Dezvoltarea unui sistem de colectare a datelor de mediu din zone greu accesibile

# Documentul de proiectare

Cuprins

[1. Introducere 1](#_Toc160527836)

[1.1 Scopul documentului 1](#_Toc160527837)

[2. Prezentare generală și abordări de proiectare 2](#_Toc160527838)

[2.1 Prezentare generală 2](#_Toc160527839)

[2.2 Presupuneri/ Constrângeri/ Riscuri 2](#_Toc160527840)

[2.2.1 Presupuneri 2](#_Toc160527841)

[2.2.2 Constrângeri 2](#_Toc160527842)

[2.2.3 Riscuri 3](#_Toc160527843)

[3. Considerațiii de proiectare 4](#_Toc160527844)

[3.1 Obiective și linii directoare (ghiduri) 4](#_Toc160527845)

[3.2 Metode de dezvoltare 4](#_Toc160527846)

[3.3 Strategii de arhitectură 4](#_Toc160527847)

[4. Arhitectura Sistemului și Proiectarea Arhitecturii 6](#_Toc160527848)

[4.1 Vedere logică 6](#_Toc160527849)

[4.2 Arhitectură hardware 6](#_Toc160527850)

[4.3 Arhitectură software 6](#_Toc160527851)

[4.4 Arhitectura informațiilor 7](#_Toc160527852)

[4.5 Arhitectura de comunicații interne 7](#_Toc160527853)

[4.6 Diagrama de arhitectură a sistemului 8](#_Toc160527854)

[5. Proiectarea sistemului 9](#_Toc160527855)

[5.1 Proiectarea bazei de date 9](#_Toc160527856)

[5.1.1 Obiecte de date și structuri de date rezultante 9](#_Toc160527857)

[5.1.2 Fișiere și baze de date 9](#_Toc160527858)

[5.2 Conversii de date 9](#_Toc160527859)

[5.3 Interfețe utilizator 10](#_Toc160527860)

[5.3.1 Intrări 10](#_Toc160527861)

[5.3.2 Ieșiri 10](#_Toc160527862)

[5.4 Proiectarea interfețelor cu utilizatorul 10](#_Toc160527863)

[6. Scenarii de utilizare 11](#_Toc160527864)

[7. Proiectare de detaliu 12](#_Toc160527865)

[7.1 Proiectare hardware de detaliu 12](#_Toc160527866)

[7.2 Proiectare software de deatliu 12](#_Toc160527867)

[7.3 Proiectare detaliată de securitate 13](#_Toc160527868)

[7.4 Proiectare de detaliu pentru performanța sistemului 14](#_Toc160527869)

[7.5 Proiectare detaliată a comunicațiilor interne (între componente) 14](#_Toc160527870)

[8. Controale pentru verificarea integrității sistemului 15](#_Toc160527871)

[Anexa A: Gestiunea modificărilor documentului 16](#_Toc160527872)

[Anexa B: Acronime 17](#_Toc160527873)

[Anexa C Documente la care se face referire 18](#_Toc160527874)

## Introducere

Identificarea sistemului:

* Numele complet al proiectului: Sistem de Monitorizare și Control al Mediului cu RC Car
* Acronim: SMCM\_RC
* Situația existentă: În prezent, nu există un sistem similar implementat.
* Situația propusă: Implementarea unui sistem de monitorizare și control al mediului utilizând un RC Car echipat cu senzori și o cameră de streaming video, controlat prin intermediul unei aplicații mobile sau web.
* Evoluția așteptată a documentului: Documentul de Proiectare a Sistemului va fi actualizat pe măsură ce se vor face modificări în proiect și vor fi adăugate noi funcționalități sau îmbunătățiri.

Considerații referitoare la securitate și confidențialitate:

* Accesul la datele colectate de senzori și la fluxul video trebuie să fie securizat pentru a proteja confidențialitatea informațiilor.
* Protocolul de comunicare între aplicație și RC Car ar trebui să fie securizat pentru a preveni accesul neautorizat sau interceptarea datelor de către terți.
* Se vor implementa măsuri de securitate pentru a preveni manipularea sau sabotarea sistemului, asigurând integritatea și fiabilitatea operațiunilor.
* Va fi luată în considerare securitatea fizică a echipamentului, inclusiv a RC Car-ului și a dispozitivelor de control, pentru a preveni furtul sau deteriorarea acestora.

### Scopul documentului

Scopul SDD-ului (Software Design Document) este de a furniza o descriere detaliată a arhitecturii și designului sistemului de software în cadrul proiectului "Sistem de Monitorizare și Control al Mediului cu RC Car". Acest document servește drept ghid pentru echipa de dezvoltare, oferind îndrumare asupra modului în care componentele sistemului ar trebui să fie proiectate și implementate. Publicul țintă al acestui document include managerul de proiect, echipa de proiect și echipa de dezvoltare. Principalele obiective ale SDD-ului sunt să ofere o înțelegere clară a arhitecturii sistemului, să stabilească interfețele și relațiile dintre diferitele componente, să detalieze cerințele de proiectare și să ofere o bază pentru implementare. De asemenea, SDD-ul poate fi împărtășit cu clienții și alte părți interesate pentru a asigura alinierea asupra interfeței cu utilizatorul și pentru a obține aprobările necesare.

## Prezentare generală și abordări de proiectare

Această secțiune descrie principiile și strategiile care vor fi utilizate ca ghiduri în momentul proiectării și implementării sistemului.

### Prezentare generală

RC Car-ul este controlat de o telecomandă fizică, nu de aplicație. Aceasta permite utilizatorului să dirijeze mișcarea RC Car-ului și să activeze funcțiile sale.

Senzorii de mediu sunt conectați direct la Raspberry Pi, care servește drept hub central pentru datele de mediu.

Un server Raspberry Pi rulează atât un server Flask, cât și un WebSocket. Serverul Flask oferă o interfață API pentru controlul altor dispozitive sau senzori, dacă este necesar, în timp ce WebSocket-ul facilitează comunicarea în timp real între Raspberry Pi și serverul central.

Serverul central, care rulează pe un PC, este un server Node.js. Aici, datele colectate de la senzorii de mediu sunt stocate într-o bază de date și sunt gestionate de serverul Node.js.

Comunicarea între serverul central și aplicația mobilă sau web se face prin intermediul WebSocket, asigurând o comunicare în timp real și actualizări rapide ale datelor.

Această arhitectură permite controlul eficient al RC Car-ului, colectarea și stocarea datelor de mediu și transmiterea acestora către aplicație pentru monitorizare și analiză ulterioară.

### Presupuneri/ Constrângeri/ Riscuri

#### Presupuneri

Hardware-ul asociat:

* RC Car-ul trebuie să fie compatibil cu modulul Arduino Nano pentru a permite controlul său.
* Senzorii de mediu, cum ar fi DHT11, BMP180 și MQ4, trebuie să fie compatibili cu Raspberry Pi pentru a permite colectarea datelor de mediu.
* Camera Raspberry Pi trebuie să fie funcțională și să ofere un flux video stabil pentru a permite monitorizarea în timp real.

Sisteme de operare:

* Raspberry Pi trebuie să ruleze un sistem de operare compatibil, cum ar fi Raspbian, pentru a permite funcționarea corectă a serverului Flask și a WebSocket-ului.
* Serverul central trebuie să ruleze un sistem de operare compatibil cu Node.js și MongoDB pentru a permite stocarea și gestionarea datelor de mediu.

Caracteristicile utilizatorilor finali:

* Utilizatorii finali trebuie să fie capabili să utilizeze o aplicație mobilă sau web pentru a monitoriza și controla RC Car-ul și pentru a vizualiza datele de mediu colectate.
* Utilizatorii trebuie să aibă cunoștințe minime despre utilizarea unei aplicații mobile sau web.

Modificări probabile ale funcționalității:

* Posibilele modificări ale funcționalității ar putea include adăugarea sau eliminarea de senzori de mediu în sistem sau extinderea funcționalității RC Car-ului prin adăugarea de componente suplimentare, cum ar fi un sistem de navigare sau de evitare a obstacolelor.
* Îmbunătățiri ale performanței și securității sistemului ar putea necesita actualizări ale software-ului sau hardware-ului existent.

#### Constrângeri

Resurse hardware și software limitate:

* Capacitatea limitată de procesare și memorie a dispozitivelor Raspberry Pi și Arduino Nano poate afecta performanța și capacitatea sistemului de a gestiona volumul mare de date colectate și transmise.
* Limitările de memorie și putere de calcul pot necesita optimizări ale software-ului pentru a asigura funcționarea eficientă a sistemului.

Cerințe de interoperabilitate:

* Asigurarea interoperabilității între diferitele componente hardware și software ale sistemului poate fi dificilă din cauza variațiilor în standardele și protocoalele de comunicare utilizate.
* Integrarea cu alte sisteme externe sau dispozitive poate necesita adoptarea standardelor și protocoalelor comune pentru a facilita schimbul de date și comunicarea între acestea.

Cerințe de securitate:

* Protejarea datelor colectate și transmise este o prioritate și poate necesita implementarea măsurilor de securitate adecvate pentru a preveni accesul neautorizat sau compromiterea informațiilor.
* Asigurarea securității comunicațiilor și a serverelor este esențială pentru a proteja integritatea și confidențialitatea datelor.

Cerințe de performanță:

* Sistemul trebuie să fie capabil să gestioneze volumul mare de date colectate și transmise în timp real, asigurând în același timp o latență minimă în transmiterea acestora.
* Optimizarea performanței software-ului și hardware-ului este esențială pentru a asigura o experiență fluentă a utilizatorului și o funcționare eficientă a sistemului în condiții diverse de utilizare.

Cerințe de rețea și comunicații:

* Fiabilitatea și stabilitatea rețelei trebuie să fie asigurate pentru a permite transmiterea eficientă a datelor între diferitele componente ale sistemului și utilizatorii finali.
* Capacitatea de a gestiona și de a reacționa la eventualele probleme de conectivitate sau întreruperi ale rețelei este esențială pentru a menține funcționarea corespunzătoare a sistemului.

