OmniCam

Introducere

Esti la job și te întrebi ce fac animalele tale de companie singure acasă? Ai auzit un sunet straniu noaptea și îți e frică sa ieși de sub pătură să verifici? Sau poate vrei o bere rece din frigider și îți e lene sa te ridici de pe canapea? Toate probleme tale sunt de acum rezolvate cu **OmniCam**.

Proiectul OmniCam constă în construirea unei mașini de mici dimensiuni, controlată prin Bluetooth/Wi-Fi, utilizând un ESP32-CAM, care permite folosirea unei camere, pentru a transmite unui feed video. În plus, se poate realiza și capturarea de imagini la calitate HD și salvarea acestora pe un card SD.

Mașinuța se va deplasa omnidirecțional cu ajutorul a patru roți mecanum.

Pe un LCD se va afisa adresa IP a placuței (pentru conectarea la mașină) sau alte mesaje (dacă vrei să îi spui colegului din bucătărie să pună berea pe masinuță).

Ideea proiectului a pornit de la dorința de a avea o mașină de mici dimensiuni, pe care să pot să o controlez de la distanță și să îmi permită să monitorizez mediul înconjurător. În plus, folosirea roților de tip mecanum a fost influentață de propria fascinație în legatură cu modul lor de funcționare și pentru a oferi un control sporit al mașinii.

Consider că proiectul este util pentru că oferă o soluție accesibilă și ușor de utilizat pentru controlul și monitorizarea la distanță a diferitelor obiective. Ca exemple fiind cele menționate mai sus. Mai mult, dacă se adaugă un led infraroșu se poate obține un feed video și noaptea; iar cu dimensiunile compacte care permit accesul în zone greu accesibile, OmniCam ar putea fi folosită pentru a căuta fantome sau chiar spionaj.

Descriere generală

Masinuța va fi controlată prin Wi-Fi sau Bluetooth, utilizând ESP32-CAM. Acesta va gestiona și o cameră pentru transmiterea unui feed video, pe un PC sau telefon (dispozitivul care controlează și mașina), sau pentru a captura imagini la rezoluție HD, care vor fi salvate pe un card SD. Se poate opta folosirea cititorul de card SD integrat în ESP32-CAM sau un adaptor separat de card MicroSD. Atât camera, cât și cititorul de carduri SD operează prin interfața SPI.

În lipsa unui multiplexor, voi folosi un Arduino Uno, care va primi comenzi prin UART de la ESP32-CAM pentru controlul restul dispozitivelor. Arduino Uno va opera patru motoare TT, folosind două module driver L298N. În plus, doresc să implementez un display LCD1602, conectat prin I2C la Arduino, pentru a afișa adresa IP a placutei ESP32-CAM (pentru conectarea prin Wi-FI) sau mesaje în funcție de

butoanele apăsate pe dispozitivul de control.

Hardware Design

⇒ *Listă componente*

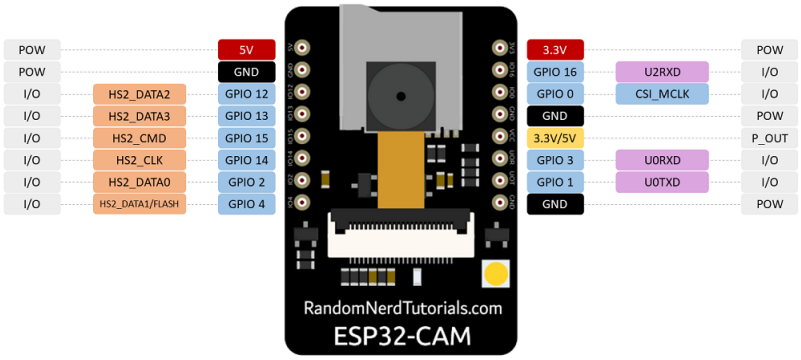
1. 1 ESP32-CAM cu modul de camera si modul SD card
2. 1 Arduino UNO
3. 2 Drivere L298N
4. 1 LCD1602 cu I2C
5. 4 Motoare TT DC, fiecare cu câte o roată mecanum
6. Breadboard power supply
7. 3 Rezistente 10k ohm
8. Breadboard
9. Breadboard power supply
10. Surse alimentare
11. Șasiu mașina

⇒ *Schema electrică*

Schema electrica Omnicam

Întrucât ESP32 are un consum mare, cel mai probabil voi folosi o sursă suplimentară sau bateriile pentru alimentarea Arduino-ului (cu LCD-ul). Breadboard power supply va fi alimentat cu o baterie de 9V.

