ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra ekonomiky, manažerství a humanitních věd

DIPLOMOVÁ PRÁCE

Systém pro sledování hladiny odpadových nádob

Autor: Kryštof Novák

Studijní program: Elektrotechnika, Energetika a management

Specializace: Management energetiky a elektrotechniky

Vedoucí práce: Ing. Martin Dobiáš, Ph.D.

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Novák** Jméno: **Kryštof** Osobní číslo: **434642**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra ekonomiky, manažerství a humanitních věd**

Studijní program: **Elektrotechnika, energetika a management**

Specializace: **Management energetiky a elektrotechniky**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Systém pro sledování hladiny odpadových nádob

Název diplomové práce anglicky:

The System for Monitoring Fill Level of a Waste Containers

Pokyny pro vypracování:

1) Navrhněte, vyrobte a otestujte měřící jednotku pro měření zaplněnosti odpadových nádob
2) Navrhněte a naprogramujte webovou aplikaci pro sběr a analýzu dat z měřících jednotek dle bodu 1
3) Analyzujte současný stav procesu sledování hladiny odpadových nádob
4) Navrhněte inovovaných proces sledování hladiny odpadových nádob s využitím předmětné senzoriky
5) Vyhodnoťte přínosy navrženého technického řešení

Seznam doporučené literatury:

1) IoT and Edge Computing for Architects: Implementing Edge and IoT Systems from Sensors to Clouds with Communication Systems, Analytics, and Security, Perry Lee, ISBN-13: 978-1839214806
2) Flask Web Development: Developing Web Applications with Python, Miguel Grinberg, ISBN-13: 978-1491991732
3) SYNEK, Miloslav. Manažerská ekonomika., 5. aktualizované a doplněné vydání Praha: Grada, 2011. ISBN 978-80-247-3494-1.
4) FOTR, Jiří a Lenka ŠVECOVÁ. Manažerské rozhodování: postupy, metody a nástroje. Třetí, přepracované vydání. Praha: Ekopress, 2016. ISBN 978-80-87865-33-0.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Martin Dobiáš, Ph.D., katedra ekonomiky, manažerství a humanitních věd FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **17.03.2021** Termín odevzdání diplomové práce: **05.01.2021**

Platnost zadání diplomové práce: **19.02.2023**

Ing. Martin Dobiáš, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. | Datum převzetí zadání

Podpis studenta

Čestné prohlášení

Prohlašuji, že jsem předloženou diplomovou práci vypracoval zcela samostatně pod vedením vedoucího práce a že jsem uvedl veškeré informační zdroje v souladu s metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, dne …… ……..

Podpis autora práce

Poděkování

Rád bych tímto poděkoval vedoucímu práce Ing. Martinu Dobiášovi, Ph.D. za jeho čas, odborné vedení a vstřícnost. Také bych chtěl poděkovat společnosti Odpadová poradenská s.r.o. za cenné rady a informace z prostředí českého odpadového hospodářství.

Abstrakt

Diplomová práce se v teoretické části zabírá rozborem problémů v nynějším systému svozu odpadu a navrhuje inovované řešení sledování zaplnění odpadových nádob, které by adresované problémy mělo řešit. Navržené řešení je v praktické části realizováno v prototypové sérii, otestováno a na základě výsledků krátkodobých a dlouhodobých testů je prohlášeno za dobře zvolené a funkční. V závěrečné části práce je realizována případová studie spočívající ve sběru reálných dat z města Chodov a diskrétní časová simulace napsaná v programovacím jazyce Python simulující zaplňování odpadových nádob a systém optimalizovaného svozu v pěti variantách. Varianta s nejlepšími výsledky je poté ekonomicky zhodnocena z pohledu města a svozové společnosti a jsou vydány doporučení a stanoveny mimoekonomické přínosy a příležitosti.

Abstract

In the theoretical part, the diploma thesis deals with the analysis of problems in the current waste collection system and proposes an innovative solution for monitoring the filling of waste containers, which should solve the addressed problems. The proposed solution is in the practical part implemented in a prototype series, tested and based on the results of short-term and long-term tests is declared well-chosen and functional. The final part of the work is a case study consisting of the collection of real data from the city of Chodov and a discrete time simulation written in the Python programming language simulating the filling of waste containers and an optimized collection system in five variants. The variant with the best results is then economically evaluated from the point of view of the city and the collection company, and recommendations are issued and non-economic benefits and opportunities are determined.

Klíčová slova

Monitoring odpadů, chytré odpadové hospodářství, smart city, chytrý svoz odpadu, systém sledování odpadů, dynamický svoz odpadu, simulace svozu odpadu

Keywords

Waste monitoring, smart waste management, smart city, smart waste collection, system of waste monitoring, dynamical waste collection, waste collection simulation

Obsah

1 Úvod

1.1 Cíl práce

Cílem této diplomové práce je zanalyzovat problémy současného stavu svozu separovaného odpadu, zjistit nynější běžné způsoby a celkovou roli sledování zaplnění odpadových nádob v kontextu svozu separovaného odpadu a na základě těchto informací navrhnout, realizovat a otestovat inovovaný systém pro sledování zaplnění odpadový nádob, popsat jeho výhody a navrhnout jeho uplatnění a možnosti integrace do nynějšího svozového systému. Tato práce se bude zabývat jak problematikou na obecné rovině, tak i ve vztahu k českým městům. Zároveň v rámci práce bude realizována případové studie, na které se vyhodnotí konkrétní přínosy, které toto řešení přináší.

1.2 Motivace

V roce 2050 má v městech a městských částech přibýt 2,5 miliardy obyvatel v porovnání s dnešní dobou (rok 2021) [1]. S přibývajícím obyvatelstvem v urbanizovaných oblastech proporcionálně přibývá i množství odpadu, které domácnosti vyprodukují a tím se zvyšují i nároky na systém odpadového hospodářství – zejména na svoz a logistiku odpadu od domácností do center na zpracování odpadu. Systém svozu odpadu probíhá s drobnými úpravami a výjimkami desítky let stále stejně – v pravidelných nastavených časových intervalech nákladní auto vyváží po fixní trase odpadové nádoby, po naplnění přerušuje svoz a jede se vysypat na skládku, třídící linku nebo do spalovny. Poté pokračuje na trase tam, kde skončilo. Tento způsob svozu odpadu je neefektivní, protože se jednak neví, kdy auto bude zaplněné, a tedy v jakém okamžiku se bude muset odklonit od trasy pro nutnost výsypu, ale také se vyvážejí ne zcela zaplněné nádoby, což dále zvyšuje náklady. Tento problém má i environmentální rozsah. Neefektivní svoz způsobuje více najetých kilometrů svozových aut, a tedy i větší množství vyprodukovaných výfukových plynů a nárůst akustického smogu oproti optimalizovanému, efektivnímu svozu.

Zároveň v případech, kdy produkce odpadů v oblasti není zcela konstantní, je velmi těžké nastavit frekvenci svozů tak, aby nedocházelo k přeplňování odpadových nádob, a tím i odkládání odpadu kolem nádob způsobující nevzhledný veřejný prostor. Tento problém se dále prohloubil s rozšířením třídění odpadu. Domácnosti začaly třídit (nejčastěji sklo, papír, plast a směsný odpad), čímž vznikla nutnost umístění specifických odpadových nádob pro jednotlivé typy odpadů, což dále ztížilo odhad zaplnění jednotlivých odpadových nádob na tříděný odpad, protože už nestačí pouze informace o produkci odpadu v dané oblasti, ale je také potřeba vědět relativní zastoupení jednotlivých typů odpadů v celkové produkci pro správné nastavení frekvence svozů jednotlivých odpadů.

Dále přichází čtvrtá průmyslová revoluce a s tím i koncept internetu věcí a chytrých měst, jehož smyslem je sbírání a agregování dat ve městech pomocí elektronických zařízení propojených do společné sítě a na základě těchto dat dělání rozhodnutí, které přispějí k lepšímu blahobytu, udržitelnosti a zefektivnění služeb a procesů ve městech. Odpadové hospodářství je nedílnou součástí tohoto

nastupujícího konceptu, nicméně jak bude popsáno v následující kapitole, česká města jsou v této oblasti stále na začátku.

Tyto všechny problémy a důvody byly motivací se zabývat tímto tématem a také hnacím motorem pro navržení řešení, které by tyto problémy mohlo vyřešit.

2 Analýza současného stavu

Pro analýzu nynějšího stavu monitoringu separovaného odpadu, je nutné se nejdříve podívat na problematiku v kontextu celkového svozu odpadu. V této kapitole se tedy bude věnováno rozboru nynějšímu stavu svozu a monitoringu separovaného odpadu. Budou podrobněji adresovány současné problémy a definovány možné motivace jednotlivých subjektů k využití monitoringu odpadů. Pro analýzu současného stavu, problémů a příležitostí byly provedeny řízené rozhovory se zástupci měst mající na starosti odpadové hospodářství, zástupci svozových firem a dvěma konzultačními společnostmi zabývajícími se odpadovým hospodářstvím v České republice.

2.1 Systém svozu separovaného odpadu

Obce dle § 5 odst. 1 písm. zákona o odpadech č. 541/2020 Sb. mají odpovědnost za nakládání s komunálním odpadem, který vzniká na území obce z nepodnikající činnosti fyzických osob. Právnické osoby a podnikající fyzické osoby mají možnost se zapojit do obecního systému nakládání s odpady na základě písemné smlouvy. Dále obce mají povinnost určit a zajistit místa pro oddělené soustřeďování jednotlivých složek komunálního odpadu (obrázek 1), a to minimálně v rozsahu nebezpečných odpadů, skla, plastů, kovů, biologických odpadů, papíru, jedlých tuků a olejů a od 1. ledna 2025 také textilu. Z toho vyplývá, že obce musejí zajistit občanům systém tříděného sběru odpadů, do kterého spadá umístění veřejně přístupných sběrných nádob, jejich pravidelné vyprazdňování a odbyt odpadu. Fyzické osoby dle zákona mají povinnost tento systém respektovat, tedy mají povinnost vyprodukovaný odpad třídit a odevzdat na určená místa dle obecní vyhlášky.

Obrázek 1: Tříděný nádobový sběr. Foto archiv zrcadlo.net

2.1.1 Zajištění svozu separovaného odpadu

Zajištění separovaného sběru – tj. umístění nádob na tříděný odpad a jeho svoz – obce řeší buď pronájmem nádob se službou svozu od soukromé svozové společnosti, případně mají nádoby vlastní a nakupují pouze službu svozu. V drtivé většině případů mají svozové firmy ve smlouvě definované

svozové dny, kdy sváží odpad s tím, že obce mají možnost si objednat mimořádný svoz z důvodu nečekaného přeplnění nádob. Ve specifických případech malých obcí (např. Dvůr Králové nebo Albrechtice) se svozy objednávají zvlášť po manuální kontrole zaplnění jednotlivých nádob. Tento dynamický svoz s manuální kontrolou funguje ale pouze pro odpady s pomalou dynamikou zaplňování a velmi malé obce, protože zaplnění buď kontroluje přímo starosta obce nebo referent pro odpady, což je ale velmi časově náročné. Při větším počtu odpadových nádob, či rychlejší dynamice zaplňování by nebylo v časových možnostech zaměstnance úřadu kontrolovat odpadové nádoby, a ještě stíhat vlastní agendu. Specializovaný zaměstnanec úřadu kontrolující zaplnění odpadových nádob by byl pro malou až střední obec finančně neunesitelný.

2.1.2 Finanční aspekty svozu odpadu

Smluvní vztah mezi obcí a svozovou firmou je téměř výhradě specifický a odvíjí se vždy od konkrétních potřeb dané obce. Nicméně ve většině případů v České republice je to modifikace z tří nejčastějších modelů platby za svoz a zpracování odpadu.

Paušální platba

Ve smlouvě mezi obcí a svozovou firmou je stanovena roční cena služby svozu odpadu na základě počtu nádob a frekvence výsypu. Zpravidla je možné ve smluvně stanoveném rozsahu počet odpadových nádob pro svoz snížit či zvýšit za stanovenou jednotkovou cenu. V roční sazbě jsou již zahrnuty poplatky za uložení odpadu.

Dělený tarif

Dělený tarif spočívá v rozdělení tarifu na dvě části – na výsyp a dopravu a na cenu za zpracování odpadu. Cena za jednotlivý výsyp a dopravu je definována ve smlouvě mezi obcí a svozovou firmou, cena za odbyt odpadu záleží na aktuální tržní ceně odpadu a platí jí obec za tunu. Odpad je možné umístit na skládku, do spalovny případně na třídící linku. Jelikož je odpad komodita jako každá jiná, při příznivé ceně konkrétního typu odpadu na trhu, může být cena za odbyt odpadu kladná. To znamená, že obec může dostávat peníze za odevzdávání vytříděného materiálu – odpadu.

Platba za hmotnost

Další model platby za odpady je domluvená cena za tunu vyprodukovaného odpadu mezi obcí a svozovou společností při fixní infrastruktuře. Fixní infrastruktura znamená jasně daný počet nádob a frekvence výsypu. Obec tedy neřeší cenu za výsyp, dopravu ani odbyt odpadu, pouze množství odpadu, které je vyprodukované na území obce. Ve smlouvě bývá stanoven počet nádob, který je možný přidat do svozového systému.

2.1.2.1 Odměny obcím za třídění odpadu – Systém EKO-KOM

Další tok peněz v rámci systému svozu odpadu pramení z odměn ze Systému EKOKOM. EKO-KOM a.s. je autorizovaná nezisková společnost zajišťující podporu zpětného odběru obalových odpadů pomocí nástroje „Systém EKO-KOM“. Princip tohoto systému stojí na zákonu o obalech č. 477/2001 Sb., který ukládá povinnost maloobchodům, dovozcům a distributorům zpětně odebírat obaly uvedené

na trh a na zákonu o odpadech č. 541/2020 Sb., který ukládá obcím a městům povinnost třídit a využívat komunální odpad, jehož nedílnou součástí je i obalový odpad. Firmy uvádějící obaly na trh platí společnosti EKO-KOM poplatek do „Systému EKO-KOM“, jehož výše se odvíjí od charakteru obalu a množství. Obce a města jako největší sběratelé komunálního odpadu, a tedy i obalů, dostávají smluvní odměny za zajištění míst zpětného odběru, obsluhu míst zpětného odběru, zajištění využití odpadu z obalů a za energetické využití odpadu z obalů. Tato odměna se odvíjí od počtu sběrných míst, množství vyprodukovaného obalového odpadu a je určena pro snížení nákladů obcí spojených se zpětným odběrem dále využitelného odpadu [2]. Obce a města se do tohoto systému zapojují dobrovolně, členství není povinné. Dle referentů pro odpadové hospodářství obcí a měst výše hlavní odměny – za obsluhu míst se zpětným odběrem (tabulka 1) - tvoří důležitý příspěvek do městských a obecních rozpočtů.

Tabulka 1: Ceník hlavní odměny obcím a městům v "Systému EKO-KOM" [3]

Veřejná sběrná síť (nádobový sběr[1], pytlový sběr[2], individuální nádobový sběr[3] organizovaný v rámci systému obce)

Velikost sídla	Odměna za obsluhu míst zpětného odběru (Kč/t vytříděných obalových komunálních odpadů)							
	Papír	Plasty	Sklo		Nápojový karton		Kov	
			směsné	čiré	samostatný sběr	sbíraný ve směsi s jinou komoditou	samostatný sběr	sbíraný ve směsi s jinou komoditou
≤ 1 000 obyvatel	3650	6960	1010	1010	5440	4940	4750	4320
1 001 až 2 000 obyvatel	2750	5660	990	990	4490	4080	3930	3560
2 001 až 5 000 obyvatel	2730	5650	980	980	4410	4000	3860	3500
5 001 až 15 000 obyvatel	2790	5680	1000	1000	4440	4040	3890	3530
15 001 až 40 000 obyvatel	3370	5890	1040	1040	4610	4180	4020	3660
40 001 až 100 000 obyvatel	3650	6390	1050	1050	4990	4530	4380	3980
≥ 100 001 obyvatel	3070	5990	1060	1060	4680	4250	4090	3720

Příklad města Chodov

Město Chodov s 13 000 obyvateli za rok 2020 shromáždilo 280 tun papíru, 150 tun plastů, 131 tun skla a 35 tun kovu (tabulka 2). Pokud by byl naplněn potenciál separace komunálního odpadu města, potom by roční odhadované vyseparované množství papíru bylo 734 tun, plastu 525 tun, skla 290 tun a kovů 59 tun, což přepočítáním pomocí tabulky 3 na obalovou složku dělá 140 tun papíru, 101 tun plastu, 128 tun skla, 3,5 tuny kovu reálné produkce a 367,4 tun papíru, 357 tun plastu, 284 tun skla a 5,9 tun kovů potenciální produkce.

Tabulka 2 - Vyprodukované množství odpadu Chodov [4]

Druh vytříděného odpadu	Shromážděné množství [t/rok 2020]	Potenciál produkce [t/rok]	Účinnost separace ze SKO [%]	Shromážděné obaly [t/rok 2020]	Potenciál shromáždění obalů [t/rok]
Papír	280,2	734,7	38%	140,1	367,4
Plast	149,9	525,2	29%	101,9	357,1
Sklo	131,0	290,3	45%	128,4	284,5
Kovy	35,2	58,9	60%	3,5	5,9

Tabulka 3 - Podíl obalové složky v komunálním odpadu

Způsob sběru	Nádoby a pytle	Sběrné dvory, sběrná místa	Ostatní sběr
Název komodity	Podíl obalové složky v hmotnostních procentech		
Papír	48 %	50 %	42 %
Plasty	68 %	68 %	68 %
Sklo (směs, čiré)	98 %	98 %	98 %
Kovy	80 %	6 %	7 %
Nápojový karton	100 %	100 %	100 %

Toto obalové množství po přepočítání na odměny za obsluhu sběrných míst se zpětným odběrem za rok 2020 (tabulka 4) jsou 390 810 Kč za papír, 578 929 Kč za plast, 128 385 Kč za sklo a 13 705 Kč za kov. Celkem tedy 1 111 829 Kč. Potenciál celkové odměny je 3 360 892 Kč.

Tabulka 4 - Odměny EKOKOM městu Chodov

Druh vytříděného odpadu	**Rok 2020 [Kč]**	**Potenciál [Kč/rok]**
Papír	390 810 CZK	1 024 924 CZK
Plast	578 929 CZK	2 028 541 CZK
Sklo	128 385 CZK	284 510 CZK
Kovy	13 705 CZK	22 916 CZK
Celkem	**1 111 829 CZK**	**3 360 892 CZK**

Pro město s 13 000 obyvateli je 1,1 mil. Kč velmi významný příspěvek do rozpočtu odpadového hospodářství a je proto důležité, aby finanční prostředky z odměn systému EKOKOM byly alokovány spravedlivě na základě skutečného vyprodukovaného množství separovaného odpadu. Tomuto problému je více věnováno v sekci kapitole 2.1.4 - „Rozdělení odměn ze Systému EKO-KOM“ v následující kapitole.

Veškeré informace v této kapitole byly se schválením brány z interních dokumentů města Chodov a POH (Plánu Odpadového Hospodářství) města Chodov pro rok 2020.

2.1.3 Subjekty v systému svozu odpadů

V systému svozu komunálního odpadu figurují tři subjekty – občan, město/obec a svozová firma – každý se svými zájmy.

Občan

Občan chce co nejlepší odpadový servis, což spočívá zejména v krátkých docházkových vzdálenostech ke sběrným nádobám od místa bydliště – tj. co nejhustší sítě sběrných nádob – a k dispozici neustálou kapacitu sběrných nádob pro vyhození odpadu. Při nedostatcích má občan možnost podávat podměty svému odpadovému referentovi na městském úřadě, který by měl podměty projít a udělat na základě toho nějaké rozhodnutí. Problémem tohoto systému zpětné vazby je, že občan si nikdy nestěžuje, že nádoba se vyváží poloprázdná nebo prázdná. Podává stížnosti pouze, když je nádoba plná. A to ještě ve zlomku případech – většinou prostě nechá odpad vedle nádoby.

Město/Obec

Město či obec má za cíl poskytnout co nejlepší servis občanům za co nejpřijatelnější cenu. Aby se toho dosáhlo, je potřeba optimálně nastavit počty, rozestavění a frekvenci svozů sběrných nádob. K optimálnímu nastavení je potřeba znát dynamiku zaplňování v dané oblasti, což je ale problém, protože zpětná vazba od občanů je pouze jednostranná a zároveň ne úplně spolehlivá. Jiným stálým nástrojem měřitelnosti zaplnění města v drtivě většině (viz. kapitola 2.2.2) nedisponují. Pokud má město několik set nádob, je nereálné, aby odpadový referent všechny nádoby obíhal a zaznamenával si jejich zaplnění v čase. Tento problém se často řeší externí konzultační společností, která ručně provede průzkum dynamiky zaplňování a na základě závěrů tohoto průzkumu město provede opatření. Takové průzkumy jsou ale velmi finančně nákladné a zároveň získané závěry mohou být dost často zavádějící, protože data získaná z ručního měření zaplněnosti (pravidelné objíždění odpadových nádob a zaznamenávání zaplnění) jsou kvůli velkým finančním nákladům měření pouze krátkodobé a nemusí tedy zachycovat reálný dlouhodobý stav.

Ještě komplikovanější to je ve velkých městech (např. Brno, Praha), kde sběr odpadu, a tedy i rozpočet na sběr odpadu, má na starosti magistrát, ale zároveň každá městská část má svého odpadového referenta, který se stará o odpady v jeho městské části. V těchto případech chce nejlepší službu nehledě na náklady nejen občan, ale i odpadový referent. To má za následek, že odpadoví referenti na žádost svých občanů doslova bombardují obor odpadů magistrátu, aby jim byly zvýšeny kapacity sběrných nádob a frekvence svozu s tím, že potenciálně vzniklé náklady nejsou jejich starost. Magistrát se snaží žádostem vyhovět, ale ve většině případů to není ekonomicky možné, tudíž jsou žádosti zamítány.

Svozové společnosti

Motivací svozových společností je uspokojit svého zákazníka (město) a mít co největší zisk. Způsob dosažení největšího zisku záleží na uzavřené smlouvě mezi městem a svozovou firmou. Pokud je domluvena paušální platba nebo dělený tarif, svozová firma nemá zájem o optimalizaci, protože z výsypu plně nevyužitých odpadových nádob a následné dopravy menšího objemu odpadu mají stejný příjem jako z výsypu a dopravy plných odpadových nádob. Jinak tomu je v případě platby za hmotnost. Město platí svozové firmě za hmotnost svezeného odpadu (s určitými podmínkami) a nemá tedy přímý ekonomický zájem optimalizovat svoz – to jsou zájmy a případné ušetřené náklady svozové firmy. Některé svozové firmy se snaží při výsypu zaznamenávat zaplnění odpadových nádob v dlouhodobém horizontu, aby měly lepší odhad frekvence svozu. Tento způsob evidence je však nefunkční z důvodu neochoty a ledabylosti posádky svozových vozů (viz. kapitola 2.2.1, Evidence při svozu odpadu).

2.1.4 Problémy nynějšího svozu odpadu

Jak bylo uvedeno v kapitole 2.1.1, většina měst a obcí nechává svážet odpady staticky jednou za určitý čas ve stanovený svozový den po fixní svozové trase. Prvotní odhad frekvence tohoto svozu a počet a rozmístění odpadových nádob se odvíjí od charakteru dané oblasti – zejména se zohledňuje počet obyvatel a hustota zalidnění. Dále se systém optimalizuje při projevení dlouhodobých zjevných nedostatků jako např. přeplňování nebo nedostatečnému využití nádob. Z diskuse s referenty odpadových hospodářství obcí a měst vyplynulo, jak již bylo uvedeno, že tento systém má své

nedostatky zejména z pohledu získávání skutečných informací o přeplňování či nedostatečnému vytížení.

Největší problémem konvenčního statického svozu je tedy absence spolehlivé informace o zaplněnosti. Informace o zaplněnosti je klíčová k uskutečnění efektivního svozu, protože některé typy odpadů (zejména tříděné odpady) jsou produkovány nárazovitě a nedá se tedy přesně odhadnout, kdy budou nádoby plné. Odhad produkce je ještě těžší v urbanizovaných oblastech s velkou fluktuací obyvatelstva a s vysokým turismem (např. pražské části). Produkci odpadu dále ovlivňují vlivy, které působí na běžný chod společnosti – státní svátky, prázdniny, velké společenské události nebo např. v poslední době pandemie COVID-19. Společně všechny tyto faktory vytváří problém špatného odhadu zaplnění odpadových nádob, z kterého se dále odvozují dílčí problémy.

Přeplňování nádob

Při poddimenzované frekvenci svozu nebo počtu odpadových nádob dochází k přeplňování nádob (obrázek 2). To má za následek nespokojené občany, kteří jdou s odpadem ke sběrným místům, ale nemají ho kam dát, protože nádoby jsou plné. Z tohoto důvodu je velmi často odpad zanechán vedle nádob, čímž vzniká nelegální skládka, a tedy i nevzhledný veřejný prostor. Za takové chování občan dle § 117 zákona o odpadech č. 541/2020 Sb. může dostat pokutu až 50 000 Kč za odkládání odpadu mimo vyhrazená místa.

