Sonar pe baza de microfon unidirectional

Introducere

Proiectul prezintă un mini-robot mobil care utilizează un **microfon unidirecțional** pentru a detecta o **sursă de sunet** cu o **frecvență specifică**. Robotul se orientează către direcția din care **amplitudinea semnalului audio** recepționat este maximă, folosind un algoritm simplu de scanare și localizare

Descriere generală

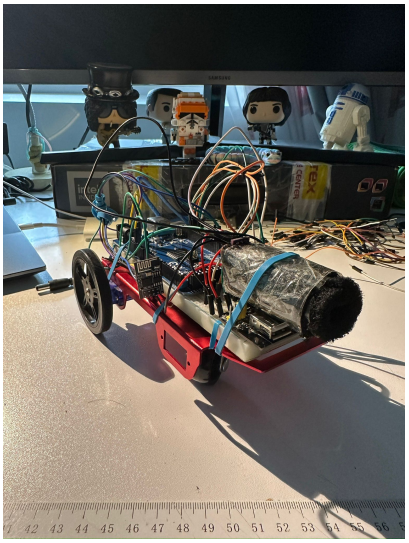
Schema bloc: [Senzor audio (microfon)] → [Amplificator semnal audio] → [Microcontroler (Arduino Uno)] → [Algoritm detecție frecvență și amplitudine] → [Control motoare pentru orientare]

Utilizatorul poate sa emita un sunet pe orice frecventa, care av fi setata din CLI in robot. Acesta, dupa trimiterea comenzii de start va incepe o rotire de 180 de grade pentru a identifica amplitudinea maxima pentru frecventa tinta. Apoi, pentru a detecta directia din care vine sunetul, recurge la o a doua rotire de 180 de grade in sens opus pentru a detecta diferentele de amplitudine cu un threshold mai mic de 30 astfel, detectand directia din care venea sunetul. La introducerea comenzii de exit, serverul se va inchide si programul se va termina.

Hardware Design

Descriere module:

- Microfon (MAX4466) unidirecțional (Handmade): captează sunetele din fața robotului și reduce zgomotul de fundal din alte direcții.
- Microcontroler (Arduino MEGA 2560): evaluează amplitudinea semnalului audio și controlează motoarele pentru orientarea robotului.
- Motoare DC (X2).
- Breadboard pentru montarea driverului de motoare L293D si pentru folosirea mai larga si mai flexibila a pinului de 3.3V (microfon + wifi module esp-10s)
- Modul WiFi ESP01-S
- Breadboard Power Supply Module pentru alimentarea la 5V a driverului pentru motoare



Software Design

* Mediu de dezvoltare: Aplicația este dezvoltată în Arduino IDE, folosind platforma Arduino Mega 2560. Codul este scris în limbaj C++ cu suportul bibliotecilor Arduino standard.

* Librării și surse 3rd-party: Nu sunt utilizate librării externe (3rd-party). Se folosesc doar facilitățile standard oferite de Arduino (Serial, analogRead, etc.).

* Algoritmi și structuri implementate:

- Goertzel analizează semnalul audio de pe pinul analogic (microfon) și calculează „amplitudinea” (puterea) pentru fiecare frecvență dintr-un interval (ex: 200 Hz – 5000 Hz).

Cum funcționează practic:

```
       Se colectează un set de mostre (samples) de la microfon.
       Pentru fiecare frecvență de interes, se rulează Goertzel și se obține
o valoare care arată cât de „puternică” este acea frecvență în
       semnal.
       Se determină care frecvență are cea mai mare amplitudine (frecvența
dominantă).
       Citirea semnalului audio: se face de pe pinul analogic A0, cu o
frecvență de eșantionare de 9 kHz. Sunt colectate 256 de probe.
```

- Comunicare serială:
  - Cum funcționează comunicarea Arduino – ESP-01S cu comenzi AT și UART

