CONSTRUIREA ȘI PROGRAMAREA UNUI ROBOT MOBIL INTEGRAT ÎNTR-O APLICAȚIE DE PARCURGERE A UNUI LABIRINT

DEAC George Antoniu[1], NĂSTASE Robert Paul[2] şi ANCUȚA Andrei Alexandru[3]
[1]Facultatea: IMST, Specializarea: Robotică, Anul de studii: 1, e-mail: chessroby@yahoo.com

Conducător ştiinţific: S. l. dr. ing. Andrei Mario IVAN

REZUMAT: Aceasta lucrare are ca scop implementarea unui robot autonom ce va rezolva și mapa un labirint. Maparea inițiala a labirintului se realizează printr-un algoritm de brute-force similar cu DFS (Depth First Search) într-un sistem de graf bidirectional cu 4 direcții posibile în fiecare nod, cu o matrice de "stack" ce vizează nodurile vizitate și un sistem de orientare similar cu cel din algoritmul LSRB[2] (Left Straight Right Back) sau RSLB[2] (Right Straight Left Back). Dupa maparea inițială, deplasarea între oricare 2 puncte arbitrare din labirint se va putea realiza pe cel mai scurt drum prin algoritmul lui Dijkstra[1]. Orientarea robotului se realizează relativ in funcție de orientarea inițială de la start a primului nod descoperit. Mișcarea și măsurarea distanței parcurse se realizează printr-un sistem de "chunk-uri" (pătrate) cu dimensiune ajustabila.

CUVINTE CHEIE: algoritm brute+force, telecontrol, robot mobil, mapare de labirint.

1. Introducere

Un labirint reprezintă un puzzle ce constituie o rețea de drumuri cu un punct de pornire și unul sau mai multe puncte de finish, care pot fi legate prin unul sau mai multe drumuri.

Obiectivul implementării noastre, față de alte metode mai simple de rezolvare (precum wall follower, regula mainii drepte/stangi), este de a mapa și rezolva un labirint ce poate avea mai multe puncte de finish, cu mai mult de un drum posibil între acestea și urmarea celei mai scurte rute în urma mapării. O altă problemă rezolvată de solutia noastră o reprezinta prezența buclelor în labirint sau a pereților nefixați de margine, un algoritm mai simplu, fără mapare de tipul wall follower ar ramane blocat în buclă în aceste condiții, neputând discerne diferența dintre nodul actual și nodurile deja vizitate.

Detectarea peretilor se realizează pe bază de proximitate, prin rotirea unui senzor ultrasonic in 3 directii perpendiculare pe axele centrale ale gabaritului (față, stânga, dreapta) encodate in regim absolut prin 2 fotocelule pentru poziționare.

O altă caracteristică a robotului este posibilitatea de telecontrol prin protocolul bluetooth, implementarea curentă suportând joystick-uri wireless precum DualShock3, DualShock4 (v1 și v2), Xbox One sau alte controllere care folosesc o schema similară de comunicație în protocolul HID (human interface device).

2. Strategia abordată

În prima etapă (vezi figura 1), robotul mapează toate nodurile labirintului (un nod este determinat când există cel puțin o schimbare de direcție sau se întâmpină o fundătură). Aici robotul străbate labirintul într-o maniera similară cu algoritmul LSRB[2], dar cu cateva diferente:

1. Se ține cont de nodurile deja descoperite pentru evitarea buclelor. În cazul întâmpinării unei bucle, robotul o interpretează precum o fundatură. Atât în cazul întâmpinării fundăturilor cât și a buclelor, robotul revine la ultimul nod mapat nevizitat și continuă maparea de acolo [2].
2. Decizia de selectare a direcției într-un nod este aleatoare pentru a oferi o probabilitate mai bună de atingere mai rapidă a finishului (cel puțin in cazul labirinturilor cu un singur finish)
3. În cazul atingerii punctului de finish, robotul nu se oprește, ci continuă maparea (se oprește doar in cazul in care se precizează dinainte la parametrii ca labirintul are o singură rezolvare).

