Mașinuță Bluetooth - VRÎNCEANU Dan

Introducere

Prezentarea pe scurt a proiectului vostru:

- ce face

Proiectul constă într-o mașinuță inteligentă cu trei moduri de operare: control manual prin Bluetooth folosind o aplicație pe smartphone, mod automat în care evită obstacolele

- care este scopul lui

Scopul acestui proiect pentru mine este să învăț cum pot integra diferite tehnologii, precum senzorii, comunicația Bluetooth și algoritmii de control, într-un sistem robotic funcțional. Mi se pare un proiect interesant și provocator, care mă ajută să înțeleg mai bine cum lucrează componentele între ele. În plus, îmi oferă ocazia să aplic ceea ce am învățat la școală într-un mod practic și interactiv. Consider că este o experiență utilă atât pentru dezvoltarea mea personală, cât și pentru viitoarele proiecte în domeniul tehnologiei și automatizării.

- care a fost ideea de la care aţi pornit

Ideea de la care am pornit a fost să fac o mașinuță controlată prin Bluetooth, o dorință pe care o aveam încă din copilărie, pentru că mi s-a părut mereu ceva foarte interesant. Pe parcurs, m-am gândit cum aș putea să o îmbunătățesc și să o fac mai inteligentă, așa că am adăugat senzori pentru a evita obstacolele si prapastii. Astfel, proiectul a devenit mult mai complex și mai captivant, combinând pasiunea din copilărie cu noile cunoștințe tehnice.

- de ce credeţi că este util pentru alţii şi pentru voi

Proiectul este util prin ceea ce face, pentru că reproduce, la scară mică, funcționalități întâlnite în tehnologia reală – cum ar fi mașinile autonome sau sistemele inteligente de transport. Modul automat ajută la înțelegerea principiilor de evitare a obstacolelor, folosite la roboți sau mașini autonome.
</note>

Descriere generală

- Circuit

Lista de piese:

Componentă	Cantitate
Placă de plastic	1
ESP32	1
Kit motor reductor + roată plastic cu cauciuc	4
Punte H Dublă MX1508, DC	1
Senzor ultrasunete HC SR-04P	1
Breadboard 400 puncte	2
Senzor IR	1
LCD 1602 cu interfață I2C	1
Baterie 18650	2
Modul powerbank 18650	1
Servomotor SG90	1
Suport acumulatori 18650, 2S	1

</note>

Software Design

Descrierea flow-ului software:

Mașina funcționează în două moduri distincte, controlate din aplicația de pe smartphone (Dabble):

Modul Manual (default la conectare) Modul Automat: Evitare obstacole + prăpastii

Starea 0 → Așteptare conectare Bluetooth (Idle):

- La pornirea microcontrollerului, mașina așteaptă conectarea prin Bluetooth.
- Pe LCD se afișează mesajul: Conecteaza-te sefule
- Nicio mișcare nu este permisă până la stabilirea conexiunii.

Starea 1 → Mod Manual (Bluetooth GamePad activ):

- După conectare, mașina intră în Mod Manual, unde poate fi controlată de utilizator folosind aplicația Dabble.
- Se afișează pe LCD: Mod: Manual
- Direcțiile disponibile:
- ↑ - Mers înainte
- ↓ - Mers înapoi
- ← - Viraj la stânga
- → - Viraj la dreapta
- În lipsa unei comenzi, motoarele sunt oprite automat.

Starea 2 → Mod Automat (Evitare obstacole și prăpastii):

- Activat prin apăsarea butonului Triangle din aplicație.