Problém přeplňování není způsobený pouze poddimenzovaným svozem, ale také často zneužíváním právnickými či podnikajícími fyzickými osobami. Právnické a podnikající fyzické osoby mají disponovat vlastními odpadovými nádobami a nesmí odevzdávat odpad do veřejných nádob. To se dle zkušeností obecních a městských referentů pro odpady děje ale velmi často. Výjimku tvoří písemný souhlas o zapojení do veřejného sběru odpadu mezi právnickou či podnikající fyzickou osobou a městem nebo obcí, poté je využití veřejných sběrných míst legální.

Obrázek 2: Přeplněná odpadová nádoba na separovaný odpad s ilegální skládkou odpadu

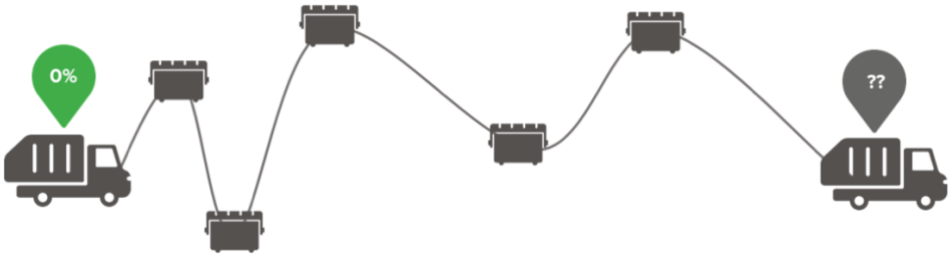
Vyvážení nenaplněných odpadových nádob

Opačný případ je vyvážení ne zcela zaplněných odpadových nádob – tedy nevyužívání potenciálu nádoby. Vyvážení nezaplněných nádob je neefektivní, protože výsypem každé nádoby vznikají zbytečné náklady. V tomto případě je důležitý model platby za službu svozu, kterou má město nebo obec uzavřenou se svozovou firmou. Pokud je domluvena paušální platba nebo dělený tarif, vznikají obci/městu náklady za zbytečný výsyp a dopravu, ale svozové firmě vysypání poloprázdné nádoby nevadí, ba dokonce to vítají, protože za ní dostane zaplaceno. Pokud je domluvena platba za hmotnost, městu nevznikají náklady na výsyp a dopravu navíc, protože to už je v ceně za vyprodukovaný odpad, náklady vznikají pouze svozové firmě.

Tento problém má také environmentální rozsah. Zbytečný výsyp odpadových nádob zvyšuje čas a počet najetých kilometrů nákladních svozových aut ve městech, tím se zvyšují lokální emise výfukových plynů a také hladina akustického smogu, což snižuje kvalitu života.

Neznalost stavu zaplnění svozového vozidla

Při svozu po fixní trase a neznalosti zaplnění nádob není jasné, kdy bude svozové auto zaplněné (obrázek 3), a tedy v jaké fázi se bude muset jet vysypat do centra na zpracování odpadu. Pokud je svoz nastaven tak, že i odklon od trasy je vždy předem definovaný po výsypu určitého počtu nádob, při nezaplněných nádobách je svozové auto taky nezaplněné, a vysypává se tedy neefektivně – ne zcela zaplněné. V případě fixního i nefixního odklonu od svozové trasy tím vznikají zbytečně najeté kilometry a tím i vzniklé náklady.



Obrázek 3: Nemožnost predikce zaplnění vozidla

Nepoctivé svozové společnosti

Dalším problémem jsou nepoctivé svozové společnosti. Jejich nepoctivost spočívá v nevyvážení odpadových nádob podle uzavřených smluv. Jsou dva hlavní důvody takového chování. První je časová tíseň při svozu, kdy zaměstnanci svozové společnosti nestíhají vyvézt všechny nádoby podle svozového plánu před koncem své směny, druhým je zaplnění auta před plánovaným odklonem z trasy. Pokud na svozové trase po zaplnění auta zbude jenom několik jednotek odpadových nádob, nepoctivé svozové firmě se nechce zvlášť vracet pro tento zbytek nádob a odpadové nádoby nejsou tedy v daný svozový cyklus vyvezeny.

Rozdělení odměn ze Systému EKO-KOM

Běžným svozovým autem s nákladním objemem kolem 20 m^3 se dají vyprázdnit desítky sběrných nádob na separovaný odpad (záleží na typu odpadu). Malé obce mají pouze malé množství odpadových nádob, a tedy jedno auto vyváží více obcí najednou. Při zaplnění auta se jede vysypat do centra na zpracování odpadu, kde se zaznamená hmotnost vytříděného odpadu na základě kterého se vyplácí odměny ze Systému EKO-KOM. Pokud jsou odpadové nádoby vyvezeny z několika obcí, odměna se poté rozpočítává dle počtu a objemu nádob v každé obci. Takové rozdělení odměn je nepřesné, protože se neví míra zaplnění nádob v jednotlivých obcích a tedy i správná produkce odpadu. Pokud obec investuje do odpadového hospodářství, vzdělává svoje obyvatelstvo a má nastavené počty nádob efektivně, aby se zcela zaplňovaly, ve výsledku bude mít menší podíl odměn než obec, která má odpadové nádoby rozmístěny neefektivně a jejíchž obyvatelstvo tolik netřídí odpad. Spravedlivé rozdělení odměn je důležité, protože příspěvky ze Systému EKOKOM, jak bylo ukázáno na příkladu v kapitole 2.1.2.1 Odměny obcím za třídění odpadu – Systém EKO-KOM, tvoří velkou část příjmů do rozpočtu odpadového hospodářství daného města/obce.

2.2 Monitoring zaplnění odpadových nádob

Nejúčinnějším řešením, jak odstranit problémy adresované v kapitole 2.1.4 Problémy nynějšího svozu odpadu je sledování zaplněnosti odpadových nádob. Při pravidelném a dostatečně frekventovaném monitoringu odpadových nádob, se získá informace o zaplněnosti, kterou je potom snadné použít k odhalení přeplňovaných, nenaplňovaných nebo nevyvážených nádob, k vytipování nádob, které mohou být zneužívány právnickými či podnikajícími fyzickými osobami, ale také k zefektivnění celkového svozu a spravedlivějšímu rozdělení odměn ze Systému EKO-KOM v případě malých měst a obcí.

2.2.1 Typy monitoringu zaplnění odpadových nádob

2.2.1.1 Evidence při svozu odpadu

Nejednoduší a zároveň dle § 39 odpadového zákona 185/2001 Sb. povinný způsob monitoringu odpadů pro obce a města je průběžná evidence odpadů. Evidence se dělá z důvodu zjištění produkce odpadů na území města či obce z čehož se kalkuluje cena za zpracování odpadu a služeb svozu. Odpad se eviduje při výsypu svozového vozidla, takže není známa produkce odpadu v jednotlivých sběrných nádobách ani sběrných místech, ale pouze z jednotlivých svozových tras. V ojedinělých případech, kdy svozové auto je vybaveno dynamickou váhou – zařízení, které dokáže vážit během výsypu odpadové nádoby – se mohou odpadové nádoby vážit zvlášť. Tento systém má ale z důvodu vysoké ceny velmi malý počet svozových aut v České republice.

Rozšířenou verzí evidence odpadů je sledování a zaznamenávání zaplněnosti odpadových nádob při výsypu. Svozová posádka při výsypu odpadové nádoby naskenuje čtečkou čárový/QR kód nebo NFC čip nalepený nádobě, pomocí kterého načte informace o nádobě a následně je doplněna manuálně informace o zaplnění. Jednodušší verze tohoto systému je ruční zapisování zaplnění do svozového protokolu. Velkou slabinou tohoto systému je ochota posádky přesně zapisovat údaje o zaplnění –

z průzkumu ve městě Chodov vyšlo, že posádka zapisuje zaplnění odpadových nádob až po dokončení směny z důvodu ušetření práce. Jak je ale popsáno v kapitole 2.2.2.1 výsledky průzkumu, tento způsob evidence je však stále spíše výjimkou.

Systém evidence odpadu dokáže zjistit množství odpadu až při svozu, tudíž není znám průběžný stav a nedá se tedy použít k řešení všech problémů adresovaných v kapitole 2.1.4 Problémy nynějšího svozu odpadu.

2.2.1.2 Průběžné sledování zaplněnosti nádob

Průběžné systémy sledování zaplněnosti nádob monitorují hladinu v odpadových nádobách kontinuálně v čase, ne tedy pouze při svozu jako v případě evidence odpadů. Nynější způsoby průběžného sledování zaplněnosti odpadových nádob se dají rozdělit na tři části – manuální, poloautomatické a automatické.

Automatické systémy pro sledování zaplněnosti nádob

Automatické systémy k získání informace o zaplněnosti nepotřebují zásah člověka a fungují tedy zcela autonomně. Systém se skládá z měřící jednotky umístěné v každé sledované odpadové nádobě, která senzoricky měří zaplnění a poté v pravidelných intervalech tuto informaci posílá pomocí bezdrátové sítě do databáze, kde se data ukládají pro další interpretaci a zpracování.

Poloautomatické systémy

Poloautomatické systémy k správné funkčnosti spoléhají na získání informace od uživatele – obyvatele, který jde vynést odpad – a poté pomáhají tuto informaci přenést na odpadový dispečink daného subjektu. Jednodušší systémy jsou realizovány pomocí QR kódu, či NFC nálepky, které při oskenování chytrým mobilním telefonem odešlou informaci, že nádoba je zaplněná. Důmyslnější systémy v sobě mají elektroniku a manuální vstup ve formě tlačítka, které při stisku odešle pomocí bezdrátové sítě informaci, že nádoba je plná a je potřeba jí vyvézt.

Tyto systémy mají velký nedostatek z důvodu neiniciativy občanů. Pro občany je snadnější odpad nechat vedle odpadové nádoby s tím, že se s tím obec/město vypořádá, než problém přeplnění nějak řešit.

Manuální systémy

V manuálním systému sledování zaplnění odpadových nádob člověk musí sám ohlásit obecnímu či městskému úřadu informaci o zaplnění. Na odpadových nádobách bývají nalepená telefonní čísla na odpadové referenty, kterým taková skutečnost může být nahlášena. Některé obce/města mají vlastní zaměstnance, jejichž částečnou náplní práce je právě takovýto sběr informací.

Nevýhody tohoto systému jsou stejné jako v poloautomatických systémech – neiniciativa občanů a tedy velmi nespolehlivá informace o zaplnění. Další problémem manuálního sledování je obtížná vizuální kontrola zaplnění v případě podzemních kontejnerů a odpadových nádob se spodním výsypem.

2.2.2 Průzkum současného stavu monitoringu zaplnění odpadových nádob v České republice

Pro zjištění současného stavu monitoringu zaplnění v český městech byl vypracován průzkum, který zjišťoval, zdali města sledují zaplnění odpadových nádob a pokud ano, jakým způsobem.

Metodika průzkumu

Byly vybrány všechny města nad 14 000 obyvatel a prozkoumány jejich webové stránky, lokální zpravodajské články a plány odpadových hospodářství (dle zákona č. 185/2001 Sb. města musí zpracovávat POH), jestli disponují informací o provádění nějaké formy automatického či poloautomatického monitoringu zaplňování odpadových nádob nebo rozšířené evidence odpadů. Pokud na webových stránkách, v POH ani nikde na webu nebyla žádná informace k dohledání, bylo usouzeno, že město nedisponuje žádným systémem pro sledování hladiny odpadů ve sběrných nádobách.

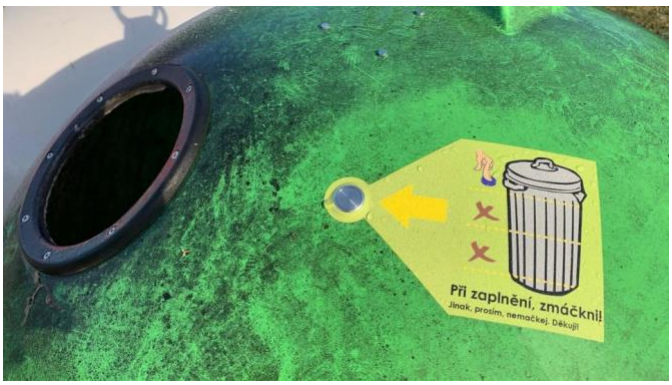
2.2.2.1 Výsledky průzkumu

Z průzkumu všech 94 měst v České republice, které mají nad 14 000 obyvatel vyplynulo, že 80 z nich v současnosti nedisponuje žádným systémem rozšířené evidence ani žádným systémem pro průběžný monitoring zaplněnosti odpadových nádob. Ve dvou z těchto 80 měst probíhal pilotní projekt, který ale byl již ukončen a nebylo v něm žádným způsobem pokračováno.

2.2.2.1.1 Ukončené projekty

Dvůr Králové nad Labem

Ve Dvoře Králové nad Labem byl v rámci pilotního projektu instalován poloautomatický systém ve formě elektroniky s tlačítkem osazené do nádob na sklo (obrázek 4). Při zaplnění nádoby, občan měl tlačítko zmáčknout a tím byl odeslán signál obecnímu úřadu o zaplněnosti.



Obrázek 4 - Pilotní projekt chytrých tlačítek ve Dvoře Králové nad Labem [5]

Projekt se neosvědčil, protože informace od občanů nebyla dostatečně spolehlivá k uskutečnění optimalizovaného svozu či nějakého jiného zásahu do systému svozu.

Krnov

V Krnově byly instalovány QR kódy na odpadové nádoby, které když obyvatel oskenoval mobilním telefonem s aplikací Zero Waste City, mohl nahlásit zaplnění [6]. Projekt již byl ukončen z důvodu nezájmu občanů.

2.2.2.1.2 Probíhající projekty

Rozšířená evidence odpadů

Rozšířená evidence odpadů ve formě nalepených QR a čárových kódů (obrázek 5) probíhá ve městech Břeclav, Zlín, Orlová, Havlíčkův Brod a Český Těšín. Svozová posádka před vysypáním každé označené odpadové nádoby musí naskenovat nalepený kód a následně do skenovacího zařízení zadat zaplnění nádoby. Tento systém pomáhá optimalizovat sběrnou síť a zlepšuje kontrolu svozových firem, jestli nádoby opravdu vyvážejí.

Obrázek 5 - Příklad rozšířené evidence odpadů ve formě čárových kódů [7]

Poloautomatický monitoring zaplnění odpadů

Ve městech Třebíč, Olomouc, Karlovy Vary, Brno, Příbram, Kolín a Praha 2 jsou nalepeny QR kódy na nadzemních odpadových nádobách, které může obyvatel telefonem oskenovat pomocí mobilní aplikace a tím nahlásit zaplnění. Údaje o zaplnění nádob jsou k dispozici v téže aplikaci. Dle vyjádření referentů pro odpadové hospodářství daných měst ale zájem občanů je tak malý, že se data nedají nijak použít. Pro správnou funkčnost by byla nutná intenzivní edukace obyvatelstva, která by ale stála mnoho peněz a času s tím, že výsledek by byl nejistý.

Poděbrady

Město Poděbrady používá k monitoringu vybraných stanovišť kamerový systém města. Nejedná se tedy o monitoring zaplnění, ale pouze vizuální kontrolu, zdali odpadové místo není přeplněné a neodkládá se tedy odpad do veřejného prostranství.

Kolín

Kolín v rámci pilotního projektu uskutečněného ve spolupráci s O2 Telefonica a.s. a svozové firmy AVE v roce 2016 osadil do podzemních nádob senzory zaplněnosti a na nadzemní nádoby nalepil QR kódy [8]. Měření v podzemních nádobách pomocí senzorů fungovalo autonomně, na ohlášení zaplnění v nadzemních nádobách bylo potřeba mít mobilní aplikaci a zaplnění zadat ručně. Toto nasazení bylo prvním velkým pokusem v České republice o automatické sledování, nicméně podle vyjádření starosty města byl projekt pouze polovičně úspěšný, protože informace o zaplněnosti sice pomohla městu

optimalizovat trasy, ale k celkovému snížení nákladů nedošlo, protože pořizovací cena 3000 Kč za senzor a provozní náklady byly moc vysoké. Po uplynutí pilotního projektu se město rozhodlo v automatickém měření pokračovat v pouze vybraných podzemních kontejnerech v počtu 21 senzorů [9].

Praha

V roce 2018 byl uskutečněn pilotní projekt chytrého svozu odpadu mezi Hl. m. Praha, integrátorem nových technologií Operátorem ICT a.s. a poskytovatelem technologie společností Sensoneo. Cílem tohoto ročního pilotního projektu bylo osadit 420 kusů senzorů do podzemních nádob a zjistit, jak projekt bude zapadat do konceptu Smart Prague 2030 a jestli bude ekonomicky profitabilní. Dle závěrečné zprávy byl označen jako přínosný a ekonomicky profitabilní a Operátor ICT se tedy rozhodl senzory odkoupit a začátkem roku 2021 v projektu pokračovat v rutinní fázi ve formě nákupu technologie od společnosti Sensoneo a provozovat projekt sám [10]. Bylo spočítáno, že NPV projektu při variantě nákupu je 14 987 073 Kč (tabulka 5) při diskontní sazbě 3,5 % a životnosti projektu 7,5 roku.

Tabulka 5 - Finanční informace pilotního projektu Chytrý svoz odpadu v Praze [10]

	Varianta nákupu	Varianta pronájmu
CAPEX	3 027 160,00 CZK	0,00 CZK
OPEX	13 688 980,00 CZK	18 370 064,00 CZK
Pozitivní přínosy	43 065 202,00 CZK	43 065 202,00 CZK
Negativní přínosy	7 822 126,00 CZK	7 821 295,00 CZK
NPV	14 987 073,00 CZK	13 775 161,00 CZK
IRR	195,34%	326,02%

3 Návrh inovovaného řešení

V průzkumu v předchozí kapitole bylo zjištěno, že drtivá většina měst nedisponuje žádným nebo nefunkčním průběžným systémem na měření zaplnění odpadových nádob, který by řešil problémy adresované v kapitole 2.1.4 Problémy nynějšího svozu odpadu. V této kapitole se bude věnováno návrhu systému, který by tyto problémy měl řešit a byl by tedy pro tyto města inovativním řešením.

3.1 Požadavky na systém monitoringu zaplnění

Hlavní úkolem nově navrženého systému je spolehlivé, přesné a frekventované měření hladiny zaplnění odpadových nádob. Dosažení tohoto cíle budou vyřešeny i všechny parciální problémy zmíněné v kapitole 2.1.4 Problémy nynějšího svozu odpadu. Další požadavky, které vyplynuly z rozhovorů s referenty měst pro odpadové hospodářství jsou následující:

- **Bezúdržbové autonomní fungování.** Systém by měl být bezúdržbový, spolehlivý a měl by sám o sobě fungovat bez zásahu člověka několik let.
- **Univerzální použití.** Systém by měl fungovat ve více typech odpadových nádob. Zejména v podzemních kontejnerech, 1100 litrových nádobách na separovaný sběr a nádobách se spodním výsypem. Dále by měl fungovat se všemi typy odpadových materiálů.
- **Frekventované měření a hlášení.** Systém by měl měřit hladinu odpadu několikrát denně a data reportovat alespoň jednou denně.
- **Jednoduchá obsluha a interpretace dat.** Data by měly být jednoduše a přehledně interpretována, aby z nich byly na první pohled jasné potřebné informace bez potřeby hlubší analýzy.
- **Příznivé provozní a investiční náklady.** Celý systém by měl být přínosný jak z ekonomického, tak i neekonomického hlediska. Jako referenční provozní a investiční náklady budou použity informace ze závěrečné zprávy pilotního projektu v Praze, která projekt přijala a uznala ho jako ekonomicky rentabilní. Finanční analýza zvolené varianty – nákupu technologie – počítala s nákupem a nasazením 1000 ks měřících jednotek, životností projektu 7,5 roku a diskontní sazbou 3,5 % s celkovými provozními výdaji (OPEX) 13 688 980 Kč a investičními výdaji (CAPEX) 3 027 160 Kč viz. tabulka 5. Po přepočítání provozních a investičních výdajů celého projektu na jeden senzor, vychází CAPEX na 3027 Kč a OPEX 152 Kč za měsíc (tabulka 6) ve variantě nákupu. V hodnotě OPEX jsou započítány i fixní provozní náklady, které s počtem měřících jednotek nesouvisí, ale je předpokládána jejich nízká hodnota a budou zanedbány. Příznivé investiční náklady nově navrženého systému jsou tedy stanoveny na maximálně 3000 Kč za měřící jednotku s měsíčním provozním nákladem 150 Kč za měřící jednotku.

Tabulka 6 - přepočítané výdaje na jeden senzor projektu chytrého svozu odpadu

	Varianta nákupu	Varianta pronájmu
CAPEX/1 senzor	3 027 CZK	0 CZK
OPEX/1 senzor/měsíc	152 CZK	204 CZK

- **Funkčnost na celém území České republiky.** Je důležité, aby systém fungoval na celém území České republiky a nebyl nijak geograficky omezen.

3.2 Návrh systému pro sledování zaplnění odpadových nádob

Z průzkumu (kapitola 2.2.2) vyplynulo, že manuální a poloautomatické systémy jsou pro naplnění požadavků zmíněných v předchozí kapitole nefunkční. Navržený systém tedy bude automatického typu skládající se z měřících jednotek v odpadových nádobách a serveru pro uložení dat do databáze, zpracování a prezentaci dat ve formě webové aplikace (obrázek 6). Měřící jednotky budou disponovat senzorikou pro měření zaplněnosti a modulem pro bezdrátovou komunikaci pro přenos dat.

Obrázek 6 - Koncept navrženého systému

3.2.1 Měřící jednotka

Nejdůležitější topologické aspekty měřící jednotky jsou výběr správné senzoriky a komunikační sítě pro přenos dat.

3.2.1.1 Výběr senzoriky

Hlavním úkolem senzoru je změření zaplněnosti. Zaplněnost se dá reprezentovat jako hladina odpadu v nádobě neboli vzdálenost mezi horní hranou odpadové nádoby a hladinou odpadu. Požadavky na senzor vzdálenosti jsou nízká cena, nízká spotřeba energie, rozsah měřitelné vzdálenosti od jednotek centimetrů po minimálně 150 cm (nejhlubší podzemní odpadové nádoby v Praze mají hloubku 150 cm) a schopnost měřit vzdálenost od libovolných materiálů. Pro změření vzdálenosti, a tedy i zaplněnosti, byly uvažovány následující řešení:

Ultrazvukový senzor

Ultrazvukový senzor funguje na bázi emitace zvuku o vysoké frekvenci (40 kHz), který je vyslán směrem k měřenému objektu, poté se od objektu odrazí a letí zpět, kde je senzorem zachycen (obrázek 7). Čas mezi odesláním a přijmutím ultrazvukového signálu t_{sig} je změřen a díky rychlosti zvuku vypočítané ze vzorcem 1, kde t_c je teplota [°C] a dosazením do vzorce 2, lze získat vzdálenost překážky (hladiny odpadu) od senzoru.

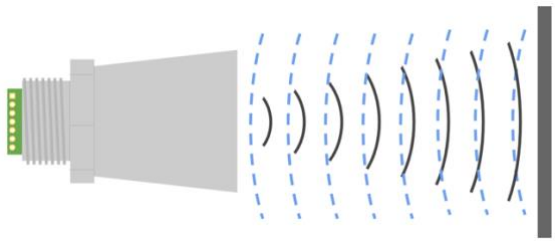
$$v_{zvuk} = 20{,}05 * \sqrt{273.16 * t_c}\ [ms^{-1}] \quad (1)$$

$$vzdálenost = \frac{t_{sig} * v_{zvuk}}{2}\ [m] \quad (2)$$

Vyslaný paprsek ultrazvuku má kuželovitý tvar s úhlem α vypočítaným ze vzorce 3, kde *v* je rychlost vlny, *d* průměr vysílače a *f* frekvence signálu. Obvyklý úhel α se u komerčních ultrazvukových senzorů pohybuje kolem 30° [11]. Ultrazvukový signál se odrazí od nejbližší překážky, která zasahuje do vyslaného kuželovitého signálu.

$$\alpha = sin^{-1}\left(1.2\frac{v}{d * f}\right)[°] \quad (3)$$

Ultrazvukové senzory jsou levné s cenou pod 10 USD, fungují s odrazem téměř všech materiálů, dokážou měřit vzdálenosti od jednotek centimetrů až po několik metrů a mají v provozním stavu nízkou spotřebu el. proudu řádově do 20 mA.