⇒*Explicații pentru utilizarea pinilor programabili*

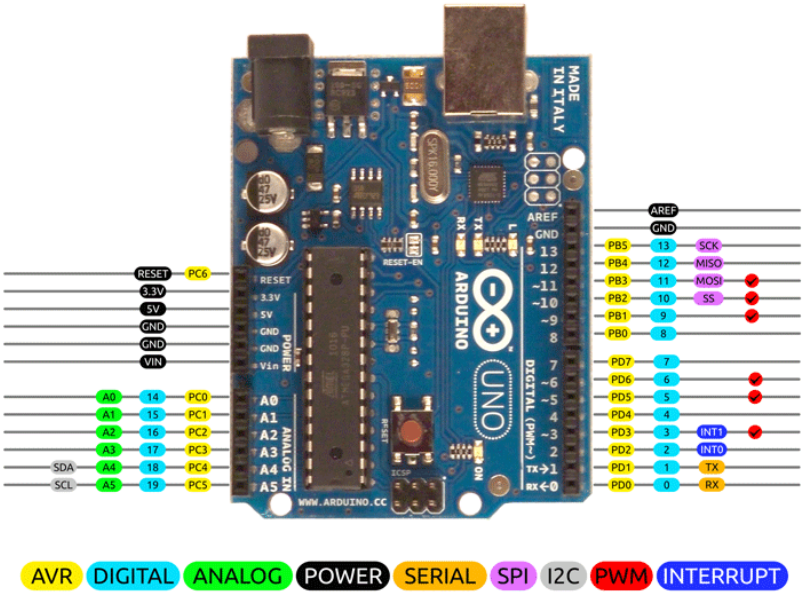


ESP32-CAM:

- GPIO 1 (RX) și GPIO 3 (TX) pentru comunicarea prin UART cu placa Arduino Uno
- Restul de pini sunt folosiți pentru cameră și SD reader

Dacă se adaugă un led infraroșu, camera poate fi folosită și pe întuneric.

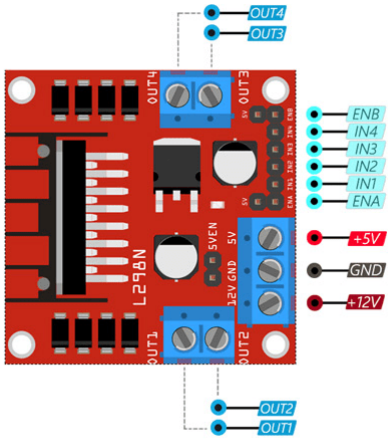
Din cauza faptului că cipul de pe Arduino funcționează la 5V, iar cel de pe ESP32 funcționează la 3.3V, este nevoie sa folosesc un divizor de tensiune pentru comunicarea prin UART (unul pentru trimitere și încă unul pentru primire)!



Arduino Uno:

- PD 5 și PD 6 - pentru controlul PWM a primului driver (roțile din față)
- PD 2, 3, 4, 7 - pentru input-ul primului driver și controlul direcției motoarelor din față
- PD 9 și 10 - pentru controlul PWM a primului driver (roțile din spate)
- PD 8, 11, 12, 13 - pentru input-ul primului driver și controlul direcției motoarelor din spate
- A4 (SDA) și A5 (SCL) - sunt folosiți pentru funcționalitatea ecranului
- A0 (RX) și A1 (TX) - comunicare UART cu ESP32-CAM

Pinii P0 și P1 care sunt prioritari RX și TX nu pot fi folositi pentru comunicarea cu ESP32-ul, deoarece sunt folositi pentru comunicarea cu laptopul și programarea placii. Astfel, este nevoie să folosesc A0 și A1 pentru UART.



