BB-8

Autor: Ion Alexandra Ana-Maria 333CC

Introducere

Proiectul constă într-un robot controlat prin bluetooh de pe telefon, care se deplaseaza asemenea droidului **BB-8 din Star Wars**. La fel ca droidul original, acesta va reda diverse sunete la anumite intervale de timp (sunetele vor fi generate aleatoriu).

Acesta are ca scop construirea cât mai fidelă a copiei lui BB-8, din punct de vedere al mecanici deplasării și sunetului. Ideea a pornit de la un videoclip în care se realizează o copie a lui R2-D2, folosind **Arduino Uno**. *Mecanicile din spatele modului de deplasare a lui BB-8* mi s-au părut mult mai interesante si mereu am fost fascinata de droizii din filmele Star Wars (o franciză iubită cu care am crescut) așa că am optat pentru realizarea sa.

Descriere generală

Schema bloc

Modul în care funcționează

BB-8 va fi controlat prin bluetooth de pe o aplicație de pe telefon, de pe care utilizatorul poate trimite indicațiile pentru direcție. Datele sunt trimise plăcuței Arduino, care la rândul său îi transmite datele modului de control al motoarelor, iar acesta din urmă va putea controleza motoarele și implicit mișcarea roților.

Robotelul se va putea **deplasa atât cu fața cât și cu spatele** și de asemenea se va putea roti astfel încât să se poată **deplasa și pe diagonală sau să se rotească pe loc pe axele Ox, Oy** (unde Ox reprezintă axa sa normală de deplasare). La anumite intervale de timp acesta va reda diverse sunete specifice droidului original.

Rotația se realizează prin schimbarea vitezei și a sensului deplasării fiecărei roți, ***ROȚILE NU AU***

CUPLAJ CARE SĂ LE PERMITĂ SĂ FIE OMNIDIREȚIONALE ȘI NU SUNT NICI ROȚI MECANUM, astfel încât spre exemplu, pentru a se putea deplasa spre dreapta, roata stângă va avea o viteză mai mare decât roata dreaptă.

