Bike Infotainment System

README

Bike Infotainment System

Introducere

Autor: **Robert Grancsa 332CA**

Un infotainment pentru biciclete, vrea să afișeze pe un display informații folositoare despre statusul curent al cursei tale precum:

- Viteza curentă
- Temperatura de afară
- Viteza medie din cursa curentă
- Locația curentă
- Status despre ce obiecte sunt pe traseu, folosind un model de image recognition, care poate identifica semne de circulație.
- Înclinația și acceleratia curentă

Pe lângă asta, dispozitivul este conectat la internet, și comunică mereu cu un server pe un Raspberry Pi, unde stochează informațiile, care mai apoi sunt salvate într-o bază de date și afișate într-un dashboard folositor, care îți arată detalii despre ultimele tale curse, cu un istoric cu unde ai fost și ce locuri ai vizitat.

Acest dispozitiv de infotainment pentru biciclete aduce o serie de beneficii semnificative pentru cicliști, oferindu-le uneltele necesare pentru a monitoriza și îmbunătăți experiența lor de ciclism.

Descriere generală

Schema proiectului este compusă dintr-un microcontroller ESP32, care conectează partea fizică a proiectului. Conectat la el, putem găsi:

1. GPS - ne va întoarce date despre locația curentă printr-un stream pe UART, aceste date fiind stocate și trimise mai departe la server o dată la câteva secunde.
2. Accelerometru - ne poate afișa starea curentă a bicicletei, dacă se apleacă stânga sau dreapta, sau cât de repede accelerează sau frânează.
3. Hall Effect senzor - folosit pentru a calcula numărul de revoluții pe secundă al bicicletei, pentru a afișa o estimare corectă a vitezei.
4. Camera - va transmite cadre o dată pe secundă către server, care mai apoi va întoarce date despre cadrul curent, mai apoi fiind afișat pe display.
5. Display - aici vor fi afișate toate datele importante, care nu sunt transmise către server dar care ajută la a vedea mai ușor starea curentă.
6. Serverul Raspberry Pi - face conexiunea cu toate instrumentele de la bord, care le stochează pe o

bază de date locală, și după le putem interoga pe un front-end cu un dashboard, care ne arată statistici despre ultimele curse.

Hardware Design

Schematicul produsul se poate vedea mai jos. Am incercat sa conectez pinii spefici SPI pe pinii specifici de SPI, pentru GPS m-am folosit de pinii de UART si pentru camera am ales sa folosesc pinii cei mai apropiati unul de altul pentru date, ca sa nu fie overlap la conectarea firelor.

Tabelul cu piesele folosite se poate vedea mai jos.

Nume Piesa	Model	Protocol
Microcontroller	ESP32-WROOM32	-
Display	ST7735	SPI
Camera	OV7670	I2S
GPS	GY-GPS6MV2	UART
Hall Effect Sensor	YS-27	ADC

Software Design

Codul este momentan functional pe partea de cod main, functionalitatile extra nu au putut fi duse pana la capat din cauza limitarilor hardware (ram insuficient pentru a encoda poze in base64), dar cele de baza cum ar fi tracking de viteza functioneaza corect.

Biblioteci

Au fost folosite biblioteci precum:

- Adafruit_GFX - pentru display si grafice
- OV7670 - pentru camera si sincronizare de ceas pe I2C/I2S
- TinyGPS++ - pentru a citi datele pe UART pe GPS
- WiFi, WiFiMulti, WiFiClient, HTTPClient - pentru transfer de date via internet

Features

Au fost folosite doua thread-uri separate, unul care se ocupata de citirea datelor, actualizarea lor si afisarea pe ecran de fiecare data cand se gaseste o actualizare a vitezea via Hall Effect sensor, iar al doilea thread, care e pinned la un alt core fata de primul, se ocupa de transmisia prin wifi a datelor catre back-end-ul care tine un loc al datelor.

Am folosit si intreruperi pentru a detecta butonul apasat, ca mai apoi sa pot cicla intre ecranul principal de viteza, si ecranul care arata display-ul.

Serverul este scris in Express, si se foloseste de MongoDB sa salveze datele persistent.
El este hostat pe raspberry pi, si pastreaza mereu legatura la device. Acest back-end face si request-uri catre Google Cloud sa clasifice obiectele din imagine de pe camera,
care mai apoi intoarce daca a gasit vreun semn de circulatie sau vreun danger pe imagine, si in caz

ca se gaseste ceva dangerous imaginea respectiva este si ea salvata in baza de date ca logging.

Aceste date pot fi vazute dupa vazute dupa aceea pe front-end-ul scris in React.

Schelet

Codul contine partea de init, care initializeaza wifi-ul, si fiecare senzor valabil precum gps-ul, camera si ecranul. Apoi avem partea de cod care introduce intreruperile si pornirea de core-uri pentru task-ul de send, si cel de update a ecranului.

Se foloseste de multe functii ajutatoare, pentru a trimite datele, pentru a da refresh la ecran si pentru a calcula distanta parcursa de la ultima activare.

Calibrare

Pentru senzorul Hall, calibrarea a fost facuta folosind print-uri la serial si folosind un magnet si regland potentiometrul de pe device pentru a ajunge la sensitivitatea dorita.

Rezultate Obținute

Codul sursa pentru proiect poate fi gasit aici

Download

Proiectul poate fi gasit pe github, si descarcat si compilat local.