

Mapping Robot - Luca Seritan 333CA

Introducere

Proiectul meu constă în construirea și programarea unui robot mic în scopul de a genera în mod autonom harta unei incaperi. Am plecat de la ideea unui aspirator autonom, aceasta fiind și una din posibilele utilități ale unui astfel de robot. Alte posibile aplicații ar putea fi explorarea zonelor periculoase/greu accesibile pentru oameni. (Desigur, cu o variantă mai complexă al aceluiași principiu)

Descriere generală

Robotul are capacitatea de a fi controlat și a transmite în timp real poziția să și poziția obstacolelor pe care le detectează pe o rețea WiFi. Localizarea este realizată cu ajutorul encoderelor de pe micromotoare, iar detectarea obstacolelor este făcută de trei senzori ultrasonici. Microcontroller-ul ce are la bază chip-ul ESP-WROOM-32 se ocupa pe de o parte de controlul și calculele de pe robot, cat și de comunicarea wireless, dublând ca un AP la rețea.

Schema Bloc



Hardware Design

Piese utilizate:

- Placa dezvoltare D1 R32, ESPDuino-32, ESP32, WiFi și Bluetooth
- Micro Motor cu Reductor și Codor Pololu
- Modul driver motor, punte H duală, DRV8833
- Senzor ultrasonic de distanță HC-SR04+
- Sursă stabilizată de tensiune LM2596
- Modul Convertor nivel logic I2C bidirectional 8 Biti TIXS0108E
- Placa prototipare
- Baterii AA litiu
- Baterie portabilă
- Roti
- Ball casters
- Piese printate

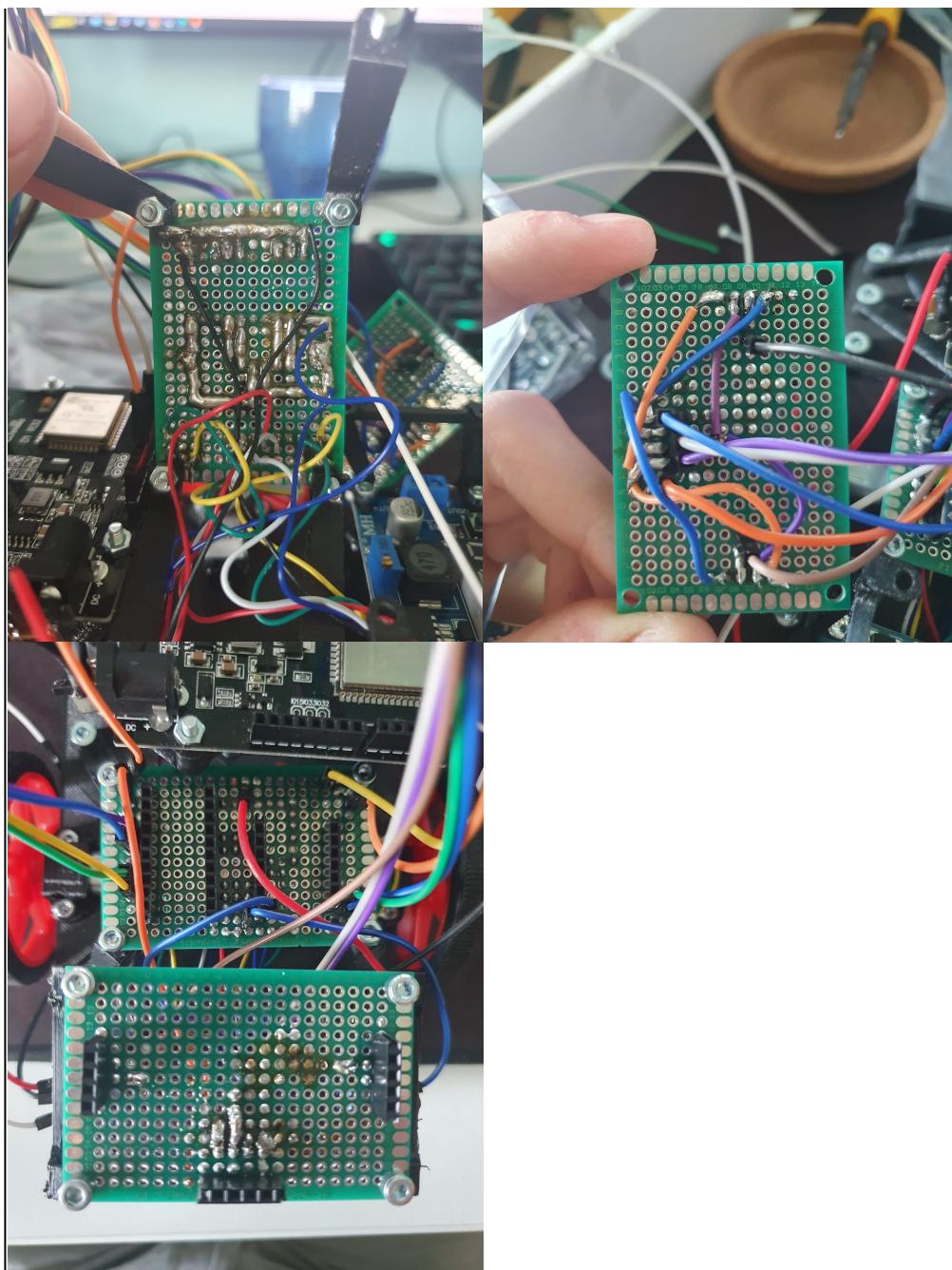
Design-ul robotului a fost realizat integral in Fusion360, iar modelul final este open source, disponibil [aici](#).