Interacțiunea cu utilizatorul

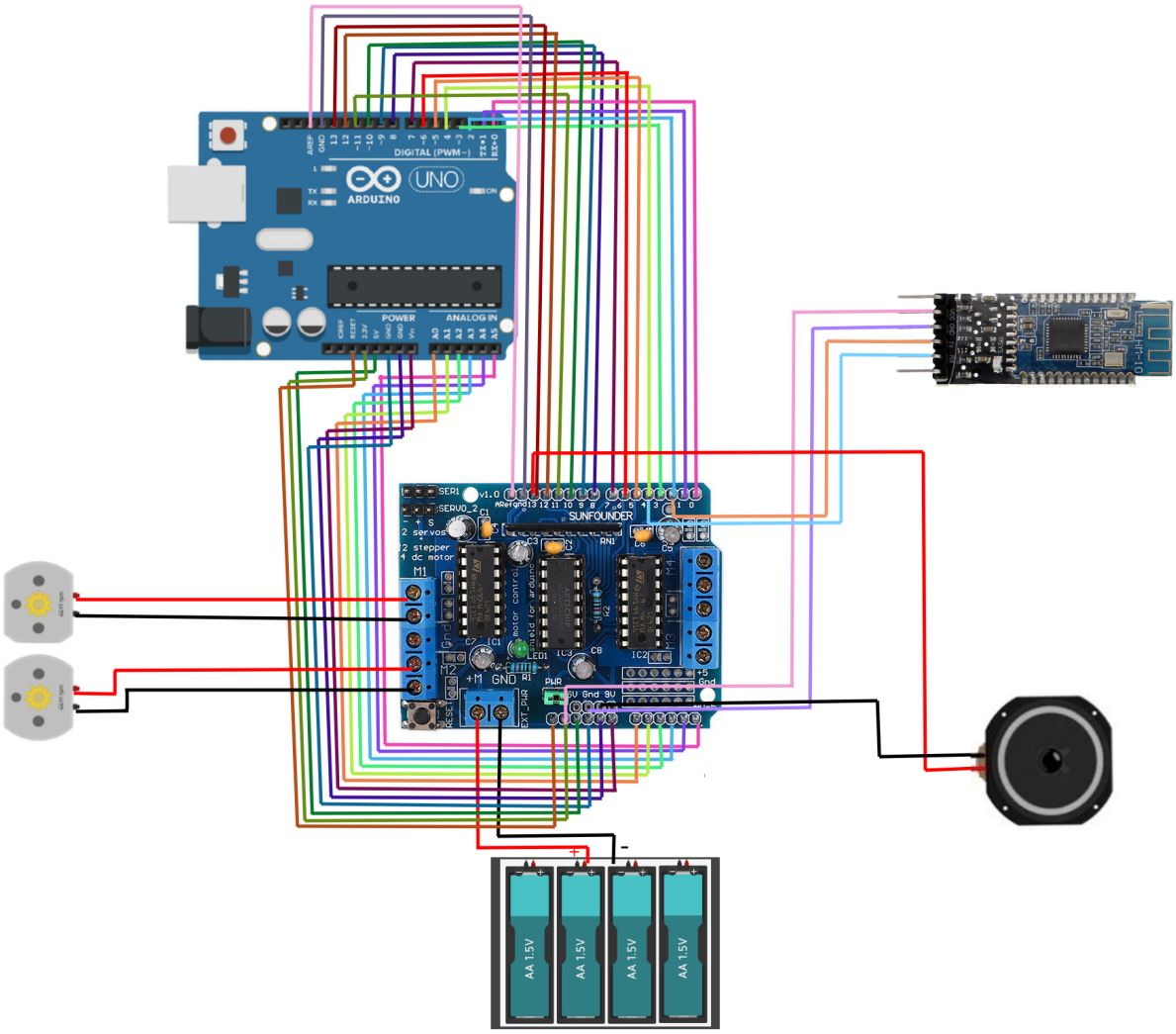
- Conectarea robotului la telefon prin bluetooth (este necesară o aplicație care poate comunica cu modulul bluetooth HM-10, deoarece acesta are conexiune de tip BLE - **telefonul nu poate comunica direct cu modulul**)
- Deplasarea robotului poate fi acum controlată prin intermediul aplicației BitBlue, pe baza joystick-ului

Hardware Design

Componente

- Arduino UNO cu ATmega328P
- Motor Driver Shield L293D
- Modul Bluetooth HM-10
- Suport baterii AA
- Boxa 40MM 3W
- 2 x Kit motor reductor + roată plastic de cauciuc
- 4 x Baterii AA 1.5V
- Cabluri mamă-tată și cabluri tată-tată

