  
**ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ**

**Факултет компютърни системи и технологии**

**Дипломна работа на тема**:

**СИСТЕМА ЗА АВТОМАТИЗИРАНО УПРАВЛЕНИЕ НА ПАРКИНГ**

|  |  |
| --- | --- |
| **София 2024** | **Дипломант:**  **Калоян Каменов Борисов КСИ, 121220159**  **Дипломен ръководител:**  **Ас. Иво Гергов** |

Съдържание

[Списък с фигури 5](#_Toc168074006)

[I. Увод 6](#_Toc168074007)

[II. Постановка на дипломната работа, цели и задачи. 7](#_Toc168074008)

[Мотивация 7](#_Toc168074009)

[Цели и задачи 7](#_Toc168074010)

[Ограничения 7](#_Toc168074011)

[Кратко описание на технологиите 8](#_Toc168074012)

[1.4.1 Python 8](#_Toc168074013)

[1.4.2 C# - .NET MAUI 8](#_Toc168074014)

[1.4.3 PostgreSQL 8](#_Toc168074015)

[1.4.4 SQLAlchemy 8](#_Toc168074016)

[1.4.5 FastAPI 9](#_Toc168074017)

[1.4.6 Uvicorn 9](#_Toc168074018)

[1.4.7 Разпознаване на обекти 9](#_Toc168074019)

[1.4.8 YOLOv8 9](#_Toc168074020)

[1.4.9 CVAT.AI 9](#_Toc168074021)

[1.4.10 Оптично Разпознаване на Символи 10](#_Toc168074022)

[1.4.11 Aiven 10](#_Toc168074023)

[1.4.12 Docker 10](#_Toc168074024)

[1.4.13 Kubernetes 10](#_Toc168074025)

[1.4.14 Google Cloud 10](#_Toc168074026)

[Разпознаване на обекти и машинно самообучение 10](#_Toc168074027)

[III. Функционално описание 13](#_Toc168074028)

[2.1 Общ Преглед на системата 13](#_Toc168074029)

[2.1.1 Описание на системата 13](#_Toc168074030)

[2.1.2 Работен процес 14](#_Toc168074031)

[2.2 Key Components ot espto and Their Functions 14](#_Toc168074032)

[2.3 Communication Protocol 14](#_Toc168074033)

[2.4 Data Collection and Processing 14](#_Toc168074034)

[2.5 User Interface 14](#_Toc168074035)

[2.6 Use Cases 14](#_Toc168074036)

[IV. Програмна реализация. 15](#_Toc168074037)

[3.1 комуникационна архитектура 15](#_Toc168074038)

[3.1.1 Комуникационна архитектура във възловата мрежа 15](#_Toc168074039)

[3.1.2 Комуникационна архитектура между потребителско устройство и възловата мрежа 15](#_Toc168074040)

[3.1.3 Комуникационна архитектура между възли за изчисляване на разстояние 15](#_Toc168074041)

[3.2 Системна архитектура на потребителското приложение 15](#_Toc168074042)

[3.3 Системна архитектура на възел 15](#_Toc168074043)

[V. Ръководство за използване и примери за употреба.  16](#_Toc168074044)

[4.1 Примерна постановка при наличие на Wi-Fi мрежа 16](#_Toc168074045)

[4.2 Примерна постановка без Wi-Fi мрежа 16](#_Toc168074046)

[4.3 Настройване на системата 16](#_Toc168074047)

[VI. Заключение 17](#_Toc168074048)

[Съкращения 18](#_Toc168074049)

[Източници 19](#_Toc168074050)

# Списък с фигури

1. [Layer Arcitecture 7](#_Toc167128811)

# I. Увод

Паркингът е част от нашето ежедневие, която е позната на всеки собственик на моторно превозно средство. Това е място, предназначено за временно пребиваване на моторното превозно средство (МПС). Паркингите обикновено не представляват особено технологична структура. Нормалният паркинг представлява голямо, отворено място, където да могат да се паркират МПС удобно. Обикновено има охранител, който се грижи превозните средства да са защитени от недоброжелатели. Охранителят също следи влизащите и излизащите автомобили, като също играе ролята на касиер, при когото се плаща престоят на МПС-то в паркинга.

