Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Systém chytrého parkoviště

Martin Franěk

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Martin Franěk

Systémové inženýrství a informatika
Informatika

Název práce

Systém chytrého parkoviště

Název anglicky

Smart Parking System

Cíle práce

Cílem práce je vytvořit prototyp chytrého parkoviště za využití Rasbpberry Pi a potřebných senzorů, které bude sloužit k monitorování obsazenosti parkovacích míst na parkovišti. Data budou následně vyobrazena na webové informační stránce.

Metodika

V této bakalářské práci budou popsány základní informace o chytrých parkovištích, jejich výhod a využívané čidla. V následující části budou popsány vlastnosti platforem Arduina a Raspberry Pi, a potřebných komponentů. V neposlední řadě představení samotné konstrukce sběrače dat s vlastním schématem zapojení a jednotlivými částmi kódu. Dále využití Raspberry Pi jakožto serveru, který bude sloužit k vyzobrazení dat z parkoviště na webové stránce.

Doporučený rozsah práce

30 – 40 stran

Klíčová slova

Rasbpberry Pi, Chytré parkoviště, Smart City, Webové rozhraní

Doporučené zdroje informací

BELL, C A. *Beginning sensor networks with Arduino and Raspberry Pi.* [New York, New York]: Apress, 2013. ISBN 1430258241.

DENNIS, A K. *Raspberry Pi home automation with Arduino : automate your home with a set of exciting projects for the Raspberry Pi!.* Birmingham: Packt Publishing, 2013. ISBN 978-1-78439-920-7.

UPTON, E. – HALFACREE, G. – GONER, J. *Raspberry Pi : uživatelská příručka.* Brno: Computer Press, 2013. ISBN 978-80-251-4116-8.

Předběžný termín obhajoby

2020/21 LS – PEF

Vedoucí práce

Ing. Marek Pícka, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 23. 2. 2021

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 23. 2. 2021

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 15. 03. 2021

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Systém chytrého parkování" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 15. 3. 2021 _____

Poděkování

Rád bych touto cestou poděkoval mému vedoucímu bakalářské práce Ing. Marku Píckovi, Ph.D. za jeho cenné rady a podnětné připomínky, které mi pomohly při tvorbě a psaní bakalářské práce.

Systém pro chytré parkování

Abstrakt

Tato bakalářská práce pojednává o možnosti využití informačních technologií v provozu parkovišť pro usnadnění vyhledávání volných parkovacích míst, která tím kromě úspory času může přispívat i k plynulosti provozu na parkovišti, ale také i k úspoře pohonných hmot. To může mít také pozitivní vliv na psychickou pohodu uživatelů aut při parkování, což v dnešní době působení mnoha psychických stresorů na lidi je nezanedbatelná výhoda.

V teoretické části je představen koncept Smart cities, tedy zavádění moderních technologií do řízení měst s cílem zlepšit kvalitu života a zefektivnit správu obcí. Dále se práce věnuje konkrétnímu prostředku Smart cities - konceptu chytrého parkování. Teoretická část popisuje již existující způsoby řešení těchto parkovišť a mini počítače, především počítač Raspberry Pi a ESP32. Dále popisuje jednotlivé komponenty, které byly využity k sestrojení systému chytrého parkoviště.

V praktické části je popsán proces tvorby prototypu systému chytrého parkoviště s využitím Raspberry Pi, ESP32 a potřebných senzorů. Zabývá se sestavením jednotlivých použitých komponentů, jejich instalací a uvedení do provozu. Získaná data budou následně vyobrazena na webové informační stránce, aby mohla sloužit k ideálnímu využití parkoviště.

Klíčová slova: Rasbpberry Pi, ESP32, Chytré parkoviště, Smart city, Webové rozhraní, IoE, Smart City

Smart Parking System

Abstract

This thesis concentrates on the possibilities of using information technology for the day to day use of car parks to make searching empty parking slots easier, which can contribute to traffic flow in the car park and help save both fuel and time. It can also have a positive effect on the mental well-being of car park users when parking, which is a significant advantage. Especially nowadays, when people are affected by many psychological stressors.

In the theoretical part, I present the concept of Smart cities, i.e. using modern technology in city management to improve the quality of life and to make the management more effective. Furthermore, the thesis deals with a specific tool of Smart cities - the concept of smart parking. The theoretical part also describes the already existing ways of designing these car parks, and minicomputers, primarily the computer Raspberry Pi and ESP32. The individual components used in the construction of a smart parking system are described in this part as well.

In the practical part, I describe the creation process of a smart car park prototype using Raspberry Pi, ESP32 and the needed sensors. It deals with the assembly of individual components used, their installation and commissioning. Gathered data will be depicted on a web information page so that it can be utilised in the ideal use of the car park.

Keywords: Rasbpberry Pi, ESP32, smart parking, Smart City, Web interface , IoE, Smart City

Obsah

Seznam obrázků

Seznam tabulek

Seznam použitých zkratek

IoE	Internet of Everything
SC	Smart Cities
RFID	Radio Frequency Identification
GPIO	General-purpose input/output
OS	Operační systém
SD	Secure Digital
HDD	Hard Disk Drive
SSD	Solid-state drive
WiFi	Wireless LAN, WLAN
IDE	Integrated Development Environment
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
PHP	Hypertext Preprocessor
SQL	Structured Query Language
RGB	red green blue

1 Úvod

Internet věcí je pro mnoho běžných lidí neznámým pojmem i přes to, že se s ním obyvatelé vyspělých zemí setkávají na denní bázi. S konceptem internetu věcí je možné se setkat například při používání chytrých domácích spotřebičů, v automobilech, v domácích meteostanicích, v automatických krmítkách pro domácí zvířata, v chytrém osvětlení ve městě a na chytrých parkovištích.

Každý, kdo využívá automobilovou dopravu, ví, že najít parkovací místo může být opravdu složité. Zvláště pokud se jedná o větší města, ve kterých je vysoká hustota zalidnění, a tím pádem i větší množství aut, která je potřeba někde zaparkovat. Například na studentském parkovišti v areálu České zemědělské univerzity v Praze se snad každý student dopravující se na univerzitu autem setkal s problémem najít volné parkovací místo. Města se snaží tento problém řešit pomocí tzv. chytrých parkovišť, která využívají moderní technologie připojené k počítačové síti.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem práce je vytvořit prototyp chytrého parkoviště za využití Rasbpberry Pi a potřebných senzorů, které bude sloužit k monitorování obsazenosti parkovacích míst na parkovišti. Data budou následně vyobrazena na webové informační stránce. Výsledkem práce je zhodnocení účinnosti vytvořeného systému pro chytré parkoviště spolu s návrhy na jeho vylepšení.

2.2 Metodika

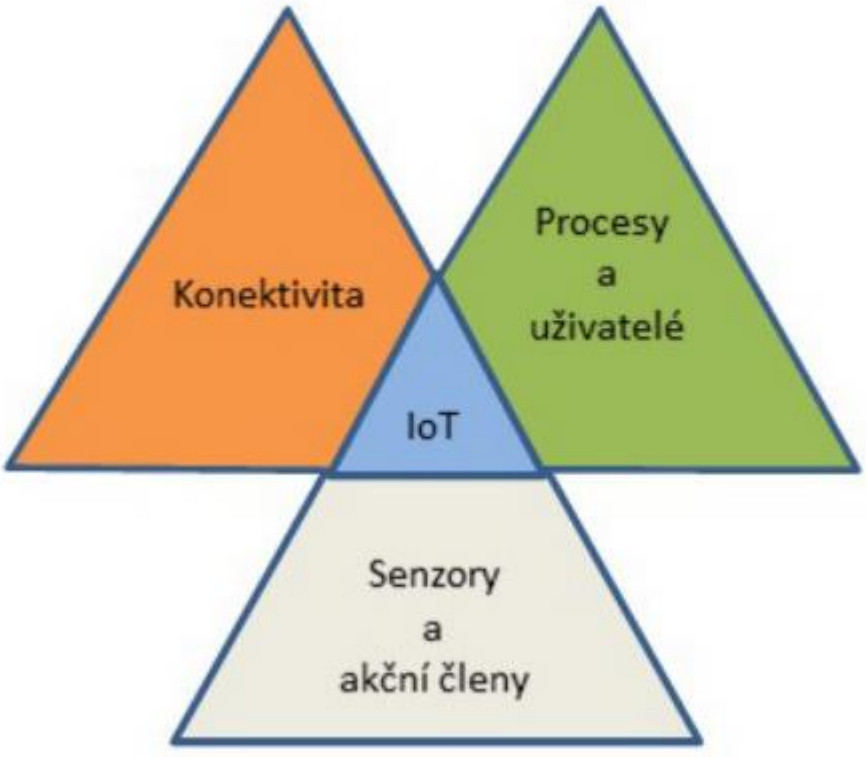
V této bakalářské práci jsou popsány základní informace o chytrých parkovištích, jejich výhody a využívaná čidla. Zároveň představuje jednotlivý hardware a software, který byl pro tento účel použit. Následující část popisuje vlastnosti platforem ESP32, Raspberry Pi a potřebných komponentů. Praktická část práce představuje postup samotné konstrukce sběrače dat s vlastním schématem zapojení a jednotlivými částmi kódu. Dále také vysvětluje využití Raspberry Pi jakožto serveru, který bude sloužit k vyobrazení dat z parkoviště na webové stránce. Po přípravě a sestrojení celého systému je systém testován v reálném prostředí. Výsledky testování jsou zaznamenávány, analyzovány a poté vyhodnoceny. Na základě získaných dat jsou navržena další možná řešení a vylepšení v systému chytrých parkovišť.

3 Teoretická východiska

3.1 Internet věcí (IoT)

„V současnosti umožňuje informační architektura založená na internetu výměnu služeb a zboží mezi veškerými prvky, zařízeními a objekty připojenými k síti. IoT využívá síťové připojení předmětů každodenní potřeby, které bývají často vybaveny určitým druhem inteligence. V tomto kontextu může být internet platformou, jejímž prostřednictvím různá zařízení elektronicky komunikují a vyměňují si informace s okolním světem. Na IoT tedy můžeme nahlížet jako na skutečnou evoluci všeho, co známe pod názvem internet – s mnohem větší vzájemnou konektivitou, lepším zpracováním informací a dokonalejšími inteligentními službami.“ (22, strana 6)

Obrázek 1 - Koncept IoT (22, strana 13)



Zdroj: (22)

IoT se skládá ze senzorů a akčních členů, jež odesílají a zároveň přijímají specifické údaje skrze konektivitu s dalšími zařízeními. V Cloudu se tyto informace

zpracují a zobrazí uživateli nebo jsou na základě těchto informací provedeny specifikované akce. (22)

Příkladem je chytrá meteostanice v chytrém rodinném domě, kdy její senzory měří teplotu, vlhkost vzduchu, tlak a rychlost větru. Tato naměřená data odešle přes bezdrátovou konektivitu do serveru, ten data zpracuje a na základě získaných informací o počasí nejenže ukáže venkovní teplotu, ale i dokáže například vypnout topení.

3.2 Smart cities

„Pojmem Smart Cities rozumíme koncept strategického řízení města, resp. obce nebo regionu (pro jednoduchost dále pouze„Smart Cities“, „koncept SC“, „SC“ bez dalšího rozlišení). Primárním cílem SC je zajištění kvalitního života obyvatelům, kdy jsou jako nástroj využívány moderní technologie pro ovlivňování kvality života ve městě, a následně k dosahování hospodářských a sociálních cílů města. Přitom dochází k synergiím mezi různými aktivitami a veřejnými službami, díky nimž město funguje – především doprava, logistika, bezpečnost, energetika, správa budov, atd.“ (20, strana 4)

Vzhledem k tomu, že do měst se stěhuje více a více lidí, města se musejí vypořádat s problémy jako například zvýšená hustota obyvatelstva nebo zvýšená hustota dopravy, nedostatečná místa k parkování, nedostatečná infrastruktura, bezpečnost a jiné. Koncept smart cities, v tomto případě konkrétně digitalizace, poskytuje efektivní nástroj, se kterým se dají tyto a další problémy alespoň částečně řešit. Města by mohla s využitím IoT zlepšovat systémy podle informacích získaných z různých senzorů (např. chytré řízení svozu odpadu ve městě by mohlo využívat ultrasonické senzory pro měření plnosti kontejneru). Ty jsou následně využity pro automatizované procesy. (21)(24)

3.2.1 Smart city v praxi

Smart city v praxi je portál sdružující informace od konkrétních měst na území České republiky, která již koncept smart city skutečně v praxi využívají. Nabízejí konzultace městům, která se o koncept zajímají, a pomáhají jim najít vhodná řešení s využitím IoT. Vydávají také články v médiích o konkrétních realizovaných projektech. (24)

3.2.1.1 Praha – řízení svozu odpadu s využitím IoT

Tento inovativní projekt dynamického svozu odpadu města má za úkol umožnit městu efektivněji plánovat svoz odpadu díky chytrým kontejnerům. Pravidelná kontrola automatizovaných svozových tras umožňuje reagovat na změny v produkci odpadu a infrastruktuře města. (24)

Obrázek 2 - Chytré kontejnery

Zdroj: (32)

Technické řešení se skládá ze třech hlavních částí. První část jsou senzory v kontejnerech kontrolující zaplněnost jednotlivých kontejnerů rozmístěných po městě. Druhá část je online sytém (Smart Waste Management System). Ten zpracovává data z kontejnerů a další data o popelářských vozech. Na základě zpracovaných dat například plánuje cesty jednotlivým popelářským vozům, aby byly co nejefektivněji využity. Třetí část je aplikace přes mobilní telefony dostupná pro občany (Citizen App). Podle ní si například obyvatel města za pomoci telefonu může naplánovat místo vyhození tříděného odpadu tak, aby nedošlo k situaci, kdy až na místě zjistí, že daný kontejner na tříděný odpad je již plný.