Obrázek 7 - Princip fungování ultrazvukového senzoru [12]

Infračervený senzor

Jsou dva způsoby měření infračerveným senzorem – pomocí triangulace a měření času letu (ToF – time of flight) pulzu. V obou případech je vyslán infračervený světelný pulz, který se odrazí od překážky a letí zpět do senzoru, kde je detekován. V případě triangulace je známá vzdálenost *d* mezi vysílačem a přijímačem na IR senzoru. K získání vzdálenosti k měřenému objektu je potřeba změřit úhel α, pod kterým se vyslaný paprsek vrátil zpět do senzoru a následně se pomocí vzorce 4 vypočítá vzdálenost *l* od objektu. Schématické zobrazení je na obrázku 8 vlevo.

$$l = \frac{d}{\tan(90° - \alpha)}\ [m] \quad (4)$$

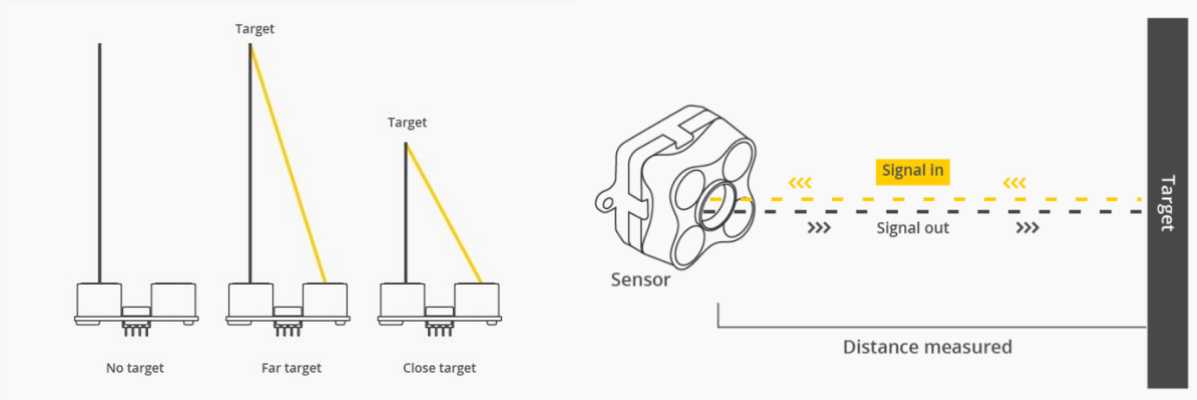
Tento způsob měření má měřící rozsah od jednotek centimetrů až po několik metrů – záleží na konkrétním senzoru. Ceny IR senzorů na principu triangulace se pohybují od 15 USD výš.

ToF IR měření je velmi podobné měření ultrazvukem – vyšle se IR signál a změří se čas t_{sig}, který udává, jak dlouho letí, než se vrátí zpět do senzoru. Díky známé rychlosti světla 299 792 458 ms^{-1} se dá pomocí vzorce 5 poté spočítat vzdálenost *l* objektu od senzoru (schématický nákres na obrázku 8 vpravo).

$$l = \frac{299\,792\,458 * t_{sig}}{2}\ [ms^{-1}] \quad (5)$$

IR ToF měření má rozsah od desítek centimetrů po několik desítek metrů. Ceny senzorů jsou od 100 USD výš.

Problémem u IR měření je sluneční svit, který také obsahuje infračervenou složku a může tedy měření rušit.



Obrázek 8 - Princip fungování IR senzoru [13]

Laserový senzor

Laserový senzor funguje na stejném principu jako IR ToF a ultrazvukové měření. Senzor vyšle proud fotonů, který se odrazí od překážky a letí zpět. Po přijetí navráceného signálu se změří čas t_{sig} a díky známé rychlosti světla 299 792 458 ms^{-1} se dá pomocí vzorce 5 spočítat vzdálenost l objektu od senzoru. Druhou, méně častou variantou detekce vzdálenosti je detekce fázového posunu signálu, z které je poté vypočítána vzdálenost objektu od senzoru.

Cena laserových senzorů začíná na 10 USD a jejich měřící rozsah je od jednotek centimetrů až po několik metrů.

Shrnutí a závěr

V tabulce 7 je binární shrnutí požadavků z úvodu kapitoly a výsledky jednotlivých zkoumaných senzorů.

Tabulka 7 - Shrnutí požadavků senzoriky

	Měření vzdálenosti od libovolných materálů	**Nízká spotřeba energie**	**Správný rozsah měření**	**Nízká cena**
Ultrazvuk	Ano	Ano	Ano	Ano
IR triangulární	Ano	Ano	Ano	Ne
IR ToF	Ano	Ano	Ne	Ne
Laser	Ano	Ano	Ano	Ano

Kvůli vyhovujícím technickým vlastnostem a nejpříznivější ceně, je vybráno ultrazvukové měření jako způsob sledování zaplněnosti odpadových nádob.

3.2.1.2 Výběr komunikační sítě

Komunikační síť bude v systému sloužit pro komunikaci mezi serverem a měřícími jednotkami. Z jednotek budou odcházet měřená a provozní data, ze serveru budou odesílány do měřících jednotek provozní příkazy a informace. Jediné dva technické požadavky na síť tedy jsou dobrá dostupnost po celé České republice a to v ideálně dobré a silné kvalitě signálu – kvůli umístění měřících jednotek do podzemních kontejnerů – a oboustranná komunikace. Energetická náročnost není tak důležitá, protože měřící jednotky budou komunikovat pouze jednou nebo dvakrát denně po dobu několika vteřin, zbytek dne budou spát. Nejdůležitější aspekt výběru je nízká provozní cena. Uvažované varianty byly následující:

SigFox

Specializovaná síť pro IoT fungující ve 868 MHz frekvenčním pásmu s rychlostí dosahující až 100 kb/s. V České republice veřejnou síť provozuje SimpleCell Network a.s. Roční cena jednoho zařízení je 140 Kč při dvou uploadech a jednom downloadu denně [14].

Lora

Lora je bezdrátová technologie také vyvinutá hlavně pro účel IoT. V Evropě pracuje v bezlicenčním frekvenčním pásmu 169 MHz, 433 MHz a 868 MHz, v USA 915 MHz. Její hlavní výhoda je relativně velký dosah od vysílače (více než 10 km v příznivém terénu) a malá spotřeba energie jak při odesílání, tak přijímání dat. Rychlosti dosahují až 50 kb/s [11]. Roční cena pro jedno zařízení je od společnosti Starnet s.r.o. 229 Kč při kapacitě odeslání 100 zpráv a přijmutí 5 zpráv denně [15].

NB-IoT

NB-IoT je další specializovaná síť pro IoT vyznačující se nízkou spotřebou energie, oboustrannou komunikací a pokrytím celé České republiky. Technologie pracuje v LTE pásmu a dosahuje mnohonásobku rychlosti oproti ostatním specializovaným IoT sítím. V České republice síť provozuje Vodafone a O2 Telefonica. Dle O2 Telefonica cena jednoho zařízení pro nízký přenos dat je zhruba 25 Kč za měsíc, nicméně konkrétní cena záleží na individuální nabídce.

GSM

GSM je nejpoužívanější světová mobilní síť sloužící k přenosu hlasu a dat (GPRS, EDGE). Jedná se o síť druhé generace (2G) a je tedy předchůdcem 3G, LTE, 4G a 5G sítí. Síť pracuje na frekvenčním pásmu 1800Mhz (Band 3) a 900Mhz (Band 8). Oproti IoT sítím je potřeba vyšší výkon pro navázání komunikace a je tedy více energeticky náročná. Rychlosti dosahují až 50 kb/s. Výhoda této sítě je nejhustější pokrytí ze všech veřejných komunikačních sítí a díky velmi kompetitivnímu trhu i velmi příznivá cena při přenosu malého množství dat. Z jednání s českými operátory vyplynulo, že cena za

komunikaci pro jednu měřící jednotku by se mohla pohybovat kolem 10 Kč/měsíc při individuální domluvě odvíjející se zejména od množství přenesených dat.

O LTE a 4G sítích nebylo uvažováno, protože jedinou výhodu oproti GSM 2G je vyšší přenosová rychlost, která v navrženém systému měření zaplnění odpadových nádob není potřeba. 5G sítě se v době psaní této práce (2Q 2021) v České republice teprve testují.

Závěr

Všechny navržené sítě by mohly být použity v aplikaci pro přenos dat z měřících jednotek v odpadových nádobách. Nakonec byla ale vybrána GSM síť z důvodu nejnižších provozních nákladů, nejlepšímu pokrytí v České republice a nejnižší pořizovací ceně komunikačních modulů.

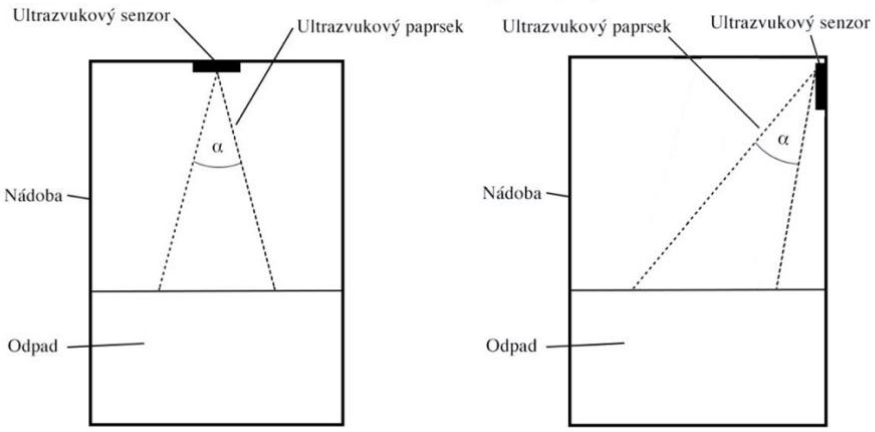
3.2.2 Databáze a webová aplikace

Data z měřících jednotek se musí nějak strukturalizovaně ukládat, aby byla možná jejich jednoduchá interpretace. K ukládání dat poslouží databázový server, na kterém poběží relační databáze. Jelikož se nečeká velké množství dat, není potřeba robustní databáze a výkonný server. Jediné požadavky jsou relační typ databáze a individuální přístup k datům – tj. možnost nastavení jiných práv uživatelů k čtení a editaci dat, protože se neplánuje dělat API vrstva. Z důvodu zkušeností autora této práce, splnění všech požadavků a GPL v2 licence (zdarma k použití pro komerční účely) byla vybrána MySQL databáze.

Aby interakce mezi daty a uživatelem byla uživatelsky přívětivá, bude vytvořena webová aplikace, jejíž úkolem bude přehledné grafické a tabulkové znázornění dat. Webová aplikace kvůli rychlému přístupu k datům poběží na stejném serveru jako databáze a jejíž programovací stack se bude skládat z frameworku Flask (programovací jazyk Python) pro backend a HTML, CSS a JS pro frontend. Pro jednodušší a přehlednější kód bude použita CSS knihovna Bootstrap 5. Flask bude použit, protože se jedná o „lightweight“ framework vhodný pro prototypování webových aplikací a je psaný v programovacím jazyce Python, který bude zároveň použit pro zpracování dat z měřících jednotek. Udržení co nejmenšího programovacího stacku je výhodné pro vývoj, protože odpadá nutnost se učit novou syntaxi případně najímat nového programátora. HTML, CSS a JS jsou standardní značkovací a programovací jazyky pro webový vývoj frontendu.

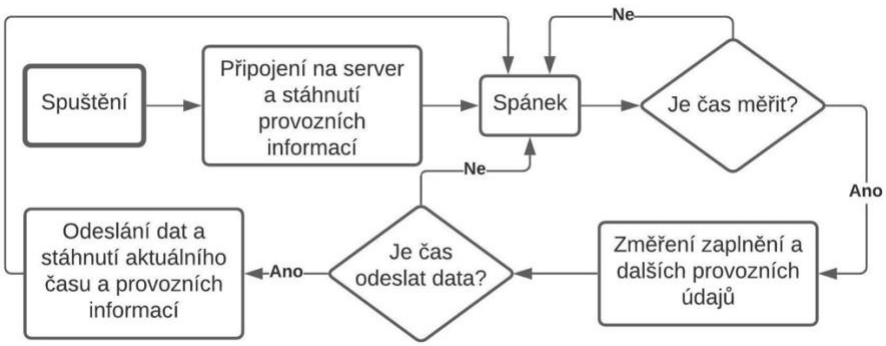
4 Realizace měřící jednotky

Jak už bylo řečeno, úkolem měřící jednotky je měření zaplněnosti odpadové nádoby a pravidelné reportování na server do databáze. Měřící jednotka bude umístěna v každé nádobě na horní straně buď na stropě nebo na boku odkud bude snímat ultrazvukovým senzorem zaplnění (obrázek 9). Úhel α je úhel paprsku ultrazvukového senzoru vypočtený ze vzorce 3.



Obrázek 9 - Umístění měřící jednotky

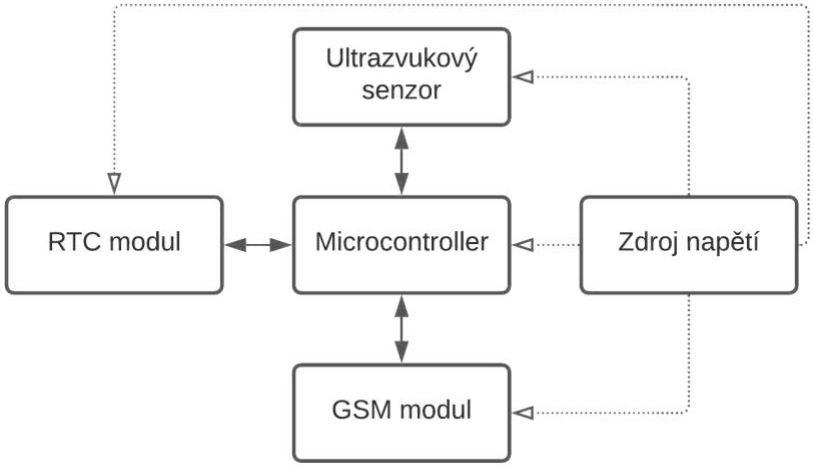
Kvůli požadavku několikaleté výdrže, je nutno co nejvíce šetřit elektrickou energii, a tedy jednotka bude většinu času ve stavu spánku. Bude se pouze probouzet pro provedení měření a odeslání dat. Pracovní proces měřící jednotky je znázorněn ve vývojovém diagramu na obrázku 10 a začíná prvotním spuštěním, kdy se do měřící jednotky vloží baterie. Poté se jednotka připojí na server, stáhne si provozní informace – aktuální čas, frekvenci měření a časy odeslání dat – které jsou potřebné ke správné funkci a usne. Spí do té doby, než přijde čas pro měření. Po změření zaplnění a provozních údajů se rozhodne, jestli se mají data i odeslat nebo pouze uložit do paměti. Pokud ne, jednotka dál spí, pokud ano, data se odešlou, jednotka si pro korekci ze serveru stáhne aktuální čas a provozní informace a usne. Tento proces se stále cyklicky opakuje.



Obrázek 10 - Vývojový diagram procesu měřící jednotky

4.1 Návrh hardwaru

Návrh jednotlivých komponent hardwaru vychází z vývojového diagramu procesu funkčnosti (obrázek 10). Celý proces funkčnosti je potřeba řídit – o to se postará mikrokontrolér. Mikrokontrolér je integrovaný obvod, který v sobě obsahuje kompletní mikropočítač a nejsou tedy potřeba podpůrné paměťové obvody pro uložení programu, operačních proměnných a zásobníku. Dále je potřeba ultrazvukový senzor pro měření zaplnění, GSM modul pro komunikaci s GSM sítí a RTC modul. RTC (real time clock) je modul reálného času, který v sobě udržuje čas a probouzí mikrokontrolér ze spánku v momentě, kdy má měřit či odeslat data. Pro funkčnost všech komponentů, musí být měřící jednotka vybavena zdrojem elektrické energie. Na blokovém diagramu na obrázku 11 jsou vidět jednotlivé komponenty a vztahy mezi nimi. Plná čára s šipkami značí směr toku dat, tečkovaná čára značí tok elektrické energie.



Obrázek 11 - Blokový diagram měřící jednotky

4.1.1 Výběr komponentů

V této podkapitole se bude věnováno výběru komponentů. Komponenty budou vybírány na základě splnění technických požadavků, ceny a zkušenosti a preference autora práce. Zdroj napětí zde nebude probírán – jemu bude věnována samostatná sekce v kapitole 4.1.2 Power management.

Mikrokontrolér

Jak bylo zmíněno výše, mikrokontrolér je mozek celé měřící jednotky a bude tedy sloužit k řízení celého procesu měření odpadu. Technické požadavky jsou následující:

- **Alespoň 15 GPIO pinů**. Pomocí GPIO pinů bude mikrokontrolér komunikovat s ostatními moduly. Předpokládaná potřeba pinů je následující: GSM modul dva piny, ultrazvukový senzor dva piny, RTC modul tři piny, serialová komunikace s počítačem pro debugging dva piny, led diody pro signalizaci dva piny, 5 pinů do rezervy
- **ADC převodník**. Čip musí být vybavený ADC převodníkem pro měření napětí baterií.
- **UART**. Čip musí disponovat UART obvodem pro sériovou komunikaci s PC pro debugging.

- **Nízká spotřeba a režim spánku**. Nízká spotřeba a režim hlubokého spánku s potřebou jednotek mikroampér jsou klíčové parametry pro dlouhou životnost baterie.
- **Velký rozsah pracovních teplot**. V odpadových nádobách může být přes zimu teplota hluboko pod bodem mrazu, v létě na druhou stranu velmi vysoké teploty z důvodu přímého slunce. Rozsah pracovních teplot byl zvolen od -20 °C do 60 °C.
- **Vnitřní teploměr**. Měření teploty bude použito jako jeden z provozních údajů.
- **Externí přerušení.** Přerušení (angl. Interrupt) říká mikrokontroléru, aby zastavil to, co právě dělá a exekuoval nějakou předdefinovanou sekvenci příkazů (ISR – interrupt service routine). V této aplikaci je potřeba vlastnost externího přerušení, které bude sloužit k probuzení mikrokontroléru ze spánku.

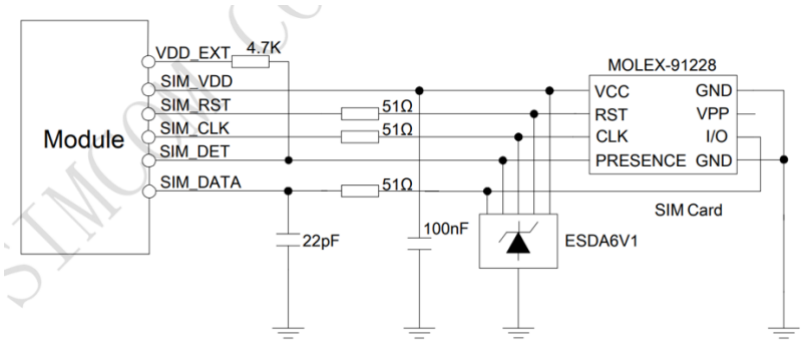
Autor této práce má zkušenosti s rodinou mikrokontrolérů AVR výrobce Microchip. Kvůli této zkušenosti, příznivé ceně a splnění všech požadavků výše, byl vybrán mikrokontrolér Microchip ATmega328p. Tento čip má frekvenci hodin až 20 MHz, pracovní napětí 1.8 až 5.5 voltu, rozsah pracovních teplot od -40 °C do 85 °C, 5 ADC pinů, 25 GPIO pinů, UART, přerušení, vnitřní teploměr a režim hlubokého spánku při spotřebě až 0.1 mikroampéru [16].

Ultrazvukový senzor

Ultrazvukových senzorů jsou na trhu doslova desítky s velmi podobnými vlastnostmi. Požadavky jsou rozsah měření od jednotek centimetrů po 1,5 metru a pracovní napětí do 5 voltů. Voděodolnost není potřeba, protože senzor bude zakrytovaný na spodní straně měřící jednotky. Byl vybrán senzor Groove ultrasonic od společnosti Seeed Studio. Jedná se o vylepšenou verzi nejpoužívanějšího spolehlivého ultrazvukového senzoru HC SR-04. Dokáže měřit od 3 do 350 cm, pracovní teplota je od -20 °C po 75 °C, pracovní napětí od 3,2 do 5,2 voltů [17] a disponuje příznivou cenou. Ultrazvukový senzor je hotový modul a nepotřebuje žádné podpůrné obvody.

GSM modul

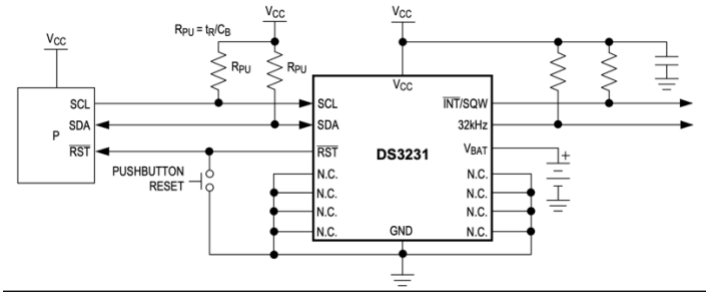
Kritické požadavky pro GSM modul jsou funkce na frekvencích využívaných v České republice (900 a 1800 MHz) a schopnost uskutečňovat datové přenosy (GPRS nebo EDGE). Byly uvažovány čtyři GSM moduly splňující tyto technické požadavky – Simcom SIM800L/C, Neoway 590E, Goouuu Tech Mini IOT-GA6 a Ai-Thinker A9. Všechny moduly mají podobné technické parametry a cenu kolem 10 USD. Z důvodu pozitivních referencí od pracovníků společnosti Jablotron a existenci prohlášení o shodě (CE), byl vybrán modul SIM800C od společnosti Simcom. Pracovní napětí modulu je 3,5 až 4,5 voltu a disponuje komunikací přes UART a I^2C. Kromě podpůrného napěťového obvodu bude muset k modulu být přidán slot na SIM kartu podle referenčního obvodu na obrázku 12 v případě 8 pinové SIM karty. Pokud operátor dodá 6 pinovou SIM kartu, piny *VDD_EXT* a *SIM_DET* na obrázku 12 zůstanou volné.



Obrázek 12 - Referenční objekt SIM slotu pro GSM modul SIM800 [18]

RTC modul

Pro dosažení co nejdelšího života baterie měřící jednotky, celý systém bude spát s výjimkou RTC modulu. Modul reálného času v sobě dokáže udržovat čas s mnohem vyšší přesností a nižší spotřebou než mikrokontrolér. Při prvotním spuštění měřící jednotky si mikrokontrolér stáhne provozní informace ze serveru a nastaví si RTC modul tak, aby ho budil periodicky či v přesně stanovený čas pro měření a odeslání informací. Požadavky na RTC modul jsou tedy následující: nízká spotřeba energie při udržování času, možnost periodického alarmu (probuzení mikrokontroléru pro měření), alespoň dva alarmy s možností nastavení přesného času (pro odesílání dat) a standardní formou sériové komunikace přes UART či I^2C. Byl vybrán RTC modul DS3231 z důvodu velkého rozšíření, spotřebou 1 µA při režimu udržování času, komunikací I^2C, dvěma alarmy, velmi přesným udržováním času s teplotní kompenzací, širokým pracovním napětím od 2,3 do 5,5 voltu a velmi přesným měřením teploty [19] – nebude tedy nutno měřit teplotu méně přesným vnitřním teploměrem mikrokontroléru. Zároveň tento čip nepotřebuje k funkci složitý podpůrný obvod, jak je vidět na obrázku 13 typického operačního zapojení.



Obrázek 13 - Typické operační zapojení DS3231 RTC modulu [19]

4.1.2 Power management

Správný management elektrické energie je klíčový pro spolehlivé fungování měřící jednotky. V této sekci se bude zabýváno výběrem zdroje napětí a výkonových součástek, tak aby to odpovídalo požadavkům jednotlivých komponent a technicko-ekonomickým požadavkům v kapitole 3.1 Požadavky na systém monitoringu zaplnění.

4.1.2.1 Výběr zdroje napětí

Měřící jednotka bude napájena z baterie, která musí vydržet pracovat po dobu několika let bez výměny v zvoleném rozsahu tepelných hodnot od -20 °C do 60 °C. Baterie trpí na efekt samovybíjení a je tedy nutné vybrat takovou chemii baterií, která má hodnotu samovybíjení co nejnižší. V tabulce 8 jsou orientační hodnoty samovybíjení jednotlivých typů baterií. Nejnižší hodnotu samovybíjení mají alkalické a lithiové baterie. Byl vybrán alkalický typ baterie z důvodu nižší ceny.

Tabulka 8 - Odhadované hodnoty samovybíjení různých typů baterií [20]

Battery system	Odhadovaná hodnota samovybíjení
Primary lithium-metal	10% za 5 let
Alkaline	2–3% za rok (7-10 života)
Lead-acid	5% za měsíc
Nickel-based	10–15% za 24 h, poté 10-15% za měsíc
Lithium-ion	5% za 24 h, poté 1–2% za měsíc

Dále je nutné se podívat na napěťové požadavky jednotlivých komponent.

- Mikrokontrolér – **1,8 až 5,5 V**
- RTC modul **2,3 až 5,5 V**
- Ultrazvukový senzor **3,2 až 5,2 V**
- GSM modul **3,5 až 4,3 V**. Doporučené napětí **4 V**.

Z požadavků na napětí je vidět, že nejcitlivější v celém systému je GSM modul - shora omezuje napětí na 4,3 voltu a zdola na 3,5 voltu. Aby systém fungoval, musí být napětí po celém dobu provozu právě v tomto rozsahu. Bylo rozhodnuto, že budou použity tři sériově zapojené alkalické AA/FR6 baterie, které mají nominální napětí 1,5 voltu a úplně vybité 1,1 voltu. Zdroj napětí složený ze tří sériové zapojených baterií bude plně nabitý mít až 4,5 voltu, plně vybitý 3,3 voltu. V tomto rozsahu napětí jsou schopny všechny komponenty pracovat kromě GSM modulu. Před GSM modul bude dán lineární regulátor napětí, který vytvoří fixní napěťovou hladinu na maximálně 4 voltech.

