Mapping Robot - Luca Seritan 333CA

Introducere

Proiectul meu consta in construirea si programarea unui robotel mic in scopul de a genera in mod autonom harta unei incaperi. Am plecat de la ideea unui aspirator autonom, aceasta fiind si una din posibilele utilitati ale unui astfel de robot. Alte posibile aplicatii ar putea fi explorarea zonelor periculoase/greu accesibile pentru oameni. (Desigur, cu o varianta mai complexa al aceluiasi principiu)

Descriere generală

Robotul are capacitatea de a fi controlat si a transmite in timp real pozitia sa si pozitia obstacolelor pe care le detecteaza pe o retea WiFi. Localizarea este realizata cu ajutorul encoderelor de pe micromotoare, iar detectarea obstacolelor este facuta de trei senzori ultrasonici. Microcontroller-ul ce are la baza chip-ul ESP-WROOM-32 se ocupa pe de o parte de controlul si calculele de pe robot, cat si de comunicarea wireless, dubland ca un AP la retea.

Schema Bloc

Hardware Design

Piese utilizate:

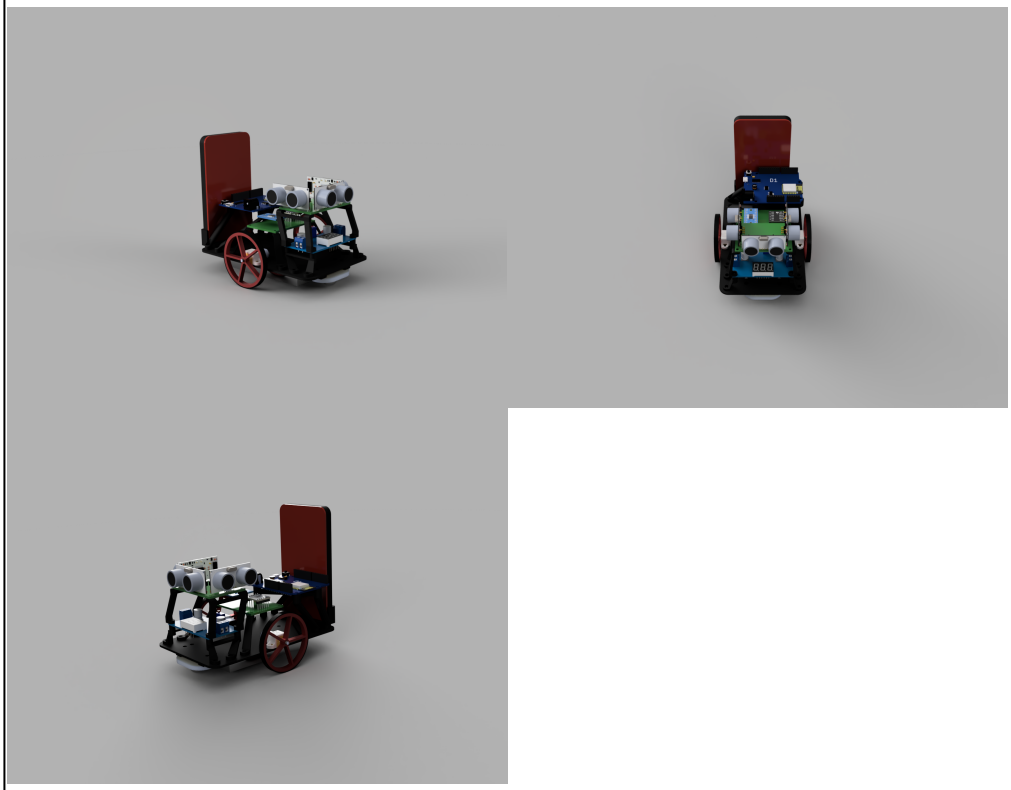
- Placa dezvoltare D1 R32, ESPDuino-32, ESP32, WiFi si Bluetooth
- Micro Motor cu Reductor și Codor Pololu
- Modul driver motor, punte H duala, DRV8833
- Senzor ultrasonic de distanta HC-SR04+
- Sursa stabilizatoare de tensiune LM2596
- Modul Convertor nivel logic I2C bidirectional 8 Biti TXS0108E
- Placa prototipare
- Baterii AA litiu
- Baterie portabila
- Roti
- Ball casters
- Piese printate

Design-ul robotului a fost realizat integral in Fusion360, iar modelul final este open source, disponibil aici.