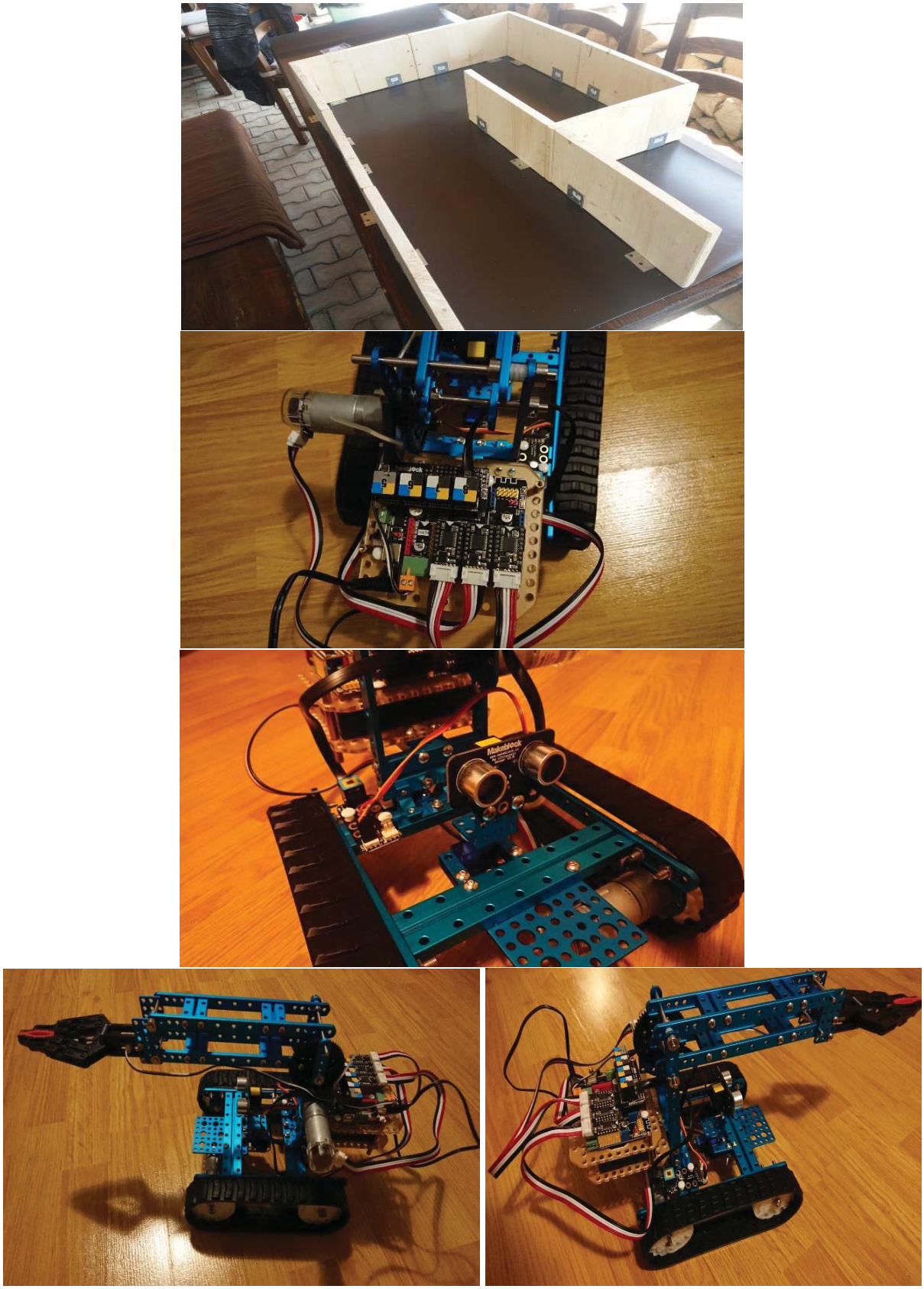

Fig. 1. Prototipul labirintului si al robotului

4. În cea de-a doua etapă, navigarea între oricare 2 puncte arbitrare se poate realiza pe cel mai scurt drum prin algoritmul lui Dijkstra [1] ce are ca "weights-uri" distanța dintre noduri.

De asemenea, în cea de-a doua etapă este posibilă implementarea de task-uri secundare folosind suita de senzori disponibili, de exemplu: manipularea obiectelor prin prehensiune, deosebirea intre obiecte folosind senzorul de culoare etc.

Din punct de vedere mecanic, robotul dispune de un braț articulat cu o axă de rotație pe y și un gripper pentru prehensiune. Deplasarea se efectuează pe 4 roți dințate: 2 active (acționate prin servomotoare) și 2 pasive, legate între ele printr-o curea de cauciuc de tip șenilă. Robotul dispune în total de 3 servomotoare cu encodere rotare cu rația 1:46 la șenile, respectiv 1:75 la braț, un motor 1:50 pentru acționarea griperului și un motor cu reductor încorporat fără encoder la flanșa senzorului ultrasonic. Toate motoarele, mai puțin cel de la flanșa senzorului ultrasonic, au o tensiune nominală de 12V. Motorul de la senzor, neavând o sarcină mare solicitantă, este acționat la 5V fără driver dedicat, direct din pinii analogici de pe PCB.