Za projektem stojí firma Sensoneo. Sansoneo nabízí v současnosti řešení odpadového hospodářství ve 40 různých zemích. (24)

3.3 Chytrá parkoviště

Jako chytrá parkoviště jsou označovaná ta parkoviště, která využívají různé způsoby pro sledování obsazenosti parkovacích míst a tím usnadňují uživatelům hledání volných míst k parkování.

3.3.1 S-oil Kampaň

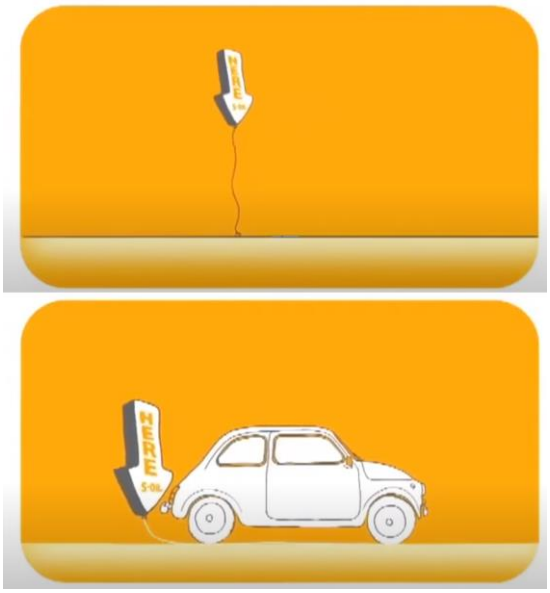
S-oil je ropná a rafinérská společnost, která má sídlo ve městě Soul v Jižní Koreji. Vydala reklamní spot poukazující na problém v přelidněném městě, kde je i nadmíra aut, a tím i velký problém s parkovacími místy. Společnost zjistila, že průměrně ujede obyvatel Soulu 500 metrů denně při hledání parkovacího místa, což je skoro 15 km za měsíc.

Obrázek 3 - S-oil kampaň

Zdroj: (32)

Pro natáčení reklamy uspořádala společnost S-oil jeden den na velkém parkovišti zajímavou kampaň a umístila na každé parkovací místo létající balonky, pomocí kterých řidiči snadněji nalezli volné parkovací místo. Tím se společnost pokusila ušetřit nejen pohonné hmoty, ale i čas řidičů. Balonky se pohybovaly nahoru a dolů na jednoduchém principu zkracování či prodlužování provázku, jak je vyobrazeno na obrázku 4. Jedná se o jednoduché řešení, které se dá považovat za předchůdce toho, čemu dnes říkáme chytré parkoviště.

Obrázek 3 - S-oil nákres



Zdroj: (32)

3.3.2 **Car tower**

Car tower je chytré řešení městského parkování. Jedná se o vertikálně rotační parkovací systém, který svojí rozlohou zabere jen o málo více, než dvě klasická parkovací místa. Díky Car tower však v tomto prostoru může být vybudováno vertikálně rotační parkoviště až pro 16 automobilů.

Obrázek 4 - Car tower

Zdroj: (33)

Jedná se o český výrobek, který byl v České republice také certifikován. Instalace probíhá bez větších stavebních úprav a také lze stavbu rozložit a opět složit na jiném místě. Uváděná životnost je 25 let. Provozní teploty mají velké rozmezí a to od -40°C až do +45°C.

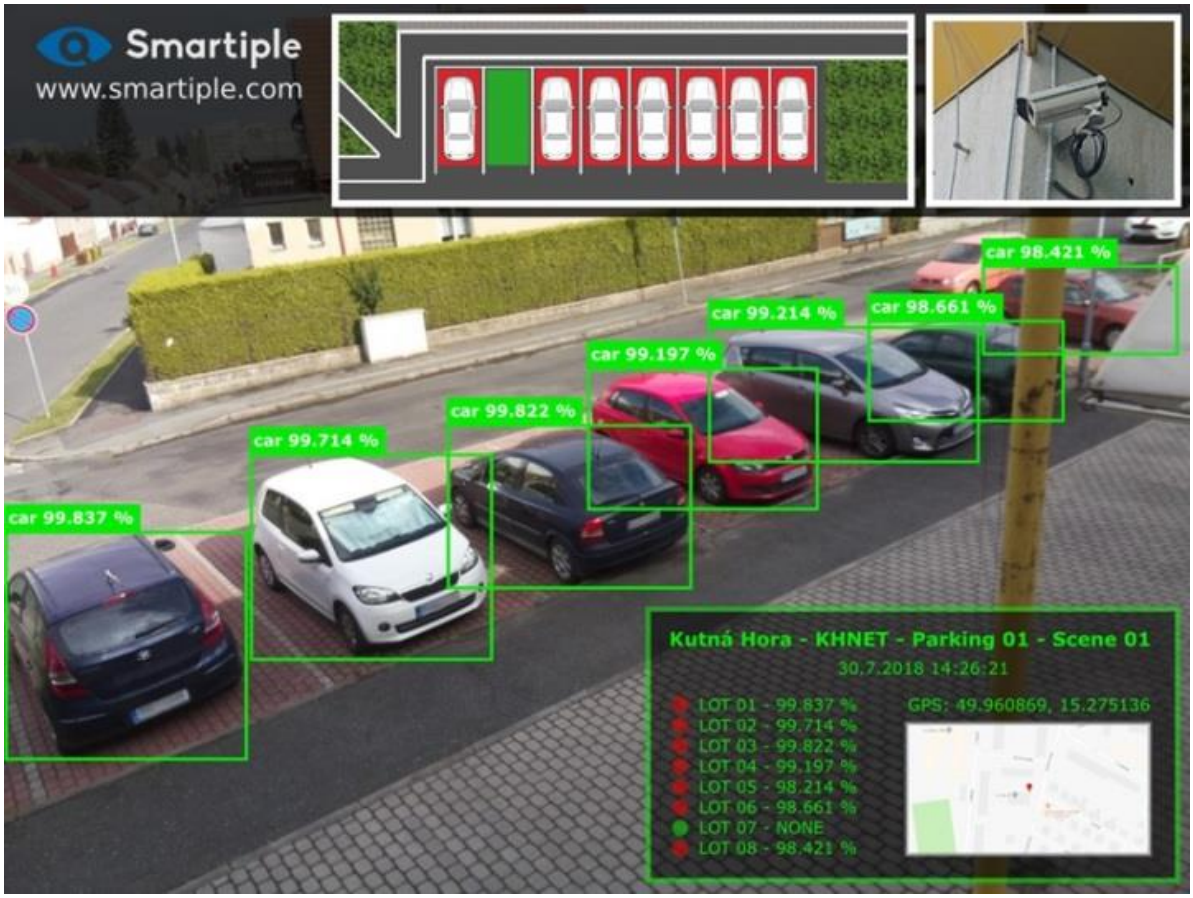
Z pohledu IT využívá tento systém pro identifikaci uživatele karty s RFID čipy. RFID je technologie identifikace uživatele na principu rádiové frekvence. Funguje podobně jako čárové kódy, tedy dokáže na krátkou vzdálenost předat bezkontaktně data. Rezervační systém je dostupný pomocí webové aplikace, kde si lze parkovací místo předem rezervovat a také zaplatit. (11)

3.3.3 **Parkinto**

Parkinto je unikátní projekt chytrého parkoviště, který detekuje obsazenost parkoviště pomocí umělé inteligence. Umělá inteligence ze záběrů z jedné či více kamer dokáže sama rozpoznat, jestli jsou určitá parkovací místa obsazená. A to vše s úspěšností 99.7 %, aktualizovanými informacemi každou jednu sekundu a s možností detekovat až 90 míst na jednu kvalitní kameru. Nevýhodou tohoto systému může být špatná umístitelnost kamer například v garážích u obchodního centra. (8)

Toto řešení použité v praxi je možné najít například i u nás v České republice v Kutné hoře. A to konkrétně před budovou spolku KHnet.info v ulici Havířská stezka. Na parkovišti je umístěna kamera a mikropočítač, který se stará o vyhodnocení obsazenosti parkoviště za pomoci umělé inteligence. Následně informace posílá na webový server, kde uživatel může vyčíst obsazenost daného parkoviště.

Obrázek 5 - Parkinto



Zdroj: (35)

3.4 Mini počítače

Mini počítače jsou velmi rozměrově malé počítače, přibližně velikosti platební karty.

3.4.1 Raspberry Pi

Raspberry Pi je jeden z nejrozšířenějších programovatelných mini počítačů (velikost kreditní karty) založených na ARM architektuře. První verze byla vydána v roce 2012 ve Velké Británii pod společností **The Raspberry Pi Foundation.** Tento počítač se stal populárním zvláště v odvětví automatizace a IoE mezi domácími kutily a fanoušky elektroniky a to především pro jeho možnosti rozšíření o různé moduly a čidla. (1)
Nabídka rozšiřujících modulů (např. výpočetní moduly, kamery, obrazovky LED a další)

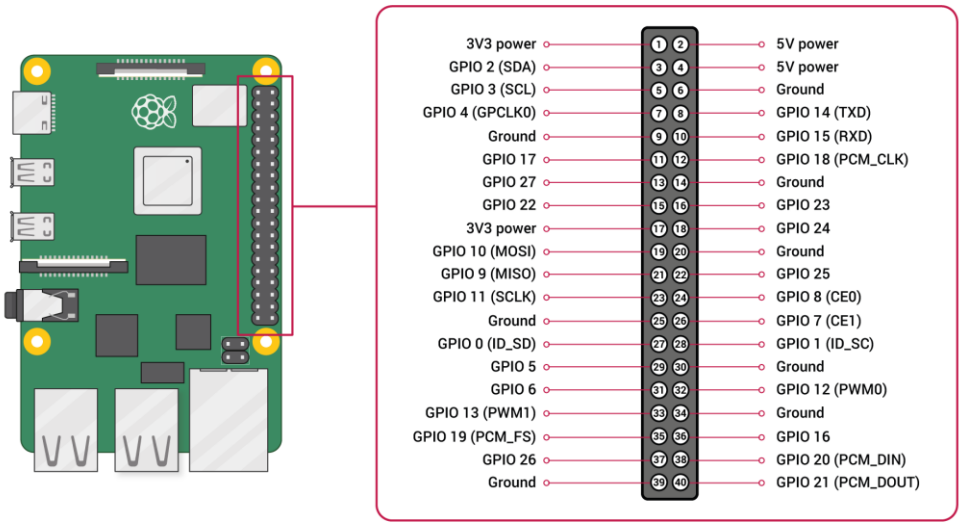
a různých čidel (vlhkosti, světla) se od vydání první verze velice rozrostla. I nadále je tato nabídka rozšiřována o nové technologie, například nedávno byl vydán modul pro otisk prstu.

Pro Raspberry Pi je výchozí operační systém Rasbian, který je doporučován officialně výrobcem. Avšak díky ARM architektuře lze do Raspberry Pi nainstalovat téměř jakoukoliv distribuci Linuxu. Na novější výkonné verze lze nainstalovat i například Windows 10 IoT Core nebo upravenou verzi Androidu.

3.4.2 **Patice P1**

Jedna z hlavních vlastností Raspberry Pi je patice P1. Tato patice se skládá z 46 pinů (viz. Obrázek 5). Verze před rokem 2014 měly pouze 26 pinů. Patice P1 dělá Raspberry Pi univerzálním, protože umožňuje připojit externí senzory, moduly a jiné prvky. (2)

Obrázek 6 - Raspberry Pi



Zdroj: (3)

Pořadí pinů na patici je velmi důležité. Piny můžeme rozdělit do tří následujících kategorií.

a) GPIO piny (general-purpose input/output) lze softwarově nastavit jako vstup nebo výstup. Při použití pinu jako vstup čte 3.3 V jako binární jedničku a 0 V jako binární nulu. Při výstupu naopak posílá 3.3V nebo 0V do připojených

periferiích. Na tomto principu lze programem ovládat moduly, nebo číst data ze senzorů připojených k GPIO pinům. Těchto pinů je na patici 27.

b) **VVC piny** jsou využívány jako zdroj stálého napětí. Na desce jsou dohromady čtyři takovéto piny. Dva poskytují 3,3 V a dva 5 V. Tyto piny jsou neprogramovatelné.
c) **Ground piny** jsou využívány jako uzemnění potřebné k uzavření elektrického obvodu. Tyto piny jsou taktéž neprogramovatelné. (2)

3.4.3 **Jiné alternativy mini počítačů**

3.4.3.1 Banana Pi

Banana Pi je minipočítač velice podobný Raspberry Pi jak vzhledem, tak architekturou. Banana Pi však disponuje lepším procesorem ARM Cortex-A7 s integrovaným grafickým čipem. Na Banana Pi není možné nahrát OS nebo uložit nějaká data, protože samo o sobě nedisponuje integrovaným čipem s pamětí. Aby bylo možné ukládat data, je nutno připojit SD kartu stejně jako u Raspberry Pi, nebo lze použít SATA konektor pro připojení klasického HDD nebo SDD disku. Raspberry Pi SATA konektor nemá a je stále vázané výhradně na velikost paměti SD karty.

