()”, în vederea verificării distanței în centimetrii. Această funcție este reprezentată de un șir de instrucțiuni logice care verifică distanța din fața mașinuței dacă se încadrează în niște distanțe prestabilite (s-au ales patru puncte de distanță). În cazul în care ajunge într-unul din segmentele de distanță, atunci funcția va verifica valoarea logică a senzorilor cu infraroșu și execută o anumită comandă de mișcare în funcție de aceștia. În figura următoare se poate vedea un desen reprezentativ.




**Figură 23** Reprezentare funcție "Urmărește ultrasonic"

Ultima funcție este reprezentată de “urmaresteLumina”, ce are ca și scop controlul mașinii cu ajutorul unei surse de lumină, de exemplu o lanternă. Funcția numită utilizează cei doi senzori fotorezistivi, amplasați pe fața mașinii (legați pe pinii analogici A0 și A3). La începutul funcției, se

inițializează două variabile cu valoarea de intrare analogică a fiecărui senzor. Aceștia înregistrează valori între 600-800, când sursa de lumina este la o distanță considerabilă sau aceasta fiind inexistentă. Pentru un calcul mai ușor, valorile atribuite variabilelor se împart la 10. Atunci când senzorii primesc un fascicul de lumină, valoarea înregistrată de aceștia scade. Prin urmare, dacă ambii senzori vor înregistra o valoare mai mare de 60, atunci mașina va sta pe loc. În cazul în care valoarea acestora este mai mică decât 60, se va compara care dintre cele două valori este mai mare în două condiții logice. Dacă senzorul din stânga indică o valoare mai mare, atunci se va realiza raportul dintre senzorul din dreapta și cel din stânga înmulțit cu 3 și va fi atribuită unui scalar ($scalar = \frac{fotoDr}{fotoSt} \times 3$). Utilizăm acest scalar deoarece dorim ca atunci când mașina este la o distanță mai mare de sursa de lumină aceasta să se deplaseze cu o viteză mai mare la început, iar pe măsură ce începe să se apropie, aceasta să fie aproape oprită. Când valoarea scalarului scade sub 1, mașina va începe să vireze în stânga sau în dreapta în funcție de situație, deoarece este în punctul în care se află foarte aproape se sursa de lumină și aceasta va avea nevoie să vireze, continuând să urmărească.




**Figură 24** Exemplificare funcție "Urmărește Lumina"

În figura 24, s-a exemplificat procesul de funcționare al funcției anterior menționate, în trei stadii, în funcție de apropierea luminii (lanternă) de senzorii fotorezistivi. În cazul I, mașina se deplasează cu o viteză mai mare decât în cazul II, iar în cazul III aceasta este oprită pe loc, executând o rotație la stânga sau la dreapta în funcție de valorile înregistrate.

## 2.2 Proiectare aplicație pe telefon

### 2.2.0 Mediul de dezvoltare Android Studio

Android Studio este un mediu de dezvoltare integrat (*IDE*) pentru sistemul de operare Android, construit pe platformele de software JetBrains si IntelliJ, design-ul aplicației fiind făcut special pentru dezvoltarea Android. Acesta este disponibil pe majoritatea sistemelor de operare și este înlocuitorul lui Eclipse Android Development Tootls ca IDE primar pentru aplicații native Android. Android Studio suportă aceleași limbaje de programare ca și InteliJ (Java si C++). De la versiunea Andoird Studio 3.0 este suportat și limbajul de programare Kotlin, preferat de cei de la Google ca și înlocuitor pentru Java. [27]

In proiectul current, s-a folosit limbajul Java deoarece este mai versatil, folosit în mai multe medii de dezvoltare, fiind unul din limbajele pe care s-a pus accent în cadrul perioadei universitare.

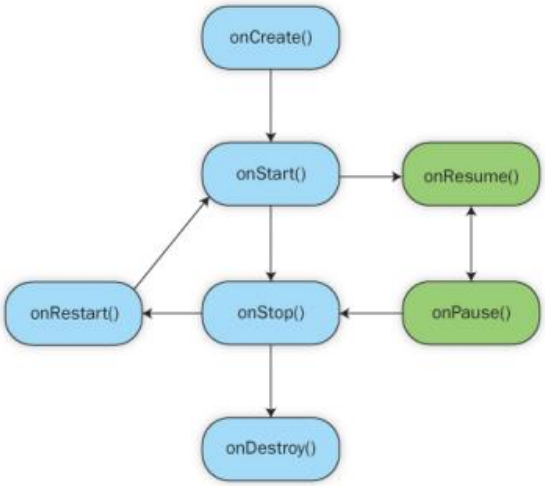
Java este un limbaj de programare bazat pe clasă și orientat pe obiecte, fiind proiectat să necesite cât mai puține dependențe de implementare. Destinat programatorilor să scrie o dată și să ruleze oriunde, codul java odată compilat poate rula pe toate platformele care acceptă Java fără a fi necesară recompilarea. [28]

### 2.2.1 Structura generală a aplicației

În strucutra de bază a fiecărei aplicații Android, se pot utiliza două componente de bază: Fragmente și Activități. Ele sunt componente diferite, fragmentele sunt folosite pentru a crea interfețe de utilizator mai dinamice, iar Activitățile sunt folosite preponderent pentru a creea puncte de acces în aplicație și pentru a gestiona Fragmentele. [29]

În prezentul proiect, s-a utilizat o activitate denumită și “ActivitatePrincipala” și mai multe Fragmente. Rolul “ActivitatiiPrincipale” este acela de a fi statică pe tot parcursul aplicației, fiind afișat în partea de sus și de jos a interfeței de utilizator, comparabil cu un *header* si un *footer*. În partea mediană a interfeței utilizatorului este regăsit un element de tip “fragmentContainerView” în care se afișează toate fragmentele create în proiect. Acest “fragmentContainerView” necesită un fragment pe care să îl încarce la rularea aplicației, acesta fiind “FragmentulMeniuPrincipal”. Cum reise din numele acestuia, fragmentul ce deseverște meniul principal are ca scop găzduirea tutror opțiunilor de navigare în alte fragmente din proiect. Meniul este sturcturat vertical și trecerea între fragmente se face cu ajutorul unor butoane cu evenimente de ieșire.

Activitățile din Android sunt structurate pe un ciclu de viață (*activity lifecycle*). Acest ciclu de viață este bazat pe 7 metode, fiecare dintre acestea având un rol important în cadrul aplicațiilor Android.

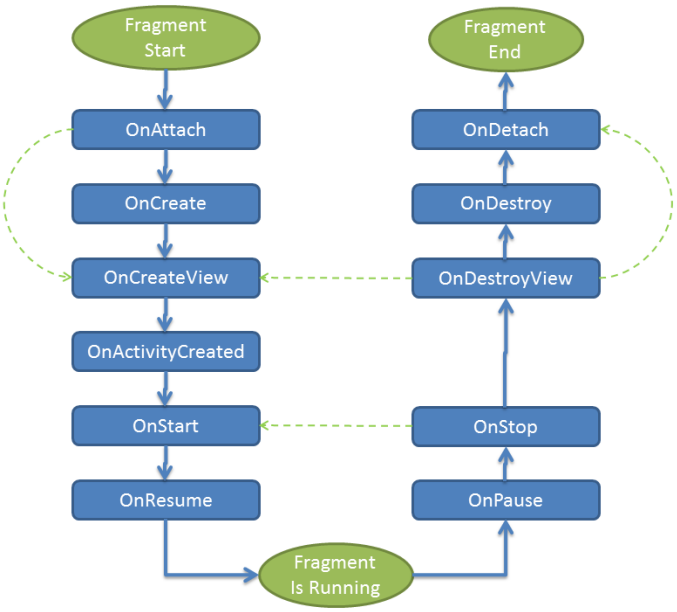



**Figură 25** Ciclu de viață al activității aplicațiilor în Android

La lanasarea aplicației, metoda „onCreate()" este apelată. În această metodă se încarcă *layout*-ul și majoritatea funcționalităților aplicației. După apelul metodei onStart(), activitatea devine vizibilă, mai apoi permițând utilizatorului să interacționeze cu aceasta, după apelul metodei onResume().

Când utilizatorul a rulat aplicația și aceasta s-a încărcat, apasă pe butonul *Home* al telefonului sau lasă aplicația „în bara" de aplicații, se apelează metoda "onStop()". În momentul în care revine în aplicație, se face apelul metodei "onRestart()", iar apoi revine în succesiunea de apeluri "onStart()" și "onResume()". Metoda "onStop()" este invocată și atunci când activitatea este întreruptă de un eveniment de sistem, un apel telefonic, o alertă de baterie scăzută, o schimbare de configurație (când telefonul trece de pe modul *portrait* în *landscape*) dar și atunci când sistemul android solicită eliberarea resurselor deoarece memoria este prea încărcată. [30] [31]

Când activitatea își încheie ciclul de viață, aceasta face apelul funcției "onDestroy()". În momentul în care este distrusă, aceasta este eliberată complet din sistem și se golesc resursele folosite de aceasta. În cazul aplicațiilor mai complexe este importantă această metodă deoarece aici trebuie să fie gestionate corect dezalocările de memorie, pentru că se pot crea *memory-leaks* sau scurgeri de memorie.




**Figură 26** Ciclul de viață al fragmentelor aplicațiilor Android

Ciclul de viață al fragmentelor în Android este strâns legat de ciclul de viață al activității pe care acestea sunt atașate. Ele sunt componente independente și reutilizabile, ce pot să fie incorporate într-o activitate pentru a crea interfețe de utilizator. Față de activități, fragmentele au câteva metode în plus. Din moment ce un fragment necesită o activitate, acesta prezintă metoda "onAttach()", metodă care are rolul de a atașa fragmentul de activitatea gazdă. Alte câteva metode specific fragmentelor sunt: "onCreateView()" în care *layout-ul* fragmentului este inflat și inițializat; "onViewCreated()" în care este garantată existența vizualizării fragmentului; "onActivityCreated" unde și activitatea gazdă și fragmentul au fost create și inițializate. La fel ca și activitățile, acestea au nevoie de metoda "onDestroy()", având în plus metoda "onDestroyView()", care distruge *layout-ul* fragmentului. La final, după ce *layout-ul* a fost distrus împreună cu fragmentul, acesta se "desprinde" de activitatea gazdă prin metoda "onDetach()". [32]

Un alt lucru important când vine vorba despre dezvoltarea aplicațiilor pe telefon îl reprezintă fișierul **`AndroidManifest.xml`**. Acest fișier descrie informații esențiale despre aplicație precum *build tools*, sistemul de operare Android, activități, servicii și permisiuni de care aplicația are nevoie pentru a accesa părți protejate ale sistemului de operare.