- Comportament:

- Se afișează pe LCD: Mod: Obstacole+, Prapastii
- Se utilizează senzorul ultrasonic pentru detecția obstacolelor și senzorul IR pentru prăpastii.
- Dacă se detectează o prăpastie (senzor IR = LOW):
- Mașina oprește, dă înapoi și virează automat pentru a evita zona periculoasă.
- Mesaj pe LCD: Prapastie STOP
- Dacă se detectează un obstacol în față:
- Se măsoară distanțele din stânga și dreapta prin rotirea servomotorului.
- Se alege direcția cea mai liberă și se virează în acea parte.
- Mesaj pe LCD: Obstacol STOP
- Dacă drumul e liber: mașina merge înainte.
- Mesaj pe LCD: Drum liber

Starea 3 → Revenire la Manual:

- Apăsarea butonului Cross din aplicație dezactivează modul automat.
- Mașina oprește orice mișcare și revine în modul manual.
- LCD-ul afișează din nou: Mod: Manual

Motivația Alegerii Bibliotecilor

Arduino Libraries

```
==== Motivația Alegerii Bibliotecilor ====
```

ESP32 Robotics Libraries

```
#include <DabbleESP32.h>       // Control Bluetooth cu aplicația mobilă
Dabble (GamePad virtual)
#include <ESP32Servo.h>        // Compatibilitate PWM pentru control precis
al servomotoarelor pe ESP32
#include <NewPing.h>           // Măsurători rapide și stabile cu senzorul
ultrasonic HC-SR04
#include <Wire.h>              // Comunicație I²C între ESP32 și periferice
#include <LiquidCrystal_I2C.h> // Afișare text pe LCD 16x2 cu interfață I²C
```

Justificare: Am ales aceste biblioteci pentru a realiza un sistem robotic autonom cu control manual prin Bluetooth și feedback vizual.

- `DabbleESP32` permite interacțiunea prin aplicația mobilă fără componente fizice externe. - `ESP32Servo` este necesară pentru compatibilitate cu PWM-ul specific ESP32, esențial în orientarea senzorului ultrasonic. - `NewPing` asigură măsurători eficiente ale distanței fără blocaje, utile pentru evitarea obstacolelor. - `Wire` este standard pentru comunicație I²C, permițând integrarea simplă a mai multor senzori sau afișaje. - `LiquidCrystal_I2C` oferă un mod convenabil de a afișa mesaje de

stare, diagnostic sau feedback pentru utilizator.

Laboratoarele

Laboratorul 0: GPIO

Folosit pentru controlul pinilor digitali – pornirea/opirea motoarelor, citirea stării senzorului IR.

Laboratorul 1: UART

Utilizat pentru comunicarea serială cu computerul prin Serial.begin() – afişează mesaje de stare și debug.

Laboratorul 3: Timere. PWM

PWM este folosit pentru a controla poziția servomotorului (ex: rotirea senzorului ultrasonic pentru detectarea obstacolelor).

Laboratorul 6: I2C

Folosit pentru comunicarea cu ecranul LCD 1602 prin interfața I2C – afişează mesaje despre obstacole, prăpastii sau conexiunea cu aplicația.

Element de Noutate al Proiectului

Integrarea controlului manual prin Bluetooth (Dabble) cu un sistem autonom de evitare a obstacolelor și prăpastiilor. Robotul poate schimba direcția și viteza în timp real, oferind feedback pe LCD și folosind un servomotor pentru scanare laterală. Această combinație de autonomie și interactivitate mobilă face proiectul versatil și inovator.

Calibrarea senzorilor

Senzor ultrasonic (HC-SR04) Am setat o distanță maximă de detecție (200 cm) și un prag pentru obstacol (20 cm). Am testat valorile în monitorul serial pentru a verifica funcționarea corectă. Senzor IR Am calibrat detectarea marginii ca semnal LOW. Am verificat comportamentul apropiind senzorul

de o margine (ex. masă). Servomotor + senzor ultrasonic Am setat unghiuri fixe (50° și 130°) pentru scanare laterală. Am verificat că senzorul măsoară distanța corect în ambele direcții.