Am urmarit cateva principii de-a lungul realizarii sale:

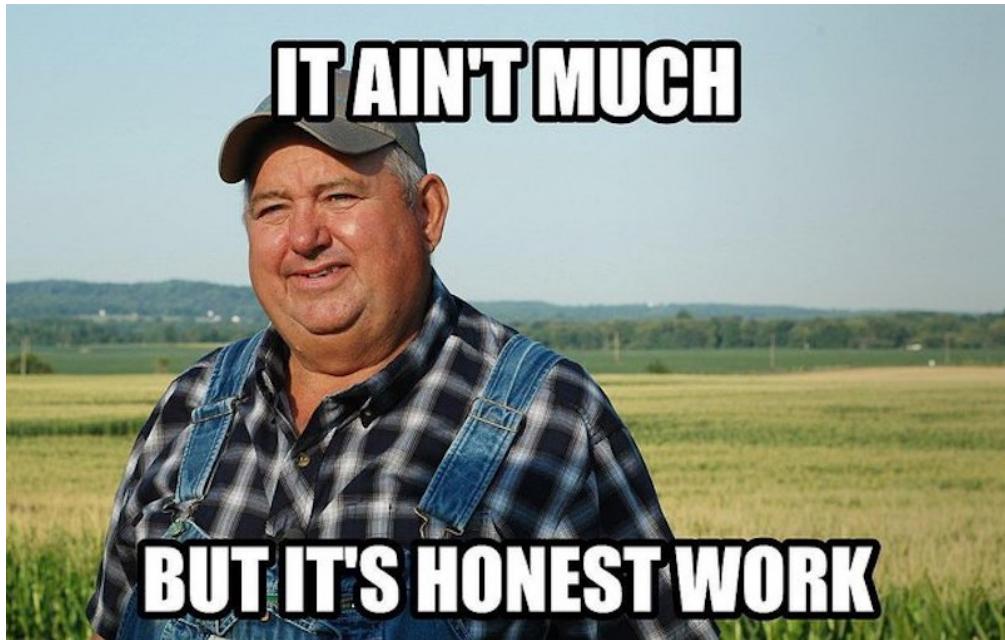
- Sa aiba dimensiuni reduse pentru a putea accesa cat mai multe locuri
- Sa aiba centrul de greutate cat mai jos, pentru stabilitate (de aceea bateriile sunt dedesup)
- Sa aiba rotile pe centru. Avand doar doua motoare, acest lucru ajuta atat la un control mai bun, cat si o localizare mai simpla (ar 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 interfiera intre ei datorita pozitionarii perpendicularare)



Software Design

Partea software a presupus două provocări diferite:

- Localizarea robotului
- Comunicarea datelor

Localizare

Unicul mecanism utilizat pentru localizarea robotului sunt codurile de pe motoare. Valorile encoderelor sunt calculate utilizând intreruperi, iar la fiecare trecere prin buclă este calculată o estimare a deplasării robotului din momentul ultimului update. Nu putem fi siguri ce traseu a urmat robotul între două treceri prin buclă, dar putem *presupune* cum s-a deplasat, iar la o frecvență suficient de mare a acestor update-uri putem avea o localizare *good enough*. Totuși, aceasta metodă acumulează erori la fiecare calcul și nu este o metodă fiabilă pentru mult timp. Acest lucru ar putea fi îmbunătățit prin adăugarea și a altor metode de localizare.