În acest proiect s-au folosit următoarele linii de cod necesare pentru conexiune în fișierul **AndroidManifest.xml**:

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
<uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
<uses-permission
android:name="android.permission.ACCESS_BACKGROUND_LOCATION" />
```

Aceste elemente de tip android din interiorul tag-ului "**<uses-permission>**" au ca scop accesul aplicației la permisiunile pentru bluetooth în scopul conectării, scanării, descoperire dispozitive, cât și pentru locație în vederea accesului la locația fixă, relativă și accesul locației în *background*, adică când aplicația rulează în fundal.

Alte aspecte legate de aplicație ce se găsesc în acest fișier XML pot fi: iconița aplicației, numele aplicației, tipul de iconiță de fundal, tema de culori a aplicației și API-ul minim pe care aplicația îl utilizează (versiunea minimă a platformei de Android pe care aplicația poate rula). Acestea se găsesc în tag-ul **<application>** și secvențele de cod pot arăta în felul următor:

```
android:icon="@drawable/icon_application"
android:label="@string/app_name"
android:roundIcon="@mipmap/ic_launcher_round"
android:supportsRtl="true"
android:theme="@style/Theme.BluetoothAplicatie"
tools:targetApi="31">
```

În cadrul tag-ului **<activity>**, s-a definit activitatea care se încarcă în momentul în care aplicația este lansată și orientarea ecranului pe toată perioada activității. Secvențele de cod pot arăta în felul următor:

```
android:name="com.example.aplicatiebluetooth.ActivitatePrincipala"
android:exported="true"
android:screenOrientation="portrait">
```

### 2.2.2 Bluetooth și elemente necesare conexiunii

Tehnologia bluetooth este o tehnologie standard wireless, de rază scurtă, care este folosită pentru a schimba informații și date între dispozitive fixe și mobile. Aceasta construiește o rețea tip *PAN*s (*personal area network sau* arie de rețea personală*)*. Bluetooth-ul operează pe o rază scurtă de până la 10 metrii (pentru o conexiune ideală), utilizând unde radio cu frecvență înaltă în benzile ISM (aceste benzi radio ISM sunt porțiuni din spectrul radio rezervate la nivel internațional pentru scopuri industriale, științifice și medicale (ISM), cu excepția aplicațiilor în telecomunicații), de la 2.402 GHz până la 2.48GHz [33].

Emisiile puternice date de frecvențele înalte din cadrul benzilor ISM pot să creeze interferențe electromagnetice și să întrerupă comunicațiile radio ce folosesc aceași frecvență.

IEEE (Institute of Electrical and Electronics Engineers) a standardizat tehnologia Bluetooth ca IEEE 802.15.1, dar la momentul de față aceștia nu mai dețin standardul.

Un prim element necesar conexiunii Bluetooth îl reprezintă UUID sau *universally unique IDentifier* care este un identificator universal unic. UUID este o etichetă de 128 de biți ce i se atribuie aceleași clase și aceluiași model de dispozitiv. Spre exemplu toate modulele bluetooth HC-05 prezintă același UUID (UUID-ul modulului este 00001101-0000-1000-8000-00805F9B34FB). Acesta se poate afla atunci când modulul bluetooth este legat în circuit cu plăcuța de dezvoltare astfel încât acesta să poată să răspundă la comenzile AT. În fișa tehnică sunt detaliate toate comenzile de interogare a modulului Bluetooth. [34]

Un al doilea element important în vederea conexiunii dintre modulul bluetooth și telefon îl reprezintă un *socket* de conexiune. Socket-ul de conexiune în protocolul de comunicare TCP/IP este compus din două elemente și anume adresa IP și Port. Interfața de conexiune prin bluetooth este similară cu cea TCP/IP deoarece necesită un socket pentru client și un socket pentru server. Cel mai comun tip de socket bluetooth este RFCOMM, ce este un tip de socket suportat de API-urile sistemului de operare Android. RFCOMM este orientat pe conexiune și transport de date prin bluetooth, fiind de asemenea cunoscut sub numele de *Serial Port Profile* (SPP). În cadrul proiectului, serverul este reprezentat de modulul HC-05, iar clientul este telefonul ce rulează aplicația de control a mașinii. [35]

### 2.2.3 Interfața grafică a aplicației

În această aplicație s-a folosit o activitate ca nod principal al interfeței utilizatorului și o serie de fragmente pentru meniul principal și funcționalitățile mașinii.

Fiecare dintre acestea prezintă câte un fișier XML în folderul de resurse "*layout*" al aplicației în care se definesc tipul de layout și diverse elemente pentru a crea o interfață de utilizator atractivă și funcțională.

Tipul de *layout* al activității principale este "LinearLayout", ce permite aranjarea elementelor într-o singură direcție, fie ea verticală sau orizontală. Acest tip de *layout* oferă utilizarea atributului **`android:layout_weight`**, pentru a specifica spațierea ocupată de fiecare element, în funcție de greutatea atribuită. În concepția design-ului, s-a folosit conceptul de "LinearLayout" părinte și "LinearLayout" copil, adică în interiorul "LinearLayout" principal (părinte) au fost introduse alte "LinearLayout" pentru a fragmenta în procente interfața.

Pentru început, se vor expune elementele de interfață grafică, experiența utilizatorului, precum și modul de utilizare al aplicației. Ca aplicația să funcționeze corect, trebuie avut în vedere alimentarea și conectarea modulului bluetooth HC-05 la placa de dezvoltare (se pornește bluetooth-ul și se va realiza împerecherea cu acesta).

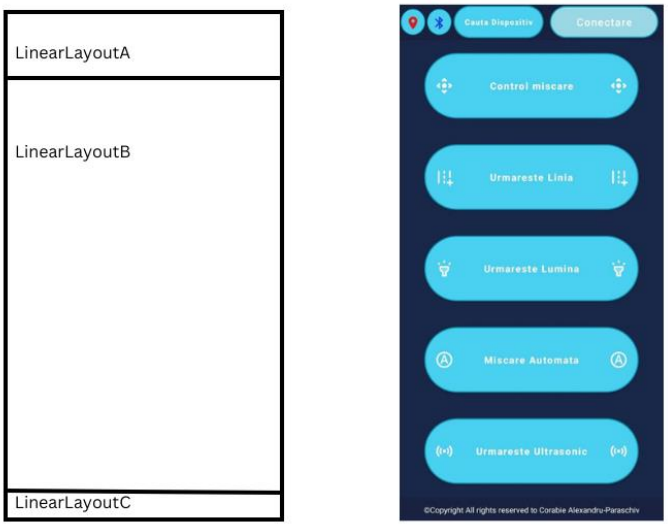
După instalarea aplicației, utilizatorul găsește în cadrul meniului de aplicații al acestuia și aplicația de față, numită "ControlMasinaBluetooth" (aplicația deține o iconiță sugestivă, care să ne ducă la ideea de control al mașinii).

La prima rulare a aplicației, utilizatorul primește o serie de mesaje de cerere a anumitor permisiuni în care este informat cu privire la necesitatea permisiunilor.


**Figură 27** Permisiuni cerute utilizatorului

În figura 27, se regăsesc permisiunile cerute utilizatorului, în vederea utilizării conforme a aplicației. Apariția acestor cereri de permisiune este dată de elementele de tip *alert dialog* sau alertă de dialog, care câștigă focusul aplicației și constă într-un mesaj explicativ menționat mai sus și două opțiuni. La apăsarea butoanelor pozitive din *alert dialog*, vor lansa pe ecran cererile de permisiune generate de sistemul de operare Android. Aceste permisiuni se referă la oferirea accesului la locația exactă, precum și "permisiune dispositive din apropiere", ultima menționată se referă la capacitatea de a se conecta și a căuta dispositive Bluetooth din aria telefonului.

După acordul sau dezacordul utilizatorului cu privire la permisiunile necesare, persoana va avea acces la interfața propriu-zisă a meniului principal (totuși pentru ca utilizatorul să se bucure de funcționalitatea aplicației, este necesar ca acesta să permită accesul la conexiunea bluetooth și la locație).




**Figură 28** Layout-ul meniului principal

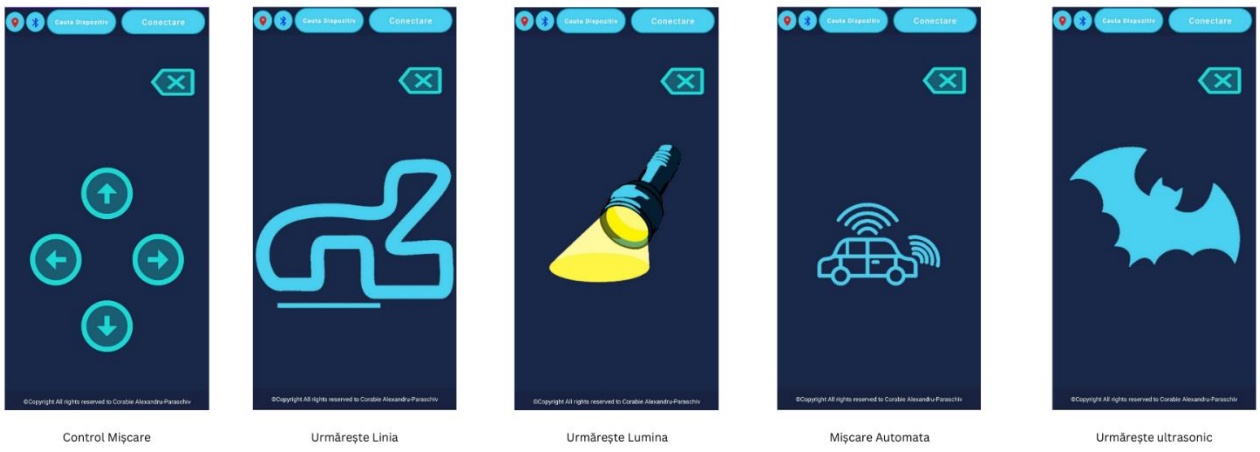
Așa cum s-a menționat anterior, aplicația rulează în activitatea principală. Acest *layout* este segmentat în trei părți (copii). În figura 28 se poate observa împărțirea activității în cele trei *layout-uri* de tip *linear*. LinearLayoutA reprezintă secțiunea de opțiuni necesare conectării și de activare a locației și/ sau a bluetooth-ului. În partea de LinearLayoutB s-a folosit un element de tip "fragmentContainerView" prin care se atașează fragmentele de activitatea principală, reprezentând cea mai mare parte a interfeței grafice. Acest *container* necesită cel puțin un fragment pe care să-l încarce, fiind fragmentul "FragmentMeniuPrincipal". În LinearLayoutC se regăsește o notație de tip *copyright* al aplicației. Pe tot parcursul nagivării în meniul aplicației și funcționalităților mașinii, părțile A și C ale *layout-ului* rămân statice.

În fișierul de resurse al aplicației, se mai găsește definirea temei aplicației, unde s-a definit pe eticheta *style* ca temă a aplicației părinte "Theme.MaterialComponents.Light.NoActionBar". Terminația ".NoActionBar" indică faptul că tema aplicației exclude bara de acțiune. Aceasta ocupa o parte considerabilă din mărimea *layout-ului* și preciza doar numele aplicației, luându-se decizia de excludere a acesteia. Se poate vedea în următoarea linie de cod inclusă în "themes.xml":

