**CAR MONITOR**

**Candidat: Horvath Paul Șerban**

**Coordonator științific: Ș.l. Dr.Ing. Lucian Prodan**

Sesiunea: Iunie 2024

Contents

[REZUMAT 4](#_Toc164282000)

[Abstract 4](#_Toc164282001)

[1. INTRODUCERE 4](#_Toc164282002)

[1.1 Aspecte generale - PROBLEM STATEMENT, masina folosita, protocoale, abrevieri, specificatiile proiectului etc. 4](#_Toc164282003)

[1.2 SOTA - State of the art, studiu de piata 4](#_Toc164282004)

[1.3 ESP-32 4](#_Toc164282005)

[1.3.1 Modulul integrat Bluetooth 4](#_Toc164282006)

[1.3.2 Modulul integrat WiFi 4](#_Toc164282007)

[1.3.3 Dependinta SPIFFS 4](#_Toc164282008)

[1.3.4 Dependinta WebAsyncServer 4](#_Toc164282009)

[1.4 ELM327 OBDII - scurta descriere protocol OBDII 4](#_Toc164282010)

[1.5 Git repo - modul de gestionare 4](#_Toc164282011)

[1.6 ELMduino 4](#_Toc164282012)

[2. ARHITECTURA 4](#_Toc164282013)

[2.1 Diagrama de decizii generica 4](#_Toc164282014)

[2.2 Arhitectura Bluetooth 4](#_Toc164282015)

[2.2.1 Diagrama de decizii detaliata 4](#_Toc164282016)

[2.3 Arhitectura WebAsyncServer 4](#_Toc164282017)

[2.3.1 Diagrama de decizii detaliata 4](#_Toc164282018)

[3. IMPLEMENTARE DETALIATA 4](#_Toc164282019)

[3.1 Implementare Bluetooth 4](#_Toc164282020)

[3.1.1 Diagrama de secventa dinamica - Din avion 4](#_Toc164282021)

[3.1.2 Diagrama de decizii detaliata 4](#_Toc164282022)

[3.1.3 Diagrama de secventa pentru rpm() - Se adauga si detalii de returnari, etc. 4](#_Toc164282023)

[3.1.4 Diagrama de secventa pentru kph() - Se adauga si detalii de returnari, etc. 4](#_Toc164282024)

[3.1.5 Diagrama de secventa pentru fuel() - Se adauga si detalii de returnari, etc. 4](#_Toc164282025)

[3.1.6 Diagrama de secventa pentru oil() - Se adauga si detalii de returnari, etc. 4](#_Toc164282026)

[3.2 Implementare WebAsyncServer 5](#_Toc164282027)

[3.2.1 Diagrama de secventa dinamica - Din avion 5](#_Toc164282028)

[3.2.2 Diagrama de decizii detaliata 5](#_Toc164282029)

[3.2.3 Diagrama de secventa pentru WebServer.begin() 5](#_Toc164282030)

[3.2.4 Diagrama de secventa pentru send data 5](#_Toc164282031)

[3.2.5 Diagrama de secventa pentru SPIFFS open file 5](#_Toc164282032)

[3.2.6 Diagrama de secventa pentru grafice 5](#_Toc164282033)

[4. DESFASURARE EXEMPLIFICATA 5](#_Toc164282034)

[5. DEZVOLTARE SI TESTARE 5](#_Toc164282035)

[5.1 Detalii de configurare - configurare config.h 5](#_Toc164282036)

[5.2 Securitate 5](#_Toc164282037)

[5.3 Stocarea datelor 5](#_Toc164282038)

[5.4 Interfata cu utilizatorul 5](#_Toc164282039)

[5.5 Posibili clienti 5](#_Toc164282040)

[5.6 Performanta sistemului - cata memorie ocupa, frecventa de operare, de returnare a datelor etc. 5](#_Toc164282041)

[5.7 Scalabilitate tehnica - se pot adauga mai multe esp uri 5](#_Toc164282042)

[5.8 Scalabilitate antreprenoriala - posibil pret, specificiatii, pachet, venit anual etc (vezi GreenSpot) 5](#_Toc164282043)

[6. Concluzie 5](#_Toc164282044)

[6.1 Obiective indeplinite 5](#_Toc164282045)

[6.2 Comparare cu SOTA 5](#_Toc164282046)

[6.3 Adaugari ulterioare 5](#_Toc164282047)

[Referinte 5](#_Toc164282048)

# REZUMAT

Lucrarea de față are ca scop utilizarea unui sistem alcătuit dintr-o placă dedicată proiectelor de tip IoT (Internet of Things), numită ESP-32 furnizată de către compania Espressif, folosind framework-ul Espressif IoT Development Framework (esp-idf). Aceasta are rolul de a fi un nod care se ocupă de preluarea datelor în timp real de la un autoturism, aflat în staționare sau în mers, prin intermediul unui convertor OBD (On board diagnostics) versiunea II, dar și de găzduirea unui server web, cu scopul de a afișa parametrii vehiculului în timp real. Placa ESP-32 este programată într-o manieră în care să poată trimite cereri de citire al parametrilor (turația motorului, viteza instantă, consum, temperatura uleiului, consumul instant de carburant, temperatura lichidului de răcire și presiunea din galeria de admisie) către un convertor conectat la portul OBD-II al vehiculului.