Main application flow

Setup

- Inițializează comunicația serială la 115200 baud.
- Configurează pinurile pentru:
  1. Motoare (IN1, IN2, IN3, IN4)
  2. Senzor IR (prăpastie)
  3. Ultrasunete (TRIG, ECHO)
  4. Servomotor (SERVO_PIN)
- Atașează servomotorul și îl poziționează inițial la 90°.
- Inițializează conexiunea Bluetooth cu aplicația Dabble.
- Configurează magistrala I2C (Wire.begin) pentru LCD.
- Inițializează LCD-ul (16×2) și aprinde lumina de fundal.
- Afișează mesaj pe ecran: “Conecteaza-te, sefule”.

```
void setup()
{
  Serial.begin(115200);
  setUpPinModes();
  Dabble.begin("MyBluetoothCar");

  Wire.begin(25, 26); // SDA, SCL
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Conecteaza-te");
  lcd.setCursor(0, 1);
  lcd.print("sefule ");

  Serial.println("Dabble și LCD inițiate, aștept conectarea...");
}
```

—

Main Loop

- Procesează comenzile de la Dabble.
- Dacă aplicația nu e conectată:
  1. Afișează mesaj și iese temporar din buclă.
- La prima conectare:
  1. Afișează “Mod: Manual” pe LCD.
- Dacă se apasă Triunghi (▲):
  1. Activează modul automat (evitarea obstacolelor + margini).
- Dacă se apasă X (✖):
  1. Dezactivează modul automat și revine la control manual.

- Dacă modul automat este activ:
  1. Apelează `avoidObstacles()` și iese din `loop`.
- Altfel:
  1. Controlează robotul în mod **manual** cu D-pad:
     1. ▲ = înainte
     2. ▼ = înapoi
     3. ◄ = stânga
     4. ► = dreapta
     5. Nimic = oprește

```
void loop()
{
  Dabble.processInput();

  if (!Dabble.isAppConnected())
  {
    lcd.setCursor(0, 0);
    lcd.print("Conecteaza-te ");
    lcd.setCursor(0, 1);
    lcd.print("sefule        ");
    delay(1000);
    return;
  }

  static bool wasConnected = false;
  if (!wasConnected)
  {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Mod: Manual");
    wasConnected = true;
  }

  if (GamePad.isTrianglePressed())
  {
    Serial.println("Mod obstacole+prapastie activat");
    obstacleAvoidanceMode = true;

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Mod: Obstacole+");
    lcd.setCursor(0, 1);
    lcd.print("Prapastii       ");
  }

  if (GamePad.isCrossPressed())
  {
    Serial.println("Mod automat oprit, revenire la manual");
    obstacleAvoidanceMode = false;
    stopMotors();
```

```
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Mod: Manual");
  }

  if (obstacleAvoidanceMode)
  {
    avoidObstacles();
    return;
  }

  if (GamePad.isUpPressed()) moveForward();
  else if (GamePad.isDownPressed()) moveBackward();
  else if (GamePad.isLeftPressed()) turnLeft();
  else if (GamePad.isRightPressed()) turnRight();
  else stopMotors();
}
```

—

Auxiliary Functions

```
// Configurează toate pinurile relevante și inițializează servomotorul.
void setUpPinModes()
{
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(IR_SENSOR_PIN, INPUT);

  myServo.attach(SERVO_PIN);
  myServo.write(90);
  Serial.println("Servomotor inițiat la 90°");
}

// Mișcări de bază pentru robot:
void moveForward()
{
  GPIO.out_w1ts = (1 << IN1) | (1 << IN3); // Set IN1 și IN3 HIGH
  GPIO.out_w1tc = (1 << IN2) | (1 << IN4); // Set IN2 și IN4 LOW
  Serial.println("Înainte");
}

void moveBackward()
{
  GPIO.out_w1ts = (1 << IN2) | (1 << IN4); // Set IN2 și IN4 HIGH
  GPIO.out_w1tc = (1 << IN1) | (1 << IN3); // Set IN1 și IN3 LOW
  Serial.println("Înapoi");
}
```

```
void turnLeft()
{
  GPIO.out_w1ts = (1 << IN2) | (1 << IN3); // IN2 și IN3 HIGH
  GPIO.out_w1tc = (1 << IN1) | (1 << IN4); // IN1 și IN4 LOW
  Serial.println("Stânga");
}

void turnRight()
{
  GPIO.out_w1ts = (1 << IN1) | (1 << IN4); // IN1 și IN4 HIGH
  GPIO.out_w1tc = (1 << IN2) | (1 << IN3); // IN2 și IN3 LOW
  Serial.println("Dreapta");
}

void stopMotors()
{
  GPIO.out_w1tc = (1 << IN1) | (1 << IN2) | (1 << IN3) | (1 << IN4); // Toți
LOW
  Serial.println("Oprire");
}
```