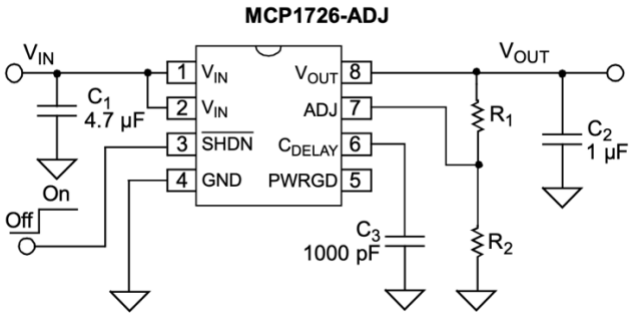
Napěťový regulátor pro GSM modul

GSM modul SIM800 má v hlubokém spánku spotřebu 1,2 mA [18], což z pohledu nízkoenergetické aplikace, která má vydržet několik let na baterie, je příliš. GSM modul se tedy bude úplně odpojovat při spánku od napětí. K tomu bude sloužit napěťový regulátor. Požadavky na napěťový regulátor tedy jsou: výstupní napětí 4 volty, možnost odpojit výstup od napětí, nízký pokles napětí (LDO regulátory), nízká spotřeba energie při spánku a výstupní proud alespoň 500 mA kontinuálně.

Byl vybrán regulátor Microchip MCP1726-ADJ, který splňuje všechny zmíněné požadavky. ADJ v názvu znamená adjusted – tj. tento regulátor má nastavitelné výstupní napětí podle vzorce 6, kde V_{adj} je typicky 0,41 voltu.

$$V_{out} = V_{adj}\left(\frac{R_1 + R_2}{R_2}\right) \quad (6)$$

Byly vybrány standardní rezistorové hodnoty R_1 = 88 700 Ohm, R_2 = 10 000 Ohm, které dohromady dělají výstupní napětí V_{out} 4,05 V. Typické zapojení regulátoru je na obrázku 14.



Obrázek 14 - Typické zapojení regulátoru Microchip MCP1726-ADJ [21]

Prvotní odhad výdrže baterií

Prvotní odhad výdrže měřící jednotky bude určen z údajů z technických dokumentací jednotlivých komponent. Bude se jednat o velmi hrubý odhad, protože bez reálného proměření průběhů elektrických proudů, nejde přesně zjistit spotřeba elektrické energie. Funkční cyklus se dělí na tři části – spánek, měření a odesílání. Při spánku je zapnutý pouze RTC modul a mikrokontrolér v režimu spánku. Při režimu měření je zapnutý mikrokontrolér, RTC modul a ultrazvukový modul. Při režimu odesílání je zapnutý mikrokontrolér a GSM modul. Typické elektrické proudy jednotlivých zařízení dle technických dokumentací jsou následující:

- Mikrokontrolér (spánek) – 1 µA.
- Mikrokontrolér (zapnutý) – 10 mA.
- Ultrazvukový senzor – 25 mA.
- GSM modul – 200 mA.
- RTC modul – 1 µA.

Předpoklady:

- Měření trvá 1 vteřinu
- Odesílání trvá 20 vteřin
- Měří se každé 2 hodiny – tj. 12krát za den
- Odesílá se jednou za den
- Ztráty přes kondenzátory a jiné ≈ 3 µA
- Konstantní samovybíjení 2 % z kapacity baterie za rok ≈ 0,137 mAh/den

Spotřeba proudu jednotlivých režimů:
Spánek (I_s)= 1 μA (mikrokontrolér) + 1 μA (RTC modul) + 3 μA (ztráty) = **5 μA**
Měření (I_m) = 10 mA (mikrokontrolér) + 1 μA (RTC) + 3 μA (ztráty) + 25 mA (ultrazvuk) ≈ **35 mA**
Odesílání (I_o) = 10 mA (mikrokontrolér) + 1 μA(RTC) + 3 μA (ztráty) + 200 mA (GSM) ≈ **210 mA**

Denní spotřeba elektrické energie E_{den} v mAh je

$$E_{den} = T_s * I_s + T_m * I_m + T_o * I_o + E_{bat} = \frac{86368*5*10^{-3}+12*35+20*210}{3600} + 0{,}137 \approx \mathbf{1{,}537\ mAh} \quad (10)$$

kde T_s [h] je celkový denní čas spánku, T_m [h] celkový denní čas měření, T_o [h] celkový denní čas odesílání, E_{bat} [mAh] ztracená denní kapacita z důvodu samovybíjení.

Typická kapacita alkalických AA (FR6) baterií je 2500 mAh – zapojením tří do série, jak bylo rozhodnuto, se zvýší napětí, ale kapacita zůstává stejná. Při zanedbání veškerých externalit, ze spotřeby energie a kapacity baterií vychází (kapacita/E_{den}), že měřící jednotka vydrží fungovat 1626 dní ≈ 4,5 roku. Reálná hodnota ale bude nižší, protože klíčový parametr výdrže je vnitřní odpor baterií.

4.1.2.2 Vnitřní odpor baterií

Baterie nejsou ideálním zdrojem napětí, protože disponují vnitřním odporem. To je především problém v případě odběru vyššího proudu. Např. GSM modul SIM800 má podle své technické dokumentace při registraci do sítě a odesílání dat až 2A proudové špičky (obrázek 16), které při vyšším vnitřním odporu můžou způsobit takový pokles napětí, že se celý systém resetuje nebo některé komponenty přestanou fungovat správně.
Pokles napětí (V_{drop}) na baterii vychází z Ohmova zákona a je tedy závislý na proudu (I) a vnitřním odporu baterie (IR). Dá se vypočítat z vzorce 7. Je předpoklad, že první provozní důvod, proč měřící jednotka přestane fungovat, bude právě kvůli zvyšujícímu se vnitřnímu odporu baterií a tím i poklesu napětí pod únosnou hranici při energeticky náročných operacích.

$$V_{drop} = I * IR \quad (7)$$

Vnitřní odpor je funkcí teploty a stavu nabití baterie, které se zjednodušeně dá reprezentovat úrovní napětí. Pro přesnější odhad výdrže měřící jednotky na tři alkalické AA baterie, bylo provedeno měření, jehož cílem bylo zjistit alespoň přibližnou závislost vnitřního odporu IR na teplotě a napětí na prázdno alkalických baterií AlzaPower Super Alkaline LR6. Tyto konkrétní baterie byly vybrány náhodně – předpokládá se, že hodnoty budou u všech alkalických baterií podobné.

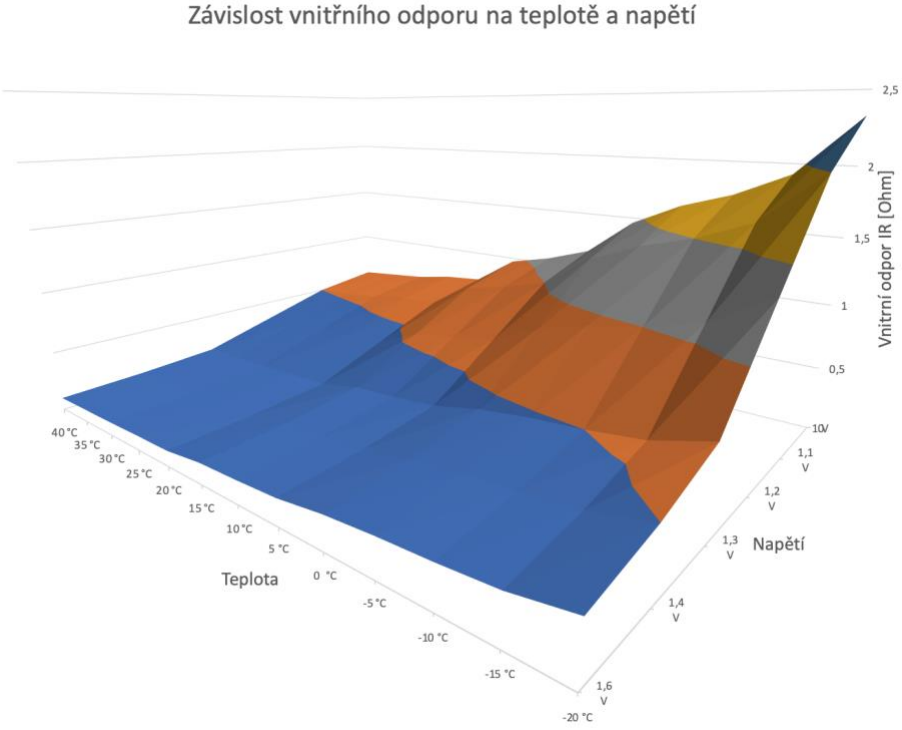
Měření vnitřního odporu alkalických baterií

Měření bylo uskutečněno v rozsahu teplot od -20 °C do 40 °C vždy po pěti stupních a v rozsahu napětí na prázdno od 1,6 voltu až do 1 voltu. Byla použita metoda stejnosměrné odporové zátěže, jejíž princip

a vzorce jsou dostupné např. na webové stránce https://learn.sparkfun.com/tutorials/measuring-internal-resistance-of-batteries/internal-resistance. Výsledky měření jsou v tabulce 9, grafické znázornění v obrázku 15.

Tabulka 9 - Výsledky měření vnitřního odporu alkalických baterií

Teplota \ Napětí	**1 V**	**1,1 V**	**1,2 V**	**1,3 V**	**1,4 V**	**1,6 V**
-20 °C	2,33 Ω	2,00 Ω	1,33 Ω	0,67 Ω	0,50 Ω	0,40 Ω
-15 °C	1,92 Ω	1,65 Ω	1,10 Ω	0,55 Ω	0,41 Ω	0,33 Ω
-10 °C	1,75 Ω	1,50 Ω	1,00 Ω	0,50 Ω	0,38 Ω	0,30 Ω
-5 °C	1,63 Ω	1,40 Ω	0,93 Ω	0,47 Ω	0,35 Ω	0,28 Ω
0 °C	1,46 Ω	1,25 Ω	0,83 Ω	0,42 Ω	0,31 Ω	0,25 Ω
5 °C	1,17 Ω	1,00 Ω	0,67 Ω	0,33 Ω	0,25 Ω	0,20 Ω
10 °C	1,05 Ω	0,90 Ω	0,60 Ω	0,30 Ω	0,22 Ω	0,18 Ω
15 °C	0,97 Ω	0,83 Ω	0,55 Ω	0,28 Ω	0,21 Ω	0,17 Ω
20 °C	0,73 Ω	0,62 Ω	0,42 Ω	0,21 Ω	0,16 Ω	0,12 Ω
25 °C	0,70 Ω	0,60 Ω	0,40 Ω	0,20 Ω	0,15 Ω	0,12 Ω
30 °C	0,64 Ω	0,55 Ω	0,37 Ω	0,18 Ω	0,14 Ω	0,11 Ω
35 °C	0,61 Ω	0,53 Ω	0,35 Ω	0,18 Ω	0,13 Ω	0,11 Ω
40 °C	0,58 Ω	0,50 Ω	0,33 Ω	0,17 Ω	0,12 Ω	0,10 Ω



Obrázek 15 - Grafické znázornění výsledků měření vnitřního odporu alkalických baterií

Z tabulky 9 je vidět, že s klesající teplotou, prudce roste vnitřní odpor baterií. Jelikož baterie v měřící jednotce budou zapojeny do série, vnitřní odpory jednotlivých baterií se budou sčítat. Vysoký vnitřní odpor při nízkých teplotách by tedy způsoboval v kombinaci s proudovými špičkami GSM modulu

(obrázek 16) takový pokles napětí, že měřící jednotka by nebyla funkční. Je tedy nutné přidat lokální zdroje energie v podobě kondenzátorů.

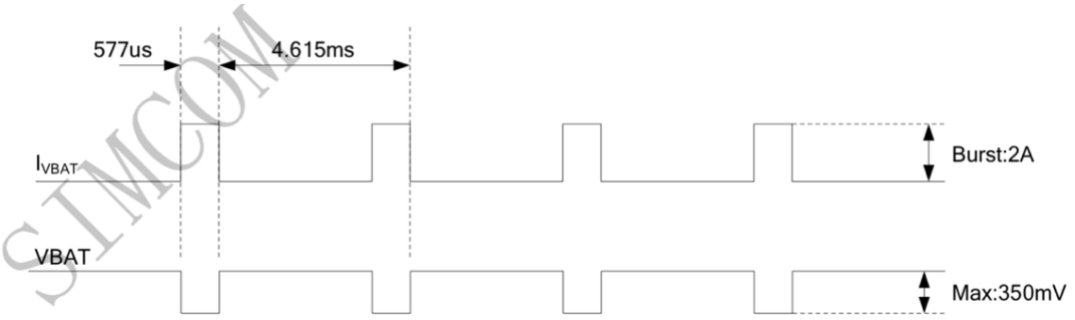
4.1.2.3 Kondenzátory

Kondenzátory slouží v obvodu pro zlepšení kvality elektrické energie odfiltrováním střídavé složky napětí či vykrytím různých proudových špiček. Vysokofrekvenční rušení může být způsobeno indukcemi v obvodu nebo integrovanými obvody pulzním oděrem proudu např. při změně stavu hradla. Pro filtraci tohoto rušení se zpravidla používají keramické blokovací kondenzátory v rozsahu hodnot 0,01 – 0,1 μF umístěné co nejblíž integrovanému obvodu, pro nízkofrekvenční rušení se používá hodnota 10 μF [22]. Hodnoty a rozmístění blokovacích kondenzátorů bývají v dokumentaci každého integrovaného obvodu, což bude také využito jako hlavní informační zdroj při návrhu schématu této aplikace.

Proudové špičky GSM modulu

Všechny komponenty, kromě GSM modulu, v měřící jednotce mají poměrně stálou spotřebu proudu nepřesahující desítky mA. Pro tyto komponenty nejsou potřeba žádné velké lokální zdroje elektrické energie, protože jejich proudový odběr je nízký a poměrně konstantní a nebude tedy způsobovat velké rušení a napěťové poklesy v systému. Výjimkou je GSM modul. Na obrázku 16 je vidět, že při registraci do sítě a odesílání dat vznikají periodicky po 4 ms proudové špičky až 2 ampéry trvající po dobu zhruba 0,5 ms. Na takhle náhlé a vysoké proudové špičky nemůžou baterie reagovat a musí být tedy použit rychlý lokální zdroj elektrické energie, které tyto špičky bude vykrývat. Minimální velikost blokovacího kondenzátoru C_{gsm} lze vypočítat z vzorce 8, kde I_{imp} je velikost pulzu, t_{imp} délka pulzu a ΔU_{nap} je maximální pokles napětí. Vypočtená hodnota C_{gsm} přímo odpovídá doporučení výrobce v technické dokumentaci, který zároveň dodává, že by měl být přidán tantalový kondenzátor o velikosti 100 μF, který má nízkou hodnotu ESR (equal series resitance) pro ještě promptnější přísun energie. Z důvodu rezervy bude pro blokovací kondenzátor použita běžná hodnota kondenzátoru 1000 μF.

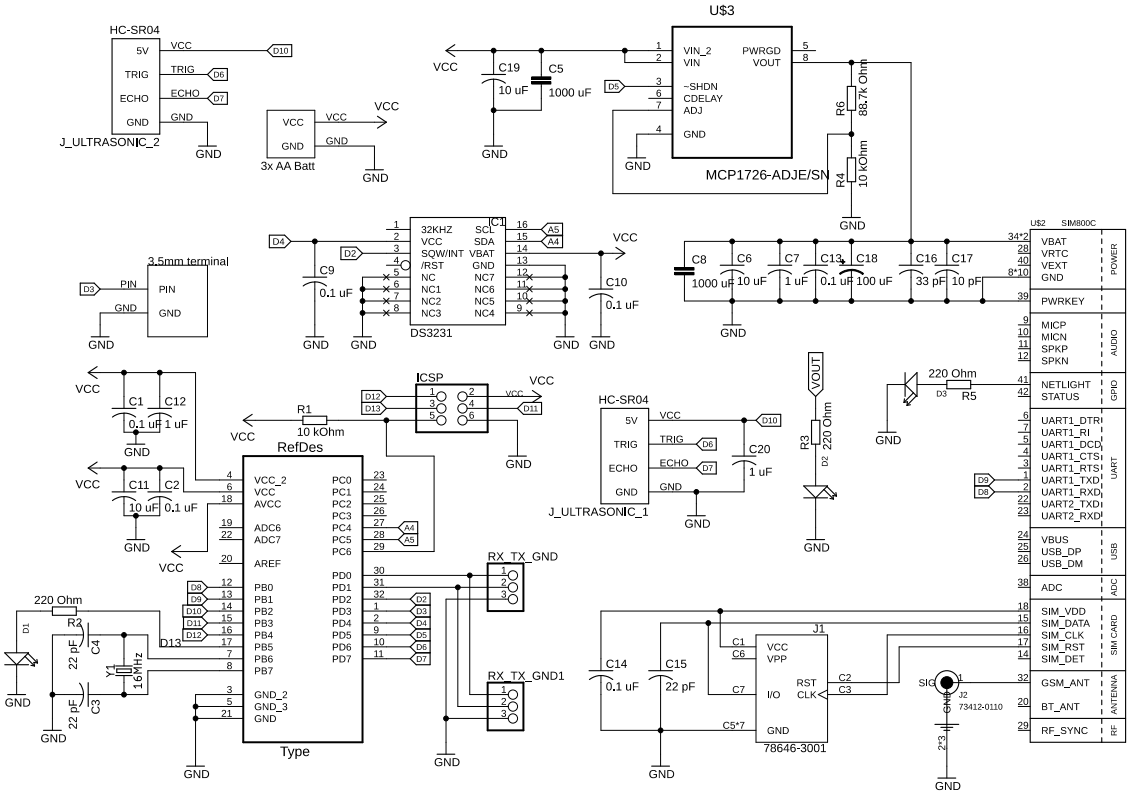
$$C_{gsm} > \frac{I_{imp} * t_{imp}}{\Delta U_{nap}} [F] \tag{8}$$



Obrázek 16 - Proudové špičky GSM modulu SIM800 [18]

4.1.3 Elektrické schéma měřící jednotky

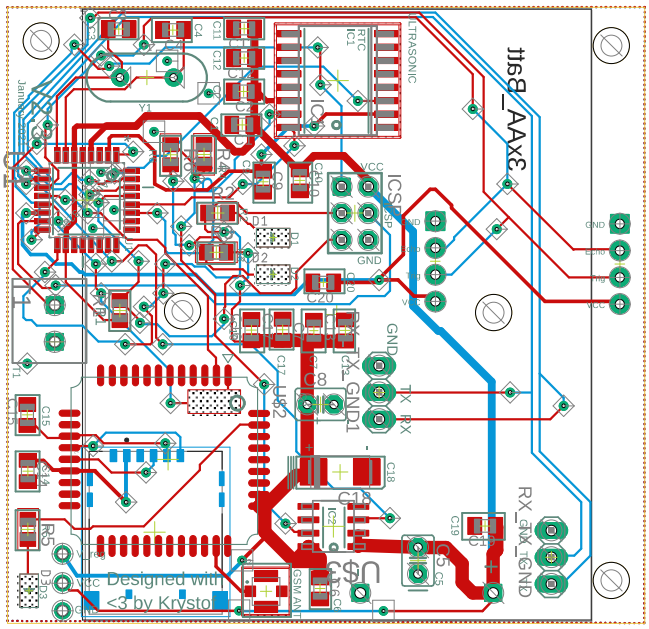
Na základě poznatků popsaných v kapitole 4.1.1 Výběr komponentů a informací z technických dokumentací jednotlivých komponent, bylo vytvořeno elektrické schéma měřící jednotky, které je na obrázku 17.



Obrázek 17 - Elektrické schéma navržené měřící jednotky

4.1.4 Deska plošných spojů

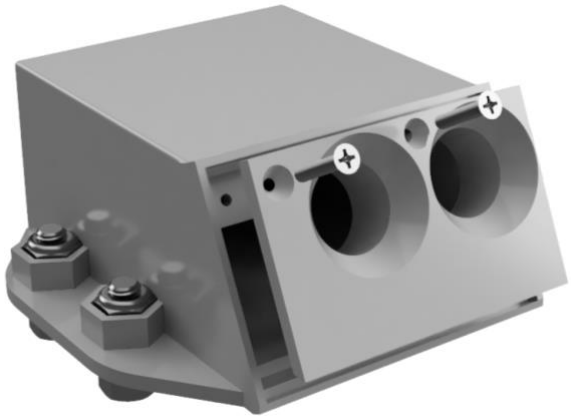
Z elektrické schématu byla poté navržena deska plošných spojů (DPS) na obrázku 18. DPS je oboustranná s rozměry 60 x 65 mm. Hlavním kritériem pro tvorbu cest a umístění komponentů byly informace z technických dokumentací. Nejdůležitější v návrhu bylo tvořit co nejširší proudové cesty pro nízký odpor a správné umístění vysokofrekvenčních prvků (konektor na GSM anténu a slot na SIM kartu) pro eliminaci rušení.



Obrázek 18 - Navržená DPS

4.1.5 Kryt měřící jednotky

Elektronika musí být ochráněna před deštěm, mechanickými vlivy a musí být nějak upevněna v odpadové nádobě. K těmto účelům bude vytvořen kryt měřící jednotky, do kterého bude elektronika vložena. Vymodelovaný 3D návrh krytu na obrázku 19 počítá s osazením na stěnu (obrázek 9 vpravo) do 1100 litrové nádoby a přichycením pomocí čtyř šroubů. Kryt je tvořen ze dvou jednolitých částí – „kapsy“, do které se zespodu vloží senzor a víka, které je upevněno dvěma šrouby. Mezi „kapsu“ a víko nejsou daná žádná těsnění, protože víko bude směřovat směrem dolů, kapsa má přesahy a je tedy předpokládáno, že dešťová voda po něm steče.



Obrázek 19 - 3D model krytu elektroniky

4.1.6 Prohlášení o shodě a EMC

Aby byl výrobek uveden na evropský trh, je potřeba, aby disponoval certifikátem o prohlášení o shodě. Tímto certifikátem výrobce prohlašuje, že zařízení je v souladu s českými a evropskými normami, které se týkají bezpečnosti, zdraví a ochrany prostředí. Pro vydání prohlášení o shodě je nejprve nutno určit,

jaké evropské direktivy se výrobku týkají a zda výrobek spadá do regulované či neregulované kategorie. Kategorie určuje, jestli výrobek potřebuje k otestování certifikované pracoviště třetí strany, případně jestli výrobce může provést testování a vydat prohlášení o shodě sám. Poté je provedeno samotné testování, vydán technický report a výrobek je označen logem CE [23].

Bylo zjištěno, že měřící jednotka spadá do neregulované sféry a že jediná platná direktiva působící na měřící jednotku, kvůli GSM modulu, je elektromagnetická kompatibilita (EMC). „Elektromagnetická kompatibilita (slučitelnost) EMC je definována jako schopnost zařízení, systému či přístroje vykazovat správnou činnost i v prostředí, v němž působí jiné zdroje elektro-magnetických signálů (přírodní či umělé), a naopak svou vlastní „elektromagnetickou činností“ nepřípustně neovlivňovat své okolí, tj. nevyzařovat signály, jež by byly rušivé pro jiná zařízení.“ [24]

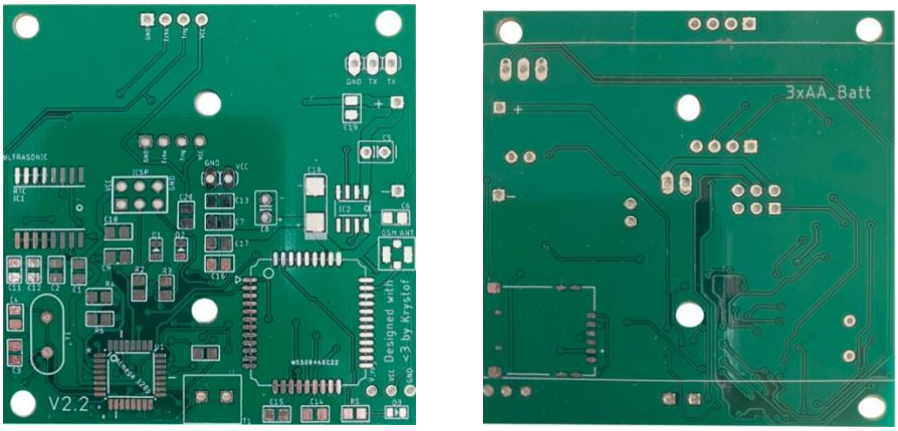
Je předpokládáno, že jediný rušivý a rušený prvek celé měřící jednotky z pohledu EMC může být GSM modul. Dále předpokládáme, že výrobek je zcela bezpečný (protože není důvod, aby nebyl – neobsahuje žádné nebezpečné materiály ani ostré hrany, napětí má do 5 voltů) jak pro člověka, tak pro své okolí a potvrzujeme, že výrobek byl navrhnut a sestaven podle běžných elektrotechnických standardů a postupů, tudíž předpokládáme, že mimo EMC splňuje všechny normy pro něj platné. Výrobce GSM modulu, společnost SimCOM Wireless Solution Co. ltd., nechala v měřící jednotce použitý modul certifikovat dle pro něj všech platných evropských norem a certifikát získala. Je tedy tvrzeno, že i celá měřící jednotka je elektromagnetický kompatibilní.