Schema Electrică



Steal Its Look

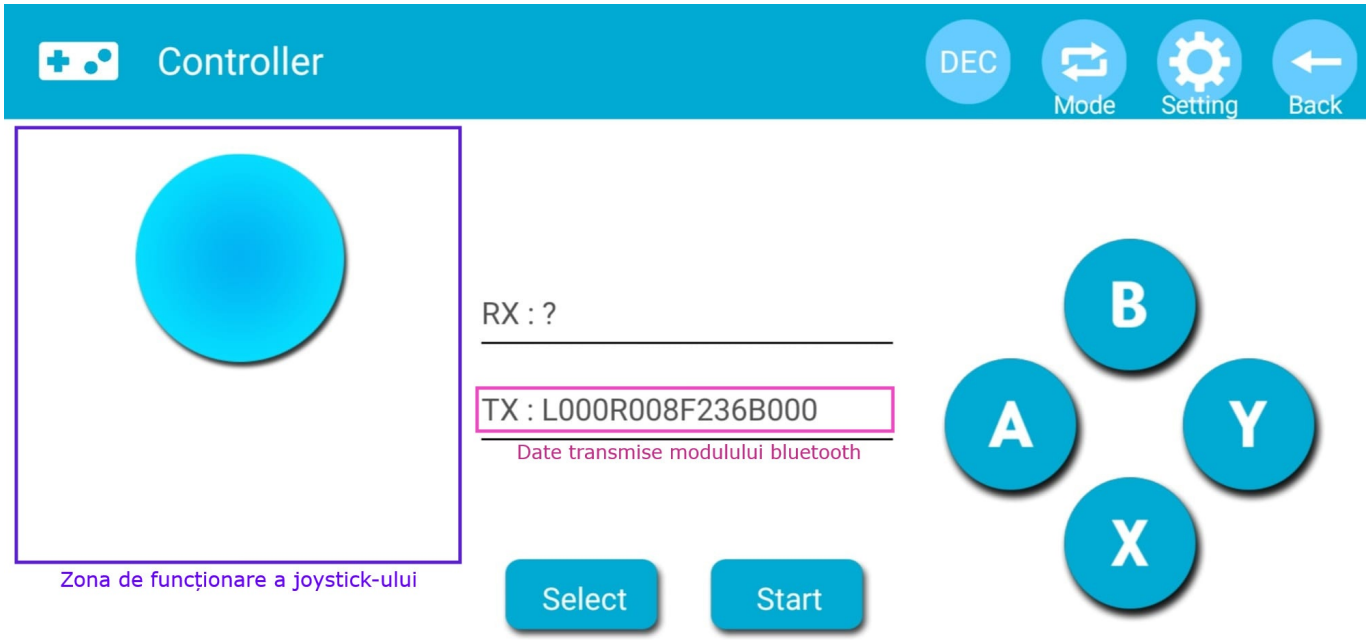
- Vopsea, alb lucios
- Markere Sharpie (Tangerine, Metallic Silver, Black)
- Bilă polistiren 8 cm
- Șasiu printat 3D
- Sferă 19.5cm printată 3D
- 6 x Magneți Neodim 6x3mm, putere 990 g

Software Design

BitBlue

Pentru a controla robotul, am folosit o aplicație care oferă suport pentru conexiunea cu modulul HM-10.

BitBlue este o aplicație disponibilă pe iOS și Android ce oferă atât suport pentru HM-10, cât și posibilitatea transmiterii datelor corespunzătoare poziției un joystick. Este singura aplicație pe care am găsit-o exact pentru aceste cerințe.



În funcție de poziția joystick-ului, aplicația transmite un *șir de 16 caractere* de forma:

- **L[000-255]R[000-255]F[000-255]B[000-255]**(transmis caracter cu caracter). Coordonatele pentru fiecare valoare (Right, Left, Forward, Backward) au limite impuse de aplicație, putând fi cuprinse între 0 și 255.

Pentru telefoanele mai noi cu Android, aplicația nu poate fi instalată de pe Play Store, dar poate fi descărcată o versisune mai veche a aplicației din surse externe.