3.4.3.2 Arduino

Vývoj Arduina započal již v roce 2005. Byl vytvořen pro studenty jako alternativa jednoduchého a cenově dostupného počítače. Na jeho vzniku se podíleli Massimo Banzi a David Cuartielles. Arduino je navrženo jako Open-source, tedy otevřený software, a je možné ho dále modifikovat nebo vyrábět upravené verze, aniž by byla porušena autorská práva. (4)

3.4.3.3 ESP32

ESP32 je levný a nenáročný systém na čipu. Systém na čipu je integrovaný obvod, který zastupuje většinu částí počítače nebo elektronického systému v jednom čipu. ESP32 disponuje WiFi a Bluetooth díky čemuž se stává čipem hojně využívaný pro IoT.

ESP32 má rozdílnou architekturu, na rozdíl od Raspberry Pi nebo Arduina na něm nelze spustit klasický OS (např. linux). Místo toho je nutné nahrát konkrétní program přes

Arduino IDE přímo do integrované paměti. A to pokaždé když chceme změnit princip použití ESP32. (27)

3.5 Využité technologie

Tato kapitola obsahuje popis jednotlivých softwarů, které byly použity v rámci výroby prototypu chytrého parkoviště.

3.5.1 HTML + CSS

HTML (Hypertext Markup Language) je značkovací jazyk využívaný při tvorbě webových stránek. Tyto stránky jsou propojeny odkazy. V HTML se vytváří obsah a struktura jednotlivých stránek. HTML se se stará o strukturu stránky, jako například, která část je nadpis, odstavec, vložení obrázku a co je odkaz a kam odkazuje. (27)

CSS (Cascading Style Sheet) je programovací jazyk, který určuje, jak by se prvky HTML webu měly skutečně vyobrazovat. Pomocí CSS upravujeme rozložení, vzhled, a formátování HTML stránky. (27)

3.5.2 PHP

PHP je dynamický skriptovací programovací jazyk, který využívá webový server pro ukládání zdrojových kódů. Skript je proveden na serveru a prohlížeči je zaslán výsledek.

V současné době je jedním z nejpopulárnějších programovacích jazyků, který je široce používán jak v komunitě otevřených zdrojů, tak v průmyslu k vytváření velkých webových aplikací a aplikačních rámců. (5)

3.5.3 Javascript

JavaScript je objektově orientovaný skriptovací jazyk podporovaný mnoha platformami. Jeho syntaxe je velmi podobná C/C++. Na webových stránkách je využíván pro vytvoření více interaktivního prostředí a příjemnější uživatelské rozhraní. JavaScript je stažen uživatelem společně s HTML a CSS konkrétní skript je spuštěn až v prohlížeči uživatele. Dále může webovou aplikaci obohatit o 2D i 3D grafiku a animace. JavaScipt v kombinace s HTML a CSS se využívá pro tvorbu moderních dynamických webů. (28)

3.5.4 **Node.js**

Node.js je open-source a cross-platform systém pro psaní back-endu. Programy jsou psané v programovacím jazyce JavaScript. Když Node.js provádí I / O operaci, jako je čtení ze sítě, přístup k databázi nebo souborovému systému, místo blokování vlákna a plýtvání cykly CPU čeká, Node.js obnoví operace, když se odpověď vrátí.To umožňuje Node.js zpracovat tisíce souběžných připojení s jedním serverem bez zavedení zátěže správy souběžnosti podprocesů, což by mohlo být významným zdrojem chyb. (29)

3.5.5 **MySQL**

MySQL je plně spravovaná databázová služba pro nasazení nativních cloudových aplikací. Je založena na jazyce SQL, který je užíván ke konfiguraci dat v databázi pomocí příkazů. Databáze obsahuje jednu nebo více tabulek, z nichž každá obsahuje záznamy nebo řádky. V těchto řádcích jsou různé sloupce nebo pole, která obsahují samotná data. (6)

3.5.6 **Python**

Python je interpretovaný, objektově orientovaný programovací jazyk s dynamickou sémantikou. Jeho integrované datové struktury v kombinaci s dynamickým zadáváním a dynamickým vázáním jej činí velmi atraktivním pro rychlý vývoj aplikací, stejně jako pro použití jako skriptovací jazyk nebo jako jazyk pro propojení existujících komponentů dohromady. Python podporuje moduly a balíčky, což zlepšuje modularitu programu a opětovné použití kódu. Interpret Pythonu a rozsáhlá standardní knihovna jsou k dispozici ve zdrojové nebo binární formě bez poplatků pro všechny hlavní platformy a lze je volně distribuovat. (7)

Rychlost interpretovaného programovacího jazyka Python v porovnání s rychlostí například s C/C++ nebo Java, je Python mnohem pomalejší. Interpretovaný jazyk je kvůli nutnosti interpreta, který zdrojový kód zavádí řádek po řádku, výrazně pomalejší oproti zmíněným kompilovaným jazykům. U kompilovaných jazyků je nejdříve nutné zdrojový kód přeložit překladačem do strojového jazyku, ve kterém program běží rychleji.

Předností Pythonu však je, že je silný v desktopových a serverových aplikacích. Dále se využívá jako skriptování na straně serveru. Naopak v mobilním odvětví se využívá programovací jazyk Python velmi zřídka. (8)

3.5.7 **Raspberry Pi OS**

Raspberry Pi OS je operační systém, který byl dříve známý jako Raspbian. Tento OS vychází z Debianu a byl upraven speciálně pro mini počítač Raspberry Pi . Raspberry Pi se používá kvůli vysoké optimalizaci právě pro ARM procesory, kterými je Raspberry pi osazen.

Pro instalaci Raspberry Pi OS je doporučován program Raspberry Pi Imager, který lze stáhnout na oficiálních webových stránkách Raspberry Pi. Tento program je velmi uživatelsky příjemný, protože obsahuje pouhá tři tlačítka. Jedno z tlačítek slouží pro zvolení ISO s operačním systémem, druhé pro vybrání SD karty, na kterou je potřeba provést instalaci, a třetí pro spuštění instalace. Pro instalaci obrazu na SD kartu není tedy potřeba žádných pokročilých znalostí. (13)

3.5.8 **Apache**

Apache je softwarový server. Apache server je Open source program, který podporuje Windows, Linux, macOS a další platformy. Dohromady s PHP a MySQL se jedná o nejvíce používané programy k tvoření webových stránek. Apache server ale podporuje i jiné databáze a programovací jazyky. Za pomoci externích modulů lze Apache server rozšiřovat o další funkce.

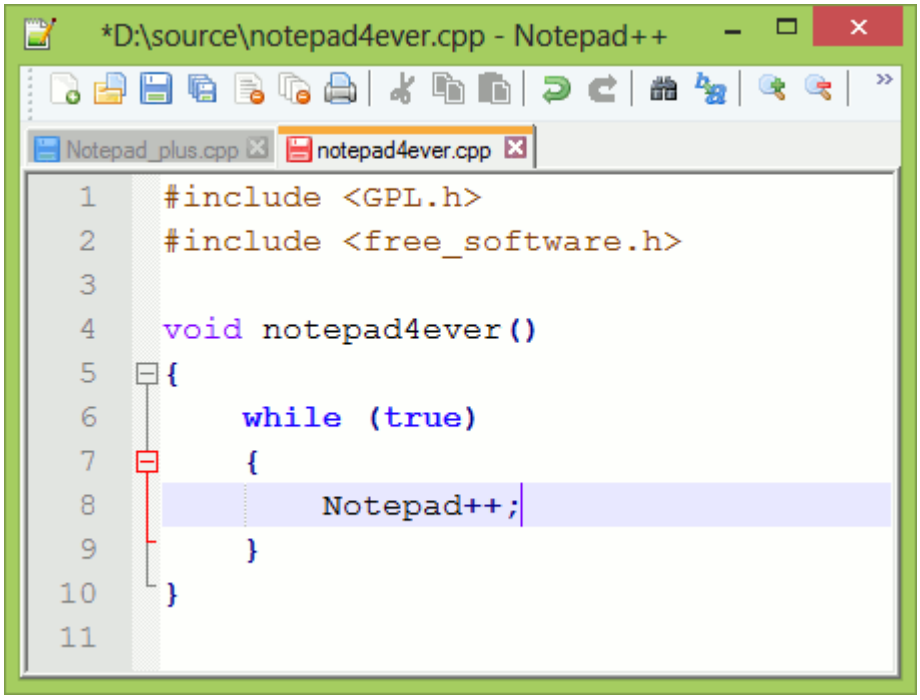
3.5.9 **phpMyAdmin**

PhpMyAdmin je open-source nástroj napsaný v programovacím jazyku php převážně pro operace s databazí MySQL přes web. V grafickém webovém uživatelském prostředí lze provádět rozsáhlé operace právě v MySQL nebo MariaDB. Zároveň však v grafickém prostředí, které je velmi intuitivní, lze stále spouštět klasické SQL příkazy.

Aby tento projekt mohl zaujmout co největší počet lidí, byl přeložen do 72 lingvistických jazyků. I díky tomu tento projekt má velmi stabilní a flexibilní codebase s historií. PhpMyAdmin byl také oceněn cenami jako Best PHP-Tool , Best PHP Application nebo Best Tool or Utility for SysAdmins.(14)

3.5.10 **Notepad++**

Obrázek 7 - Notepad++



Zdroj: (34)

Notepad++ lze přirovnat k programu *Poznámkový blok*, který zná každý uživatel Windows. Notepad++ je vhodný jako náhrada za právě zmiňovaný *Poznámkový blok,* ale i jako editor zdrojových kódů. Jako editor zdrojových kódů ho lze využít díky podpoře barevného rozlišení syntaxe a to ve všech rozšířenějších jazycích (např. C, C++, Java, HTML, PHP, Javascript, CSS a SQL). Notepad++ neumí napovídat v psaní programu ani najít chyby v programu, pouze v syntaxi. (30)

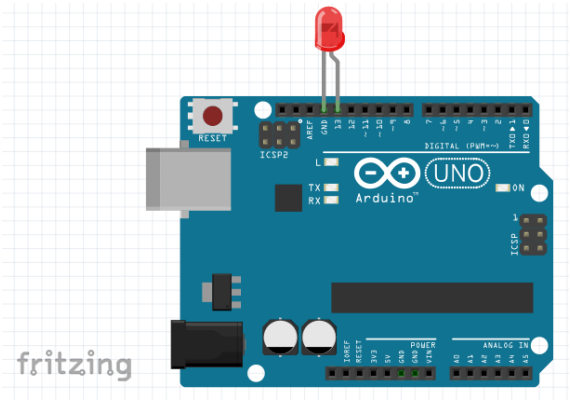
3.5.11 **Arduino IDE**

Arduino IDE je vývojové prostředí podporující programovací jazyky C a C++. Je využíváno pro psaní a následné nahrání programu do mini počítače. Původně šlo nahrávat programy pouze do Arduino, avšak s rostoucí popularitou Arduino ostatní výrobci začali implementovat do Arduino IDE různé knihovny a překladač. Dnes už lze využívat Arduino IDE téměř pro všechny jednodeskové mini počítače.

3.5.12 **Fritzing**

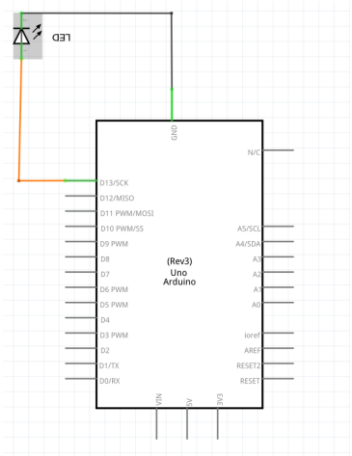
Fritzing je open-source nástroj pro tvorbu dokumentace prototypů, navrhování schémat elektronických obvodů, rozvrhování a výrobu desek plošných spojů. Vývojové prostředí Fritzingu se skládá ze tří pohledů na zapojení elektronického obvodu. Na příkladu, který je vyobrazen na obrázku 7, je pohled na příkladový projekt se zapojením arduina s blikající ledkou. Na obrázku 7 je vidět pohled ve vývojovém prostředí na montážní desku. V této části je možné navrhnout zapojení všech možných modulů do Raspberry Pi (arduina, Banana Pi atd.). Na obrázku 8 je pohled na arduino s led, ale tentokrát zobrazuje schéma zapojení, ve kterém je možné zkontrolovat propojení a jednotlivé součástky. Na obrázku 9 je pohled na desku plošných spojů, kde je možné navrhnout desku pro osazení jednotlivých elektronických součástek. Fritzing umožnuje takto vytvořený návrh uložit a následně objednat. Desku plošných spojů pošle balíkem až k uživateli domů. Zásilka je odesílána z Německa do celého světa. Na obrázku 10 je pohled na vývojové prostředí s programovacím jazykem Processing. Konkrétně je vidět program na blikání ledky zapojené do arduina. (15)

Obrázek 8 - Arduino s LED diodou



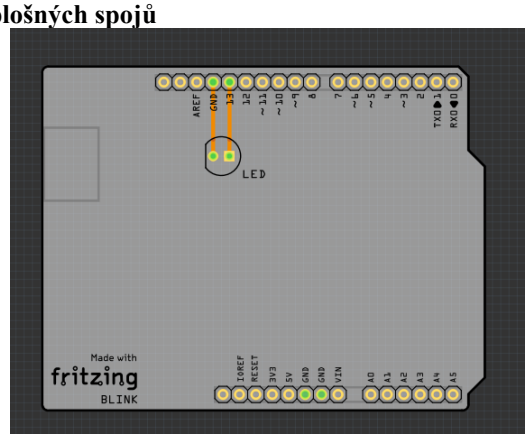
Zdroj: Vlastní zpracování

Obrázek 9 - Schéma zapojení



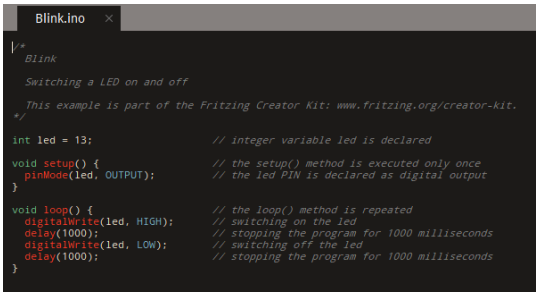
Zdroj: Vlastní zpracování

Obrázek 10 - Návrh desky plošných spojů



Zdroj: Vlastní zpracování

Obrázek 11 - Vývojové prostředí



Zdroj: Vlastní zpracování

3.6 Využitý hardware

V této části bakalářské práce najdeme jednotlivé hardwarové části, využité moduly a jejich porovnání.