```
<style name="Theme.BluetoothAplicatie"
parent="Theme.MaterialComponents.Light.NoActionBar"></style>
```

După ce utilizatorul a vizualizat meniul principal, acesta poate începe procesul de conectare. Procesul de conectare începe prin apăsarea butonului "Cauta Dispozitiv", ce are ca rol verificarea împerecherii telefonului cu modulul Bluetooth. De asemenea, acesta verifică și dacă modulul Bluetooth este pornit, precum și dacă acesta se află în aria de scanare. Dacă aceste condiții au fost îndeplinite, atunci butonul "Conectare" trece din starea inactivă (fals) în starea activă (pozitiv), stare în care acesta poate fi accesat. În momentul în care utilizatorul are ca și eveniment de intrare apăsarea butonului, începe procesul de conectare. Indiferent dacă conectarea este realizată cu succes sau nu, el primește *feedback* printr-un mesaj de tip *toast*. Acest mesaj este afișat în partea de jos a ecranului și dispare după o scurtă perioadă de timp. Indiferent de fragmentul în care utilizatorul se află în momentul conectării cu succes, acesta este trimis către meniul principal și textul butonului din "Conectare" devine "Deconectare".

Design-ul fragmentelor este asemănător, cu diferența că în realizarea sa s-au utilizat diferite imagini reprezentative, cât și butoane (în cazul fragmentului "Control Miscare"). Toate fragmentele prezintă un buton de "înapoi", ce are rolul de a reveni la meniul principal în cazul în care acest aspect este dorit de utilizator.




**Figură 29** Funcțiile de control ale mașinii în interfața utilizatorului

### 2.2.4 Clase și secvențe de cod

În vederea trecerii anumitor secvențe de cod importante, este necesară explicarea metodelor create în cadrul proiectului.Pentru început, vom vorbi despre metodele create cu scopul de a gestiona permisiunile utilizatorului.

Prima metodă este reprezentată de “cererePermisiuneBluetooth”, care are rolul de a oferii aplicației permisiuni de utilizare bluetooth, permisiuni ce sunt oferite de către utilizator. Metoda este apelată în “onStart()”, “onStart()” fiind apelată atât la rularea aplicației, cât și atunci când aplicația revine din metoda “onStop()” (atunci când utilizatorul decide să utilizeze dispozitivul în alt scop, lăsând aplicația rulată în bara de aplicații).

Asemănătoare cu prima metodă menționată, metoda “cererePermisiuneLocatie()” cere permisiunea utilizatorului de a utiliza locația acestuia, necesară pentru descoperirea dispozitivelor din apropiere. La nivel de cod, aceste metode se bazează pe o condiție logică. În interiorul condiției logice, este apelată metoda “shouldShowRequestPermissionRationale”, din cadrul clasei “ActivityCompat”, ce primește ca argumente activitatea la care face referire și tipul de permisiune trecut în fișierul Manifest.xml . Condiția numită verifică dacă utilizatorul a mai refuzat anterior solicitarea acesteia și dacă a fost respinsă, aplicația va afișa o casetă de dialog care va câștiga focus-ul aplicației de tip “AlertDialog”. În caseta de dialog se poate seta contextul (activitatea în care să apară), titlul, mesajul, un buton pozitiv și unul negativ. Dacă butonul pozitiv primește acțiunea “onClick()”, metoda “requestPermissions” din clasa “ActivityCompat” va fi apelată. Aceasta ia ca argumente contextul, o listă de tip *string* în care este trecută permisiunea sau permisiunile din fișierul manifest. Imediat după, aceasta va fi apelată de sistem metoda “onRequestPermissionsResult”, ce a fost suprascrisă cu obiectivul de a afișa un mesaj scurt utilizatorului cu privire la acordul sau refuzul permisiunii. Dacă permisiunea a fost refuzată și a doua oară, atunci cererile intră într-un mod inactiv și trebuie date manual (din setările telefonului).

Alte două metode ce stau la baza funcționalităților aplicației sunt date de metodele “pornesteBluetooth” și “pornesteLocatie”. Metoda “pornesteBluetooth” verifică dacă bluetooth-ul nu este activat. Dacă este acesta este pornit, funcția nu execute nimic. Cum a fost utilizată și anterior, caseta de dialog apare și în aceste cazuri pentru a îi oferi utilizatorului posibilitatea de a accepta sau de a refuza activarea serviciilor.

Metoda „pornesteLocatie” este bazată pe clasele furnizate de librăriile google și prezintă importuri de clase, importate în antetul activității principale [36]. Acestea sunt:

```
import com.google.android.gms.common.api.ResolvableApiException;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.location.LocationSettingsRequest;
import com.google.android.gms.location.LocationSettingsResponse;
import com.google.android.gms.location.Priority;
import com.google.android.gms.location.SettingsClient;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
```

În interiorul clasei "ActivitatePrincipala", înainte de apelul metodei "onCreate()" are loc declarația variabilei "managerLocatie" de tipul "LocationManager". Aceasta este folosită pentru a verifica dacă utilizatorul are locația pornită sau nu. În interiorul metodei se mai crează instanțele claselor "LocationRequest", unde se setează ca atribute prioritatea cererii și intervalul de milisecunde la care cererea locației este cerută. Ulterior, este creată o instanță a clasei "LocationSettingsRequest", căreia ii sunt atribuite cererea de locație inițializată anterior și apelată metoda "setAlwaysShow(true)", metodă ce indică faptul că aplicația are nevoie să furnizeze setările de locație. În următoarele linii de cod, se poate observa inițializarea unui obiect în vederea verificării setărilor de localizare ale utilizatorului.

```
  SettingsClient setariClient = LocationServices.getSettingsClient(this);
```

Cu ajutorul instanțelor anterioare, se crează un obiect cu identificatorul "Task", de tipul "LocationSettingsResponse", numit "task". Utilizând metoda "checkLocationSettings()" a obiectului setariClient și configurându-l cu obiectul "cereSetariLocatie", acest obiect de tip Task va fi utilizat ulterior pentru a gestiona rezultatul verificării setărilor de localizare.

```
Task<LocationSettingsResponse>
task=setariClient.checkLocationSettings(cereSetariLocatie.build());
task.addOnFailureListener(this, new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception exceptie) {
        if(exceptie instanceof ResolvableApiException){
            try {
                ResolvableApiException resolvableApiException =
(ResolvableApiException) exceptie;
resolvableApiException.startResolutionForResult(ActivitatePrincipala.this,
LOCATIE_COD_PORNIRE);}
            catch(IntentSender.SendIntentException sendIntentException){
                sendIntentException.printStackTrace()}}});
```

În cazul “task.addOnFailureListener”, este apelat în momentul în care obiectul de tip “task” nu reușește să recepționeze setările de locație ale telefonului, ceea ce înseamnă că are nevoie să pornească locația. Aspectul menționat anterior se realizează prin inițializarea unei instanțe “ResolvableApiException” care apelează “startResolutionForResult(ActivitatePrincipala.this, LOCATIE_COD_PORNIRE)”. Metoda “startResolutionForResult” are ca scop gestionarea răspunsului utilizatorului la cererea de activare a setărilor de localizare.

Metoda salt la meniul principal a fost creată cu scopul de a trimite utilizatorul la fragmentul meniului principal, în momentul în care socket-ul de conexiune este deschis. Se creează doua instanțe ale obiectelor “Fragment” si “FragmentManager” cu ajutorul cărora se se va face tranziția la meniul principal. Procesul de tranziție se face în urmatoarea linie de cod în care fragmentul se “tranzacționează”, făcându-se referire la id-ul container-ului. Acesta conține fragmentul și instanța clasei “Fragment”, ce face referire la fragmentul dorit prin constructorul său.

```
   FragmentTransactiontranzitieFragment =
fragmentManager.beginTransaction().replace(R.id.fragmentContainerView,
fragmentMeniuPrincipal);

   tranzitieFragment.commit();
```

O altă implementare importantă la nivelul proiectului este implementarea interfeței „FluxIesireDate”.Datele trimise către placa de dezvoltare sunt trimise în cadrul fragmentelor respectivelor funcționalități, iar apoi sunt transferate către activitatea principală, unde se creează *socket-ul* de conexiune. Activitatea principală conține metoda “preiaDateIesire()” de tipul “OutputStream”, care returnează acest tip de dată sub forma variabilei “fluxDateBtIesire”.

În intereriorul fragmetelor, se creează o instanță a clasei “OutputStream” și datele sunt trimise către activitatea principală prin următorul apel al metodei interfeței suprascrisă anterior. Se poate observa la nivel de cod în urmatoarele secvențe.

Secvența “ActivitatePrincipala.java”:

```
@Override

public OutputStream preiaDateIesire() {

    return fluxDateBtIesire;

}
```

Secvență Fragmente:

```
ActivitatePrincipala activitatePrincipala =

            (ActivitatePrincipala).getActivity();

if (activitatePrincipala != null) {

    fluxDateBtIesire = activitatePrincipala.preiaDateIesire();

}
```

Funcția “registerReceiver” este utlizată pentru înregistrarea obiectului de tip “BroadcastReceiver” (adică Btreceiver), pentru a primi și a putea interpreta mesajele emise (*broadcast*), cu ajutorul acțiunilor setate în “intentFilter”. Atunci când se înregistrează un “BroadcastReceiver” cu un “IntentFilter”, acesta devine capabil să recepționeze mesajele care sunt transmise prin intermediul acțiunilor specifice în “intentFilter”. Așadar, atunci când este declanșată o acțiune declarată în “intentFilter” (de exemplu momentul în care se caută modulul bluetooth dorit în lista dispozitivelor împerecheate și este găsit), “Btreceiver” va fi activ și metoda sa “onReceive()” este apelată. [37]

```
registerReceiver(Btreceiver, intentFilter);
```

Linia de cod se mai sus se asigură de faptul că “Btreceiver” primește mesajele de la *broadcast*, mesaje care corespund cu acțiunile din “intentFilter”. Aceste acțiuni sunt:

```
ACTION_DISCOVERY_STARTED, ACTION_DISCOVERY_FINISHED,
ACTION_ACL_DISCONNECTED, ACTION_ACL_CONNECTED, ACTION_FOUND.
```

Butonul “CautaDispozitivBtn” din proiect are rolul de a verifica dacă modulul Bluetooth este împerecheat cu telefonul, pentru a putea iniția procesul de conectare. Dacă permisiunile au fost acordate, atunci adaptorul bluetooth începe căutarea dispozitivelor.

Toate dispozitivele împerecheate se salvează într-o listă de dispozitive unice “<Set>”, și le parcurge căutând dispozitivul cu numele “HC - 05”. Dacă acesta este găsit, numele este salvat în variabila “bluetoothDispozitiv”, iar adaptorul încheie căutarea device-urilor prin metoda cancelDiscovery(). Daca nu este găsit, metoda “cancelDiscovery()” nu este apelată.

```
Set<BluetoothDevice> dispozitiveImperechiate =

                bluetoothAdaptor.getBondedDevices();
```

Metodele “startDiscovery()” și “cancelDiscovery()” joacă un rol important deoarece activează acțiunile din intentFilter, ceea ce înseamnă că metoda suprascrisă “onReceive()” din instanta “Btreceiver” va fi apelată. Daca “cancelDiscovery()” va fi și el apelat, condiția if din interiorul metodei va seta butonul “conectareBtn” ca true (acțiunea se efectuează pe *threadu-ul* interfeței utilizatorului).

La nivel de cod, partea de conexiune se face în interiorul butonului “ConectareBtn”. Atât conectarea cât și deconectarea necesită resurse destul de multe și pentru a nu bloca interfața utilizatorului, în momentul în care se creează conexiunea, aceste acțiuni se realizează pe un *thread* separat [38]. Acest principu se aplică și metodei “onReceive()” menționate mai sus.

```
new Thread(new Runnable() {

      @Override

      public void run() { ..... }

}).start();
```

Sintaxa inițializării unui fir de execuție nou arată ca și secvența de cod scrisă mai sus. În interiorul obiectului “Thread” este prezentă o interfață funcțională de tip *Runable*, ce definește și suprascrie metoda “run()”.

În interiorul metodei “run()”, se scrie codul ce se vrea executat pe threadul nou creat, iar apoi obiectul “Thread” este închis și i se atribuie apelul metodei “start()” ce porneste execuția acestuia.

```
BluetoothSocket =
bluetoothDispozitiv.createRfcommSocketToServiceRecord(UUID.
        fromString("00001101-0000-1000-8000-00805F9B34FB"));
bluetoothSocket.connect();
ConectareBtn.post(new Runnable() {
    @Override
    public void run() {
        ConectareBtn.setText("Deconectare");
    }
});
boolConexiuneReusita = true;
fluxDateBtIesire = bluetoothSocket.getOutputStream();
saltLaMeniulPrincipal();

runOnUiThread(new Runnable() {
    @Override
    public void run() {
        Toast.makeText(getApplicationContext(), "Conectat la HC-05!",
Toast.LENGTH_SHORT).show();
    }
});
```

În interiorul rulării threadu-ului, se utilizează clasa “runOnUIThread”, ce are ca scop trimiterea către thread-ul principal ( raspunzător de interfața grafică) un mesaj scurt de tip toast. Sintaxa de scriere a acestuia e foarte similară cu cea a obiectului “Thread”.

Schimbarea textului butonului de conectare se face într-un mod similar, instanța butonului la conectare apelează metoda “post()”, metodă care setează textul în “Deconectare”. Acest lucru este necesar deoarece acțiunea trebuie trimisă către interfața utilizatorului.

În apelul metodei “onStart()” din cadrul activității principale, se verifică dacă utilizatorul are permisiunile necesare. Dacă nu, metodele explicate anterior “cererePermisiuneBluetooth()” și “cererePermisiuneLocatie()” sunt apelate.

Scopul liniilor de cod din metoda “onDestroy” este acela de a închide socket-ul de conexiune, dacă aceasta rămâne deschis la intrarea activității în procesul de dezalocare. Acest lucru s-a implementat deoarece pe parcursul dezvoltării aplicației după ce conexiunea era activa, mai existau unele crash-uri ale aplicației. S-a realizat aspectul menționat anterior și în cazurile în care aplicația era închisă și socket-ul rămânea deschis pe modulul bluetooth, chair dacă clientul se deconecta. Acest fapt necesită resetarea manuală a plăcii de dezvoltare Arduino sau a modulului bluetooth, lucru nedorit pentru un produs final calitativ.

Pe partea de trimitere de date din cadrul fragmentelor răspunzătoare de funcționalitățile mașinii, se pot exemplifica urmatoarele două tipuri de secvențe de cod.

Un prim exemplu este trimiterea datelor de tip “char” într-un listener, ce conferă butoanelor acțiuni în cadrul evenimentelor de “onClick” și de “onLongClick”. Butoanele cărora li s-a atribuit evenimentul de apăsare lungă, sunt prezente în fragmentul de control al mișcării, în vederea realizării unei mișcări cursive a mașinii.