#### Riscuri

Risc de compatibilitate hardware și software: Variabilitatea echipamentelor hardware și software utilizate poate duce la dificultăți în asigurarea compatibilității și interoperabilității între componentele sistemului. Pentru a reduce acest risc, se va efectua o evaluare detaliată a compatibilității între diferitele dispozitive și software-uri utilizate, iar implementarea standardelor și protocoalelor comune va fi prioritară.

Risc de securitate cibernetică: Vulnerabilitățile de securitate în software sau rețele pot expune sistemul la riscul de acces neautorizat, compromiterea datelor sau atacuri cibernetice. Pentru a reduce acest risc, se vor implementa măsuri de securitate robuste, cum ar fi criptarea datelor, autentificarea utilizatorilor și monitorizarea continuă a activității de rețea.

Risc de fiabilitate hardware și software: Defectele hardware sau erorile de software pot duce la funcționarea necorespunzătoare a sistemului sau la eșecul acestuia. Pentru a reduce acest risc, se va efectua o testare riguroasă a hardware-ului și software-ului înainte de implementare, iar implementarea unor mecanisme de backup și recuperare a datelor va fi luată în considerare.

Risc de performanță scăzută: Datorită volumului mare de date și complexității sistemului, există riscul ca performanța să fie sub așteptări, ceea ce ar putea afecta experiența utilizatorului și funcționarea generală a sistemului. Pentru a reduce acest risc, se va efectua un test de performanță extensiv pentru a identifica și a remedia eventualele puncte slabe și se va optimiza continuu software-ul și hardware-ul pentru a asigura o performanță optimă.

Risc de dependență a resurselor externe: Sistemul poate depinde de resurse externe, cum ar fi serviciile de cloud sau furnizorii de date, iar indisponibilitatea acestora ar putea afecta funcționarea sistemului. Pentru a reduce acest risc, se va efectua o evaluare atentă a dependențelor externe și se vor implementa strategii de backup și redundanță pentru a asigura disponibilitatea continuă a resurselor critice.

Strategiile propuse pentru reducerea acestor riscuri includ:

* Realizarea unei analize exhaustive a riscurilor și a vulnerabilităților sistemului înainte de implementare.
* Implementarea măsurilor de securitate adecvate, inclusiv criptarea datelor, autentificarea utilizatorilor și monitorizarea activității de rețea.
* Testarea riguroasă a hardware-ului și software-ului pentru a identifica și a remedia eventualele erori sau defecte înainte de implementare.
* Utilizarea unor strategii de backup și recuperare a datelor pentru a asigura disponibilitatea continuă a informațiilor critice.
* Stabilirea unor parteneriate solide cu furnizorii de resurse externe și dezvoltarea unor planuri de continuitate a afacerilor pentru a gestiona eventualele indisponibilități sau întreruperi ale serviciilor.

## Considerațiii de proiectare

### Obiective și linii directoare (ghiduri)

Eficiență și performanță: Accentul principal este pus pe eficiența și performanța sistemului, asigurându-ne că acesta răspunde rapid la solicitările utilizatorilor și procesează datele într-un mod eficient.

Fiabilitate și stabilitate: Prioritatea este acordată fiabilității și stabilității sistemului, pentru a evita căderile și erorile, asigurându-ne că sistemul este disponibil și funcțional în permanență.

Simplificare și claritate: Ne străduim să proiectăm o interfață simplă și intuitivă pentru utilizatori, cu o navigare clară și funcționalități ușor de înțeles, pentru a îmbunătăți experiența utilizatorului.

Securitate și confidențialitate: O atenție deosebită este acordată securității și confidențialității datelor, asigurându-ne că informațiile utilizatorilor sunt protejate împotriva accesului neautorizat și a atacurilor cibernetice.

Flexibilitate și scalabilitate: Proiectăm sistemul să fie flexibil și scalabil, astfel încât să poată crește și să se adapteze la cerințele și volumul de date în creștere, fără a necesita modificări majore de design.

Documentație și comentarii: Ne angajăm să furnizăm documentație detaliată și comentarii relevante în codul sursă, pentru a facilita înțelegerea și dezvoltarea ulterioară a sistemului de către alți dezvoltatori.

Uzabilitate și accesibilitate: Accentul este pus pe uzabilitate și accesibilitate, asigurându-ne că sistemul poate fi utilizat ușor și eficient de către toți utilizatorii, indiferent de abilități sau nevoi specifice.

Aceste obiective și linii directoare sunt fundamentale pentru designul sistemului și al software-ului și ne vor ghida în dezvoltarea unei soluții care să îndeplinească nevoile și așteptările utilizatorilor, în timp ce respectă standardele de calitate și securitate.

### Metode de dezvoltare

Pentru designul sistemului și al software-ului, urmărim o abordare orientată pe obiecte și structurat, folosind mai multe tehnologii și instrumente adecvate pentru fiecare componentă a sistemului. Alegem această abordare deoarece oferă o organizare clară și modulară a codului și a funcționalităților, facilitând dezvoltarea, testarea și întreținerea ulterioară a sistemului.

În ceea ce privește metodele și instrumentele utilizate, ne bazăm pe următoarele:

UML (Unified Modeling Language): Folosim diagrame UML pentru a modela structura și comportamentul sistemului, inclusiv diagrame de clase, diagrame de secvențe și diagrame de activități, pentru a ajuta la înțelegerea și comunicarea designului sistemului între membrii echipei de dezvoltare.

Programarea orientată pe obiecte (OOP): Ne bazăm pe principiile și conceptele OOP pentru a organiza codul în clase și obiecte, facilitând reutilizarea, modularitatea și extensibilitatea sistemului.

Metodologia Agile: Adoptăm o abordare Agile în dezvoltarea sistemului, folosind iterații scurte și cicluri de feedback pentru a livra continuu valoare clienților și pentru a răspunde flexibil la schimbările cerințelor și priorităților.

Biblioteci și framework-uri specifice: Selectăm și utilizăm biblioteci și framework-uri specifice pentru fiecare componentă a sistemului, în funcție de cerințe și nevoile tehnice ale proiectului. De exemplu, pentru partea de frontend, putem folosi React Native pentru dezvoltarea aplicației mobile, în timp ce pentru partea de backend putem utiliza Express.js pentru crearea serverului Node.js.

Bune practici de codare și standarde: Respectăm bunele practici de codare și standardele de dezvoltare, pentru a asigura o calitate ridicată a codului și o coerență în întregul proiect.

### Strategii de arhitectură

Pentru proiectul nostru, decizia de a utiliza o arhitectură bazată pe microservicii este crucială pentru a asigura o dezvoltare și o scalabilitate eficiente. Prin împărțirea sistemului în servicii independente, cum ar fi serviciul de control al RC car-ului, serviciul de comunicație cu senzorii și serviciul de interfață cu utilizatorul, putem gestiona fiecare aspect al sistemului în mod separat și eficient.

Această abordare ne permite să dezvoltăm și să extindem fiecare componentă a sistemului în mod independent, fără a afecta celelalte. De exemplu, putem adăuga noi funcționalități la serviciul de control al RC car-ului fără a influența serviciul de comunicație sau interfața cu utilizatorul. De asemenea, putem scala și gestiona fiecare serviciu în mod individual în funcție de cerințele sale specifice.

În ceea ce privește alternativele, am luat în considerare și arhitectura monolitică și arhitectura bazată pe servicii. Cu toate acestea, arhitectura bazată pe microservicii se potrivește cel mai bine nevoilor noastre, oferind flexibilitate, scalabilitate și gestionare eficientă a sistemului nostru RC car.

Utilizarea limbajului de programare Python: Am ales Python datorită versatilității sale și suportului puternic pentru dezvoltarea de aplicații web și IoT.

Reutilizarea bibliotecilor și framework-urilor existente: Am optat pentru utilizarea de biblioteci și framework-uri Python precum Flask pentru serverul web și Biblioteca RPi.GPIO pentru interacțiunea cu GPIO-urile Raspberry Pi-ului.

Extinderea și îmbunătățirea continuă a software-ului: Avem planuri pentru adăugarea de funcționalități suplimentare, cum ar fi controlul vocal al RC car-ului și integrarea cu asistenții virtuali.

Paradigma interfeței utilizatorului: Interfața utilizatorului este concepută pentru a fi simplă și intuitivă, cu elemente de control ușor de utilizat și o prezentare clară a datelor de la senzori.

Detectarea și recuperarea erorilor: Implementăm mecanisme robuste pentru detectarea și gestionarea erorilor, inclusiv înregistrarea detaliilor erorilor și furnizarea de mesaje de eroare clare pentru utilizatori.

Gestionarea memoriei și a resurselor: Ne asigurăm că software-ul nostru utilizează eficient memoria și resursele hardware disponibile, evitând astfel supraîncărcarea sau blocarea sistemului.

Utilizarea bazei de date MongoDB pentru persistența datelor: Alegem MongoDB pentru gestionarea datelor sistemului nostru, datorită flexibilității sale și capacitatii de scalabilitate.

Comunicarea prin WebSocket: Folosim WebSocket pentru comunicarea în timp real între diferitele componente ale sistemului nostru, asigurând o comunicare rapidă și eficientă.

Managementul rețelei și a datelor distribuite: Implementăm protocoale și algoritmi pentru gestionarea comunicației și a datelor distribuite între diferitele componente ale sistemului.

Abordări generalizate pentru control: Proiectăm software-ul pentru a permite controlul flexibil și modular al diferitelor aspecte ale sistemului, permițând adaptarea ușoară la cerințele specifice ale utilizatorului.

Concurență și sincronizare: Implementăm mecanisme pentru gestionarea concurenței și sincronizarea operațiilor asincrone în cadrul sistemului nostru, asigurând coerența și integritatea datelor.

Mecanisme de comunicare: Folosim diverse mecanisme de comunicare, cum ar fi WebSocket pentru comunicația în timp real și REST API-uri pentru interacțiunea cu alte sisteme sau servicii.