3.6.1 Raspberry Pi 3 Model B v1.2

Obrázek 12 - Raspberry Pi 3 Model B v1.2

Zdroj: (36)

Raspberry Pi 3 Model B v 1.2 je mozkem celého prototypu chytrého parkoviště. Oproti starším verzím Raspberry Pi tento model disponuje WiFi modulem. Díky této vlastnosti je tedy vhodný pro tento prototyp chytrého parkoviště kvůli možnosti zřídit server pro webovou aplikace a vytvořit program, který pracuje s moduly napojenými na GPIO v jednom zařízení.

Konkrétní technické parametry Raspberry Pi 3 Model B v1.2, který byl využit v této bakalářské práci.

Tabulka 1 - Parametryy Raspberry Pi

Datum vydání	Únor 2016
Architektura	ARMv8-A (64/32-bit)
SoC	Broadcom BCM2837
Procesor (CPU)	1.2 GHz 64-bit quad-core ARM Cortex-A53
Video (GPU)	Broadcom VideoCore IV @ 250 MHz (BCM2837: 3D part of

	GPU @ 300 MHz, video part of GPU @ 400 MHz) OpenGL ES 2.0 (BCM2835, BCM2836: 24 GFLOPS / BCM2837: 28.8 GFLOPS) MPEG-2 and VC-1 (with license), 1080p30 H.264/MPEG-4 AVC high-profile decoder and encoder (BCM2837: 1080p60)
Paměť (SDRAM)	1 GB (sdílená s GPU)
USB 2.0 porty	4 (přes zabudovaný pětiportový USB hub; jeden USB port vnitřně propojen s ethernet portem)
Video vstup	15-pinový MIPI konektor kamerového rozhraní (CSI)
Video výstup	HDMI (rev 1.3 & 1.4), 14 HDMI rozlišení od 640×350 do 1920×1200 plus různé PAL a NTSC standardy, kompozitní video (PAL a NTSC) via 3,5 mm TRRS jack sdílený s výstupem zvuku, MIPI konektor rozhraní displeje (DSI)
Zvukový vstup	Přes I2C rozhraní
Zvukový výstup	Analogový (přes 3,5mm jack), digitální (přes HDMI), I2S
Interní paměť	MicroSDHC, USB Boot Mode
Integrovaná síť	10/100 Mbit/s Ethernet + WiFi 802.11n a Bluetooth 4.1
Nízkoúrovňové periferie	17× GPIO plus stejné funkce a HAT ID sběrnice
Jmenovitý výkon	300 mA (1,5 W) v průměru na volnoběh 1,34 A (6,7 W) při maximálním zatížení (připojený monitor, klávesnice, myš a WiFi)
Rozměry	85,60 mm × 56,5 mm × 17
Hmotnost	45 g

Zdroj: (36)

3.6.2 **ESP32**

Obrázek 13 - ESP32

Zdroj: (37)

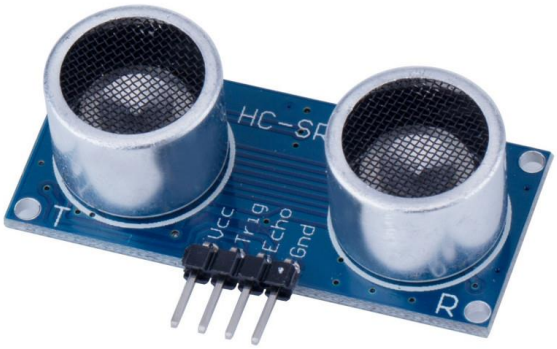
ESP32 je mini počítač hojně využívaný pro IoT projekty převážně kvůli malým rozměrům a také kvůli integrované WiFi a Bluetooth. Na rozdíl od předchůdce ESP8266 má ESP32 dokumentaci nejen v čínštině, ale i v angličtině. Rozšířené možnosti připojení k ESP32 a jen nepatrné zvýšení ceny oproti předchůdci vedlo k rychlé adaptaci v IoT zařízeních.

Tabulka 2 - Parametry ESP32

POČET JÁDER	2
Architektura	32 bits
WiFi	IEEE802.11 b/g/n
Bluetooth	Ano - classic & BLE
RAM	520KB
Flash	Extern QSPI – 16MB
GPIO	22

3.6.3 **Ultrazvukový senzor HC-SR04**

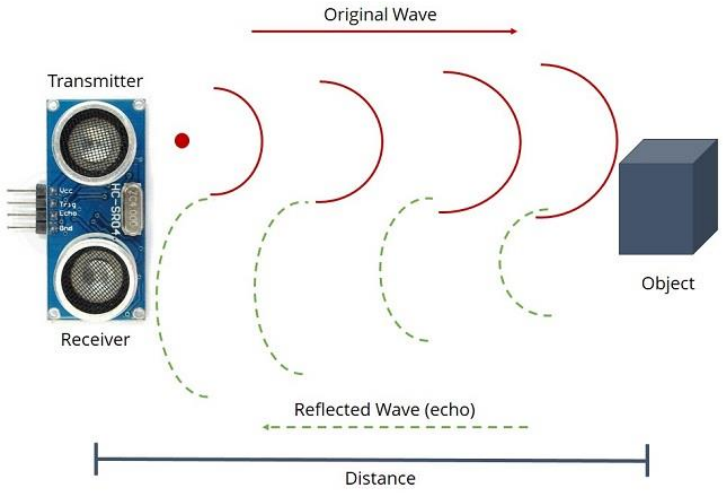
Obrázek 14 - Ultrazvukový senzor HC-SR04



Zdroj: (17)

HC-SR04 je ultrazvukový senzor, který funguje na principu vydávání pro nás neslyšitelného (ultrasonického) zvuku, a následného zachycení odrazu těchto zvuků od objektu před senzorem. (16) Konkrétně transmitter na pinu Trig vyšle signál o vysoké frekvenci. Tento signál narazí na objekt, od kterého se odrazí zpět. Echo signálu je po odražení následně zachyceno druhým transmitter na pinu. (16)

Obrázek 15 - Princip ultrazvukového senzoru



Zdroj: (16)

HC-SR04 snímá vzdálenosti od 2 do 400cm v efektivním úhlu <15 °, což je ideální vzdálenost pro detekování vozidla na parkovacím místě. Odchylka měření je ± 3 mm. (17)

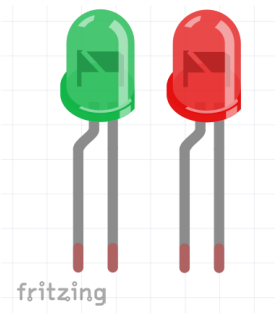
Tabulka 3 - Parametry HC-SR04

Pracovní napětí	5 VDC
Pracovní proud	15 mA
Pracovní frekvence	40 kHz
Maximální dosah	400 cm
Minimální dosah	2 cm
Efektivní úhel	15 °
Měřící úhel	30 °
Trigger Input Signal	10µS TTL pulse
Echo Output Signal	Input TTL lever signal and the range in proportion
Rozměry	45 × 20 × 15 mm

Zdroj: (17)

3.6.4 **Světelné diody 5mm**

Obrázek 16 - Světelná dioda



Zdroj: (18)

Vysoce svítivé diody červené a zelené barvy o průměru 5mm byly v projektu určeny pro indikace stavu volno–zelená a obsazeno–červená. (18)

Tabulka 4 - Parametry LED diod

Typ LED diody	Through-Hole
Velikost LED diody	5mm
Vlnová délka	525nm
Vyzařovací úhel	60~70°
Napětí v propustném směru Uf	3.2~3.6V
Proud v propustném směru If	20mA
Výrobce	Hebei

Zdroj: (18)

4 Vlastní práce

V teoretické části bakalářské práce již byly představeny jednotlivé software a hardware komponenty. Praktická část je zaměřena na vytvoření systému chytrého parkoviště s následným otestování jeho použitelnosti a efektivity. V kapitole 4.1 je popsán postup instalace potřebného softwaru pro webovou aplikaci. Následující kapitoly se zabývají sestavením jednotlivých komponentů představených v teoretické části a popisem mého zdrojového kódu, který byl naprogramován za účelem vytvoření systému chytrého parkoviště. Po vytvoření byl systém chytrého parkoviště podroben testování v reálném prostředí. Výsledky testování byly poté vyhodnoceny a jsou shrnuty v následujících kapitolách.

4.1 Příprava prostředí pro webovou aplikaci

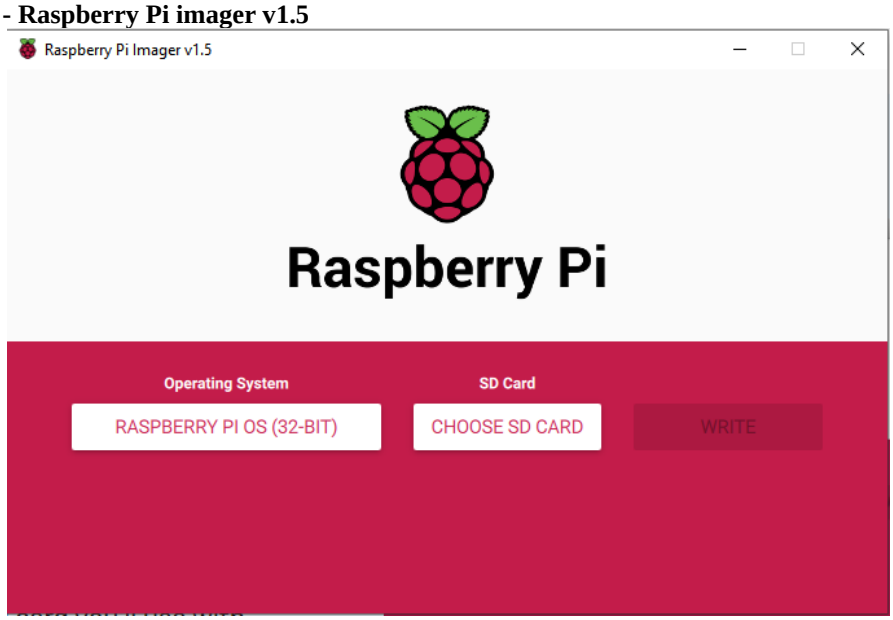
V této kapitole je popsána potřebná příprava a instalace programů pro vytvoření webového serveru na Raspberry Pi.

4.1.1 Instalace Raspberry Pi OS

Protože Raspberry Pi je dodávané bez OS, je potřeba jako první nainstalovat OS. Zvolil jsem doporučovaný OS od výrobce, Raspberry Pi OS (dříve znám pod názvem Rasbian).

Instalaci Raspberry Pi OS na SD kartu jsem provedl za pomoci oficiálního programu Raspberry Pi Imager v1.5 k tomuto účelu určený.

Obrázek 17 - Raspberry Pi imager v1.5



Zdroj: Vlastní zpracování

Tento proces je velmi jednoduchý. Stačilo vybrat námi zvolený obraz OS, v tomto případě Raspberry Pi OS (32-BIT), a zvolit SD kartu, na kterou proběhne instalace.

Po zapsání OS na kartu, což trvalo přibližně deset minut, jsem SD kartu vložil do Raspberry Pi a připojil potřebné periferie (monitor, klávesnici a myš).

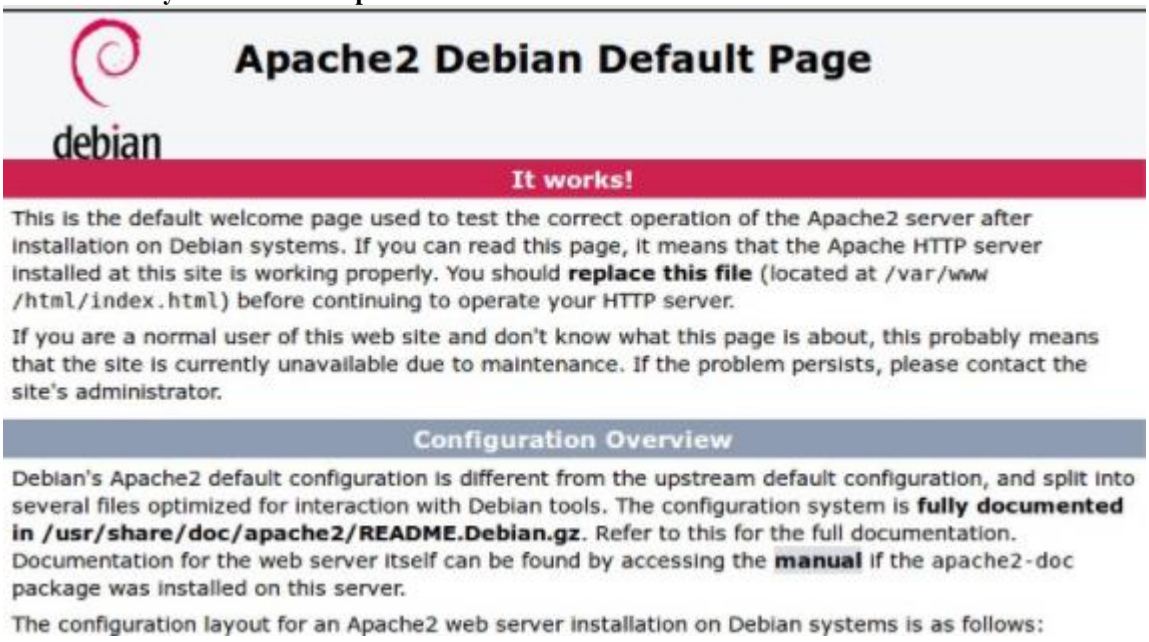
Při připojení správného zdroje napájení se Raspberry Pi samo spustí a načte OS z SD karty. Při prvním spuštění jsem byl vyzván k připojení k internetu, což jsem uskutečnil pomocí WiFi, kterou Raspberry Pi disponuje. Po úspěšném připojení k internetu se Raspberry Pi samo aktualizuje na nejnovější verzi OS. Po vytvoření uživatelského účtu OS naběhne a zobrazí uživateli klasickou plochu, která je velmi podobná ostatním linuxovým distribucím.