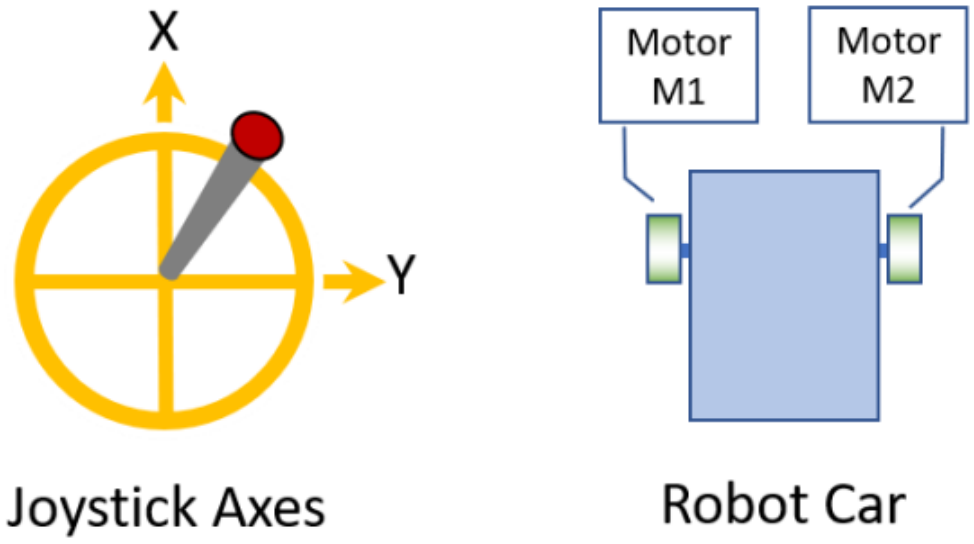
Algoritmi utilizați

Pentru a putea controla robotul astfel încât să meargă în mai multe direcții sau să se rotească în funcție de coordonatele joystick-ului, am utilizat indicațiile prezentate în cadrul acestui articol.

1.Stabilirea axelor și diverse ipoteze

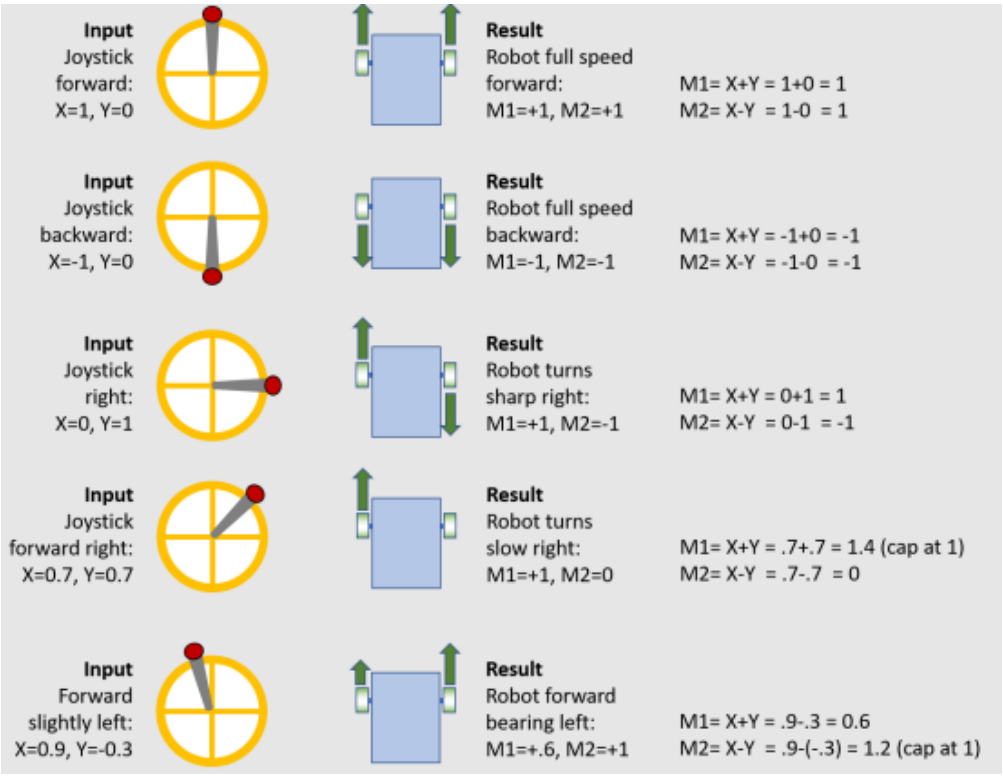
- Roțile sunt plasate conform diagramei (pentru a merge înainte, roata M2 merge înainte și roata M1 înapoi)
- Viteza este simplificată la valori între -1 și 1 , unde -1 (cea mai mare viteză, merge înapoi) și 1 (cea

mai mare viteză, merge înainte)
- Pentru a roti robotul, roțile vor avea viteze (și direcții în unele cazuri) diferite



2. Ecuații rezultate

- Rezultă ecuațiile **M1 = X+Y** și **M2 = X-Y**



Codul aplicației

Mediul de dezvoltare utilziat este **Arduino IDE**. Din punct de vedere al librăriilor utilizate, am folosit **AFMotor.h** pentru controlul motoarelor, **SoftwareSerial.h** pentru utilizarea modulului bluetooth și **NewTone.h**, ca alternativă mai bună la folosirea funcției *tone()* pentru redarea sunetului (funcția

tone() și interfața definită cu SoftwareSerial au întreruperi comune, așa că folosirea funcției *tone()* conduce la funcționarea greșită a modulului Bluetooth).