```
DownBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        try{
            char date = '2';
            fluxDateBtIesire.write((date + "\r \n").getBytes());
            System.out.println(date);
        }catch(Exception e){}
    }
});
```

```
DownBtn.setOnLongClickListener(new View.OnLongClickListener() {
    @Override
    public boolean onLongClick(View v) {
        try{
            char date = '8';
            fluxDateBtIesire.write((date + "\r \n").getBytes());
            System.out.println(date);
        }catch(Exception e){}
        return false;
    }
});
```

Un al doilea exemplu este trimiterea datelor la deschiderea fragmentelor într-o instrucțiune cu un număr necunoscut de pași. Aceasta instrucțiune “while” este încheiată la momentul apăsării butonului care readuce utilizatorul la meniul principal, amplasat în fiecare fragment.

Trimiterea datelor se face în acest caz într-un thread separat. În momentul testării aplicației nu au fost întâmpinate probleme de încărcare a memoriei telefonului. Această situație trebuie luată în considerare deoarece se face transmiterea datelor într-o buclă. Datele având o dimensiune mică, acestea nu consumă resurse mari din performanța unui telefon cu specificații modeste și nu alterează calitatea experienței utilizatorului cu aplicația.

```
char date = '$';
new Thread(new Runnable() {
    @Override
    public void run() {
        try {
            boolean bucla = true;
            while(bucla){
                fluxDateBtIesire.write((date + "\r \n").getBytes());
                System.out.println(date);
                Thread.sleep(50);
                if(InapoiBtn.isPressed() ){
                    break;
                }
            }
        }catch(Exception e){}
    }
}).start();
```

# 3 LIMITE DE PROIECTARE ȘI DIRECȚII VIITOARE

## 3.1 Limite de proiectare

Ca în orice proiect de început, există anumite limite de proiectare care pot periclita buna funcționare a acestuia. Prin evidențierea lor se dorește îmbunătățirea viitoarelor proiecte asemănătoare, dar și evitarea acestor eroi sau disfuncționalități.

În primul rând, în vederea proiectării celor patru funcționalități, s-au utilizat o serie de senzori care au ocupat un număr considerabil de intrări/ ieșiri analogice și digitale. Un număr redus de pini disponibili pot duce la imposibilitatea implementării unor noi funcții și trebuie avut în vedere o eventuală plăcuță de dezvoltare cu mai mulți GPIO (*General Purpose Input/Output*). Un exemplu de plăcuță de dezvoltare care este dotată cu mai mulți pini, o reprezintă plăcuța Arduino Mega 2560, care prezintă 54 de pini digitali și 16 pini analogici. Având în alcătuire un număr considerabil de pini, putem afirma faptul că este o plăcuță potrivită pentru majoritatea proiectelor care vor să ridice standardul de complexitate.

În al doilea rând, senzorul REES52 IR oferă valori eronate în cazul în care aceștia iau contact cu razele soarelui. Razele intense ale soarelui pot conține o cantitate semnificativă de radiații infraroșii, iar acestea pot să înșele capacitatea de captare a senzorilor. Probabil este de preferat căutarea unui senzor cu capacități asemănătoare, dar mai puțin sensibil la razele soarelui.

În al treilea rând, senzorul ultrasonic HC-SR04 măsoară eronat sau cu întârziere distanța atunci când în fața sa se află colțul ascuțit al obiectelor sau acestea prezintă un aspect rotunjit. Când vine vorba de obiecte ascuțite, undele sonore pot fi reflectate în direcții diferite, ceea ce poate duce la o măsurare eronată sau la o întârziere în detectarea distanței corecte. Există posibilitatea ca undele sonore să fie dispersate într-o altă direcție și să nu revină direct la senzor. Un exemplu în care mașinuța a întâmpinat dificultăți la întâlnirea cu anumite obiecte ascuțite îl reprezintă plasarea unei cutii în fața acesteia. Mașinuța se oprește foarte aproape de cutie, uneori lovind-o. În ceea ce privește aspectul rotunjit al obiectelor, undele sonore pot fi absorbite sau dispersate într-o măsură mai mare decât la obiectele plane sau cu suprafețe netede. Acest lucru poate afecta capacitatea senzorului de a detecta cu precizie distanța până la obiect. Un exemplu concret îl reprezintă plasarea unei mingi în fața mașinuței.

În altă ordine de idei, construcția propriu-zisă a mașinuței prezintă și ea aspecte deficitare. Aici putem să vorbim despre amplasarea senzorilor de pe partea din față la o elevație mai mică de 6 cm. Prin urmare, mașina se poate lovii de obiecte ce sunt mai mici de înălțimea anterior precizată. Tot în ceea ce privește construcția mașinii, amplasarea plăcuței de dezvoltare în interiorul acesteia face accesul cablului de programare USB mai dificil, precum și verificarea sau modificarea anumitor aspecte din

cadrul conectării cablurilor. Având în vedere că proiectul prezent face referire la o mașinuță controlată prin bluetooth, amplasarea plăcuței poate să reprezinte o limită care nu ar putea să fie modificată.

## 3.2 Direcții viitoare

În vederea imbunătățirii actualului proiect, au fost stabilite câteva direcții viitoare.

Crearea unei aplicații care răspunde la toate mediile de utilizare reprezintă un prim aspect de îmbunătățit. La momentul actual, interfața de utilizator a aplicației are un amplasament bun doar pe dispozitive tip telefon. Se poate realiza o îmbunătățire prin adaptarea codului de la limbajul Java la limbajul Dart. Limbajul Dart este considerat un limbaj *cross-platform,* deoarece codul rulat pe acesta funcționează pe mai multe sisteme de operare și platforme (precum Linux, iOS, macOS, Android șamd...). Acesta face parte din framework-ul *flutter*, care este popular prin faptul că oferă o versatilitate ridicată și la crearea interfeței utilizatorului (proiectele rulate pot fi adaptate cu mare ușurință la aproape toate formatele de rezoluție).

Implementarea unei funcții de *auto-parking* reprezintă o altă idee pentru un posibil viitor proiect. Această funcție poate să implice instalarea și adăugarea unui nou senzor ultrasonic HC-SR04, dar de această dată în partea din spate a mașinii. Se pot configura diferit și senzorii deja existenți pentru a putea parca într-un loc liber dintre două mașini. Acesta poate măsura distanța dintre mașini și își poate da seama dacă este posibilă realizarea parcării în respectivul loc.

O altă posibilă implementare o reprezintă controlul mașinii cu ajutorul senzorului giroscopic al telefonului. În funcție de înclinarea telefonului, utilizatorul poate să controleze mașina mult mai facil și fără accesarea butoanelor de control din cadrul “Control mișcare”. Chiar dacă pare un aspect relativ inutil (având deja butoanele de control), această funcționalitate oferă o utilizare mult mai facilă și plăcută a mașinii.

În final, o ultimă direcție viitoare o reprezintă montarea unui ventilator mic în interiorul carcasei mașinii. Lipsa ventilatorului poate să constituie și o limită, întrucât suprasolicitarea (prin folosire îndelungată, într-o singură sesiune) punții H duce la ridicarea temperaturii plăcii de expansiune și de dezvoltare. Suprasolicitarea punții H poate duce la deteriorarea materialului din care a fost fabricat, la afectarea conexiunilor electrice din interiorul său (de exemplu prin scurtcircuitare), precum și la apariția diverselor disfuncționalități în ceea ce privește funcționarea corectă a componentelor și a funcționalităților implementate.

# 4 CONCLUZII

În acest proiect, s-a realizat o mașinuță controlată de la distanță, cu ajutorul unui dispozitiv, prin intermediul conexiunii bluetooth. Proiectul a fost compus dintr-o serie de funcționalități și s-a propus atingerea puncte esențiale în cadrul dezvoltării sale. S-au utilizat două medii de dezvoltare și anume Arduino IDE și Android Studio, fiind unele dintre cele mai utilizate și facile medii de dezvoltare din industria *software*.

Mașinuța a fost alcătuită dintr-o plăcuță Arduino, o extensie compatibilă cu aceasta, o serie de senzori, precum și o machetă reutilizată dintr-o mașină de jucărie din comerț, fiindu-i scoase toate piesele care au alcătuit-o. Macheta (mașinuța din plastic) a fost modificată în așa fel încât să ofere un spațiu (pe cât posibil) acceptabil pentru amplasarea celor menționate anterior. Materialele electronice au fost alese din mai multe puncte de vedere, precum raportul calitate-preț al acestora, dar și funcționalitatea lor demonstrată de literatura de specialitate.

Aplicația pe telefon a constat în crearea unui mediu permisiv conectării și menținerii conexiunii stabile, în scopul realizării unei transmisii de date conforme. Aplicația prezintă interfața utilizatorului, care este alcătuită din diverse elemente precum meniul principal, secțiuni separate și mesaje de informare/ cerințe de permisiuni.

De-a lungul perioadei de implementare, au fost întâmpinate diverse erori care au fost remediate sau înlocuite. La început, s-a dorit ca aplicația Android să funcționeze pe bază de “activități”, însă în momentul tranziției între activități acestea nu rulau în fundal, fapt pentru care conexiunea era întreruptă și astfel s-a decis utilizarea “fragmentelor”. Totodată, setarea imaginilor și realizarea formei pe butoanele de pornire bluetooth și locație a fost mai dificilă întrucât elementele de tip “Button” furnizate de Android prezintă erori la un anumit interval nestabilit, fiind necesară implementarea unor elemente de acest tip din cadrul altei biblioteci.

În final, proiectul a ajuns în punctul dorit de la bun început. Ambele componente ale proiectului comunică corect una cu cealaltă, completându-se perfect în vederea realizării unei experiențe plăcute pentru posibili utilizatori. Partea de direcții viitoare prezintă posibile funcționalități ce pot ajuta atât proiectul de față din punct de vedere al componenței sale, cât și experiența persoanelor ce le utilizează.

# BIBLIOGRAFIE

[Dew23] - Dewi, Ayu Sandra, et al. "Developing tutorial video for supporting distance learning on basic automotive control." *AIP Conference Proceedings*. Vol. 2671. No. 1. AIP Publishing, 2023.

[Cha16] - Chakraborty, Debarun, et al. "Android application based monitoring and controlling of movement of a remotely controlled robotic car mounted with various sensors via bluetooth." *2016 International Conference on Advances in Electrical, Electronic and Systems Engineering (ICAEES)*. IEEE, 2016.

[Sho] - Shoeb, Mohammed, et al. "Self-Driving Car: Using Opencv2 and Machine Learning." *The International journal of analytical and experimental modal analysis (IJAEMA), ISSN* 0886-9367.

[Als19] - Alshehri, Faris, et al. "Smart parking system for monitoring cars and wrong parking." *2019 2nd International Conference on Computer Applications & Information Security (ICCAIS)*. IEEE, 2019.

[Had20] - Hadi, Haneen Abdulhussein. "Line Follower Robot Arduino (using robot to control Patient bed who was infected with Covid-19 Virus)." *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*. IEEE, 2020.

# REFERINȚE WEB


[1] https://en.wikipedia.org/wiki/Arduino

[2] https://en.wikipedia.org/wiki/Arduino_Uno#cite_note-princeton-4

[3] https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf

[4] https://content.arduino.cc/assets/UNO-TH_Rev3e_sch.pdf

[5] https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf

[6] https://dronebotworkshop.com/dc-motors-l298n-h-bridge/

[7] https://docs.arduino.cc/learn/microcontrollers/analog-output

[8] https://www.adafruit.com/product/3777

[9] https://katgates.com/assets/uploads/files/412_ARDUINO_SENSOR_LINE_TRACKING_KY-033.pdf

[10] https://www.fierceelectronics.com/sensors/what-ir-sensor

[11] https://5.imimg.com/data5/YT/KV/MY-1833510/arduino-ir-infrared-obstacle-avoidance-sensor-module.pdf

[12] https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

[13] https://ullisroboterseite.de/projekte-teedipper/SG90%209%20g%20Micro%20Servo.pdf

[14] https://www.arduino.cc/reference/en/libraries/servo/

[15] https://www.electrothinks.com/2023/05/ky-018-photoresistor-module.html

[16] https://datasheetspdf.com/pdf/1402029/Joy-IT/KY-018/1

[17] https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter

[18] https://components101.com/wireless/hc-05-bluetooth-module

[19] https://developer.electricimp.com/resources/uart

[20] https://roboticsbackend.com/arduino-setup-loop-functions-explained/

[21] https://www.arduinolibraries.info/libraries/servo

[22] https://reference.arduino.cc/reference/cs/language/functions/communication/serial/

[23] https://reference.arduino.cc/reference/tr/language/functions/time/delay/

[24] https://en.wikipedia.org/wiki/Threshold_voltage

[25] https://en.wikipedia.org/wiki/Speed_of_sound

[26] https://studymind.co.uk/notes/calculating-distance-travelled/#:~:text=To%20calculate%20distance%20travelled%20in,to%20calculate%20the%20distance%20travelled.

[27] https://en.wikipedia.org/wiki/Android_Studio

[28] https://en.wikipedia.org/wiki/Java_(programming_language)

[29] https://www.alibabacloud.com/tech-news/technology/giqt3nalkw-android-advanced-fragment-vs-activity#:~:text=Fragment%20and%20Activity%20are%20two,the%20Fragments%20and%20their%20interactions.

[30] https://medium.com/@w.fauzia009/activity-vs-fragment-lifecycle-6de1935a0432

[31] https://www.informit.com/articles/article.aspx?p=2168079&seqNum=4

[32] https://learn.microsoft.com/en-us/xamarin/android/platform/fragments/creating-a-fragment

[33] https://en.wikipedia.org/wiki/Bluetooth

[34] https://en.wikipedia.org/wiki/Universally_unique_identifier

[35] https://developer.android.com/reference/android/bluetooth/BluetoothSocket

[36] https://developer.android.com/training/location

[37] https://developer.android.com/guide/components/broadcasts

[38] https://developer.android.com/reference/java/lang/Thread

# A. CODUL SURSĂ ARDUINO