3. Probleme întâmpinate și optimizări aduse

Datorită iregularităților mari între motoare și a fenomenelor mecanice imprevizibile, corectia PID (proportional, integral, derivative) inclusă în librăria standard nu este eficienta pe distanțe mari. Apare atât problema abaterii de la direcție, cât și problema abaterii de la distanța unitate (pasul encoder-ului rotar)

Posibilele soluții prevăzute sunt:

1. Implementarea unui sistem de "chunk-uri", pătrățele cu unitate predefinită. Astfel după fiecare chunk parcurs, robotul se oprește și își corectează pe loc traiectoria în funcție de micropașii numărați de encoder-ul rotar

2. Folosirea unei accelerații mai mici prin a varia parametrul de viteză (prevăzut în librăria folosită) în timp sau direct prin modificarea dinamică a modulației prin pulsuri (PWM) a driverelor motoarelor

3. Plănuirea dinainte a vitezei fiecărui motor pentru mișcarea în următoarea zonă în funcție de distanța dintre pereți și senzorul ultrasonic, iar dacă robotul nu se află între 2 pereți vom avea un caz similar, care este de asemenea prevăzut (vezi figura 2).

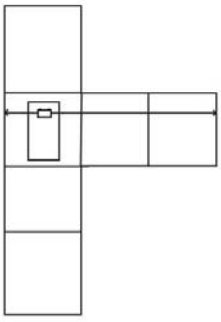

Fig. 2. Planificarea vitezei pe zone

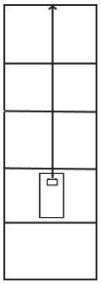

Fig. 3. Corectarea distanţei

4. Pentru corectarea distanței se poate efectua o ajustare dinamică în fiecare zonă în funcție de distanța din fața robotului (vezi figura 3).

Datorită memoriei SRAM limitate de 8KB a microcontroller-ului ATmega2560 și mărimea semnificativă a librăriilor utilizate: pentru encodere, PWM, senzori și pentru shieldul USB[4] (ce oferă posibilitatea de a comunica prin bluetooth cu device-urile de tip HID), s-a ridicat problema reducerii memoriei folosite activ (în cazul labirinturilor mari) .

Posibilele soluții prevăzute sunt:

1. Reducerea numărului de noduri: nu se mapează fiecare unitate parcursă din "chunk" ca nod, doar când se întâmpină o schimbare de direcție sau o fundătură. Așadar fiecare nod va conține 4 pointeri și 4 distanțe, iar dacă se va dori navigarea într-o anumită porțiune de drum care nu este nod, aceasta se va realiza prin numărarea chunk-urilor de la unul dintre cele 2 noduri care constituiesc drumul până în aceea regiune [3].

2. Implementarea unui sistem de swap, care să mapeze labirintul într-o memorie flash externă și să încarce dinamic porțiunile vecine chunk-ului curent

De asemenea, pot apărea iregularități ocazionale ale senzorului ultrasonic. Posibilele soluții prevăzute se referă la întocmirea mediei aritmetice între mai multe citiri și reluarea citirilor diferite cu mai mult de N% față de media celorlalte (procentul ar putea fi modificat în funcție de distanța citită, deoarece variațiile neprevăzute cresc odată cu apropierea de range-ul maxim al senzorului).

În momentul de față lucrarea se află în următorul stadiu (vezi figura 4):

1. Sistemul de deplasare în chunk-uri unitate este aproape gata (trebuie adaugate metodele de corecție dinamică descrise mai sus)

2. Telecontrolul prin joystick este funcțional din punct de vedere al codului, trebuie extinsă interfața de comunicare SPI a plăcii de bază [4] pentru a mai suporta un port serial software (viteza limitată de comunicare poate reprezenta o altă posibilă problemă) (figura 4.)

