FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Bakalářská práce

Automatický systém sběru dat pro dopravní průzkum

Studijní program: B0613A140005 – Informační technologie
Studijní obor: B0613A140005AI – Aplikovaná informatika
Autor práce: **Filip Král**
Vedoucí práce: Ing. Jana Kolaja Ehlerová PhD.

Liberec 2024

FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Zadání bakalářské práce

Automatický systém sběru dat pro dopravní průzkum

Jméno a příjmení:	**Filip Král**
Osobní číslo:	M21000091
Studijní program:	B0613A140005 Informační technologie
Specializace:	Aplikovaná informatika
Zadávající katedra:	Ústav nových technologií a aplikované informatiky
Akademický rok:	2023/2024

Zásady pro vypracování:

1. Proveďte rešerši systémů pro rozpoznávání obrazu s důrazem na rozpoznání dopravních značek.
2. Navrhněte zařízení pro sběr geolokalizovaných snímků a jeho konektivitu.
3. Implementujte automatický přenos dat na server s rozpoznáváním obrazu.
4. Zajistěte zobrazení dat na webovém rozhraní s možností statistického zhodnocení.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	30 – 40 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	čeština

Seznam odborné literatury:

[1] ŠONKA, Milan, Václav HLAVÁČ a Roger BOYLE. *Image processing, analysis, and machine vision*. Fourth Edition. Australia: Cengage Learning, [2015]. ISBN 978-1-133-59369-0.

[2] ZHAO, JD, ZM Bai and HB Chen. Research on Road Traffic Sign Recognition Based on Video Image. 10th Internation Conference on Intelligent Computation Technology and Automation (ICICTA 2017) [2017], pp.110-113. DOI 10.1109/ICICTA.2017.31

[3] CANNON, Jason: Linux Administration: The Linux Operating System and Command Line Guide for Linux Administrators. CreateSpace Independent Publishing Platform [2016]. ISBN 1523915951, 9781523915958.

Vedoucí práce:	Ing. Jana Kolaja Ehlerová, Ph.D. Ústav nových technologií a aplikované informatiky
Datum zadání práce:	12. října 2023
Předpokládaný termín odevzdání:	14. května 2024

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Josef Chaloupka, Ph.D.
garant studijního programu

V Liberci dne 19. října 2023

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.
Jsem si vědom toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.
Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.
Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.
Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS STAG se shoduje s textem tištěné podoby práce.
Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.
Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

8. 5. 2024 Filip Král

Automatický systém sběru dat pro dopravní průzkum

Abstrakt

Tato bakalářská práce popisuje návrh a implementaci, která se zabývá vytvořením a následným testováním systému automatického sběru dat pro dopravní průzkum. Cílem této práce bylo navržení architektury pro detekci a klasifikaci dopravních značek za pomocí Raspberry Pi 4 a serverového prostoru. Pro řešení problému byla vybrána detekce a klasifikace pomocí neuronových sítí, které fungují jako detekční a klasifikační skripty na straně serverové. Nakonec byla architektura systému navrhnuta jako sběr snímků a zapisování do jejich metadat u Raspberry v jednom kroku, ve druhém kroku jako detekce, klasifikace a celkové vyhodnocení na straně serverové. Řešení je automatizované za pomocí automatického odesílání pořízených snímků z Raspberry na server. Automatizace detekce a klasifikace na serverové části je připravena, ale při testování nepoužita. Výstupem je webová aplikace, která graficky zobrazuje statistické vyhodnocení. Jako programovací jazyk pro tuto práci byl použit Python.

Klíčová slova: Detekce, Klasifikace, Dopravní značky, Model neuronové sítě, Webová aplikace, Raspberry Pi 4B, Trénování, GPS, EfficientDet, TensorFlow

Automatic data collection system for traffic surveys

Abstract

This bachelor thesis describes the design and implementation that deals with the creation and subsequent testing of an automatic data collection system for traffic surveys. The aim of this work was to design an architecture for traffic sign detection and classification using Raspberry Pi 4 and server space. Neural network detection and classification was chosen to solve the problem by using neural networks to act as server-side detection and classification scripts. Finally, the architecture of the system was designed as image collection and writing to its metadata at the Raspberry in one step, in the second step as detection, classification and overall evaluation at the server side. The solution is automated by automatically sending the captured images from the Raspberry to the server. Automation of detection and classification on the server side is ready but not used in testing. The output is a web application that graphically displays the statistical evaluation. Python was used as the programming language for this thesis.

Keywords: Detection, Classification, Traffic signs, Neural network model, Web application, Raspberry Pi 4B, Training, GPS, EfficientDet, TensorFlow

Poděkování

Za vznik této bakalářské práce bych rád poděkoval své vedoucí za vedení práce a za napomáhání při řešení vzniklých problémů. Dále bych také rád poděkoval svým blízkým za podporu při studiu.

Obsah

Seznam zkratek

COCO	Common Objects in Context
FPS	Frames Per Second
FTP	File Transfer Protocol
GTSRB	German Traffic Sign Recognition Benchmark
GPIO	General Purpose Input/Output
HAT	Hardware Attached on Top
IoT	Internet of Things
SFTP	SSH File Transfer Protocol
SSH	Secure Shell
VSFTPD	Very Secure FTP Daemon

Úvod

Cílem této bakalářské práce je navrhnout a implementovat automatický systém sběru dat pro dopravní průzkum.

Automatické čtení dopravních značek představuje inovativní a vysoce relevantní technologii v oblasti dopravního managementu a bezpečnosti silničního provozu. Díky pokroku v oblasti počítačového vidění a umělé inteligence je dnes možné vyvinout sofistikované systémy, které jsou schopny rozpoznávat a interpretovat různé typy dopravních značek s vysokou přesností a efektivitou.

Pro vytvoření softwaru byl zvolen programovací jazyk Python díky svému velkému pokrytí knihoven pro strojové učení. Pro vytvoření neuronových sítí a k natrénování jejich modelů je potřeba velkého objemu dat. V případě této bakalářské práce byly použity 3 veřejně dostupné datasety. Dataset GTSRB, který obsahuje 43 dopravních značek, plus další dataset, ze kterého bylo použito 19 druhů dopravních značek, byl použit pro natrénování klasifikátoru, který určuje druh dopravní značky. A COCO dataset byl použit k natrénování modelu detekce dopravních značek v obraze. Pro detekci byl zvolen detektor z rodiny detektorů EfficientDet a klasifikátor byl vytvořen za pomocí knihovny TensorFlow.

Pro implementaci a testování byl použit jednodeskový počítač Raspberry Pi 4 s připojenou Arducam kamerou a připojeným GPS modulem značky Waveshare. Bylo také využito serverového prostoru, na kterém funguje detekce a následná klasifikace. Serverový prostor také poskytuje Mongo databázi pro ukládání dat o klasifikované značce.

Výstupem z celého dopravního průzkumu je webová aplikace využívající framework Flask. Webová aplikace využívá grafického zobrazení dat pro graf a pro informační mapu s body výskytů dopravních značek.

1 Vypracované návrhy řešení

První studie byla vytvořena v roce 2017 v Číně. Autoři v úvodu uvádějí problematiku zhušťování dopravy a vyšší pravděpodobnosti výskytu dopravních nehod. Tato studie se zamýšlí nad přístupem pro detekci a klasifikaci dopravních značek ve videozáznamu.

V prvních částech studie autoři popisují, jak při detekci dopravních značek postupovali. Nejprve je zde krok předzpracování obrazu. To znamená proces snížení velikosti obrazu, úpravu jasu a filtraci obrazu. Tyto úkony jsou důležité pro odstranění šumu a pro zlepšení přesnosti v následující segmentaci. Autoři zde použili dva přístupy, segmentaci podle barvy dopravní značky, kdy se snímek převede do formátu HSV, následují morfologické operace a filtrace, které pomáhají oddělit značku od jejího pozadí. Druhý přístup je segmentace tvarových dopravních značek, kdy se používají záchytné body, které definují tvar. Následovala extrakce příznaků histogramu orientovaných gradientů, která pomohla, společně s lineárním jádrem SVM, k vytvoření efektivního klasifikátoru.

Závěrem autoři uvádějí, že jimi vytvořený klasifikátor dosahoval identifikace s 90% úspěšností. [20]

Další práce byla vytvořena v roce 2021 na VUT v Brně studentem fakulty elektrotechniky a komunikačních technologií na ústavu radioelektroniky panem Michalem Sichou. Bakalářská práce se zabývá detekcí a následným rozpoznáváním dopravních značek v reálném čase. Tento systém byl implementován na různých zařízeních. Dopodrobna o implementaci se bude psát v kapitole 8 na stránce 30, ta je pro tuto práci nejdůležitější.

V první části autor popisuje, jaké dopravní značení se v České republice používají, jen to pouze úvod do dopravního značení, které je snad každému z nás přinejmenším zevrubně známo. V podčásti první části práce autor popisuje datasety, které se mohou pro tento druh systémů použít. Budou zmíněny dva, které použil jak autor práce, tak i byly použity v této práci, protože jsou tyto datasety nejvíce využívané

ve světě, GTSRB a COCO. GTSRB je dataset pro natrénování modelu pro klasifikaci značek. Je zde přítomno přes 30 tisíc snímků výřezů dopravních značek ve 43 skupinách, jako jsou například značka hlavní silnice, dej přednost v jízdě nebo zákazu vjezdu. Druhý dataset se používá k detekci dopravních značek v obrazu.

V druhé části autor dokumentuje, na jakém hardwaru tento systém testoval. Jednalo se o Jetson Nano od společnosti nVidia. Jetson Nano je malý počítač, který se využívá také k vývoji umělých inteligencí. Nejčastěji se využívá pro klasifikaci obrázků, detekci objektů nebo zpracovávání řeči. Z výsledků práce vyplývá, že pro tento systém byl Jetson Nano více méně vyhodnocen jako nedostačující. Proto autor využil pro výpočet i grafickou kartu nVidia Geforce GTX 1050 Ti, která si při výpočtu vedla značně lépe. Nicméně ani grafická karta nebyla moc dostačující. Respektive, počet snímků, který zvládla karta zpracovat a vyhodnotit by se mohl za určitých podmínek považovat za minimálně přípustný.

V další části autor popisoval, jaké metody se využívají pro detekci objektů v obrazech, a také dopodrobna popsal princip a fungování neuronových sítí. Co se týče rozpoznávání dopravních značek autor zmiňuje, že využil detektor objektů zvaný EfficientDet, který byl pro tuto práci použit také. O této neuronové síti bylo hovořeno v předchozí části.

Zbytek kapitol byl shrnut do jednoho odstavce. Pro klasifikaci autor využil knihovnu TensorFlow, pro kterou natrénoval klasifikační model z datasetu GTSRB. Také popisuje celkovou jeho implementaci systému, jaký zvolil postup a jaké měl výsledky z vytváření modelů neuronových sítí pro detekci a klasifikaci. A nakonec závěr. V závěru autor zmiňuje, jaké výsledné hodnoty měl jeho systém implementovaný na dvou již výše zmíněných zařízeních. Na počítači Jetson Nano měl systém výsledek 0,35 FPS a pro výpočet na grafické kartě 5-6 FPS. Zmiňuje se také o tom, jaká opatření by se musela zavést, kdybychom chtěli nadále využít počítač Jetson Nano a jaký by mohl být potenciálně další vývoj práce pro tento pro tento typ řešení. [9]

2 Dopravní značky

Dopravní značky jsou jednoduché obrazové symboly určené k řízení a regulaci provozu na pozemních komunikacích. Dopravní značky jsou prostředky, které upozorňují účastníky silničního provozu na nebezpečná místa nebo události, upozorňují na zákazy, příkazy nebo omezení, nebo jsou kombinací jednoduchých obrazových symbolů nebo značek a barevných tabulí, které vysvětlují, doplňují nebo omezují význam jiné dopravní značky. Význam dopravních značek je obvykle definován předpisy o silničním provozu.

