Apostol Rares

331CA

RC Car with camera

Introducere

Proiectul meu consta in construirea unei masini cu telecomanda controlata de la distanta prin intermediul unei aplicatii Android, folosind Wi-Fi.

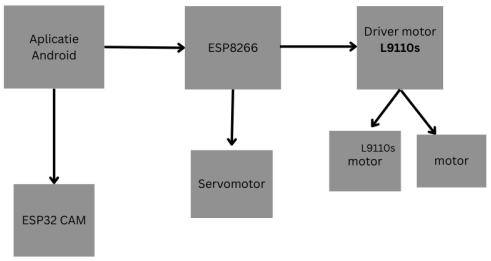
Masina va putea fi controlata în timp real din aplicatie, permitand deplasarea inainte și inapoi, precum și virarea stanga-dreapta printr-un volan implementat in aplicatie. Desi ar fi fost mai usor ca masina sa vireze doar din niste sageti, am dorit un control precis, bazat pe volan.

Un alt aspect important al proiectului este integrarea unei camere video montate pe masina. Aceasta va transmite video către aplicatia Android. Utilizatorul va putea astfel controla masina chiar si in afara campului vizual. Am folosit un ESP32 CAM pentru camera.

Comunicarea dintre aplicatie si masina se va face printr-o retea Wi-Fi locala. Pentru asta, un ESP8266 va fi pornit in mod access-point, iar celelalte dispozitive se vor conecta la aceasta retea.

Descriere generală

Un ESP8266 va fi pornit in mod AP cu o adresa IP statica, iar dispozitivul Android si ESP32-CAM(cel din urma tot cu o adresa IP statica, pentru usurinta) se vor conecta la aceasta retea. Telefonul va trimite date de control catre ESP8266, care va actiona asupra motoarelor. De asemenea, telefonul va accesa si pagina web pe care transmite ESP32-ul, si o va afisa in aplicatie, printr-un WebView.



Hardware Design

Piese:

- ESP32-CAM
- ESP8266
- Driver motor L9110s
- Modul USB to serial HT42B534 (pentru programarea ESP-ului)
- 2 x Suport baterii 4 x R6
- Micro Servomotor SG90 180°
- 2 x Motor DC 3V-6V cu reductor 1:48
- Masina lego
- Intrerupator

Schema electrica:

Utilizare PINI :

- ESP8266
  - **D7, D6** - pini PWM pentru controlul unui motor
  - **D2, D1** - pini PWM pentru controlul celui de-al doilea motor
  - **D0** - pini PWM pentru controlul servomotorului
  - **Vin** - alimentare dintr-un set de baterii dedicat pentru microcontrolere.
  - **Gnd** - conectat la un gnd comun pentru toate dispozitivele, esential pentru PWM.
- ESP32-CAM
  - Am vrut ca acest microcontroller sa se ocupe numai de camera, deci singurii pini utilizati sunt Vin si GND.
  - Pentru programarea acestui microcontroller, trebuie folosit convertorul HT42B534: se conecteaza RX de pe conector la U0T, TX la U0R, pinul IO0 trebuie conectat la ground, si se alimenteaza prin

convertor,prin USB-C. Dupa incarcarea codului, se scoate alimentarea, si firul de la IO0 la ground.
- L9110s - driver pentru motoare
  - pentru ca un microcontroller nu poate furniza destul curent pentru doua motoare, a fost necesara utilizarea unui driver de motoare.
  - **M1(A), M1(B)** - pini pentru puterea si sensul unui motor
  - **M2(A), M2(B)** - pini pentru puterea si sensul celuilalt motor
  - **Vin** - alimentat din baterie de 6V.
  - **Gnd** - comun cu celelalte componente
- Servomotor SG90
  - **S** - pin pentru semnalul PWM
  - **Vin** - alimentat la 6V
  - **Gnd** - comun cu celelalte componente

Software Design

Flow program:

- Se deschide ESP8266 in mod AP, cu o adresa IP statica.
- Aplicatia Android va trimite comenzi prin UDP la ESP8266, pe portul 8888. Pentru a nu trimite mesaje degeaba, se trimit doar atunci cand se schimba o valoare(de ex se apasa pe acceleratie)
- ESP8266 proceseaza comenzile, si controleaza servo-motorul si cele doua motoare. Pentru a primi mesajele, am folosit AsyncUDP, ca sa nu se blocheze cat timp proceseaza pachetele.
- ESP32 se va conecta la reteaua ESP8266, si va fi accesata camera prin intermediul aplicatiei de pe telefon.