Convertorul, sau cititorul OBD-II, este furnizat de compania VGate, și este echipat cu un chip de tip ELM327, care permite transferul datelor primite de la ECU (Electronic Control Unit) al autoturismului prin portul OBD-II către ESP-32 prin Bluetooth. A fost ales acest dispozitiv datorită versatilității și compatibilității crescute cu alte dispozitive. Acest dispozitiv trimite cereri primite de la placa ESP-32 către ECU prin coduri standard numite PID (Parameter ID).

Un PID este un șir de bytes, unde fiecare byte are o însemnătate. De exemplu, pentru citirea vitezei actuale ale vehiculului vom trimite 010D. Citirea oricărui parametru se face prin byte-ul 01 (pe care îl vom folosi de acum încolo), urmat de parametrul dorit (în acest caz 0D). Pentru turația motorului avem PID-ul 0C, pentru consumul de combustibil avem 5E, pentru temperatura uleiului între 01 si 1F, depinzând de producătorul vehiculului, pentru temperatura lichidului de răcire avem 05 iar pentru presiunea galeriei de admisie se va referi prin valoarea 0B.

Mai departe, ECU va returna valoarea dorita către cititorul OBD-II (dotat cu ELM327), care la rândul lui va trimite valoarea către ESP-32. Acesta va afișa valoarea pe un server web, cu scopul de a afișa parametrii autoturismului într-o maniera user-friendly.

Placa ESP-32 este programată astfel încât să folosească funcții dintr-un API open-source, numit ELMduino, disponibil pe plăcile de tip ESP-32 Development board. Acest API are rolul de a forma și identifica PID-urile și de a le trimite prin Bluetooth către cititorul OBD-II. Pe de alta parte, este necesar ca placa sa fie conectată la o rețea pentru a putea găzdui cu succes serverul cu scopul afișării datelor. Această placă a fost aleasă datorită capabilităților WiFi si Bluetooth pe care le prezintă.

# Abstract

# INTRODUCERE

## **Aspecte generale - PROBLEM STATEMENT, masina folosita, protocoale, abrevieri, specificatiile proiectului etc.**

## SOTA - State of the art, studiu de piata

## ESP-32

### Modulul integrat Bluetooth

### Modulul integrat WiFi

### Dependinta SPIFFS

### Dependinta WebAsyncServer

## ELM327 OBDII - scurta descriere protocol OBDII

## Git repo - modul de gestionare

## ELMduino

# ARHITECTURA

## Diagrama de decizii generica

## Arhitectura Bluetooth

### Diagrama de decizii detaliata

## Arhitectura WebAsyncServer

### Diagrama de decizii detaliata

# IMPLEMENTARE DETALIATA

## Implementare Bluetooth

### Diagrama de secventa dinamica - Din avion

### Diagrama de decizii detaliata

### Diagrama de secventa pentru rpm() - Se adauga si detalii de returnari, etc.

### Diagrama de secventa pentru kph() - Se adauga si detalii de returnari, etc.

### Diagrama de secventa pentru fuel() - Se adauga si detalii de returnari, etc.

### Diagrama de secventa pentru oil() - Se adauga si detalii de returnari, etc.

## Implementare WebAsyncServer

### Diagrama de secventa dinamica - Din avion

### Diagrama de decizii detaliata

### Diagrama de secventa pentru WebServer.begin()

### Diagrama de secventa pentru send data

### Diagrama de secventa pentru SPIFFS open file

### Diagrama de secventa pentru grafice

# DESFASURARE EXEMPLIFICATA

# DEZVOLTARE SI TESTARE

## Detalii de configurare - configurare config.h

## Securitate

## Stocarea datelor

## Interfata cu utilizatorul

## Posibili clienti

## Performanta sistemului - cata memorie ocupa, frecventa de operare, de returnare a datelor etc.

## Scalabilitate tehnica - se pot adauga mai multe esp uri

## Scalabilitate antreprenoriala - posibil pret, specificiatii, pachet, venit anual etc (vezi GreenSpot)

# Concluzie

## Obiective indeplinite

## Comparare cu SOTA

## Adaugari ulterioare

# Referinte