Dopravní značky jsou jedním z nejpraktičtějších globálních systémů vizuální komunikace. Na rozdíl od jiných systémů nejsou dopravní značky dosud úplně mezinárodně harmonizovány, a proto se liší nejen formou, ale i kvalitou.

2.1 Historie dopravních značek

Ve starověkých Pompejích byly nalezeny sloupky oddělující náměstí od vozovky, vyvýšené chodníky na vozovce a vyvýšené přechody připomínající nejmodernější konstrukce přechodů pro chodce.

Kolem roku 120 př. n. l. začali Římané kolem silnic umisťovat milníky, které označovaly vzdálenost do Říma (odtud rčení "všechny cesty vedou do Říma"). Po pádu Římské říše byly milníky zavedeny v 17. a 18. století Augustem II. Silným, vládcem Polska a Saska. Kolem roku 1700 zavedl ruský král Petr Veliký takzvanou verstu - jednotku délky. Kolem roku 1750 se na křižovatkách v Německu začaly objevovat dřevěné "křížové značky"; po roce 1789 byly tyto značky nahrazeny "hodinovými kameny", na nichž se vzdálenosti označovaly v hodinách.

V moderní době se signalizace pro řízení železniční a silniční dopravy inspirovala námořními signály: v roce 1868 bylo zaznamenáno použití primitivního mechanického semaforu na křižovatce v Londýně. Tyto dopravní signály měly tvar kříže a připomínaly postavu s rozpaženýma rukama - „STŮJ" nebo se svěšenýma rukama

- „volno”.

V roce 1903 vstoupil ve Spojeném království v platnost zákon o motorových vozidlech (3 Edw. VII, c. 30) a na světě se začaly používat první dopravní značky ve tvaru kruhů a trojúhelníků.

V prvním desetiletí 20. století zavedly první dopravní značky v jednotlivých zemích národní automobilové kluby, například Italský turistický klub a Kaiserlicher Automobile Club v Německu. V roce 1920 se v Nizozemsku konala dopravní konference, na níž se místní turistická organizace ANWB, pochlubila, že instalovala 400 výstražných značek v mezinárodně uznávaných tvarech a barvách.

Dopravní značení bylo v Československu zavedeno mnohem později. V roce 1935 bylo instalováno prvních pět typů výstražných značek, o tři roky později následovaly další druhy značek pro různé účely. V té době se již uvažovalo o prosvětlovaných značkách. Protože v té době ještě nebyla k dispozici retroreflexní fólie, byly některé značky vybaveny malými kulatými odrazkami. S traťovými značkami podobného provedení se můžeme na českých železnicích setkat dodnes. [17]

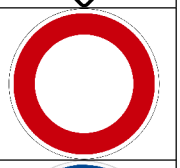
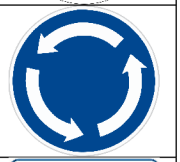
2.2 Kategorie dopravních značek

Důležitý vývoj pro existenci dopravních značek se stala v roce 1968 takzvaná „Vídeňská úmluva o dopravních značkách a signálech”, která vešla v plnou platnost v roce 1978. Tato mezinárodní dohoda kategorizuje a sjednocuje vzhled a význam dopravních značek, které se vyskytují v zemích, které z této úmluvy, při vytváření pravidel pro dopravní provoz, vycházejí. Kategoricky se dopravní značky v úmluvě řadí podle písmen A-H. [19]

- A – Varovné značky
- B – Značky přednosti v jízdě
- C – Zákazové značky
- D – Příkazové značky
- E – Zvláštní předpisové značky
- F – Informační a oznamovací značky
- G – Směrové, poziční a lokační značky

- H – Doplňující tabulky

V následující tabulce jsou znázorněné ukázky dopravních značek pro každou kategorii. Všechny uvedené značky jsou z prostředí českých dopravních komunikací.

Kategorie	**Značka**
Varovné značky	
Značky přednosti v jízdě	
Zákazové značky	
Příkazové značky	
Zvláštní předpisové značky	
Informační a oznamovací značky	
Směrové, poziční a lokační značky	
Doplňkové tabulky	

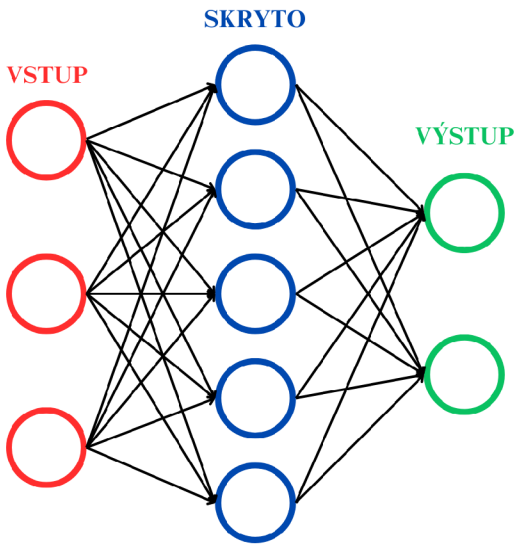
Tabulka 2.1: Kategorie značek. Obkresleno z Wikipedia. [1]

[1] https://cs.wikipedia.org/wiki/Seznam_dopravních_značek_v_Česku

3 Neuronové sítě

Neuronová síť je program nebo model strojového učení, který rozhoduje podobně jako lidský mozek, používaje procesy, které napodobují způsob, jakým biologické neurony spolupracují při identifikaci jevů, vážení možností a dospívání k závěrům.

Každá neuronová síť se skládá z vrstev uzlů neboli umělých neuronů – vstupní vrstvy, jedné či více skrytých vrstev a výstupní vrstvy. Každý uzel je propojen s ostatními a má svou váhu a prahovou hodnotu. Pokud je výstupní hodnota jednotlivého uzlu nad danou prahovou hodnotou, je tento uzel aktivován, což znamená, že data jsou odeslána do další vrstvy sítě.



Obrázek 3.1: Zjednodušená ukázka funkčnosti neuronové sítě

Každý uzel lze představit jako svůj vlastní model lineární regrese se vstupními daty, vahami, prahovou hodnotou a výstupem. Váhy přiřazené každému uzlu určují důležitost jednotlivé proměnné, přičemž větší váhy mají větší vliv na výstup ve srovnání s ostatními vstupy. Všechny vstupy jsou následně násobeny svými odpo-

vídajícími váhami a sečteny. Výsledek je poté proveden aktivizační funkcí. Pokud výstup překročí danou prahovou hodnotu, uzel je aktivován a data jsou poslána do další vrstvy sítě.

Uvažujme rozhodnutí, zda jít na přednášku (Ano: 1, Ne: 0) na základě tří faktorů:

- Je přednáška povinná? (Ano: 1, Ne: 0)
- Půjde tam kamarád? (Ano: 1, Ne: 0)
- Chce tam student jít? (Ano: 0, Ne: 1)

Nyní přiřadíme váhy těmto faktorům:

- Váha pro docházku (W1 = 5)
- Váha pro sociální kontakt (W2 = 3)
- Váha pro pocity studenta (W3 = 2)

Byla nastavena prahová hodnota na 3. S těmito hodnotami můžeme vypočítat výstup pomocí aktivizační funkce. Pokud výstup překročí 0, student půjde na přednášku; jinak ne. Tento příklad ilustruje, jak neuronová síť může na základě vah a prahových hodnot rozhodovat o složitějších situacích.

Existují různé typy neuronových sítí, zahrnující perceptron, vícevrstvý perceptron (MLP), konvoluční neuronové sítě (CNN) a rekurentní neuronové sítě (RNN). Každý typ je určen pro různé účely, jako jsou rozpoznání obrazu, analýza vzorů nebo práce s časovými daty. V následujícím odstavci jsou tyto neuronové sítě podrobněji popsány.

- Perceptron
  - Perceptron je nejstarší typ neuronové sítě, vytvořený Frankem Rosenblattem v roce 1958. Tato síť má jednoduchou strukturu a slouží k binární klasifikaci na základě vstupních dat.
- Vícevrstvý perceptron (MLP)
  - Vícevrstvý perceptron, známý také jako multi-layer perceptron (MLP), se skládá z vstupní vrstvy, jedné nebo více skrytých vrstev a výstupní vrstvy. Tato síť, ačkoli je často označována jako MLP, je ve skutečnosti

tvořena sigmoidními neurony, nikoliv perceptrony. Je základem pro oblasti jako počítačové vidění, zpracování přirozeného jazyka a další aplikace v oblasti neuronových sítí.

- Konvoluční neuronové sítě (CNN)
  - Konvoluční neuronové sítě jsou podobné feedforward[1] sítím, ale jsou obvykle využívány pro rozpoznávání obrazu, analýzu vzorů a počítačové vidění. Tyto sítě využívají principy lineární algebry, zejména maticového násobení, k identifikaci vzorů v obrazech.
- Rekurentní neuronové sítě (RNN)
  - Rekurentní neuronové sítě jsou identifikovány svými zpětnými smyčkami. Tyto učící algoritmy jsou hlavně využívány při práci s časovými daty pro předpovídání budoucích událostí, například při předpovědi vývoje na burze nebo při prognózách prodejů. [5] [10]

[1] Feedforward sítě jsou základním typem neuronových sítí, které propojují neurony pouze jedním směrem, od vstupu přes skryté vrstvy až k výstupu, a jsou často využívány pro klasifikaci a regresi.

4 MongoDB

MongoDB je NoSQL databázový systém, který se zaměřuje na ukládání dat ve formátu BSON (Binary JSON). Jedná se o dokumentově orientovanou databázi, což znamená, že data jsou ukládána ve formě dokumentů, obvykle ve formátu JSON. Každý dokument obsahuje klíče a hodnoty, podobně jako jsou uloženy v běžných objektech v programovacím jazyce. MongoDB používá model bez schématu, což znamená, že nemá pevně definovanou strukturu pro data. To umožňuje flexibilitu při ukládání a aktualizaci dat. MongoDB také podporuje indexování, sharding a replikaci, což pomáhá zajišťovat výkon a dostupnost v rozsáhlých a komplexních systémech.

4.1 Použití ve světě

MongoDB je často využíván pro projekty, které vyžadují rychlý přístup k velkým objemem nestrukturovaných nebo polo strukturovaných dat. Je vhodný pro situace, kdy je potřeba agilní přístup k datům a měnícím se požadavkům, například v moderních webových aplikacích, analytických systémech a IoT (Internet of Things) aplikacích.

4.2 Rozdíl mezi SQL a NoSQL databázemi

Na vyšší úrovni mají databáze NoSQL a SQL mnoho podobností. Oba typy databází podporují ukládání dat a dotazy a umožňují získávat, aktualizovat a mazat uložená data. Nicméně pod povrchem se skrývají značné rozdíly, které ovlivňují výkon, škálovatelnost a flexibilitu NoSQL oproti SQL.

