S-Park

Cuprins:

Argument - Utilitate practică

În zilele noastre, numărul mașinilor crește exponențial, problema locurilor de parcare în orașe devenind din ce în ce mai critică. Una dintre modalitațile de rezolvare a acestei problme este construirea mai multor locuri de parcare, cu toate acestea, pe toți ne deranjează clădirile imense din mijlocul orașelor, a căror scop este numai adăpostirea mașinilor. Dificultațile pe care le prezintă soluțiile sctuale sunt costurile mari, necesitatea de spații voluminoase, ineficența din punct de vedere al gestionării cursului mașinilor, din cauza necesității spațiului de manevră, și nu în ultimul rând, aspectul și imposibilitatea integrării în ansamblul arhitectural al zonei în care sunt amplasate, în special în centrele istorice si arhitecturale.

Astfel, proiectul prezentat face parte dintr-unul mai larg, încearcând să rezolve problemele prezentate, prin dezvoltarea unei parcări inteligente, subterane, modulare, de tip carusel, eficiente din punct de vedere al spațiului utilizat, al costului și al timpului de construcție, putând fi cu ușurință integrată în orice spațiu arhitectural.

Partea mecanică

Ideea noastră a pornit de la un concept deja existent, numit Rotary Parking, însă noi l-am adaptat renunțând la roata dințată din partea superioară, care bloca accesul mașinilor atunci cand vehiculele intrau prin partea de sus. Astfel, am ajuns sa realizăm o parcare cu o cupolă unde am amplasat o serie de role pe care lanțurile gondolelor alunecă. Mai mult decât atât, parcarea noastră este **subterană**, spre deosebire de Rotary Parking, care este supraterană.

Capacul cutiei reprezintă nivelul solului, adică nivelul de la care mașinile încep coborârea catre subteran. În interiorul cutiei se află gondolele, placa de dezvoltare ESP32, driver-ul de motor, cititorul de carduri RFID și motorul DC care învârte mecanismul. De acesta sunt atârnate 8 gondole (locuri de parcare). Ele sunt acționate de o roată dințată, în partea inferioară iar în partea superioară se află cupola care elimină nevoia de a avea o a 2-a roata dințată, așa cum se prezintă conceptul Rotary Parking, permițând, astfel, intrarea mașinilor prin partea superioară a sistemului.

În proiectul nostru, placa de dezvoltare ESP32 are 3 funcții principale: comunicarea cu cititorul de carduri, acționarea motorului și determinarea poziției actuale a locului de parcare. Placa comunică cu server-ul și îi trimite informații pentru a afișa pe aplicație date importante, cum ar fi numărul de locuri libere.

Sursele de energie pot varia, datorită ușurinței implementării proiectului, astfel, surse de energie regenerabilă, precum energia solară sau eoliană, pot fi folosite, mai ales când acestea sunt prezente în abundența. Desigur, acestea ar trebui să aibă o sursă de rezervă de energie, în cazurile excepționale, pentru a elimina riscul blocării vreunei mașini, din cauza lipsei de curent.

Modul de funcționare

Proiectul nostru are 3 părți principale:

- Interfața cu utilizatorii, prin intermediul unui modul RFID și prin aplicația web

- Interfața cu server-ul, prin Web Sockets

- Comanda motorului

Utilizatorilor li se vor pune la dispoziție două modalități de interacțiune cu parcarea:

Prima, **fizică**, prin intermediul cardurilor RFID, iar a doua, **virtuală**, comunicând direct cu server-ul prin aplicația de telefon.

În starea de repaus, parcarea, cu excepția situației când aceasta este plină, va prezenta un loc liber. Un utilizator va putea parca în acest spațiu, după care va înregistra parcarea, fie prin acționarea cardului, fie prin aplicația pe telefon. În cazul înregistrării prin card, modulul de comandă va trimite datele cardului către server, așteptând validarea acestora.

În ambele cazuri, după validarea datelor, parcarea va pune la dispoziție următorul loc liber.

Revendicarea mașinii se va face într-un mod similar: persoana va accesa parcarea prin card sau aplicație, server-ul va valida datele, iar în cazul acesta, parcarea va aduce la dispoziția utilizatorului mașina parcată de acesta. După revendicare, va rămâne în aceeași poziție, oferind locul eliberat următoarei persoane.