4.1.2 Instalace Apache

Protože v tom to projektu je Raspberry Pi použito převážně jako lokalní server pro hostování webové aplikace, bylo zapotřebí nainstalovat Apache http server.

Jako první je třeba aktualizovat balíčky pomocí příkazu *sudo apt update*. Poté za příkazem *sudo apt install apache2 -y* následuje instalace Apache2. Pro otestování funkčnosti serveru je potřeba přes prohlížeč v Raspberry Pi naštívit adresu *http://localhost/*. Pokud instalace proběhla v pořádku, zobrazí se obrazovka jako na obrázku 17.

Obrázek 18 - Výchozí stránka Apache



Zdroj: Vlastní zpracování

V tuto chvíli server již plně funguje a dokáže hostovat webové stránky psané v HTML. V */var/www/html/* lze nalézt *index.html,* který obsahuje výchozí stránku z obrázku 17.

To však pro tento účel nestačilo a dále bylo třeba nainstalovat PHP modul. To jsem provedl pomocí příkazu *sudo apt install php libapache2-mod-php -y.* Následně bylo opět potřeba otestovat funkčnost. Tu jsem otestoval pomocí příkazu *sudo rm index.html,* který odstraní *index.html,* a příkazu *sudo nano index.php,* který vytvoří *index.php.* Do tohoto souboru jsme pro otestování vložili jednoduchý kód *<?php echo "hello world"; ?>* .

A pro následné otestování jsem v prohlížeči navštívil *http://localhost/,* kde se zobrazila pouze prázdná bílá stránka s textem „hello world“, což značí, že mnou nainstalovaný Apache server v této fázi zvládá překládat i php.

4.1.3 Instalace MySQL

Pro ukládání dat do databáze bylo nutné nejdříve nainstalovat MySQL příkazem *sudo apt install mariadb-server.* Po úspěšné instalaci jsem nastavil heslo. Ve výchozím nastavení MySQL není nastavené žádné heslo, což znamená, že k serveru MySQL se lze připojit bez jakéhokoliv ověřování. Dalším logickým krokem bylo zvolení hesla a základní

nastavení MySQL serveru. Toto se nastavuje v instalaci nastavení pomocí příkazu *sudo mysql_secure_installation*. Tento příkaz spustí instalaci vyžadující vstup od uživatele, konkrétně právě zadání hesla a nastavení základních bezpečnostních protokolů.

4.1.4 **Instalace phpmyadmin**

Pro usnadnění práce s databází jsem nainstaloval grafické prostředí pro zprávu databáze phpmyadmin pomocí příkazu *sudo apt install phpmyadmin*. V průběhu instalace je zapotřebí po dotázání se na typ webového serveru, který využívám, zadat v tomto případě *apache2*. Následně jsem byl vyzváni ještě pro zadání hesla do rozhraní phpmyadmin.

Ve výchozím nastavení phpmyadmin rozhraní nenechá přihlásit při použití *root* uživatele, proto bylo nutné, abych vytvořil nového uživatele pro MySQL server.
Pro vytvoření nového uživatele je nejdříve nutné přihlasit se do rozhraní příkazového řádku MySQL pomocí *sudo mysql -u root -p*. Nového uživatele jsem vytvořil pomocí příkazu *GRANT ALL PRIVILEGES ON *.* TO 'jmeno'@'localhost' IDENTIFIED BY 'heslo' WITH GRANT OPTION;* ve kterém " jméno" bylo nahrazeno mnou zvoleným přihlašovacím jménem a " heslo" bezpečným heslem.

Dále jsem nakonfiguroval Apache2, abych přes něj mohl přistupovat do grafického prostředí phpmyadmin. K tomu bylo zapotřebí vložit na konec souboru *apache2.conf* vložit řádek *Include /etc/phpmyadmin/apache.conf*. To jsem provedl za pomoci příkazu *sudo nano /etc/apache2/apache2.conf*. a zmiňovaný řádek vložil nakonec souboru. Poté jsem pouze restartoval Apache příkazem *sudo service apache2 restart.* Po restartu jsem navštívil *http://localhost/phpmyadmin,* kde už se zobrazila přihlašovací stránka phpmyadmin, což potvrdilo úspěšnost nainstalování a propojení Apache,MySQL s phpmyadmin.

4.1.5 **Instalace Node.js**

Node.js jsem v této BP použil jako komunikační prostředek mezi senzory a uživatelem. Před samotnou instalací bylo potřeba přidat repositář od společnosti NodeSource, která se stará o správu nejaktuálnějších verzí Node.js. To jsem provedl pomocí příkazu *curl -sL https://deb.nodesource.com/setup_10.x | sudo bash -*. Po přidání repositáře jsem provedl instalaci příkazem *sudo apt install nodejs*.

4.2 Hardware

V této kapitole je popsán postup při přípravě hardwarových částí bakalářské práce.

4.2.1 Návrh zapojení

Před kompletací komponentů do funkčního celku jsem si vytvořil návrh zapojení v programu Fritzing. Jak se konkrétní komponenty zapojují, jsem vyčetl z dokumentací, které jsou obvykle k nalezení na produktové stránce daného e-shopu, odkud jsem komponenty objednával.

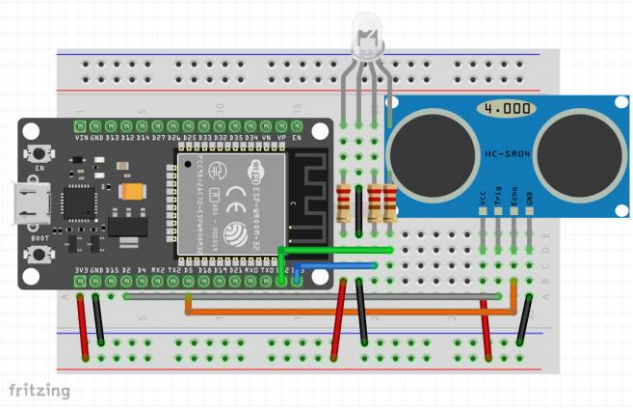
Tato část je velmi důležitá pro přehlednou správu zapojení GPIO pinů, protože jejich číselné označení je důležité při psaní programu.

4.2.1.1 Zapojení ESP32

Zapojení Ultrasonického senzoru a RGB led diody k ESP32 šlo provést přímo a nebylo potřeba použít převodníky.

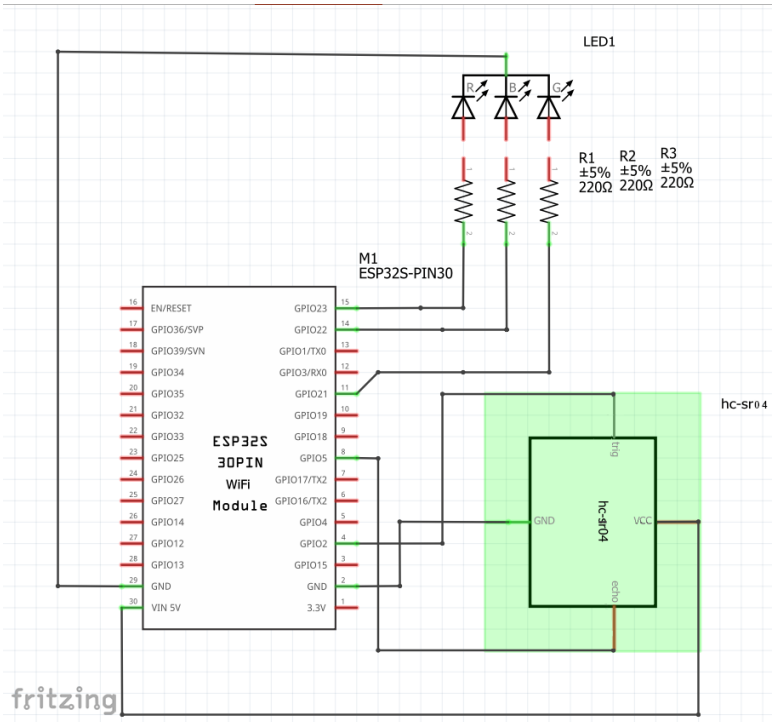
Na obrázku 18 lze vidět nepájivé pole, které jsem použil pro lepší přehled zapojení. Při kompletaci jsem nepájivé pole nepoužil a zapojil vše přímo, abych vytvořil co nejmenší produkt. Takto sestrojený produkt pracuje po připojení ke zdroji napájení bez intervence člověka.

Obrázek 19 - Návrh ESP32



Zdroj: Vlastní zpracování

Obrázek 20 - Elektrické schéma



Zdroj: Vlastní zpracování

4.3 Programování

Po úspěšné instalaci softwaru a zapojení hardwaru je zapotřebí vytvořit tři programy, jeden pro Raspberry Pi, druhý pro ESP32 a třetí pro webovou aplikaci. Všechny zdrojové kódy jsou v příloze A.

4.3.1 IDE

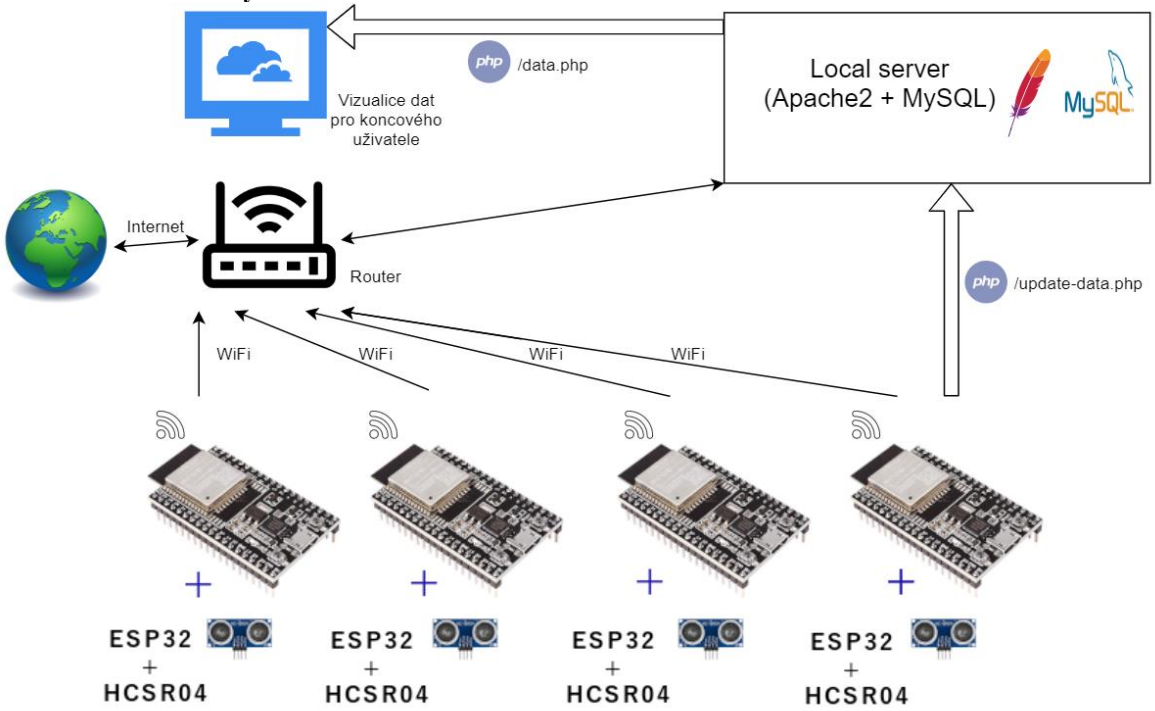
Zdrojový kód webové aplikace pro tuto bakalářskou práci jsem psal ve vývojovém prostředí Notepad++. Konkrétně všechen zdrojový kód v jazycích HTML, CSS, JavaScirpt, Node.js a MySQL

Pro napsání zdrojového kódu pro Raspberry Pi a ESP32 jsem použil Arduino IDE. Arduino IDE umí nativně pracovat s Raspberry Pi a všechny potřebné knihovny už obsahuje. Pro funkčnost s ESP32 bylo nutné nainstalovat potřebné knihovny z *https://dl.espressif.com/dl/package_esp32_index.json* .

4.3.2 Návrh systému

Na obrázku 20 je jednoduchý návrh komunikace v systému chytrého parkoviště.