4.2.1 Struktura

Databáze SQL jsou založeny na tabulkách, zatímco NoSQL databáze mohou být orientovány na dokumenty, páry klíč-hodnota nebo grafické struktury. V NoSQL databázi může dokument obsahovat páry klíč-hodnota, které mohou být následně řazeny a vnořeny.

4.2.2 Škálovatelnost

SQL databáze škálují vertikálně, obvykle na jednom serveru, a vyžadují od uživatelů zvětšení fyzického hardwaru pro zvýšení kapacity úložiště. Naopak NoSQL databáze nabízí horizontální škálovatelnost, což znamená, že pro zvýšení datové zátěže stačí přidat více serverů. To z nich činí lepší volbu pro moderní cloudové infrastruktury s distribuovanými prostředky.

4.2.3 Jazyky

Databáze SQL používají jazyk SQL (Structured Query Language). NoSQL databáze využívají JSON (JavaScript Object Notation), XML, YAML nebo binární schémata, umožňující práci s nestrukturovanými daty. SQL má předdefinované pevné schéma, zatímco NoSQL databáze jsou flexibilnější.

4.2.4 Podpora

SQL je populární standardní jazyk, který je dobře podporován mnoha různými databázovými systémy, zatímco NoSQL má různé úrovně podpory v různých databázových systémech.

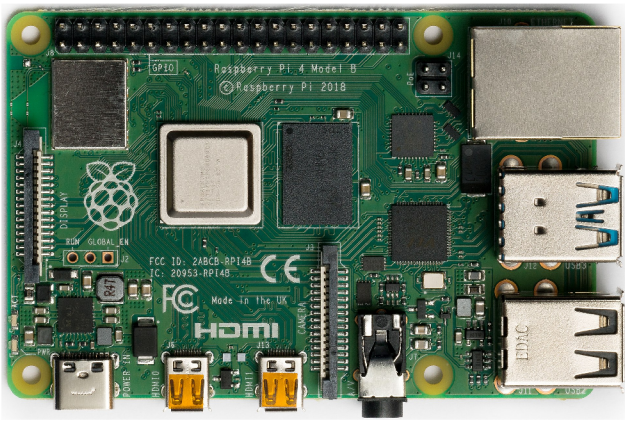
Co se týče podpory, obecně lze říci, že pro databáze SQL je dostupná širší pomoc než pro NoSQL. To je způsobeno tím, že SQL je etablovanější technologie s mnohem více uživateli a vývojáři, kteří vám mohou pomoci s vašimi problémy. Naopak NoSQL je stále relativně nový, s menším množstvím dostupné pomoci na fórech nebo od komunity. Vaše možnosti podpory mohou být omezené, pokud narazíte na obtíže při používání této technologie. [4][2]

5 Raspberry Pi 4B

Jedná se o jednu z nejnovějších verzí rodiny jednodeskových počítačů Raspberry Pi, kterou vyvinula nadace Raspberry Pi Foundation. Tento malý počítač je určen pro různé vývojové, vzdělávací a kutilské projekty, je poháněn čipem Broadcom BCM2711 a vybaven čtyřjádrovým procesorem ARM Cortex-A72 s maximální frekvencí 1,5 GHz. 2 GB, 4 GB, nebo 8 GB operační paměti typu LPDDR4 byly použity pro realizaci této práce.

Z hlediska konektivity je tento jednodeskový počítač vybaven dvěma porty USB 3.0, dvěma porty USB 2.0, gigabitovým Ethernetem, dvěma výstupy HDMI podporujícími rozlišení až 4K a konektorem pro kartu microSD pro ukládání operačního systému a dat Systém je vybaven Má vestavěné moduly Wi-Fi 802.11ac a Bluetooth 5.0 pro bezdrátovou komunikaci.

Raspberry Pi 4B podporuje řadu operačních systémů, včetně Raspberry Pi OS. Pro napájení vyžaduje konektor microUSB s napětím 5 V a minimálním proudem 3A, ale lze použít i nový standard USB-C.



Obrázek 5.1: Raspberry Pi 4B. Zdroj: Wikipedia. [1]

[1] https://commons.wikimedia.org/wiki/File:Raspberry_Pi_4_Model_B_-_Top.jpg

Díky svým malým rozměrům a nízké spotřebě energie se Raspberry stalo oblíbenou platformou pro vývoj zařízení internetu věcí, domácích serverů, mediálních center a dalších projektů vyžadujících levný a výkonný jednodeskový počítač.

Díky pinům GPIO pro připojení různých senzorů, modulů a dalších periferií nabízí Raspberry široké možnosti rozšíření a přizpůsobení. Tato rozhraní jsou důležitou vlastností pro experimenty a projekty, které vyžadují interakci s vnějším světem. Speciálně užitečné jsou také prvky modulu HAT, které lze připojit k pinům GPIO a přidat tak specifické funkce, například displeje nebo složitější moduly.

Raspberry Pi je vhodný pro širokou škálu použití, od jednoduchých projektů pro začátečníky až po složitější vývojové úlohy pro pokročilé uživatele Komunita obklopující Raspberry Pi je aktivní a nabízí mnoho návodů, návodů a projektů, čímž přispívá ke kutilství a vzdělávání ve světě přispívá k popularitě této platformy v kutilském a vzdělávacím světě. V rámci komunity vznikla řada netradičních projektů, jako je vývoj robotů, monitorování životního prostředí a vytváření vlastních zařízení internetu věcí. Kromě toho se malinové počítače staly oblíbeným nástrojem pro výuku programování, hardwarového inženýrství a informatiky.

Umožňují také efektivní využití kontejnerové technologie a virtualizace. Díky dostatečné paměti lze pro fungování využít více kontejnerů nebo virtuálních počítačů, což otevírá dveře vývoji a testování softwaru v izolovaném prostředí. [18]

6 Python knihovny

V této kapitole bude věnována pozornost důležitým knihovnám, které byly použity pro implementaci celého systému, jak pro část, která sbírá obrázky, tudíž část Raspberry a část serverová, která se stará o detekci, klasifikaci a ukládání do databáze. Konkrétní use case diagram pro tuto implementaci je uveden v kapitole 8 na straně 30.

6.1 Knihovny pro Raspberry

První balíček nejdůležitějších knihoven jsou knihovny, které jsou využity pro sběr dat. Patří sem knihovny jako například libcamera [13], která umožňuje práci a komunikaci s fotoaparáty a kamerami pro vestavěné zařízení. Tato knihovna musí být instalována jako balíček v systému a v Pythonu tuto knihovnu využívá knihovna picamera, případně novější verze picamera2, která byla použita pro práci s kamerou.[15] Další knihovna, která byla použita, je oficiální knihovna výrobce GPS modulu Waveshare. Ta se dá stáhnout na oficiálních stránkách a poskytuje základní metody pro získávání dat z GPS modulu.[6] Další knihovna s názvem piexif byla využita pro ukládání metadat času a GPS souřadnic do obrázků. S touto knihovnou systém pracuje i na serverové části, protože dokáže jak ukládat, tak i metadata číst.[16] Poslední z hlavních knihoven na Raspberry části je knihovna paramiko. Paramiko je knihovna, která poskytuje implementaci protokolu SSH pro vytváření šifrovaných a bezpečných spojení. Tato knihovna byla použita pro odesílání obrázků na server. Knihovna umožňuje i využití SSH klíče, takže není nutné zadávat přímo do kódu heslo, které se musí vytvořit pro úspěšné připojení ke vzdálenému počítači. Stačí si jen na Raspberry vytvořit veřejný SSH klíč a vložit ho na server do složky .ssh/authorized_keys.[14]

6.2 Knihovny pro server

Druhý balíček nejdůležitějších knihoven jsou knihovny, které jsou použity na serverové části. V procesu posloupnosti chodu systému bude jako první zmíněna knihovna pro práci s databází. Jak už bylo zmíněno v předchozí kapitole, byla použita databáze MongoDB, důvody použití jsou zmíněny v kapitole 8 na straně 30. Byla tedy použita knihovna PyMongo. PyMongo je oficiální Pythonový klient pro MongoDB. Umožňuje podstatně robustní řešení pro připojení, úpravu, vkládání a mazání databází. Podporuje také příkazy pro hledání, mazání a jakoukoliv práci s daty v kolekci. [8] Další dvě knihovny, které byly použity na serverové části jsou knihovny pro detekci a klasifikaci obrázků. K těmto knihovnám bude zmíněna i hlubší teorie pro lepší porozumění funkčnosti. Co se týče implementace a využití, budou dále zmíněny v kapitole 8 na straně 30.

6.2.1 EfficientDet

První z těchto knihoven je knihovna EfficientDet. EfficientDet je model detekce objektů založený na hlubokém učení, který je navržen tak, aby dosahoval vysoké přesnosti detekce objektů při zachování vysoké výpočetní efektivity. Model kombinuje prvky několika předchozích architektur, jako jsou RetinaNet a EfficientNet, a dosahuje tak významného zlepšení výkonu.

EfficientDet je založen na myšlence principu ”škálování zdrojů”. Jinými slovy, využívá efektivní síťovou architekturu, která se dokáže přizpůsobit různým úrovním přesnosti a efektivity. Některé klíčové prvky, díky nimž EfficientDet funguje, jsou tyto:

- **EfficientNet jako stavební prvek** EfficientDet používá architekturu EfficientNet jako základní stavební prvek. Tato architektura je optimalizována tak, aby bylo dosaženo vysoké přesnosti při minimálních výpočetních zdrojích.
- **BiFPN** (Bi-directional Feature Pyramid Network). BiFPN je nová architektura propojení v Pyramidální síti vlastností (FPN), která umožňuje efektivní propojení různých úrovní vlastností v síti a zlepšuje schopnost modelu detekce objektů pracovat s objekty různých velikostí.
- **Kompozitní škálování** Kompozitní škálování je technika, která vyvažuje přesnost a účinnost modelu škálováním hloubky a šířky sítě i rozlišení vstup-

ních obrazů. Tato technika umožňuje dosáhnout vysoké přesnosti a účinnosti při různých úrovních výpočetních zdrojů.

- **Detekční hlavičky** EfficientDet používá různé detekční hlavičky k předpovědi polohy a třídy objektů na základě různých úrovní pyramidální sítě prvků.

Jako celek tedy EfficientDet dosahuje vysoké přesnosti a účinnosti při detekci objektů v obrazech využitím efektivní síťové architektury a pokročilých technik škálování a propojování prvků. [11]

6.2.2 TensorFlow

Jako knihovnu pro klasifikaci byla použita knihovna TensorFlow, přesněji její vysokoúrovňové API Keras. Keras je vysokoúrovňové rozhraní API pro hluboké učení, které poskytuje jednoduché a intuitivní rozhraní pro vytváření, trénování a nasazování neuronových sítí v jazyce Python. Je navrženo tak, aby bylo snadno použitelné pro začátečníky a zároveň poskytovat dostatečnou flexibilitu a výkon pro pokročilé výzkumníky a inženýry. Níže je uveden podrobnější popis toho, jak Keras funguje:

- **Modularita** Keras je vysoce modulární. Základními stavebními bloky jsou vrstvy představující různé operace a transformace, například plně propojená vrstva (Dense), konvoluční vrstva (Conv2D) a rekurentní vrstva (LSTM, GRU). Uživatelé mohou snadno vytvářet složité architektury sítí kombinováním a propojováním.