```
#include <Servo.h>

#define pinFotoSt A0
#define pinIrSt A1
#define pinIrDr A2
#define pinFotoDr A3

// UART
// pinRX = 0
// pinTX = 1
int pinDirSt = 2;
int pinDirDr = 4;
int pinPWMSt = 5;
int pinPWMDr = 6;
int pinIrUlSt = 7;
int pinIrUlCn = 8;
int pinIrUlDr = 9;
int pinServo = 10;
int pinEcho =12;
int pinTrigger=13;

int d1 = 200;
int d2 = 500;
int d3 = 1000;
int d4 = 1200;

byte dateSeriale;

Servo servo;

float pingTimpRaspuns=0;
float pingDistantaRaspuns=0;
float distantaObstacol=0;

int distantaMica = 10;
int distantaMedie = 19;
int distantaMare = 23;
int distantaDeOprire = 60;

float distantaInFata = 0;
float distantaInStanga = 0;
float distantaInDreapta = 0;
```

```
bool InfrarosuDr = false;
bool InfrarosuSt = false ;

float fotoSt = 0;
float fotoDr = 0;

float scalar = 0;


void setup() {

  Serial.begin(9600);

  pinMode(pinFotoSt, INPUT);
  pinMode(pinFotoDr, INPUT);

  pinMode(pinIrSt,INPUT);
  pinMode(pinIrDr,INPUT);


  pinMode(pinIrUlSt, INPUT);
  pinMode(pinIrUlCn, INPUT);
  pinMode(pinIrUlDr, INPUT);

  pinMode(pinEcho, OUTPUT);
  pinMode(pinTrigger, INPUT);

  servo.attach(pinServo);
  servo.write(120);// centru

  pinMode(pinDirSt, OUTPUT);
  pinMode(pinDirDr, OUTPUT);
  pinMode(pinPWMSt, OUTPUT);
  pinMode(pinPWMDr, OUTPUT);
}

void loop() {


  if(Serial.available() > 0){

    dateSeriale = Serial.read();
    //fata
    if(dateSeriale == 49){ // 1 in ascii
      inFata();
      dateSeriale = "\n";
```

```
  dateSeriale = "\0";
  delay(d3);
  opresteMiscare();
  Serial.print(dateSeriale);
}

//fata lung
if(dateSeriale == 55){ // 7 in ascii
  inFata();
  dateSeriale = "\n";
  dateSeriale = "\0";
}

//spate
if(dateSeriale == 50){ // 2 in ascii
  inSpate();
  dateSeriale = "\n";
  dateSeriale = "\0";
  delay(d3);
  opresteMiscare();
}

if(dateSeriale == 56){ // 8 in ascII
  inSpate();
  dateSeriale = "\n"
  dateSeriale = "\0";
}

if(dateSeriale == 51){ // 3 in ascii
  inStanga();
  dateSeriale = "\n";
  dateSeriale = "\0";
  delay(700);
  opresteMiscare();
}

if(dateSeriale == 54){ // 6 in ascii

  inStanga();
  dateSeriale = "\n";
  dateSeriale = "\0";
}

if(dateSeriale == 52){ //4 in ascii
  inDreapta();
  dateSeriale = "\n";
dateSeriale = "\0";
```

```
    delay(700);
    opresteMiscare();
    }

    if(dateSeriale == 53){
    inDreapta();
    dateSeriale = "\n";
    dateSeriale = "\0";
    }

    if(dateSeriale == 36){//  $ in ascii
      urmarireBandaInfrosu();
      dateSeriale = "\0";
    }

    if(dateSeriale == 35){// #  in ascii
      miscareAutonoma();
      dateSeriale = "\0";

    }

    if(dateSeriale == 37 ){// &  in ascii
      urmaresteLumina();
      dateSeriale = "\0";

    }

    if(dateSeriale == 38 ){// %  in ascii
    urmaresteUltrasonic();
    dateSeriale = "\0";

    }

    if(dateSeriale == 64){ // @ in ascii
      opresteMiscare();
      dateSeriale= "\n";
      dateSeriale = "\0";
    }

  }
dateSeriale= "\n";
}



void urmarireBandaInfrosu(){
```

```
  bool peLinie = 0;
  bool stangaSenzorInf = !digitalRead(pinIrUlSt);
  bool centruSenzorInf = !digitalRead(pinIrUlCn);
  bool dreaptaSenzorInf = !digitalRead(pinIrUlDr);



  if( stangaSenzorInf != peLinie && centruSenzorInf != peLinie &&
dreaptaSenzorInf == peLinie){//caz 2
    inDreapta();
  }
  else if( stangaSenzorInf != peLinie && centruSenzorInf == peLinie &&
dreaptaSenzorInf == peLinie){//caz 3
    inDreaptaRapid();
  }
  else if( stangaSenzorInf == peLinie && centruSenzorInf != peLinie &&
dreaptaSenzorInf == peLinie){//caz 4
    inFata();
  }
  else if( stangaSenzorInf == peLinie && centruSenzorInf == peLinie &&
dreaptaSenzorInf != peLinie){//caz 5
    inStangaRapid();
  }
  else if( stangaSenzorInf == peLinie && centruSenzorInf != peLinie &&
dreaptaSenzorInf != peLinie){//caz 6
    inStanga();
  }
  else if( stangaSenzorInf != peLinie && centruSenzorInf == peLinie &&
dreaptaSenzorInf != peLinie){//caz 7
    inFata();
  }


}


void miscareAutonoma(){
  InfrarosuSt = !digitalRead(pinIrSt);
  InfrarosuDr = !digitalRead(pinIrDr);
  Serial.println(InfrarosuSt);
  Serial.println(InfrarosuDr);
  distantaInFata = verificaDistanta();
  Serial.println(distantaInFata);
  evitareInfrarosuLateral();
  evitareUltrasonica();
}
```

```
float verificaDistanta() {

  digitalWrite(pinEcho, LOW);
  delayMicroseconds(10);
  digitalWrite(pinEcho, HIGH);
  delayMicroseconds(10);
  digitalWrite(pinEcho, LOW);

  pingTimpRaspuns=pulseIn(pinTrigger,HIGH);
  delay (25);
  pingDistantaRaspuns = (pingTimpRaspuns * 0.0350);
  distantaObstacol = pingDistantaRaspuns/2;
  return distantaObstacol;
}

 void servoRotireCapturaDistanta(){

  servo.write(120);// mijloc
  delay(d2);
  servo.write(60);// dreapta
  delay(d2);
  Serial.print("Distanta in dreapta: ");
  distantaInDreapta = verificaDistanta();
  Serial.print(distantaInDreapta);
  Serial.println("cm");
  delay(d2);
  servo.write(120);
  delay(d2);
  servo.write(180);// stanga
  Serial.print("Distanta in stanga: ");
  distantaInStanga = verificaDistanta();
  Serial.print(distantaInStanga);
  Serial.println("cm");
  delay(d2);
  servo.write(120);
  delay(d2);


}


void evitareInfrarosuLateral(){
  if(InfrarosuSt == 0 && InfrarosuDr == 1){
    dreaptaEvitare();
  }else if(InfrarosuSt == 1 && InfrarosuDr == 0){
    stangaEvitare();
  }else if(InfrarosuSt == 1 && InfrarosuDr == 1){
```

```
    inStangaRapid();
  }else if(InfrarosuSt == 0 && InfrarosuDr == 0){
    inFata();
  }
}

void evitareUltrasonica(){
  if(distantaInFata <= distantaMedie){

    opresteMiscare();

    if(distantaInFata <= distantaMica ){
      inSpate();
      delay(300);
      opresteMiscare();
    }
    servoRotireCapturaDistanta();
    if((distantaMica < distantaInStanga && distantaInStanga < distantaMare) &&
(distantaMica < distantaInDreapta && distantaInDreapta < distantaMare)){

      if(distantaInDreapta > distantaInStanga){
        inDreaptaRapid();
        delay(350);
      }else if(distantaInDreapta < distantaInStanga){
        inStangaRapid();
        delay(350);
      }

    }else if( distantaMica < distantaInStanga && distantaInStanga <
distantaMare || distantaMica < distantaInDreapta && distantaInDreapta <
distantaMare ){
      if(distantaMica < distantaInStanga && distantaInStanga < distantaMare){
          inDreaptaRapid();
          delay(350);
      }else if(distantaMica < distantaInDreapta && distantaInDreapta <
distantaMare){
          inStangaRapid();
          delay(350);
      }
    }else if(distantaInDreapta < distantaMica && distantaInStanga <
distantaMica){
    inDreaptaRapid();
    delay(d4);
    }
    opresteMiscare();
  }
```