## Arhitectura Sistemului și Proiectarea Arhitecturii

Proiectul are la baza cateva componente principale cum ar fi: macheta , telecomanda , 2 placi de dezvoltare Arduino , un Arduino Nano si un Arduino Uno , doua Esp8266 , Raspberry Pi , camera Raspberry Pi , si un Electronic Speed Controller .

Macheta este reprezentata de o masina rc pe care am modificat-o astfel incat sa pot sa montez senzori , microcontrollere si celelate componente.

Telecomanda masinii este responsabila pentru controlul masinii dar si pentru sustinerea unui telefon intr-un suport special , pentru a putea urmarii livestream-ul camerei montate pe masina , in aplicatia mobile.

Pentru telecomada , am folosit doua potentiometre , unul pentru acceleratie , oprire respectiv mers inapoi , iar celalt potentiometru pentru a putea controla directia masinii , cat si unghiul de viraj. Aceste doua potentiometre le-am conectat la un Arduino Uno si am citit aceste valori analogice. Mai departe pentru a trimite aceste date la receptorul montat pe masina a trebuit sa caut o metoda sa fac acest lucru. Am folosit comunicarea seriala intre Arduino si Esp8266 prin conectarea pinului TX al Arduino la pinul RX al Esp8266. Dupa trimiterea acestor date , Esp8266 al telecomenzii va trebui sa trimita mai departe datele pentru a ajunge la Electronic Speed Controller si la un servo motor folosit pentru directia masinii.

Pentru a primi aceste date , am folosit inca un Esp8266 pe care l-am montat pe macheta , apoi , am reusit prin programarea celor doua Esp8266 sa am o cumincatie intre cele 2 folosind un protocol de comunicatie numit EspNow, care este optimizat pentru comunicarea între dispozitive ESP8266 și ESP32, fără a fi necesară o conexiune la un router WiFi. Acest protocol este ideal pentru aplicații care necesită comunicații rapide și eficiente între dispozitivele ESP, cum ar fi sistemele de automatizare, jocurile multiplayer și multe altele.

Deoarece acest modul de comunicare nu are nevoie de internet , este ideal pentru proiectul meu , pentru ca pot utiliza acest sitem fara a avea nevoie de o conexiune directa la un router , adica il pot folosi aproape oriunde.

Dupa primirea datelor pe Esp-ul 8266 de pe macheta a trebuit sa conectez si celelate componente pentru a reusi sa pun in fucntiune masina. Esc l-am conectat direct la un pin digital al esp-ului , pin care trebuie sa aiba PWM , altfel nu o sa functioneze . A urmat conectarea servo-ului la un alt pin digital , esc-ul find folosit pentru controlul motorului.

Odata cu conectarea acest doua componente , a trebuit modificat si codul esp-ului de pe macheta , deoarece trebuia acum sa reglez si sa testez valori optime pentru controlul esc-ului si a servo-ului. Pentru a realiza acest lucru a trebuit sa inteleg cum functioneaza ambele componente si dupa mai multe teste am reusit sa realizez mersul inainte si inapoi folosind acest esc , mai ramanea partea de viraj .

Deoarece esc-ul se alimenteaza extern de la o acumulator de 7.2V nu a fost o problema functionarea optima a acestuia , dar din cauza ca Esp8266 este bazat pe o tensiune 3.3V , valorile trimise de la telecomanda , erau mapate diferit si virajul se realiza foarte ineficient , si foarte lent.

Din cauza acestei probleme am adaugat inca un Arduino Nano care primeste datele de la servo prin comunicatia seriala , iar apoi acel servo l- am conectat la acest Arduino , si dupa mici modificari in cod si teste , a functionat cum era de asteptat.

Alimentarea telecomenzii se realizeaza folosind o baterie de 9V conectata pe pinul de Vin si Ground al placii . Pentru alimentarea Esp-ului din telecomanda , am ales alimentarea directa de la Arduino , cu 5V , care este suficienta pentru Esp8266.

Alimentarea Esc-ului , Esp-ului si Arduino Nano , sunt realizate de un acumulator de 7.2V 5000mah care poate furniza un votaj de 5V ideal pentru aceste componente , dar si pentru motorul electric al machetei.

Odata cu finalizarea acestor provocari , a urmat sa montez un Raspberry Pi pe aceasta macheta , umpreuna cu niste senzori de mediu , led-uri si inclusiv camera Raspberry-ului.

Pentru camera respectiva am reusit sa fac un sistem in care pot deplasa camera spre dreapta sau stanga si in sus sau jos , folosind 2 servou-uri care vor fi actionate de ulilizator direct din aplicatia Android.

Colectarea datelor de mediu va fi realizata cu ajutorul unor senzori precum: DHT11 , care este folosit pentru obtinerea datelor temperaturii si umiditatii, BMP180 , folosit pentru obtinerea datelor despre presiune atmosferica , temperatura si altitudine , MQ4 , senzor care indica concetratia de metan.

Dupa conectarea acestor senzori la Raspberry , am realizat un server de Flask , un framework pentru python , unde am definit toti sezorii si am scris codul necesar pentru functionarea lor. Apoi aceste date obtinute de la senzori trebuiau cumva trimise la un alt server sau direct la aplicatie.

Eu am ales sa fac un server separat in node js unde sa primesc aceste date de la Raspberry si apoi sa le salvez intr-o baza de date mongo. Dupa configurarea beazei de date si am putut stoca cu succes datele primite de la Raspberry.

Dar serverul meu trimite mai departe aceste date si catre aplicatia Android pentru a vedea in timp real aceste date de mediu. Toate aceasta cumincatie se realizeaza printr-un mod special de comunicatie numit WebSocket.

WebSocket este un protocol de comunicație bidirecțională care permite o conexiune persistentă între un client și un server prin intermediul unei conexiuni TCP/IP. Procesul începe cu un handshake HTTP, în care clientul solicită schimbarea protocolului la WebSocket, iar serverul confirmă și stabilește conexiunea WebSocket. După deschiderea conexiunii, atât clientul, cât și serverul pot trimite și primi mesaje în timp real, cum ar fi texte, șiruri JSON sau imagini. Conexiunea WebSocket rămâne deschisă și activă, permițând trimiterea și primirea de date fără a fi necesară deschiderea conexiunilor noi pentru fiecare mesaj. Procesul se încheie atunci când clientul sau serverul decide să închidă conexiunea, trimitând un mesaj de închidere și primind un răspuns de confirmare. WebSocket este un protocol eficient și puternic, utilizat frecvent în dezvoltarea aplicațiilor web în timp real, cum ar fi chat-ul online sau jocurile multiplayer.

In acelasi timp trebuie sa trimit date spre Raspberry pentru a putea controla alti senzori de pe macheta , led-uri si cele 2 servo-uri folosite pentru controlul unghiului camerei.

Cu ajutorul a doua slidere implementate in aplicatie , trimit datele lor pe server , de unde sunt trimise mai departe la Raspberry , de unde , dupa parsarea acestor date , vom putea controla miscarea servo-urilor in sus , jos , stanga si dreapta.

Aplicatia proiectului este realizata cu ajutorul React-Native si Typescript, care permit construirea aplicațiilor mobile native folosind JavaScript și React. Aplicatia este formata din 4 pagini principale: pagina Home , pagina Control , pagina Statistici si pagina pentru Camera.

### Diagrama de arhitectură a sistemului

A diagram of a flowchart

Description automatically generated

## Proiectarea sistemului

### Proiectarea bazei de date

Temperature

* Tip: Număr
* Lungime: N/A
* Sursa: Datele primite de la senzorul de temperatură
* Reguli de validare: Necesară
* CRUD: Crea, citire, actualizare, ștergere
* Stocarea datelor: În baza de date MongoDB
* Ieșiri: Temperatura curentă
* Aliasuri: N/A
* Descriere: Temperatura măsurată de senzor

Humidity

* Tip: Număr
* Lungime: N/A
* Sursa: Datele primite de la senzorul de umiditate
* Reguli de validare: Necesară
* CRUD: Crea, citire, actualizare, ștergere
* Stocarea datelor: În baza de date MongoDB
* Ieșiri: Umiditatea curentă
* Aliasuri: N/A
* Descriere: Umiditatea măsurată de senzor

Gaz:

* Tip: Număr
* Lungime: N/A
* Sursa: Datele primite de la senzorul de calitate a aerului
* Reguli de validare: Necesară
* CRUD: Crea, citire, actualizare, ștergere
* Stocarea datelor: În baza de date MongoDB
* Ieșiri: Calitatea aerului curentă
* Aliasuri: N/A
* Descriere: Calitatea aerului măsurată de senzor

Pressure

* Tip: Număr
* Lungime: N/A
* Sursa: Datele primite de la senzorul de presiune
* Reguli de validare: Necesară
* CRUD: Crea, citire, actualizare, ștergere
* Stocarea datelor: În baza de date MongoDB
* Ieșiri: Presiunea atmosferică curentă
* Aliasuri: N/A
* Descriere: Presiunea atmosferică măsurată de senzor

Altitude:

* Tip: Număr
* Lungime: N/A
* Sursa: Datele calculate din presiunea atmosferică
* Reguli de validare: Necesară
* CRUD: Crea, citire, actualizare, ștergere
* Stocarea datelor: În baza de date MongoDB
* Ieșiri: Altitudinea curentă
* Aliasuri: N/A
* Descriere: Altitudinea calculată pe baza presiunii atmosferice

#### Fișiere și baze de date

În cadrul arhitecturii sistemului, datele vor fi stocate într-o bază de date MongoDB. Structura fizică a stocării datelor va fi gestionată de MongoDB, iar fișierele de date vor fi stocate în directorul specificat în configurarea serverului MongoDB.

MongoDB utilizează un model de stocare bazat pe colecții și documente. Fiecare document este un înregistrare JSON care conține datele propriu-zise. Documentele sunt stocate în cadrul unei colecții, care este echivalentul unei tabele din bazele de date relaționale.