- **Modely** Základním stavebním prvkem Kerasu je model. ´Model v Kerasu je obvykle posloupnost vrstev, které tvoří neuronovou síť. Uživatelé mohou definovat modely sekvenčně pomocí sekvenčních modelů nebo vytvářet vícevláknové a vícevýstupové modely pomocí rozhraní API funkcí. Modely Keras lze snadno trénovat, vyhodnocovat a používat k predikci nových dat.

- **Optimalizátory a ztrátové funkce** Keras poskytuje širokou škálu vestavěných optimalizátorů pro trénování modelů, včetně SGD, Adam a RMSprop. Uživatelé mohou také definovat vlastní optimalizátory. Kromě toho Keras obsahuje řadu ztrátových funkcí pro různé úlohy učení, jako je klasifikace, regrese a segmentace.

- **Trénování a vyhodnocování** Keras poskytuje jednoduché API pro trénování modelů pomocí metody fit(), kde lze zadat tréninková data, ztrátové funkce, optimalizátory a další parametry trénování. Po trénování lze modely vyhodnotit na validačních datech pomocí metody evaluate(), která vypočítá přesnost modelu na základě zadaných metrik.

- **Podpora GPU a distribuovaného trénování** Keras je navržen tak, aby byl kompatibilní s GPU, což umožňuje trénovat modely na grafické kartě pomocí knihoven, jako jsou TensorFlow a PyTorch. Kompatibilita s TensorFlow také umožňuje použití distribuovaného trénování a trénování na více serverech. [7]

Jako poslední zmíněné knihovny jsou knihovny, které byly použity pro webovou aplikaci. Hlavní knihovna pro webovou aplikaci je knihovna Flask. Flask je framework pro vývoj webových aplikací v jazyce Python. Další možné frameworky pro webové aplikace, které využívají jazyk Pythnou jsou například Django, FastAPI nebo Bottle. Další hlavní knihovny pro webovou aplikaci jsou knihovny Folium a Bokeh. Folium je knihovna, která zpracovává interaktivní vizualizaci mapy a její vlastní konfigurace od vzhledu, tak například k vlastním bodům na mapě. POužívá mapové podklady z OpenSteetMap, Mapbox a dalších zdrojů. Je založena na knihovně Leaflet.js. [3]

Poslední podstatná knihovna této kapitoly je knihovna Bokeh. Tato knihovna byla využita pro tvorbu interaktivních vizualizací dat ve webové aplikaci, přesněji pro vizualizaci interaktivního grafu dat z počtem dopravních značek a jejich tříd. Bokeh grafy se dají různě konfigurovat. Knihovna poskytuje možnosti úpravy barev, vzhledu, typů grafů a nástrojů, které mohou s grafy ve webové aplikaci pracovat, to jsou nástroje jako například přiblížení a oddálení grafu, volného pohybu v grafu nebo také interaktivní ukazatel hodnot, který reaguje při najetí myši. [12]

7 Konfigurace serveru

K serverové části byl poskytnut virtuální server naší fakulty. Server disponuje s uložištěm o velikosti 64GB a operační pamětí o velikosti 8GB. Na serveru běží operační systém Linux v distribuci Ubuntu 64-bit ve verzi 22.04 jammy. K správnému chodu a zabezpečení serveru bylo nutné vykonat několik kroků. První kroky vedly k nainstalování SSH daemona, nastavení klíču pro přístup bez hesel, instalace a konfigurace firewallu. Nakonec byl nainstalován balíček fail2ban, který chrání síťové připojení na server zejména přes port 22, tudíž přes SSH, a úspěšně blokuje a uvádí IP adresy po několikanásobně neúspěšném pokusu o přihlášení se do takzvaného "jail", tedy vězení, kde jsou IP adresy ponechány po určitý čas. Po uplynutí tohoto času jsou IP adresy uvolněny a uživatel se může opět přihlásit. Příkladem může být jednoduché pozorování a zjištění, že během několika minut se na server, přes SSH, chtělo přihlásit něco přes 100 uživatelů, zejména z IP adres lokalizovaných v Číně nebo Rusku. Dále byl nainstalován daemon pro FTP, přesněji VSFPTD, a povoleny porty ve firewallu pro komunikaci s okolím. Jako poslední z hlavní konfigurace byl nainstalován Python ve verzi 3.10. a kontejnerizační nástroj Docker ve kterém běží databáze. [1]

7.1 Webový server

Pro instanci webového serveru byl použit Nginx a to hlavně z důvodu známosti konfigurace. Pro implementaci této webové aplikace, jaká je přítomna v této práci, je Nginx více než dostačující. Oproti například Apache HTTP server má Nginx pro statické stránky více než dvojnásobnou rychlost hlavně díky funkcionalitě Nginx, která dokáže na jednom jádře zprocesovat několik požadavků ve stejném čase. Zatímco Apache pro každý požadavek vytváří jeho vlastní vlákno. Než bude zmíněna konkrétní konfigurace je dobré zmínit, že Nginx využívá HTTPS protokol, pro který automaticky obnovuje certifikát služba certbot, která využívá certifikáty

Let's Encrypt. Nyní tedy k samotné konfiguraci. V první řadě muselo být Nginx povoleno ve firewallu. Pro účely práce bylo ve firewallu povolena možnost "Nginx Full". Tato možnost zahrnuje povolení portu 80, tj. pro HTTP a portu 443, tj. pro HTTPS. Konfigurační soubory Nginx se implicitně v linuxové struktuře nachází ve složce `/etc/nginx`. Pro konfiguraci samotných webových stránek slouží složka `/etc/nginx/sites-available/`. V této složce je možné vytvořit několik konfiguračních souborů. Jelikož na tomto serveru běží jen jedna webová stránka, bylo zvoleno upravení konfigurace hlavního souboru `default`. Jelikož je webová aplikace dynamická, Nginx potřebuje využívat nějakého modulu, který dokáže tento obsah zpracovávat. Byl tedy nainstalován modul PHP-FPM, který je pro zpracovávání dynamického obsahu pro Nginx typický. Tento modul je totiž nutné také zahrnout do hlavní konfigurace. Ukázka hlavní části konfigurace tak, aby nginx byl schopen zobrazit spuštěné python aplikace na portu 8000.

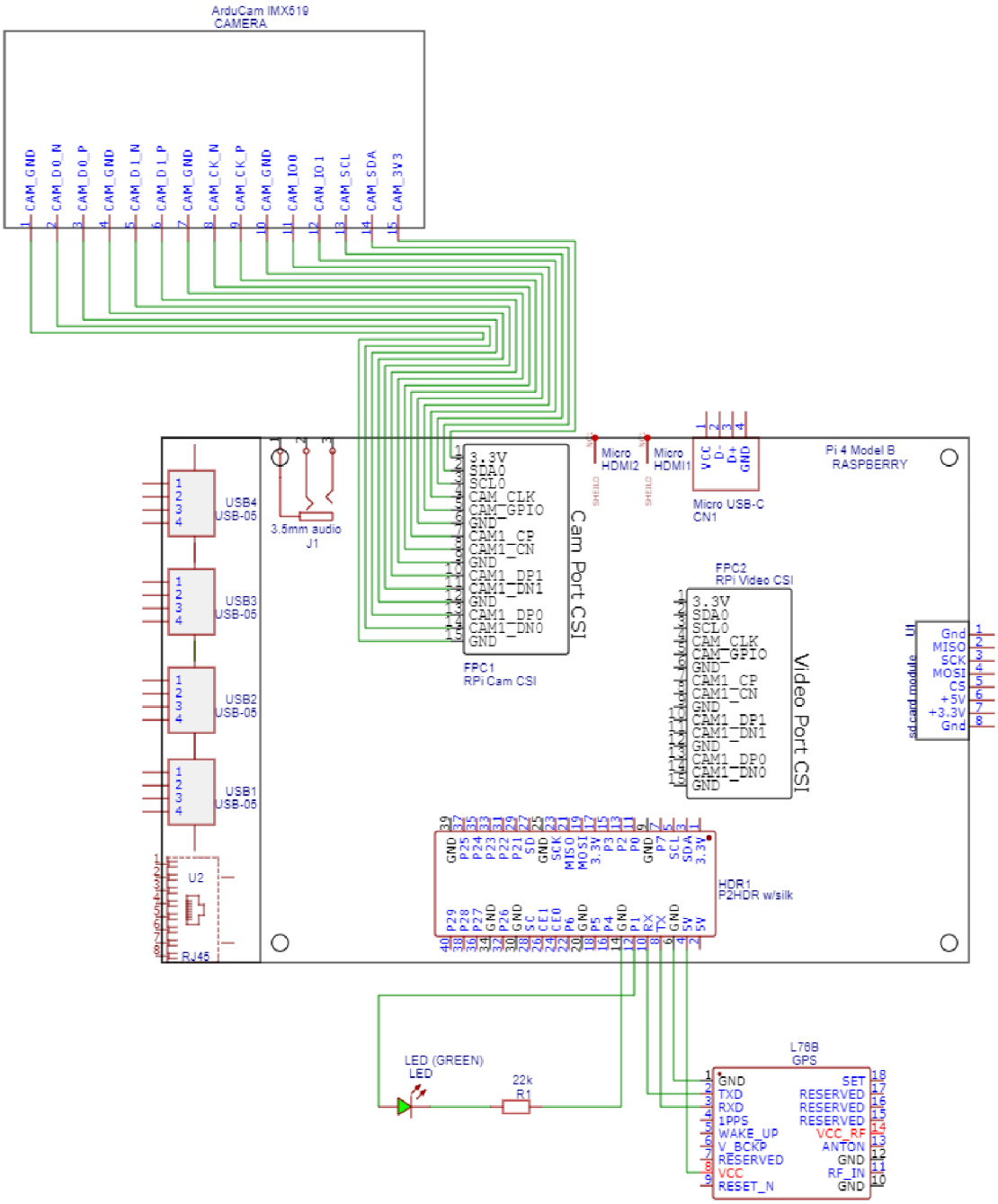
```
index index.html index.htm index.nginx-debian.html;

server_name doprava.nti.tul.cz;

location / {
# First attempt to serve request as file, then
# as directory, then fall back to displaying a 404.
proxy_pass http://127.0.0.1:8000;
proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Proto $scheme;
proxy_redirect off;
proxy_buffering off;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection "upgrade";
autoindex off;
}
```

8 Implementace

V některých z předchozích kapitol bylo zmíněno něco málo o implementaci a odkazováno na tuto kapitolu. V této kapitole bude popsána kompletní implementace funkčního systému od sběru dat až po webovou aplikaci.