În cazul în care mai multe persoane încearcă revendicarea mașinilor în același timp, aceste cereri for fi plasate într-o **coadă** și vor fi procesate pe rând.

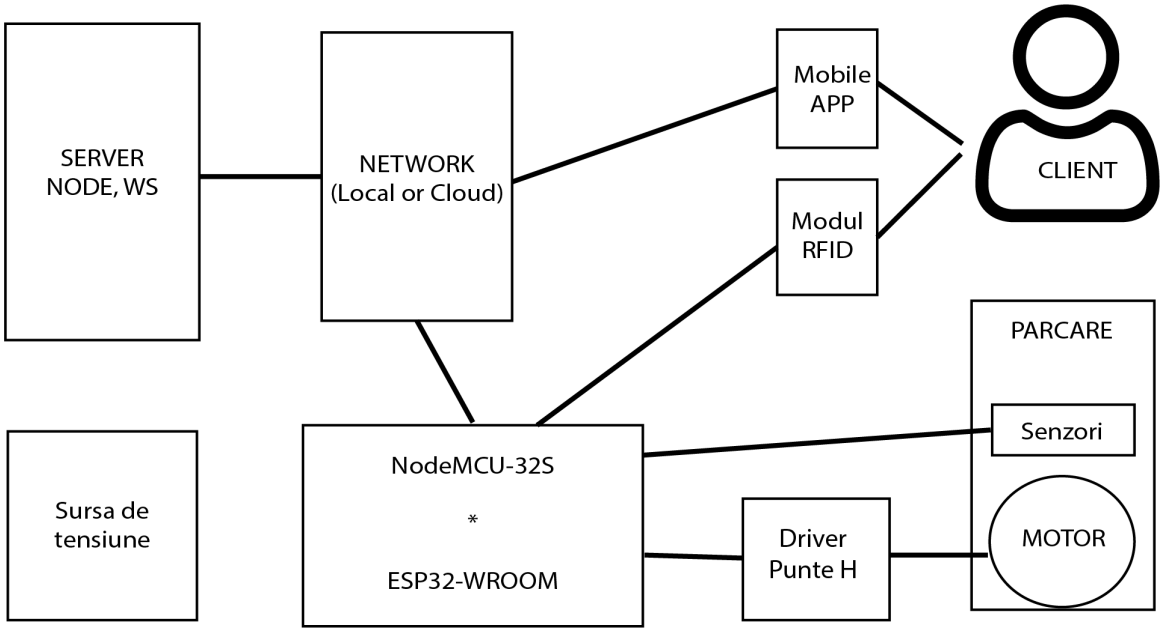
Deși nu sunt prezenți în machetă, s-au luat în considerare în program senzori de prezență umană, astfel încât mișcarea parcării să nu înceapă decât în absența persoanelor din aria de pericol.

Server-ul va putea accesa oricând informațiile despre parcare și starea acesteia. De asemenea server-ul va putea seta diferiți parametri de funcționare.

- Modulul de comandă se va conecta la rețea prin Wi-Fi.

- Alimentarea modulului de comandă se poate face atât la 6 - 12V, prin portul _VIN_ al microcontroller-ului cât și la 3.3V, cu alimentare directă.

- Alimentarea motorului se va face separat, la 6-9 V, și va fi comandată printr-un driver cu punte H.



Materiale utilizate:

- Placă de dezvoltare NodeMCU-32S bazată pe ESP32-WROOM

- Modul RFID-RC522

- Driver de motoare cu punte H L298N

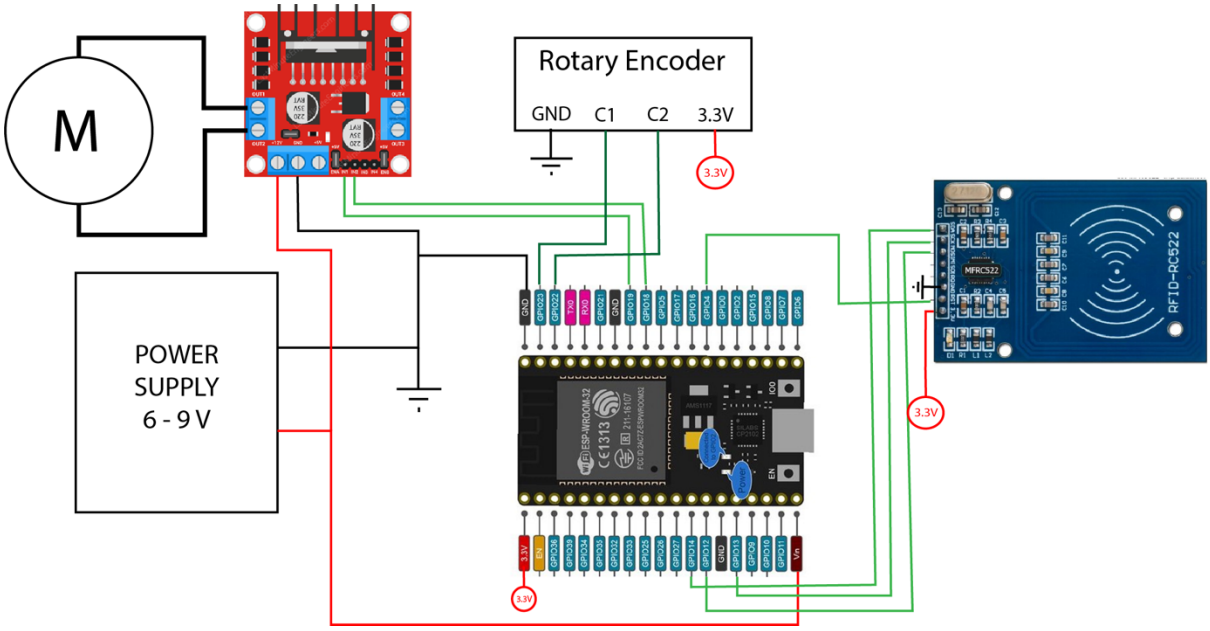
- Sursă de alimentare 6-9 V (2 baterii Li-Ion de 3.7V)

- DC motor cu un reductor amplasat pe acesta

- Encoder magnetic pentru a afla numărul de rotații ale motorului

Partea electronică

Pentru partea electrică am utilizat elementele prezentate în schema de mai jos:



- Am folosit microcontroller-ul ESP32, deoarece aveam nevoie de o eficiență sporită a programului, dar și datorită capabilității sale de conectare prin Wi-Fi. Puteam folosi și un Raspberry Pi, însă acesta era mult mai încet, de aceea ne-am rezumat la placa NodeMCU-32S.

- Pentru a permite utilizarea cardurilor RFID, am avut nevoie de un modul RFID-RC522. Acesta a putut fi implementat cu ușurință, adăugând astfel o utilitate în plus proiectului.

- Pentru punerea în mișcare a parcării, am folosit un motor DC și un driver de motor cu punte H: L298N, deoarece doream să controlăm cât de rapid și cât de mult să se învârtă acesta.

- Sursa de alimentare este compusă din 2 baterii Li-Ion de 3.7 V, care împreună ne oferă 6-9 V, putere suficientă pentru alimentarea proiectului.

În continuare, luăm în considerare și server-ul cu care microcontroller-ul comunică, acesta fiind conectat prin Wi-Fi.

Folosim encoder-ul magnetic pentru a determina cât de mult s-a învârtit motorul, astfel, putem calcula distanța dintre locurile de parcare și să îi spunem motorului, precis, cât de tare să se învârtă, pentru prezentarea următorului loc de parcare.

Parcarea noastră este complet autonomă și nu necesită decât acționarea acesteia prin aplicație sau prin apropierea unui card RFID.

Avantaje

Conceptul prezentat de noi oferă o mulțime de avantaje, printre care se numără:

1. **Prețul redus al unui loc de parcare**. În mod normal, costul unui loc de parcare în subteran începe de la 23.000€, însă un spațiu existent în parcarea noastră costă doar 15.000-18.000€, un preț mult mai avantajos.

2. **Confortul vizual și integrarea aproape inobservabilă a parcării**. Datorită conceptului de parcare de tip carusel, parcarea noastră se comportă ca un lift subteran, partea supraterană fiind mult mai redusă ca dimensiune decât metodele existente, astfel, aceasta putând fi integrată oriunde, în orice stil arhitectural.