4.2 Výroba měřící jednotky

4.2.1 Hardware

Deska plošných spojů

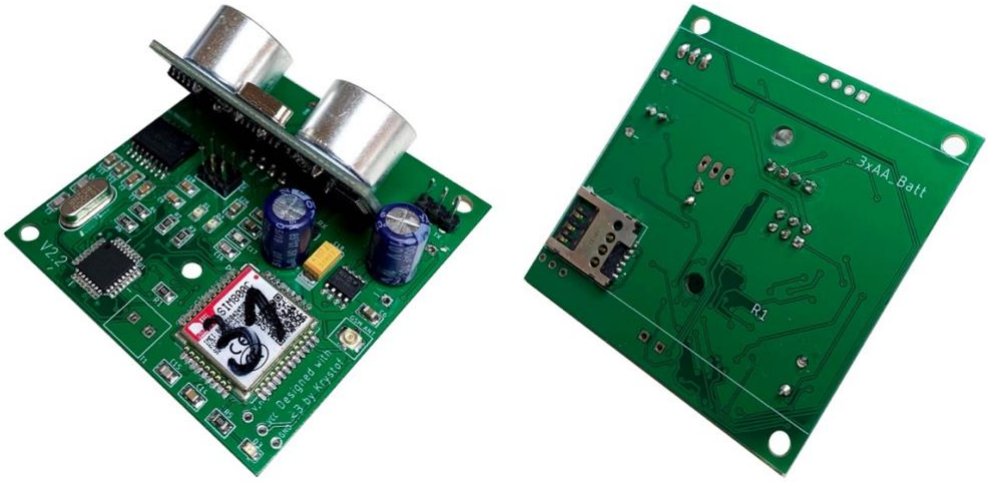
Na základě návrhu z 4.1.4 Deska plošných spojů, byly vygenerovány výrobní gerber soubory, které byly poslány do výroby. Hotová DPS je na obrázku 20 níže. Zároveň byla vyrobena šablona, přes kterou bude aplikována pájecí pasta pro pájení povrchových komponentů.



Obrázek 20 - Přední a zdaní strana vyrobéné DPS

Osazení komponenty

Z navrženého elektrického schématu z kapitoly 4.1.3 Elektrické schéma měřící jednotky byl vygenerován seznam komponentů, který byl zakoupen. Komponenty se dělí na drátové (THT – through hole) a povrchové (SMD – surface mount devices). Nejprve pomocí zakoupené šablony byla aplikována pájecí pasta, poté se osadily SMD komponenty a deska s komponenty se nechala zapéct (pájení přetavením) v pájecí peci podle teplotního profilu pájecí pasty. Následně se ručně dopájely THT komponenty. DPS osazená komponenty je na obrázku 21.



Obrázek 21 - Osazená DPS

Kryt

Kryt elektroniky byl vytisknut na 3D tiskárně. 3D tisk je vhodný na prototypování a výroby malých sérií, protože není nutná výroba formy a je tedy možné dělat rychlé iterace modelu bez velkých nákladů na výrobu. Nevýhoda 3D tisku je poměrně vysoká cena za tisk (jak z pohledu materiálu, tak provozních nákladů) a malá rychlost.

Kryt měřící jednotky byl vytisknut na 3D tiskárně Prusa MK3S se speciální širší 0,8 mm tryskou pro vysokou mechanickou odolnost. Jako materiál byl použit PETG, který disponuje vysokou tepelnou odolností, vysokou mechanickou odolností a poměrně jednoduchým nastavením tisku. Kryt měřící jednotky (víko a kapsa) váží 85 gramů a tiskne se zhruba tři hodiny. Vytisknutý kryt s vloženým senzorem je vidět na obrázku 22.

Obrázek 22 - Vyrobený kryt měřící jednotky se senzorem

4.2.2 Firmware

Firmware mikrokontroléru byl napsaný v programovacím jazyce C a C++. Hlavní důraz byl kladen na spolehlivý, bezpečný a efektivní tok programu, protože měřící jednotka má fungovat autonomně po dlouhou dobu bez jakéhokoliv zásahu. Případná softwarová chyba by mohla znamenat „zaseknutí“ programu a s tím i spojené vysoké náklady na fyzickou výměnu měřící jednotky. Program také disponuje interním záznamem provozních dat, který neustále zaznamenává provozní stavy pro případnou identifikaci provozních problémů.

Tok programu přímo kopíruje funkční vývojový diagram na obrázku 10. Pseudoalgoritmus programu vypadá následovně:

1. Inicializuj veškeré proměnné a komunikaci s PC a GSM modulem;
2. Zapni GSM modul a stáhni data ze serveru o frekvenci měření, času odesílání a aktuální čas;
3. Nastav RTC modul pro buzení a usni;
4. Spi do doby, dokud tě RTC neprobudí;
   4.1. Pokud RTC probudí pro měření
      4.1.1. Zapni ultrazvukový senzor, proveď měření zaplněnosti a provozních veličin a ulož data
      4.1.2. Nastav RTC pro probuzení, vypni ultrazvukový senzor a usni
      4.1.3. Jdi na krok 4
   4.2. Pokud RTC probudí pro odeslání
      4.2.1. Zapni ultrazvukový senzor, proveď měření zaplněnosti a provozní veličin a ulož data
      4.2.2. Připrav uložená data k odeslání
      4.2.3. Zapni GSM modul a proveď sadu příkazů pro odeslání
      4.2.4. Odešli data a stáhni si aktuální čas, frekvenci měření a času odesílání (pro případ změny)
      4.2.5. Nastav RTC pro probuzení, vypni veškeré periferie a usni
      4.2.6. Jdi na krok 4

Byly také provedeny unit testy kritických funkcí, aby se zjistilo a napravilo případné neočekávané chování při neočekávaných vstupech. Je možné, že komunikace mezi serverem a GSM modulem, případně mezi mikrokontrolérem a GSM modulem bude někdy rušená a v přenesených datech budou chyby. Pokud by chybová data byly např. časy probuzení, bylo by možné, že by se mikrokontrolér nikdy neprobudil. Pro tyto případy byly napsány unit testy, které testují předáním chybných nebo nesmyslných dat bezpečnostní funkce kritických částí programu.

4.3 Testování měřící jednotky

Testování měřících jednotek probíhalo na dvě části – laboratorní testování a testování v reálných podmínkách. Pro laboratorní testy byly postaveny tři prototypy, na kterých se odladily zjevné chyby. Poté na základě poznatků z laboratorního testování bylo postaveno dalších 50 měřících jednotek, které následně byly nasazeny do reálných podmínek – odpadových nádob. Pro oba typy testů byla vytvořena provizorní jednoduchá databáze, kam chodily data.

4.3.1 Laboratorní testy

Cílem laboratorních testů bylo zjistit technické parametry, ověřit funkci a odstranit zjištěné chyby.

Základní funkčnost

Pro prvotní určení funkčnosti měřících jednotek (MJ) byly měřící jednotky nastaveny, aby měřily každé dvě hodiny a posílaly data jednou denně v 12:00 po dobu 10 dní.

V tabulce 10 je seznam problému, jejich příčin a provedené řešení.

Tabulka 10 - Problémy při testování a jejich řešení

Problém	Příčina	Řešení
MJ je nespolehlivá a chová se „divně“	Mikrokontroléru dochází RAM paměť při seskupování dat	SW záplata - důraz na šetření RAM
Vysoká spotřeba energie během spánku	Ultrazvukový senzor necháván při spánku zapnutý	SW záplata – ultrazvukový senzor vypnut na spánek
Restartování při sepnutí regulátoru napětí a nižších teplotách	Při sepnutí regulátoru napětí dochází k okamžitému nabití všech kapacit za regulátorem, což při zvýšené IR způsobuje velký pokles napětí a restart mikrokontroléru	SW záplata – regulátor se nezapíná najednou, ale postupně pomocí PWM – tím dochází i k postupnému nabití kapacit, což eliminuje problém
Vysoký pokles napětí při větších proudech	Úzké cesty na DPS	HW iterace – rozšíření cest

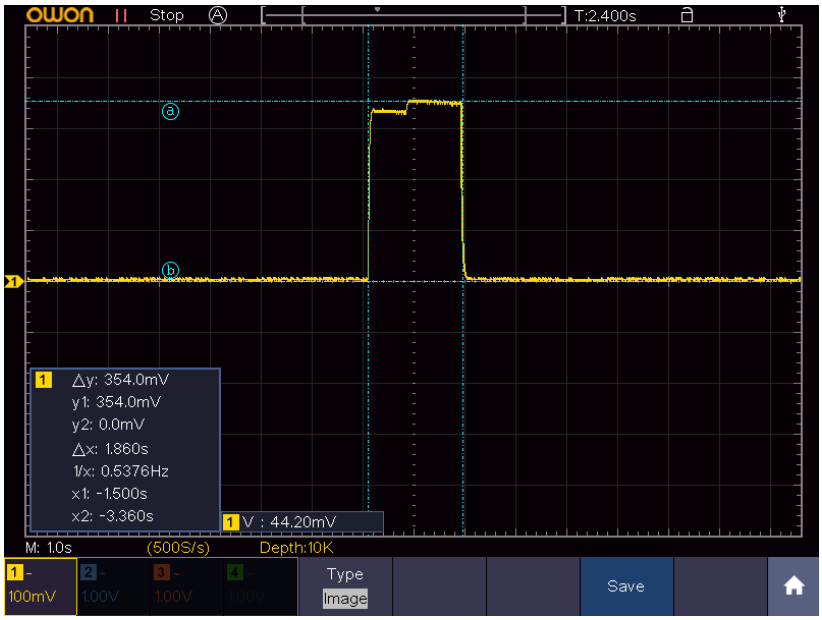
Po vyřešení problémů v tabulce výše, všechny tři měřící jednotky dokončili základní funkční test bez chyb a návrh byl označen jako funkční.

Spotřeba elektrické energie

Po základním testu funkčnosti byly proměřeny elektrické proudy jednotlivých stavů a z toho odvozena spotřeba elektrické energie při denním režimu 12 měření a jedno odeslání dat.

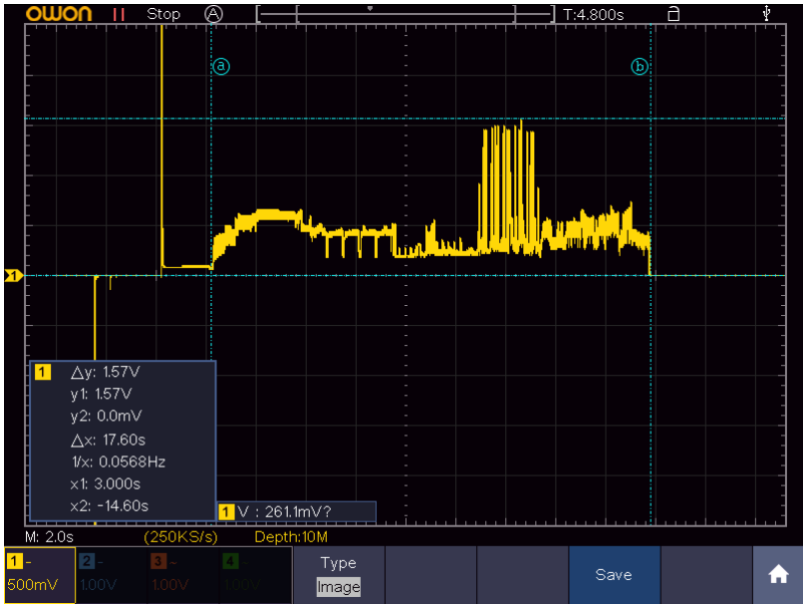
Elektrický proud během spánku je konstantní a může být tedy měřen ampérmetrem. Byla změřena hodnota **8 μA**. Tato hodnota v sobě zahrnuje ztráty v kondenzátorech, spotřebu RTC a mikrokontroléru v režimu spánku.

Pro zjištění elektrické proudu a délky stavu měření byl použit osciloskop a měření napětí přes 21 ohmový rezistor. Z ohmova zákona (napětí na rezistoru/odpor) je možné získat proud. Na obrázku 23 je vidět průběh napětí na rezistoru. Pro jednoduchost bude uvažováno, že se jedná o dokonalý obdélníkový průběh s amplitudou 0,354 voltu a délkou 1,86 vteřiny. 0,354 voltu odpovídá zhruba **17 mA** (0,354/21), což v kombinaci s délkou pulzu 1,86 vteřiny dělá spotřebu elektrické energie **0,0087 mAh** za jedno měření.



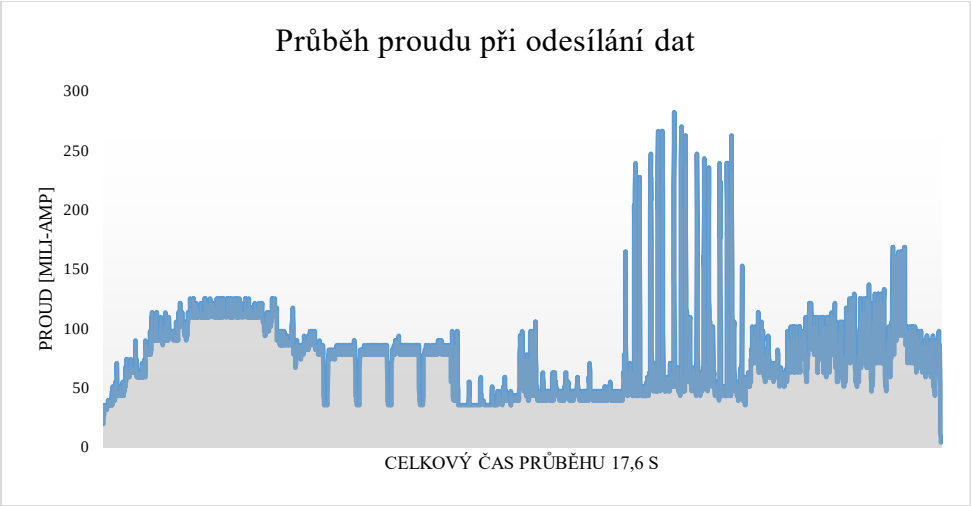
Obrázek 23 - Měření spotřeby stavu měření - průběh napětí na 21 ohm rezistoru

Průběh proudu při odesílání dat na obrázku 24 byl měřen osciloskopem přes 5,1 ohmový odpor. Celé odesílání dat a příjem informací ze serveru trvalo 17,6 vteřin s největší proudovou špičkou **307 mA** (1.57 V/5.1 ohm).



Obrázek 24 - Průběh proudu měřící jednotky při odesílání dat

K získání elektrické energie byly napěťové hodnoty z průběhu z obrázku 24 přepočítány na proud (obrázek 25), poté integrovány. Pro připojení, odeslání a příjmu dat, vyšla spotřeba elektrické energie na **0,37 mAh**.



Obrázek 25 - Průběh elektrického proudu měřící jednotky při odesílání dat

Po dosazení naměřených hodnot do vzorce 10, vyšlo, že měřící jednotka má denní spotřebu **0,612 mAh** a s kapacitou baterie 2500 mAh by bez dalších vlivů teoreticky měla vydržet fungovat 11 let na baterii. Taková výdrž je ale nereálná, protože nerespektuje vnitřní odpor baterií.

Pro přesnější odhad, byl proveden test spotřeby, ve kterém dvě měřící jednotky byly nastaveny, aby v cyklu měřily každých 5 vteřin a odesílaly data jednou za minutu do té doby, než se vybije baterie. Jeden cyklus se tedy skládá z jednoho odeslání dat a 12 měření zaplněnosti – jeden cyklus má tedy reprezentovat jeden den. Jedna měřící jednotka měřila v pokojové teplotě 22 °C, druhá v mrazáku nastaveném na – 20 °C. Měřící jednotka umístěná v pokojové teplotě vydržela 2990 cyklů a přestala posílat při dosažení napětí baterie 3,7 voltu. Druhá měřící jednotka umístěná v mrazáku vydržela posílat do napětí baterie 4 volty. Při dosažení 4 voltů, první měřící jednotka absolvovala 1400 cyklů. Pokud vezmeme nejhorší možný scénář – baterie měřící jednotky je 4 volty a zrovna přišla zima s teplotami -20 °C, měřící jednotka by v ten moment přestala fungovat po 1400 dnech – tj. necelých čtyřech letech provozu s měřením 12krát denně a odesíláním jednou denně. I tento výpočet ale mnoho zanedbává a je spíše přesnějším odhadem. Pro přesnou informaci výdrže by musel být proveden dlouhodobý test spotřeby energie.

Krytí

Měřící jednotka byla otestována z pohledu stupně ochrany krytem, normy IEC 529 (EN 60529, případně ČSN EN 60529). Požadavky na kryt měřící jednotku byl, aby ochránil elektroniku před kapající vodou (alespoň IPx1) a vniknutím drobného cizího předmětu (alespoň IP3x). Testování ukázalo, že kryt elektronice zajišťuje stupeň krytí IP43 – stupeň krytí chrání před vniknutí nástrojem a drátem s průměrem větším než 1 mm a vodní tříšti stříkající v úhlu 60° vertikálně, v množství 10 litrů za minutu po dobu nejméně 5 minut.

4.3.2 Test v reálném prostředí – projekt Starý Plzenec

Pro dlouhodobější test většího rozsahu v reálných podmínkách byla domluvena spolupráce mezi městem Starý Plzenec, poradenskou společností Odpadová poradenská s.r.o. a mnou. Hlavním cílem projektu bylo zjistit dynamiku zaplňování a vytíženost vytipovaných 19 sběrných stanovišť na plasty a papír

v celkovém součtu 32 nádob a stanovit případná doporučení úpravy svozových cyklů, případně počtu nádob na stanovištích. Vedlejší cíle projektu byly zjištění základní funkčnosti měřících jednotek v reálném prostředí, zjištění spolehlivosti odesílání a celkové vyhodnocení vhodnosti technologie. Projekt měl trvání čtyři měsíce – měřící jednotky byly dva měsíce v nádobách na papír, dva měsíce v nádobách na plast. Starý Plzenec byl klient, který si studii objednal, já jsem dodával surová data a poradenská společnost data vyhodnotila a zformulovala doporučení pro optimalizaci sběru a svozu. V této práci se nebude zabýváno konkrétními výsledky a vyhodnocenými doporučeními tohoto projektu, protože se jedná o informace v majetku města Starý Plzenec. Budou pouze popsány technické poznatky – tj. vyhodnoceny vedlejší cíle projektu.

Jak je vidět na obrázku 26, měřící jednotky byly nainstalovány do 1100 litrových sběrných nádob na separovaný odpad na vnitřní boční stranu v horní části.

Obrázek 26 - Příklad umístěných měřících jednotek z projektu Starý Plzenec

4.3.2.1 Vyhodnocení vedlejších cílů projektu

Základní funkčnost v reálném prostředí

Nainstalované měřící jednotky měřily a posílaly data po celou dobu projektu. Z provozních dat, konkrétně napětí, byla nepřímo potvrzena dlouhá životnost baterie, protože za celou čtyřměsíční dobu projektu napětí průměrně kleslo o pouze 0,1 voltu.

V první fázi projektu byly jednotky naprogramovány, aby měřily pouze jednou denně před odesláním dat. To se ukázalo jako nedostatečné, protože kvůli rychlé dynamice zaplňování plastů a papíru (odpadová nádoba se zaplnila do druhého dne) v některých případech nebylo vidět, jestli nádoba byla vyvezena či ne. Změna na frekventovanější měření – každé dvě hodiny – problém vyřešila.

Na základě těchto poznatků byl návrh hardwaru a firmwaru vyhodnocený jako správně zvolený a funkční.

Spolehlivost měřících jednotek

Všechny, kromě jedné, měřící jednotky měly 100 % spolehlivost odesílání. Z průběžně posílaných provozních dat z nespolehlivé jednotky bylo patrné, že k nestabilitě dochází z důvodu slabého signálu. Hardwarové přezkoumání ukázalo, že konektor GSM antény byl špatně napájený.

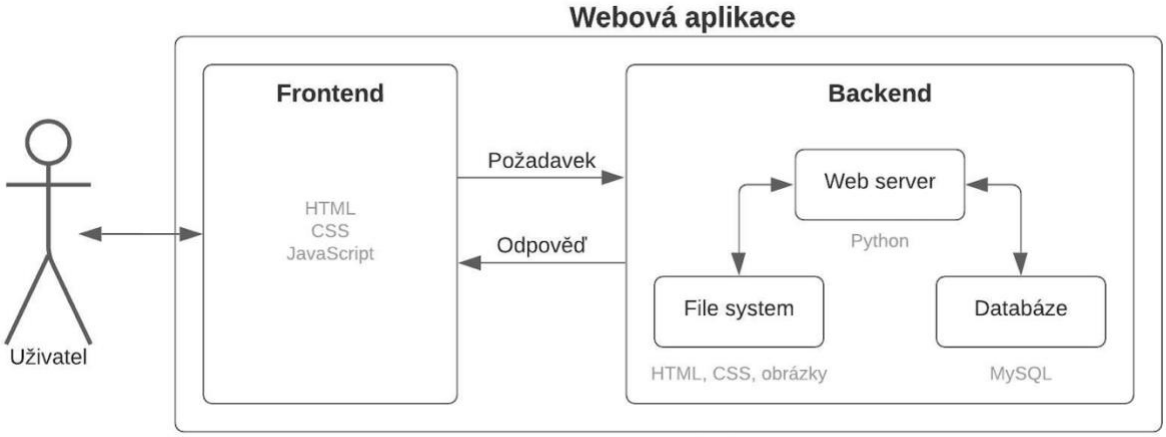
Vhodnost použité technologie

Ultrazvukové měření zaplnění pomocí měřících jednotek a posílání dat na server se ukázalo jako vhodně zvolené a funkční. Mírným nedostatkem technologie měřením ultrazvuku je u stlačitelných odpadů občasné hlášení zaplněnosti, i když odpadová nádoba zaplněná není. Typický příklad je na obrázku 27. Pod senzorem je nestlačená krabice, která dává falešný dojem zaplněnosti. Bohužel neexistuje žádná rozumně použitelná senzorika, která by takové situace mohla vyřešit. I člověk potřebuje použít fyzickou sílu, aby zjistil, jestli je odpadová nádoba opravdu plná.

Obrázek 27 - Příklad situace, kdy dochází k falešnému oznámení zaplnění

5 Realizace webové aplikace

Webová aplikace bude sloužit k interakci mezi uživatelem a daty z databáze v přehledné a uživatelsky přívětivé formě. Frontend aplikace bude psán v HTML, CSS a JavaScriptu. Backend se bude skládat z MySQL databáze s kterou bude komunikováno pomocí SQL příkazů a webového serveru. Logika aplikace bude sestavena v programovacím jazyce Python 3.8. Schéma aplikace je vidět na obrázku 28. Celá webová aplikace bude hostována na serverech Amazonu, pomocí služby AWS – Amazon web services.



Obrázek 28 - Schéma webové aplikace

5.1 Funkcionality aplikace

Základní obecný požadavek na aplikaci je přehledná a srozumitelná interpretace dat z měřících jednotek. Pro návrh databáze a páteřní kostry funkcionalit bude použita technika user stories. User stories (uživatelské příběhy) jsou v softwarovém vývoji popisy unikátních situací, v kterých se uživatel může nacházet a na základě kterých se poté staví funkcionality aplikace. Tyto příběhy neříkají jenom co uživatel chce dělat, ale také proč. Typická kostra user story je: *Jako uživatel, chci funkcionalitu, abych dostal něco.*

5.1.1 User stories aplikace

- Jako uživatel se chci **přihlásit** do aplikace, abych viděl data ze svých měřících jednotek.
- Jako uživatel chci vidět seřazený **tabulkový přehled** všech svých osazených odpadových nádob, abych věděl, jak jsou zaplněné.
- Jako uživatel chci **grafický vývoj** zaplněnosti jednotlivých odpadových nádob, abych viděl dynamiku zaplňování.
- Jako uživatel chci vědět **celkový objem odpadu** v osazených nádobách, abych věděl, kolik aut budu potřebovat k vývozu.

- Jako uživatel chci být **barevně upozorněn** na přeplněné odpadové nádoby, abych mohl poslat úklidovou četu na specifický vývoz.
- Jako uživatel chci mít možnost **exportovat data**, abych je mohl použít k vlastnímu dalšímu zpracování.
- Jako uživatel chci vidět **tabulkový přehled** zaplnění každé nádoby, abych mohl lépe zanalyzovat zaplnění.
- Jako uživatel chci **zvolit frekvenci měření a odesílání** dat měřících jednotek, abych nastavil vhodný interval pro dané stanoviště.

