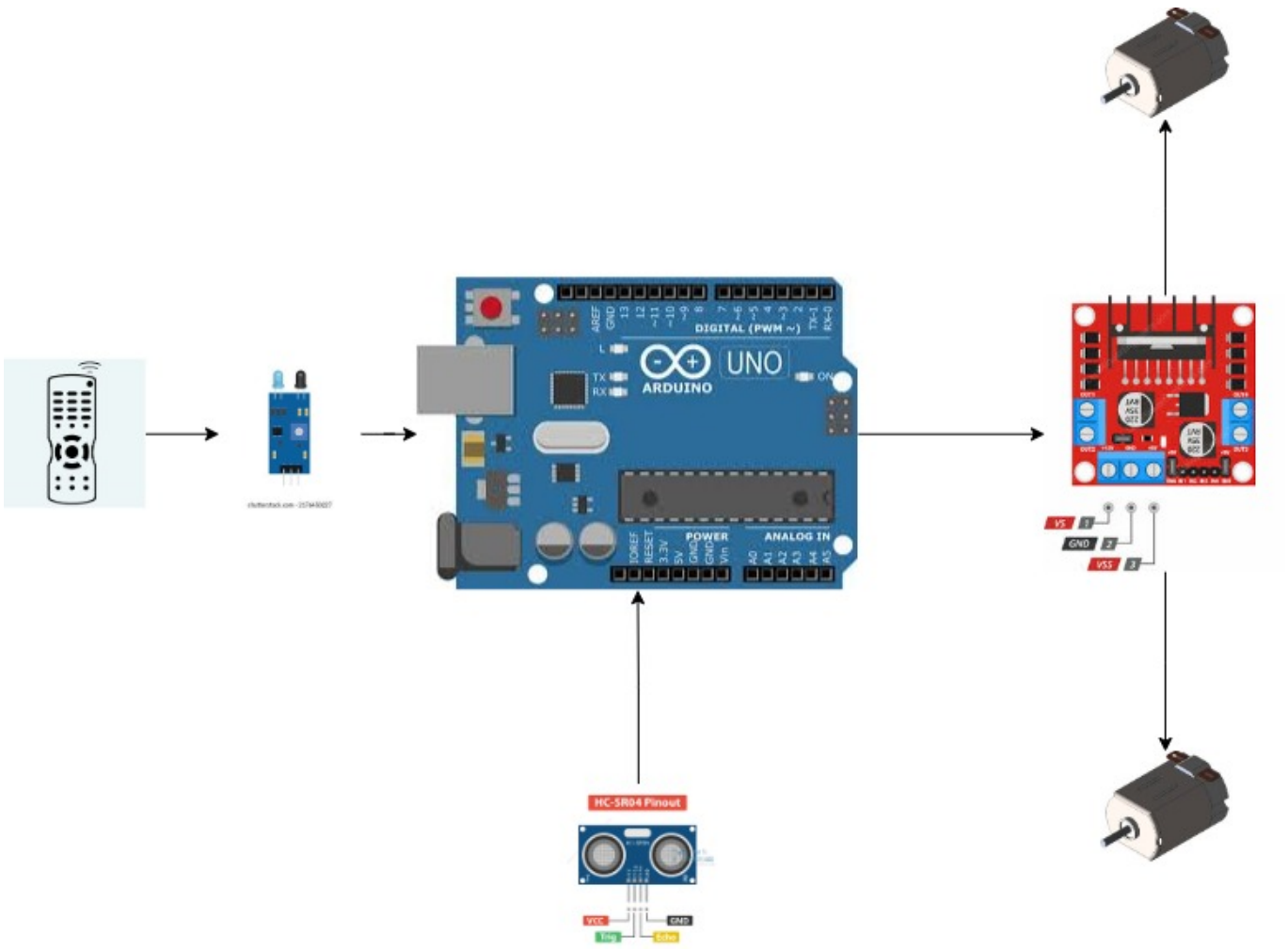
Self Parking Robot

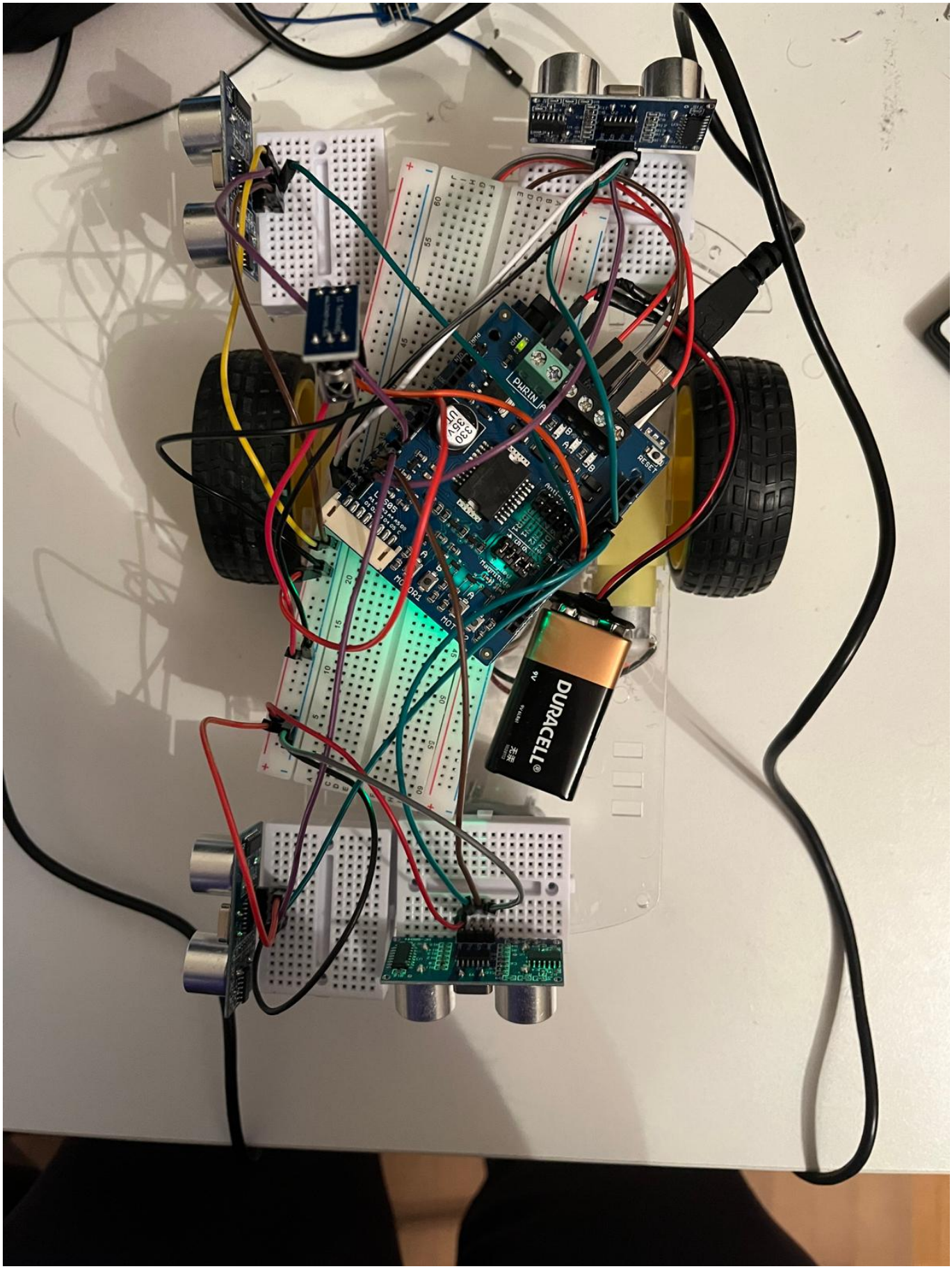
Introducere

- 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-rosu, acest semanl este procesat de un Arduino UNO, care trimite un semnal catre driverul de motoare care controleaza motoarele DC. La primirea unei anumite comenzi, microcontroller-ul incepe executia manevrei de parcare. Prima data cauta un loc potrivit(merge in fata pana da de un spatiu gol) se positioneaza conform algoritmului manevrei pentru care a primit comanda si incepe sa o execute pana ajunge la o distanta potrivita de marginile zonei delimitate. In timpul manevrei robotul nu mai poate primi alta comanda