Pentru modelul de date definit pentru senzori în aplicația noastră, datele vor fi stocate într-o colecție numită "sensors" în baza de date MongoDB. Fiecare document din colecția "sensors" va conține datele corespunzătoare unui singur set de date provenit de la senzori.

În ceea ce privește structurile de fișiere și locațiile acestora, MongoDB va gestiona aceste detalii intern. MongoDB folosește un sistem de fișiere specific pentru a stoca datele pe disc, iar locația acestor fișiere este specificată în configurarea serverului MongoDB. De obicei, MongoDB stochează datele într-un director specificat în configurarea sa, iar numele exact al acestui director poate varia în funcție de sistemul de operare și de configurarea specifică a serverului MongoDB.

Pentru a gestiona datele în cadrul arhitecturii sistemului nostru, trebuie să ne asigurăm că serverul MongoDB este configurat și rulează corect și că aplicația noastră este conectată la baza de date MongoDB pentru a putea stoca și accesa datele corespunzătoare senzorilor.

##### Baze de date

Pentru proiectarea detaliată a fișierelor de sistem de gestionare a bazelor de date (DBMS) în cadrul aplicației noastre, utilizăm MongoDB ca și soluție de stocare a datelor. MongoDB utilizează un model de stocare flexibil bazat pe documente JSON, care sunt stocate în colecții.

În MongoDB, fișierele de date sunt gestionate intern de serverul MongoDB și nu necesită o interacțiune directă cu utilizatorul. Structura fizică a stocării datelor este gestionată de MongoDB în mod transparent pentru dezvoltatori. Totuși, putem defini schema și configurațiile corespunzătoare pentru a asigura performanța și scalabilitatea necesare a sistemului nostru.

Pentru proiectarea detaliată a fișierelor DBMS în cadrul aplicației noastre, vom avea în vedere următoarele aspecte:

**Colecția "sensors"**: Aceasta este colecția în care vor fi stocate datele provenite de la senzori. Fiecare document din această colecție va reprezenta un set de date de la senzori și va conține câmpurile pentru temperatura, umiditate, aer, presiune și altitudine.

**Structura documentelor din colecția "sensors"**: Fiecare document va fi un obiect JSON ce va conține următoarele câmpuri:

**temperature**: de tip număr, reprezentând temperatura înregistrată de senzor.

**humidity**: de tip număr, reprezentând umiditatea înregistrată de senzor.

**air**: de tip număr, reprezentând calitatea aerului înregistrată de senzor.

**pressure**: de tip număr, reprezentând presiunea atmosferică înregistrată de senzor.

**altitude**: de tip număr, reprezentând altitudinea înregistrată de senzor.

**Indexarea colecției "sensors"**: Pentru a asigura o performanță optimă a interogărilor, vom indexa colecția "sensors" pe unul sau mai multe câmpuri, cum ar fi timestamp-ul sau un alt câmp relevant pentru cerințele noastre de interogare.

**Configurarea și administrarea serverului MongoDB**: Va trebui să configurăm și să administrăm serverul MongoDB pentru a asigura o funcționare corectă și eficientă a bazei de date. Acest lucru include configurarea parametrilor de stocare, gestionarea backup-urilor și monitorizarea performanței.

### Conversii de date

Pentru a gestiona conversiile de date necesare în cadrul aplicației noastre, vom utiliza diverse funcții și metode disponibile în limbajul de programare și bibliotecile pe care le folosim. În mod specific, vom avea nevoie de metode pentru a converti datele din diferite formate, cum ar fi string-uri în numere sau invers, formate de data și timp etc.

De obicei, astfel de conversii de date sunt gestionate în cadrul codului aplicației noastre, folosind funcții și metode specifice limbajului de programare și bibliotecilor asociate. De exemplu, în javascript, pentru a converti un string într-un număr, putem folosi funcția parseInt() sau parseFloat(), iar pentru a converti un obiect de data și timp într-un string formatat, putem folosi metodele toIsoString()

Pentru conversiile de date mai complexe sau pentru manipularea datelor în formate specifice, putem scrie funcții personalizate sau putem utiliza biblioteci terțe specializate în manipularea datelor.

#### Intrări

Pentru sistemul nostru de control al mașinii RC, mijloacele de intrare folosite de utilizatori sunt telecomanda pentru controlul direct al mașinii RC și aplicația mobilă pentru monitorizarea și controlul prin intermediul smartphone-ului. Iată o descriere detaliată a mijloacelor de intrare și a aspectului interfețelor utilizatorilor:

Telecomanda RC:

* Mijloc de intrare: Butonii direcționali și butoanele de control al vitezei de pe telecomanda RC.
* Fluxul de date: Comenzile de la butoanele direcționale și butoanele de control al vitezei sunt transmise de telecomandă către mașina RC pentru controlul mișcării și funcțiilor.
* Aspectul interfeței utilizatorului: Telecomanda poate avea butoane direcționale pentru controlul direcției (înainte, înapoi, stânga, dreapta) și butoane pentru controlul vitezei (accelerare, frânare).

Aplicația mobilă:

* Mijloc de intrare: Ecran tactil al smartphone-ului.
* Fluxul de date: Utilizatorii pot interacționa cu aplicația mobilă folosind ecranul tactil pentru a trimite comenzi către mașina RC și pentru a monitoriza datele de la senzori.
* Aspectul interfeței utilizatorului: Aplicația mobilă poate avea următoarele ecrane și elemente de interfață:
* Ecran de control al direcției și vitezei: Butonii virtuali pentru controlul direcției și vitezei mașinii RC.
* Ecran de monitorizare a datelor: Afișează datele de la senzori, cum ar fi temperatură, umiditate, presiune, altitudine etc.

Criteriile de editare pentru elementele de date pot include:

* Limitarea valorilor de intrare pentru a se potrivi cu limitele fizice ale mașinii RC și ale mediului înconjurător.
* Validarea formatului datelor introduse (de exemplu, numere pentru viteza, date și ore pentru timestamp-uri).
* Asigurarea completării obligatorii a anumitor câmpuri critice pentru a preveni date incomplete sau incorecte..

#### Ieșiri

Rapoarte:

* Cod/Nume: Raportul de diagnostic al mașinii RC
* Conținut: Acest raport conține informații despre starea generală a mașinii RC, inclusiv datele de la senzori (temperatură, umiditate, presiune, altitudine) și eventualele erori sau avertismente.
* Scopul: Principali utilizatori ai acestui raport sunt tehnicienii de service sau utilizatorii avansați care doresc să efectueze diagnosticarea mașinii RC și să identifice eventualele probleme.
* Restricții de acces: Accesul la acest raport ar trebui să fie restricționat doar la utilizatorii autorizați, cum ar fi tehnicienii de service sau administratorii sistemului, pentru a asigura confidențialitatea datelor și securitatea sistemului.

Ecrane de afișare a datelor:

* Cod/Nume: Ecranul de monitorizare a datelor în aplicația mobilă
* Conținut: Acest ecran afișează în timp real datele de la senzori, cum ar fi temperatura, umiditatea, presiunea și altitudinea, într-un format ușor de înțeles și de interpretat.
* Scopul: Utilizatorii principali ai acestui ecran sunt utilizatorii finali care doresc să monitorizeze mediul înconjurător și starea mașinii RC în timp real.
* Restricții de acces: Accesul la acest ecran ar trebui să fie disponibil pentru toți utilizatorii aplicației mobile, cu excepția cazului în care anumite date sunt considerate sensibile și trebuie restricționate doar la anumiți utilizatori autorizați.

### Proiectarea interfețelor cu utilizatorul

nterfețele proiectate pentru sistemul nostru de control al mașinii RC includ atât interfețe hardware, cum ar fi telecomanda și senzorii, cât și interfețe software, cum ar fi aplicația mobilă și serverele. Iată o scurtă descriere a fiecărei interfețe:

Interfața Hardware:

* Telecomanda: Aceasta este principala interfață prin care utilizatorii interacționează direct cu mașina RC. Telecomanda permite utilizatorilor să controleze direcția și viteza mașinii RC.
* Senzorii: Senzorii montați pe mașina RC (cum ar fi senzorii de temperatură, umiditate, presiune și altitudine) sunt, de asemenea, considerați interfețe hardware. Acești senzori colectează date despre mediul înconjurător și starea mașinii și le transmit către unitatea centrală pentru procesare ulterioară.

Interfața Software:

* Aplicația Mobilă: Aceasta este interfața principală pentru utilizatorii finali. Prin intermediul aplicației mobile, utilizatorii pot monitoriza datele de la senzori, pot controla mașina RC și pot primi rapoarte și notificări despre starea mașinii.
* Serverele (Node.js și Flask): Acestea sunt interfețele software care facilitează comunicarea între diferitele componente ale sistemului. Serverul Node.js gestionează comunicarea între aplicația mobilă și senzori, în timp ce serverul Flask gestionează comunicarea între Raspberry Pi și serverul central.

## Scenarii de utilizare

Scenariu operațional: Utilizare și monitorizare a mediului înconjurător cu mașina RC

Utilizatorul pornește telecomanda și mașina RC.

Sistemul RC car inițiază conectarea cu telecomanda și confirmă starea de conexiune.

Utilizatorul direcționează mașina RC către zona de interes folosind butoanele direcționale de pe telecomandă.

Senzorii de temperatură, umiditate, presiune și altitudine montați pe mașina RC încep să colecteze date despre mediul înconjurător și să le trimită la unitatea centrală.

Datele colectate sunt transmise de la mașina RC către Raspberry Pi prin intermediul conexiunii WebSocket.

Raspberry Pi primește datele și le transmite către serverul central de Node.js.

Serverul central procesează datele și le stochează în baza de date.

Utilizatorul deschide aplicația mobilă.

Utilizatorul accesează secțiunea de monitorizare a mediului înconjurător.

Aplicația mobilă solicită datele despre mediul înconjurător de la serverul central.

Serverul central furnizează datele stocate în baza de date către aplicația mobilă.

Utilizatorul vizualizează datele despre temperatură, umiditate, presiune și altitudine pe ecranul aplicației mobile.