Driver L298N

- IN1 & IN2 - Pini de intrare a motorului A. Folosit pentru a controla direcția de rotație a motorului A
- IN3 & IN4 - Pini de intrare a motorului B. Folosit pentru a controla direcția de rotație a motorului B
- ENA - Activează semnalul PWM pentru motorul A
- ENB - Activează semnalul PWM pentru motorul B
- OUT1 & OUT2 - Pinii de ieșire ai motorului A
- OUT3 & OUT4 - Pinii de ieșire ai motorului B

⇒ ***Funcționalitatea componentelor***

- Comunicarea prin UART

În acest video se poate observa comunicarea ESP32-CAM către Arduino UNO prin UART. ESP trimite (la 1.5s) un semnal către Arduino, iar Arduino îl trimite înapoi. Dacă mesajul primit este același ca cel trimis, ledul de pe ESP se aprinde. Arduino afișează mesajul primit pe al doilea rând al ecranului, iar pe primul rând timpul in milisecunde.

Software Design

Datorită faptului că am folosit două placi de microcontrollere, a fost nevoie sa realizez două coduri separate. Mediul de dezvoltare pe care l-am folosit este Arduino IDE.

Pentru codul de Arduino Uno am folosit librariile:

1. <LiquidCrystal_PCF8574.h> pentru controlul ecranului LCD prin intermediul modulului I2C
2. <Wire.h> este utilizată pentru comunicația I2C (pentru interacțiunea cu modulul LCD)
3. <SoftwareSerial.h> pentru crearea unui port serial (virtual) suplimentar pe Arduino folosind pinii analogici, ceea ce este util pentru comunicarea cu ESP32-CAM, întrucât nu pot folosi pinii RX și TX de pe plăcuță (sunt folosiți pentru programare si debug)

Implementarea codului Arduino Uno:

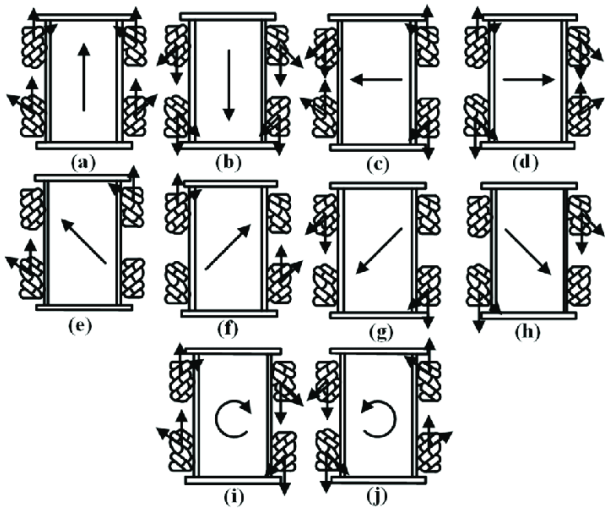
1. Inițializare: Se realizează în cadrul funcției setup() și reprezintă configurarea pinilor pentru motoare,

inițializarea LCD-ului și a comunicării seriale.
2. Comunicare Serială: Monitorizarea și gestionarea mesajelor primite de la un dispozitiv ESP32 prin intermediul unui port serial software. În cadrul funcției loop(), care rulează în mod continuu, se verifică dacă sunt disponibile mesaje pe portul serial și gestionează comenzile primite. Funcția handleCommand este folosită pentru controlul motoarelor sau afișarea mesajelor pe LCD. Comunicarea cu ESP-ul se realizeaza la rata de 115200.
3. Controlul Motoarelor: Implementarea funcțiilor care controlează direcția și viteza motoarelor în funcție de comenzile primite. Pentru fiecare tip de direcție de deplasare există o funcție specifică (cu nume sugestiv).
4. Afișare Mesaje pe LCD: În cadrul funcției handleCommand() pe se gestioneaza afișarea comenzilor sau a adresei IP a placuței ESP32-CAM.

Din moment ce motoarele un au fost de cea mia buna calitate, a fost nevoie sa le calibrez cumva, astfel încât sa aiba toate aceeași viteză. Dacă motoarele nu au aceeași viteza, mașinuța se va deplasa în direcții necorespunzătoare.

Cum se poate observa roata din stânga se deplasează puțin mai repede decat cea din dreapta. Astfel, forțele necesare pentru deplasare nu sunt egale, iar mașina nu se v-a mișca pe direcția dorită.

În plus, a fost nevoie sa setez controlul direcției fiecărei roți pe baza deplasării dorite în cadrul funcțiilor după următoarea schemă.



Pentru realizarea codului Arduino am folosit cunoștințele acumulate pentru UART, PWM și I2C.