1. Rolul componentelor

```
            Arduino este creierul sistemului: citește date de la senzori,
controlează motoare și decide ce și când să comunice.
            ESP-01S (ESP8266) este modulul WiFi, folosit doar ca „modem” –
el se ocupă de conectarea la rețea și de trimiterea/primirea datelor
```

```
            pe internet.
  * Comunicarea între Arduino și ESP-01S se face prin UART (Serial1), adică
două fire: TX și RX, plus GND comun.
        2. Inițializarea și configurarea ESP-01S cu comenzi AT
           La pornire, Arduino trimite o serie de comenzi AT către ESP-01S
pentru a-l configura:
           a) Resetarea modulului ESP-01S
                Comandă: AT+RST
                Ce face: Resetează modulul ESP-01S, ca să fie sigur că
pornește dintr-o stare curată.
           b) Testarea comunicării
                Comandă: AT
                Ce face: Verifică dacă ESP-01S răspunde la comenzi. Dacă
răspunde cu OK, comunicarea funcționează.
           c) Oprirea echo-ului
                Comandă: ATE0
                Ce face: Dezactivează echo-ul, adică ESP-01S nu mai repetă
fiecare comandă primită. Astfel, răspunsurile sunt mai clare.
           d) Setarea modului de funcționare
                Comandă: AT+CWMODE=1
                Ce face: Pune ESP-01S în modul „stație” (client WiFi), ca să
se poată conecta la o rețea wireless existentă.
           e) Conectarea la rețeaua WiFi
                Comandă: AT+CWJAP="SSID","PAROLA"
                Ce face: Conectează ESP-01S la rețeaua WiFi cu numele și
parola specificate. Dacă totul merge bine, răspunde cu WIFI GOT IP.
        3. Conectarea la un server TCP (de exemplu, un PC cu Python)
               După ce ESP-01S este conectat la WiFi, Arduino îi spune să se
conecteze la un server TCP (de obicei, un script Python care ascultă
               pe un anumit port):
               a) Inițierea conexiunii TCP
                  Comandă: AT+CIPSTART="TCP","IP_SERVER",PORT
                  Ce face: Deschide o conexiune TCP către adresa IP și
portul specificate. Dacă reușește, răspunde cu CONNECT.
        4. Trimiterea datelor de la Arduino la server prin ESP-01S
           Când Arduino vrea să trimită date către server (de exemplu,
rezultatul analizei Goertzel), folosește două comenzi AT:
               a) Anunțarea lungimii mesajului
                  Comandă: AT+CIPSEND=lungime
                  Ce face: Spune ESP-ului că urmează să trimită un mesaj de
o anumită lungime (număr de caractere). ESP răspunde cu > dacă e gata
                  să primească datele.
               b) Trimiterea efectivă a datelor
                  Comandă: (nu e o comandă AT, ci datele efective)
                  Ce face: Arduino trimite textul dorit (ex: Dominanta: 700
Hz | Magn: 63.3 | TARGET GASIT!). ESP-01S îl transmite pe internet
                  către serverul TCP.
        5. Primirea comenzilor de la server (PC → ESP-01S → Arduino)
Serverul (de exemplu, un script Python) poate trimite comenzi către Arduino
           (ex: start, 700). Aceste comenzi ajung la ESP-01S prin rețea, iar
ESP-01S le transmite pe UART către Arduino. Arduino citește aceste
```

```
           comenzi cu Serial1.readStringUntil('\n') și le interpretează:
               Dacă primește start, pornește scanarea.
               Dacă primește un număr, setează frecvența țintă.
        6. Răspunsuri și gestionarea erorilor
           Pentru fiecare comandă AT, Arduino așteaptă un răspuns specific
(ex: OK, CONNECT, >, WIFI GOT IP).
           Dacă nu primește răspunsul așteptat într-un anumit timp,
consideră că a apărut o eroare și poate încerca din nou.
           Dacă conexiunea WiFi sau TCP se pierde, Arduino încearcă periodic
să se reconecteze.
        7. Exemple de comenzi AT folosite și explicații
```

| Comandă AT | Explicație |

————————————	———————————————————————————-
AT	Testează dacă ESP-01S răspunde.
AT+RST	Resetează modulul ESP-01S.
ATE0	Dezactivează echo-ul (nu mai repetă comenzile primite).
AT+CWMODE=1	Setează modulul în mod stație (client WiFi).
AT+CWJAP=“SSID”,”PAROLA”	Conectează la rețeaua WiFi cu numele și parola date.
AT+CIPSTART=“TCP”,”IP”,PORT	Deschide o conexiune TCP către IP și portul specificate.
AT+CIPSEND=lungime	Pregătește ESP-01S să primească un mesaj de o anumită lungime.
(date efective, ex: start\n)	Trimite datele efective către server, după ce primește promptul >.

8. Fluxul complet de comunicare Arduino pornește și configurează ESP-01S cu comenzi AT. ESP-01S se conectează la WiFi și apoi la serverul TCP. Arduino trimite date către server folosind AT+CIPSEND și apoi datele efective. Serverul poate trimite comenzi către Arduino (prin ESP-01S), care le interpretează și acționează în consecință. Dacă apare o eroare sau conexiunea se pierde, Arduino încearcă să refacă legătura.

- Controlul fluxului: logica programului se bazează pe o buclă principală (loop()), în care sunt alternate etapele de:

```
    1) Citire din serial (polling)
    2) Achiziție de date
    3) Analiză spectrală
```

* Gasirea sursei sunetului:

```
1) Verifică conexiunea la WiFi și la serverul TCP.
2) Dacă nu e conectat, încearcă periodic să se reconecteze.
3) Ascultă comenzi de la server (prin ESP-01S):
4) Dacă primește o frecvență (ex: 700), o setează ca țintă.
5) Dacă primește start, pornește secvența de scanare (activează modul
SCANNING).
6) Când este în modul SCANNING sau RETURNING și are o frecvență țintă:
   a) Colectează mostre de la microfon.
   b) Rulează Goertzel pentru fiecare frecvență din intervalul dorit.
   c) Găsește frecvența dominantă și amplitudinea maximă.
   d) Trimite rezultatul către server (ex: Dominanta: 700 Hz | Magn: 63.3 |
```

```
TARGET GASIT!).
   e) Controlează motoarele pentru a roti sonar-ul:
      i) În modul SCANNING, face pași de rotație (de obicei la dreapta), la
fiecare pas colectând date.
      ii) După ce a terminat toți pașii (ex: 36 pași pentru 180°), trece în
modul RETURNING (revenire).
      iii) În modul RETURNING, continuă să rotească până când amplitudinea
se apropie de maximul detectat, apoi revine la IDLE (așteptare).
```