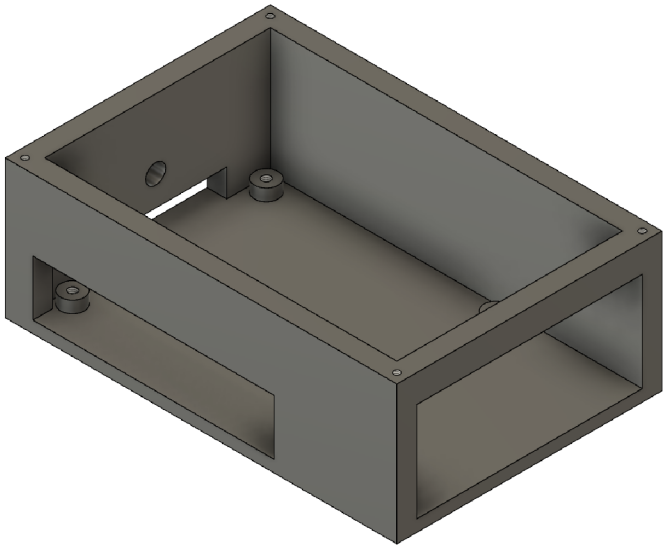
8.1 Kompletace hardwaru

Tento systém využívá jednodeskový počítač Raspberry Pi 4 B ve verzi s 8GB operační paměti. Pro získávání GPS souřadnic byl použit GPS modul od značky Waveshare, přesněji L76K. Variant tohoto modulu je několik a liší se hlavně v nabídkách, jaké polohovací systémy dokáže modul využívat. Asi jako poslední z hlavních rozdílů je čas, za který se modul dokáže spustit. Použitý modul L76K disponuje využitím polohovacích systémů GPS, GLONASS, BDS a QZSS; ve zkratce polohovací systémy USA, Ruska, Číny a Japonska. Jak už bylo zmiňováno v kapitole 6 na straně 24, výrobce poskytuje i veřejnou knihovnu pro python, která dokáže modul obsloužit, nicméně pro účely této bakalářské práce bylo nutné některé metody upravit nebo vytvořit nové. Další periferií, která je v této implementaci využívána, je kamera. Kamera je od výrobce ArduCam s čipem IMX519 s rozlišením 16MP a automatickým ostřením. Poslední součástkou připojenou k Raspberry je zelená LED dioda, která signalizuje zapnutý skript pro sběr obrázků a následné odeslání obrázků na server. Jako zdroj napájení pro Raspberry je použito 12V napájení z vozidla. Na obrázku 8.1 můžete vidět diagram zapojení pro Raspberry a všemi jeho periferiemi již zmíněnými v předchozím textu.

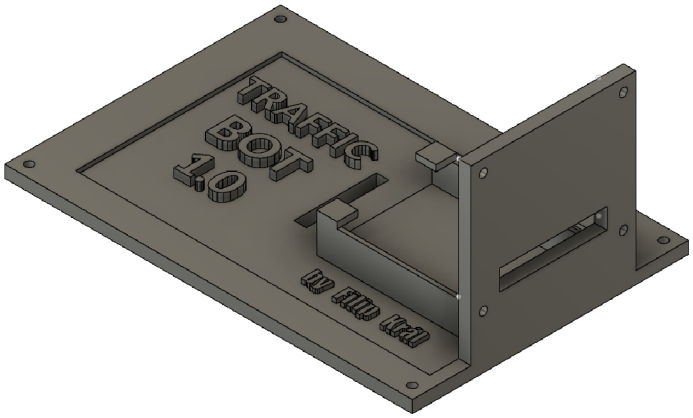


Obrázek 8.1: Diagram zapojení

Pro potřeby implementace byla vymodelována krabička s víkem. Tato krabička byla důležitá pro testování, nýbrž Raspberry muselo být v automobilu uchyceno na palubní desce. K tomu byl použit magnetický držák pro mobilní telefony. Kovový plíšek byl nalepen zespodu krabičky tak, aby kamera mířila na čelní sklo. Na obrázcích 8.2 a 8.3 je ukázka vymodelované krabičky a jejího víka.



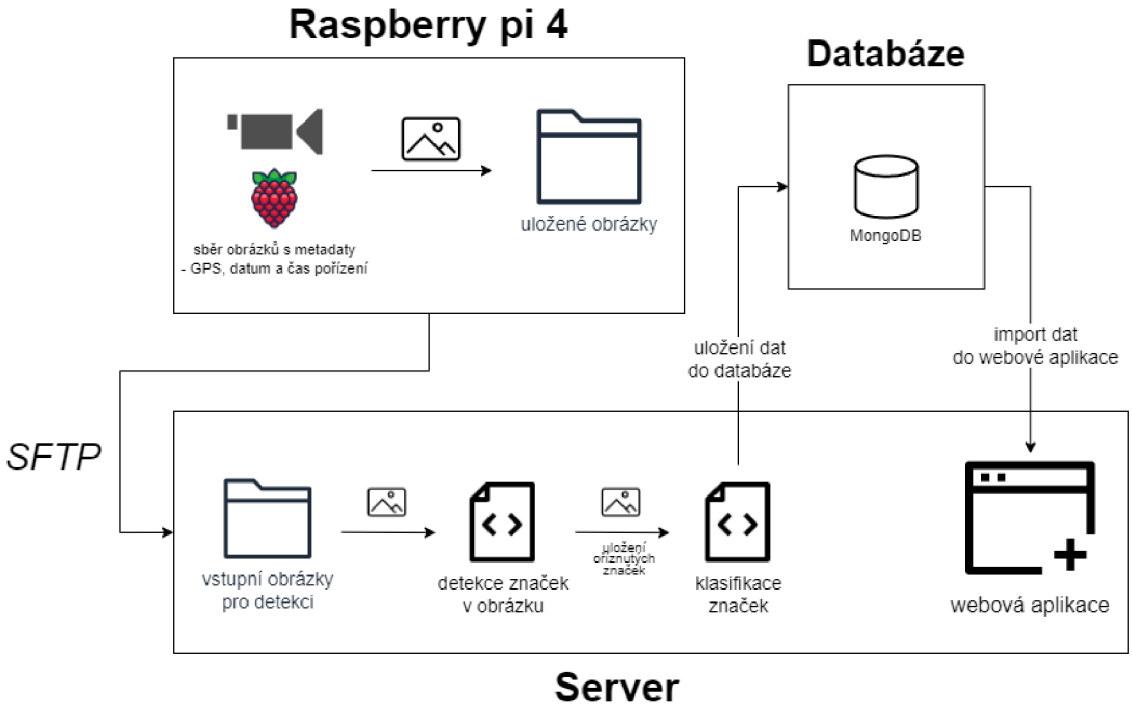
Obrázek 8.2: Model krabičky



Obrázek 8.3: Model víka

8.2 Use case diagram

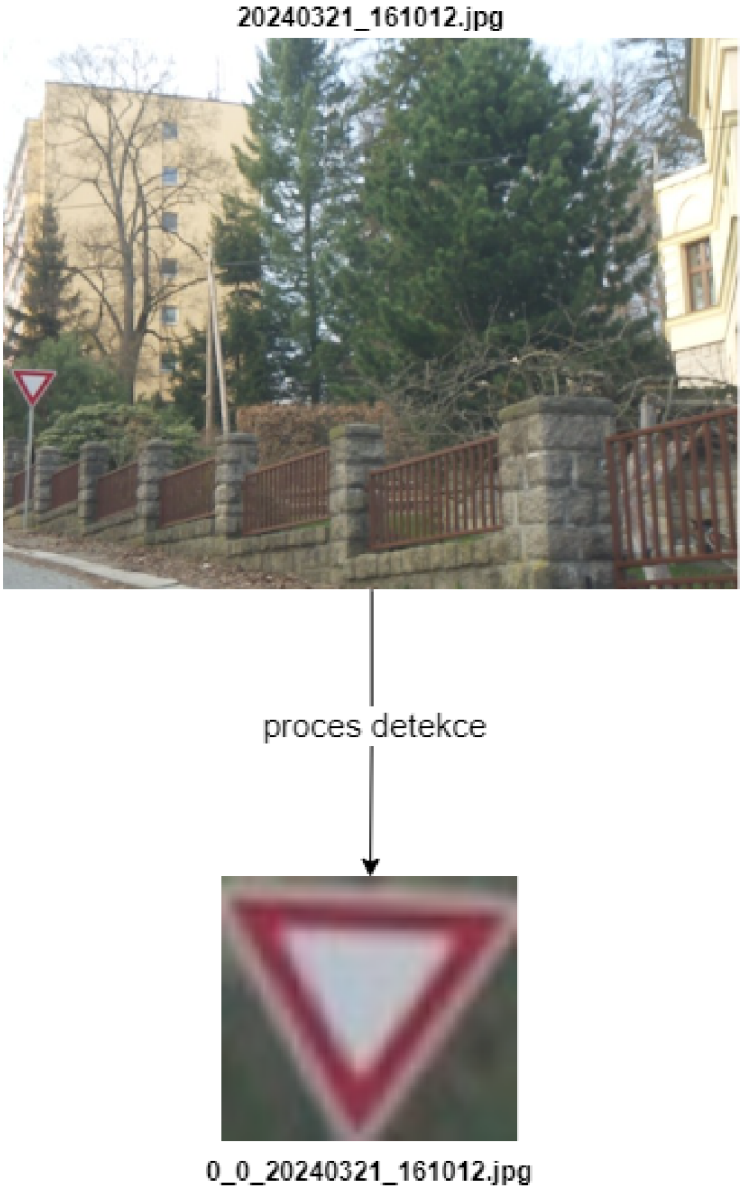
V této podkapitole bude popsán use case diagram pro implementaci. Na obrázku 8.4 je vidět postup funkčnosti kódu. Celý proces začíná na straně Raspberry, které sbírá obrázky a ukládá do jejich metadat GPS souřadnice. Obrázky ukládá s časem pořízení v názvu souboru, takže například ve stylu 20240210_165248.*jpg*. Následně jsou obrázky posílány přes protokol SFTP na server, na kterém běží zbytek funkcionalit celého systému. Nejdřív vstupní obrázky musí projít detekcí, následně klasifikací. Po úspěšné klasifikaci se data o obrázku uloží do databáze a z databáze se následně data promítají do webové aplikace.



Obrázek 8.4: Use case diagram

8.3 Rozpoznávání

Na obrázku 8.5 je jednoduše znázorněn proces detekce značky v obrázku. Na prvním obrázku je vidět původní scéna z ulice a na druhém obrázku je již výřez dopravní značky, který je pak použit pro klasifikaci dopravní značky. Následný zapis dat z klasifikace do databáze je možno vidět na následujícím výpisu z MongoDB.