Utilizatorul poate controla și direcția și viteza mașinii RC prin intermediul telecomenzii.

Utilizatorul oprește mașina RC și închide aplicația mobilă după finalizarea monitorizării mediului înconjurător.

Scenariul de mai sus ilustrează modul în care utilizatorii interacționează cu sistemul pentru a monitoriza mediul înconjurător folosind mașina RC și aplicația mobilă asociată. Sistemul permite utilizatorilor să monitorizeze în timp real datele colectate de senzori, să controleze mașina RC și să primească notificări relevante în funcție de datele măsurate.

## Proiectare de detaliu

### Proiectare hardware de detaliu

**Tehnologii Hardware**

1.0 Arduino Nano este o placă de dezvoltare mică și versatilă, care face parte din familia de plăci Arduino. Acesta este similar cu alte plăci Arduino, cum ar fi Arduino Uno, dar vine într-un factor de formă mai mic, fiind potrivit pentru proiecte cu restricții de spațiu. Cu toate acestea, Arduino Nano păstrează funcționalitățile și caracteristicile de bază ale plăcilor Arduino, permițând dezvoltatorilor să creeze proiecte interactive și dispozitive electronice cu ușurință.

Arduino Nano este echipat cu un microcontroler ATMega328P, care rulează la o frecvență de 16 MHz și vine cu o memorie flash de 32 KB și 2 KB de memorie RAM. Acesta dispune de o serie de pini de intrare/ieșire digitale și analogice, care pot fi utilizate pentru conectarea diferitelor senzori, module și alte componente electronice.

Placa Arduino Nano are, de asemenea, o gamă largă de caracteristici și interfețe, inclusiv porturi USB pentru programare și comunicație serială, un regulator de tensiune încorporat pentru alimentare și un bootloader pre-instalat, care facilitează programarea și încărcarea codului pe placă.

Datorită dimensiunilor sale compacte și versatilității sale, Arduino Nano este folosit într-o varietate largă de proiecte electronice, inclusiv în domenii precum roboți, senzori, dispozitive IoT, sisteme de monitorizare și control, și multe altele. Fiind compatibil cu mediul de dezvoltare Arduino IDE și având o comunitate activă de utilizatori și dezvoltatori, Arduino Nano este o alegere populară pentru cei care doresc să înceapă să exploreze lumea fascinantă a electronicii și programării.

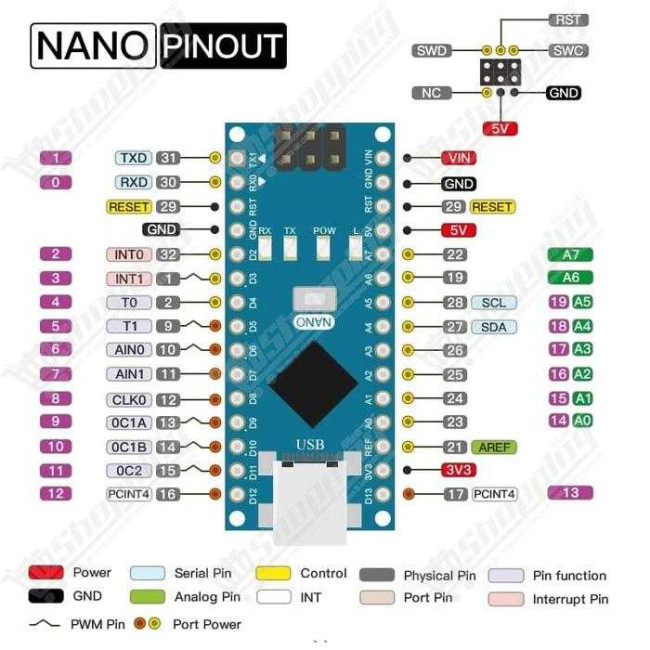


Figura 1.0 – Prezentare Generala Arduino Nano

**Caracteristici Arduino Nano**

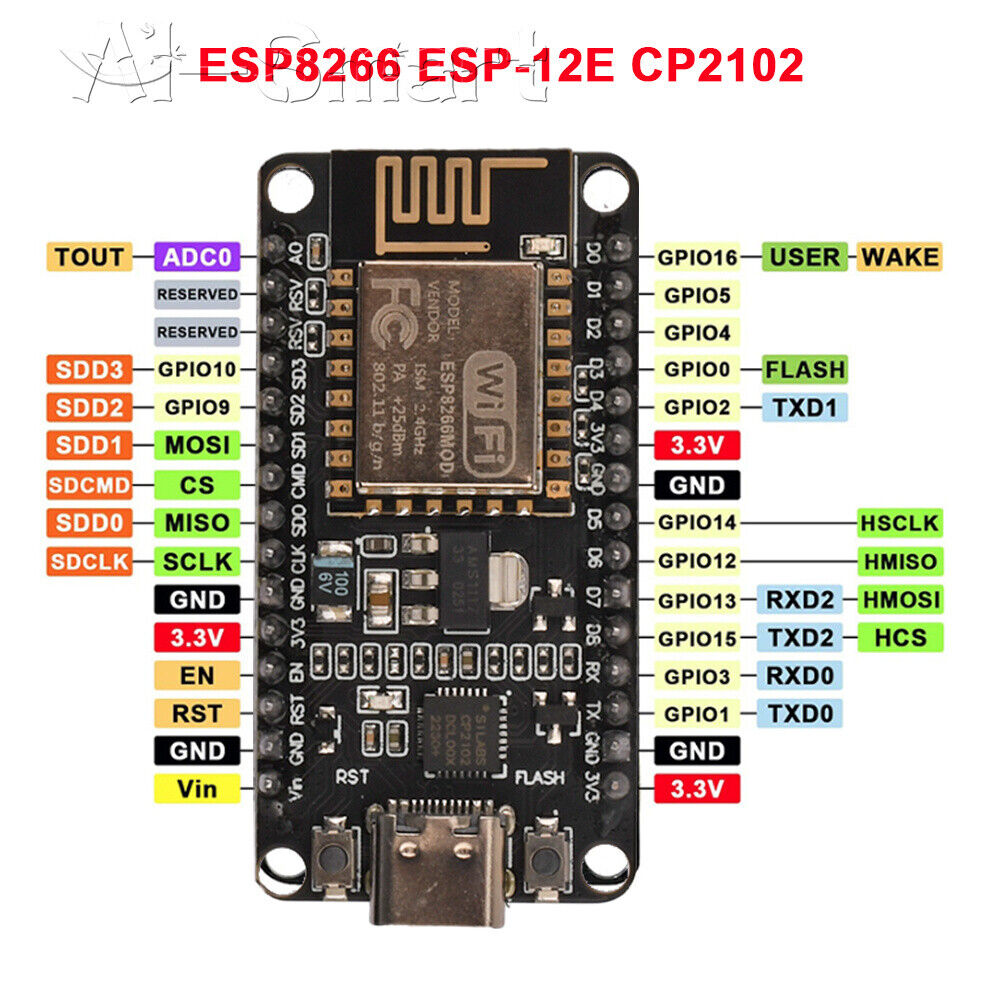
* ATmega328P Microcontroller is from 8-bit AVR family
* Operating voltage is 5V
* Input voltage (Vin) is 7V to 12V
* Input/Output Pins are 22
* Analog i/p pins are 6 from A0 to A5
* Digital pins are 14
* Power consumption is 19 mA
* I/O pins DC Current is 40 mA
* Flash memory is 32 KB
* SRAM is 2 KB
* EEPROM is 1 KB
* CLK speed is 16 MHz
* Weight-7g
* Size of the printed circuit board is 18 X 45mm
* Supports three communications like SPI, IIC, & USART

1.1 ESP8266 este o placă de dezvoltare WiFi mică și puternică, care este utilizată pentru a adăuga conectivitate WiFi la diferite dispozitive electronice. Acesta este echipat cu un microcontroler de tipul ESP8266 și poate funcționa independent sau poate fi conectat la alte plăci Arduino sau microcontrolere pentru a adăuga funcționalitate WiFi la acestea.

Placa ESP8266 vine în diferite variante și modele, printre care cele mai populare sunt ESP8266-01 și ESP8266 NodeMCU. Aceste plăci oferă diferite interfețe și caracteristici, dar toate sunt echipate cu un chip WiFi integrat și un set de pini de intrare/ieșire digitale și analogice, care permit conectarea diferitelor senzori, module și alte componente electronice.

Una dintre caracteristicile distinctive ale ESP8266 este capacitatea sa de a comunica prin WiFi, permițând dispozitivelor să se conecteze la rețele WiFi existente și să trimită și să primească date prin intermediul rețelei WiFi. Acest lucru deschide o gamă largă de posibilități pentru dezvoltarea de aplicații IoT (Internet of Things), cum ar fi senzori de mediu, sisteme de monitorizare și control, și multe altele.

Cu ajutorul mediului de dezvoltare Arduino IDE și a bibliotecilor disponibile, programarea și dezvoltarea de aplicații pentru ESP8266 este simplă și accesibilă. ESP8266 este utilizat într-o varietate largă de proiecte și aplicații, datorită dimensiunilor sale mici, puterii sale de procesare și conectivității sale WiFi. Este o alegere populară pentru dezvoltatorii care doresc să adauge conectivitate WiFi la proiectele lor electronice și să le conecteze la internet.