Configurare timer0

Timer1 era deja folosit de librăriile utilizate, așa că am utilizat Timer0 pentru redarea sunetului la anumite intervale de timp. Frecventa este setata la 1hz, ceea ce înseamnă că la fiecare 0.01s va intra în funcția *ISR(TIMER0_COMPA_vect)*. Astfel după un anumit interval (cuprins între 10s și 15s, timer-ul va semnala că trebuie redat sunetul).

timer0.cpp

```
ISR(TIMER0_COMPA_vect)
{
    countSelf++;

    // La fiecare threshold/100 secunde este pornit sunetul
    if (countSelf == threshold)
    {
        // Generare repetari audio
        audioRepeat = random(100, 200);

        // Reinitializarea count si threshold
        countSelf = 0;
        threshold = random(1000, 2000);
    }
}

void configure_timer0()
{
    // Configurare pentru Timer 0 în mod CTC
    // Frecventa de 100 Hz -> 0.01 secunde
    TCCR0A = 0;
    TCCR0B = 0;
    TCNT0 = 0;
    OCR0A = 155;
    TCCR0A |= (1 << WGM01); // CTC mode
    TCCR0B |= (1 << CS00);
    TCCR0B |= (1 << CS02);
}

void init_timer0()
{
    TIMSK0 |= (1 << OCIE0A);
}
```

Redare sunet

Sunetul redat va fi generat mereu în mod aleatoriu conform comentariilor din cod. Am urmărit modelul de generare a sunetelor lui R2D2, cu mici diferențe datorită utilizării *NewTone.h* în locul

funcției *tone()* și a delay-urilor.

audioPlay.cpp

```
void audioPlay()
{
    // 4 moduri posibile pentru generarea sunetului
    // k - 2*i, k + 2*i , k - 10*i, k + 10*i
    int sign = random(1, 2);
    int multiply = random(1, 2);
    if (sign == 2)
        sign = -1;
    if (multiply == 1)
        multiply = 10;

    // Redare sunet
    int k = random(1000, 1800);
    for (int i = 0; i < random(10, 20); i++)
        NewTone(PIN_SPEAKER, k + (sign * multiply * i));
}
```

Deplasare

Deplasarea a fost realizată conform algoritmului menționat. De asemenea, am scalat viteza cu 0.85 pentru a face robotul mai stabil (am limitat viteza).

move.cpp

```
void Move(double x, double y)
{
    double m1 = x + y;
    double m2 = x - y;

    // Daca robotul merge cu spatele, inverseaza m1 si m2
    if (x < 0)
    {
        double aux = m1;
        m1 = m2;
        m2 = aux;
    }

    // Incadrare m1 si m2 intre -1 si 1
    if (m1 > 1)
        m1 = 1;
    if (m1 < -1)
        m1 = -1;

    if (m2 > 1)
        m2 = 1;
    if (m2 < -1)
```

```
        m2 = -1;

    // Setare viteza
    motor1.setSpeed(abs((int)(m1 * 255 * 0.85f)));
    motor2.setSpeed(abs((int)(m2 * 255 * 0.85f)));

    // Setare directie pentru fiecare roata
    if (m1 >= 0)
        motor1.run(FORWARD);
    else
        motor1.run(BACKWARD);

    if (m2 >= 0)
        motor2.run(BACKWARD);
    else
        motor2.run(FORWARD);
}
void Stop()
{
    // Oprire motoare
    motor1.setSpeed(0);
    motor1.run(RELEASE);
    motor2.setSpeed(0);
    motor2.run(RELEASE);
}
```