```
}

void urmaresteUltrasonic(){
  InfrarosuSt = !digitalRead(pinIrSt);
  InfrarosuDr = !digitalRead(pinIrDr);
  distantaInFata = verificaDistanta();
  Serial.println(distantaInFata);

  if(distantaInFata < distantaMica && (InfrarosuSt == 0 && InfrarosuDr == 0)){
    inSpate();
  }else if(distantaInFata < distantaMica && (InfrarosuSt == 1 && InfrarosuDr
== 1)){
    inSpate();
  }else if(distantaInFata < distantaMica && (InfrarosuSt == 1 && InfrarosuDr
== 0)){
    inSpateDreapta();
  }else if(distantaInFata < distantaMica && (InfrarosuSt == 0 && InfrarosuDr
== 1)){
    inSpateStanga();
  }else if((distantaInFata < distantaMare && distantaInFata > distantaMedie)
&& (InfrarosuSt == 0 && InfrarosuDr == 0)){
    inFataIncet();
  }else if((distantaInFata < distantaMare && distantaInFata > distantaMedie)
&& (InfrarosuSt == 1 && InfrarosuDr == 1)){
    opresteMiscare();
  }else if((distantaInFata < distantaMare && distantaInFata > distantaMedie)
&& (InfrarosuSt == 1 && InfrarosuDr == 0)){
    inStanga();
  }else if((distantaInFata < distantaMare && distantaInFata > distantaMedie)
&& (InfrarosuSt == 0 && InfrarosuDr == 1)){
    inDreapta();
  }else if((distantaMica > distantaInFata && distantaInFata < distantaMedie)
&& (InfrarosuSt == 1 && InfrarosuDr == 1)){
    opresteMiscare();
  }else if((distantaMica > distantaInFata && distantaInFata < distantaMedie)
&& (InfrarosuSt == 0 && InfrarosuDr == 0)){
    inFataIncet();
  }else if((distantaMica > distantaInFata && distantaInFata < distantaMedie)
&& (InfrarosuSt == 1 && InfrarosuDr == 0)){
    inStanga();
  }else if((distantaMica > distantaInFata && distantaInFata < distantaMedie)
&& (InfrarosuSt == 0 && InfrarosuDr == 1)){
    inDreapta();
  }else if((distantaMare < distantaInFata && distantaInFata <
distantaDeOprire) && (InfrarosuSt == 0 && InfrarosuDr == 0)){
    inFataIncet();
```

```
  }else if((distantaMare < distantaInFata && distantaInFata <
distantaDeOprire) && (InfrarosuSt == 1 && InfrarosuDr == 1)){
    opresteMiscare();
  }else if((distantaMare < distantaInFata && distantaInFata <
distantaDeOprire) && (InfrarosuSt == 1 && InfrarosuDr == 0)){
    inStanga();
  }else if((distantaMare < distantaInFata && distantaInFata <
distantaDeOprire) && (InfrarosuSt == 0 && InfrarosuDr == 1)){
    inDreapta();
  }else if((distantaInFata > distantaDeOprire) && (InfrarosuSt == 0 &&
InfrarosuDr == 0)){
    opresteMiscare();
  }else if((distantaInFata > distantaDeOprire) && (InfrarosuSt == 1 &&
InfrarosuDr == 1)){
    opresteMiscare();
  }else if((distantaInFata > distantaDeOprire) && (InfrarosuSt == 1 &&
InfrarosuDr == 0)){
    inStanga();
  }else if((distantaInFata > distantaDeOprire) && (InfrarosuSt == 0 &&
InfrarosuDr == 1)){
    inDreapta();
  }
}

void urmaresteLumina(){

  int vitezaPWM = 40;
  fotoDr = analogRead(pinFotoDr)/10;
  fotoSt = analogRead(pinFotoSt)/10;

  if(fotoSt > 60 && fotoDr > 60){
    opresteMiscare();
  }else{

    if(fotoSt > fotoDr){
      scalar = fotoDr / fotoSt*3;
      digitalWrite(pinDirSt,HIGH);
      analogWrite(pinPWMSt,(scalar*vitezaPWM));
      digitalWrite(pinDirDr,LOW);
      analogWrite(pinPWMDr,vitezaPWM);

      if(scalar < 1){
        inDreapta();
      }


    }else if(fotoSt <= fotoDr) {
```

```
      scalar = fotoSt / fotoDr*3;
      digitalWrite(pinDirSt,HIGH);
      analogWrite(pinPWMSt,vitezaPWM);
      digitalWrite(pinDirDr,LOW);
      analogWrite(pinPWMDr,(scalar*vitezaPWM));

      if(scalar < 1){
        inStanga();
      }
    }
  }
}


void dreaptaEvitare(){
    digitalWrite(pinDirSt,HIGH);
    analogWrite(pinPWMSt,10);
    digitalWrite(pinDirDr,LOW);
    analogWrite(pinPWMDr,200);
}

void stangaEvitare(){
    digitalWrite(pinDirSt,HIGH);
    analogWrite(pinPWMSt,200);
    digitalWrite(pinDirDr,LOW);
    analogWrite(pinPWMDr,10);
}

void inSpate(){
    digitalWrite(pinDirSt,LOW);
    analogWrite(pinPWMSt,100);
    digitalWrite(pinDirDr,HIGH);
    analogWrite(pinPWMDr,100);
}

void inFataIncet(){
    digitalWrite(pinDirSt,HIGH);
    analogWrite(pinPWMSt,85);
    digitalWrite(pinDirDr,LOW);
    analogWrite(pinPWMDr,85);
}


void inSpateDreapta(){
    digitalWrite(pinDirSt,LOW);
    analogWrite(pinPWMSt,50);
    digitalWrite(pinDirDr,HIGH);
```

```
    analogWrite(pinPWMDr,150);
}


void inSpateStanga(){
    digitalWrite(pinDirSt,LOW);
    analogWrite(pinPWMSt,150);
    digitalWrite(pinDirDr,HIGH);
    analogWrite(pinPWMDr,50);
}

void inFataStanga(){

    digitalWrite(pinDirSt,HIGH);
    analogWrite(pinPWMSt,150);
    digitalWrite(pinDirDr,LOW);
    analogWrite(pinPWMDr,50);
}

void inFataDreapta(){
    digitalWrite(pinDirSt,HIGH);
    analogWrite(pinPWMSt,50);
    digitalWrite(pinDirDr,LOW);
    analogWrite(pinPWMDr,150);
}



void inFata(){
    digitalWrite(pinDirSt,HIGH);
    analogWrite(pinPWMSt,100);
    digitalWrite(pinDirDr,LOW);
    analogWrite(pinPWMDr,100);
}

void inStanga(){
  digitalWrite(pinDirSt,LOW);
  analogWrite(pinPWMSt,150);
  digitalWrite(pinDirDr,LOW);
  analogWrite(pinPWMDr,150);
}

void inStangaRapid(){
  digitalWrite(pinDirSt,LOW);
  analogWrite(pinPWMSt,190);
  digitalWrite(pinDirDr,LOW);
  analogWrite(pinPWMDr,190);
```

```
}


void inDreapta(){
  digitalWrite(pinDirSt,HIGH);
  analogWrite(pinPWMSt,150);
  digitalWrite(pinDirDr,HIGH);
  analogWrite(pinPWMDr,150);
}

void inDreaptaRapid(){
  digitalWrite(pinDirSt,HIGH);
  analogWrite(pinPWMSt,190);
  digitalWrite(pinDirDr,HIGH);
  analogWrite(pinPWMDr,190);
}

void opresteMiscare(){
  digitalWrite(pinDirSt,LOW);
  analogWrite(pinPWMSt,0);
  digitalWrite(pinDirDr,LOW);
  analogWrite(pinPWMDr,0);
};
```

# B. CODUL SURSĂ ANDROID STUDIO

## Fisier AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"
/>
    <uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
    <uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
    <uses-permission
android:name="android.permission.ACCESS_BACKGROUND_LOCATION" />

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@drawable/icon_application"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.BluetoothAplicatie"
        tools:targetApi="31">

        <activity

android:name="com.example.aplicatiebluetooth.ActivitatePrincipala"
            android:exported="true"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>

            <meta-data
                android:name="android.app.lib_name"
                android:value="" />
        </activity>
    </application>

</manifest>
```