**Caracteristici Esp8266**

* Microcontroler: CPU Tensilica RISC de 32 de biți Xtensa LX106
* Tensiune de Operare: 3,3V
* Tensiune de Intrare: 7-12V
* Pinuri Digitale I/O (DIO): 16
* Pinuri de Intrare Analogice (ADC): 1
* UART-uri: 1
* SPI-uri: 1
* I2C-uri: 1
* Memorie Flash: 4 MB
* SRAM: 64 KB
* Viteză de Ceas: 80 MHz
* USB-TTL bazat pe CP2102 este inclus pe placă, permitând conectarea directă și utilizarea imediată (Plug n Play)
* Antenă PCB

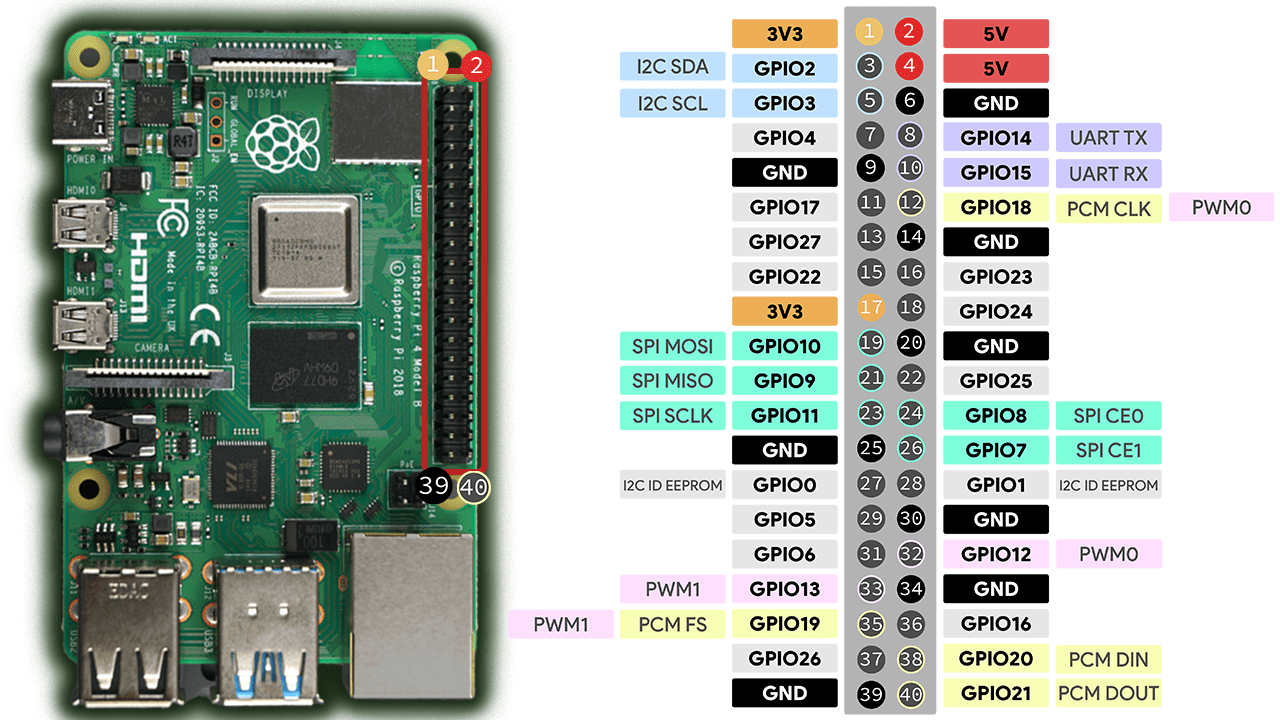
1.2 Raspberry Pi 3 este un computer de dimensiuni mici, bazat pe un procesor ARM, care a fost proiectat pentru a fi accesibil și versatil. Este unul dintre cele mai populare single-board computers (SBC) disponibile pe piață, fiind utilizat într-o varietate largă de proiecte și aplicații.

Raspberry Pi 3 este echipat cu un procesor quad-core ARM Cortex-A53 cu o frecvență de 1.2 GHz și 1 GB de memorie RAM, oferind suficientă putere de calcul pentru a rula o gamă largă de aplicații și sisteme de operare. Acesta dispune de mai multe porturi de conectivitate, inclusiv porturi USB, port Ethernet, port HDMI și un port GPIO (General Purpose Input/Output), care permit conectarea diferitelor periferice și componente.

Una dintre caracteristicile distinctive ale Raspberry Pi 3 este conectivitatea sa WiFi și Bluetooth integrate, care permit dispozitivului să se conecteze la rețele WiFi existente și să comunice cu alte dispozitive Bluetooth. Acest lucru îl face potrivit pentru proiecte IoT, automatizare și rețele de tipul smart home.

Raspberry Pi 3 rulează o varietate de sisteme de operare, inclusiv Raspbian (o distribuție Linux optimizată pentru Raspberry Pi), Ubuntu, Windows 10 IoT Core și altele. Acest lucru oferă flexibilitate în dezvoltarea de aplicații și proiecte pe Raspberry Pi 3, fiind potrivit pentru utilizatori cu diferite niveluri de experiență în programare și electronică.

Cu ajutorul portului GPIO și a bibliotecilor disponibile, Raspberry Pi 3 poate fi utilizat pentru a interacționa cu o varietate largă de senzori, module și alte componente electronice, permițând dezvoltatorilor să creeze proiecte interactive și aplicații complexe. În concluzie, Raspberry Pi 3 este o platformă puternică și versatilă, potrivită pentru o varietate largă de proiecte și aplicații, de la proiecte educative și hobby-uri la soluții IoT și industriale.



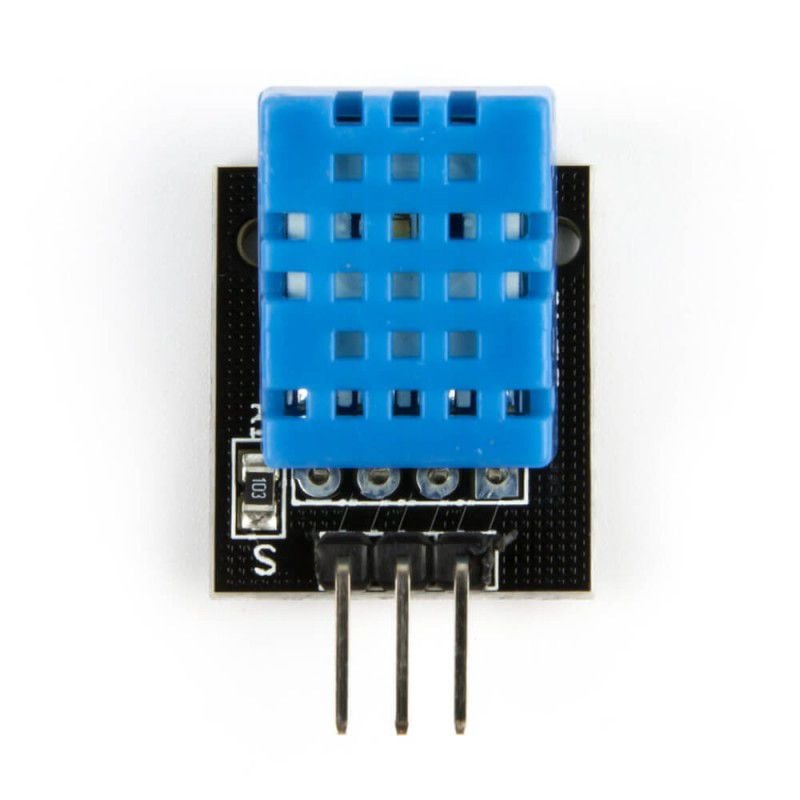
**Caracteristici Raspberry PI**

* Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
* 1GB LPDDR2 SDRAM
* 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
* Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)
* Extended 40-pin GPIO header
* Full-size HDMI®
* 4 USB 2.0 ports
* CSI camera port for connecting a Raspberry Pi camera
* DSI display port for connecting a Raspberry Pi touchscreen display
* 4-pole stereo output and composite video port
* Micro SD port for loading your operating system and storing data
* 5V/2.5A DC power input
* Power-over-Ethernet (PoE) support (requires separate PoE HAT)

1.3 DHT11 este un senzor de temperatură și umiditate digital, care este folosit într-o varietate largă de aplicații, de la controlul climei în interiorul locuințelor la monitorizarea mediului înconjurător în sistemele de automatizare. Acest senzor este accesibil, ușor de utilizat și oferă măsurători precise ale temperaturii și umidității, ceea ce îl face popular în proiecte electronice și IoT.

DHT11 utilizează un singur pin digital pentru comunicare și alimentare, ceea ce îl face ușor de integrat în proiecte. Acesta returnează datele sub formă digitală, ceea ce înseamnă că nu este necesară o conversie analog-digitală suplimentară. Senzorul este echipat cu un element senzorial care detectează temperatura și umiditatea din mediul înconjurător și transmite aceste date către placa de dezvoltare.

Un avantaj major al DHT11 este că este compatibil cu o varietate largă de platforme hardware, inclusiv Arduino, Raspberry Pi și altele. Acest lucru permite dezvoltatorilor să-l utilizeze în diferite proiecte fără a fi necesară o reconfigurare suplimentară.



1.4 BMP180 este un senzor de presiune și temperatură digital, utilizat pentru a măsura presiunea atmosferică și temperatura ambientală în diferite aplicații. Acest senzor oferă măsurători precise și stabile ale presiunii și temperaturii, ceea ce îl face util într-o varietate largă de proiecte, de la monitorizarea condițiilor meteorologice la navigație și altimetrie.

BMP180 utilizează un protocol de comunicare I2C pentru a transmite datele către placa de dezvoltare și poate fi conectat la o gamă largă de platforme hardware, inclusiv Arduino, Raspberry Pi și altele. Acest lucru face senzorul ușor de integrat în proiecte și permite dezvoltatorilor să îl utilizeze în diferite aplicații fără a fi necesară o reconfigurare suplimentară.