Prelucrarea datelor primite

În cadrul funcției *loop()* am realizat citirea datelor de la modulul bluetooth și le-am prelucrat. Un șir complet de coordonate conține 16 caractere. Astfel la fiecare 16 caractere trimise, se va procesa buffer-ul curent și se vor calcula coordonatele. Dacă valoarea coordonatelor este 0, robotului îi va fi semnalat să se oprească, altfel datele vor fi transmise funcției *Move(double x, double y)*. De asemeneam se realizează și redarea sunetelor (dacă **audioRepeat > 0**, atunci se va reda o parte scurtă de audio în funcția *audioPlay()* care nu influențează în vreun fel primirea datelor).

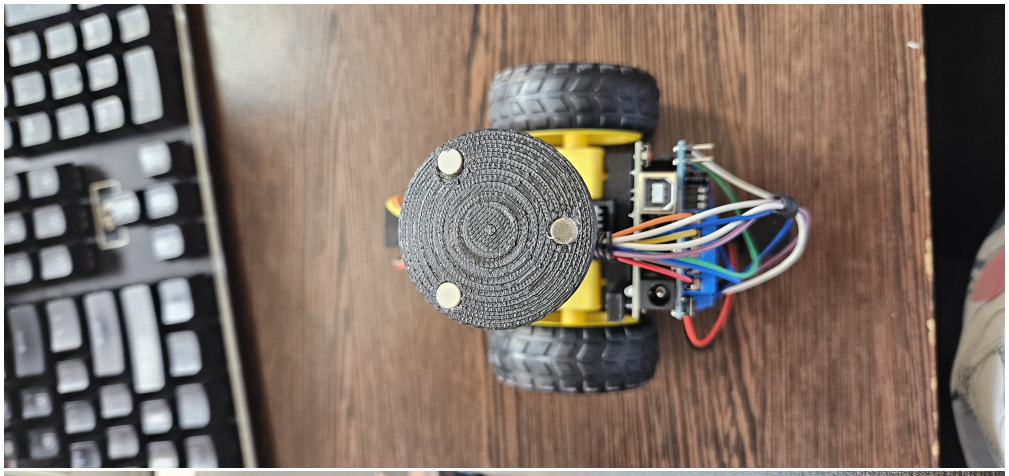
data.cpp

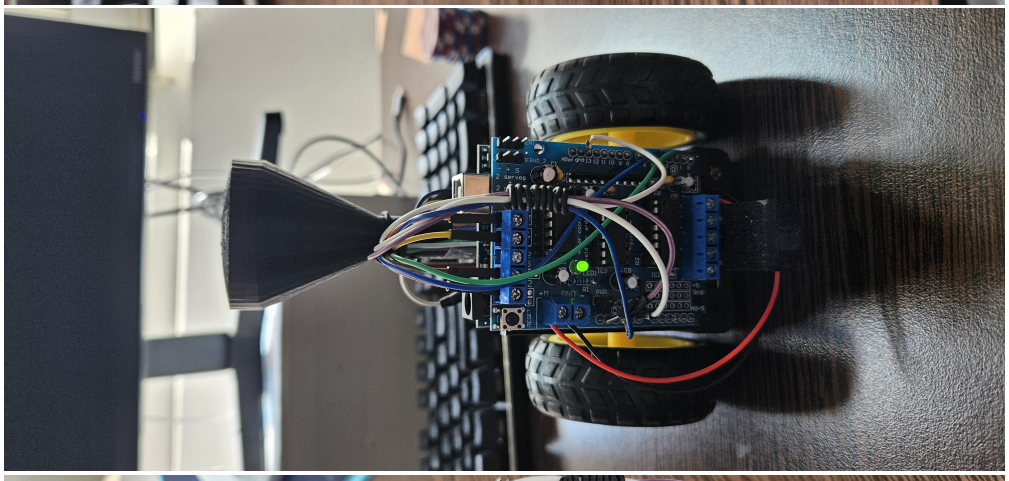
```
void loop()
{

    HM10.listen();

    // Preluare input de la HM10
    if (HM10.available() > 0)
    {

        // Citire caracter si concatenare
        char c = HM10.read();
        command += c;
```