Din cauza conexiunii slabe a cablurilor sau din cauza bibliotecii de serializare pe Arduino, la comunicarea serială apar mesaje corupte la Arduino. Pentru a rezolva această problemă, am realizat un sistem de acknowledgement pentru verificarea comenzilor. Astfel, dacă Arduino-ul primeste un mesaj corupt, aceasta va trimite un mesaj NACK, astfel încât ESP-ul să trimită ultima comandă pană ajunge un mesaj corect.

Pentru codul de ESP32-CAM am folosit codul exemplu de testare a modulului de cameră. Acest cod realizeaza conexiunea la o adresa WIFI și trimite un feed video. În plus eu am adaugat butoanele pentru controlul direcției.

O scurtă descriere a fiecărui fișier folosit:

- camera_index.h: conține (în format hexa) pagina web pe care utilizatorul o accesează. Pentru șirul generarea sirului hexa din html m-am folosit de site-ul: https://gchq.github.io/CyberChef/

configurația: https://gchq.github.io/CyberChef/#recipe=Gzip('Dynamic%20Huffman%20Coding','index.html.gz','',false)To_Hex('0x',0)Split('0x',',0x')&oeol=FF

- camera_pins.h: aici sunt definiți de pe placuță. Placuța pe care am folosit-o eu este de tipul: CAMERA_MODEL_AI_THINKER
- CameraWebServer.ino: utilizeaza biblioteca <esp_camera> pentru inițializarea ESP-ului. Pentru stabilirea conexiunii Wi-Fi și pornirea unui web server se folosește de biblioteca <WiFi.h>. După ce ESP-ul reușește să se conecteze la WIFI, aceasta va trimite pe serial un mesaj cu adresa paginii pentru control, pentru ca Arduino-ul să o afișeze pe LCD.
- app_httpd.cpp: acest fișier conține implementarea serverului HTTP si rezolva cererile primite de la client, pentru controlul camerei sau a direcției. Acest cod oferă funcționalități precum streaming video, caputra de imagine, schimbarea rezoluției sau a altor aspecte ale camerei.

Pentru folosirea și modificarea codului ESP32-CAM am folosit cunoștințele acumulate pentru UART, SPI și WIFI.

Rezultate Obţinute

Interfața web:

Concluzii

Realizarea proiectului Omnicam a fost o experiență interesantă, care mi-a oferit o perspectivă aplicată asupra tehnologiilor complexe de robotică.

Am învățat să integrez hardware-ul cu software-ul într-un mod eficient, coordonând microcontrolerele și sistemele de control.

Complexitatea proiectului s-a reflectat în provocările de sincronizare a componentelor mecanice, cât și a celor software, de exemplu a conexiunii la WIFI, a comunicarii seriale, a controlului camerei.

În plus, am învățat să sudez componente: cablurile de la motoare și suportul de baterii, rezistorul de pe ESP pentru antena.

Download

Codul și schema proiectului se află aici: omnicam_proiect_pm_sirbu_vlad-stefan.zip

Bibliografie/Resurse

Resurse hardware

- ESP32-CAM:

- https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/DFR0602_Web.pdf
- https://docs.ai-thinker.com/_media/esp32/docs/esp32_cam_sch.pdf

- Arduino:

- https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf

- L298N:

- https://components101.com/modules/l293n-motor-driver-module
- https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf

- LCD 1602 cu modul I2C:

- https://www.handsontec.com/dataspecs/module/I2C_1602_LCD.pdf

Resurse software

- ESP32-CAM:

- https://randomnerdtutorials.com/esp32-cam-video-streaming-face-recognition-arduino-ide/
- https://www.diyengineers.com/2023/04/13/esp32-cam-complete-guide/
- https://www.youtube.com/watch?v=bIJoVyjTf7g

- Comunicare seriala între Arduino si ESP32-CAM:

- https://www.programmingboss.com/2023/01/serial-communication-between-arduino-and-esp32-CAM-UART-data-communication.html#gsc.tab=0

- L298N:

- https://projecthub.arduino.cc/lakshyajhalani56/l298n-motor-driver-arduino-motors-motor-driver-l298n-7e1b3b

- LCD 1602 cu modul I2C:

- https://www.handsontec.com/dataspecs/module/I2C_1602_LCD.pdf

Export to PDF