Obrázek 8.5: Detekce

```
traffic_data>
    db.image_data.find({ "filename": "0_0_20240321_161012.jpg" })
    [
      {
        _id: ObjectId('6613f56e99be181ed466b5ce'),
        filename: '0_0_20240321_161012.jpg',
        predicted_class: 'Dej přednost v jízdě',
        latitude: 50.773421166667,
```

```
    longitude: 15.0833635,
    time: '2024-04-08 13:47:26',
    probability: 95
  }
]
```

8.3.1 Detekce

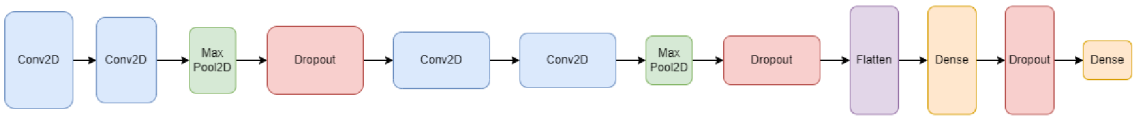
Pro detekci byla použita již zmíněná knihonva EfficientDet. Pro samotný model detektoru byl použit dataset COCO. Tento dataset obsahuje přes 330 000 obrázků různých objektů, které je možné vidět kolem sebe. Každý obrázek v datasetu je anotovaný. Anotováním se rozumí opatření obrázku správným popiskem, segmentačními maskami a zobrazením ROI - Range of Interest. Tento, většinou čtverec nebo obdélník, který má 4 souřadnice jeho rohů, určuje pozici daného objektu. Pro tuto bakalářskou práci, která se zabývá detekcí dopravních značek, byl použit model, který byl sestaven pouze pro detekci dopravních značek a pokud možno žádných jiných objektů. Detektor má samozřejmě i nějaké odchylky a může se stát, že určí i objekt, který není dopravní značkou, ale je mu podobný.[1]

8.3.2 Klasifikace

Klasifikační model byl natrénován na datové sadě GTSRB[2]. Tento dataset obsahuje 43 druhů dopravních značek, jako například značka hlavní silnice, některé zákazové značky a některé upozorňující značky. Pro zlepšení výsledků a pro větší škálů rozpoznaných značek byl model, společně s GTSRB, natrénován i o dalších 19 druhů dopravních značek, protože hodně frekventované značky, jako například zákaz stání, zákaz zastavení nebo jednosměrný provoz v ulici, původní dataset GTSRB neobsahuje. Nicméně pro úspěšné trénování musela být struktura nového datasetu upravena, protože neodpovídá původní struktuře GTSRB. Byla potřeba upravit anotace, které jsou zapsány v .csv souborech. Následný model neuronové sítě se skládá z několika vrstev. Ty jsou znázorněny na obrázku 8.6.

Tento model je sestaven pomocí knihovny Keras s použitím sekvenčního modelu.

[1] https://github.com/zylo117/Yet-Another-EfficientDet-Pytorch
[2] https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign



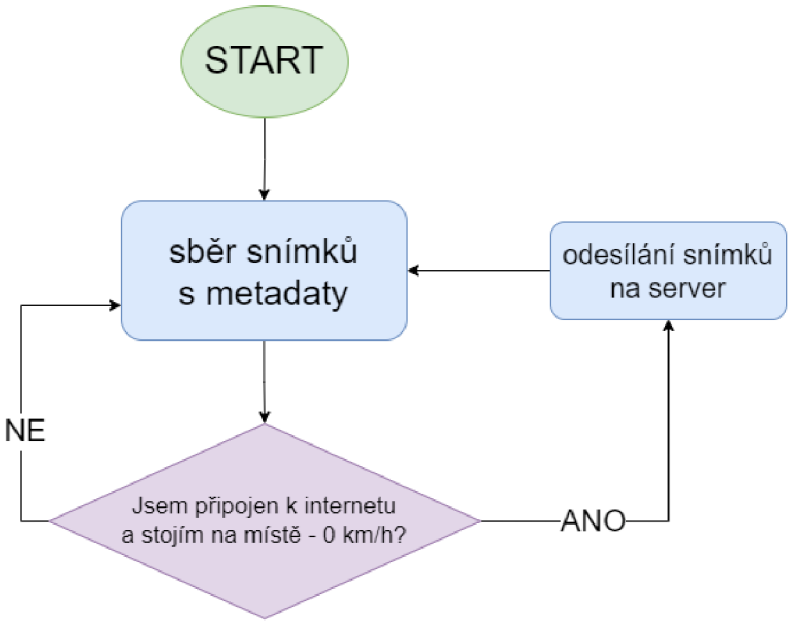
Obrázek 8.6: Klasifikační model a jeho vrstvy

- První konvoluční vrstva Conv2D má filtr nastaven na 32 o velikosti jádra 5x5. Aktivační funkcí je ReLU, což znamená, že každý výstupní pixel je aktivován, pokud je vstup větší než nula. Tato vrstva je vstupní vrstvou.
- Druhá konvoluční vrstva Conv2D má opět filtr nastaven na 32 o velikosti jádra 5x5 s aktivací ReLU. Tato vrstva opět aplikuje konvoluci na vstupní data.
- Následuje poolovací vrstva MaxPool2D s poolovacím oknem o velikosti 2x2, která redukuje rozměry vstupních dat a tím zjednodušuje model. Poolování se provádí pomocí operace max, kde je každý výstup z poolovací oblasti roven maximální hodnotě v této oblasti.
- Vrstva Dropout s poměrem 0.25 náhodně deaktivuje určitý počet jednotek v předchozí vrstvě během trénování, což pomáhá předcházet přetrénování a zlepšuje výkonnost modelu.
- Následuje opětovné použití konvolučních vrstev (s nastaveným filtrem 64 o velikosti jádra 3x3) a poolovacích vrstev pro další extrakci rysů a zmenšení rozměrů dat.
- Po několika vrstvách konvoluce a poolingu následuje plně propojená vrstva Dense s 256 neurony, která přijímá vstupní data zploštěná pomocí Pro vrstvu Flatten je aktivační funkcí opět ReLU. Další vrstva Dropout s poměrem 0.5 opět pro regulaci přetrénování.
- Poslední vrstva Dense je plně propojená s 62 neurony, které odpovídají počtu tříd (nebo kategorií) v datasetu. Aktivační funkcí je softmax, která vrací pravděpodobnostní distribuci přes třídy.

Model je zkompilován s cílovou funkcí pro kategorickou klasifikaci a optimalizátorem Adam s výchozími parametry. Jako metriky jsou použity přesnost a ztrátová funkce. Model byl natrénován s přesností 96%.

8.4 Automatizace

Automatizace celého systému spočívá v tom, že Raspberry hned po startu systému začne vykonávat kód, je tam pro tyto případy vytvořena služba, která se zapne automaticky po startu systému. Kód následně sbírá a ukládá obrázky a zjišťuje, jestli je Raspberry připojeno do jakékoliv eduroam sítě a jestli je podle GPS nulová rychlost pohybu vozidla. Jakmile jsou tyto podmínky splněny, kód automaticky přes SFTP pošle celý obsah složky s obrázky na server a následně obsah složky smaže. Takto tento proces funguje až do vypnutí Raspberry z napájení.



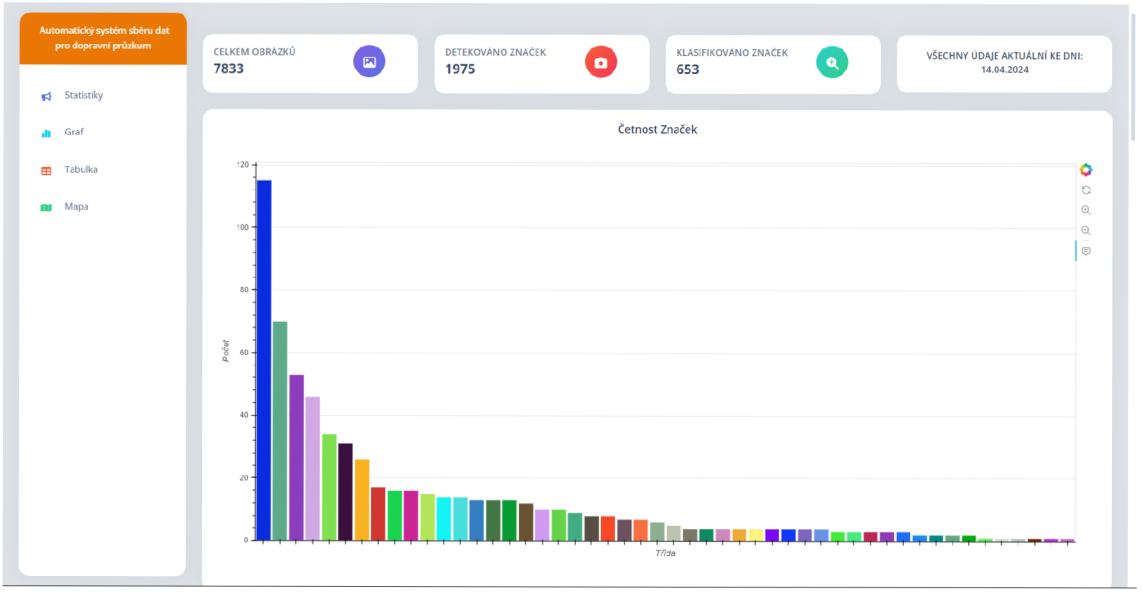
Obrázek 8.7: Automatizace pro Raspberry

Automatizace na serverové straně je znatelně jednodušší. Pro webovou aplikaci je vytvořena služba, která nechává aplikaci zapnutou, tak dlouho, jak jen je potřeba. V kódu se poté nachází několik příkazů, které automaticky v určitý čas ve dne zapínají metody pro aktualizaci počtu pořízených obrázků, konečný údaj se poté zapíše do kolekce v databázi. Aktualizuje také počet detekovaných obrázků stejným způsobem, jako u celkového počtu pořízených obrázků. Zapíná také skripty pro detekci a klasifikaci. Všechny tyto úkony jsou nastavené tak, aby probíhaly 1 hodinu před půlnocí daného dne. Avšak pro potřeby a demonstraci systému jsou tyto úkony zakomentované, tudíž nefunkční. Následující blok je ukázka automatizace, která je připravena na serveru.

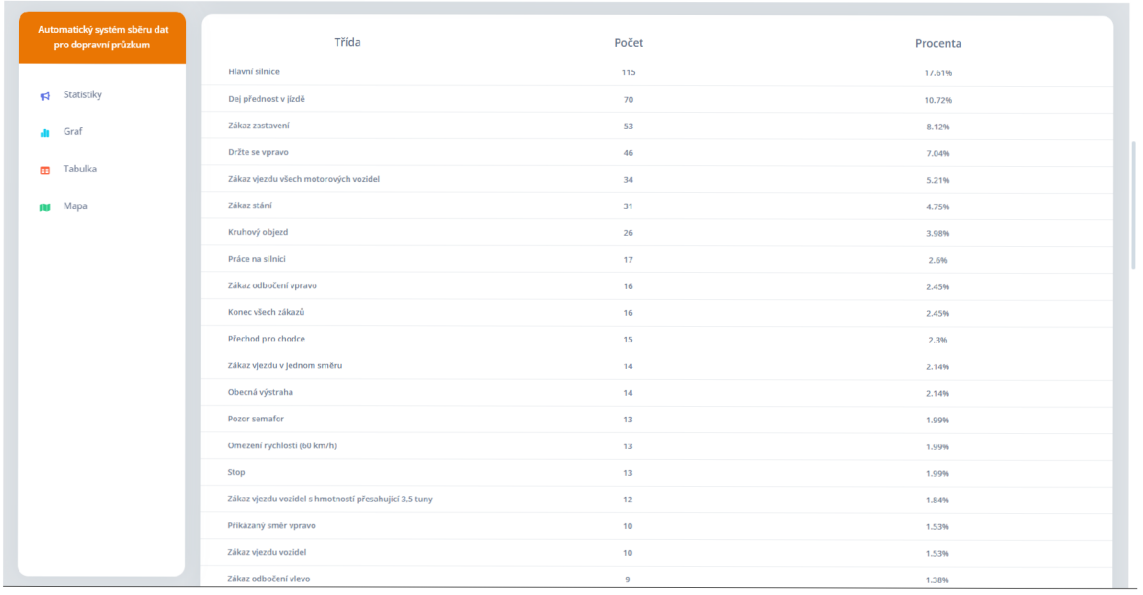
```
def schedule_update():
    schedule.every().day.at("23:00").do(detection)
    schedule.every().day.at("23:45").do(classification_files_count)
    schedule.every().day.at("23:58").
                                do(delete_contents_of_directories)
    schedule.every().day.at("00:00").do(update_and_insert_file_count)
    schedule.every().day.at("00:00").
                                do(update_and_insert_detected_count)

    while True:
        schedule.run_pending()
        time.sleep(60)
```