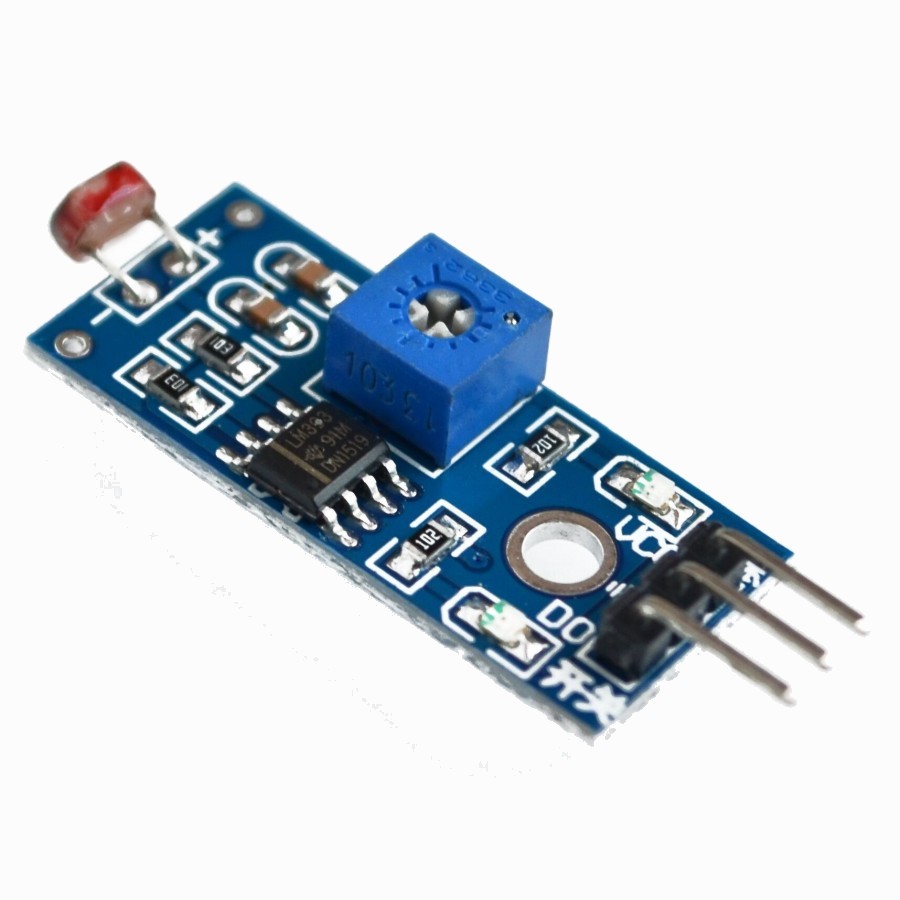
Un avantaj major al BMP180 este că oferă informații despre presiunea atmosferică, care poate fi utilizată pentru a estima altitudinea sau pentru a monitoriza schimbările de presiune în timp real. Acest lucru îl face util în aplicații precum sistemele de navigație, dronele, vremea și altele.



1.5 Un senzor de lumină este un dispozitiv utilizat pentru a detecta nivelul de lumină ambientală din mediul înconjurător și pentru a transmite aceste informații către placa de dezvoltare sau microcontrollerul asociat. Acesta este utilizat într-o varietate largă de aplicații, inclusiv sisteme de iluminat inteligent, sisteme de siguranță și supraveghere, și multe altele.

Senzorii de lumină pot utiliza diferite tehnologii pentru a detecta nivelul de lumină, inclusiv fotorezistențe (LDR), fotodiodă sau fototranzistori. Aceste dispozitive convertește lumina într-un semnal electric, care poate fi apoi interpretat de către placa de dezvoltare pentru a determina nivelul de lumină.

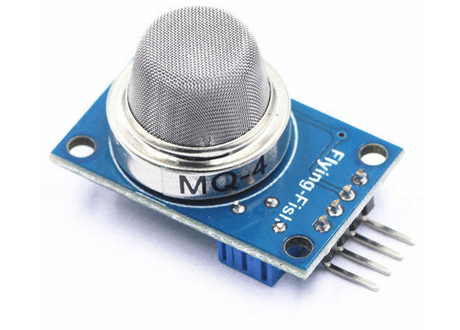
Un avantaj major al senzorilor de lumină este că oferă informații precise și în timp real despre nivelul de lumină ambientală, ceea ce permite sistemelor să reacționeze în mod automat la schimbările de lumină. Acest lucru este util în aplicații precum sistemele de iluminat inteligent, care ajustează luminozitatea în funcție de lumina ambientală pentru a economisi energie și pentru a crea un mediu confortabil.



1.6 Senzorul MQ-4 este un senzor de gaze care este utilizat pentru detectarea concentrațiilor de gaz metan (CH4) din mediul înconjurător. Acesta poate fi utilizat într-o varietate largă de aplicații, inclusiv în sistemele de siguranță și detectare a incendiilor, precum și în sistemele de monitorizare a calității aerului.

Principiul de funcționare al senzorului MQ-4 se bazează pe schimbările rezistenței electrice a elementului senzorial în prezența gazului metan. Senzorul conține un element senzorial care este expus la gazele din mediu. Atunci când gazul metan interacționează cu elementul senzorial, acesta determină o schimbare în rezistența electrică a senzorului, care poate fi măsurată și interpretată de către placa de dezvoltare asociată.

Un avantaj major al senzorului MQ-4 este capacitatea sa de a detecta gazul metan în concentrații mici, ceea ce îl face util în detectarea scurgerilor de gaz sau a altor situații de risc. Acest lucru îl face potrivit pentru utilizare în sistemele de siguranță și detectare a incendiilor, precum și în sistemele de monitorizare a calității aerului în spații închise.



1.7 Servo motorul este un tip special de motor electric, utilizat pentru controlul precis al poziției unui arbore de ieșire. Acesta este compus dintr-un motor electric, un set de angrenaje și un circuit de control, care împreună permit motorului să fie controlat cu precizie pentru a atinge o anumită poziție angulară.

Principiul de funcționare al unui servo motor se bazează pe feedback-ul de poziție, adică motorul primește un semnal de control și, în funcție de acesta, își ajustează poziția arborelui de ieșire. Acest lucru permite servo motorului să atingă poziții precise și să rămână stabil în aceste poziții, fără să fie necesară intervenția umană.

Un avantaj major al servo motorului este capacitatea sa de a oferi control precis asupra poziției și vitezei, ceea ce îl face ideal pentru o varietate largă de aplicații, inclusiv în robotică, aeromodele, sisteme de direcționare auto și multe altele. De asemenea, servo motorul este ușor de utilizat și de integrat în diferite proiecte, fiind disponibil într-o varietate de dimensiuni și specificații pentru a se potrivi nevoilor specifice ale fiecărui proiect.

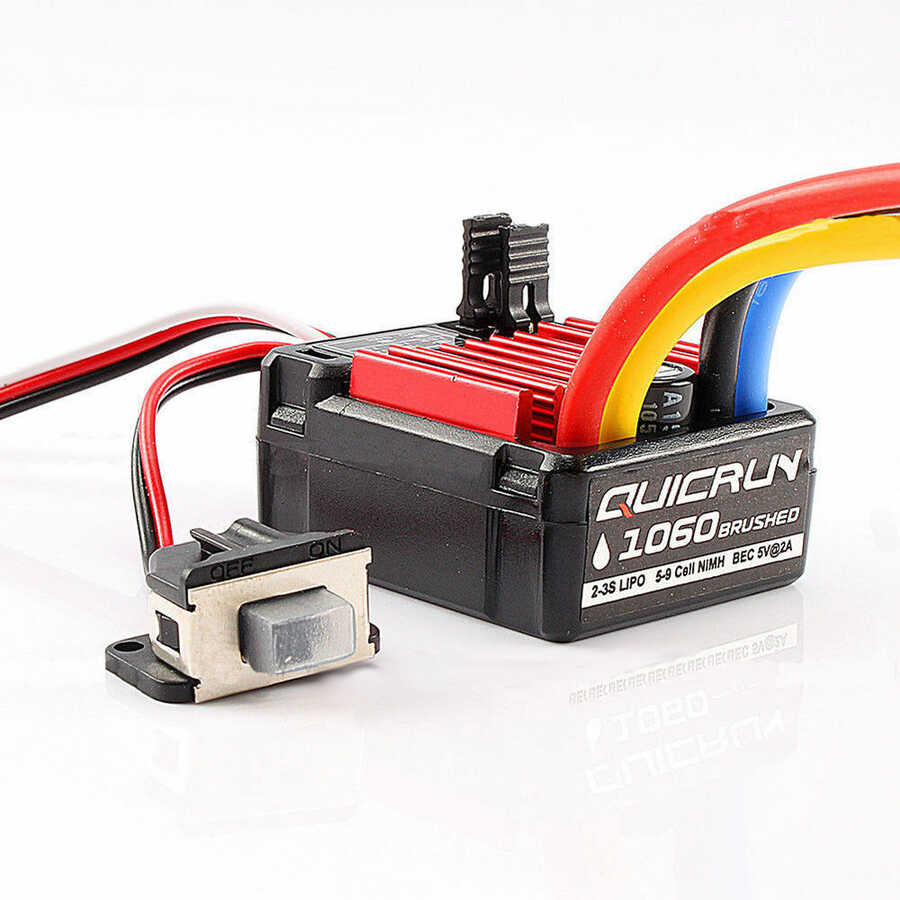


1.7 ESC (Electronic Speed Controller) este un dispozitiv electronic folosit pentru a controla viteza și direcția motoarelor electrice, în special în aplicații precum vehiculele RC (radio controlate). Funcționând ca o punte H electronică, ESC primește semnale de control de la un receptor radio și convertește aceste semnale în tensiuni și curenti adecvați pentru motoarele electrice, permițând controlul precis al vitezei și direcției vehiculului.

Principiul de funcționare al unui ESC implică conversia semnalelor de control de la receptorul radio în semnale PWM (Pulse Width Modulation), care controlează apoi curentul și tensiunea aplicată motoarelor electrice. Acest lucru permite ESC-ului să ajusteze viteza și direcția vehiculului în timp real, în funcție de semnalele primite de la telecomandă.

Un avantaj major al ESC-urilor este că oferă control precis și liniar al vitezei și direcției vehiculului, permițând utilizatorilor să controleze vehiculul în modul dorit. Acestea sunt disponibile într-o varietate de dimensiuni și specificații, pentru a se potrivi diferitelor tipuri de vehicule RC și aplicații.