Z uživatelských příběhů vyplynulo, že základních funkcionality aplikace by měly být následující:

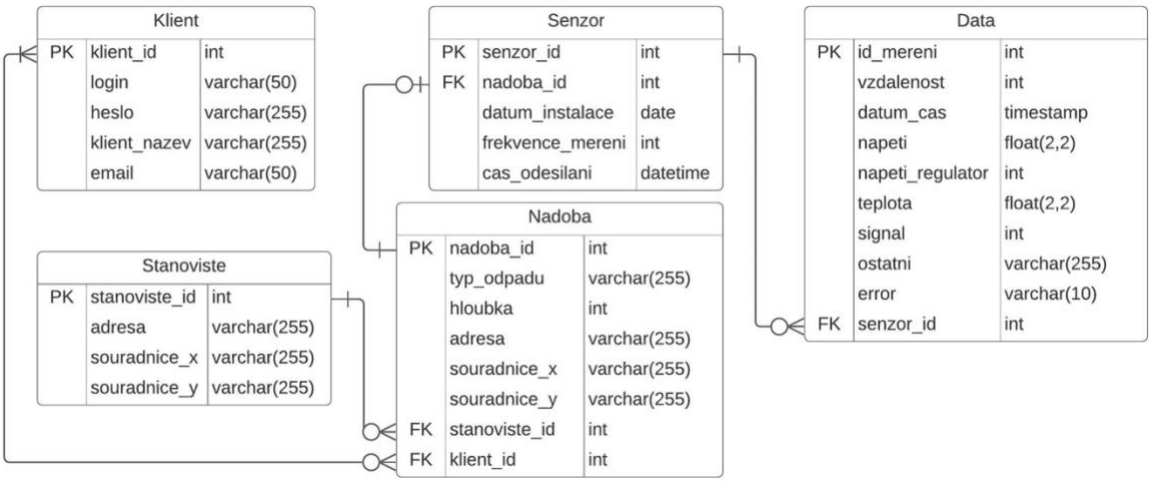
- Přihlášení, odhlášení
- Tabulkový přehled všech nádob s možností seřazení dle aktuálního naplnění a barevném označení přeplněných nádob
- Detail odpadové nádoby s historickým tabulkovým a grafickým přehledem
- Výpočet celkového objemu odpadu
- Možnost nastavení časů měření a odesílání měřících jednotek

5.2 Databáze

Pro strukturalizované uložení dat z měřících jednotek a jejich prezentaci byla zvolena MySQL databáze.

5.2.1 Databázový model

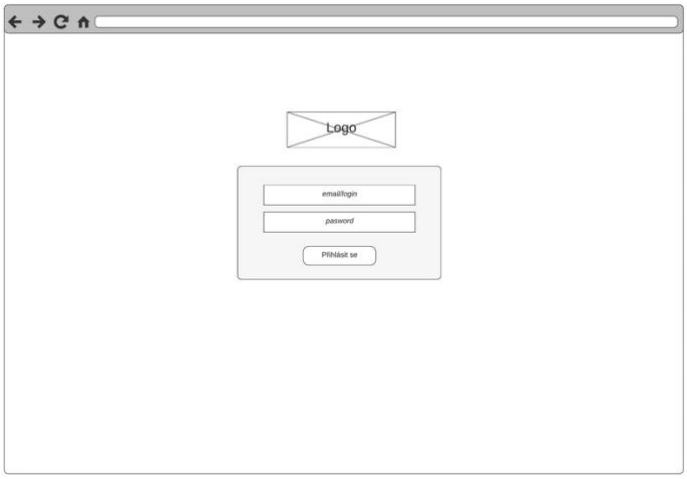
Schématický model databáze je vidět na obrázku 29. Databáze bude tvořena z pěti tabulek *klient, senzor, data, stanoviště* a *nádoba*. Data z měřících jednotek budou ukládány do tabulky *data*. S každým odesláním dat bude poslán i unikátní klíč *senzor_id,* díky kterému se příchozí data budou moci spárovat s informacemi z tabulek *senzor, nádoba, klient* a *stanoviště*. Pomocí tohoto spárování bude určeno, k jakému senzoru data patří, a tedy v jaké nádobě a na jakém stanovišti je senzor umístěn a jak je nádoba naplněná.



Obrázek 29 - Schématický model databáze

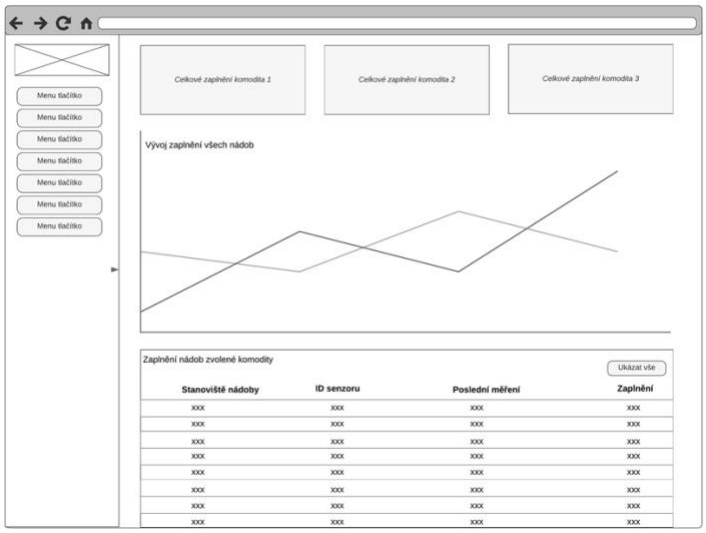
5.3 Design aplikace

Prvotní náčrt hlavních částí aplikace budou použity „wireframes“ – tj. schématické grafické zobrazení rozložení webové aplikace. Návrh vzhledu byl koncipován tak, aby byl co nejjednodušší a zároveň splňoval požadované funkce zmíněné v kapitole 5.1 Funkcionality aplikace. Na obrázku 30 je přihlašovací obrazovka do aplikace.



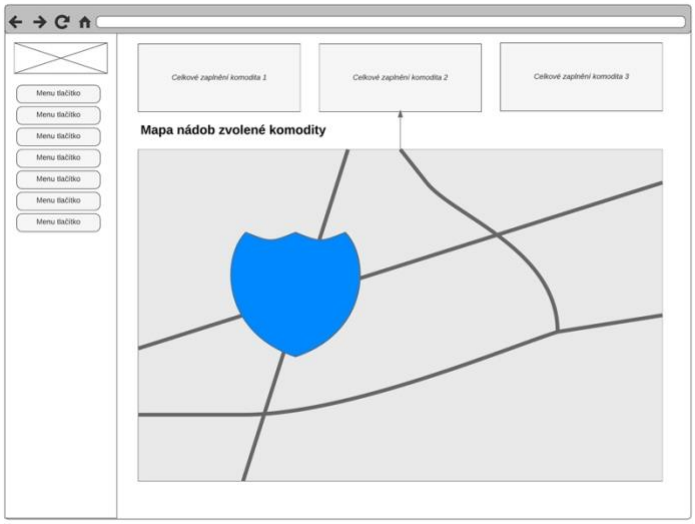
Obrázek 30 – Wireframe přihlašovací obrazovka aplikace

Domovská stránka aplikace je na obrázku 31. Zde uživatel vidí přehled zaplnění v grafické a tabulové formě všech svých nádob podle zvoleného typu odpadu. Grafická forma ukáže kumulativní procentuální zaplnění všech nádob daného typu, v tabulové formě jsou nádoby seřazeny podle zaplnění.



Obrázek 31 – Wireframe domovská stránka aplikace

Aplikace bude také obsahovat sekci mapy (obrázek 32). Na této mapě bude, po zvolení typu odpadu, přehled všech sledovaných odpadových nádob, kde po kliknutí na danou nádobu se zobrazí poslední měřené informace.



Obrázek 32 – Wireframe mapa v aplikaci

Rozložení sekce detailu konkrétní odpadové nádoby je na obrázku 33. Po kliknutí na konkrétní odpadovou nádobu se zobrazí grafický historický vývoj zaplnění, ukáže se lokace na mapě a zároveň se vypíšou historická data.

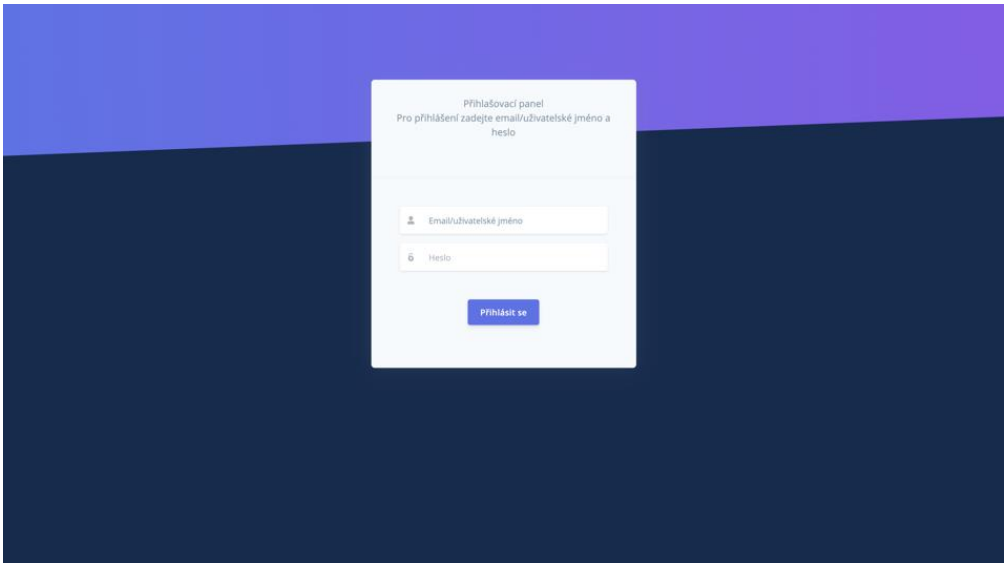
Obrázek 33 – Wireframe detail nádoby v aplikaci

5.4 Frontend a backend

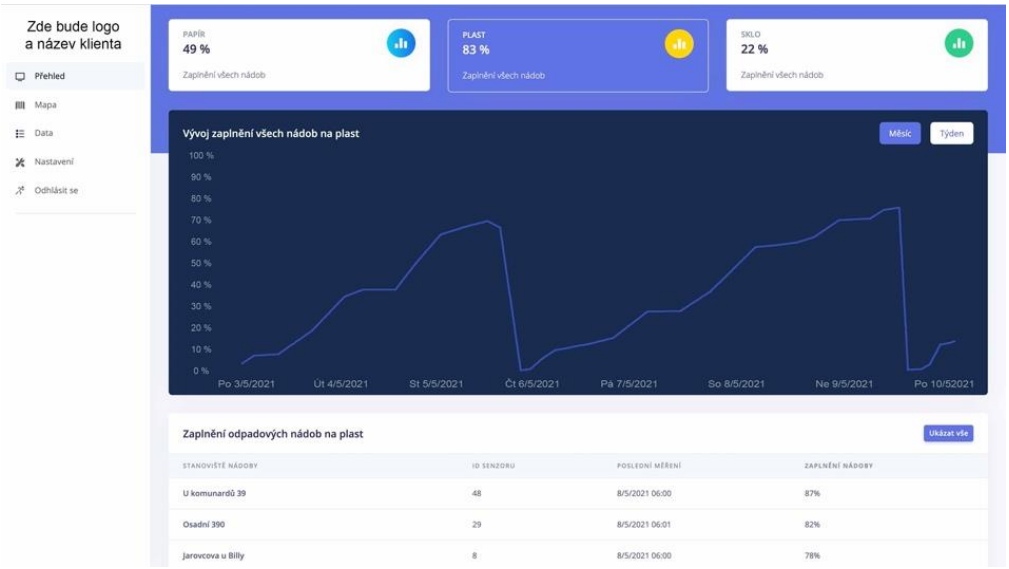
Frontend aplikace (to, co uživatel vidí) bude napsán v jazycích HTML, CSS a Javascript a bude se odvíjet od designu wireframes. Pro usnadnění stylování aplikace bude použita knihovna stylů Bootstrap 5, pro vizualizaci dat javascriptová knihovna Chart JS. Backend a tedy i logika aplikace bude napsána v programovacím jazyce Python s použitím mikroframeworku Flask. Webový server bude přímo připojen do databáze bez API mezivrstvy. Pokud by data z databáze měly být sdílena s třetí stranou (svozová firma, majitel nádob,…) mimo webovou aplikaci, bylo by nutné z důvodů bezpečnosti a uživatelské přívětivosti vytvořit k databázi API. Webový server a databáze bude hostován na serverech Amazonu pomocí služby Amazon Web Services (AWS) EC2. Využitím této služby odpadnou starosti ohledně hardwaru a při případném růstu a potřeby většího výkonu a místa pro aplikaci, je možné pronajaté servery jednoduše rozšířit bez jakýchkoliv investičních nákladů.

5.5 Výsledná aplikace

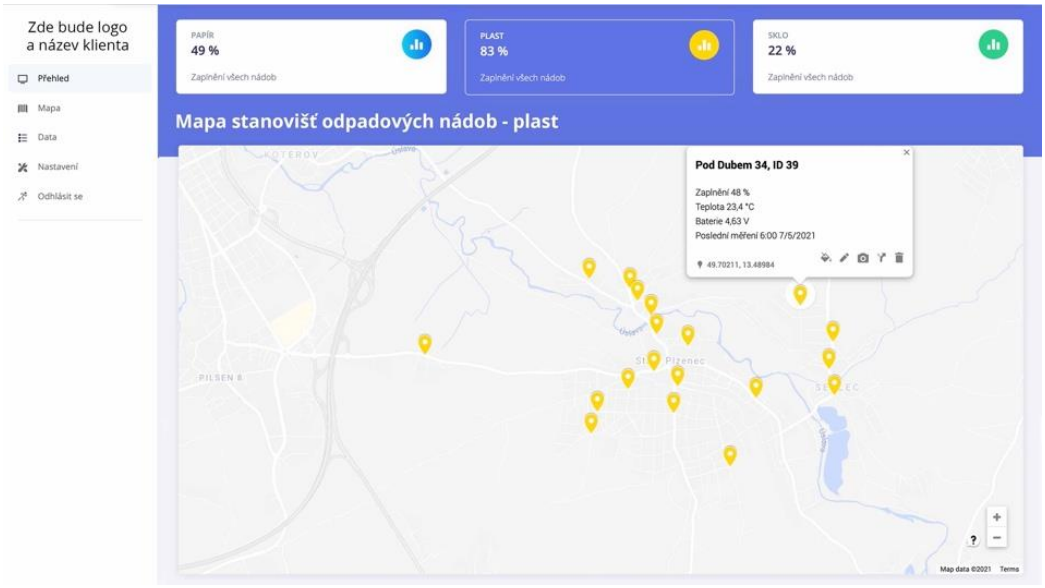
Hotová aplikace je vidět obrázcích 34 až 37. Vzhled aplikace vychází z navržených wireframes.



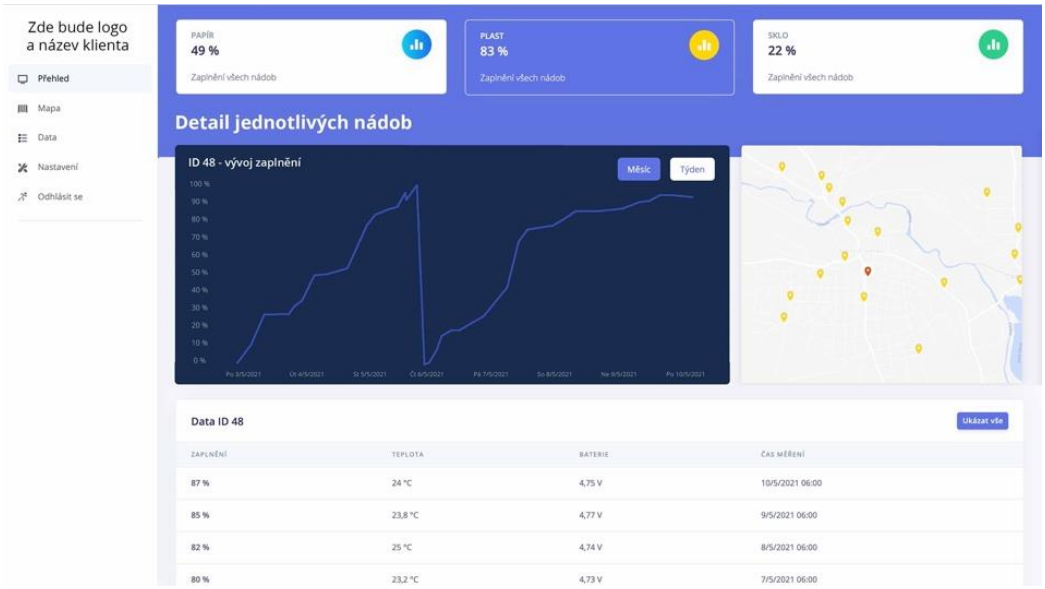
Obrázek 34 - Přihlašovací stránka aplikace



Obrázek 35 - Domovská stránka aplikace



Obrázek 36 - Mapa aplikace



Obrázek 37 - Detail nádoby v aplikaci

6 Případová studie

Pro ověření teoretických poznatků a vytyčení případných ekonomických přínosů navrženého řešení bude zpracována případová studie monitoringu odpadových nádob a optimalizace svozu spočívající v reálné aplikaci navrženého systému, získání reálných dat zaplňování, vytvoření simulace zaplňování a optimalizovaného svozu v diskrétním čase a ekonomickém zhodnocení navrženého projektu.

Partnerem této studie je město Chodov, které projevilo zájem o zjištění efektivnosti nynějšího stavu svozu separovaných nádob na sklo a ověření si přesnosti jejich zavedeného systému rozšířené evidence odpadů. Tato studie se bude zabývat jak ekonomickým pohledem města, tak i svozové firmy. Z důvodů práce se soukromými daty města Chodov a svozové společnosti, budou v této studii po dohodě všech stran použité informace a naměřená data upraveny tak, aby se zamezilo porušení soukromého vlastnictví dat, ale zároveň aby neutrpěly na žádné ztrátě své informační hodnoty.

Informace

Město Chodov má 13 500 obyvatel s průměrnou hustotou obyvatelstva 950 obyvatel na km^2. Odpad sváží soukromá firma (Chodovské technické služby s.r.o.) a odpadové nádoby jsou majetkem města. Tento scénář a tyto údaje jsou v prostředí svozu odpadů měst zcela běžné a dá se tedy předpokládat možnost zobecnění aplikace na ostatní města v České republice.

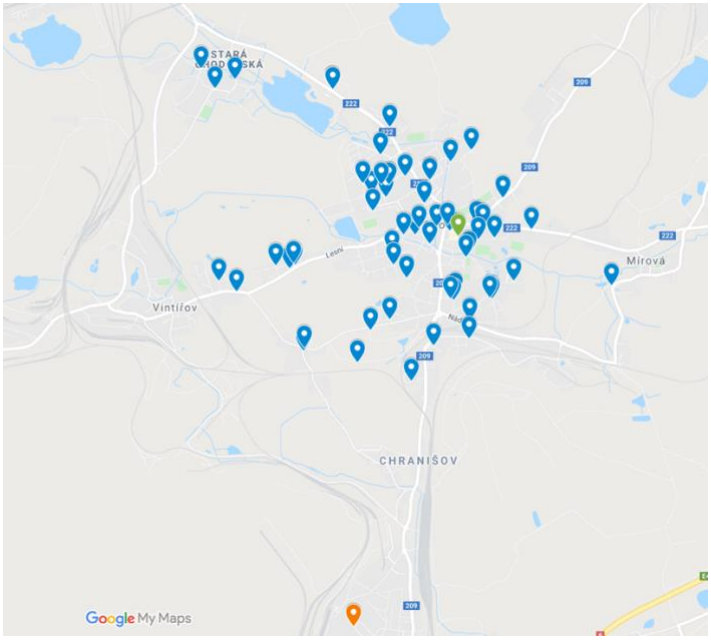
Chodov vlastní 60 ks nádob na tříděné sklo s využitelným objemem 1100 litrů. V současné situaci jsou všechny nádoby vyváženy jednou za čtyři týdny (28 dní) po fixní svozové trase s tím, že cena svozu se odvíjí od počtu vyvezených nádob. V této případové studii se nebude zabýváno možnými omezeními optimalizace smluvními kontrakty – bude předpokládáno, že celková roční cena svozu je přímo úměrná počtu vyvezených nádob

Získání dat

Bylo osazeno 60 měřících jednotek do všech tříděných nádob na sklo města Chodov. Měření probíhalo každé dvě hodiny s odesíláním dat jednou za den. Měřilo se po dobu tří měsíců tak, aby se zachytily tři svozové cykly. Ukázka osazených nádob je na obrázku 38.

Obrázek 38 - Příklad instalace v městě Chodov

Rozmístění odpadových nádob je vidět na obrázku 39. Modře jsou označeny měřené odpadové nádoby, zeleně depo svozových aut technických služeb a oranžově je třídící linka, kam se odváží odpad. Každá odpadová nádoba má své unikátní ID, které je spojeno s unikátním ID senzoru a GPS souřadnicemi.



Obrázek 39 - Rozmístění měřených nádob, Chodov

Naměřené hodnoty

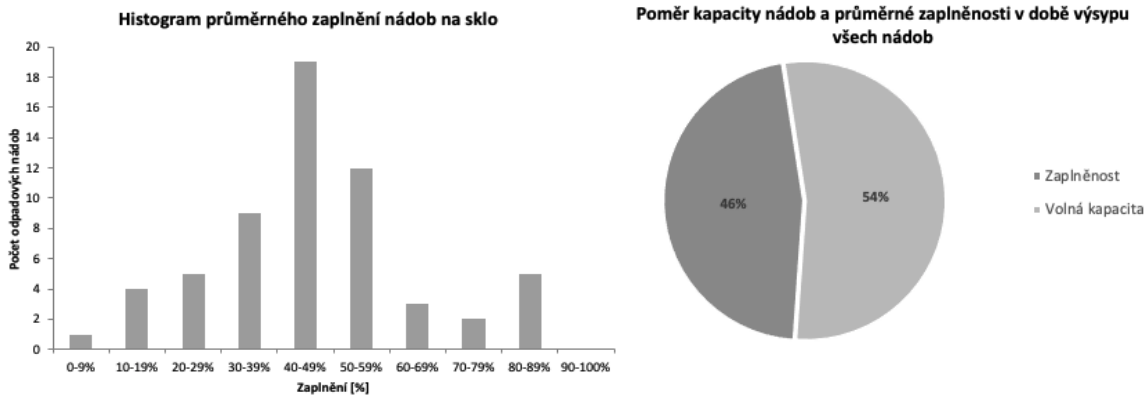
Naměřené průběžné hodnoty byly shrnuty do tabulky 11. V této tabulce jsou vidět hodnoty zaplnění jednotlivých odpadových nádob při výsypu 1, výsypu 2 a výsypu 3. Výsypová frekvence byla 28 dní. Na základě hodnot zaplnění jednotlivých nádob a výsypů byl spočítán rozptyl a průměr zaplnění každé nádoby při výsypu. Z průměru zaplnění se vypočítala relativní a absolutní denní dynamika zaplňování. Informace o dynamice zaplňování s rozptylem se použijí jako podklady pro simulaci zaplňování a jednotlivých variant svozů.