8.5 Webová aplikace

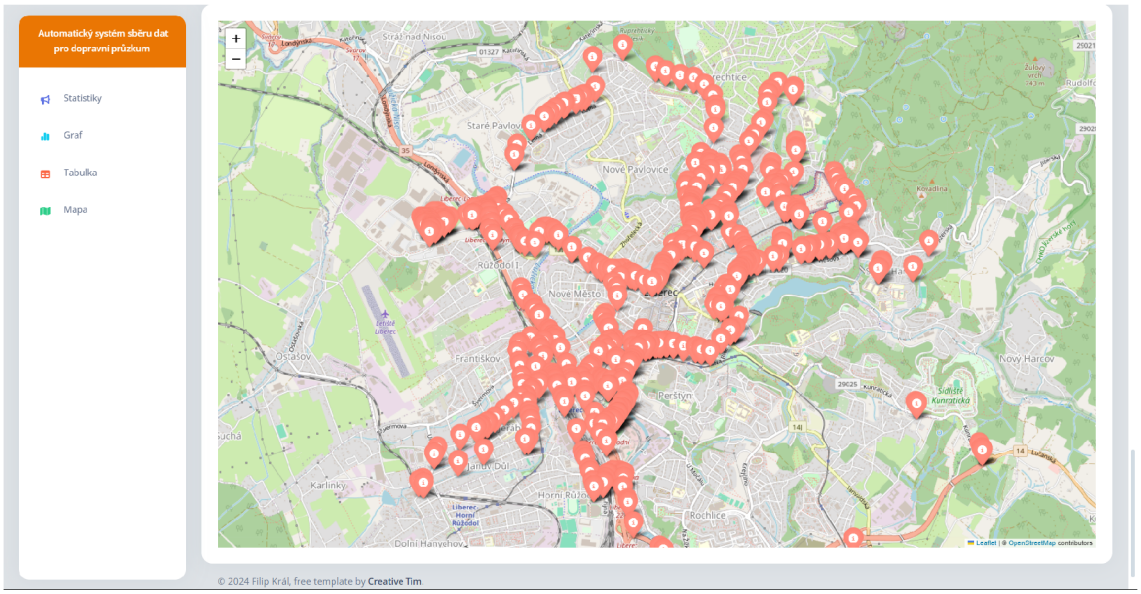
Jak již bylo zmíněno v kapitole 6 na stránce 24 k webové aplikaci byl použit Python a jeho framework Flask. První z těchto knihoven je knihovna EfficientD se jedná čistě o statistické zhodnocení, je tedy aplikace koncipována jako zobrazení důležitých dat z testování. Aplikace je dostupná na doméně https://doprava.nti.tul.cz. Zobrazení se dělí na několik částí. V horní části stránky jsou jednoduché statistiky o tom, jaký je celkový počet pořízených obrázků, celkový počet potenciálně detekovaných značek a celkový počet klasifikovaných značek. Jako doplňující informace je datum aktuálních údajů. Pod statistikami se nachází část s grafem, který zobrazuje počet značek pro každou třídu, která byla klasifikována. Pod grafem je tabulka, která k počtu značek pro každou třídu přidává i procentuální podíl mezi všemi klasifikovanými značkami. A nakonec pod tabulkou je mapa, která zobrazuje body s klasifikovanými značkami. Tyto body jsou umístěny přesně podle GPS souřadnic z pořízeného obrázku. Aplikace využívá pro stylování veřejně dostupnou šablonu, která využívá Bootstrap. Je ale nutné podotknout, že originální šablona byla mnohem komplexnější a pro potřeby této aplikace bylo využito rozložení jen hlavní stránky, takzvaného "dashboardu". Původní rozvržení bylo totiž určeno jako návrh akciového portfolia a sledování akcií v reálném čase. Jako názorná ukázka vzhledu webové aplikace jsou určeny snímky



Obrázek 8.8: Statistiky a graf



Obrázek 8.9: Tabulka s podílem



Obrázek 8.10: Mapa zobrazující dopravní značky

9 Výsledky a testování

V průběhu testování bylo vytvořeno několik postupů, jak odchytit případné chyby a kontrolovat výstupy. Po natrénování prvního klasifikačního modelu nastal čas na první jízdu a celkové testování automatizace. Hned při prvních jízdách se objevily problémy se správným připojením, naštěstí se Raspberry podařilo úspěšně připojit do eduroam sítě, i když bylo z mnoha zdrojů vyslyšeno, že to není tak jednoduché, ba dokonce nemožné.

Obrázek 9.1: Průběh testování

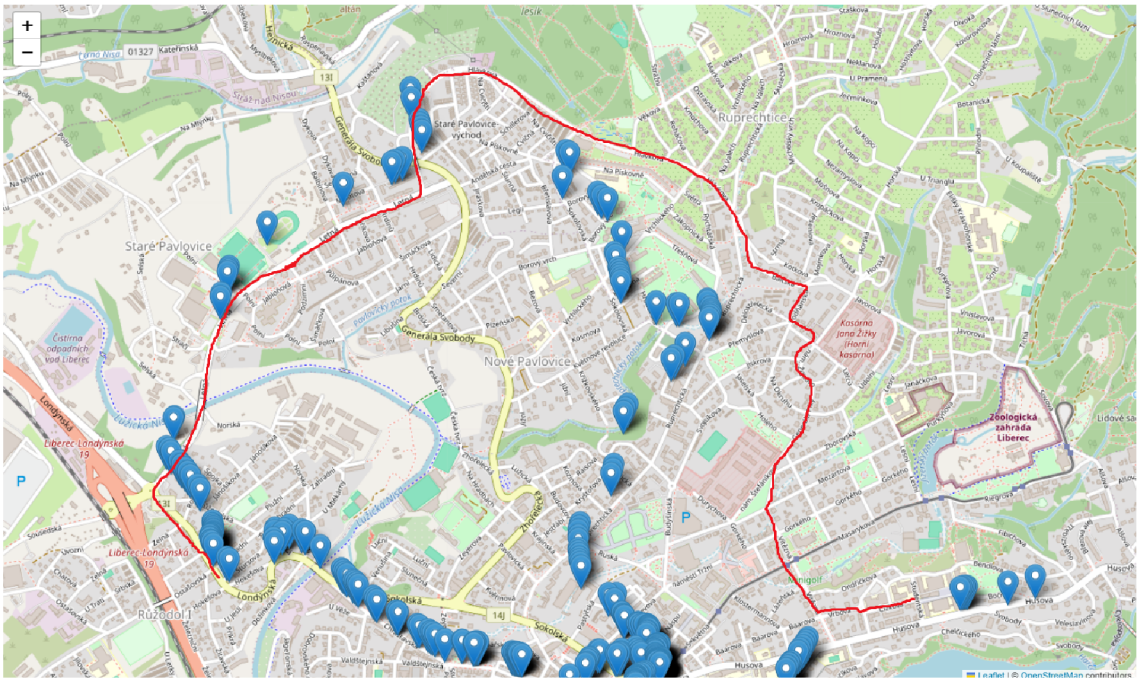
Obrázek 9.2: Průběh testování; uchycení sestavy

9.1 Statistiky z testování

Celkový počet provedených testovacích jízd je kolem 5 až 7. Každá testovací jízda trvala kolem 30 minut až hodiny. Byly prováděny výhradně po ulicích Liberce z důvodu velké saturace dopravních značek, jízda po dálnici, případně po okresních silnicích by nebyla v tomto případě moc efektivní a Raspberry by zbytečně sbíralo větší počet snímků bez dopravních značek. Počasí při testovacích jízdách bylo vesměs slunečné a bez deště, to ale neznamená, že by sestava nedokázala v horších světelných podmínkách sběr snímků, ale jízdy byly plánovány pro nejlepší možný výsledek a účinek. V půlce jedné jízdy se spustil déšť a GPS z nějakého důvodu přestala úplně komunikovat, až do konce deště, který trval několik minut. Tato zvláštnost nebyla ničím neobvyklým, jelikož GPS patří k těm levnějším, které se na trhu dají koupit - tento modul stál 14$, není vždy zaručena perfektní funkčnost.

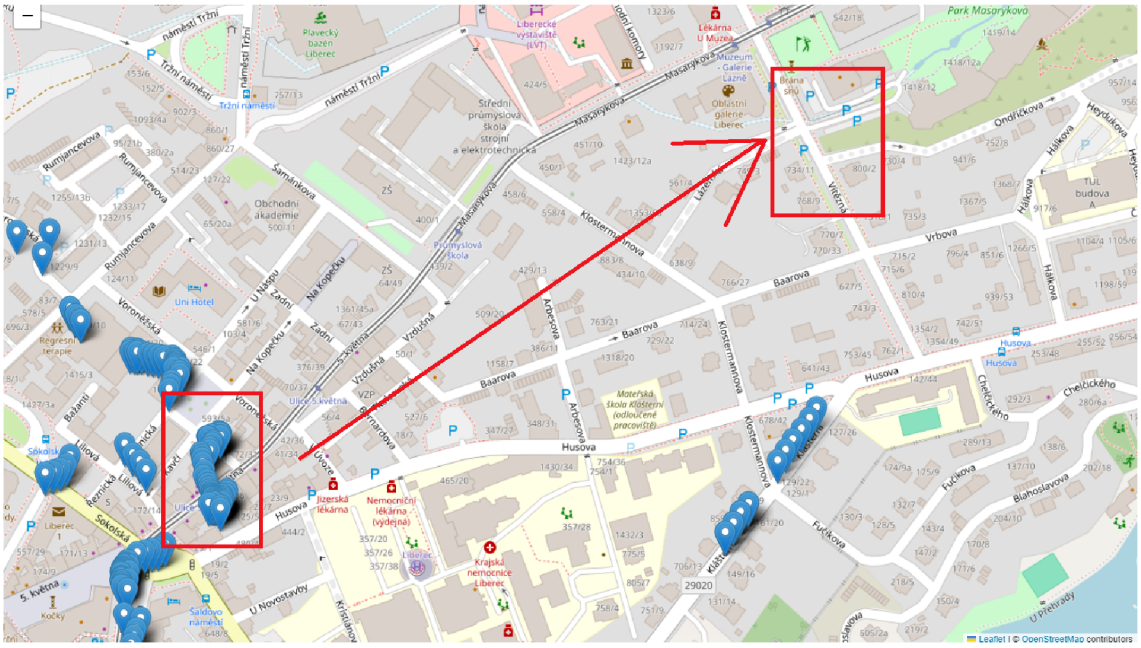
9.1.1 Odstraňování závad polohovacího modulu

Jak již bylo zmíněno v předchozím odstavci, s modulem bylo po dobu testování několik problémů. Jakmile byla automatizace plně odchycena, nastal čas se plně věnovat GPS, protože ta při prvních jízdách vykazovala GPS souřadnice podivného formátu. S tímto problémem bylo řešení na delší dobu, protože knihovna, kterou výrobce poskytuje, nemá absolutně žádnou dokumentaci. Naštěstí se úpravy, které byly provedeny v knihovně, projevily hned při první testovací jízdě, čistě určené pro GPS modul.