La fiecare trecere prin buclă, putem vedea cât de mult s-au modificat valorile celor două codere:

$\Delta \text{encoder} = \text{encoder_count} - \text{previous_encoder_count}$.

Pot calcula cât de mult s-a deplasat o roată a robotului, știind că cate tick-uri are un encoder pe rotație:

$d = \frac{\Delta \text{encoder}}{\text{ticks_per_rotation}} * 2\pi r_{\text{wheel}}$

Pot calcula și cât de mult s-a rotit robotul, gândindu-ne că s-a deplasat de-a lungul cercului pe care se situează cele două roți:

$\Delta \Theta = \frac{\Delta \text{encoder}_{\text{left}} - \Delta \text{encoder}_{\text{right}}}{\pi * d_{\text{wheels}}}$

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:

$$\begin{aligned} x_{t+1} &= x_t + d_m \cos(\Theta_m) \\ y_{t+1} &= y_t + d_m \sin(\Theta_m) \\ \Theta_{t+1} &= \Theta_t + \Delta \Theta \end{aligned}$$

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 distrugere **complet** ideea de localizare mai sus mentionata.

Din fericire, placuta bazata pe chip-ul ESP-WROOM-32 are capacitatii 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 trimit 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

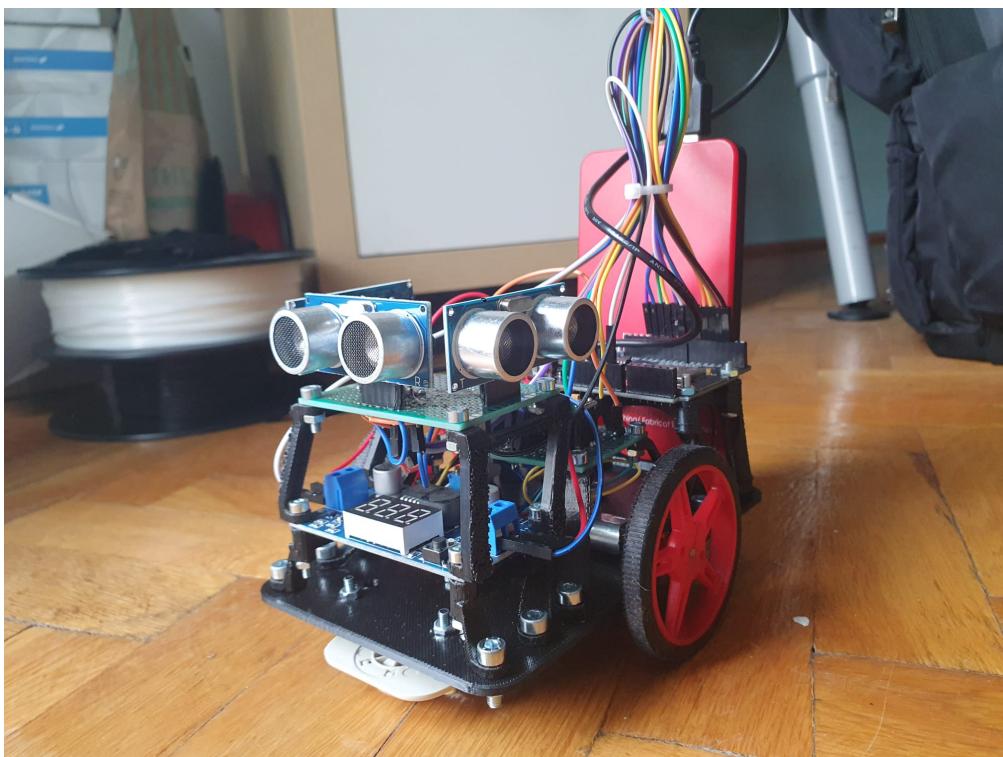
Din fericire, acest microcontroller este **dual core**.

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

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

Rezultate Obținute

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
- Utilizarea 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](#)

From:

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

Permanent link:

<http://ocw.cs.pub.ro/courses/pm/prj2023/adarmaz/room-mapper> 

Last update: **2023/05/29 13:57**