

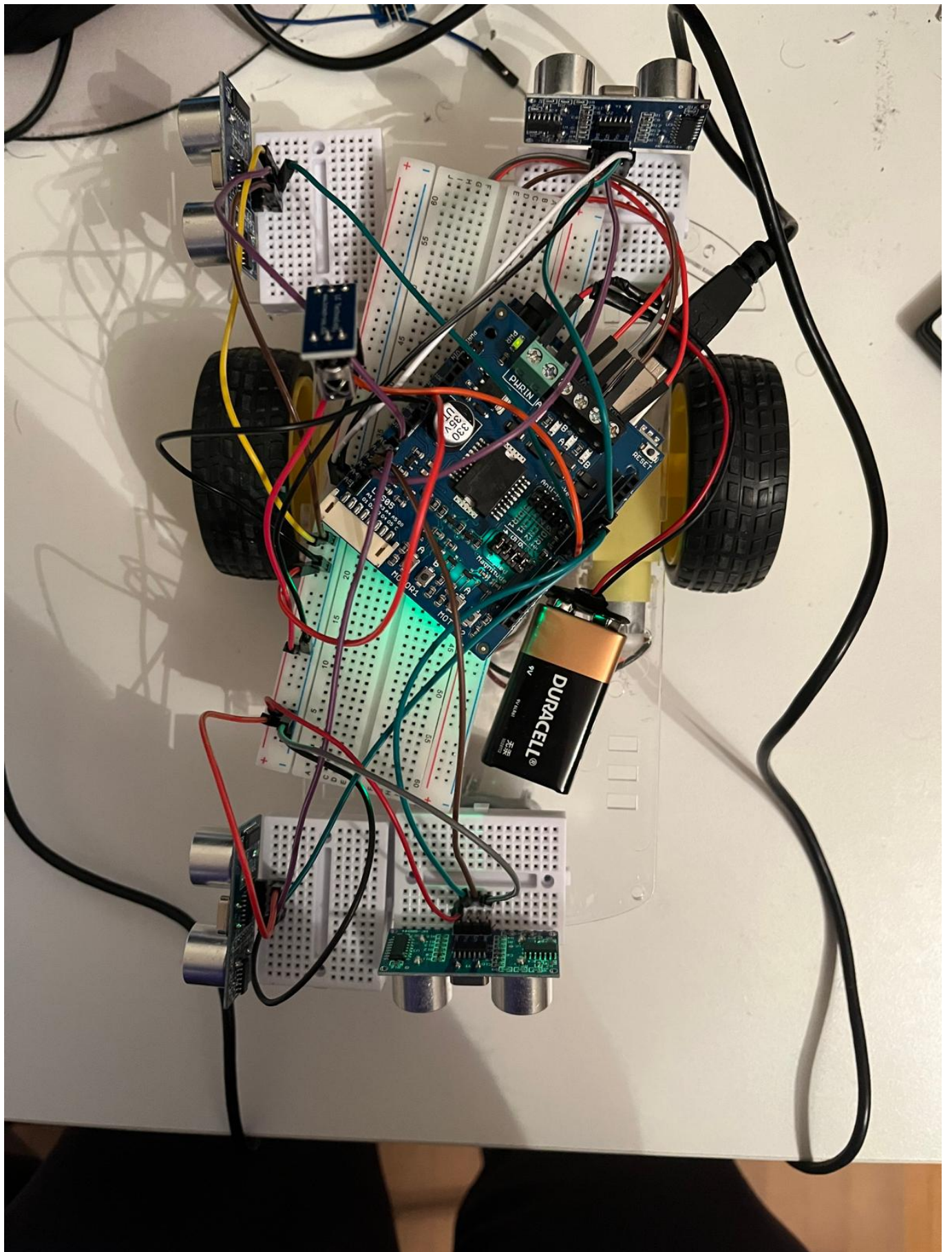
Self Parking Robot

Introducere

- Robotul propus este un robot controlat prin telecomanda cu posibilitatea de a executa manevre de parcare(laterala, cu fata sau cu spatele).
- Scopul lui este atat de invatare pentru mine, dar si de a demonstra anumite mecanisme necesare masinilor autonome.
- Ideea mi-a venit intr-o seara cand ma gandeam la posibile idei de proiecte. Nu am avut un motiv ulterior in spate pentru tema asta, doar mi s-a parut o idee interesanta si provocatoare din punct de vedere tehnic.
- Acest proiect este o posibilitate foarte buna de invatare.