Тази система може да работи при по-малки паркинги, където е лесно да се следят влизащите и излизащите автомобили. При тези по-малки паркинги, също не се създава натоварено движение на входа или изхода на паркинга. Не се случва често да е необходимо да се изчакват коли. В днешно време, много хора предпочитат да използват лични превозни средства, като автомобили, за да се придвижват от едно място до друго. Това създава растяща нужда от повече паркинги. Заради ограниченото място, не е добра опоция да се изграждат големи количества паркинги, а вместо това да има по-малко, но по-големи паркинги. Тези по-големи паркинги създават трудности при плащане и организация на вход и изход. Тук идва нужда за изграждане на система, която да позволява лесен достъп до паркинга и лесен изход от него. Забавянето при влизане или излизане трябва да бъде сведено до минимум.

За да се наблюдават превозните средства, които влизат и излизат от паркинга, е необходимо да се следят уникалните МПС. Това става лесно, тъй като всяко МПС има уникален регистрационен номер, който е поставен на видно място върху превозното средство. Интелигентната паркинг система може да използва система от камери, която да наблюдава автомобилите и да проверявва регистрационните им номера в база данни от регистрирани потребители и така да ограничава достъпът на неоторизирани лица. Използвайки тази система за достъп е възможно да разберем точният момент на вход и на изход на дадено МПС и да ги запишем в сървър. Така можем да позволим потребителите да плащат сметките си онлайн, без да имаме нужда от касиер на паркинга, който да бави процеса на изход от паркинга.

# II. Постановка на дипломната работа, цели и задачи.

## 2.1 Мотивация

В днешното забързано ежедневие, не е допустимо да съществуват толкова остарели и неудобни системи като паркингите, в това им състояние. Необходимо е да съществува система, която да улеснява потребителите, за да може всичко да се случва бързо и удобно. Не трябва да се налага да има опашки на входовете и изходите на паркингите, докато се чака да се отвори бариера или докато се чака да се прочете бар код, който да отвори изходът. Всички тези забавяния довеждат до трафик и изнервят всички участници в движението.

## 2.2 Цели и задачи

Основните цели и задачи на тази разработка са да адресират проблемите със забавянето при паркингите и да предоставят лесен и удобен начин на потребителите да взаимодействат с паркингите. Това ще се случи, чрез приложение за всяка широко използвана операционна система, което да предоставя потребителски интерфейс. Това приложение ще позволява да бъде регистриран потребител и след това, потребителят да може сам да добави автомобилът си. След като автомобил е добавен, автоматизираното управление на паркингът трябва да започне да отваря входът и изходът, когато регистрираният автомобил се доближи. Приложението трябва да предоставя на потребителите лесен начин за плащане на паркинга, така че всеки да може да заплати в удобно за него време. Паркингът ще използва система от камери, които да определят кога доближава автомобил и дали този автомобил е в списъкът на регистрираните автомобили.

## 2.3 Ограничения

За разработката на тази система е необоходимо да се съобразим с някои ограничения. Един от най-големите проблеми, който изниква при разпознаване на текст, чрез камера, е яснотата на изображението, заснето от камерата. Това изображение трябва да бъде ясно, за да може системата да определи текстът, записан върху регистрационният номер. Необходимо е камерите да бъдат оборудвани с поляризиращ филтър, който да елиминира отблясъци от светлина, както и зоната около камерата да бъде добре осветена, за да може автомобилите да бъдат разпознавани дори и нощем.

Друго ограничение е самият потребител. Тази система разчита на способността на потребителят да използва мобилно устройство или компютър. Ако потребителят няма възможност да използва такова устройство, той не би могъл да се регистрира в паркинг системата и следователно, той не би могъл да използва паркинг съоражението.

## 2.4 Кратко описание на технологиите

### 1.4.1 Python

Избраният език за реализация на голяма част от тази дипломна работа е Python. Езикът позволява лесна работа с голямо количество библиотеки и структури от данни. Това помага с бърза реализация на сложни системи, използвайки само няколко реда програмен код.

Езикът за програмиране има употреба при статистически проучвания, заради лесната работа със структури от данни. Също е предпочитан за използване при работа с изкуствен интелект, заради простият си синтаксис. Това прави програми, написани на този език, лесни за четене и поддръжка. Езикът предоставя гъвгави инструменти за работа, голямо количество от библиотеки и голямо общество от разработчици, които го поддържат.