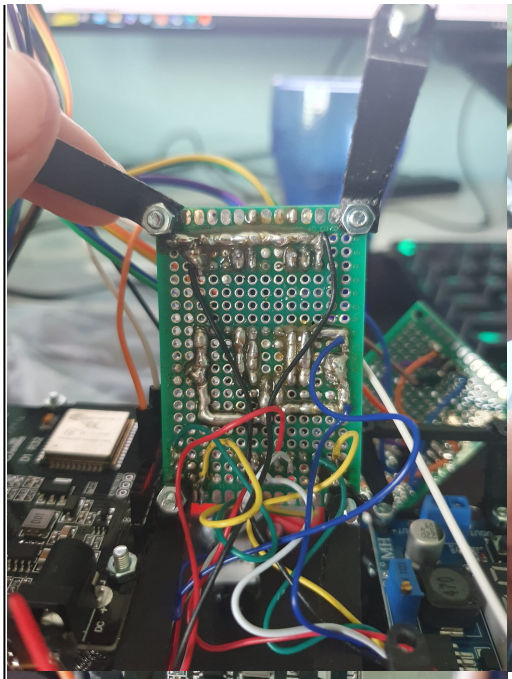
Am urmarit cateva principii de-a lungul realizarii sale:

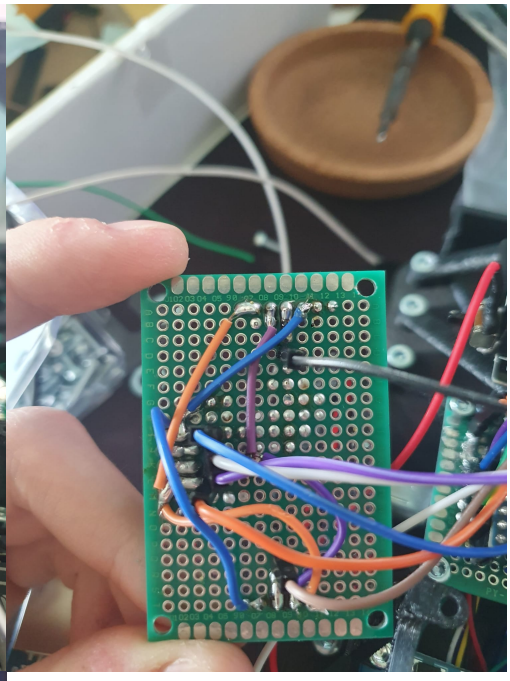
- Sa aiba dimensiuni reduse pentru a putea accesa cat mai multe locuri
- Sa aiba centrul de greutate cat mai jos, pentru stabilitate (de aceea bateriile sunt dedesubt)
- Sa aiba rotile pe centru. Avand doar doua motoare, acest lucru ajuta atat la un control mai bun, cat si o localizare mai simpla (ar fi devenit mai complicat daca avea doar tractiune fata/spate si mai greu de condus)
- Sa fie usor, pentru a nu solicita motoarele si a putea folosi roti cat mai mari - acest lucru ajuta pe de o parte la deplasarea pe un teren denivelat, iar pe de alta parte la o viteza mai mare a robotului.

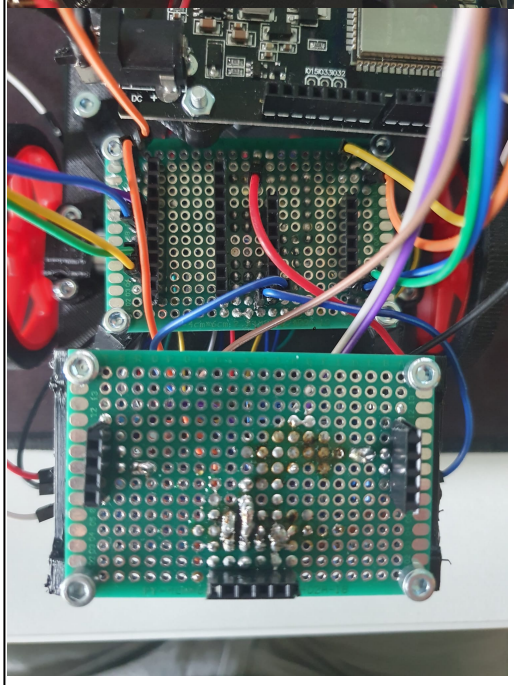
Etapa	Screenshot Fusion360
Forma intiala sasiu	
Amplasare aproximativa componente	
Crearea suportilor	
Adaugare placuta senzori si baterie	
Adaugare componente electrice si cosmetizare	