Tabulka 11 - Naměřené hodnoty, případová studie Chodov

ID senzoru	Zaplnění vysyp 1	Zaplnění vysyp 2	Zaplnění vysyp 3	Průměrné zaplnění při výsypu	Rozptyl zaplnění [%]	Dynamika zaplňování [%/den]	Dynamika zaplňování [m^3/den]	Rozptyl denního zaplňování [m^3/den]
50	38%	54%	44%	45%	0,44%	1,62%	0,018071429	0,00018822
52	31%	51%	43%	42%	0,68%	1,49%	0,016085317	0,00029515
53	61%	29%	49%	47%	1,79%	1,66%	0,017765873	0,00077352
54	37%	54%	40%	44%	0,54%	1,56%	0,017940476	0,00023464
55	80%	57%	67%	68%	0,85%	2,43%	0,026823413	0,00036738
56	56%	37%	50%	48%	0,62%	1,70%	0,018224206	0,00026932
57	50%	46%	55%	50%	0,13%	1,79%	0,018769841	0,00005616
58	42%	46%	52%	47%	0,17%	1,66%	0,017242063	0,00007430
59	30%	12%	14%	19%	0,65%	0,67%	0,008250000	0,00028041
60	6%	9%	15%	10%	0,16%	0,36%	0,002924603	0,00006843
61	34%	38%	41%	38%	0,08%	1,35%	0,014142857	0,00003553
62	39%	28%	36%	34%	0,21%	1,23%	0,013117063	0,00009231
63	55%	40%	48%	48%	0,37%	1,70%	0,018617063	0,00015825
64	71%	88%	79%	79%	0,50%	2,83%	0,031183036	0,00021432
65	69%	90%	77%	79%	0,75%	2,81%	0,031253968	0,00032292
66	50%	35%	50%	45%	0,50%	1,61%	0,016740079	0,00021395
67	46%	53%	52%	50%	0,11%	1,80%	0,019402778	0,00004621

68	65%	35%	58%	53%	1,64%	1,88%	0,019642857	0,00070967
69	89%	82%	92%	88%	0,19%	3,13%	0,03356746	0,00008124
70	41%	46%	53%	47%	0,21%	1,67%	0,017176587	0,00009273
71	36%	28%	40%	35%	0,24%	1,23%	0,012505952	0,00010379
72	52%	59%	57%	56%	0,08%	2,00%	0,021825397	0,00003514
73	25%	20%	16%	20%	0,12%	0,73%	0,008773810	0,00005026
74	71%	64%	65%	67%	0,11%	2,38%	0,026583333	0,00004738
75	50%	47%	58%	52%	0,23%	1,85%	0,018988095	0,00010115
76	42%	15%	32%	30%	1,27%	1,06%	0,011261905	0,00054674
77	13%	21%	12%	16%	0,15%	0,56%	0,006765873	0,00006512
78	59%	59%	59%	59%	0,00%	2,11%	0,023156746	0,00000001
79	41%	35%	38%	38%	0,06%	1,36%	0,014906746	0,00002498
80	54%	62%	61%	59%	0,13%	2,11%	0,022785714	0,00005474
81	41%	29%	34%	35%	0,25%	1,24%	0,013793651	0,00010855
101	36%	58%	41%	45%	0,93%	1,61%	0,018472143	0,00040244
102	36%	55%	38%	43%	0,69%	1,53%	0,017878710	0,00029797
103	52%	31%	45%	43%	0,76%	1,52%	0,016176548	0,00032917
104	44%	57%	50%	50%	0,31%	1,80%	0,019823571	0,00013243
105	80%	47%	72%	66%	2,03%	2,36%	0,024964325	0,00087512
106	66%	30%	46%	48%	2,17%	1,70%	0,018997698	0,00093911
107	57%	46%	50%	51%	0,20%	1,83%	0,02022996	0,00008793
108	36%	40%	37%	37%	0,04%	1,34%	0,014836468	0,00001560
109	33%	12%	19%	21%	0,81%	0,76%	0,008803929	0,00035030
110	6%	10%	10%	8%	0,04%	0,30%	0,003061230	0,00001776
111	40%	40%	44%	41%	0,04%	1,48%	0,015710357	0,00001600
112	31%	33%	28%	31%	0,04%	1,10%	0,012583651	0,00001830
113	59%	46%	45%	50%	0,39%	1,78%	0,020627401	0,00016888
114	82%	84%	78%	81%	0,06%	2,90%	0,032542321	0,00002569
115	72%	98%	83%	84%	1,14%	3,01%	0,034625556	0,00049466
116	41%	40%	43%	41%	0,02%	1,48%	0,015858115	0,00001056
117	48%	56%	58%	54%	0,20%	1,92%	0,020387103	0,00008748
118	62%	35%	42%	46%	1,27%	1,65%	0,019073214	0,00054903
119	89%	75%	82%	82%	0,31%	2,94%	0,032300714	0,00013243
120	39%	55%	41%	45%	0,50%	1,62%	0,018576687	0,00021500
121	32%	26%	27%	28%	0,07%	1,00%	0,011295298	0,00002963
122	59%	58%	56%	58%	0,01%	2,06%	0,022924306	0,00000466
123	21%	21%	21%	21%	0,00%	0,75%	0,008301071	0,00000031
124	83%	76%	82%	80%	0,11%	2,86%	0,031088095	0,00004626
125	48%	39%	42%	43%	0,16%	1,54%	0,017033631	0,00006785
126	49%	16%	39%	35%	1,90%	1,23%	0,012798631	0,00082161
127	14%	17%	13%	15%	0,04%	0,52%	0,006102599	0,00001683
128	56%	52%	55%	54%	0,03%	1,94%	0,021187659	0,00001263
129	47%	34%	38%	40%	0,34%	1,41%	0,015916825	0,00014642

Shrnutím tabulky 11 do grafické podoby (obrázek 40) ve formě histogramu a poměru zaplnění k volné kapacitě je přehledně a jasně vidět, že většina (38 z 60) odpadových nádob se vyváží méně než z poloviny zaplněná a že celková průměrná efektivita využití nádob je pouze 46 % (obrázek 40 vpravo). Tyto údaje napovídají, že by v tomto případě měl být prostor na optimalizaci svozu a nákladů.



Obrázek 40 - Grafické znázornění dat z Chodova

6.1 Simulace produkce a svozu odpadu

Pro zjištění optimální ekonomické varianty svozu odpadu, byla vytvořena simulace v programovacím jazyce Python. Simulace se skládá z dvou základních částí – simulace zaplňování odpadových nádob a svoz odpadu. V simulaci je vytvořeno prostředí s časem ubíhajícím diskrétně po dnech, v kterém jsou odpadové nádoby kontinuálně zaplňovány na základě dat z tabulky 11 a svozové auto, které v nadefinovaný čas, záležící na variantě, po vypočítaných trasách nádoby objíždí a vyprazdňuje je.

Výpočet tras

Pro výpočet tras mezi odpadovými nádobami byla vytvořena matice vzdáleností a matice časů, která reprezentuje vzdálenosti, resp. časy z každého bodu do každého bodu – tj. z každé odpadové nádoby do každé odpadové nádoby, depa a třídící linky. Názorná matice vzdáleností *A* je na obrázku 41, kde vzdálenost d_{ij} představuje vzdálenost z bodu a_i do bodu a_j. Tato matice byla vytvořena pro časy a pro vzdálenosti pomocí Google maps matrix API, které umožňuje získání aktuální dynamických vzdálenostních a časových dat mezi dvěma body zohledňující omezení a hustotu dopravy v daném čase. Matice jsou vytvořeny vždy při začátku simulace pomocí modulu *get_distance_matrix* (příloha 7).

$$A = \begin{array}{c} \\ a_i \\ a_{i+1} \\ a_{i+2} \\ \\ a_{i+n} \end{array} \begin{array}{c} \begin{array}{ccccc} a_j & a_{j+1} & a_{j+2} & & a_{j+n} \end{array} \\ \begin{bmatrix} 0 & d_{12} & d_{13} & \dots & d_{1n} \\ d_{21} & 0 & d_{23} & \dots & d_{2n} \\ d_{31} & d_{32} & 0 & \dots & d_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & d_{n3} & \dots & 0 \end{bmatrix} \end{array}$$

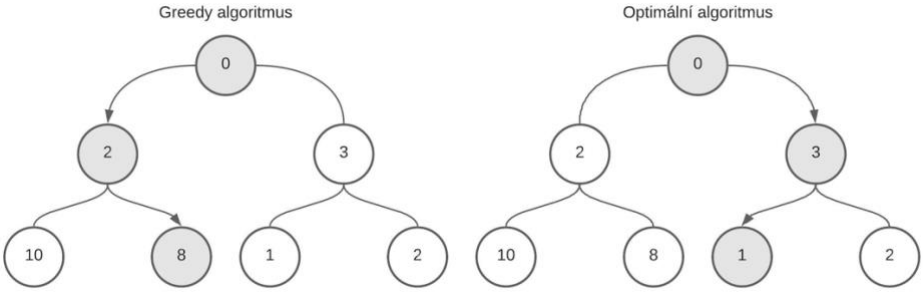
Obrázek 41 - Názorná matice vzdáleností

Získáním matice vzdáleností je možné vypočítat nejkratší možnou trasu mezi body. Tomuto problému se říká „problém obchodního cestujícího“ (travelling salesman problém – TSP). Tento problém je laicky definován jako najití nejkratší možné trasy procházejícími všemi body grafu (mapy) a vracející se zpět do původního bodu. V této aplikaci bude použit upravený TSP, který nepočítá s návratem do původního bodu (depa), ale do konečného bodu, což bude třídící linka odpadů, kde bude vůz vysypán. Poté bude manuálně přidána vzdálenost mezi třídící linkou a depem jako návratová cesta. Najití nejkratší cesty pomocí naivního (brute force) algoritmu je velmi časově náročné. Počet kombinací $n_{kombinací}$ lze vypočítat ze vzorce 9, kde *n* je počet bodů v grafu. V případě města Chodov a nádob na sklo to je 60 odpadových nádob, depo a třídící linka, což dělá $5{,}0758*10^{85}$ možných cest. Tolik kombinací se nedá vypočítat v rozumném čase, tudíž se používají alternativní algoritmy, které hledají suboptimální řešení.

$$n_{kombinací} = (n-1)! \tag{9}$$

Při výpočtu nejkratší trasy v této případové studii byl použit greedy algoritmus, který spočívá v podívání se na všechny sousední body a následné přemístění do bodu, který je nejblíž. Příklad je na obrázku 42. Jak je dále vidět na obrázku 42, v tomto případě se nejedná o optimální řešení, protože celková délka trasy vybraná greedy algoritmem je 10 jednotek, optimální trasa má ale délku 4 jednotky. Nicméně

v kontextu najití nejkratších cest mezi odpadovými nádobami, tento algoritmus poskytoval podobné výsledky jako ostatní používané heuristické a aproximační algoritmy. Greedy algoritmus byl zvolen z důvodu své jednoduchosti, uspokojivým výsledkům a nízkým nárokům na výpočetní výkon.



Obrázek 42 - Greedy algoritmus

6.1.1 Varianty simulace

Bylo vytvořeno pět variant simulace. V každé variantě je simulována dynamika zaplňování odpadových nádob stejně, liší se ve způsobu svozu. Varianta 0 je referenční varianta reprezentující nynější způsob svozu odpadu. Varianty 1 až 5 simulují optimalizovaný svoz.

Varianta 0

Varianta 0 simuluje nynější způsob svozu odpadu. Nynější způsob spočívá ve fixní frekvenci svozu po fixní trase. To znamená, že odpady se svážejí pravidelně jednou za určitý počet dní po předem definované trase. Když nastane den svozu, auto vyjíždí z depa a začíná svážet odpady po své trase. Pokud se auto zaplní během svozu, jede se vysypat do třídičky odpadů a pokračuje ve svozu. Ke konci svozu jede do třídičky odpadů se vysypat a poté zpět do depa. Zdrojový kód simulace varianty 0 je v příloze 1.

Varianta 1

Varianta 1 simuluje svoz po stejné fixní trase jako varianta 0 s tím, že se vysypávají pouze nádoby, které jsou zaplněny z více než 45 %. Pokud se svozové auto na trase zaplní, jede se vysypat do třídičky a vrací se zpět na trasu. Po svezení všech vytipovaných nádob auto jede do třídičky odpadů, kde se vysype a jede zpět do depa. Premisa je taková, že pokud dynamika zaplňování nádoby je tak pomalá, že ve svozovém dnu není nádoba zaplněna ani z 45 %, je možné jí vyvézt až následující svozový cyklus, aniž by se přeplnila a tím utrpěla kvalita služby sběru odpadu z pohledu občanů. Zdrojový kód simulace varianty 1 je v příloze 2.

Varianta 2

Varianta 2 je plně optimalizovaný dynamický svoz odpadu. V této variantě svoz odpadu nastává až když se nějaká nádoba zcela zaplní. Po zjištění zaplnění nějaké nádoby se vezme zaplnění všech nádob, seřadí se od největšího k nejmenšímu a vybírají se nejzaplněnější nádoby do té doby, dokud stačí kapacita auta. Poté se vypočítá nejkratší trasa této podmnožiny nejzaplněnějších nádob tak, aby start byl v depu a cíl v třídičce odpadů. V této variantě nenastane během svozu nečekané zaplnění auta, a tedy nutnost se

odklonit od trasy do třídící linky, protože zaplnění auta je předem známo. Po vysypání auta na třídící lince auto jede zpět do depa.

Varianta 3

Varianta 3 je dynamický svoz po fixní trase. Tato varianta je velmi podobná variantě 2. Rozdíl je v tom, že se jezdí vždy po fixní trase. Poté, co se některá nádoba zaplní, vezme se naplnění všech nádob, seřadí se a vyberou se nejzaplněnější nádoby do té doby, než je vyčerpána kapacita auta. Poté auto vyjíždí po fixní trase jako ve variantách 0 a 1, vyváží vytipované nádoby a po vyvezení všech nádob jede na třídičku odpadů se vysypat. Následně jede zpět do depa.

Tato varianta byla vytvořena, protože některé svozové společnosti si chtějí zachovat své svozové trasy z důvodu familiárnosti případných překážek a problémů na trase.

Varianta 4

Varianta 4 je kombinací varianty 2 a varianty 1. Jedná se o svoz v pravidelném intervalu nádob se zaplněním nad 45 %. Před svozem jsou vytipovány nádoby s naplněním větším než 45 % a je vypočítána nejkratší trasa mezi nimi s cílem v třídičce odpadů. Pokud na svozové trase se auto zaplní, jede se vysypat a vrací se zpět na trasu. Na konci svozu se auto vysypává na třídící lince a vrací se zpět do depa.

Varianta 5

Varianta 5 je dynamický svoz nádob se zaplněním vyšším než 80 %. V této variantě dojde ke svozu, pokud se některá nádoba zcela zaplní. Poté se spočítá nejkratší trasa mezi nádobami, které jsou zaplněné více než z 80 % a vyvezou se. Tato varianta má reprezentovat stav, kdy je požadována co nejvyšší výtěžnost nádob nehledě na náklady svozu.

6.1.2 Vstupní údaje

Vstupní údaje do simulace jsou v tabulce 12. Informace byly získány od svozové společnosti Chotes s.r.o., města Chodov a poradenské společnosti Odpadová poradenská s.r.o. *Průměrná rychlost při svozu* je průměrná rychlost, kterou se auto pohybuje při svozu mezi nádobami. Rychlost a tedy i čas při cestě do třídičky nebo zpět do depa je brána z dynamických dat Googlu, protože se předpokládá, že svozové auto při přejezdech, když není v módu svážení, se dokáže pohybovat stejně rychle jako běžný automobil.

Tabulka 12 - Vstupní údaje do simulace

Svozová společnost		
Frekvence svozu každých	28	dní
Max zaplnění vozu	20	m^3
Čas výsypu nádoby	1	min
Čas na třídící lince	20	min
Příprava auta a posádky na svoz	10	min
Ukončení svozu	10	min
Hodinové náklady	800	Kč/h
Náklady na km	15	Kč/km
Průměrná rychlost při svozu	10	km/h
Město		
Cena výsypu jedné nádoby (kolik město platí svozové spol.)	250	Kč

6.1.3 Výsledky simulace

Simulace každé varianty byla nastavena na délku trvání jednoho roku. Poskytnuté výsledky boudou použity při ekonomickém zhodnocení. Výsledky simulací jednotlivých variant jsou v tabulkách 13 až 18.

Tabulka 13 - Výsledky varianty 0

Varianta 0		
Počet najetých kilometrů	595,11	km
Provozní čas	89,84	h
Celkové náklady na svoz	**82 122**	**Kč**
Počet svozů	13	
Počet výsypů	26	
Počet vyvezených nádob	780	
Průměrné zaplnění nádob při výsypu	46,39%	
Cena svozu město	**195 000**	**Kč**

Tabulka 14 - Výsledky varianty 1

Varianta 1		
Počet najetých kilometrů	578,6	km
Provozní čas	85,74	h
Celkové náklady na svoz	78 532	Kč
Počet svozů	13	
Počet výsypů	26	
Počet vyvezených nádob	565	
Průměrné zaplnění nádob při výsypu	62,48%	
Cena svozu město	141 250	Kč
Rozdíl oproti referenční variantě 0		
Celkové náklady na svoz	- 3 590	Kč
Cena svozu město	- 53 750	Kč

Tabulka 15 - Výsledky varianty 2

Varianta 2		
Počet najetých kilometrů	479,92	km
Provozní čas	71,13	h
Celkové náklady na svoz	64 101	Kč
Počet svozů	19	
Počet výsypů	19	
Počet vyvezených nádob	486	
Průměrné zaplnění nádob při výsypu	68,85%	
Cena svozu město	121 500	Kč
Rozdíl oproti referenční variantě 0		
Celkové náklady na svoz	- 18 021	Kč
Cena svozu město	- 73 500	Kč

Tabulka 16 - Výsledky varianty 3

Varianta 3		
Počet najetých kilometrů	603,92	km
Provozní čas	84,18	h
Celkové náklady na svoz	77 609	Kč
Počet svozů	19	
Počet výsypů	19	
Počet vyvezených nádob	486	
Průměrné zaplnění nádob při výsypu	68,85%	
Cena svozu město	121 500	Kč
Rozdíl oproti referenční variantě 0		
Celkové náklady na svoz	- 4 513	Kč
Cena svozu město	- 73 500	Kč

Tabulka 17 - Výsledky varianty 4

Varianta 4		
Počet najetých kilometrů	523,7	km
Provozní čas	80,92	h
Celkové náklady na svoz	73 718	Kč
Počet svozů	13	
Počet výsypů	26	
Počet vyvezených nádob	572	
Průměrné zaplnění nádob při výsypu	61,79%	
Cena svozu město	143 000	Kč
Rozdíl oproti referenční variantě 0		
Celkové náklady na svoz	- 8 344	Kč
Cena svozu město	- 52 000	Kč

Tabulka 18 - Výsledky varianty 5

Varianta 5		
Počet najetých kilometrů	771,71	km
Provozní čas	120,88	h
Celkové náklady na svoz	110 068	Kč
Počet svozů	47	
Počet výsypů	47	
Počet vyvezených nádob	391	
Průměrné zaplnění nádob při výsypu	86,32%	
Cena svozu město	97 750	Kč
Rozdíl oproti referenční variantě 0		
Celkové náklady na svoz	27 946	Kč
Cena svozu město	- 97 250	Kč

Z výsledků jednotlivých simulací je vidět, že oproti referenční variantě 0 dosahuje největší úspory z pohledu svozové firmy varianta 2 (tabulka 15) a z pohledu města varianta 5 (tabulka 18). Varianta 5 je ale extrémní varianta, kdy město tlačí na vysypávání nádob zaplněných nad 80 % nehledě na náklady svozu. Náklady svozu svozové firmy ale převyšují příjmy, tudíž tato varianta není dlouhodobě udržitelná. Z těchto důvodů bude v ekonomickém zhodnocení pracováno pouze s variantou 2.

6.2 Ekonomické zhodnocení

V ekonomickém zhodnocení se bude podíváno na ekonomické aspekty projektu inovovaného svozu odpadu z pohledu města a svozové společnosti. Cena za službu monitoringu zaplnění odpadových nádob bude fixně daná a nebude o ní diskutováno.

Je návrh uskutečnění projektu sledování zaplněnosti odpadových nádob a dynamického svozu odpadu (varianta 2) spočívající v instalaci 60 měřících jednotek do nádob na sklo. Systém sledování zaplněnosti město/svozová společnost zakoupí jako službu – tj. bude placen měsíční poplatek ve výši 100 Kč za každou měřící jednotku s investicí 0 Kč. Projekt má životnost 7 let a diskont 3,5 %. Diskont byl převzat ze studie projektu „Chytrý svoz odpadu“ od společnosti Operátor ICT a.s. a Pražského magistrátu. V projektu bude počítáno s 2,5% roční eskalací nákladů na svoz a 2,5% roční eskalací ceny výsypu jedné nádoby vycházející z průměrné inflace za poslední tři roky. Ušetřené náklady na svoz vycházející z optimalizovaného systému budou počítány jako fiktivní příjmy. Jelikož se ale nejedná o reálné příjmy, ale pouze o ušetřené náklady, nebudou se danit. Zároveň se počítá s nulovými náklady na údržbu, protože veškerou údržbu zajišťuje společnost dodávající řešení v rámci měsíčního poplatku. Jediné provozní náklady budou náklady na systém – tj. měsíční poplatek za měřící jednotku. Tento poplatek zůstane po celou dobu trvání projektu fixní.

Jelikož je počáteční investice nulová, bude se projekt hodnotit pouze ukazatelem čisté současné hodnoty (NPV).

6.2.1 Výsledky pohled město

Z pohledu města se není díváno na ušetřené náklady svozu (hodinové náklady, náklady na km), ale pouze ušetřené náklady za platbu svozu svozové firmě. Je předpokládáno, že město platí svozové firmě za množství výsypů nádob, které může dynamicky měnit. Vstupní informace modelu jsou v tabulce 19,

Tabulka 19 - Vstupní informace ekonomického modelu pohled město

Životnost projektu	7	let
Roční rozdíl množství vysypaných nádob oproti variantě 0	-294	
Cena výsypu jedné nádoby	250	Kč
Roční ušetřené náklady za výsyp nádob (výnos)	73 500	Kč
Diskont	3,50 %	
Měsíční jednotkový náklad na systém	100	Kč/ks
Počet měřících jednotek	60	ks
Eskalace nákladů za výsyp nádob	2,5 %	

Při vstupech z tabulky 19 vychází NPV projektu na **54 748 Kč**. Průběh cash flow, diskontovaného cash flow a diskontovaného kumulovaného cash flow je vidět v tabulce 20.

Tabulka 20 - Průběh CF, DCF, DCCF případové studie pohled město

Rok	CF	DCF	DCCF
0	- CZK	- CZK	- CZK
1	3 338 CZK	3 225 CZK	3 225 CZK
2	5 221 CZK	4 874 CZK	8 098 CZK
3	7 151 CZK	6 450 CZK	14 549 CZK
4	9 130 CZK	7 956 CZK	22 505 CZK
5	11 159 CZK	9 395 CZK	31 900 CZK
6	13 237 CZK	10 769 CZK	42 669 CZK
7	15 368 CZK	12 079 CZK	**54 748 CZK**

Citlivostní analýza

Ekonomické parametry *cena výsypu jedné nádoby* a *měsíční náklady na systém* nejvíce ovlivňují čistou současnou hodnotu projektu. Citlivostní analýza těchto parametrů byla provedena v tabulce 21. Proměnlivý parametr c*ena výsypu jedné nádoby* má simulovat výsyp různých typů nádob – v případě Chodova všechny nádoby jsou klasické 1100 litrové nádoby s vrchním výsypem. Výsyp těchto nádob je nejlevnější, protože jsou na trhu nejrozšířenější a mechanika výsypu je velmi jednoduchá. V Praze a velkém množství měst ale tyto typy nádob na sklo vůbec nejsou. V Praze jsou pouze 1100 a 3200 litrové nádoby se spodním výsypem a podzemní kontejnery s objemem 3000 litrů. Cena výsypu těchto nádob je mnohem dražší, protože je potřeba speciální vozidlo s hydraulickým ramenem. Jeden výsyp těchto nádob stojí zhruba 400 Kč, 850 Kč a 800 Kč popořadě – cena se odvíjí od počtu a umístění nádob [25]. Dále je vidět, že cena 100 Kč za měřící jednotku (parametr měsíční náklady na systém) je téměř hraniční, aby NPV bylo ještě kladné. Při ceně 120 Kč/měsíc/měřící jednotka je NPV záporné a projekt už nedává z ekonomického pohledu smysl.

Tabulka 21 - Citlivostní analýza projektu, pohled města

Měsíční jednotkové náklady na systém [Kč/ks] \ Cena výsypu jedné nádoby [Kč]	150	250	350	450	550	650	750	850
50	76 874 CZK	274 872 CZK	472 870 CZK	670 868 CZK	868 867 CZK	1 066 865 CZK	1 264 863 CZK	1 462 861 CZK
60	32 849 CZK	230 847 CZK	428 845 CZK	626 844 CZK	824 842 CZK	1 022 840 CZK	1 220 838 CZK	1 418 837 CZK
70	- 11 176 CZK	186 823 CZK	384 821 CZK	582 819 CZK	780 817 CZK	978 815 CZK	1 176 814 CZK	1 374 812 CZK
80	- 55 200 CZK	142 798 CZK	340 796 CZK	538 794 CZK	736 793 CZK	934 791 CZK	1 132 789 CZK	1 330 787 CZK
90	- 99 225 CZK	98 773 CZK	296 771 CZK	494 770 CZK	692 768 CZK	890 766 CZK	1 088 764 CZK	1 286 762 CZK
100	- 143 250 CZK	54 748 CZK	252 747 CZK	450 745 CZK	648 743 CZK	846 741 CZK	1 044 740 CZK	1 242 738 CZK
110	- 187 275 CZK	10 724 CZK	208 722 CZK	406 720 CZK	604 718 CZK	802 717 CZK	1 000 715 CZK	1 198 713 CZK
120	- 231 299 CZK	- 33 301 CZK	164 697 CZK	362 695 CZK	560 694 CZK	758 692 CZK	956 690 CZK	1 154 688 CZK
130	- 275 324 CZK	- 77 326 CZK	120 672 CZK	318 671 CZK	516 669 CZK	714 667 CZK	912 665 CZK	1 110 664 CZK
140	- 319 349 CZK	-121 350 CZK	76 648 CZK	274 646 CZK	472 644 CZK	670 642 CZK	868 641 CZK	1 066 639 CZK
150	- 363 373 CZK	-165 375 CZK	32 623 CZK	230 621 CZK	428 620 CZK	626 618 CZK	824 616 CZK	1 022 614 CZK

6.2.2 Výsledky pohled svozová firma

Z pohledu svozové firmy jsou zajímavé náklady na svoz. Vstupní informace ekonomického modelu jsou v tabulce 22.

Tabulka 22 - Vstupní informace ekonomického modelu, pohled svozová firma

Životnost projektu	7	let
Roční ušetřené náklady za výsyp nádob (výnos)	18 021	Kč
Diskont	3,50%	
Měsíční jednotkový náklad na systém	100	Kč/ks
Počet měřících jednotek	60	ks
Eskalace nákladů svozu odpadu	2,5%	

Při vstupech z tabulky 22 vychází NPV projektu na **-318 882 Kč**. Průběh cash flow, diskontovaného cash flow a diskontovaného kumulovaného cash flow je vidět v tabulce 23.