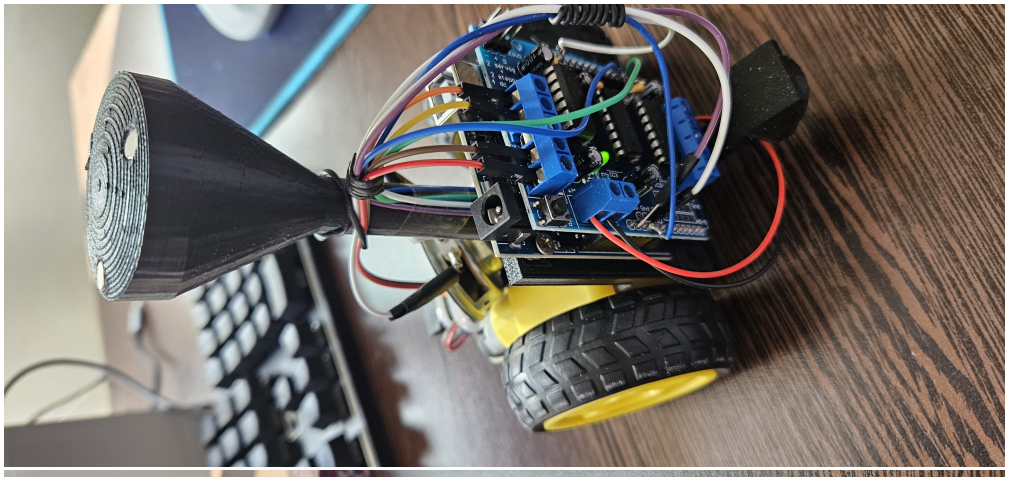
```
        // Atunci cand este receptionata o comanda (16 caracatere),
preia datele
        if (command.length() == 16)
        {

            leftCoord = command.substring(1, 4).toInt();
            rightCoord = command.substring(5, 8).toInt();
            forwardCoord = command.substring(9, 12).toInt();
            backCoord = command.substring(13, 16).toInt();

            /* In cazul in care toate coordonatele sunt 0, opreste
motoarele,
             * altfel seteaza vitezele si directiile corepunzatoare
             */
            if (!leftCoord && !rightCoord && !forwardCoord && !
backCoord)
                Stop();
            else
            {
                double y = (rightCoord - leftCoord) / 255.0;
                double x = (forwardCoord - backCoord) / 255.0;
                Move(x, y);
            }

            // Reseteaza buffer-ul
            command = "";
        }
    }

    // Redare audio in continuare daca audioRepeat > 0
    if (audioRepeat)
    {
        audioPlay();
        audioRepeat -= 1;
    }
    else
        noNewTone(SPEAKER_PIN);
}
```