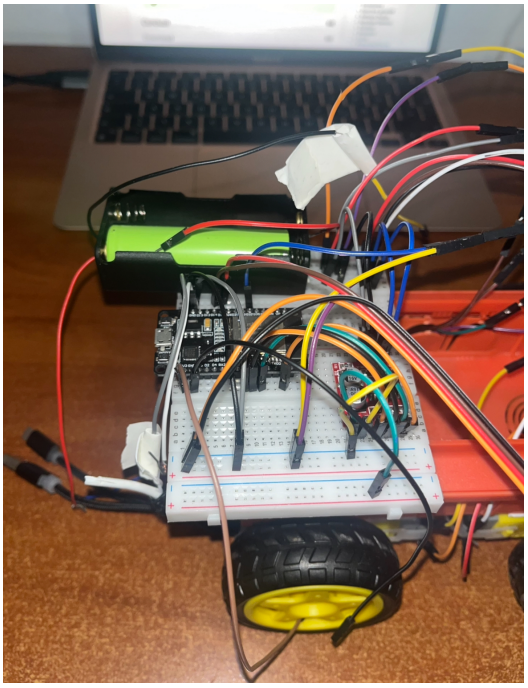
—

Sensor and Navigation Functions

```
// Returnează distanța măsurată cu ultrasunete.
int readPing()
{
  delay(70);
  int cm = sonar.ping_cm();
  if (cm == 0) cm = MAX_DISTANCE;

  Serial.print("Distanță: ");
  Serial.print(cm);
  Serial.println(" cm");
  return cm;
}

// Rotește servomotorul spre dreapta, măsoară, revine.
int lookRight()
{
  myServo.write(50);
  delay(500);
  int distance = readPing();
  myServo.write(90);
  return distance;
}

// Rotește servomotorul spre stânga, măsoară, revine.
int lookLeft()
{
```

```
  myServo.write(130);
  delay(500);
  int distance = readPing();
  myServo.write(90);
  return distance;
}

// Evită obstacole și margini. Apelată în mod automat.
void avoidObstacles()
{
  int senzorIR = digitalRead(IR_SENSOR_PIN); // LOW = prăpastie
  int distanceR = 0;
  int distanceL = 0;
  distance = readPing();

  if (senzorIR == LOW)
  {
    stopMotors();
    lcd.setCursor(0, 1);
    lcd.print("Prapastie STOP  ");
    delay(200);

    moveBackward();
    delay(400);
    stopMotors();
    delay(200);

    turnLeft(); // întoarcere completă
    delay(700);
    stopMotors();
    delay(200);
    return;
  }

  if (distance <= OBSTACLE_DISTANCE)
  {
    stopMotors();
    lcd.setCursor(0, 1);
    lcd.print("Obstacol STOP   ");
    delay(100);
    moveBackward();
    delay(300);
    stopMotors();
    delay(200);

    distanceR = lookRight();
    delay(200);
    distanceL = lookLeft();
    delay(200);

    if (distanceR >= distanceL)
```