### 1.4.2 C# - .NET MAUI

Езикът C# е използван за реализация на мобилно приложение, което да позволява достъп до данните от системата за паркиране. Използвана е структурата за програмиране Multi-platform App UI (MAUI), която от своя страна стъпва върху структурата .NET.

Това е популярна структура, която позволява разработка на приложения, които могат да работят върху различни операционни системи, използвайки един и същи код. Структурата позволява дадено приложение да работи върху Android, iOS, Windows и macOS, без да са необходими промени в кода. Тази структура е наследникът на Xamarin.Forms. Тя разширява функционалността на старата структура, като добавя увеличена производителност и по-лесни за разширяване приложения.

Структурата MAUI работи като унифицира интерфейсите за програмиране на приложения на Android, iOS, Windows и macOS и предоставя собствени интерфейси, които сами определят коя платформа трябва да използват. [1]

### 1.4.3 PostgreSQL

PostgreSQL е система за управление на релационни бази данни с отворен код. Системата използва и разширява езикът Structured Query Language (SQL) и предоставя функционалност за сигурно съхранение на данни. Системата за управление може да работи под всички често използвани операционни системи, предоставя голямо количество инструменти за улеснено управление или разширяване на функционалност и има широк кръг от потребители, което прави решаването на проблеми лесно. [2]

### 1.4.4 SQLAlchemy

SQLAlchemy е библиотека за Python, която позволява лесна работа с голямо количество системи за управление на релационни бази данни. Библиотеката предоставя унифициран начин за достъпване на данните в базите данни, което ни позволява лесно да сменим типа на релационната база данни, без да се налага да се променя кодът, който използва базата данни. Това прави тази библиотека много ценна при създаване на каквото и да е приложение, което работи с база данни.

За да се създаде връзка с база данни, е необходимо да има известни адрес на сървъра на базата, порт на сървъра на базата, потребителско име, парола и име на базата данни. След това библиотеката сама се грижи за тази връзка и ние просто трябва да създаваме сесии, с които да работим.

### 1.4.5 FastAPI

FastAPI представлява библиотека на Python, която позволява изграждане на интерфейси за достъп до сървър. Библиотеката се работи директно с езика Python, което значи, че потребителят просто трябва да посочи парче код, което иска да използва като интерфейс за достъп, и библиотеката FastAPI сама се грижи за изграждането на интерфейсa.

След като се опишат входните точки на интерфейса, FastAPI може да генерира стандартизиран документ, който описва всички интерфейси и данни, които приема приложението, познат като OpenAPI файл. Този файл се използва от клиентски приложения за да се генерира код, който да позволи връзка с нашето приложение.

### 1.4.6 Uvicorn

Uvicorn e уеб сървър за Python. Представлява имплементация на Asynchronous Server Gateway Interface (ASGI) спецификацията. Тази спецификация описва интерфейси за изграждане на асинхронни и синхронни сървъри на Python.

Uvicorn е напълно съвместим с OpenAPI спецификацията и следователно е съвместим с FastAPI. Така в едно приложение можем да използваме FastAPI за да създадем OpenAPI съвместими входни точки за сървърът ни и след това да пуснем Uvicorn, който да обслужва входните точки.

### 1.4.7 Разпознаване на обекти

Разпознаване на обекти или Object Detection представлява технология, при която се тренира невронна мрежа, която да разпознава дадени обекти. Тази технология е техника за ‚компютърно зрение‘, което буквано представлява способността на дадена машина да ‚вижда‘. Ще навлезем по-дълбоко в невронни мрежи и разпознаване по-късно в тази глава.

### 1.4.8 YOLOv8

YOLOv8 или You Only Look Once представлява модел за разпознаване на обекти на Ultralytics, който е бърз, точен и лесен за употреба. Моделът може да класифицира обекти, да разпознава обекти, да сегментира обекти, да проследява обекти и да определя пози. Моделът има няколко различни размери, като по-големите модели са по-точни, но и изискват повече ресурси за да работят. Най-лекият модел е nano, последван от small, последван от medium, след това large и най-големият модел x. [3]

### 1.4.9 CVAT.AI

Това е платформа за анотиране на данни. Анотацията представлява процес, при който потребител минава през сет от данни с цел да им даде някакво определение. В тази дипломна работа, са анотирани около 150 различни изображения на автомобили с регистрационни номера, за да бъдат използвани за обучение на модел за разпознаване. Платформата предоставя уеб интерфейс, който улеснява работата и множество инструменти, които покриват всякакви сценарии, с които можем да се сблъскаме при анотиране на данни. Платформата е безплатна за ползване при по-малки проекти.