Descriere generală

Robotul este controlat folosind o telecomanda cu infra-roșu, acest semnal este procesat de un Arduino UNO, care trimite un semnal către driverul de motoare care controlează motoarele DC. La primirea unei anumite comenzi, microcontroller-ul începe execuția manevrei de parcare. Prima dată caută un loc potrivit(merge în față până găsește un spațiu gol) se poziționează conform algoritmului manevrei pentru care a primit comanda și începe să o execute până ajunge la o distanță potrivită de marginile zonei delimitate. În timpul manevrei robotul nu mai poate primi altă comandă



Hardware Design

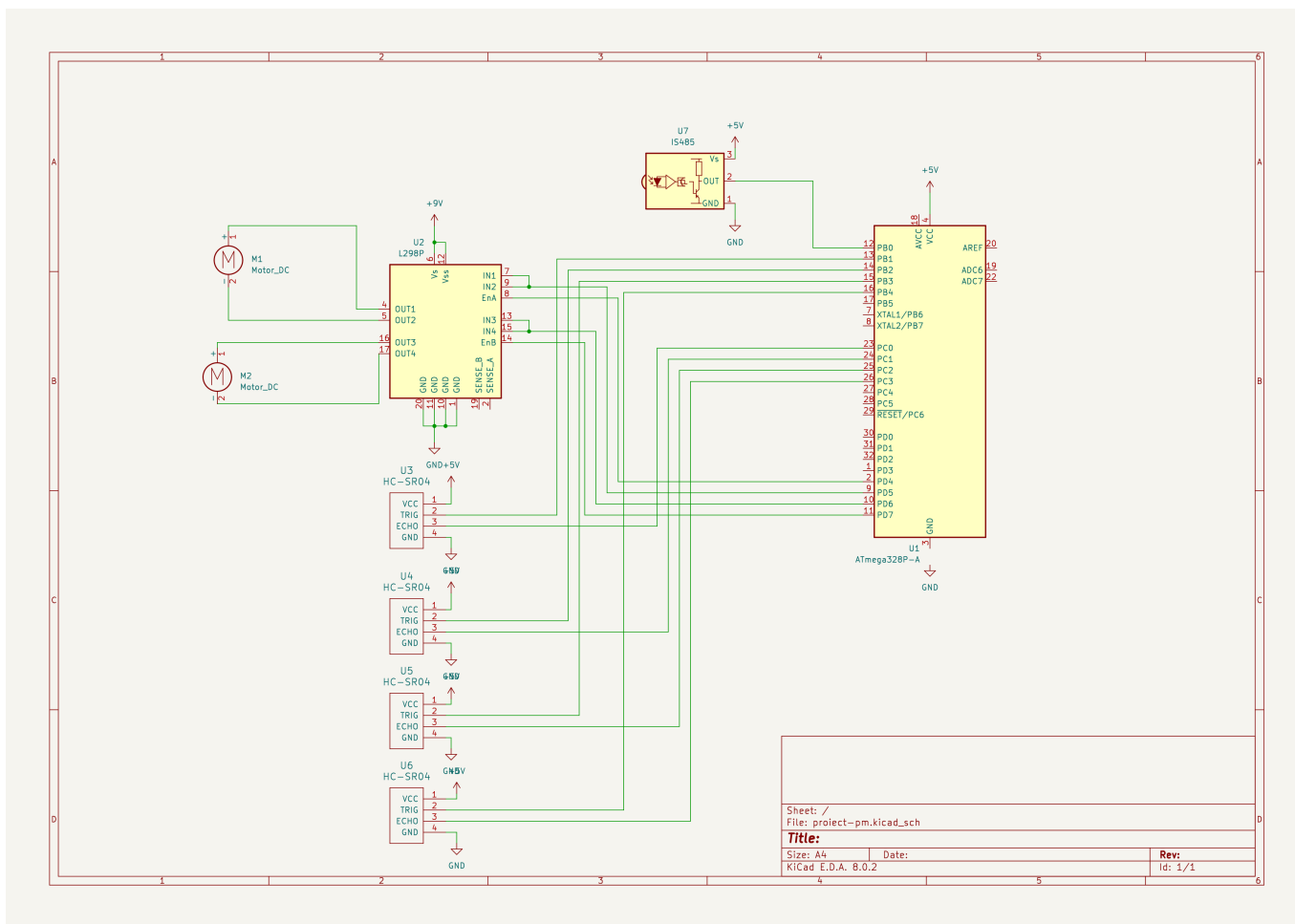
Piese necesare

- Arduino UNO - microcontroler
- Driver motoare - pentru a putea controla motoarele
- Motoare DC - pentru deplasare
- Senzori de distanta HC-SR04 - pentru a detecta obstacole sau puncte de referinta pentru parcare
- Senzor infrarosu - pentru control de la distanta
- Telecomanda
- Suport baetrii 6V - pentru curentul catre driver

Pin out

- PD0, PD1 - senzor ultrasonice fata, e convenabil dpdv al firelor
- PD2, PD4 - senzor ultrasonice lateral, e convenabil dpdv al firelor
- PD11, PD12 - senzor ultrasonice spate, e convenabil dpdv al firelor
- PD13 - senzor infrarosu, e convenabil pentru fire
- PD3, PD5 - motor A, pentru ca sunt pini cu PWM si sunt convenabili ca pozitionare
- PD6, PD9 - motor b, pentru ca sunt pini cu PWM si sunt convenabili ca pozitionare

Schema electrica



Software Design

Prezentarea uneltelor folosite

- mediul de dezvoltare: **PlatformIO**(folosit initial) & **Arduino IDE**(folosit ulterior pentru facilitarea utilizarii IRremote)
- pentru implementare am folosit biblioteca **IRremote** pentru controlul receiverului de infrarosu pentru a nu implementa protocolul NEC si **Arduino.h** pentru afisat pe seriala in general