## Fisier ActivitatePrincipala.java

```
package com.example.aplicatiebluetooth;

import static androidx.constraintlayout.helper.widget.MotionEffect.TAG;

import android.Manifest;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.IntentSender;
import android.content.pm.PackageManager;
import android.location.LocationManager;
import android.os.Build;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;

import com.google.android.gms.common.api.ResolvableApiException;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.location.LocationSettingsRequest;
import com.google.android.gms.location.LocationSettingsResponse;
import com.google.android.gms.location.Priority;
import com.google.android.gms.location.SettingsClient;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;

import java.io.IOException;
import java.io.OutputStream;
import java.util.Set;
import java.util.UUID;

public class ActivitatePrincipala extends AppCompatActivity implements
FluxIesireDate {


    @Override
    public OutputStream preiaDateIesire() {
        return fluxDateBtIesire;
    }
```

```
    private final int BLUETOOTH_COD_PERMISIUNE=80;
    private final int LOCATIE_COD_PERMISIUNE=81;

    private final int LOCATIE_COD_PORNIRE =82;


    Button CautaDispozitivBtn, ConectareBtn, ActivareBtBtn, ActivareLocBtn
;


    BluetoothAdapter bluetoothAdaptor;
    BluetoothDevice bluetoothDispozitiv;
    BluetoothSocket bluetoothSocket;
    IntentFilter intentFiltru;

    OutputStream fluxDateBtIesire;


    boolean boolConexiuneReusita = false;

    private LocationManager managerLocatie;



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



        ActivareBtBtn = findViewById(R.id.btnPermisiuneBT);
        ActivareLocBtn = findViewById(R.id.btnPermisiuneLoc);
        CautaDispozitivBtn = findViewById(R.id.btn1);
        ConectareBtn = findViewById(R.id.btn2);


        bluetoothAdaptor = BluetoothAdapter.getDefaultAdapter();

        intentFiltru = new IntentFilter();
        intentFiltru.addAction(BluetoothAdapter.ACTION_DISCOVERY_STARTED);
        intentFiltru.addAction(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
        intentFiltru.addAction(BluetoothDevice.ACTION_ACL_DISCONNECTED);
        intentFiltru.addAction(BluetoothDevice.ACTION_ACL_CONNECTED);
        intentFiltru.addAction(BluetoothDevice.ACTION_FOUND);

        managerLocatie = (LocationManager)
getSystemService(LOCATION_SERVICE) ;

        registerReceiver(Btreceiver, intentFiltru);



        ConectareBtn.setEnabled(false);
```

```
            ActivareBtBtn.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {

                    if(bluetoothAdaptor == null){

Toast.makeText(ActivitatePrincipala.this,"Dispozitivul nu suporta
BLUETOOTH!",Toast.LENGTH_LONG).show();
                    }else
if((ContextCompat.checkSelfPermission(ActivitatePrincipala.this,
Manifest.permission.BLUETOOTH_CONNECT ) ==
PackageManager.PERMISSION_GRANTED ) &&

(ContextCompat.checkSelfPermission(ActivitatePrincipala.this,
Manifest.permission.BLUETOOTH_SCAN ) == PackageManager.PERMISSION_GRANTED
)){
                        pornesteBluetooth();
                    }else{
                        cererePermisiuneBluetooth();
                    }

                }
            });

            ActivareLocBtn.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {


if(ContextCompat.checkSelfPermission(ActivitatePrincipala.this,
Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED)
                    {
                        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S)
{
                            pornesteLocatie();
                        }else{
                            Toast.makeText(ActivitatePrincipala.this,
"Locatia trebuie pornita manual!",Toast.LENGTH_LONG).show();
                        }
                    }else{
                        cererePermisiuneLocatie();
                    }

                }
            });


            CautaDispozitivBtn.setOnClickListener(new
View.OnClickListener() {
                @Override
                public void onClick(View v) {


if((ContextCompat.checkSelfPermission(ActivitatePrincipala.this,
Manifest.permission.BLUETOOTH_CONNECT ) ==
PackageManager.PERMISSION_GRANTED ) &&

(ContextCompat.checkSelfPermission(ActivitatePrincipala.this,
```

```
Manifest.permission.BLUETOOTH_SCAN ) == PackageManager.PERMISSION_GRANTED )
) {
                        if(bluetoothAdaptor.isEnabled() &&
managerLocatie.isProviderEnabled(LocationManager.GPS_PROVIDER)){
                            bluetoothAdaptor.startDiscovery();
                            Set<BluetoothDevice> dispozitiveImperechiate =
bluetoothAdaptor.getBondedDevices();
                            for (BluetoothDevice hc05Arduino :
dispozitiveImperechiate) {
                                Log.e(TAG,"Nume dispozitive imperechiate:
"+ hc05Arduino.getName());
                                if (hc05Arduino.getName().equals("HC-05"))
{
                                    bluetoothDispozitiv = hc05Arduino;
                                    bluetoothAdaptor.cancelDiscovery();
                                    break;
                                }
                            }
                        }else{
                            builderCautaDispozitiv();
                        }
                    }else{
                        cererePermisiuneBluetooth();

                    }
                }
            });

            ConectareBtn.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {

                    if
(ContextCompat.checkSelfPermission(ActivitatePrincipala.this,
Manifest.permission.BLUETOOTH_CONNECT) == PackageManager.PERMISSION_GRANTED
&&

ContextCompat.checkSelfPermission(ActivitatePrincipala.this,
Manifest.permission.BLUETOOTH_SCAN) == PackageManager.PERMISSION_GRANTED) {
                        if (boolConexiuneReusita == false) {
                            new Thread(new Runnable() {
                                @Override
                                public void run() {
                                    try {
                                        if
(ContextCompat.checkSelfPermission(ActivitatePrincipala.this,
Manifest.permission.BLUETOOTH_CONNECT)
                                                ==
PackageManager.PERMISSION_GRANTED &&

ContextCompat.checkSelfPermission(ActivitatePrincipala.this,
Manifest.permission.BLUETOOTH_SCAN)
                                                        ==
PackageManager.PERMISSION_GRANTED){

                                            bluetoothSocket =
bluetoothDispozitiv.createRfcommSocketToServiceRecord(UUID.fromString("0000
1101-0000-1000-8000-00805F9B34FB"));
                                            bluetoothSocket.connect();
```

```
                                        ConectareBtn.post(new Runnable() {
                                            @Override
                                            public void run() {

ConectareBtn.setText("Deconectare");
                                            }
                                        });
                                        boolConexiuneReusita = true;
                                        fluxDateBtIesire =
bluetoothSocket.getOutputStream();
                                        saltLaMeniulPrincipal();

                                        runOnUiThread(new Runnable() {
                                            @Override
                                            public void run() {

Toast.makeText(getApplicationContext(), "Conectat la HC-05!",
Toast.LENGTH_SHORT).show();
                                            }
                                        });
                                        }else{
                                            cererePermisiuneBluetooth();
                                            cererePermisiuneLocatie();
                                        }
                                    } catch (Exception e) {
                                        boolConexiuneReusita = false;
                                        try{
                                            bluetoothSocket.close();
                                        }catch (Exception exception){
                                            exception.printStackTrace();
                                        }
                                        ConectareBtn.post(new Runnable() {
                                            @Override
                                            public void run() {

ConectareBtn.setText("Conectare");
                                            }
                                        });
                                        runOnUiThread(new Runnable() {
                                            @Override
                                            public void run() {

Toast.makeText(getApplicationContext(), "Nu s-a putut efectua conectarea la
HC-05!", Toast.LENGTH_SHORT).show();
                                            }
                                        });
                                        e.printStackTrace();
                                        Log.e(TAG, "Bluetooth disconnect
exception1" + e.getMessage());
                                    }
                                }
                            }).start();
                        } else {
                            new Thread(new Runnable() {
                                @Override
                                public void run() {
                                    try {
                                        bluetoothSocket.close();
```

```
                                        Log.e(TAG,"Starea socket-ului la
deconectare " + bluetoothSocket.isConnected());
                                        boolConexiuneReusita = false;
                                        ConectareBtn.post(new Runnable() {
                                            @Override
                                            public void run() {

ConectareBtn.setText("Conectare");
                                            }
                                        });
                                        runOnUiThread(new Runnable() {
                                            @Override
                                            public void run() {

Toast.makeText(getApplicationContext(), "Deconectat la HC-05!",
Toast.LENGTH_SHORT).show();
                                            }
                                        });
                                    } catch (IOException e) {
                                        e.printStackTrace();
                                        Log.e(TAG, "Bluetooth disconnect
exception2" + e.getMessage());
                                    }
                                }
                            }).start();
                        }
                    } else {
                        cererePermisiuneBluetooth();
                    }
                }
            });
    }

    @Override
    protected void onStart() {
        super.onStart();

        if((ContextCompat.checkSelfPermission(ActivitatePrincipala.this,
Manifest.permission.BLUETOOTH_CONNECT ) !=
PackageManager.PERMISSION_GRANTED ) &&

(ContextCompat.checkSelfPermission(ActivitatePrincipala.this,
Manifest.permission.BLUETOOTH_SCAN ) != PackageManager.PERMISSION_GRANTED
)) {
            cererePermisiuneBluetooth();
        }

        if(ContextCompat.checkSelfPermission(ActivitatePrincipala.this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
            cererePermisiuneLocatie();
        }
    }


    @Override
    protected void onDestroy() {
        super.onDestroy();
```

```
        if(bluetoothSocket.isConnected()){
            try {
                bluetoothSocket.close();
            }
            catch (IOException e) {
                e.printStackTrace();
                Log.e("Socket", "Socket-ul  bluetooth nu s-a putut inchide
la distrugerea aplicatiei!" + e.getMessage());
            }
        }
    }

    BroadcastReceiver Btreceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            Log.i("Btreceiver", "se executa aici!");
            if
(BluetoothAdapter.ACTION_DISCOVERY_FINISHED.equals(intent.getAction())) {
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        ConectareBtn.setEnabled(true);
                    }
                });
            }
        }
    };

    private void cererePermisiuneBluetooth(){
        if((ActivityCompat.shouldShowRequestPermissionRationale(this,
android.Manifest.permission.BLUETOOTH_CONNECT))
                &&
(ActivityCompat.shouldShowRequestPermissionRationale(this,
Manifest.permission.BLUETOOTH_SCAN)) )
        {
            new AlertDialog.Builder(this)
                    .setTitle("Cerere permisiune bluetooth")
                    .setMessage("Aceasta permisiune bluetooth este necesara
pentru a controla dispozitivul bluetooth!")
                    .setPositiveButton("ok", new
DialogInterface.OnClickListener() {
                        @Override public void onClick(DialogInterface
dialog, int which) {
                            ActivityCompat.requestPermissions(
                                    ActivitatePrincipala.this,
                                    new String[]
{Manifest.permission.BLUETOOTH_CONNECT,Manifest.permission.BLUETOOTH_SCAN},
                                    BLUETOOTH_COD_PERMISIUNE); } })
                    .setNegativeButton("anuleaza", new
DialogInterface.OnClickListener() {
                        @Override public void onClick(DialogInterface
dialog, int which) {
                            dialog.dismiss(); } })
                    .create().show(); }
        else{ ActivityCompat.requestPermissions(this, new
String[]{android.Manifest.permission.BLUETOOTH_CONNECT,

Manifest.permission.BLUETOOTH_SCAN},BLUETOOTH_COD_PERMISIUNE);
        }
```

```
    }

    private void cererePermisiuneLocatie(){
        if(ActivityCompat.shouldShowRequestPermissionRationale(this,
android.Manifest.permission.ACCESS_FINE_LOCATION))
        {
            new AlertDialog.Builder(this)
                    .setTitle("Cerere permisiune locatie")
                    .setMessage("Aceasta permisiune de locatie este
necesara pentru a controla dispozitivul bluetooth!")
                    .setPositiveButton("ok", new
DialogInterface.OnClickListener() {
                        @Override public void onClick(DialogInterface
dialog, int which) {
                            ActivityCompat.requestPermissions(
                                    ActivitatePrincipala.this,
                                    new String[]
{Manifest.permission.ACCESS_FINE_LOCATION},
                                    LOCATIE_COD_PERMISIUNE); } })
                    .setNegativeButton("anuleaza", new
DialogInterface.OnClickListener() {
                        @Override public void onClick(DialogInterface
dialog, int which) {
                            dialog.dismiss(); } })
                    .create().show();
        }else{ ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},LOCATIE_COD_PERMISIUNE);
        }
    }



    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull
String[] permissions, @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
        if (requestCode == BLUETOOTH_COD_PERMISIUNE || requestCode ==
LOCATIE_COD_PERMISIUNE  ) {
            if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                Toast.makeText(this, "Permisiune acordata ",
Toast.LENGTH_SHORT).show();
            }
            else {
                Toast.makeText(this, "Permisiune refuzata",
Toast.LENGTH_SHORT).show();
            }
        }
    }

    private void builderCautaDispozitiv(){
        new AlertDialog.Builder(this)
                .setTitle("Bluetooth sau locatie dezactivate!")
                .setMessage("Pentru a cauta masina este necesara activarea
de locatie si bluetooth! Se poate face mai rapid cu ajutorul butoanelor din
stanga sus.")
                .setNegativeButton("anuleaza", new
DialogInterface.OnClickListener() {
```

```
                    @Override public void onClick(DialogInterface dialog,
int which) {dialog.dismiss(); } })
                .create().show();
    }

    private void pornesteBluetooth(){

        if(!bluetoothAdaptor.isEnabled()){
            new AlertDialog.Builder(this)
                .setTitle("Bluetooth Dezactivat!")
                .setMessage("Este necesar sa porniti bluetooth-ul!")
                .setPositiveButton("accept", new
DialogInterface.OnClickListener() {
                    @Override public void onClick(DialogInterface dialog,
int which) {

if(ContextCompat.checkSelfPermission(ActivitatePrincipala.this,
Manifest.permission.BLUETOOTH_CONNECT ) ==
PackageManager.PERMISSION_GRANTED ){
                            if(!bluetoothAdaptor.isEnabled()){
                                bluetoothAdaptor.enable();
                            }
                        }else{
                            Toast.makeText(ActivitatePrincipala.this,"Este
nevoie sa acordati permisiunile prima data",Toast.LENGTH_SHORT).show();
                        }
                         } })
                .setNegativeButton("anuleaza", new
DialogInterface.OnClickListener() {
                    @Override public void onClick(DialogInterface dialog,
int which) {
                        dialog.dismiss(); } })
                .create().show();
        }else{
            Toast.makeText(ActivitatePrincipala.this,"Bluetooth deja
pornit!",Toast.LENGTH_SHORT).show();
        }
    }

    private void pornesteLocatie(){

        if(managerLocatie.isProviderEnabled(LocationManager.GPS_PROVIDER)){
            Toast.makeText(ActivitatePrincipala.this, "Locatia este deja
pornita!",Toast.LENGTH_SHORT).show();
        }

        com.google.android.gms.location.LocationRequest cereLocatie =  new
com.google.android.gms.location.LocationRequest.Builder(Priority.PRIORITY_H
IGH_ACCURACY)
                .setIntervalMillis(10000)
                .setMinUpdateIntervalMillis(5000)
                .build();

        LocationSettingsRequest.Builder cereSetariLocatie = new
LocationSettingsRequest.Builder();

        cereSetariLocatie.addLocationRequest(cereLocatie);
        cereSetariLocatie.setAlwaysShow(true);
```

```
        SettingsClient setariClient =
LocationServices.getSettingsClient(this);

        Task<LocationSettingsResponse> task =
setariClient.checkLocationSettings(cereSetariLocatie.build());

        task.addOnFailureListener(this, new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception exceptie) {
                if(exceptie instanceof ResolvableApiException){
                    try {
                        ResolvableApiException resolvableApiException =
(ResolvableApiException) exceptie;

resolvableApiException.startResolutionForResult(ActivitatePrincipala.this,
LOCATIE_COD_PORNIRE);
                    }
                    catch(IntentSender.SendIntentException
sendIntentException){
                        sendIntentException.printStackTrace();
                    }
                }
            }
        });

    }

    private void saltLaMeniulPrincipal(){
        Fragment fragmentMeniuPrincipal= new FragmentMeniuPrincipal();
        FragmentManager fragmentManager = getSupportFragmentManager();
        FragmentTransaction tranzitieFragment =
fragmentManager.beginTransaction().replace(R.id.fragmentContainerView,
fragmentMeniuPrincipal);
        tranzitieFragment.commit();
    }


}
```

## Fisier FluxIesireDate.java (interfață)

```
package com.example.aplicatiebluetooth;

import java.io.OutputStream;

public  interface FluxIesireDate
{
    OutputStream preiaDateIesire();
}
```