### 1.4.10 Оптично Разпознаване на Символи

OCR или Оптично Разпознаване на Символи представлява технология, при която снимка на текст, било той ръчно написан или принтиран, се подава на конвертор, който превръща снимката в текст, разбираем от машина. Тази технология позволява автоматично въвеждане на данни от документи, което елиминира ръчното въвеждане и така увеличава скоростта на обработка на данни. Проблемът е, че снимката трябва да бъде ясна, за да може алгоритъмът да прочете данните коректно.

### 1.4.11 Aiven

Aiven е платформа за управление на данни. Платформата предоставя възможност данни да бъдат следени, оптимизирани и съхранявани. Предоставя безплатен план, който можем да използваме за да съхраняваме данните си в база данни, работеща с PostgreSQL. Необходимо е да бъде настроен сървър, което става лесно, чрез предоставен потребителски интерфейс.

### 1.4.12 Docker

Docker е инстртумент, използван за създаване на изображение на приложението, което се използва за управление на базата данни. Чрез това изображение, можем да създадем контейнер, който съдържа нашето приложение и необходимите му зависимости. Това ни позволява да изпълняваме нашето приложение независимо от операционната система, върхо която работи Docker. Инструментът е безплатен и е широко използван в индустрията.

### 1.4.13 Kubernetes

Kubernetes е система за оркестрация на контейнери. Тази система се грижи за работата на контейнерите, които са пуснати в нея. Управлява ресусрсите на компютърът, на който работи и осигурява непрекъсната работа на контейнери. Ако даден контейнер падне, Kubernetes трябва да го върне в работно състояние. Този инструмент е доказал полезността си и се използва в най-големите компании.

### 1.4.14 Google Cloud

Това е решението на Google за лесно създаване и управление на проекти в облака. Платформата предоставя потребителски интерфейс за управление на приложение и централизирано място, от където да се наблюдават всички включени системи. Има голямо количество разширения, които можем да използваме за да развием нашето приложение. Платформата предоставя безплатен план, който може да бъде използван за малки приложения. За нашите цели, ще използваме Google Cloud за да управляваме Kubernetes cluster, в който да работи нашият контейнер, който съдържа софтуерът за връзка с базата данни.

## 2.5 Разпознаване на обекти и машинно самообучение

Машинното самообучение представлява разработка, при която на даден алгоритъм се предоставят познати данни, от които да се самообучи, за да може да разпознава тези данни, при непознати ситуации.

Алгоритъмът за самообучение представлява мрежа от параметри, които много наподобяват неврони. Тези невронни мрежи имат променливо количество параметри, като те са разположени на нива. Входното ниво представлява данните, които подаваме. След това съществуват даден брой скрити нива, където се случва обработката на данните. Скритите нива са последвани от изходно ниво, където виждаме какъв е изходът от невронната мрежа. Невроните на различните нива са свързани с невроните от предишното и следващото ниво, като силата на връзката определя как би се активирал невронът. При слаба връзка между два неврона, силна активация на един неврон ще доведе до слаба активация в друг неврон. Тренирането на невронната мрежа представлява точно определянето на силата на връзките между нивата. Подават се данни на входа и се описва желаният изход. Задачата на невронната мрежа е да обработи входните данни и да се доближи максимално до желаният изход.

При трениране на алгоритъм е необходимо голямо количество разнообразни входни данни, за да може алгоритъмът да бъде генерализиран за типа данни. Ако на даден алгоритъм се подадат малко количество данни, той ще се научи само на няколко конкретни случая и няма да бъде добър, когато му бъдат подадени данни, които не е виждал до сега. Също, заради начинът по който невронната мрежа определя теглата между нивата, е необходимо неколкократно изпълнение на тренировъчния процес. При първо преминаване през данните, е възможно връзките между нивата да не бъдат достатъчно ясно изразени, което да доведе до несигурност в изходните данни. Изпълнението на тренировъчният процес се нарича епоха. Препоръчва се да бъдат използвани няколко тренировъчни епохи, като техният брой не може да бъде определен универсално, заради всички разлики при обучението на различни модели. Оптималните епохи обикновено се определят на прицнипът проба-грешка.