* Funcții principale implementate în cod

```
 1. setupMotors()
    Inițializează pinii folosiți pentru controlul motoarelor (direcție și
enable).
    Pune toate motoarele în stare oprită la pornire.
 2. rotateRightStep()
    Activează ambele motoare pentru a roti sonar-ul spre dreapta (un motor
înainte, celălalt înapoi).
    Folosește PWM (putere maximă) pe pinii de enable.
    După un delay (MOTOR_DELAY), oprește motoarele.
 3. stopMotors()
    Dezactivează toți pinii de control ai motoarelor și pinii de enable.
    Oprește complet mișcarea.
 4. sendATcommand(cmd, expectedResponse, timeout)
    Trimite o comandă AT către ESP-01S prin UART.
    Așteaptă un răspuns specific (expectedResponse) sau timeout.
    Returnează true dacă răspunsul a fost primit, false altfel.
    Folosit pentru toate etapele de configurare și trimitere date prin WiFi.
 5. sendATcommandNoReply(cmd, delayAfter)
    Trimite o comandă AT fără să aștepte un răspuns specific.
    Folosit pentru resetare sau golire buffer.
 6. connectWiFi()
    Rulează secvența de comenzi AT pentru a conecta ESP-01S la rețeaua WiFi.
    Setează variabila de stare espConnectedToWiFi.
 7. connectTCPServer()
    Trimite comenzi AT pentru a conecta ESP-01S la serverul TCP (PC/Python).
    Setează variabila de stare espConnectedToServer.
 8. goertzel(samples, numSamples, targetFreq, samplingRate)
    Implementarea algoritmului Goertzel.
    Primește un vector de mostre și returnează amplitudinea pentru o
frecvență dată.
    Folosit pentru a detecta frecvența dominantă în semnalul audio.
 9. readSamples()
    Citește un număr fix de mostre de la microfon (A0) la o frecvență de
eșantionare stabilită.
    Populează vectorul samples pentru analiza Goertzel.
10. setup()
    Inițializează comunicația serială, motoarele și ESP-01S.
    Forțează o primă încercare de conectare la WiFi și server la pornire.
11. loop()
    Bucla principală:
```

```
    Gestionează reconectarea la WiFi și server dacă e nevoie.
    Primește comenzi de la server (frecvență țintă, start).
    Dacă este în mod SCANNING sau RETURNING, citește mostre, rulează
Goertzel pe un spectru de frecvențe, trimite rezultatul la server și
    controlează mișcarea motoarelor.
```

Rezultate Obţinute

Hardware-ul a fost cea mai “tricky” parte deoarece microfonul folosit, de altfel un microfon recomandat pentru proiecte cu Arduino, nu este suficient de performant. Impedanta necesara acestuia a fost neprevazuta, astfel nefunctionand optim fara ca acesta sa aibe o sursa puternica de impamantare cum ar fi osciloscopul folosit in teste (Hantek 2D42). Astfel pentru rezultate optime in analiza sunetului, acesta trebuia sa fie in permananta conectat la osciloscop. Astfel, mobilitatea robotului a fost drastic redusa. Insa, in timp ce acesta era conectat, atat izolarea fonica cu scopul de a face microfonul unidirectional, cat si algoritmul folosit oentru analizarea semnalelor a functionat fara probleme. Cu atat mai mult si comunicarea intre ESP si Arduino, desi complexa, a fost una reusita astfel obtinand si conectivitate prin WiFi (LAN) la un server TCP in python.

Asadar, proiectul a decurs asa cum trebuie, singura imperfectiune si piedica fiind neajunsul hardware al microfonului, in rest scopurile acestui proiect au fost atinse.

Concluzii

Analiza de sunete si semnale poate fi o sarcina prea complexa pentru un ATMEGA, insa cu algoritmii si eficientizarile potrivite acestea se pot realiza. Comunicarea atat intrea deviceuri si WiFi a fost bine servita de comunicarea UART si AT. Intrearuperile de semnal au fost vitale in folosirea motoarelor DC pentru oprirea lor in puncte specifice dar si folosirea PWM a fost un punct principal in utlizarea acestora.

Download

Video despre comunicare wifi din proiect: https://youtube.com/shorts/hZ5QWcgfgBw?si=eNvZWDUsbxx5RMFM Video despre problemele micrfonului: https://youtube.com/shorts/dZWT_SvphCc?si=yuMNT8lP_xU6A4pt