Swiffer Autonom

Introducere

Proiectul consta in crearea unui robot “autonom” care urmareste un algoritm de tip zig-zag. Acesta poate fi oprit, pornit sau resetat prin intermediul modulului Bluetooth. Pe viitor se poate adapta la a se conecta la un server ML pentru a invata camera autonom, iar Swiffer-ul poate fi schimbat cu un modul de aspirator.

Descriere generală

Descrierea sumara a modulelor si a modului de interactiune

1. Descrierea modulelor hardware:

- **Unitatea de procesare (ATmega328P):** Reprezinta “creierul” sistemului, fiind responsabil cu rularea algoritmului principal si coordonarea tuturor perifericelor.
- **Modulul de alimentare si putere (L298N & Divizor):** Gestioneaza energia. Regulatorul integrat asigura 5V stabili pentru logica sistemului, puntea H controleaza curentii mari pentru locomotie, iar divizorul de tensiune permite citirea nivelului bateriei in conditii de siguranta.
- **Modulul de perceptie (Senzori HC-SR04 si TCRT5000):** Culeg date din mediul fizic, masurand distanta pana la pereti/obstacole si detectand prezenta suprafetei de rulare.
- **Sistemul de locomotie (Motoare DC):** Executa miscarea fizica (directie si viteza) a sasiului.
- **Modulul de comunicatie (Bluetooth HC-05):** Asigura o legatura seriala wireless bidirectionala cu un terminal extern pentru monitorizare si debugging.

2. Modul de interactiune (Fluxul de functionare):

Sistemul interactioneaza printr-o bucla continua de tip **Achizitie date → Procesare → Actiune**. Pachetul de baterii alimenteaza intregul ansamblu. In timpul rularii, senzorii colecteaza date din mediu si le transmit catre microcontroler sub forma de impulsuri sau niveluri de tensiune. ATmega328P analizeaza aceste intrari si, pe baza algoritmului de decizie, trimite semnale de control (PWM si directie logica) catre driverul L298N. Driverul actioneaza ca un amplificator, cupland motoarele la curentul bateriei pentru a executa deplasarea. In paralel, microcontrolerul raporteaza constant starea sistemului (baterie, distante, decizii de viraj) prin intermediul modulului Bluetooth.

Hardware Design

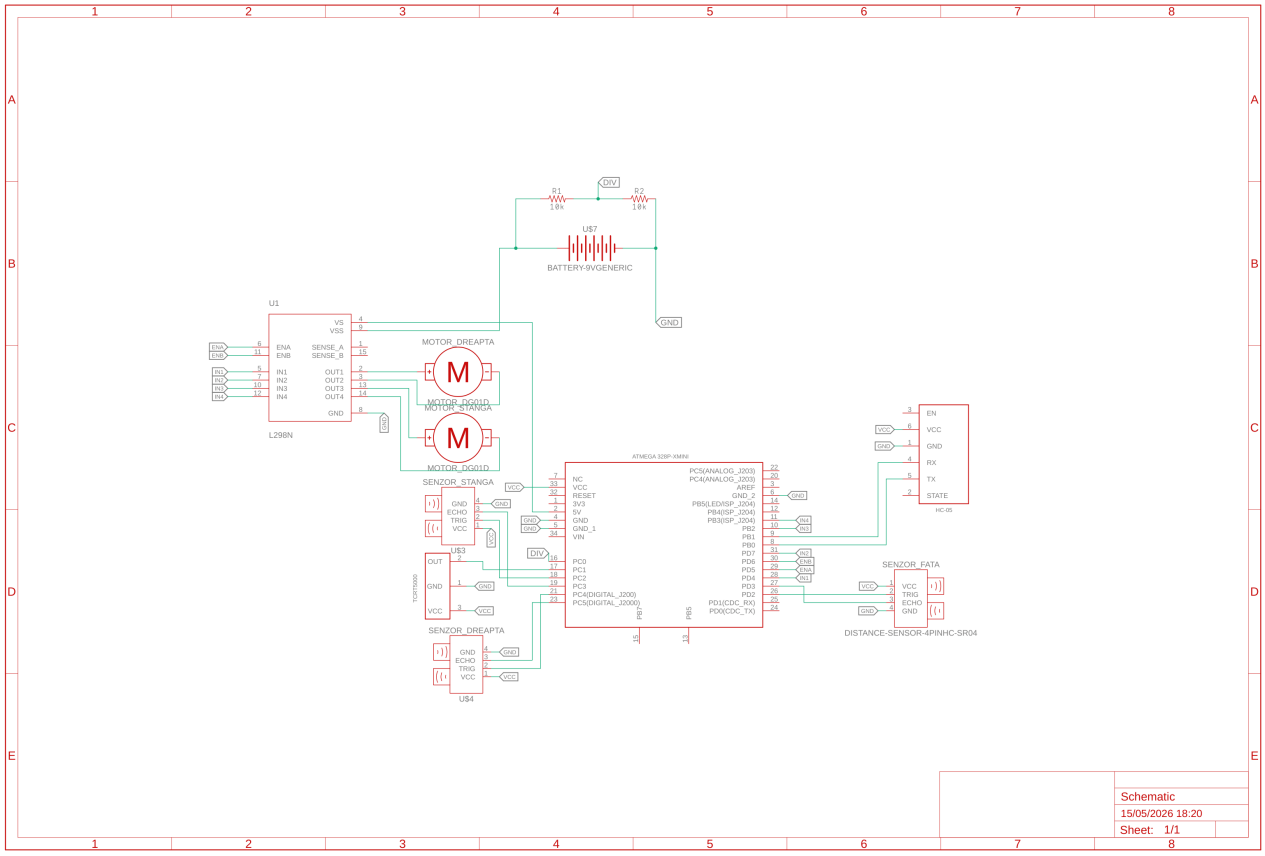
1. Listă de piese (BOM - Bill of Materials)