Obrázek 21 - Návrh systému



Zdroj: Vlastní zpracování

Raspberry Pi je zde použito jako lokální server, na kterém běží potřebná webová aplikace, která dostává a následně zpracovává data z ESP32. Z ESP32 jsem vytvořil zařízení, které ke svému provozu potřebuje pouze WiFi připojení a připojení napájecího kabelu. Mimo napájecího kabelu je ESP32 je kompletně bezdrátové. Přes připojení WiFi komunikuje se serverem (Raspberry Pi) a posílá mu data. Uživatel si následně data zobrazí přes prohlížeč po zadání IP adresy Raspberry Pi.

4.3.3 Zdrojový kód ESP32

V této části bakalářské práce jsem popsal jednotlivé důležité části zdrojového kódu pro ESP32.

4.3.3.1 Získání dat ze senzoru

Obrázek 22 - ESP32 Získání dat ze senzoru

```
void readData() {
  digitalWrite(trigPin, LOW);  // uvolni piny
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);// na 10 microsekund nastaví HIGH
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);// čtě echoPin, vrací cask v mikrosekundách (jak dlouho)
  distance = duration * 0.034 / 2;  // vypocet vzdalenosti podle rychlosti zvuku
  Serial.print("Distance: ");
  Serial.println(distance);
  if(distance>50){
    availability="true";
    led("Green");
    }else{
    availability="false";
    led("Red");
      }
  delay(1000);
}
```

Zdroj: Vlastní zpracování

Na obrázku 21 je část zdrojového kódu pro ESP32, konkrétně část, ve které se zpracovávají data ze ultra sonického senzoru.

Funkce *readData()* vyšle signál na *trigPin* senzoru. Ten vydá ultrasonický signál a následně uloží do proměnné *duration* dobu, za kterou se odražený signál vrátil. Když už mám dobu trvání a znám rychlost zvuku, mohu už snadno vypočítat vzdálenost mezi senzorem a objektem, od kterého se zvuk odrazil. Následně jen výsledek vydělím dvěmi, protože časový údaj obsahuje cestu k objektu a zpět. Poté probíhá kontrola podmínky, jestli je objekt blíže než 50 cm nebo dále. Následně podle toho rozsvěcí zelenou nebo červenou led diodu a ukládá informaci o obsazenosti daného parkovacího místa.

4.3.3.2 Předání dat Raspberry Pi

Obrázek 23 - Předání dat

```
void uploadData() {
  if (WiFi.status() == WL_CONNECTED) {  //zkontroluje WiFi připojení
    HTTPClient http;

    http.begin(serverName); //vytvoří spojení s Raspberry Pi serverem

    http.addHeader("Content-Type", "application/x-www-form-urlencoded"); // nastavení headeru

  // příprava HTTP POST
    String httpRequestData = "api_key=" + apiKeyValue +
                             "&sensor=" + sensorName  +
                             "&location=" + sensorLocation +
                             "&availability=" + availability +
                             "&realValue=" + distance +
                             "&ipAdress=" + ipadress +
                             "&senzorID=" + mac +
                             "";
    Serial.print("httpRequestData: ");
    Serial.println(httpRequestData);

    int httpResponseCode = http.POST(httpRequestData); // pošle HTTP POST request

    if (httpResponseCode > 0) {
      Serial.print("HTTP Response code: ");
      Serial.println(httpResponseCode);
      String payload = http.getString();
      Serial.println(payload);
    }
    else {
      Serial.print("Error code: ");
      Serial.println(httpResponseCode);
    }

    http.end();
  }
  else {
    Serial.println("WiFi Disconnected");
  }

}
```

Zdroj: Vlastní zpracování

Funkce *uploadData()* začíná kontrolou připojení k WiFi, navázáním spojení se serverem běžícím na Raspberry Pi a následně nastavení hlavičky HTTP POST.

Data odeslaná v HTTP POST:

- api_key – Klíč pomocí kterého server ověří že přišel HTTP POST vážně z mého ESP32, a ne od někoho kdo se serveru snaží poslat falešná data.
- Sensor – Jméno senzoru v tomto případě *HC-S04*.

- location – Místo, kde je senzor umístěň. např. Parkoviště 1.
- availability – informace o dostupnosti daného parkovacího místa.
- realValue – Skutečná hodnota načtená ze senzoru. (Pro ladění)
- ipAdress – IP adresa pod kterou je připojeno ESP32 k síti.
- senzorID – Mac adresa ESP32, kterou jsem použil jako univerzální ID.

Dále program HTTP POST odešle a zkontroluje, že nedošlo k žádným chybám.

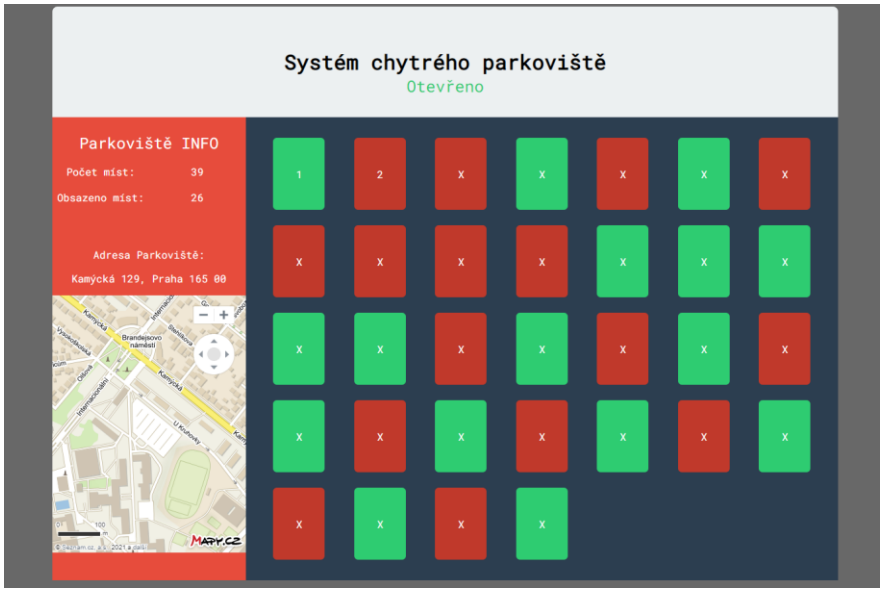
4.3.4 Zdrojový kód webové aplikace

Tato část bakalářské práce popisuje jednotlivé důležité části zdrojového kódu pro webovou aplikaci.

4.3.4.1 Frontend

Frontend je grafické rozhraní webové stránky, přes které se uživateli zobrazí potřebná data v prohlížeči.

Obrázek 24 - Frontend



Zdroj: Vlastní zpracování

V hlavičce se nachází zelený text „otevřeno“, který se mění na „zavřeno“ podle otvírací doby.

V levém postranním panelu jsou v reálném čase vyobrazeny informace o celkovém počtu parkovacích míst a z toho počet obsazených míst. V tomto panelu se nachází ještě mapa s adresou konkrétního parkoviště.

Největší část webové aplikace zabírají virtuální parkovací místa. Ta, která jsou označena číslem, si berou data z ESP32 a mění barvu zelená/červená podle obsazenosti parkovacího místa. Ostatní byla přidána jen jako test pro zobrazení více parkovacích míst, než v tuto chvíli mám senzorů. Barva nasimulovaných parkovacích míst byla volena náhodně (místa s písmenem X).

4.3.4.2 Zpracování dat

Obrázek 25 - Zpracování dat

```php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $api_key = test_input($_POST["api_key"]);
    if($api_key == $api_key_value) {
        $sensor = test_input($_POST["sensor"]);
        $location = test_input($_POST["location"]);
        $availability = test_input($_POST["availability"]);
        $realValue = test_input($_POST["realValue"]);
        $ipAdress = test_input($_POST["ipAdress"]);
        $senzorID = test_input($_POST["senzorID"]);
        include 'conn.php'; //připojení k Databázi
        $sql = "SELECT id FROM senzordata WHERE senzorID='" . $senzorID . "'"; //
        $result = $mysqli->query($sql);
        if ($result->num_rows > 0) {
            $sql = "UPDATE senzordata SET   ipAdress = '" . $ipAdress . "',
                                            availability = '" . $availability . "' WHERE senzorID='" . $senzorID . "'";
        } else {
            $sql = "INSERT INTO senzordata (senzor, senzorID, ipAdress, location, availability)
                    VALUES ('" . $sensor . "', '" . $senzorID . "','" . $ipAdress . "', '" . $location . "', '" . $availability . "')";
            }
        if ($mysqli->query($sql) === TRUE) {
            echo "Vloženo do databáze";
        }
        else {
            echo "Chyba: " . $sql . "<br>" . $mysqli->error;
        }
        $sql = "INSERT INTO senzordata_history (senzorID, ipAdress, availability, realValue)
        VALUES ('" . $senzorID . "', '" . $ipAdress . "', '" . $availability . "', '" . $realValue . "')";
        if ($mysqli->query($sql) === TRUE) {
            echo "Vloženo do databáze";
        }
        else {
            echo "Chyba: " . $sql . "<br>" . $mysqli->error;
        }
        $mysqli->close();
    }
    else {
        echo "Špatný api klíč.";
    }
```

Zdroj: Vlastní zpracování

Tato část zdrojového kódu zajišťuje zpracování dat z HTTP POST, který dostává od ESP32 se senzorem. Nejdříve ověří API klíč, ten musí být identický s tím z ESP32. Nadále uloží data z HTTP POST do dočasné proměnné a připojí se k databázi. Dále zkontroluje, jestli ESP32, ze kterého data přišla, už je uložený v databázi podle *senzorID*. Pokud ano, jen aktualizuje záznam novými daty. Pokud ne, vytvoří nový záznam. Mimo jiné všechna data ukládá vždy i do tabulky *senzor_history*, která slouží pro odladění programu.

4.3.4.3 Otevírací hodiny

Obrázek 26 - Otevírací hodiny

```php
<?php
//Otvírací hodiny
$storeSchedule = [
    'Mon' => ['05:00 AM' => '10:00 PM'],
    'Tue' => ['05:00 AM' => '10:00 PM'],
    'Wed' => ['05:00 AM' => '10:00 PM'],
    'Thu' => ['05:00 AM' => '10:00 PM'],
    'Fri' => ['05:00 AM' => '11:00 PM'],
    'Sat' => ['06:00 AM' => '11:00 PM'],
    'Sun' => ['06:00 AM' => '11:00 PM']
];

// Aktuální Timestamp
$timestamp = time();

// Výchozí status
$status = '<span class="close">Zavřeno</span>';

// Získává aktuální čas
$currentTime = (new DateTime())->setTimestamp($timestamp);

// smyčka kterí projede každý den
foreach ($storeSchedule[date('D', $timestamp)] as $startTime => $endTime) {

    // pro každý den nastaví počátek a konec otvíracích hodin
    $startTime = DateTime::createFromFormat('h:i A', $startTime);
    $endTime   = DateTime::createFromFormat('h:i A', $endTime);

    // Zkontroluje jestli se aktuální čas nachází v v rozmezí otvíracích hodin
    if (($startTime < $currentTime) && ($currentTime < $endTime)) {
        $status = '<span class="open">Otevřeno</span>';
        break;
    }
}

echo "$status";
```

Zdroj: Vlastní zpracování

V této části kódu je nutno nastavit otevírací hodiny daného parkoviště, podle kterých se bude měnit status Otevřeno / Zavřeno. Jedná se jednoduchou smyčku, ve které se kontroluje, jestli aktuální čas je v rozmezí otevíracích hodin. Pokud ano, změní status na *Otevřeno*, pokud ne, zůstane výchozí status *Zavřeno*.

4.3.4.4 Zobrazení dat uživateli

Následující obrázek ukazuje část zdrojového kódu, která se zabývá zobrazováním dat v reálném čase.

Obrázek 27 - Websocket

Client.side

```
io.on('connection', (socket) => {
    clients.push(socket);
    console.log('user connected');

    socket.on('disconnect', () => {
        clients.splice(clients.indexOf(socket), 1);
        console.log('user disconnected');
    });

});

app.get('/update-clients', (req, res) => {
    clients.forEach(socket => socket.emit('sensor-update'));
    res.send('ok');
});

http.listen(port, () => {
    console.log('running');
});
```

Server-side

```
function fetchData() {
    $.get("parkingSlot.php", function (result) {
            var html = '';
            result.forEach(record => {
                html +=
                '<div class="lot-box '+
                (record.availability ==
                'true' ? 'available' : 'unavailable') +'"">'
                + record.id +
                '</div>';
            });
            console.log(html);
            $('#data').html(html);
        });
        $.get("stats.php", function (result) {
            $("#AllSlots").text(result.allSlots);
            $("#FreeSlots").text(result.freeSlots);
        });
}

$(document).ready(function () {
    fetchData();
});

</script>
<script src="http://localhost:3000/socket.io/socket.io.js"></script>
<script>
  const socket = io("http://localhost:3000");
  socket.on('sensor-update', function() {
     console.log('sensor update');
     fetchData();
  });
</script>
```

Zdroj: Vlastní zpracování

Programovací jazyk PHP umožnuje komunikaci s uživatelem pouze na bázi žádosti a odpovědi. Uživatel navštíví stránku, tím pošle žádost, a webový server s PHP pošle odpověď s daty, které si uživatel vyžádal. Pokud se data na serveru změní a uživatel si nepožádá o data znovu, server mu je sám nepošle. Z tohoto důvodu bylo nutné v programovacím jazyku Node.js napsat program (Obrázek 26), který dostává informace od snímače a pokaždé, když jsou data změněna, pošle přes websocket aktualizovaná data všem uživatelům, kteří v danou chvíli jsou na stránce. To vše bez potřeby, aby uživatel stránku musel aktualizovat.