Detalii de implementare

- Pentru implementare am folosit o buna parte din laboratoare:
 - **GPIO**: cititul senzorilor de distanta si controlul directiei la motoare
 - **PWM**: pentru controlul motoarelor
 - **Interruperi & Timere**: ca sa citesc senzorii de distanta periodic
 - **UART**: pentru debug in special(am si o implementare proprie bazata pe laborator, dar nu a mers foarte bine asa ca am folosit utilitarul expus prin Arduino.h, dar o sa pun si acest lab pentru ca am incercat sa fac o comunicare intre 2 arduino-uri prin UART, dar nu a mai fost nevoie)
- Pentru a face o descriere succinta a arhitecturii proiectului am 3 componente principale:
 - **motor**: controlul motoarelor prin PWM si expune o interfata pentru controlul motoarelor usor de folosit abstractizand logica din spate
 - **distance**: pentru detectarea obstacolelor si referintelor pentru parcare
 - **IRremote**: pentru controlul robotului prin telecomanda
- Aceste componente sunt unite in **main.c**
- Motoarele folosesc in spate PWM, folosesc Timer-ul 0 ambele canale pentru asta, in modul Fast PWM fara prescaler, asta insemnand ca valoarea maxima e intre cea setata in registrii OCR0A/OCR0B si 255
- Senzorul de ultrasonice functioneaza pe un principiu simplu: eu dau un trigger, in spate se trimite o unda ultrasonica care se va intoarce inapoi si va fi detectata de senzor, noi calculam timpul parcurs si folosind formula $(0.000004 * time / 2 * 34300)$ putem afla distanta parcursa de unda.
- Aceste valori au fost aflate part experimental si part cautand pe net in dorinta de a gasi un punct de plecare
- Pentru a calibra timpul dintre 2 citiri am folosit tot metoda experimentală incercand sa gasesc un interval astfel incat sa nu se calce in picioare senzorii