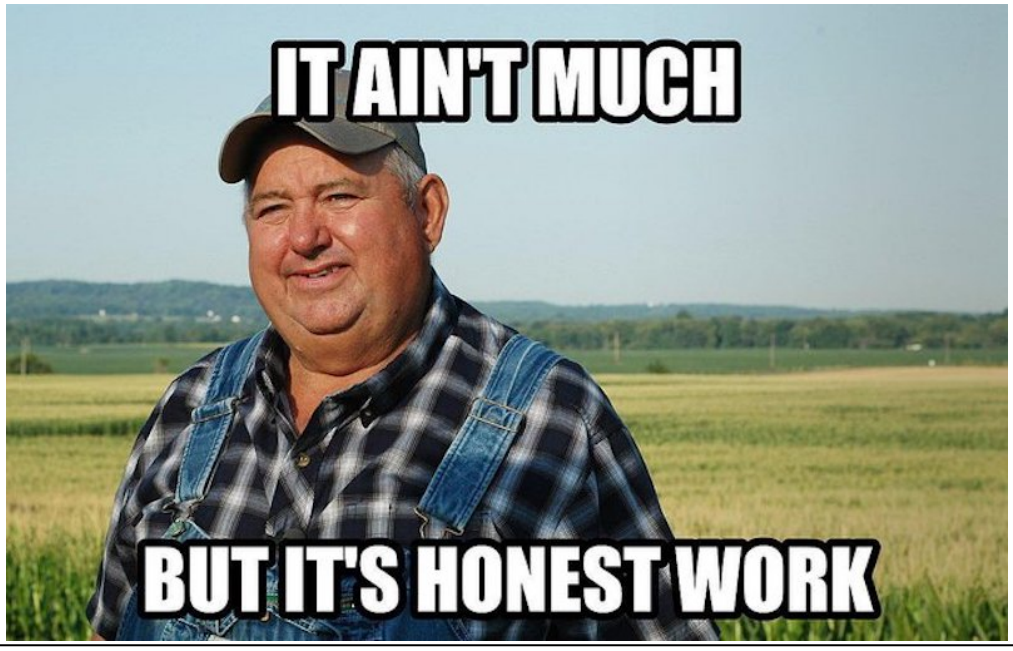
Toate componentele electronice sunt fie montate pe sasiu cu cate un suport special, fie sunt montate pe o placuta de prototipare care este apoi prinsa de sasiu. Am optat pentru lipirea unor headeri de pini pe placutele de prototipare, in locul lipirii direct a pieselor pentru inlocuirea usoara a acestora in caz de defect. De asemenea, nu riscam sa defectez piesele in timp ce lucram la lipituri.







Daca le numaram cu grija, observam ca din placuta din poza centrala, cea care corespunde celor trei senzori ultrasonici ies doar 6 fire (pentru 12 pini). Acest lucru se datoreaza pe de o parte faptului ca GND si 3.3V sunt comune celor trei senzori, iar mai putin evident este faptul ca putem lega in paralel si cei trei pini de pulse ai senzorilor. Acest lucru face economie de **doua fire si doi pini** si pastreaza functionalitatea. (Cand vrem sa citim un senzor, putem trimite o unda din toti senzorii de-o data si sa citim impulsul de echo doar de pe cel care ne intereseaza, iar senzorii nu vor interfera intre ei datorita pozitionarii perpendiculare)



Software Design

Partea software a presupus doua provocari diferite:

- Localizarea robotului
- Comunicarea datelor

Localizare

Unicul mecanism utilizat pentru localizarea robotului sunt encoderele de pe motoare. Valorile encoderelor sunt calculate utilizand intreruperi, iar la fiecare trecere prin bucla este calculata o estimare a deplasarii robotului din momentul ultimului update. Nu putem fi siguri ce traseu a urmat robotul intre doua treceri prin bucla, dar putem *presupune* cum s-a deplasat, iar la o frecventa suficient de mare a acestor update-uri putem avea o localizare *good enough*. Totusi, aceasta metoda acumuleaza erori la fiecare calcul si nu este o metoda fiabila pentru mult timp. Acest lucru ar putea fi imbunatatit prin adaugarea si a altor metode de localizare.

La fiecare trecere prin bucla, putem vedea cat de mult s-au modificat valorile celor doua encodere:

$$\Delta encoder = encoder\_count - previous\_encoder\_count$$.

Putem calcula cat de mult s-a deplasat o roata a robotului, stiind cate tick-uri are un encoder per rotatie:

$$d = \frac{\Delta encoder}{ticks\_per\_rotation} * 2 \pi radius_{wheel}$$

Putem calcula si cat de mult s-a rotit robotul, gandindu-ne ca s-a deplasat de-a lungul cercului pe care se situeaza cele doua roti:

$$\Delta \Theta = \frac{\Delta encoder_{left} - \Delta encoder_{right}}{\pi * distance_{wheels}}$$

Pentru a actualiza pozitia curenta, vom presupune ca robotul s-a deplasat pe o distanta $d_{m} = \frac{d_{left} + d_{right}}{2}$ la un unghi $\Theta_{m}=\Theta + \frac{\Delta \Theta}{2}$, actualizand in final coordonatele:

$$x_{t+1} = x_{t} + d_{m}cos(\Theta_{m})$$ $$y_{t+1} = y_{t} + d_{m}sin(\Theta_{m})$$ $$\Theta_{t+1} = \Theta_{t} + \Delta \Theta$$

Comunicarea

Initial ma gandeam ca robotul sa salveze harta pe un card SD si sa poata fi apoi vizionata. Apoi m-am gandit ca e foarte neinteresant si ca ar fi mult mai smecher sa o transmita in timp real pe laptop pentru vizionare. Aceasta idee a atras dupa sine si o provocare: programul meu ar deveni IO bound, ceea ce ar distruge **complet** ideea de localizare mai sus mentionata.

Din fericire, placuta bazata pe chip-ul ESP-WROOM-32 are capacitati integrate de WiFi, asa ca am ales acest mediu pentru transmitere.

La pornirea codului, robotul creeaza o retea WiFi pentru care este si AP. Apoi asteapta conectarea unui alt dispozitiv la aceasta retea.

Partea de comunicare este compusa din:

- robot
- un client sub forma unui script python, care ruleaza pe un alt dispozitiv

La fiecare trecere prin bucla robotul:

- verifica daca a primit vreun pachet de comanda si actualizeaza vitezele motoarelor
- transmite catre laptop un pachet cu datele curente: pozitie si distanta pana la obstacole

Clientul:

- ia input de la un joystick si trimite un pachet catre robot cu vitezele pe care sa le dea motoarelor
- asculta pentru pachete de la robot si realizeaza in timp real harta pe baza datelor primite

Pentru comunicare am ales sa folosesc protocolul UDP. Am facut aceasta alegere din mai multe motive:

- am vrut o comunicare simpla, direct cu socketi, fara biblioteci complexe care sa adauge un overhead
- eram familiarizat cu acest protocol
- este o aplicatie cu viteza foarte mare, cu multe update-uri si multe pachete. Daca se pierd cateva din cand in cand, nu e *mare chestie*
- nu am vrut un protocol cu gestiunea vitezei de transmitere (precum TCP) deoarece este o aplicatie real time wireless, pe o conexiune care nu stiam de la bun inceput cat de reliable e, iar ultimul lucru pe care mi-l doream ar fi fost ca protocolul sa incetineasca si mai mult comunicarea

Organizarea sarcinilor

Asa cum am mentionat, trebuie ca robotul sa indeplineasca doua task-uri cu sarcini complet diferite:

- localizarea care este CPU bound si are nevoie de o frecventa cat mai mare
- comunicarea care este IO bound si care incetineste cu mult viteza codului

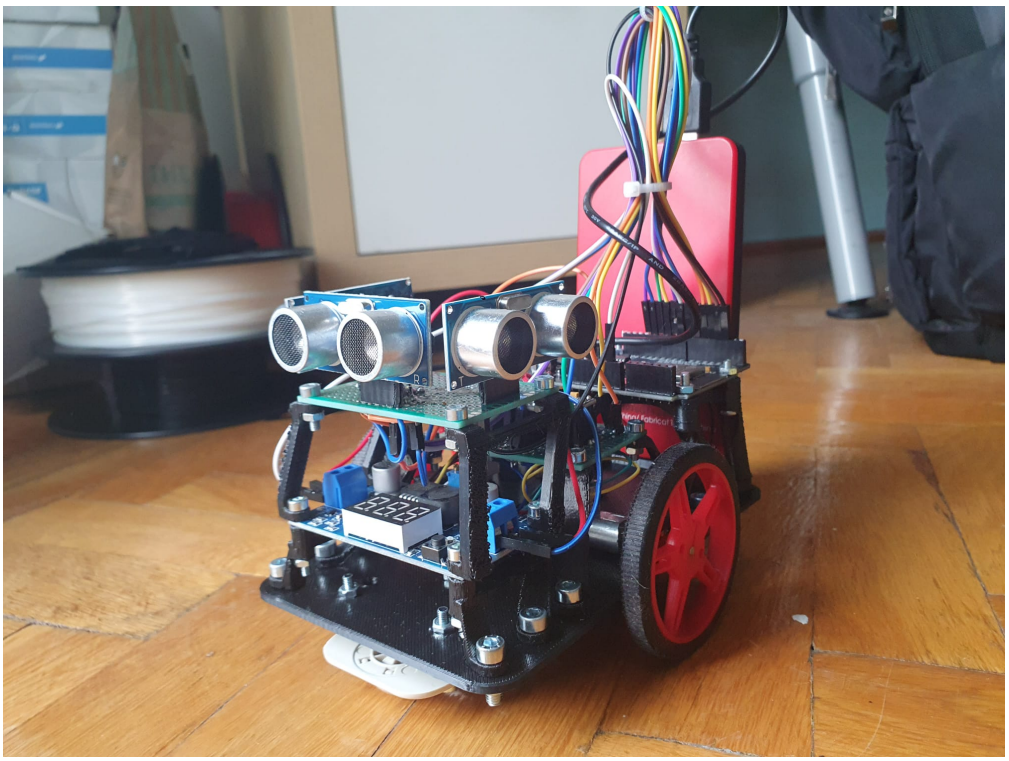
Din fericire, acest microcontroller este **dual core**.

Din acest motiv, am impartit rularea codului pe cele doua core-uri in cel mai firesc mod posibil:

- un core se ocupa strict de calculele de localizare
- un core se ocupa strict de comunicare

Rezultate Obţinute

Robotul este functional (cel putin ca proof of concept).



Concluzii

Sunt foarte multumit de rezultat, am avut multe provocari si cred ca am reusit sa aduc robotul intr-o forma buna. Am invatat multe lucruri de-a lungul acestui proiect si m-am distrat realizandu-l. Totodata, deja am multe idei pentru imbunatatiri ale unor versiuni viitoare:

- Trecerea la un sasiu cu 4 motoare, pentru a putea trece mai usor peste denivelari
- Utlizarea unor metode mai bune de localizare: adaugarea unui giroscop, utilizarea unei camere video
- Utilizarea unor metode mai bune de detectie a obstacolelor: senzor LIDAR, camera video
- Schimbarea metodei de comunicare pentru a fi mai fiabila
- Protejarea canalului de comunicare. Metoda curenta functioneaza, insa sunt sigur ca microcontroller-ul poate fi atacat foarte usor cu un DOS.

Download

Codul este disponibil pentru download pe github.

Bibliografie/Resurse

Datasheet ESP32

Multithreading pe ESP32

Tutorial HC-SR04

Tutorial WiFi ESP32

Export to PDF