3. **Conceptul dezvoltat în prezent poate fi integrat pentru mai multe pracări**. Cu siguranță vor exista mai multe parcări de acest gen, de aceea noi am scris codul în așa fel încât există posibilitatea din aplicația web să se aleagă parcarea unde dorește utilizatorul să parcheze, astfel, proiectul este pregătit și pentru un viitor pas înainte.

4. **Eficiența din punct de vedere volumetric**. Spațiul necesar pentru construirea unui loc de parcare de tip S-Park ocupă jumătate din spațiul necesar unui loc obișnuit, deoarece, în cazul proiectului nostru, nu este deloc necesară proiectarea unui spațiu de manevră pentru autovehicule, acestea fiind ușor fixate pe locul de parcare.

5. **Ideea noastră nu se rezumă doar la utilizarea publică**. Parcarea noastră este o soluție foarte bună și pentru locurile de parcare din zonele cu blocuri sau case. În zonele rezidențiale, locatarii își pot adăposti autovehiculul în interiorul parcării private, acesta fiind complet izolat de factorii externi, de pericole precum hoți, fenomene naturale: zăpadă, iarna; căldură exagerată, vara etc. Datorită construcției sub pământ, parcarea noastră este capabilă să suporte toate aceste primejdii, confortul fiind mult îmbunătățit.

Construcția codului - Software

Limbajele de programare folosite au fost: C standard (utilizat pentru controlul părții fizice a robotului), Javascript (pentru back-end și front-end (script-uri pentru trimiterea informațiilor către server))

Pentru partea de server, s-a folosit Node.js, cu modulele WS.

Pentru C am folosit modulele: *espressif-SDK*, *FreeRTOS* și biblioteca *"mfrc522.h"*.

Codul pe care l-am construit este împărțit în 3 secțiuni:

1. **Secțiunea embedded** (cea care se ocupă de controlarea părții fizice)

2. **Secțiunea back-end** (cea care se ocupă de tot ceea ce înseamnă comunicarea cu server-ul și implicarea lui în procesele noastre)

3. **Secțiunea front-end** (cea care se ocupă de website-ul parcării: *academia.go.ro:8075*

La pornirea proiectului, se urmăresc următorii pași:

1. Inițializarea modulului, având următoarele componente:

- Inițializarea pinilor

- Stabilirea conexiunii Wi-Fi

- Conectarea la server prin Web Sockets și înregistrarea parcării pe server prin informațiile acesteia (id, număr de locuri de parcare)

2. Comunicarea cu server-ul, se face bidirecțional, prin Web Sockets, prin tag-uri specifice fiecărei instrucțiuni. Pentru cercetare-dezvoltare, server-ul are control complet asupra funcționalității modulului de comandă, acesta prezentând api – endpoints și pentru funcții specifice, cum ar fi mișcarea motoarelor cu o anumită viteză sau la o anumită poziție. Server-ul are și rol în setarea parametrilor interni, cum ar fi distanța între locurile de parcare etc. Modulul de comandă va iniția comunicarea, în cazul prezenței unui nou card RFID, sau în cazul unei erori. Comenzile de la server la modul sunt următoarele:

- `_spot x` - se cere prezentarea locului de parcare x. *În cazul în care există deja o cerere pentru această acțiune, aceasta se va adăuga în coada de cereri și va fi efectuată ulterior.*

- `_speed x` - motorul va începe să se învârtă cu viteza x

- `_pos x` - cere deplasarea motorului la poziția x (poziția reprezentând contorul incrementat de encoder). *În cazul în care există deja o cerere pentru această acțiune, aceasta se va adăuga în coada de cereri și va fi efectuată ulterior.*

- _get_status - parcarea va furniza date despre starea acesteia (starea de mișcare, prezența persoanelor, locul actual de parcare prezentat etc.)

- _get_info - parcarea va furniza informații generale despre aceasta (id, număr de locuri de parcare, distanța dintre locuri etc.)