Тренировката на мрежата се състои от две части: същинското трениране и валидацията. Вече описахме какво представлява тренирането на мрежата. След като приключи етапът на трениране, следва етап на валидиране. В този етап невронната мрежа трябва да премине през данни, които ние сме подготвили предварително, но този път без да знае какъв изход трябва да получи. Ние, разбира се, знаем какъв трябва да бъде изходът и след като мрежата премине данните, трябва да сравним изходът на алгоритъма с очакваният от нас изход. Ако резултатът, който сме получили е задоволителен, значи мрежата е готова за използване. Ако видим разлики между очакваният резултат и полученият резултат, следва да тренираме мрежата отново. При последващи тренирания на мрежата, трябва да бъдат направени промени, за да не получим същият изход отново. Какви ще бъдат тези промени зависи изцяло от какви проблеми сме видели при изходните данни. Ако, например, много от тестовите ни данни не са получили резултат изобщо, е възможно мрежата да не е била тренирана достатъчно пъти. Следва да увеличим броят епохи за следващо трениране. Съществува друг вариант, при който мрежата ни връща резултат, но той е погрешен. Този проблем е по-сложен за решаване, тъй като той може да бъде породен от много различни причини. Една от най-често срещаните причини е проблем с входните данни. Възможно е да не е определен коректно изходът от алгоритъма или данните да не са напълно ясни, което може да доведе до изкривяване на очакванията. В тази ситуация алгоритъмът работи правилно, но данните, които предоставяме, не са верни, поради човешка грешка. Другият вариант е, когато данните ни са коректни, но изходът все още да не е верен, е когато невроните в мрежата не са достатъчно или не са разпределени между нивата коректно. Неправилното разположение на невроните и връзките между тях може да доведе до сериозни изкривявания в изхода, тъй като всяко следващо ниво представлява филтриран резултат от предишно. За нещастие, няма готова рецепта, която да определи колко неврони трябва да имаме на всяко от вътрешните нива или колко вътрешни нива трябва да имаме. Всяка невронна мрежа е различна и изисква да бъде настроена според конкретните си нужди. Това, обикновено, се случчва на принципа проба-грешка. От добрата страна, е доста лесно да настроим входните и изходните неврони, тъй като те се определят директно от нашите данни. Ако, например, имаме алгоритъм, на който подаваме изображение на цифра, и искаме да разпознае каква е тази цифра, то входът ни би имал по един неврон за всеки пиксел от изображението. Този вход, трябва да ни даде цифра от 0 до 9, следователно изходните неврони трябва да бъдат 10 на брой, по един за всеки възможен изход. Вътрешните неврони трябва да бъдат нагодени, за да получим оптимален резултат с нашите данни.

При разпознаването на обекти се ползва много подобен подход. Трябва да създадем сет от данни, който да представя обектите, които търсим. Колкото е по-голям сетът, толкова по-добре ще бъде обучена мрежата. Важно е сетът да е не-само голям, а и да бъде разнообразен. Трябва да предоставим ясни изображения, които да са добре анотирани. По този начин, чрез разнообрази и анотирани изображения, ще получим по-генерализиран модел, който ще може да разпознава обекти по-добре. След това трябва да тренираме невронната мрежа, чрез тези данни, подобно на примера с цифрите, но този път по-сложно.