Fisier activitate_principala.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.appcompat.widget.LinearLayoutCompat
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:weightSum="100"
    tools:context=".ActivitatePrincipala">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="7"
        android:orientation="horizontal"
        android:weightSum="100"
        android:background="@color/albastruInchis">


        <LinearLayout
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="10"
            android:orientation="vertical">

            <com.google.android.material.button.MaterialButton
                android:id="@+id/btnPermisiuneLoc"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_marginHorizontal="1.5dp"
                android:layout_marginVertical="8dp"

android:background="@drawable/resurse_btn_locatie_bluetooth"
                android:drawableTop="@drawable/ic_location_on_24"
                app:backgroundTint="@null"
                >

            </com.google.android.material.button.MaterialButton>

        </LinearLayout>

        <LinearLayout
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="10        "
            android:orientation="vertical"
            >


            <com.google.android.material.button.MaterialButton
                android:id="@+id/btnPermisiuneBT"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
```

```
                android:layout_gravity="center"
                android:gravity="center"
                android:layout_marginHorizontal="1.5dp"
                android:layout_marginTop="8dp"
                android:layout_marginBottom="8dp"

android:background="@drawable/resurse_btn_locatie_bluetooth"
                android:drawableTop="@drawable/ic_bluetooth_enable_24"
                app:backgroundTint="@null">
            </com.google.android.material.button.MaterialButton>

        </LinearLayout>

        <LinearLayout
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="35"
            android:orientation="vertical">

            <Button
                android:id="@+id/btn1"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_gravity="center"
                android:gravity="center"
                android:text="Cauta Dispozitiv"
                android:layout_marginHorizontal="2dp"
                android:layout_marginVertical="3dp"
                android:textAllCaps="false"
                android:textSize="11sp"
                android:textStyle="bold"
                app:backgroundTint="@null"
                android:background="@drawable/resurse_btn_cauta"
                android:textColor="@color/white"/>

        </LinearLayout>


        <LinearLayout
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="45"
            android:orientation="vertical">

            <Button
                android:id="@+id/btn2"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_marginHorizontal="8dp"
                android:layout_marginVertical="3.5dp"
                android:background="@drawable/resurse_btn_conectare"
                app:backgroundTint="@null"
                android:text="Conectare"
                android:textAllCaps="false"
                android:textColor="@color/white"
                android:textSize="15sp" />

        </LinearLayout>
```

```
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="88">


        <androidx.fragment.app.FragmentContainerView
            android:layout_gravity="center"
            android:id="@+id/fragmentContainerView"

android:name="com.example.aplicatiebluetooth.FragmentMeniuPrincipal"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:background="@color/albastruInchis2"
            tools:layout="@layout/fragment_meniu_principal" />
    </LinearLayout>


    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:orientation="horizontal"
        android:layout_weight="5">


        <TextView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:background="@color/albastruInchis"
            android:text="©Copyright All rights reserved to Corabie
Alexandru-Paraschiv"
            android:textAlignment="center"
            android:textColor="@color/white"
            android:layout_gravity="center"
            android:gravity="center"
            android:textSize="12dp">

        </TextView>

    </LinearLayout>

</androidx.appcompat.widget.LinearLayoutCompat>
```

## Fisier FragmentMeniuPrincipal.java

```
package com.example.aplicatiebluetooth;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;

import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentTransaction;

public class FragmentMeniuPrincipal extends Fragment {

    Button FragmentCtrlMiscareBtn,FragmentUrmaresteLinia,
FragmentUrmaresteLuminaBtn, FragmentMiscareAutomataBtn,
FragmentUrmaresteUltrasonicBtn;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {

        View view = inflater.inflate(R.layout.fragment_meniu_principal,
container, false);

        FragmentMiscareAutomataBtn=
view.findViewById(R.id.btnFragmentMiscareAutomata);
        FragmentUrmaresteUltrasonicBtn =
view.findViewById(R.id.btnFragmentUrmaresteUltrasonic);
        FragmentUrmaresteLuminaBtn =
view.findViewById(R.id.btnFragmentUrmaresteLumina);
        FragmentCtrlMiscareBtn=view.findViewById(R.id.btnFrgCtrlMiscare);
        FragmentUrmaresteLinia=view.findViewById(R.id.btnFrgUrmLinia);

        Fragment fragmentCtrlMiscareTranzitie = new
FragmentControlMiscare();
        Fragment fragmentUrmaresteLinia = new FragmentUrmaresteLinia();
        Fragment fragmentUrmaresteLumina = new FragmentUrmaresteLumina();
        Fragment fragmentUrmaresteUltrasonic = new
FragmentUrmaresteUltrasonic();
        Fragment fragmentMiscareAutomata = new FragmentMiscareAutomata();


        FragmentCtrlMiscareBtn.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                FragmentTransaction tranzitieFragment =
getActivity().getSupportFragmentManager().beginTransaction();

tranzitieFragment.replace(R.id.fragmentContainerView,fragmentCtrlMiscareTra
nzitie).commit();
            }
        });

        FragmentUrmaresteLinia.setOnClickListener(new
```

```
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                FragmentTransaction tranzitieFragment =
getActivity().getSupportFragmentManager().beginTransaction();

tranzitieFragment.replace(R.id.fragmentContainerView,fragmentUrmaresteLinia
).commit();
            }
        });

        FragmentUrmaresteUltrasonicBtn.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                FragmentTransaction tranzitieFragment =
getActivity().getSupportFragmentManager().beginTransaction();

tranzitieFragment.replace(R.id.fragmentContainerView,fragmentUrmaresteUltra
sonic).commit();
            }
        });

        FragmentUrmaresteLuminaBtn.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                FragmentTransaction tranzitieFragment =
getActivity().getSupportFragmentManager().beginTransaction();

tranzitieFragment.replace(R.id.fragmentContainerView,fragmentUrmaresteLumin
a).commit();
            }
        });

        FragmentMiscareAutomataBtn.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                FragmentTransaction tranzitieFragment =
getActivity().getSupportFragmentManager().beginTransaction();

tranzitieFragment.replace(R.id.fragmentContainerView,fragmentMiscareAutomat
a).commit();
            }
        });


        return view;



    }
}
```

## Fisier fragment_meniu_principal.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.appcompat.widget.LinearLayoutCompat
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical"
    tools:context=".FragmentMeniuPrincipal"
    android:backgroundTint="@color/albastruInchis2">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="85"
        android:weightSum="100"
        android:orientation="vertical">


        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="20"
            android:weightSum="100"
            android:orientation="horizontal" >

            <LinearLayout
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="5"/>

            <Button
                android:layout_weight="90"
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:id="@+id/btnFrgCtrlMiscare"
                android:text="Control miscare"
                android:textAllCaps="false"
                android:layout_margin="20dp"
                app:backgroundTint="@null"
                android:background="@drawable/resurse_btn_meniu"
                android:textColor="@color/white"
                android:textStyle="bold"
                android:drawableLeft="@drawable/ic_control_miscare_30dp"
                android:drawableRight="@drawable/ic_control_miscare_30dp"/>

            <LinearLayout
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="5"/>

        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="0dp"
```

```
            android:layout_weight="20"
            android:weightSum="100">

            <LinearLayout
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="5"/>

            <Button
                android:layout_weight="90"
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:id="@+id/btnFrgUrmLinia"
                android:text="Urmareste Linia"
                android:textAllCaps="false"
                android:layout_margin="20dp"
                app:backgroundTint="@null"
                android:background="@drawable/resurse_btn_meniu"
                android:textColor="@color/white"
                android:textStyle="bold"
                android:drawableLeft="@drawable/ic_urmareste_linia_30dp"
                android:drawableRight="@drawable/ic_urmareste_linia_30dp"
                />

            <LinearLayout
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="5"/>

        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="20"
            android:weightSum="100">

            <LinearLayout
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="5"/>

            <Button
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="90"
                android:id="@+id/btnFragmentUrmaresteLumina"
                android:text="Urmareste Lumina"
                android:textAllCaps="false"
                android:layout_margin="20dp"
                app:backgroundTint="@null"
                android:background="@drawable/resurse_btn_meniu"
                android:textColor="@color/white"
                android:textStyle="bold"
                android:drawableLeft="@drawable/ic_urmareste_lumina_30dp"

android:drawableRight="@drawable/ic_urmareste_lumina_30dp"/>

            <LinearLayout
```

```
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="5"/>

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="20"
        android:weightSum="100">

        <LinearLayout
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="5"/>

        <Button
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="90"
            android:id="@+id/btnFragmentMiscareAutomata"
            android:text="Miscare Automata"
            android:textAllCaps="false"
            android:layout_margin="20dp"
            app:backgroundTint="@null"
            android:background="@drawable/resurse_btn_meniu"
            android:textColor="@color/white"
            android:textStyle="bold"
            android:drawableLeft="@drawable/ic_miscare_auto_30dp"
            android:drawableRight="@drawable/ic_miscare_auto_30dp"/>

        <LinearLayout
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="5"/>

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="20"
        android:weightSum="100">

        <LinearLayout
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="5"/>

        <Button
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="90"
            android:id="@+id/btnFragmentUrmaresteUltrasonic"
            android:text="Urmareste Ultrasonic"
            android:textAllCaps="false"
            android:layout_margin="20dp"
            app:backgroundTint="@null"
```

```
                android:background="@drawable/resurse_btn_meniu"
                android:textColor="@color/white"
                android:textStyle="bold"
                android:drawableLeft="@drawable/ic_urmareste_ultrasonic"
                android:drawableRight="@drawable/ic_urmareste_ultrasonic"/>

            <LinearLayout
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="5"/>

        </LinearLayout>


    </LinearLayout>


</androidx.appcompat.widget.LinearLayoutCompat>
```

## Fisier FragmentUrmaresteUltrasonic.java

```
package com.example.aplicatiebluetooth;

import android.os.Bundle;

import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentTransaction;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;

import java.io.OutputStream;


public class FragmentUrmaresteUltrasonic extends Fragment {

    Button InapoiBtn;
    private OutputStream fluxDateBtIesire;


    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {

        View view =
inflater.inflate(R.layout.fragment_urmareste_ultrasonic, container, false);

        ActivitatePrincipala activitatePrincipala = (ActivitatePrincipala)
```

```
getActivity();
        if (activitatePrincipala != null) {
            fluxDateBtIesire = activitatePrincipala.preiaDateIesire();
        }

        InapoiBtn = view.findViewById(R.id.btnInapoi);

        Fragment fragmentMeniuPricipal = new FragmentMeniuPrincipal();

        InapoiBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                FragmentTransaction tranzitieFragment =
getActivity().getSupportFragmentManager().beginTransaction();

tranzitieFragment.replace(R.id.fragmentContainerView,fragmentMeniuPricipal)
.commit();

                try{
                    char date = '@';
                    fluxDateBtIesire.write((date + "\r \n").getBytes());
                    System.out.println(date);
                }catch(Exception e){}

            }
        });


        char date = '&';
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    boolean bucla = true;
                    while(bucla){
                        fluxDateBtIesire.write((date + "\r
\n").getBytes());
                        System.out.println(date);
                        Thread.sleep(50);
                        if(InapoiBtn.isPressed() ){
                            break;
                        }
                    }
                }catch(Exception e){}
            }
        }).start();

        return  view;
    }
}
```

## Fisier FragmentUrmaresteUltrasonic.java

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    tools:context=".FragmentUrmaresteUltrasonic"
    android:background="@color/albastruInchis2"
    android:orientation="vertical"
    android:weightSum="100">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="26"
        android:gravity="end"
        android:orientation="horizontal">

        <Button
            android:id="@+id/btnInapoi"
            app:backgroundTint="@null"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical"
            android:layout_marginEnd="30dp"
            android:background="@drawable/reurse_btn_inapoi"
            android:gravity="end" />

    </LinearLayout>


    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="74">

        <ImageView
            android:id="@+id/imageView3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:src="@drawable/urmareste_ultrasonic" />
    </LinearLayout>

</LinearLayout>
```

# CD / DVD