4.4 Testování

Testování chytrého systému parkoviště je zaměřeno na správnou funkci a detekci možných nedostatků.

Při testování byla stanovena hypotéza, kterou se v této práci pokusím potvrdit, popřípadě vyvrátit. Stanovená hypotéza zní následovně: *Navrhovaný systém chytrého parkoviště je konkurenceschopný v porovnání s již existujícími systémy a v některých parametrech je může dokonce předčit.*

4.4.1 Příprava testování

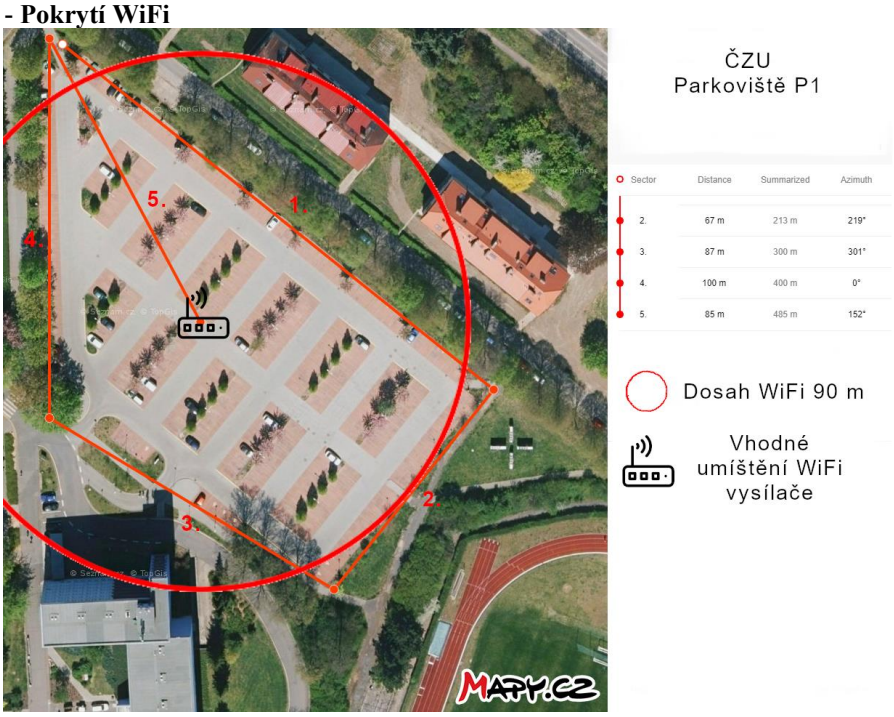
Před začátkem testování systému chytrého parkoviště je zapotřebí určit proměnné, které mohou mít vliv na výslednou funkčnost celého systému. Jedná se především o světelné podmínky, počet parkovacích míst a typ stavby parkoviště (zastřešené, nezastřešené).

Systém chytrého parkoviště byl následně testován ve dne, za šera i ve tmě. Další proměnnou v testu je počet parkovacích míst, kde byl systém vyzkoušen na jednom i více parkovacích míst. Třetí proměnnou je typ stavby parkoviště týkající se jeho zastřešení. Opět byl systém testován na zastřešeném i venkovním parkovšti.

4.4.2 Umístění Raspberry Pi

Snímače umístěné na jednotlivých parkovacích místech je nutné připojit k WiFi síťi, z toho vyplývá důležitost zvolení správného místa pro WiFi vysílač (router). Společně s routerem umístíme i server běžící na Raspberry Pi, který z důvodu menší odezvy a lepší stability celého systému připojíme síťovým kabelem.

Obrázek 28 - Pokrytí WiFi



Zdroj: Vlastní zpracování

Na ilustračním obrázku číslo 27 by stačil pouze jeden WiFi vysílač, který pokryje celé parkoviště P1. Na větších, podzemních, nebo zastřešených parkovištích může být dosah WiFi slabší kvůli překážkám v šíření WiFi signálu. Tento problém lze vyřešit WiFi extendery.

4.4.3 Hledání volného parkovacího místa

ESP32 s ultrasonickým senzorem tvoří funkční snímač, který je nutno umístit na každé parkovací místo a zapojit do elektrické sítě. Led diody na snímači mění svoji barvu (červená nebo zelená) na základě obsazenosti parkovacího místa a slouží tím jako vizuální pomůcka pro řidiče, kteří tak snáze mohou nalézt volné parkovací místo.

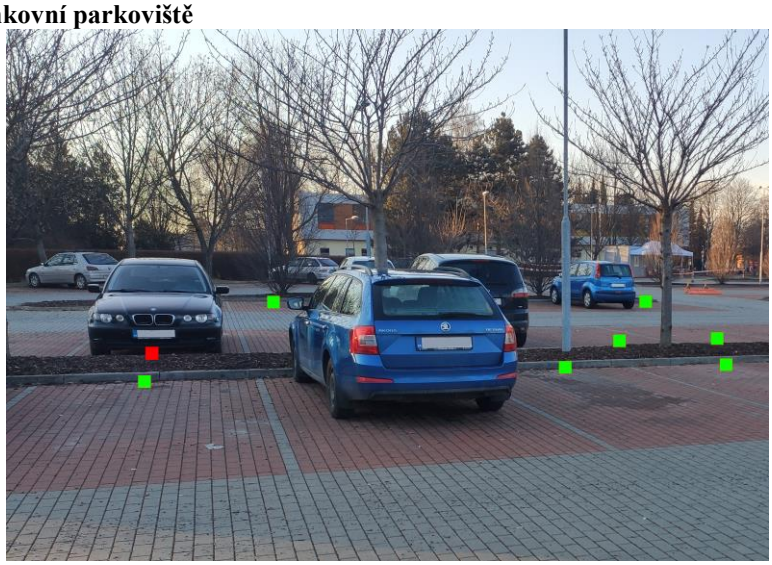
4.4.4 Testování v modelových situacích

Modelové situace byly vytvořeny na reálných místech, a to konkrétně na parkovišti České zemědělské univerzity v Praze a v soukromé garáži. Tyto situace také popisují možná prostředí, ve kterých by tento systém chytrého parkoviště bylo možné reálně použít.

4.4.4.1 Venkovní parkoviště

Pro první modelovou situaci jsem zvolil parkoviště České zemědělské univerzity v Praze (Obrázek 28).

Obrázek 29 - Venkovní parkoviště



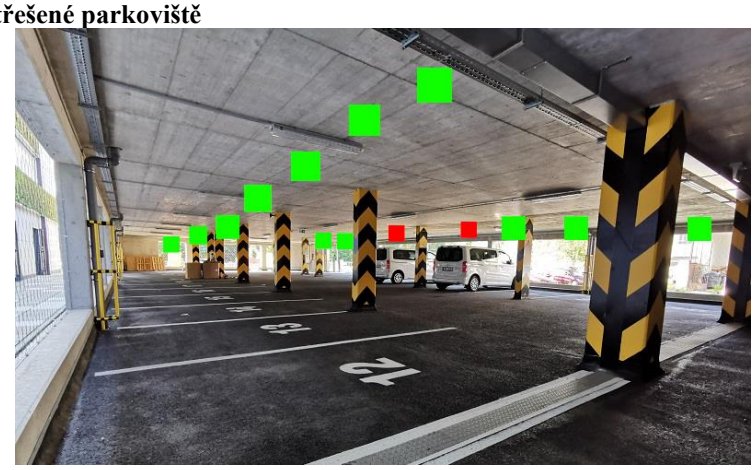
Zdroj: Vlastní zpracování

V tomto případě bylo testování prováděno s deseti snímači na deseti parkovacích místech. Snímač je vždy nutné umístit na venkovních parkovištích na konec každého jednoho sledovaného parkovacího místa. Protože na parkovišti nebyla dostupná potřebná WiFi síť, vytvořil jsem hotspot ze svého mobilního telefonu, který s přehledem pokryl mnou testovanou oblast.

4.4.4.2 Zastřešené parkoviště

Druhá modelová situace probíhala v zastřešené garáži. Obrázek 29 je pouze ilustrační.

Obrázek 30 - Zastřešené parkoviště



Zdroj: (38)

V zastřešených parkovištích jsem snímač umístil na strop nad každé parkovací místo. V mém případě se opět jednalo o deset snímačů přidělaných na strop garáže tak, aby snímaly parkovací místa pod sebou. Umístění snímačů vypadalo jako na ilustračním obrázku číslo 30.

5 Výsledky testování systému chytrého parkoviště

Po provedených testech v reálném prostředí jsem dospěl k následujícím zjištěním.

5.1 Kvalita ultrasonického senzoru

Ultrasonický senzor byl využit společně s ESP32 jako snímač obsazenosti parkovacích míst. Při testování se ukázalo, že jeho omezená snímací vzdálenost pro detekci aut (4 metry) na parkovacím místě je plně dostačující. Ani proměnlivé světelné podmínky neovlivnily tuto skutečnost, protože senzor pracuje na principu odražení zvuku. Stejných výsledků bylo dosaženo ve venkovních i krytých prostorech.

5.2 Webová aplikace

Všechny potřebné informace, které jsem při testování chtěl získat z webové aplikace, se dle předpokladu zobrazily. Jednalo se o vzorek deseti parkovacích míst, na která jsem v reálném čase osobně viděl a mohl tak kontrolovat správnost zobrazených údajů. Kdybych však neměl oční kontakt s parkovacími místy, nebylo by vyobrazení informací zcela přehledné.

5.3 Celková cena

Celková cena byla vypočítána jako součet všech použitých komponentů v systému chytrého parkoviště. Počet komponentů, které jsou potřebné k vytvoření snímače, je nutno vynásobit počtem parkovacích míst, pro které chceme kontrolovat obsazenost. V tomto případě bylo použito deset snímačů, což ukazuje následující tabulka.

Tabulka 5 - Celková cena

PRODUKT	**POČET**	**CENA ZA JEDNOTKU**
Raspberry Pi 3 Model B v1.2	1	1080 Kč
Micro SD 32GB	1	249 Kč
ESP32	10	240 Kč
HC-SR04	10	42 Kč
RGB led dioda	10	4 Kč
Propojovací kabely	10	10 Kč
Celkem		**4289 Kč**

Zdroj: Vlastní zpracování

6 Diskuse

Vytvořený systém chytrého parkoviště byl podroben testování jeho funkčnosti, efektivnosti a jeho použitelnosti v reálném prostředí. Ukázalo se, že použité komponenty byly pro tento účel dostačující. Je však nutné brát v potaz, že testování probíhalo krátkodobě a sestavený systém nebyl nainstalován do parkoviště trvale. Při použití tohoto systému v plánované stavbě parkoviště by bylo vhodné plánovat instalaci systému předem, z důvodu možnosti zahrnout do plánu přívod napájecího kabelu na každé parkovací místo. Na stávajících parkovištích lze však tento systém instalovat dodatečně, což bývá většinou komplikovanější, avšak řešitelné.

Při testování vyšlo najevo, že světelné podmínky nemají vliv na data získávaná ze snímačů. Ani počet parkovacích míst neovlivňoval funkčnost testovaného systému, protože jej lze rozšiřovat o další parkovací místa podle potřeby bez nutnosti dalších zásahů kromě nahrání programu a zapojení do elektrické sítě.

6.1 Alternativní řešení

Následující podkapitoly rozebírají funkčnost komponentů, které v mém systému pracovaly dobře, přes to je však možné je pro další použití změnit a vylepšit.

6.1.1 Senzor

Ultrasonické senzory se v průběhu testování prokázaly jako použitelné v obou modelových situací. Protože ultrasonický senzor funguje na principu odraženého zvuku od objektu, mohl by nastat problém při výskytu ultrasonického šumu v blízkosti snímačů. V takovémto případě by ultrasonický senzor přestal správně měřit vzdálenost objektu, nebo by se stal úplně nepoužitelným. Tento problém lze vyřešit náhradou ultrasonického senzoru optickým senzorem. Optický senzor funguje na stejném principu jako ultrasonický, ale místo zvukového signálu a jeho následném zachycení vysílá laserový paprsek. Tyto dva typy senzorů se hodí při vylepšení obyčejných parkovišť na chytrá, neboť nevyžadují extrémní zásahy do stavby parkoviště, pouze přivedení napájecích kabelů na jednotlivá parkovací místa. Také by teoreticky mohly mít oba senzory problém s detekováním motocyklu, kdyby nebyl zaparkován na středu parkovacího místa.

Oba zmíněné senzory by bylo možné nasadit na již vybudovaná parkoviště.

Kdyby se však počítalo s potřebou kontroly obsazenosti místa již při stavbě parkoviště, bylo by možné umístit pod povrch parkovacích míst magnetické senzory. Následně by odpadla potřeba pokrýt celé parkoviště WiFi signálem.

6.1.2 **Server**

Použití Raspberry Pi jako serveru se ukázalo pro sběr dat ze snímačů a jejich následné zpracování a hostování webové aplikace naprosto vyhovující. Je to levné a prostorově nenáročné zařízení, což zvyšuje využitelnost v téměř jakýkoliv situacích.

Pokud již parkoviště nějaké servery používá, například pro kamerový systém, a běží na operačním systému Windows nebo Linux, bylo by možné hostování webové aplikace na nich.

6.1.3 **PHP**

Webová aplikace byla napsána v programovacím jazyce PHP. V průběhu vytváření této bakalářské práce jsem narazil na problémy funkčnosti PHP. Pro webovou aplikaci, která pracuje s daty z různých senzorů, bych pro napsání celé webové aplikace dnes již nezvolil PHP, nýbrž Node.js. Za pomoci Node.js jsem však nedostatky, které měla samotná moje aplikace v PHP, opravil pomocí technologie websocket přes Node.js. Toto řešení je plně funkční a použitelné i v praxi.