Pentru realizarea acestui proiect, a fost utilizată următoarea configurație hardware, axată pe un echilibru între eficiența energetică și acuratețea senzorilor:

- **Unitate de Control:** 1x Placă de dezvoltare ATmega328P Xplained Mini.
- **Sistem de Locomoție:**
  - 1x Șasiu robot 2WD (din plexiglas).
  - 2x Motoreductoare de curent continuu (Motoare TT, raport 1:48).
  - 2x Roți cauciucate + 1x Roată pivotantă (Caster wheel).
  - 1x Modul Driver Motoare L298N (cu regulator de tensiune LM7805 integrat).
- **Sistem de Percepție (Senzori):**
  - 3x Module ultrasonice HC-SR04 (amplasate frontal, stânga, dreapta).
  - 1x Senzor optic reflexiv infraroșu TCRT5000 (pentru detecția marginilor / “Swiffer”).
- **Sistem de Comunicație:** 1x Modul Bluetooth HC-05.
- **Sistem de Alimentare:**
  - 1x Suport pentru 6 baterii tip AA.
  - 6x Baterii Alcaline AA 1.5V (Tensiune totală nominală: 9V).
- **Componente Pasive & Conectică:**
  - 2x Rezistor 10kΩ
  - 1x Breadboard mini (170 puncte).
  - Set fire conexiune Dupont (Tată-Tată, Mamă-Tată).

2. Scheme Electrice și Maparea Pinilor

Pentru a vizualiza interconectarea componentelor, se va consulta schema electrică de ansamblu de mai jos. *(Notă tehnică: Alimentarea logicii de 5V a microcontrolerului se face prin pinul de +5V al modulului L298N, acesta coborând tensiunea de 6V a bateriei printr-un regulator integrat).*



15/05/2026 18:21 C:\Users\Rares\Downloads\Schematic.pdf (Sheet:)

Tabelul Conexiunilor Hardware (Pinout Map): Pentru interfațarea senzorilor cu perifericele interne ale ATmega328P s-a utilizat următoarea mapare:

Modul Extern	Pin Modul	Conexiune ATmega328P	Rol / Periferic Utilizat
Modul HC-05	TX	PB0	Comunicație Serială (USART)
Modul HC-05	RX	PB1	Comunicație Serială (USART)
L298N (Driver)	ENA	PD5	Control Viteză Motor Stânga (PWM Hardware)
L298N (Driver)	ENB	PD6	Control Viteză Motor Dreapta (PWM Hardware)
L298N (Driver)	IN1, IN2	PD4, PD7	Control Direcție Motor Stânga (GPIO)
L298N (Driver)	IN3, IN4	PB2, PB3	Control Direcție Motor Dreapta (GPIO)
Divizor Tensiune	Vout (Intersecție)	PC0 (A0)	Monitorizare nivel baterie (ADC)
TCRT5000	A0	PC1 (A1)	Detecție analogică margine / podea (ADC)
HC-SR04 (Față)	Trig, Echo	PD2, PD3	Măsurare distanță față (GPIO)
HC-SR04 (Stânga)	Trig, Echo	PC2 (A2), PC3 (A3)	Măsurare distanță stânga (ADC folosit ca GPIO)
HC-SR04 (Dreapta)	Trig, Echo	PC4 (A4), PC5 (A5)	Măsurare distanță dreapta (ADC folosit ca GPIO)

3. Diagrame de Semnal

Pentru interfațarea corectă a perifericelor s-au analizat următoarele semnale cheie:

A. Diagrama de semnal pentru senzorii HC-SR04 (Time-of-Flight) Microcontrolerul emite un impuls logic HIGH de 10µs pe pinul Trigger. Senzorul răspunde cu un semnal HIGH pe pinul Echo, a cărui durată este proporțională cu distanța. Această durată este măsurată hardware folosind întreruperile externe și un Timer.

B. Diagrama semnalului PWM pentru L298N Controlul locomoției și aplicarea metodei de "Soft Start" se realizează prin modularea lățimii impulsurilor (PWM) aplicate pe pinii ENA și ENB ai driverului, dictând astfel cuplul și viteza motoarelor.

Software Design

1. Mediu de dezvoltare

Pentru scrierea, compilarea si incarcarea codului pe microcontroler (ATmega328P), am renuntat la clasicul Arduino IDE in favoarea **Visual Studio Code (VS Code) impreuna cu extensia PlatformIO**.

Aceasta alegere ne-a oferit un flux de lucru mult mai profesional. PlatformIO permite impartirea codului in mai multe fisiere (cum ar fi separarea logicii de Bluetooth in `hc05_comm.hpp`), ofera un sistem de autocomplete mult mai destept si gestioneaza automat compilarea fara sa ne incurcam in setari ascunse. Desi folosim framework-ul Arduino pentru usurinta, mediul VSC + PlatformIO ne-a permis sa scriem un cod mult mai curat si modular.

2. Librarii si surse 3rd-party

Ca sa pastram o viteza de reactie maxima a robotului si sa nu incarcam memoria procesorului cu cod inutil, am decis sa limitam la extrem folosirea librariilor externe. Astfel, am folosit doar:

- **<Arduino.h>** - Importata automat de PlatformIO, esentiala pentru functiile de baza (`millis()`, `analogRead()`, `digitalWrite()`).
- **<SoftwareSerial.h>** - Libraria standard folosita pentru a crea un port serial virtual pe alti pini, necesara pentru comunicarea cu modulul Bluetooth HC-05.
- **"hc05_comm.hpp"** - Un fisier header propriu creat pentru a pastra functiile de trimitere si primire mesaje separat de fisierul principal `main.cpp`.

Nu am folosit librarii terte (gen NewPing) pentru senzorii ultrasonici tocmai pentru a putea controla manual timpii de asteptare si a preveni blocarea procesorului.

3. Algoritmi si structuri de date

Programul a fost gandit pe o arhitectura **100% non-blocanta**. Asta inseamna ca nu exista nicio instructiune de tip `delay()` in bucla principala, permitand robotului sa citeasca senzorii, sa conduca motoarele si sa trimita telemetrie pe telefon in acelasi timp.