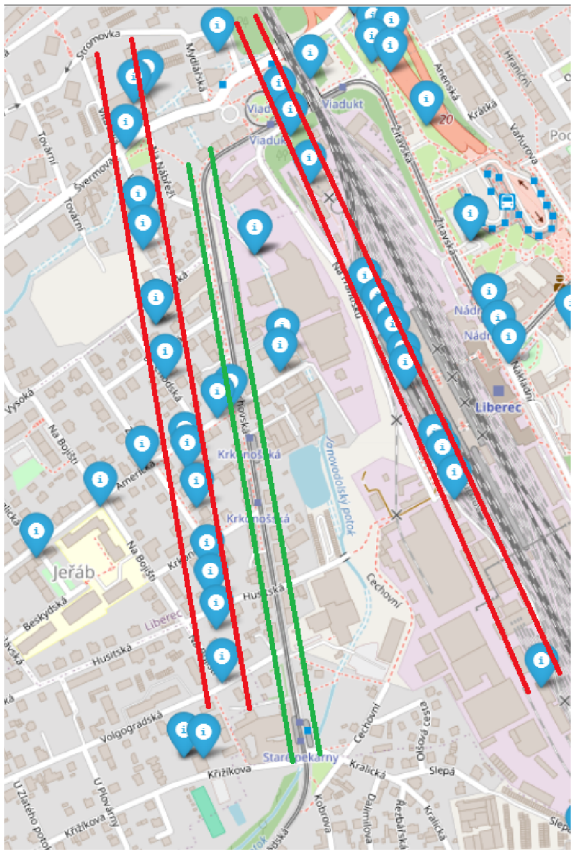
Obrázek 9.3: Chybné polohování #1

Na obrázku 9.3 je možné vidět špatně napolohovaná cesta. Správná cesta je zvýrazněna červenou linkou, natož podle GPS a jeho uložených poloh je patrné, že nejsou tam, kde by měly být. Je ale s podivem, že modul po nějaké době, přibližně po 10 minutách jízdy, začal vykazovat správné GPS souřadnice.



Obrázek 9.4: Chybné polohování #2

Obrázek 9.4 odkazuje na tu stejnou testovací jízdu jako u obrázku 9.3. Na tomto obrázku je podrobně zvýrazněn příklad toho, jak moc se GPS mýlilo. Je ale třeba zdůraznit, že tyto body byly vytvořeny po 2 až 3 minutách od kompletního startu GPS. Bylo tedy GPS správně inicializováno? Byl nějaký problém s komunikací se satelity? Proč je výstup webové aplikace korektnější než tento? Na tyto otázky je jednoduchá odpověď. Modul je z výroby v základu nastaven pro použití japonského polohovacího systému QZSS, to je pro použití v Evropě při nejmenším nevyhovující už jen z prvních nasbíraných dat. Po zjištění této informace byl modul nastaven pro použití GPS, výsledky z jízd jen při použití GPS ale stále nebyly vyhovující. Bylo tedy rozhodnuto pro použití všech systémů, které modul podporuje, mezi tyto systémy patří GPS, GLONASS, BDS a QZSS. Po této konfiguraci modul vykazuje nejlepší možné výsledky, které je schopno generovat. Výsledky se nicméně odvíjí od celkové ceny, proto je tedy brát v potaz to, že polohování není nejpřesnější a některé polohy značek nejsou přesné, ani v okruhu 5 až 10 metrů.



Obrázek 9.5: Rušení signálu

Předchozí obrázek 9.5 je důležité vysvětlit. Tato jízda byla jednou z posledních, které byly provedeny. Zelené pruhy v obrázku označují skutečně projetou ulici, nicméně určené polohy značek, jsou na obou stranách - ulice byla projeta v obou směrech. Tato skutečnost při testování chtěla zamyšlení nad tím, proč tomu tak je. U ulice, která byla projeta, je dobré si povšimnout toho, že v ní vede kolejiště pro tramvaje a tramvaje, pro svoji funkčnost, potřebují i tramvajové vedení. Při konzultaci bylo shledáno závěru, že za rušení signálu nemůže jen stínění vysokých budov, které se také při testování objevovalo, ale i rušení, které je způsobeno vysokým napětím. Shledáno bylo tak hlavně z důvodu, že se nikde jinde toto rušení, mimo tramvajové pásy, neobjevilo. Pro vyřešení problematiky s GPS v několika aspektech by znamenalo pořízení lepšího modulu.

Závěr

Prvním z cílů práce bylo seznámit se s již vytvořenými systémy pro rozpoznávání dopravních značek v obrazu. Pro tento cíl bylo navrhnuto řešení pomocí detekce a následné klasifikace. Pro detekci byl použit model, který rozpoznává jen dopravní značky na pořízených snímcích. Model klasifikace byl natrénován za použití datasetu GTSRB, který obsahuje 43 druhů unikátních dopravních značek. Tento dataset byl ale ještě rozšířen o dalších 19 druhů dopravních značek, aby byl výstup z klasifikace v co největším spektru. Druhým cílem bylo navrhnout zařízení pro sběr geolokalizovaných snímků. Pro tento cíl byla vytvořena sestava, která jako počítač používá Raspberry Pi 4B s ArduCam kamerou a GPS modulem značky Waveshare. Sestava byla následně osazena do vymodelované krabičky, aby bylo zapříčiněno co nejlepšímu pohodlí při testování. V automobilu byla následně sestava připevněna pomocí magnetu na palubní desku před čelní sklo. Pro třetí bod byla implementována logika, které automaticky odesílá pořízené snímky přes SFTP na server, na kterém probíhá detekce a klasifikace. Výsledky z klasifikace jsou následně automaticky odesílány do Mongo databáze, která je na serveru také přítomna. Posledním bodem zadání bylo vytvořit webovou aplikaci, která bude zobrazovat statistické zhodnocení z výsledků testování. Aplikace využívá frameworku Flask pro Python. Je rozdělena na několik částí, dvě nejhlavnější části stránky jsou graf, který zobrazuje počet značek, pro jednotlivé třídy a mapa, na které jsou vidět body, které ukazují na klasifikované značky. Po kliknutí na libovolný bod se zobrazí informace o druhu dopravní značky a o datumu pořízení snímku. Aplikace je dostupná na webové adrese https://doprava.nti.tul.cz. Při testování bylo zjištěno horší funkčnosti modulu GPS z důvodu nízké ceny, stínění vysokých budov a přítomnosti vedení vysokého napětí, které pravděpodobně ovlivňovalo proces polohování.

Jako budoucí vylepšení tohoto řešení by se dalo považovat pořízení kvalitnějšího a lepšího GPS modulu, pro lepší polohování. Jako další z možných vylepšení by se dalo považovat zvětšení a upravení datasetu o další dopravní značky, nicméně je

tento proces hlavně náročný časově, protože data, která jsou dostupná na internetu, nejsou vždy vyhovující a individuální sběr dalších dopravních značek by zabral hodně času. Toto vylepšení by značně zlepšilo výstupy z klasifikátoru. Jako poslední možné vylepšení by se dalo považovat zlepšení funkčnosti algoritmu, protože momentálně klasifikátor ukládá do databáze i značky, které jsou fakticky na stejném místě, i když jejich snímky byly pořízeny v jiný čas, a to mírně degraduje výsledky statistik. Nicméně řešení této skutečnosti nebylo pro zadání podstatné.

Použitá literatura

[1] Cannon, Jason. *Linux Administration: The Linux Operating System and Command Line Guide for Linux Administrators*. CreateSpace Independent Publishing Platform, 2016. ISBN: 1523915951.

[2] Coursera, Inc. *SQL vs. NoSQL: The Differences Explained + When to Use Each*. 2023. URL: https://coursera.org/share/b38e4206718136ceed5505ff966f44de.

[3] *Folium - Folium 0.1.dev1+g3ccc47c documentation*. URL: https://python-visualization.github.io/folium/latest/.

[4] Chodorow, Kristina a Michael Dirolf. *MongoDB: The Definitive Guide*. 1st. O'Reilly Media, Inc., 2010. ISBN: 1449381561.

[5] IBM. *What is a neural network?* 2024. URL: https://www.ibm.com/topics/neural-networks.

[6] *L76K GPS Module - Waveshare Wiki*. URL: https://www.waveshare.com/wiki/L76K_GPS_Module.

[7] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[8] *PyMongo 4.6.3 documentation*. URL: https://pymongo.readthedocs.io/en/stable/.

[9] Sicha, Marek. *Detekce dopravních značek v reálném čase [online]*. Bakalářská práce. SUPERVISOR: Ing. Tomáš Bravenec. 2021. URL: https://theses.cz/id/61mpqv/.

[10] Šonka Milan, Václav Hlaváč a Roger Boyle. *Image processing, analysis, and machine vision*. 4th. Australia: Cengage Learning, 2015. ISBN: 9781133593690.

[11] Tan, Mingxing, Ruoming Pang a Quoc V. Le. *EfficientDet: Scalable and Efficient Object Detection*. 2020. arXiv: 1911.09070 [cs.CV].

[12] *Welcome to Bokeh documentation*. URL: https://docs.bokeh.org/en/latest/.

[13] *Welcome to Libcamera documentation*. URL: https://libcamera.org/docs.html.

[14] *Welcome to Paramiko's documentation! - Paramiko documentation*. URL: https://docs.paramiko.org/en/latest/.

[15] *Welcome to Picamera documentation*. URL: https://picamera.readthedocs.io/en/release-1.13/.

[16] *Welcome to Piexif's documentation! - Piexif 1.1.x documentation*. URL: https://piexif.readthedocs.io/en/latest/.

[17] Wikipedie. *Dopravní značka — Wikipedie: Otevřená encyklopedie*. [Online; navštíveno 6. 04. 2024]. 2023. URL: https://cs.wikipedia.org/w/index.php?title=Dopravn%C3%AD_zna%C4%8Dka&oldid=22471871.

[18] Wikipedie. *Raspberry Pi — Wikipedie: Otevřená encyklopedie*. [Online; navštíveno 3. 12. 2023]. 2023. URL: https://cs.wikipedia.org/w/index.php?title=Raspberry_Pi&oldid=23393438.

[19] Wikipedie. *Vídeňská úmluva o dopravních značkách a signálech — Wikipedie: Otevřená encyklopedie*. [Online; navštíveno 6. 04. 2024]. 2023. URL: https://cs.wikipedia.org/w/index.php?title=V%C3%ADde%C5%88sk%C3%A1_%C3%BAmluva_o_dopravn%C3%ADch_zna%C4%8Dk%C3%A1ch_a_sign%C3%A1lech&oldid=23455287.

[20] Zhao, J.D., Z.M. Bai a H.B. Chen. „Research on Road Traffic Sign Recognition Based on Video Image“. In: (2017), s. 110–113. DOI: 10.1109/ICICTA.2017.31.

Seznam obrázků

Seznam tabulek

A Odkaz na GitHub repozitář

Pro získání zdrojových kódů můžete navštívit následující odkaz na GitHub:
https://github.com/filipkralson/BP_TUL2023-2024

Při přípravě textu byl využit program ChatGPT 3.5.