Detectarea obstacolelor

- pentru detectarea distantelor folosesc 4 senzori de distanta cu ultrasonice
- cand senzorul din fata sau cel din spate detecteaza un obiect forteaza oprirea robotului
- pentru a detecta locurile disponibile pentru parcare folosesc senzorii din lateral si caut secventa urmatoare: ambii senzori detecteaza un obiect, doar senzorul din spate detecteaza un obiect, nici un senzor nu detecteaza un obiect, senzorul din fata detecteaza iar un obiect
- prin acest algoritm simplu reusim sa gasim un loc liber si putem incepe manevrele de parcare

Algoritm de parcare

- pentru parcare am decis sa folosesc 2 algoritmi de parcare(parcare laterala si parcare cu spatele)
- parcare laterala:
 - dupa ce pozitionez robotul in dreptul obiectului intorc robotul la 45 de grade
 - dau cu spatele
 - indrept robotul

- înainte puțin dacă este cazul
- parcare cu spatele:
 - întorc robotul la 90 de grade
 - dau cu spatele până sunt aproape de un obiect

Rezultate Obținute

Rezultat final

* Rezultatul principal de care sunt cel mai mandru este că am obținut ceva care merge, folosind o bună parte din ce am învățat la laborator. Asta a fost și motivul pentru care am implementat folosind AVR-C ca să pot să pun în aplicare și să înțeleg mai bine ce am folosit la laboratoare semestrul asta.

* Rezultatele sunt multumitoare, sigur it's far from perfect și probabil o să mai lucrez la acest proiect ca un mini proiect personal, dar în opinia mea e mult mai importantă călătoria și faptul că am fost pus într-un context pe care nu îl știam super bine (partea de hardware)

Concluzii

* M-am confruntat cu destule dificultăți pe parcurs, câteva dintre ele fiind: * Am purtat lupta vieții mele cu driverele de motor, am schimbat 3 drivere: unul a venit ars și îmi scotea pe un pin de input 5V, unul avea 4 pini de în care cereau PWM și am rămas rapid fără timere pe care să calculez PWM-ul pentru că aveam nevoie de timere și la receiver-ul de IR și la senzori cu ultrasonice și ultimul care a fost un succes (al 3-lea e mereu cu noroc) * Si senzorii cu ultrasonice mi-au dat batai de cap și în special faptul că eu aveam nevoie de 4 * Partea de lucru cu infraroșu a fost cam crâncenă și până la urmă am ajuns să folosesc o bibliotecă pentru că era mai ușor * După ce am trecut peste aceste probleme implementarea a fost straight-forward (aveam componentele separat testate și implementate și a trebuit să le pun cap la cap) * Implementarea curentă nu include toate încercările și abordările la care m-am gândit și care nu au mers (să folosesc un singur pin de echo pt senzori de distanță, să conectez 2 Arduino-uri unul pentru senzori și unul pentru motoare, să implementez algoritmul NEC pentru infraroșu, lipiturile făcute, încercarea de a adăuga un buzzer și LED-uri pentru semnalizare) pe care le-am scos pentru că nu funcționau corect împreună cu codul deja scris * Experiența a fost fun și aș mai repeta-o, dar ce aș face mai bine ar fi:

- organizare mai bună ca timp
- scris de teste
- să încerc să vin prima dată cu un PoC înainte să fac restul implementării
- să îmi setez așteptări la joasă

Download

Arhiva cu schemele și codul: Ciulacu Codrut-Cristian, 333CC → [self_parking_robot_codrut_ciulacu.zip](#)

Bibliografie/Resurse

- [Forum AVR](#)
- [laboratoare](#)
- [Datasheet 328p](#)
- [Pinout Arduino](#)
- [Docs arduino pentru Serial printing](#)
- [Forum programare in general](#)
- [paralela pini PCINT_vect](#)
- [Docs shield](#)

[Export to PDF](#)

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

Permanent link:
<http://ocw.cs.pub.ro/courses/pm/prj2024/aungureanu/codrut.ciulacu>



Last update: **2024/05/26 22:44**