Logica principala se bazeaza pe doua mari concepte:

- **Masina de Stari Finita (FSM):** Creierul robotului este o variabila (`stareRobot`) care ii dicteaza ce face in orice moment:
  - **Starea 0 (Stop):** Asteapta comenzi, motoarele sunt oprite.
  - **Starea 1 (Mers Inainte):** Robotul inainteaza si scaneaza continuu cu cei 3 senzori frontali si cu senzorul de podea.
  - **Starea 2 (Mers Inapoi):** Se activeaza automat daca vede un obstacol sub 30 cm sau daca ramane fara podea. Da cu spatele cateva sute de milisecunde pentru a-si face loc.
  - **Starea 3 (Rotire):** Executa virajul efectiv pe baza deciziei luate in starea 2, dupa care revine in starea 1.
- **Algoritmul de Acoperire Organica (Memoria deciziilor):** Pentru ca robotul sa nu se blocheze intr-o bucla repetitiva (cum ar fi sa faca mereu stanga si sa se invarta in cerc), am implementat un vector `istoricDecizii[10]`. Acesta memoreaza ultimele 10 directii alese. Cand robotul da de un obstacol, algoritmul alege directia de viraj pe baza senzorilor laterali, dar timpul de rotire este generat **aleatoriu** (intre 400 si 900 milisecunde). Daca memoria detecteaza ca robotul face prea multe viraje in aceeasi directie sau se loveste de prea multe ori intr-un timp scurt, declanseaza o masura de urgenta: sterge memoria si executa o rotatie foarte lunga, tot aleatorie, pentru a evada din acel colt al camerei.

4. Surse si functii implementate

Pentru a pune in practica algoritmii de mai sus, codul a fost impartit in cateva functii vitale si puternic optimizate hardware:

- **`setup()` si `loop()`**: In `setup()` initiem pinii si generam un `randomSeed()` citind “zgomotul” de pe un pin analogic liber (A6), asigurandu-ne ca deciziile robotului sunt diferite la fiecare pornire. In `loop()` se ruleaza Masina de Stari si se trimite telemetria spre telefon (voltaj, distanta) la fiecare 1000 de milisecunde, calculat via `millis()`.
- **`getDistanta()`**: Aceasta este functia senzorilor ultrasonici. Secretul ei sta in parametrul de timeout modificat la 3500 microsecunde. Astfel, procesorul nu asteapta dupa ecouri venite de la 4 metri distanta, ci inchide ascultarea la aprox. 60 cm. Asta a eliberat procesorul si a reparat complet lag-ul de pe comenzile Bluetooth.
- **`citesteVccReal()`**: Functia care masoara voltajul real al bateriei LiPo folosind referinta interna de 1.1V a cipului. Aici am folosit tehnica de “Double Dummy Read” (citire in gol a pinilor analogici) pentru a da timp condensatorului intern din microcontroler sa se goleasca, evitand astfel

combinarea valorilor de la senzorul de podea cu cele de la baterie.
- **`salveazaDecizie()`**: Functia care gestioneaza memoria circulara de 10 pozitii, adaugand noul viraj (Stanga sau Dreapta) in vector si mentinand logica de evadare alerta.
- **Functiile de miscare** (`mergiInainteIncet()`, `rotesteStangaIncet()` etc.): Setari rapide de stare logica (HIGH/LOW) si PWM pe pinii L298N pentru a controla puntea H fara a aglomera bucla principala.

Concluzii

Proiectul ne-a demonstrat ca o placuta simpla cu microcontroler ATmega328P poate sa gestioneze multi senzori si sa ia decizii destul de destepte, daca codul este scris eficient si curat.

Cea mai mare lectie invatata a fost legata de functiile blocante. Pentru un robot care trebuie sa reactioneze in timp real si sa asculte si de comenzi prin Bluetooth, instructiunile de tip delay() sau timeout-urile prea mari la senzori sunt complet interzise. De asemenea, am vazut ca limitarile fizice ale cipului (cum ar fi un singur voltmetru intern impartit la toti pinii) pot fi pacalite usor prin trucuri software, cum sunt citirile duble.

Din punct de vedere al traseului, am realizat ca daca lasam robotul sa faca doar viraje fixe (de exemplu, mereu la 90 de grade), acesta va ramane blocat rapid intr-o bucla infinita pe acelasi traseu. Introducerea algoritmului cu timpi de viraj aleatori si stocarea istoricului de decizii au fost cheia pentru a obtine un comportament mult mai inteligent si o acoperire foarte buna a intregii camere.

Download

GitHub: https://github.com/raresciociea/swiffer-autonom

Jurnal

13.05.2025 - Au fost lipite cu cositor firele care duc la motoare si la divizorul de tensiune. Dovada ca etapa hardware a fost indeplinita este transmiterea pe serial Bluetooth a citiriilor senzorilor, si rotirea motoarelor.