DIGITAL (PWM ~)
UNO
ARDUINO
POWER
ANALOG IN
HC-SR04 Pinout
VCC
GND
Trig
Echo
VS
GND
VSS


DURACELL
9V
RESET

Hardware Design

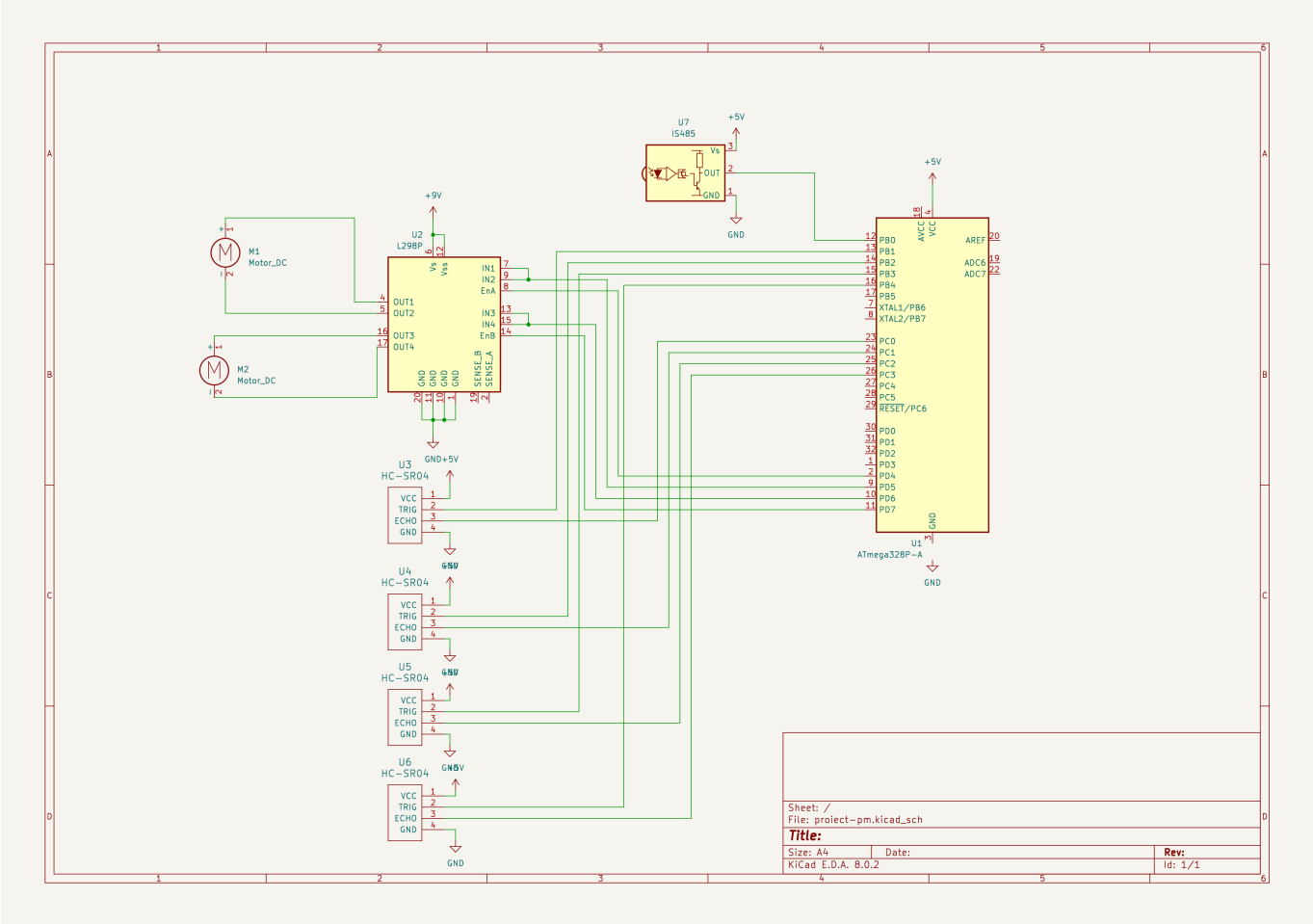
Piese necesare

- 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 facilizarea utilizarii IRremote)
- pentru implementare am folosit libraria **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 moatoare
  - **PWM**: pentru controlul motoarelor
  - **Intreruperi & 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 aceast 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 experimentala 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

Algoritmul de parcare

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

    - inaintez putin daca este cazul
- parcarea cu spatele:
    - intorc robotul la 90 de grade
    - dau cu spatele pana sunt aproape de un obiect

Rezultate Obţinute

Rezultat final

* Rezultatul principal de care sunt cel mai mandru este ca am obtinut ceva care merge, folosind o buna parte din ce am invatat la laborator. Asta a fost si motivul pentru care am implementat folosind avr-c ca sa pot sa pun in aplicare si sa inteleg mai bine ce am folosit la laboratoare semestrul asta.

* Rezultatele sunt multumitoare, sure it's far from perfect si probabil o sa mai lucrez la acest proiect ca un mini proiect personal, dar in opinia mea e mult mai importanta calatoria si faptul ca am fost pus intr-un context pe care nu il stiam suuuper bine(partea de hardware)

Concluzii

* M-am confruntat cu destule dificultati pe parcurs, cateva dintre ele fiind: * Am purtat lupta vietii mele cu driverele de motor, am schimbat 3 drivere: unul a venit ars si imi scotea pe un pin de input 5V, unul avea 4 pini de in care cereau pwm si am ramas rapid fara timere pe care sa calculez pwm-ul pentru ca aveam nevoie de timere si la receiver-ul de ir si la senzori cu ultrasonice si ultimul care a fost un succes(al 3-lea e mereu cu noroc) * Si senzorii cu ultrasonice mi-au dat batai de cap si in special faptul ca eu aveam nevoie de 4 * Partea de lucru cu infrarosu a fost cam crancena si pana la urma am ajuns sa folosesc o librarie pentru ca era mai usor * Dupa ce am trecut peste aceste probleme implementarea a fost straight-forward(aveam componentele separat testate si implementate si a trebuit sa le pun cap la cap) * Implementarea curenta nu include toate incercarile si aproach-urile la care m-am gandit si care nu au mers(sa folosesc un singur pin de echo pt senzori de distanta, sa conectez 2 arduino-uri unul pentru senzori si unul pentru motoare, sa implementez algoritmul NEC pentru infrarosu, lipiturile facute, incercarea de a adauga un buzzer si led-uri pentru semnalizare) pe care le-am scos pentru ca nu functionau corect impreuna cu codul deja scris * Experienta a fost fun si as mai repeta-o, dar ce as face mai bine ar fi:

- organizare mai buna ca timp
- scris de teste
- sa incerc sa vin prima data cu un PoC inainte sa fac restul implementarii
- sa imi setez asteptari la joase

Download

Arhiva cu schemele si 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