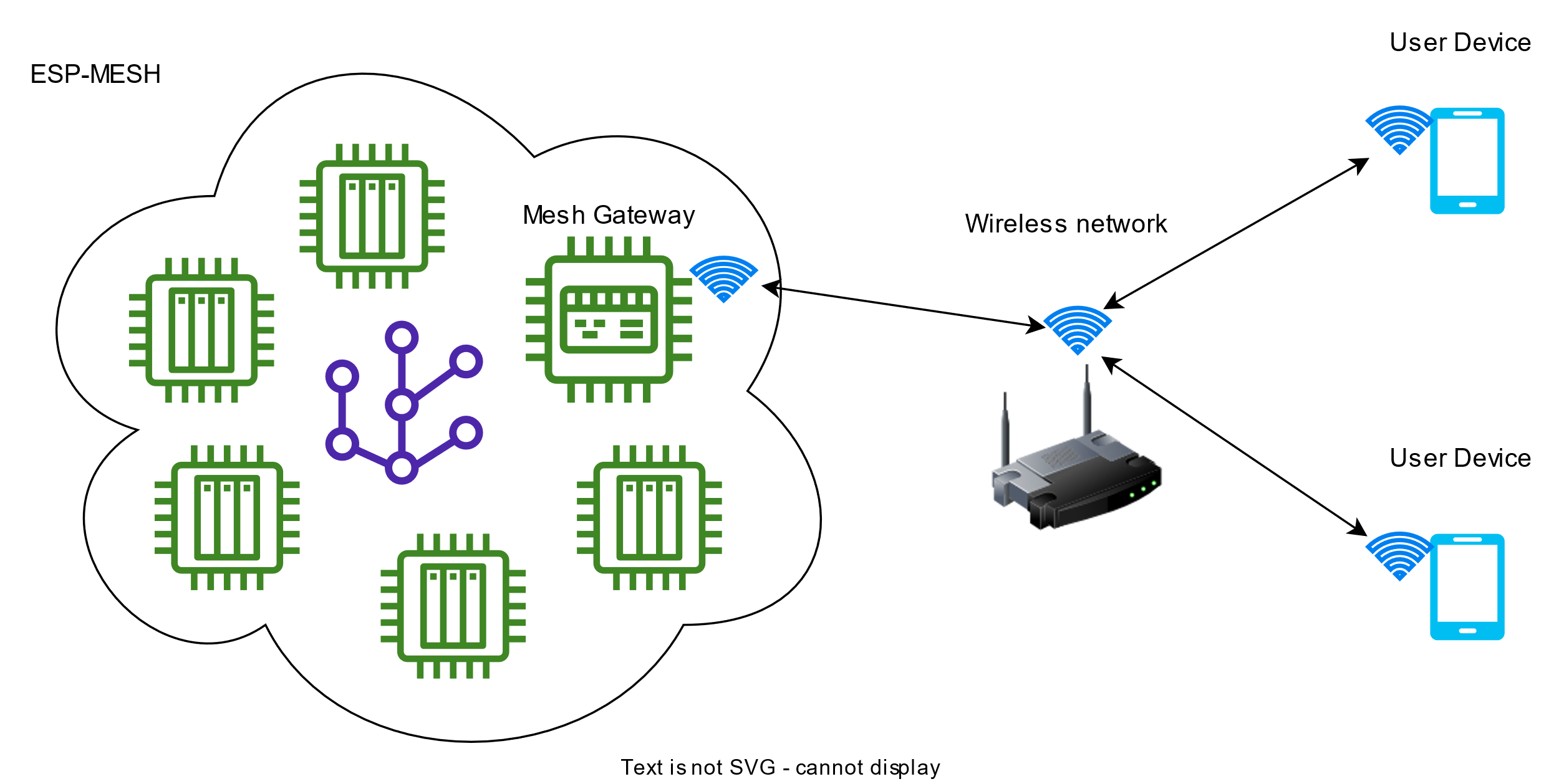
При разпознаването на обекти имаме конкретен очакван изход, но входът не е толкова изчистен, както би бил при разпознаване на единични цифри. Разпознаването на обекти предполага, че обектът се намира в дадена среда и не е единственото нещо в изображението. За да може нашият алгоритъм да разпознава обектите ефективно, независимо от средата, той трябва да може да разпознава единични характерни черти на обектите и чрез тях да може да разпознае самият обект. Това работи добре, когато обектите са ясно изразени и видиими добре. При замъглени обекти, където характеристиките им не са ясно отличени, може да има проблем, при който не може да бъде разпознат обекта. Заради начинът, по който работят моделите за разпознаване на обекти, те могат да бъдат обучени да разпознават много различни обекти, чрез индивидуалните им черти. В едно голямо изображение, можем да имаме, например, три различни обекта, върху които да е обучен нашият модел и той да успее да ги разпознае по характерните им черти. След като обектите са разпознати, следващата задача на модела за разпознаване на обекти е да определи къде в изображението се намират разпознатите обекти. При по-простите модели, това се случва с намиране на координатите на изображението, в който е разположен обектът, и определяне на правоъгълник, който обхваща дадения обект. При по-сложните модели, може да бъде определен конкретният силует на обекта, точните му граници и позицията му в изображението.