Link test roti:
https://drive.google.com/file/d/1JDcxumcGRbAPboI4evX0atG_J3Q3IFuz/view?usp=drivesdk

20.05.2026 - Am rezolvat problema uriasa de lag de pe Bluetooth, din cauza careia trebuia sa apasam de multe ori pe telefon ca sa prindem comanda de Start sau Stop. Problema era de la pulseIn(), care bloca procesorul timp de 25 de milisecunde pentru fiecare senzor ultrasonic cand robotul era in spatiu deschis. Am scazut timeout-ul la 3500 microsecunde (suficient pentru vreo 60cm in fata), iar acum aplicatia de pe telefon raspunde instant, din prima.

22.05.2026 - Am reparat citirea divizorului de tensiune pentru baterie. Dupa ce am scos delay-urile din cod, telemetria incepuse sa arate aiurea o valoare blocata in jur de 3V, desi la multimetru aveam 8.4V. Bug-ul era cauzat de schimbarile extrem de rapide de pe multiplexorul ADC intre senzorul de podea (A1) si baterie (A0). Am rezolvat prin adaugarea unei “citiri in gol” (dummy read) si a unei micro-pauze de stabilizare hardware, curatand condensatorul intern al ADC-ului. Acum tensiunea afisata este stabila si corecta.

24.05.2026 - Am imbunatatit radical algoritmul de acoperire a camerei. Am renuntat la timpii ficsi de viraj care invarteau robotul in aceleasi zone si am trecut pe timpi de rotire complet aleatori (intre 400ms si 900ms). In plus, am activat si memoria ultimelor 10 decizii salvate intr-un vector. Daca robotul se loveste des in acelasi loc sau face aceeasi intoarcere la nesfarsit, sistemul isi da seama ca e blocat, reseteaza istoricul si forteaza o manevra lunga de evadare (U-Turn) ca sa iasa din colturi.
Link Demo: https://drive.google.com/file/d/1l5hyUvKaTeUjokgSLg2P7t9O1l73x-RN/view?usp=drivesdk

Bibliografie/Resurse

===== Resurse Hardware (Datasheet-uri) =====

- **ATmega328P Xplained Mini** - User Guide-ul oficial Microchip (utilizat pentru maparea pinilor fizici ai placii si schemele de alimentare): ATmega328P Xplained Mini User Guide (PDF)
- **Microcontroler ATmega328P** - Datasheet complet (utilizat pentru intelegerea arhitecturii, registrilor interni, Timere, ADC, USART si intreruperi): ATmega328P Datasheet (PDF)
- **Driver Motoare L298N** - STMicroelectronics Datasheet (consultat pentru limitele de curent, caderea de tensiune pe puntea H si logica de control): L298N Datasheet (PDF)
- **Senzor Ultrasonic HC-SR04** - Specificatii tehnice (utilizat pentru extragerea diagramelor de semnal si a formulei de calcul a distantei bazate pe viteza sunetului): HC-SR04 Datasheet (PDF)
- **Modul Bluetooth HC-05** - Manual de utilizare (consultat pentru tensiunile de operare si pinii de comunicatie RX/TX): HC-05 Datasheet (PDF)
- **Senzor Infrarosu TCRT5000** - Vishay Datasheet (utilizat pentru curbele de reflectie si intelegerea circuitului emitator-receptor): TCRT5000 Datasheet (PDF)
- **Motoare DC cu Reductor (TT Motors 1:48)** - Specificatii generice pentru motoarele galbene de sasiu (utilizate pentru a calcula curentul de stall necesar a fi suportat de L298N): TT Motor Datasheet (PDF)

===== Resurse Software =====

- **OCW UPB - Proiectare cu Microprocesoare:** Suportul teoretic de laborator pentru configurarea perifericelor hardware si notiuni de electronica: ocw.cs.pub.ro/courses/pm
- **Arduino Reference:** Documentatia oficiala pentru arhitectura de cod si functiile standard (millis, pini I/O) utilizate in sistemul non-blocant: Arduino Language Reference
- **WaveDrom:** Unealta utilizata pentru generarea diagramelor de semnal (timing diagrams) din sectiunea de Hardware Design: wavedrom.com

Export to PDF