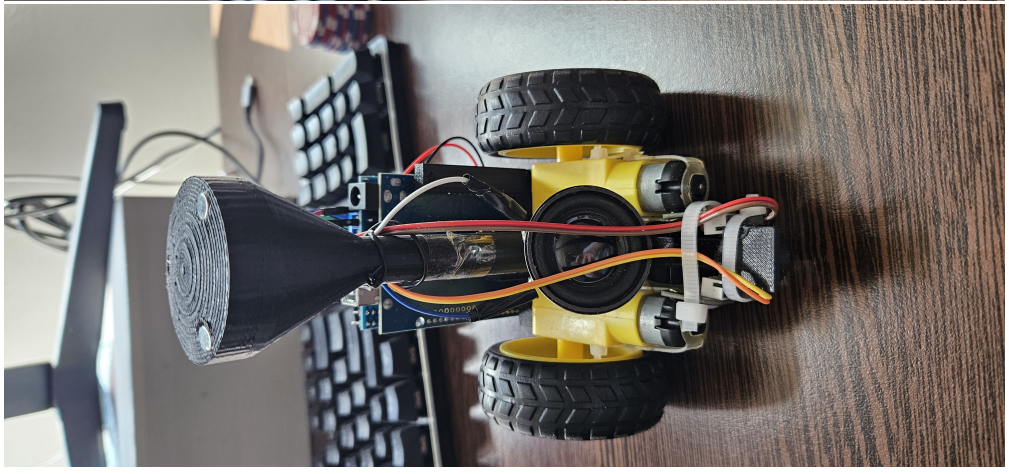
Rezultate Obţinute

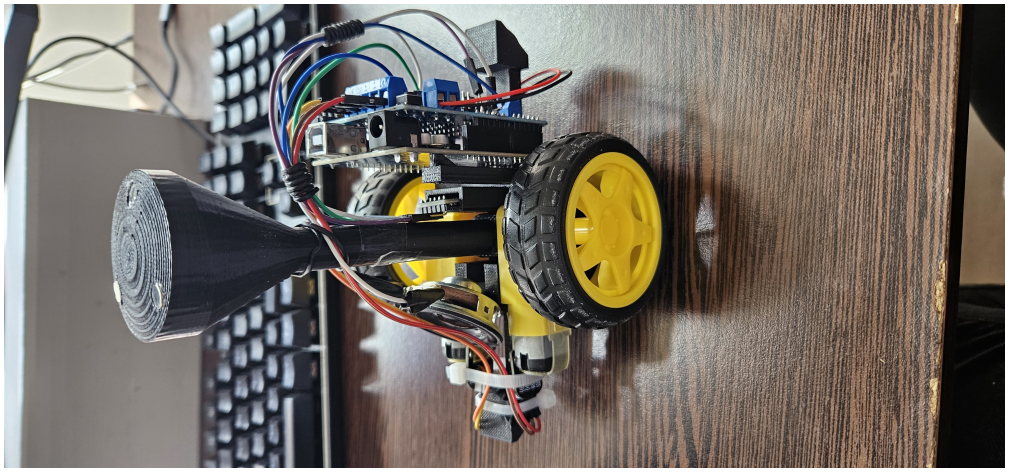
Hardware











Exteriorul

Video functionare

Jurnal

5-6 Mai - Crearea paginii de wiki, completarea descrierii și completarea listei de piese și a schemei bloc

16-17 Mai - Conectarea motoarelor la motor driver shield și testarea lor (am întâlnit probleme datorate lipturilor făcute)

18-19 Mai - Schimbarea motoarelor cu unele care nu necesită lipituri și realizare controlului prin modulul de Bluetooth

20-21 Mai - Realizarea codului funcțional pentru controlul robotului (controlat de pe telefon, prin

bluetooth, coordonatele sunt preluate de la un joystick). Schimbarea mod de redare sunet, nu voi mai folosi cititorul de card SD, deoarece comunica cu placuța Arduino prin SPI, iar pinii digitali 11 si 12 sunt deja folosiți de shield-ul care conține L293D. Voi folosi doar un difuzor, iar sunetele vor fi generate aleatoriu pe baza unui algoritm.

22-25 Mai - Reglaje la modul de deplasare din punct de vedere fizic (rezolvare probleme legate de echilibrul robotului)

26-27 Mai - Finalizarea părții software (deplasarea robotului concomitentă cu redarea sunetelor + limit testing)

28 Mai - Arts & Crafts

Download

Surse software și fișiere componente 3D

Bibliografie/Resurse

Resurse Hardware

- Datasheet ATmega328P
- L293D Motor Driver Shield
- Datasheet Modul bluetooth HM-10
- Introduction to L293D Motor Driver Shield

Resurse Software

- BitBlue App
- R2D2 - Sound-Generator
- Playing Audio on Arduino
- NewTone.h, alternativa lui tone()
- Star Wars BB8 Remote Control Printed Robot
- Repository - Model utilizare biblioteca AFMotor
- How to Control a Robot’s Wheel Motors Based on Joystick Movements

Export to PDF