6.1.4 **Webová aplikace**

V rámci této bakalářské práci jsem vytvořil webovou aplikaci, přes kterou si uživatel může vyobrazit potřebné informace o parkovišti. Uživatel se z webové aplikaci bez problémů dozví, jestli je parkoviště otevřeno a kolik je celkem volných parkovacích míst. Problém však nastává, pokud by si uživatel chtěl ověřit konkrétní parkovací místo a nevěděl jeho číslo. Webová aplikace graficky vyobrazuje konkrétní parkovací místa v řádcích po sedmi, podle čísel. To znamená, že uspořádání parkovacích míst graficky neodpovídá reálnému rozvržení parkovacích míst. Tento problém bych při v praxi vyřešil editací CSS tak, aby parkovací místa zobrazená na webové stránce odpovídala uspořádaní parkovacích míst konkrétního parkoviště.

6.2 Nabyté zkušenosti

Při vytváření této bakalářské jsem si rozšířil své znalosti mini počítače Raspberry Pi a ESP32. Zároveň jsem se zdokonalil v programování, a to především v jazycích Node.js, php a C. Také jsem získal cenné zkušenosti v oblasti hardwaru při kompletaci snímače.

7 Závěr

Cílem mé bakalářské práce bylo vytvořit prototyp funkčního systému chytrého parkoviště za využití Raspberry Pi. I přes to, že nebyly použity žádné cenově náročné komponenty, podařilo se tento cíl splnit. Zároveň se potvrdila stanovená hypotéza, že navrhovaný systém chytrého parkoviště je konkurenceschopný a v některých parametrech může předčit již realizovaná chytrá parkoviště. Je však nutné vždy brát v potaz velikost parkoviště, jeho architekturu, náročnost stavebních úprav v případě, že se systém instaluje dodatečně na již stávající parkoviště.

Získávané informace z chytrých parkovišť o obsazenosti míst jsou následně zobrazovaná potencionálním uživatelům parkovišť. Toto zobrazení by podle mé získané zkušenosti mělo být jednoduché a co nejpřehlednější, aby se uživatel se uživatel co nejrychleji dozvěděl hledanou informaci.

Při tvorbě této bakalářské práce jsem se setkal s různými názory na chytrá parkoviště. Skupina lidí dokonce vyjádřila obavy, že využití počítačové sítě v dalších životních situací by mohlo vést k nežádoucímu shromažďování dat o uživatelích chytrých parkovišť. Avšak mnou navržený systém neshromažďuje žádná soukromá data o uživatelích, která by mohla být nějak zneužita, jako například jméno uživatele, platební metody nebo SPZ vozidla.

Jiná skupina lidí naopak vidí v chytrých parkovištích mnoho pozitiv, především pohodlné, rychlé a snadné nalezení volného parkovacího místa. Tím je přecházeno stresovým situacím, které mnoho lidí při parkování zažívá. Na počátku mé práce jsem si tento možný pozitivní vliv na lidskou psychiku neuvědomil, ale je přínosné, pokud díky systému chytrého parkoviště můžeme další stresující faktor v našem životě redukovat.

8 Seznam použitých zdrojů

1. Raspberry.org. *Raspberry pi* [online]. [cit. 14.3.2021]. Cambridge: Raspberry Pi Foundation. Dostupné z: <https://www.raspberrypi.org/>.
2. Raspberrypi.org. *GPIO* [online]. [cit. 14.3.2021]. Cambridge: Raspberry Pi Foundation. Dostupné z: <https://www.raspberrypi.org/documentation/usage/gpio/>.
3. Raspberrypi.org. *GPIO pinout diagram* [online]. [cit. 14.3.2021]. Cambridge: Raspberry Pi Foundation. Dostupné z: <https://www.raspberrypi.org/documentation/usage/gpio/images/GPIO-Pinout-Diagram-2.png>.
4. SELECKÝ, M. *Arduino.* Brno: Computer Press, 2016. ISBN 978-80-251-4849-5.
5. SIAME, A., KUNDA, D. Evolution of PHP Applications: A Systematic Literature Review. *International Journal of Recent Contributions from Engineering* [online]. Zambie: School of Engineering Science and Technology, 2017, vol. 5, [cit. 14.3.2021]. ISSN: 2197-8581. Dostupné z: <https://doi.org/10.3991/ijes.v5i1.6437>.
6. NIXON, R. Learning PHP, *MySQL & JavaScript: with jQuery, CSS & HTML5*. 4. vydání. Beijing: O'Reilly Media, 2014. ISBN: 978-05-961-5713-5.
7. DRAKE, F., L. Python.org. *What is Python? Executive summary* [online]. Poslední revize: 19.2.2021. [cit. 14.3.2021]. Beaverton: Python Software Foundation. Dostupné z: <https://www.python.org/doc/essays/blurb/>.
8. VARTIKA02. Geeksforgeeks.org. *Disadvantages of python* [online]. Poslední revize: 14.1.2019. [cit. 14.3.2021]. Noida: GeeksforGeeks. Dostupné z: <https://www.geeksforgeeks.org/disadvantages-of-python/>.
9. ŠORČÍK, L., HEFKA, L. Smartiple.com. *Parkinto* [online]. Poslední revize: 1.1.2021. [cit. 14.3.2021]. Praha: Smartiple, s.r.o. Dostupné z: <https://smartiple.com/>.
10. Apache.org. *Apache http Server Documentation* [online]. Poslední revize: 7.8.2020. [cit. 14.3.2021]. Forest Hill, USA. Dostupné z: <https://httpd.apache.org/>.
11. SLAVÍK, J., SLAVÍKOVÁ, P. Smartcityvpraxi.cz. *Hlavní město Praha otestuje inovativní způsob řízení svozu odpadu s využitím internetu věcí* [online]. Poslední

revize: 7.1.2021. [cit. 14.3.2021]. Praha: Smart city v praxi. Dostupné z: <http://www.smartcityvpraxi.cz/>.

12. Cartower.cz. *Car tower CZ* [online]. [cit. 14.3.2021]. Praha: Cartower. Dostupné z: <https://www.cartower.cz/wp-content/uploads/2020/01/Car-Tower-CZ.pdf>.
13. Raspberry.org. *Software* [online]. [cit. 14.3.2021]. Cambridge: Raspberry Pi Foundation. Dostupné z: <https://www.raspberrypi.org/software/>.
14. MÜLLER, O., DELISLE, M., TUREK, A., M., ČIHAŘ, M., HICKING, G. Phpmyadmin.net. *Documentation* [online]. Poslední revize: 15.10.2020. [cit. 14.3.2021]. Dostupné z: <https://www.phpmyadmin.net/>.
15. KJELL. Fritzing.org. *Fritzing learning* [online]. Poslední revize: 13.1.2021. [cit. 14.3.2021]. Dostupné z: <https://fritzing.org/>.
16. SANTOS, R. Randomnerdtutorials.com. *Complete Guide for Ultrasonic Sensor HC-SR04 with Arduino* [online]. Poslední revize: 17.11.2013. [cit. 14.3.2021]. Dostupné z: <https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/>.
17. Rpishop.cz. *Ultrazvukový senzor HC-SR04* [online]. [cit. 14.3.2021]. Dostupné z: <https://rpishop.cz/senzory/815-ultrazvukovy-senzor-hc-sr04.html>.
18. Svetsoucastek.cz. *LED 5mm zelená 2000mcd/70° difúzní Hebei 560PG2D* [online]. [cit. 14.3.2021]. Dostupné z: <https://www.svetsoucastek.cz/led-dioda-5mm-hebei-560pg2d-p336/>.
19. Rpishop.cz. *Modul senzoru plynu MQ-5* [online]. [cit. 14.3.2021]. Dostupné z: <https://rpishop.cz/soucastky/2438-modul-senzoru-plynu-mq-5.html>.
20. SLAVÍK, J. a kolektiv. Mmr.cz. *Metodika Smart Cities: Metodika pro přípravu a realizaci konceptu Smart Cities na úrovni měst, obcí a regionů* [online]. [cit. 14.3.2021]. Praha: Ministerstvo pro místní rozvoj ČR, 2018. Dostupné z: <https://mmr.cz/getmedia/f76636e0-88ad-40f9-8e27-cbb774ea7caf/Metodika_Smart_Cities.pdf.aspx?ext=.pdf>.
21. SALAZAR, J., SILVESTRE, S. *Internet věcí* [online]. [cit. 14.3.2021. Praha: Vysoké učení technické v Praze, 2017. ISBN 978-80-01-06231-9. Dostupné z: <http://techpedia.fel.cvut.cz/download/?fileId=806&objectId=110>.
22. SALAZAR, J., SILVESTRE, S. *Internet věcí* [online]. [cit. 14.3.2021. Praha: Vysoké učení technické v Praze, 2017. ISBN 978-80-01-06231-9. Dostupné z: <http://techpedia.fel.cvut.cz/download/?fileId=806&objectId=110>.

23. SALAZAR, J., SILVESTRE, S. *Internet věcí* [online]. [cit. 14.3.2021. Praha: Vysoké učení technické v Praze, 2017. ISBN 978-80-01-06231-9. Dostupné z: <http://techpedia.fel.cvut.cz/download/?fileId=806&objectId=110>.
24. SLAVÍK, J., SLAVÍKOVÁ, P. Smartcityvpraxi.cz. *Zajímavé projekty* [online]. Poslední revize: 7.1.2021. [cit. 14.3.2021]. Praha: Smart city v praxi. Dostupné z: <http://www.smartcityvpraxi.cz/zajimave_projekty_356.php>.
25. cz.farnell.com. MCP3008-I/P [online]. [cit. 14.3.2021]. Dostupné z: <https://cz.farnell.com/microchip/mcp3008-i-p/10bit-adc-2-7v-8ch-spi-16dip/dp/1627174?gross_price=true&mckv=sg48XjgBr_dc|pcrid|490629366880|plid||kword||match||slid||product|1627174|pgrid|104911742542|ptaid|pla-326755459603|&CMP=KNC-GCZ-SHOPPING-Whoop-LO-31-Aug-20/>.
26. laskarduino.cz. 8 kanály Obousměrný převodník logických úrovní 5V a 3.3V [online]. [cit. 14.3.2021]. Dostupné z: <https://www.laskarduino.cz/8-kanaly-obousmerny-prevodnik-logickych-urovni-5v-a-3-3v/>.
27. COX, L., K. Blog.hubspot.com. *Web Design 101: How HTML, CSS, and JavaScript Work* [online]. Poslední revize: 26.10.2020. [cit. 14.3.2021]. Dostupné z: <https://blog.hubspot.com/marketing/web-design-html-css-javascript>.
28. KOĎOUSKOVÁ, B. Rascasone.cz. *Javascript pro začátečníky: Co to je a jak funguje* [online]. Poslední revize: 2.2.2021. [cit. 14.3.2021]. Dostupné z: <https://www.rascasone.com/cs/blog/co-je-javascript-pro-zacatecniky>.
29. DAHL, R. Nodejs.dev. *Learn* [online]. [cit. 14.3.2021]. Dostupné z: <https://nodejs.dev/learn>.
30. HO, D. Notepad-plus-plus.org. *What is Notepad++* [online]. Poslední revize: 15.2.2021. [cit. 14.3.2021]. Paříž. Dostupné z: <https://notepad-plus-plus.org/>.
31. SLAVÍK, J., SLAVÍKOVÁ, P. Smartcityvpraxi.cz. *Zajímavé projekty: Odpady* [online]. Poslední revize: 7.1.2021. [cit. 14.3.2021]. Praha: Smart city v praxi. Dostupné z: <http://www.smartcityvpraxi.cz/zajimave_projekty/356-odpady.jpg>.
32. Youtube. *S-oil „here“ balloon* [online]. [cit. 14.3.2021] Dostupné z: <https://www.youtube.com/watch?v=ccM8RLLKasw>
33. Cartower.cz. *Banner car tower* [online]. [cit. 14.3.2021]. Praha: Cartower. Dostupné z: <https://www.cartower.cz/wp-content/uploads/2019/02/2banner_CarTower.jpg>.

34. HO, D. Notepad-plus-plus.org. *Notepad 4ever* [online]. Poslední revize: 15.2.2021. [cit. 14.3.2021]. Paříž. <http://notepad-plus-plus.org/assets/images/notepad4ever.gif>.
35. ŠORČÍK, L., HEFKA, L. Smartiple.com. *Kutná Hora parking Havířská* [online]. Poslední revize: 1.1.2021. [cit. 14.3.2021]. Praha: Smartiple, s.r.o. Dostupné z: <https://smartiple.com/images/ref-khnet-havirska.jpg>.
36. Rpishop.cz. *Raspberry Pi 3, model B, 64-bit* [online]. [cit. 14.3.2021]. Dostupné z: <https://rpishop.cz/15347-large_default/raspberry-pi-3-model-b-64-bit.jpg>
37. Hadex.cz. *ESP32* [online]. [cit. 14.3.2021]. Dostupné z: <https://www.hadex.cz/img/zbozi/m432.jpg>.
38. Vysocina-news.cz. *parkoviště Žižkov* [online]. [cit. 14.3.2021]. Dostupné z: <https://vysocina-news.cz/wp-content/uploads/2021/01/Parkoviste_Zizkov_2.jpg>.

9 Přílohy

Příloha A: zdrojovyKod.zip

Příloha A se zdrojovým kódem byla nahrána do UISu jako soubor zip.