Program ESP8266:

- Am folosit VsCode, PlatformIO.
- ESPAsyncUDP.h, ESP8266WiFi.h, Servo.h

Program ESP32:

- Am folosit tot VsCode si PlatformIO, dar am utilizat un exemplu de cod din acest repo

Program Android:

- Android Studio, Kotlin si Jetpack Compose

Implementarea Software in detaliu:

1. **Aplicatia Android**

- RCCarApp() - este o functie composable, care contine elementele care se afiseaza pe ecran: cele doua pedale, un slider din care se seteaza puterea motoarelor, si volanul. Contine cateva variabile care se schimba atunci cand utilizatorul interactioneaza cu componentele. Am folosit doua functii **LaunchedEffect** , care urmaresc schimbarile variabilelor si la modificari, lanseaza corutine, pentru trimiterea mesajelor si rotirea volanului.
- **suspend fun sendCommands()** - pornita din LaunchedEffect, primeste ca parametri variabilele,

si trimite un pachet UDP catre ESP8266, cu toate datele necesare pentru control: unghiul volanului, puterea, ce pedala este apasata.
- Pentru afisarea camerei, am utilizat un WebView, care acceseaza pagina creata de ESP32.

2. **Cod ESP8266**

- Se porneste in mod AP, cu o adresa IP statica, folosind WiFi.config()
- Pentru procesarea pachetelor, am folosit o functie de callback, **udp.onPacket** , care se executa automat atunci cand este primit un pachet UDP. Aici, se transforma pachetul in string, si cu ajutorul functiei **parse_message()** se modifica niste variabile globale, folosite in loop(). Am ales sa folosesc AsyncUdp pentru a nu bloca executia in loop, asteptand un mesaj. Astfel, codul din onPacket se executa doar atunci cand este primit un pachet nou
- In **loop()** , sunt controlate motoarele, in functie de variabilele globale. Se roteste servomotorul(am dat un delay de 20 de ms pentru a avea suficient timp), si sunt controlate puterea si sensul motoarelor din fata: pentru a merge inainte, pinii A trimit semnalul PWM, si pinii B sunt setati pe LOW. Pentru a da cu spatele, invers: pinii A sunt setati pe LOW, pinii B trimit PWM. Daca nicio pedala nu e apasata, atunci toti pinii sunt setati pe LOW.

3. **Cod ESP32 CAM**

Am folosit codul de aici

Am modificat cateva lucruri: esp32 va primi o adresa statica, va folosi grayscale(jpeg nu e suportat de camera). De asemenea, rezolutia este foarte mica(QQVGA), camera merge destul de prost si am incercat sa fie totusi functionala, chiar daca calitatea video nu este excelenta

Concluzii

Mi s-a parut mult de munca, mai ales la partea de hardware, unde am avut probleme cu unele piese. Sistemul de servo a luat destul de mult timp, am incercat sa il fac cat mai precis. Am incercat sa scriu singur codul pentru camera, dar nu am gasit niciun fel de documentatie pentru rhyx-m21-45, si am folosit un exemplu din arduino.

Cu toate astea, proiectul mi-a placut, mai ales cand la final am vazut ca masina chiar merge, si o pot controla din telefon, cum mi-am dorit.

Download

Github Android

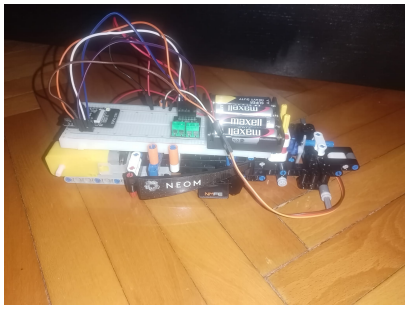
Github esp8266

Github esp32 cam

Jurnal

Hardware-ul este aproape gata: mai trebuie conectate firele de la driver la motoare, dar vreau sa le lipesc. Am testat sistemul de servo directie. Pentru ca esp-ul nu are usb, a trebuit sa folosesc un convertor, si a durat ceva pana sa ma prind cum sa programez microcontrollerul. Din cauza asta, o sa astept pana cand codul este gata ca sa lipesc firele si sa fac designul final.

Camera trebuia sa fie OV2640 dar a venit altceva Camera primita e rhyx-m21-45, care merge foarte prost.



Video servodirectie aici

Update 1: cumva am stricat ESP32-ul, nu mai merge:). Aveam acasa un esp8266, o sa continui proiectul cu el, dar nu are camera. Am mai adaugat in plus in software: volan(se poate controla precis servomotorul), pedale de acceleratie si frana, si un slider de unde se alege viteza. Nu am pus inca rotile, dar toate piesele functioneaza.

Control motor

Ambele motoare

Control servo

Update 2: am reusit sa pornesc ESP32-ul. Pentru ca am scris deja codul pentru controlul motoarelor pe esp8266, o sa continui cu ambele: esp32 pentru camera, esp8266 pentru motoare.

Proiect complet :

1. Masina gata
2. Aplicatia

Bibliografie si resurse

ESP32 CAM

Export to PDF