# III. Функционално описание

## 2.1 Общ Преглед на системата

### 2.1.1 Поглед отгоре върху цялата система

Систечата за измерване на разстояние е съставена от 2 основни, независими, елемента. Това са потребителстото приложение и мрежата за измерване, които си говорят по TCP канал. Мрежата за измерване е съставена от неопределен брой устройства, които си общуват по един от два начина – чрез ESP-MESH за системни съобщения и през ESP-NOW за съобщения, съврани с измерването на разстоянието. Едно от Esp32 устройствата се повишава автоматично в корен на мрежата, който се грижи за предаване на комуникацията между мрежата и клиентските устройства. Този корен изпълнява функциите едновременно на шлюз (Gateway) и на нормален възел в мрежата. Компонентът от кода, който се занимава с нормалното функциониране на устройството като възел, не знае и не се интересува кое устройство е коренът – това на което се изпълнява кодът на сегашното устройство или някое друго устройстово в мрежата. На Фигура 1 е показан опростен поглед на системата.



Фигура 1

На фигурата е показана типична постановка на системата за измервания на разстояния, където има предварително установена мрежа от шест устройства. Едно от устройствата изпълнява ролята на шлюз, който е свързан към точка за достъп. Към същата точка за достъп са свързани две потребителски устройства, които могат да си говорят с мрежата.

### **2**.1.2 Взаимодействие между услугите

Работният процес на системата започва при включване на устройствата, които съсдават мрежата за измерване на разстояния. Първата стъпка е инициализилането на устроствата и започването на договарянето за създаването на мрежата. Създаването на мрежата е вградена част от протокола ESP-MESH и процедурите по създавене и подръжка са подробно описани в сайта на производителя. Тъй като, този протокол се нуждае от това да се върже към точка за достъп (access point), ако устройствата не успеят да се свържат към такава, то те започват работа като такава, което улеснява работата със систмата в случаите където няма точки за достъп. Втората стъпка е вписването на клиентското устройство в същата точка като мрежта за измерване и включването на клиентското приложение. След като клиентът установи връзка с мрежата за измерване, той може да изпраща съобщения към мрежата. Всички съобщения между мрежата и клиента минават през шлюза, като той има грижата да разпределя и препраща съобщенията и командите до всяко заинтересовано устройство, било то потребителско или участник в мрежата за измервания.

## 2.2 Key Components ot espto and Their Functions

## 2.3 Communication Protocol

* Explain the communication protocol used for sending and receiving Wi-Fi messages. This might include details on the frequency of messages, handling of data packets, and any error correction mechanisms.

## 2.4 Data Collection and Processing

* **Data Collection**: Describe how data is collected from the Wi-Fi messages, including any preprocessing steps.
* **Statistical Analysis**: Explain the statistical methods used to analyze the collected data and calculate the most likely distance.
* **Accuracy and Calibration**: Discuss any techniques used to improve accuracy, such as calibration procedures or filtering algorithms.

## 2.5 User Interface

* Describe the design and functionality of the user interface. How does it display the distance measurements, and what options does it provide to the user for interacting with the system?

## 2.6 Use Cases

* Provide a brief description of the main use cases that will be detailed later in Chapter 4. This serves as a preview for the reader.

# IV. Програмна реализация.

## 3.1 комуникационна архитектура

### 3.1.1 Комуникационна архитектура във възловата мрежа

### 3.1.2 Комуникационна архитектура между потребителско устройство и възловата мрежа

### 3.1.3 Комуникационна архитектура между възли за изчисляване на разстояние

## 3.2 Системна архитектура на потребителското приложение

## Системна архитектура на възел

# V. Ръководство за използване и примери за употреба.

## 4.1 Примерна постановка при наличие на Wi-Fi мрежа

## 4.2 Примерна постановка без Wi-Fi мрежа

## Настройване на системата

# VI. Заключение

# Съкращения

8

ASGI - Asynchronous Server Gateway Interface 7

MAUI - Multi-platform App UI 6

YOLO - You Only Look Once 8

МПС - Моторни Превозни Средства 5

# Източници

|  |  |
| --- | --- |
| [1] | "What is MAUI," Microsoft, [Online]. Available: https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui?view=net-maui-8.0. |
| [2] | "About PostgreSQL," PostgreSQL. [Online]. |
| [3] | "YOLOv8," Ultralytics, [Online]. Available: https://github.com/ultralytics/ultralytics. |