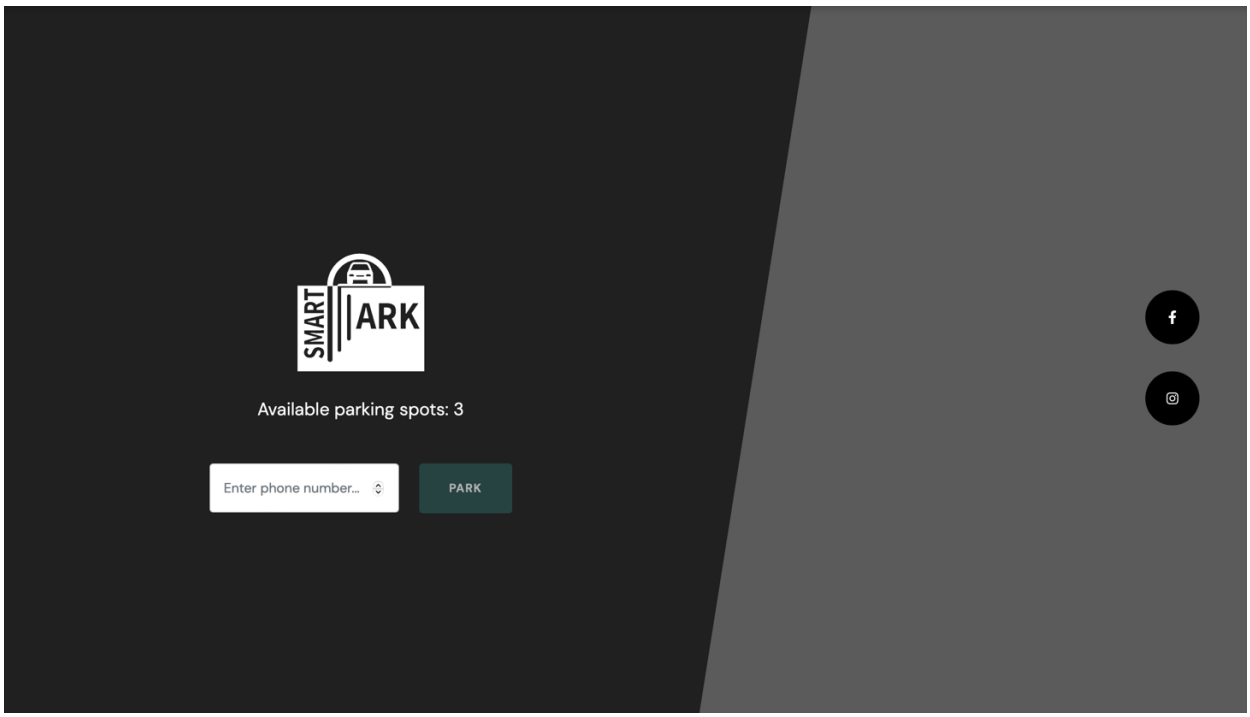
- _set_dist x, _set_offset x, _set_crt x setează parametrii interni ai modulului.

3. Al doilea task este cel de **monitorizare RFID**: Monitorizarea prezenței cardurilor RFID se face prin pooling (se verifică integritatea cardului, pe server). O dată pe secundă se verifică prezența unui nou card. În cazul în care a fost detectat un astfel de card, se va trimite un mesaj adecvat către server pentru validarea ID-ului.

4. Al treilea task se ocupă de controlul mișcării. Acesta va monitoriza cozile de comandă: coada de poziții (pentru depanare și cercetare) și coada principală, în care sunt memorate locurile de parcare dorite pentru a fi prezentate. Cât timp cozile nu sunt goale, se va lua următoarea cererea și se a acționa motorul, prin Pulse Width Modulation, pentru rezolvarea acestora.

5. Citirea endoder-ului. Pentru citirea encoder-ului rotativ, s-a atașat o funcție de întrerupere pe pinii conectați la acesta care, la schimbarea valorilor, verifică direcția de deplasare și crește sau scade un contor global.

Website-ul comunică tot prin server cu celelalte module prezente. Pe pagina acestuia se poate insera un număr de telefon (care este verificat să aibă minim 10 caractere) și, în viitor, se dorește implementarea unui drop-down list cu id-urile parcărilor, pentru posibilitatea selectării unei parcări dorite, din listă. Acesta permite și vizualizarea locurilor de parcare libere, pentru eficientizarea timpului de așteptare.



Bibliografie:

Espressif-SDK: *https://www.espressif.com/en/products/software/esp-sdk/overview*

FreeRTOS: *https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/freertos.html*

MFRC522.h: *https://github.com/miguelbalboa/rfid*