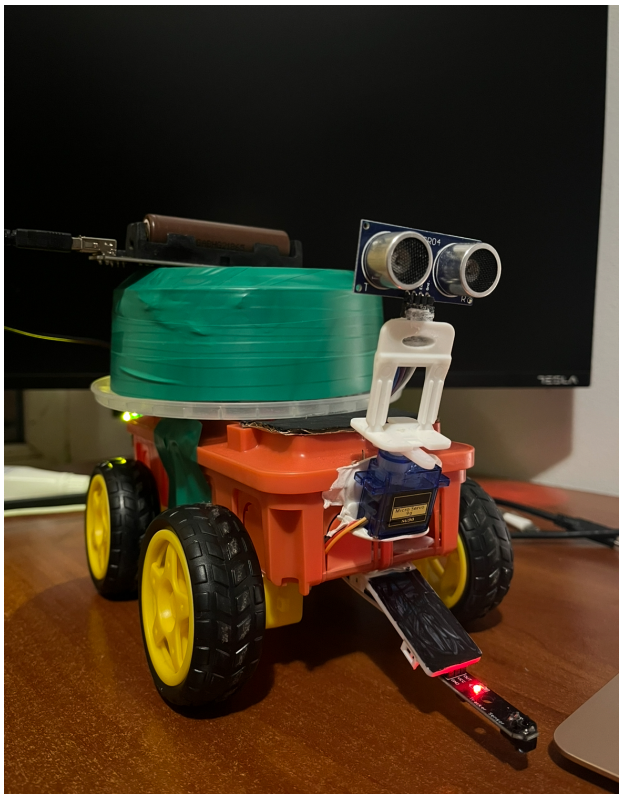
```
    {
      turnRight();
      delay(500);
      stopMotors();
    }
    else
    {
      turnLeft();
      delay(500);
      stopMotors();
    }
  }
  else
  {
    moveForward();
    lcd.setCursor(0, 1);
    lcd.print("Drum liber      ");
  }
}
```

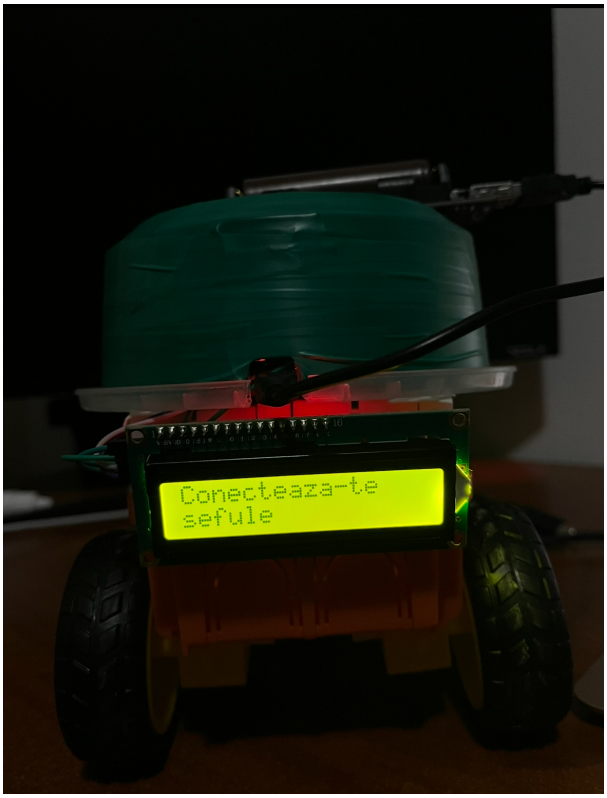


}

VIDEO

https://www.youtube.com/shorts/16-T-_LEwkY

Rezultate Obţinute





Concluzii

Acest proiect este un robot autonom controlabil prin Bluetooth, capabil să evite obstacole și prăpastii cu ajutorul senzorilor, oferind în același timp control manual precis prin aplicația Dabble. Este o combinație reușită între inteligență integrată, interfață prietenoasă și control adaptiv al motoarelor.

Download

bluetoothcar_v1.1_vrinceanu.zip

Bibliografie/Resurse

LCD Help: https://www.youtube.com/watch?v=860eErq9c3E>

Piese: https://sigmanortec.ro https://www.optimusdigital.ro/ro/

From:
http://ocw.cs.pub.ro/courses/ - **CS Open CourseWare**

Permanent link:
http://ocw.cs.pub.ro/courses/pm/prj2025/fstancu/dan.vrinceanu

Last update: **2025/05/28 07:28**