ESC-urile sunt utilizate în mod obișnuit în vehiculele RC, precum mașini, avioane și elicoptere, dar sunt și utilizate în alte aplicații, cum ar fi dronele și vehiculele autonome. Datorită capacității lor de a oferi control precis și fiabilitate în funcționare, ESC-urile sunt un element esențial în lumea vehiculelor RC și a automatizării.



1.8 Raspberry Pi Camera este o cameră digitală de dimensiuni mici, special concepută pentru a fi utilizată cu plăcile Raspberry Pi. Aceasta oferă capacitatea de a captura imagini și video de înaltă calitate într-un format compact și ușor de utilizat. Camera este conectată direct la conectorul dedicat de pe placa Raspberry Pi, fără a fi necesară utilizarea de cabluri sau adaptatoare suplimentare.

Principiul de funcționare al camerei Raspberry Pi implică capturarea imaginilor și a videoclipurilor utilizând senzorul de imagine și procesorul de imagine integrate în cameră. Aceasta permite utilizatorilor să captureze imagini și videoclipuri de înaltă rezoluție, cu detalii precise și culori vii.

Un avantaj major al camerei Raspberry Pi este versatilitatea sa și ușurința de utilizare. Aceasta poate fi utilizată într-o varietate largă de aplicații, inclusiv în domeniul monitorizării și supravegherii, fotografiei și videografiei, și multe altele. De asemenea, camera Raspberry Pi este compatibilă cu o varietate de software și biblioteci, care permit utilizatorilor să realizeze diferite funcționalități, cum ar fi recunoașterea facială, detectarea de mișcare și altele.



### Proiectare software de detaliu

Pentru proiectarea software de detaliu a serverului de control al mașinii RC, vom detalia fiecare aspect al codului și funcționalității acestuia:

1. Identificator serviciu:

* Numele serviciului: RCControlService

2. Clasificare:

* Tipul de serviciu: Server pentru controlul și monitorizarea mașinii RC.

3. Definiție:

* Scopul serviciului: Asigurarea comunicației între aplicația client (dispozitivul mobil sau altă interfață de control) și mașina RC, gestionarea comenzilor și feedback-ului.

4. Cerințe:

* Acceptarea și gestionarea comenzilor de la client.
* Transmiterea comenzilor către mașină și primirea feedback-ului de la mașină.
* Salvarea datelor de la senzori în baza de date și furnizarea acestora către client la cerere.
* Cerințe nonfuncționale:
* Reacție rapidă la comenzi pentru a asigura un control precis al mașinii.
* Fiabilitate în comunicarea cu mașina și stocarea datelor în baza de date.

5. Structuri de date interne:

* Utilizarea MongoDB pentru stocarea datelor de la senzori.

6. Constraingeri:

* Limitări de performanță: Timpul de răspuns trebuie să fie sub 50 ms pentru a asigura un control precis al mașinii.
* Limitări de securitate: Autentificare și autorizare adecvată pentru accesul la serviciu.

7. Compoziție:

* Serviciul utilizează Express pentru gestionarea rutei HTTP, WebSocket pentru comunicare în timp real și Mongoose pentru interacțiunea cu baza de date MongoDB.

8. Utilizatori/Interacțiuni:

* Interacțiunea cu clientul prin intermediul WebSocket și HTTP.

9. Procesare:

* Procesarea comenzilor primite de la client și trimiterea acestora către mașină.
* Salvarea datelor de la senzori în baza de date și furnizarea acestora către client.

10. Interfețe/Exporturi:

* API-uri pentru comunicarea cu clientul.
* Interfețe grafice pentru afișarea datelor de la senzori.

A screen shot of a computer program

Description automatically generated

Configurare Server Express

A computer screen shot of text

Description automatically generated

Conectarea la baza de date Mongo

A computer screen shot of code

Description automatically generated

Trimiterea unui mesaj prin Socket

const DataPage: React.FC<Props> = ({*navigation*}) => {

  const sensorData = useSelector((*state*: RootState) => *state*.sensor);

  const [blinkAnimation, setBlinkAnimation] = useState(new Animated.Value(1));

  const data = [

    {

      text: 'Temperatura',

      image: require('../assets/temp100.png'),

      value: sensorData.temperatureValue,

      measureUnit: ' °C',

    },

    {

      text: 'Umiditate',

      image: require('../assets/hum1.png'),

      value: sensorData.humidity,

      measureUnit: ' %',

    },

    {

      text: 'Presiune',

      image: require('../assets/meter6.png'),

      value: sensorData.pressureValue,

      measureUnit: ' Pa',

    },

    {

      text: 'Altitudine',

      image: require('../assets/a.png'),

      value: sensorData.altitude,

      measureUnit: ' m',

    },

    {

      text: 'Gas Metan',

      image: require('../assets/m1.png'),

      value: sensorData.airValue,

      measureUnit: ' ppm',

    },

    {

      text: 'Distanta',

      image: require('../assets/ultra.png'),

      value: sensorData.humidity,

      measureUnit: ' cm',

    },

  ];

  const distance = 30

  useEffect(() => {

    const animation = Animated.timing(blinkAnimation, {

      toValue: distance < 15 ? 0.8 : distance >= 15 && distance < 30 ? 0.8 : 1,

      duration: 500,

      useNativeDriver: true,

    });

    animation.start();

*return* () => {

      animation.stop();

    };

  }, [distance]);

  const selectImage = () => {

*if* (distance < 15) {

*return* require("../assets/wifi2.png");

    } *else* *if* (distance >= 15 && distance < 30) {

*return* require("../assets/wifi1.png");

    } *else* {

*return* require("../assets/wifi.png");

    }

  };

*return* (

    <View style={styles.dataBackground}>

      <Text style={styles.liveSensorTitle}>Date Senzori</Text>

      <View style={styles.sensorsContainer1}>

        {data.map((*item*, *index*) => (

          <View key={*index*} style={styles.liveSensorBackground}>

            <View style={styles.sensorsContainer1}>

              <View style={styles.sensorItem}>

                <View>

                  <Image style={styles.tempImage} source={*item*.image} />

                </View>

                <View>

                  <Text style={styles.tempCardTitle}>{*item*.text}</Text>

                  <Text style={styles.tempCardText}>

                    {*item*.value}

                    {*item*.measureUnit}

                  </Text>

                </View>

              </View>

            </View>

          </View>

        ))}

      </View>

Componenta DataPage:

* Este o componentă funcțională React care primește propria navigare și afișează datele de la senzori.

Utilizarea useSelector:

* Este folosit din Redux pentru a selecta starea senzorului din starea globală a aplicației.

Starea blinkAnimation:

* Este o stare locală folosită pentru a controla o animație care va face să clipească un element în funcție de distanța detectată.

Datele senzorilor:

* Sunt definite într-un array data care conține informații despre diferitele tipuri de date de la senzori, precum temperatură, umiditate, presiune, altitudine etc.

Efectul useEffect:

* Este folosit pentru a declanșa și a opri o animație în funcție de distanța detectată.
* Efectul se activează și se oprește în funcție de schimbările în valoarea distance.

Funcția selectImage:

* Este o funcție auxiliară care selectează imaginea corespunzătoare în funcție de distanța detectată.
* Imaginea este selectată în funcție de valorile pragului de distanță (mai mic de 15, între 15 și 30, și mai mare sau egal cu 30).

Renderea componentelor:

* Componenta afișează datele senzorilor într-o listă, fiecare element al listei conținând o imagine, numele senzorului și valoarea sa.

Această componentă React realizează afișarea datelor de la senzori și aplică un efect de animație în funcție de distanța detectată. Ea folosește Redux pentru a accesa starea globală a aplicației și implementează logica necesară pentru a gestiona și afișa datele de la senzori în interfața utilizatorului.

### Proiectare detaliată a comunicațiilor interne (între componente)

Pentru proiectul meu, care implică comunicarea între mai multe componente, inclusiv serverul central, Raspberry Pi și aplicația mobilă, proiectarea detaliată a comunicațiilor interne trebuie să fie bine gândită pentru a asigura o funcționare eficientă a sistemului. Iată o descriere a proiectării comunicațiilor interne:

Protocol de comunicare:

* Stabilirea unui protocol de comunicare standardizat între serverul central, Raspberry Pi și aplicația mobilă pentru a asigura interoperabilitatea și compatibilitatea între componentele sistemului.
* Alegerea unui protocol eficient care să permită transmiterea datelor în timp real între aceste componente, cum ar fi protocolul WebSocket sau HTTP pentru comunicația între serverul central și aplicația mobilă, și protocolul MQTT pentru comunicația între Raspberry Pi și serverul central.

Topologie de rețea:

* Configurarea unei topologii de rețea care să faciliteze comunicarea eficientă între componentele sistemului, asigurându-se că toate dispozitivele sunt conectate la aceeași rețea locală.
* Utilizarea unei conexiuni cablate sau wireless între Raspberry Pi și serverul central, în funcție de disponibilitatea infrastructurii de rețea și cerințele de performanță.

Gestionarea conexiunilor:

* Implementarea unui mecanism de gestionare a conexiunilor pentru a asigura stabilitatea și fiabilitatea comunicațiilor între componentele sistemului, incluzând mecanisme de reconectare în cazul în care o conexiune este pierdută temporar.
* Monitorizarea stării conexiunilor și gestionarea erorilor în mod adecvat pentru a minimiza impactul asupra funcționării sistemului în cazul întreruperilor de rețea sau a altor probleme tehnice.

Securitatea comunicațiilor:

* Implementarea unor măsuri de securitate pentru protejarea datelor transmise între componentele sistemului, inclusiv criptarea datelor sensibile și autentificarea între serverul central și aplicația mobilă.
* Utilizarea unor mecanisme de autentificare și autorizare pentru a asigura că doar dispozitivele și utilizatorii autorizați au acces la datele și funcționalitățile sistemului.
* Prin proiectarea detaliată a comunicațiilor interne în acest mod, se va asigura o interacțiune eficientă și sigură între componentele sistemului, contribuind la funcționarea corectă și fiabilă a întregului sistem de control al RC carului.