3. Codul pentru rotirea senzorului ultrasonic trebuie testat și trebuie montate fotodiodele

4. Algoritmul lui Dijkstra [1] trebuie adaptat pentru structura specială a grafului (4 pointeri diferiți pentru fiecare obiect de tip nod)

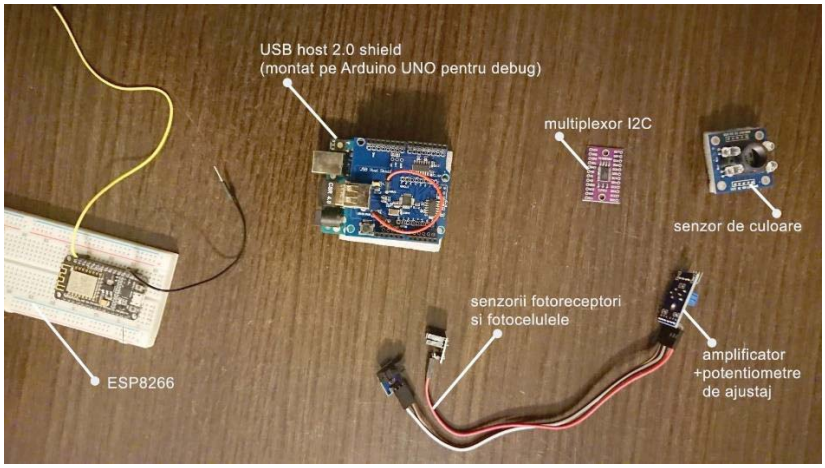

Fig. 4. Componente integrate

Calculul distanței senzor ultrasonic – perete se realizează astfel:

Viteza sunetului la 1atm, 25°C:

$$v \approx 340\ m/s = 0{,}034\ cm/\mu s \quad (1)$$

$$t = x/v = 10/0{,}034 = 294\ \mu s \quad (2)$$

Distanța (vezi figura 5):

$$x = t * 0{,}034 / 2 \text{ (semnalul se reflectă înapoi, parcurgând 2*x)} \quad (3)$$

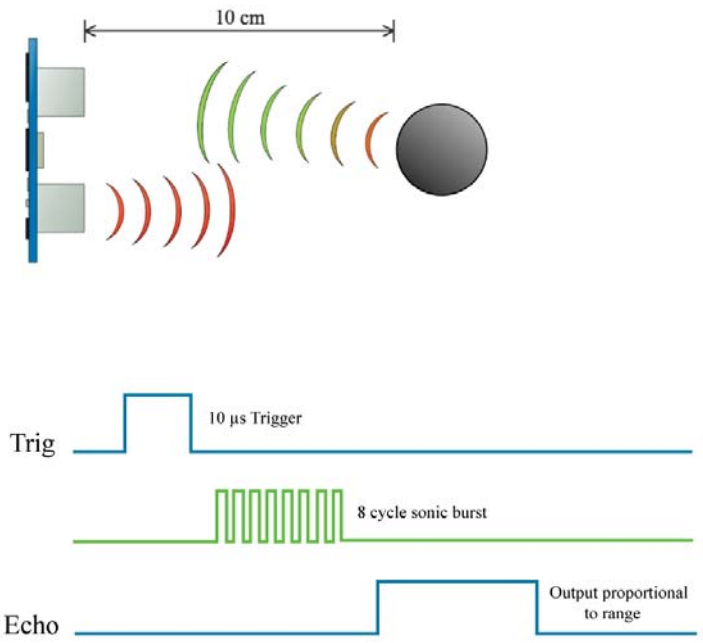

Fig. 5. Reflexia semnalului ultrasonic

4. Concluzii

Algoritmul implementat rezolvă o problemă esențială, proiectul poate fi extins la scală mai mare în diverse aplicații. Robotul va fi capabil să rezolve orice tip de labirint și să aleagă cel mai scurt drum între oricare 2 puncte.

Algoritmul propus este optimizat să aibă o complexitate mică a spațiului și dispune de proceduri de corectare a eventualelor erori ce ar putea fi întâmpinate.

Am modificat un kit standard ce dispunea de o structură modulară cu piese din aluminiu anodizat, plănuim pe viitor să extindem capabilitatea hardware limitată a kit-ului prin adăugarea unui microcontroller secundar espressif ESP8266 pentru implementarea unei interfete web de configurare a parametriilor și vizualizare a traseului mapat, un multiplexor I2C pentru extinderea numărului de adrese (în cazul adăugării de senzori adiționali).

5. Bibligrafie

8. Bibliografie
[1]. https://en.wikipedia.org/wiki/Dijkstra[1]%27s_algorithm
[2]. http://www.ijmer.com/papers/(NCASG)%20-%202013/20.pdf
[3]. https://en.wikipedia.org/wiki/Maze_solving_algorithm
[4]. https://www.circuitsathome.com/store/usb-host-shield-2-0-released/

9. Notații

Următoarele simboluri sunt utilizate în cadrul lucrării:
v = Viteza sunetului la 1atm, 25°C [cm/µs]
x = distanta senzorului ultrasonic [cm]
t = timpul de raspuns al senzorului ultrasonic [µs]