Tabulka 23 - Průběh CF, DCF, DCCF ekonomického modelu, pohled svozová firma

Rok	CF	DCF	DCCF
0	- CZK	- CZK	- CZK
1	- 53 528 CZK	- 51 718 CZK	- 51 718 CZK
2	- 53 067 CZK	- 49 538 CZK	- 101 257 CZK
3	- 52 593 CZK	- 47 436 CZK	- 148 693 CZK
4	- 52 108 CZK	- 45 409 CZK	- 194 102 CZK
5	- 51 611 CZK	- 43 455 CZK	- 237 557 CZK
6	- 51 101 CZK	- 41 571 CZK	- 279 128 CZK
7	- 50 579 CZK	- 39 754 CZK	- **318 882 CZK**

Citlivostní analýza

Z citlivostní analýzy na *parametry eskalace nákladů svozu odpadu* a *měsíční jednotkové náklady na systém* (tabulka 24) je vidět, že i kdyby nastal velký nečekaný meziroční růst nákladů (budoucích fiktivních výnosů) 6 %, projekt by ani tak nebylo ekonomicky smysluplné realizovat. Aby NPV projektu bylo 0 a projekt se tedy vyplatilo realizovat, musel by být měsíční jednotkový náklad na systém **27,5 Kč** s *eskalací nákladů svozu odpadu 2,5 %.*

Tabulka 24 - Citlivostní analýza ekonomického modelu, pohled svozové společnosti

Měsíční jednotkové náklady na systém [Kč/ks]	Eskalace nákladů (budoucích fiktivních výnosů) svozu odpadu [%]							
	2,5%	3%	3,5%	4,0%	5%	5,0%	5,5%	6%
10	77 340 CZK	79 708 CZK	82 122 CZK	84 584 CZK	87 093 CZK	89 651 CZK	92 259 CZK	94 917 CZK
20	33 315 CZK	35 683 CZK	38 098 CZK	40 559 CZK	43 068 CZK	45 626 CZK	48 234 CZK	50 893 CZK
30	- 10 709 CZK	- 8 341 CZK	- 5 927 CZK	- 3 466 CZK	- 957 CZK	1 602 CZK	4 209 CZK	6 868 CZK
40	- 54 734 CZK	- 52 366 CZK	- 49 952 CZK	- 47 491 CZK	- 44 981 CZK	- 42 423 CZK	- 39 815 CZK	- 37 157 CZK
50	- 98 759 CZK	- 96 391 CZK	- 93 977 CZK	- 91 515 CZK	- 89 006 CZK	- 86 448 CZK	- 83 840 CZK	- 81 182 CZK
60	- 142 783 CZK	-140 416 CZK	-138 001 CZK	- 135 540 CZK	-133 031 CZK	- 130 473 CZK	- 127 865 CZK	- 125 206 CZK
70	- 186 808 CZK	-184 440 CZK	-182 026 CZK	- 179 565 CZK	-177 055 CZK	- 174 497 CZK	- 171 889 CZK	- 169 231 CZK
80	- 230 833 CZK	-228 465 CZK	-226 051 CZK	- 223 589 CZK	-221 080 CZK	- 218 522 CZK	- 215 914 CZK	- 213 256 CZK
90	- 274 858 CZK	-272 490 CZK	-270 075 CZK	- 267 614 CZK	-265 105 CZK	- 262 547 CZK	- 259 939 CZK	- 257 280 CZK
100	- 318 882 CZK	-316 514 CZK	-314 100 CZK	- 311 639 CZK	-309 130 CZK	- 306 571 CZK	- 303 964 CZK	- 301 305 CZK
110	- 362 907 CZK	-360 539 CZK	-358 125 CZK	- 355 664 CZK	-353 154 CZK	- 350 596 CZK	- 347 988 CZK	- 345 330 CZK

6.2.3 Diskuse výsledků a závěr ekonomického zhodnocení

Ekonomický ukazatel NPV varianty 2 optimalizovaného svozu vyšel **54 748 Kč** z pohledu města a **(- 318 882 Kč)** z pohledu svozové firmy. Tyto čísla potvrzují premisu na začátku práce zmiňující motivace jednotlivých subjektů. Svozové společnosti chtějí vysypávat co největší počet odpadových nádob – ideálně nezaplněných – protože v drtivé většině kontraktů mezi nimi a městy/obcemi je to právě počet výsypů, který určuje finální roční cenu za svoz. Čím více nádob vysypávají, tím větší mají příjmy. Na druhou stranu města, jak je vidět z vypočítaného NPV, by ze zavedení systému mohly ekonomicky benefitovat, protože v případě naddimenzované svozové infrastruktury by se zvýšila výtěžnost jednotlivých nádob a systém by byl efektivnější. V případě poddimenzované svozové infrastruktury by systém pravděpodobně přímé náklady na svoz neušetřil, ale vyřešil by přeplňování odpadových nádob, čímž by se zlepšila služba občanům a ušetřilo by se za uklízecí čety, které jezdí mimo svozový plán uklízet kolem odpadových nádob. Z pohledu města je tedy projekt optimalizovaného svozu doporučen, z pohledu svozové společnosti nikoliv. Častý případ je, že město je vlastníkem místních technických služeb, které provádějí svoz odpadu. V tomto případě by projekt byl doporučen městu i svozové společnosti (místním technickým službám), protože celý systém svozu by se tím zefektivnil a oba subjekty vnímané jako jeden by ušetřily náklady.

6.3 Diskuse obecných přínosů a příležitostí navrženého systému

Navržený systém by kromě ušetřených nákladů ve formě optimalizace svozu a vyřešení problémů adresovaných v kapitole 2.1.4 Problémy nynějšího svozu odpadu, mohl také při dlouhodobějším provozu poskytnou detailní data o produkci jednotlivých typů odpadů v jednotlivých oblastech města. Tyto data by potom mohly sloužit k pochopení obyvatelstva z pohledu jejich odpado-produkčních návyků a následně by bylo jednodušší cílit edukaci týkající se třídění odpadů či samotné produkce odpadu. To by mohlo městům, a tedy i obyvatelům v dlouhodobém horizontu ušetřit další peníze, protože nevytříděný komunální odpad se v České republice ve většině skládkuje. Dle nového odpadového zákona č. 541/2020 sb. v roce 2021 platí pro obce a města skládkovací poplatek 500 Kč za tunu s postupným zvyšováním až do roku 2029, kde bude poplatek 1850 Kč za tunu a zároveň dojde k zákazu skládkování. Méně směsného komunálního odpadu by ušetřilo městům a občanům skládkovací poplatky. Dále by se mohly dlouhodobější data použít k optimalizaci rozmístění nádob a k optimalizaci

vozového parku a personálního obsazení svozových firem či technických služeb měst, což by mohlo dále snížit náklady na odpadové hospodářství měst.

7 Závěr

Cílem této diplomové práce bylo zanalyzovat problémy současného stavu svozu separovaného odpadu, zjistit nynější běžné způsoby a celkovou roli sledování zaplnění odpadových nádob v kontextu svozu separovaného odpadu a na základě těchto informací navrhnout, realizovat a otestovat inovovaný systém pro sledování zaplnění odpadových nádob, popsat jeho přínosy a navrhnout jeho uplatnění a možnosti integrace do nynějšího svozového systému.

Od městských referentů starající se o odpadové hospodářství bylo zjištěno, že největším problémem nynějšího svozu je absence aktuální a spolehlivé informace o zaplnění nádob. Bez této informace je velmi těžké nastavit infrastrukturu sběru odpadu tak, aby se nádoby nepřeplňovaly ale zároveň, aby se co nejvíce využíval jejich objem a svoz byl tak efektivní. Dalšími problémy jsou nespravedlivé rozdělení odměn ze Systému EKOKOM v případě malých obcí a kontrola nepoctivých svozových společností, které nevyvážejí nádoby podle uzavřených smluv. Všechny tyto problémy řeší průběžný monitoring zaplnění odpadových nádob. Jak bylo ale zjištěno z průzkumu současného stavu monitoringu zaplnění odpadových nádob ve všech 94 městech nad 14 000 obyvatel v České republice, pouze sedm měst provozuje nějaký typ systému sledování zaplnění opadových nádob. Pět měst používá poloautomatický způsob monitoringu, dvě města plně automatický. Poloautomatický způsob se, dle zkušeností odpadových referentů měst, ukazuje jako nefunkční z důvodu poskytování nespolehlivých informací. Plně automatickým systémem sledování zaplněnosti disponuje pouze Kolín a Praha, které sledují podzemní kontejnery. Na základě těchto zjištění byl pro města nedisponujícím žádným nebo nefunkčním systémem, navržen inovovaný systém sledování odpadových nádob, který by řešil problémy nynějšího svozu odpadu. Systém se skládá z elektronických měřících jednotek umístěných v odpadových nádobách, serveru a webové aplikace, která přijímá data z jednotek, ukládá je a prezentuje uživateli v přehledném a uživatelsky přívětivém stylu. Měřící jednotka měří zaplněnost pomocí ultrazvukového senzoru a přenáší data přes GSM síť. Webová aplikace používá MySQL databázi a je napsaná v programovacím jazyce Python s použitím frameworků Flask a Plotly Dash.

V rámci realizace byla vyrobena prototypová série 50 kusů měřících jednotek, která byla otestována v laboratorních testech a dlouhodobém čtyřměsíčním testu v reálných podmínkách ve spolupráci s městem Starý Plzenec, kdy bylo 32 měřících jednotek osazeno do nádob na papír s následným přendáním po dvou měsících do nádob na plast. Po odstranění nedostatků návrhu, které vyplynuly z testování, byl technický koncept uznán jako dobře zvolený a funkční.

Pro zjištění ekonomických přínosů byla realizována případová studie, která spočívala v simulaci optimalizovaného sběru odpadu v několika variantách na základě reálných dat a následném ekonomickém vyhodnocení z pohledu města a pohledu svozové firmy. Ekonomická a provozní vstupní data byla získána od svozové společnosti města Chodov, data o zaplnění byla získána aplikací realizovaného systému sledování zaplnění. Bylo nainstalováno 60 měřících jednotek do všech nádob na tříděné sklo, kterými město Chodov disponuje a měřilo se zaplnění každé odpadové nádoby po dobu tří svozových cyklů – tří měsíců. Z dat se ukázalo, že město Chodov má velmi naddimenzovanou infrastrukturu odpadových nádob na sběr skla, protože nádoby se vyvážejí v průměru z 46 % zaplněné.

Ze shromážděných dat byla vypočítána dynamika zaplňování jednotlivých nádob, což byl základní podklad pro simulaci. Diskrétní časová simulace byla napsána v programovacím jazyce Python a simulovala zaplňování den po dni s optimalizovaným svozem v šesti variantách po dobu jednoho roku. Pro získání vzdáleností a časů mezi odpadovými nádobami, třídící linkou a depem, bylo použito Google Maps matrix API. Vypočítání nejkratších tras mezi jednotlivými místy představoval problém obchodního cestujícího, který byl řešen greedy algoritmem. Referenční varianta 0 simulovala nynější stav – svozová firma jednou za 28 dní vyváží všechny nádoby po fixní svozové trase nehledě na zaplnění. V této variantě vyšly náklady svozové firmy na roční svoz odpadu 82 122 Kč, náklady města na službu svozu 195 000 Kč. Varianta 1 počítala pouze s vysypáváním nádob nad 45 % zaplnění po fixní svozové trase. Premisa byla taková, že pokud je zaplnění pod 45 %, nádoba se může vyvézt až následující svozový cyklus, aniž by se přeplnila. Rozdíly oproti referenční variantě vyšly na -3 590 Kč ročních nákladů na svoz, -53 750 Kč náklady města na službu svozu. Varianta 2 simulovala plně dynamický svoz spočívající v najití nejkratší možné trasy nejzaplněnějších nádob. Náklady na svoz v této variantě oproti referenční variantě byly -18 021 Kč pro svozovou firmu, náklady města - 73 500 Kč. Varianta 3 simulovala dynamický svoz po fixní trase. Náklady této varianty oproti referenční variantě vyšly na - 4 513 Kč pro svozovou firmu, -73 500 Kč pro město. Varianta 4 reprezentovala svoz v pravidelném intervalu (stejném jako varianta 0) nádob se zaplněním nad 45 % po co nejkratší trase. Oproti referenční variantě vyšly náklady -8 344 Kč pro svozovou společnost a -52 000 Kč pro město. Poslední varianta 5 měla simulovat požadavek města co nejvyššího vytěžování objemu odpadových nádob nehledě na náklady svozu. Vyvážely se pouze nádoby s naplněním nad 80 % po co nejkratší trase. Rozdíl nákladů oproti referenční variantě vyšel na +27 946 Kč pro svozovou společnost a -97 250 Kč pro město. V této variantě byly ale příjmy svozové společnosti nižší než náklady na svoz, tudíž byla zavržena jako dlouhodobě neudržitelná a tedy neuskutečnitelná.

Pro ekonomické zhodnocení byla vybrána varianta 2, protože dosahovala nejvyšších ušetřených nákladů jak z pohledu města, tak i svozové firmy. Byl zhodnocen projekt, který počítal s instalací 60 měřících jednotek po dobu sedmi let s tím, že služba monitoringu a optimalizace svozu je ve fixním měsíčním poplatku 100 Kč/měřící jednotka a nejsou tedy žádné investiční náklady z pohledu pořizovatele služby (města/svozové společnosti). Z ekonomické analýzy vyšlo NPV projektu z pohledu města 54 748 Kč a z pohledu svozové společnosti -318 882 Kč. Pro město tedy projekt sledování zaplnění odpadových nádob s následnou optimalizací byl doporučen, pro svozovou společnost nikoliv. Jedinou výjimku tvoří situace, kdy město vlastní městské technické služby, které zajišťují svoz odpadu. V tomto případě by byl projekt doporučen i svozové společnosti, protože optimalizovaným svozem dojde k celkovému snížení nákladů na svoz všech stran a zároveň poskytne mimoekonomické přínosy.

Seznam použité literatury

[1] United Nations, "68% of the world population projected to live in urban areas by 2050, says UN | UN DESA | United Nations Department of Economic and Social Affairs." https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html (accessed Apr. 04, 2021).

[2] E.-K. A.s., "Průvodce systémem sdruženého plnění povinností zpětného odběru a využití odpadu z obalů."

[3] E.-K. A.s., "Příloha č. 3 - Ceník odměn obcím a městům." Accessed: Apr. 14, 2021. [Online]. Available: https://www.ekokom.cz/uploads/attachments/Obce/zmeny od 3Q2020/Priloha_3_Sazebniky_odmen_od_3Q_2020.pdf.

[4] EKOKOM, "Standardy složení komunálních odpadů a podílu obalové složky," 2021. https://www.ekokom.cz/uploads/attachments/Obce/zmeny od 1Q2021/Priloha_4_Standardy_slozeni_KO_od_1Q 2021.pdf (accessed May 19, 2021).

[5] Václav Plecháček, "Chytrá tlačítka zajišťují ve Dvoře Králové u některých sběrných nádob rychlejší odvoz skla | Hradec Králové," *hradec.rozhlas.cz*, Mar. 26, 2019. https://hradec.rozhlas.cz/chytra-tlacitka-zajistuji-ve-dvore-kralove-u-nekterych-sbernych-nadob-rychlejsi-7800987 (accessed Apr. 26, 2021).

[6] Círová Dita, "Aplikace odhalí přeplněné koše: Krnov," Feb. 05, 2018. https://www.krnov.cz/aplikace-odhali-preplnene-kose/d-26000 (accessed Apr. 27, 2021).

[7] Melzer Tomáš, "S efektivnějším svozem pomáhají čárové kódy | OFICIÁLNÍ STRÁNKY MĚSTA ZLÍNA," Nov. 01, 2019. http://m.zlin.eu/s-efektivnejsim-svozem-pomahaji-carove-kody-aktuality-5258.html (accessed Apr. 27, 2021).

[8] MěÚ Kolín, "Kolín se zapojil do pilotního projektu chytrého odpadového hospodářství | mukolin.cz," 2016. https://www.mukolin.cz/cz/o-meste/062669-kolin-se-zapojil-do-pilotniho-projektu-chytreho-odpadoveho-hospodarstvi.html (accessed Apr. 27, 2021).

[9] Slizek David, "Kolín: Senzory v kontejnerech nic neušetří, město je ale uklizenější - Lupa.cz," 2016. https://www.lupa.cz/clanky/kolin-senzory-v-kontejnerech-nic-neusetri-mesto-je-ale-uklizenejsi/ (accessed Apr. 27, 2021).

[10] Operátor ICT a.s., "Projekt chytrého svozu odpadu - závěrečná zpráva."

[11] K. Novák, "Dálkový monitoring odpadových nádob na tříděný odpad," ČVUT FEL.

[12] Burnett Roderick, "Ultrasonic vs Infrared (IR) Sensors - Which is better? | MaxBotix Inc.," Nov. 27, 2017. https://www.maxbotix.com/articles/ultrasonic-or-infrared-sensors.htm (accessed Apr. 29, 2021).

[13] Terabee, "Choosing the Right Distance Sensor for Your Application." https://www.terabee.com/choosing-right-distance-sensor-your-application/ (accessed May 20, 2021).

[14] "Ceník / VOP | O nás | Sigfox." https://sigfox.cz/cs/o-nas/cenik-vop (accessed Apr. 29, 2021).

[15] STARNET s.r.o., "Ceník LORA pro IoT," 2018. https://www.starnet.cz/download/cenik-iot.pdf (accessed Apr. 29, 2021).

[16] Microchip Inc., "Microchip Atmega328p datasheet," 2021. Accessed: Apr. 30, 2021. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061B.pdf.

[17] "Grove - Ultrasonic Distance Sensor - Seeed Studio." https://www.seeedstudio.com/Grove-Ultrasonic-Distance-Sensor.html (accessed Apr. 30, 2021).

[18] SIMCom, "SIM800H&SIM800L_Hardware Design_V2.02," 2015.

[19] "RTC DS3231 data sheet." Accessed: May 01, 2021. [Online]. Available: www.maximintegrated.com.

[20] "Elevating Self-discharge - Battery University," 2011. https://batteryuniversity.com/index.php/learn/article/elevating_self_discharge (accessed May 01, 2021).

[21] Microchip, "MCP1726 datasheet."

[22] “Decoupling Capacitor vs Bypass Capacitor – Working & Applications,” 2019. https://components101.com/articles/decoupling-capacitor-vs-bypass-capacitors-working-and-applications (accessed May 02, 2021).
[23] “CE Cetification, Manufacturers | Internal Market, Industry, Entrepreneurship and SMEs.” https://ec.europa.eu/growth/single-market/ce-marking/manufacturers_en (accessed May 03, 2021).
[24] P. Ing Jiří Svačina and Cs. Ústav Radioelektroniky, “ELEKTROMAGNETICKÁ KOMPATIBILITA.”
[25] H. město Praha, “Smlouva o poskytnutí služeb č.ino/54/11/010585/2016,” 2016.

Seznam tabulek

Seznam obrázků

Přílohy

Příloha 1 – Zdrojový kód simulace svozu varianta 0

```
#varianta 0 - konvencni svoz - pravidelne vyvazeni nadob jednou za [frekvence_svozu] dni
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from functions import tsp,get_distance_matrix
import info_svoz

#import dat
df = pd.read_excel('gps_coordinates.xls') # can also index sheet by name or fetch all sheets
produkce = np.array(df['zaplnovani_absolut'].tolist())
vzdalenosti, casy = get_distance_matrix()

#informace o svoze
cena_vysypu = 250 #Kc
frekvence_svozu = 28 #jak casto (ve dnech) se odpad svazi
zivotnost_projektu = 1 #v letech
limit_vozu = 19 #v m3 - limit vozu 20 m3
cas_vysypu = 1 #jak dlouho trva zastaveni, vysypani a celkovy proces na jednu popelnici [min]
cas_tridicka = 20
id_depa = 0
id_tridicky = 1
priprava_svoz_cas = 40 #min zahrnuje pripravu a taky ukonceni svozu

#svozove naklady
hodinovka_posadka = 800 #Kc/
km_naklady = 15 #Kc/km

#ekonomicke informace
eskalace = 0.03 #rocni eskalace cen - zhruba kopiruje prumernou inflaci

#pomocne promenne
zaplneni = np.zeros(produkce.size)
pocet_nadob = produkce.size-2
celkova_cena_svozu = 0
zivotnost_dni = zivotnost_projektu*365
eskalace_denni = eskalace/365
celkovy_pocet_vysypu = 0
celkovy_pocet_svozu = 0
celkovy_pocet_kilometru = 0
celkovy_cas = 0
nadob_vyvezenych_celkem = 0
celkove_naklady_svoz = 0
celkove_naplneni = 0

#zde zacina simulace
for den in range(0, zivotnost_dni):
            zaplneni = zaplneni + produkce #zaplnovani jednotlivych popelnic - scitani dvou listu
            cena_vysypu = cena_vysypu*(1+eskalace_denni)
            hodinovka_posadka = hodinovka_posadka*(1+eskalace_denni)
            km_naklady = km_naklady*(1+eskalace_denni)

            #DEN SVOZU
            if(den % frekvence_svozu == 0 and den != 0):
                        print("-----------------------------------------------Svozovy den, den:", den, "-------
-----------------------------------")

                        #pomocne promenne k pocitani tras
                        path = tsp()
                        predchozi_nadoba = 0
                        ujete_kilometry = 0
```

```
61                      cas_na_trase = 0
62                      vyvezeny_objem = 0
63                      naplneni_auta = 0
64                      nadob_vyvezenych = 0
65                      pocet_vysypu = 0
66                      vyvezene = []
67
68                      #priprava na svoz
69                      cas_na_trase = cas_na_trase + priprava_svoz_cas
70
71                      celkovy_pocet_svozu = celkovy_pocet_svozu + 1
72
73                      #OBJIZDENI jednotlivych kontejneru
74                      for popelnice in path:
75
76                              #zde se scitaji celkem najete kilometry
77                              ujete_kilometry = ujete_kilometry + vzdalenosti[popelnice][predchozi_nadoba]
78                              cas_na_trase = cas_na_trase + casy[popelnice][predchozi_nadoba]
79                              predchozi_nadoba = popelnice
80                              # print(popelnice)
81                              # print("Ujeta vzdalenost", ujete_kilometry)
82                              # print("Cas na trase", cas_na_trase)
83                              vyvezene.append(popelnice)
84
85                              #VYSYPANI
86                              if(popelnice != 0 and popelnice != 1): #0, 1 jsou depo a tridicka - nevysypavat
87
88                                      vyvezeny_objem = vyvezeny_objem + zaplneni[popelnice]
89                                      naplneni_auta = naplneni_auta + zaplneni[popelnice]
90                                      celkove_naplneni = celkove_naplneni + zaplneni[popelnice]
91                                      zaplneni[popelnice] = 0
92                                      nadob_vyvezenych = nadob_vyvezenych + 1
93                                      cas_na_trase = cas_na_trase + cas_vysypu
94
95                                      #limit auta
96                                      if(naplneni_auta > limit_vozu):
97                                              print("Vozidlo naplneno", naplneni_auta, "jedu se
98 vysypat")
99                                              #odjedeme vozidlem na tridicku a vratime se k nynejsi
100 popelnici
101                                             ujete_kilometry = ujete_kilometry +
102 vzdalenosti[id_tridicky][popelnice]*2
103                                             cas_na_trase = cas_na_trase +
104 casy[id_tridicky][popelnice]*2
105                                             naplneni_auta = 0
106                                             pocet_vysypu = pocet_vysypu + 1
107                                             cas_na_trase = cas_na_trase + cas_tridicka
108                                             print("Ujete kilometry", ujete_kilometry)
109                                             print("-------------")
110
111                     #konec svozu -> nutno vysypat a z tridicky dostat zpet do depa
112                     print("naplneni vozidla", naplneni_auta)
113
114                     pocet_vysypu = pocet_vysypu + 1
115                     naplneni_auta = 0
116                     print("pocet_vysypu", pocet_vysypu)
117                     cas_na_trase = cas_na_trase + cas_tridicka
118
119                     #cesta zpet do depa
120                     ujete_kilometry = (ujete_kilometry + vzdalenosti[id_depa][popelnice])
121                     cas_na_trase = (cas_na_trase + casy[id_depa][popelnice])
122                     print("Ujete kilometry za cely svozovy den", ujete_kilometry)
123                     print("Cas na trase za cely svozovy den", cas_na_trase)
124
125                     celkovy_pocet_vysypu = celkovy_pocet_vysypu + pocet_vysypu
126
127                     celkova_cena_svozu = celkova_cena_svozu + nadob_vyvezenych * cena_vysypu
128                     celkovy_pocet_kilometru = celkovy_pocet_kilometru + ujete_kilometry
129                     celkovy_cas = celkovy_cas + cas_na_trase
```

```
                        nadob_vyvezenych_celkem = nadob_vyvezenych_celkem + nadob_vyvezenych
                        celkove_naklady_svoz    =    celkove_naklady_svoz    +    (ujete_kilometry*km_naklady    +
cas_na_trase/60*hodinovka_posadka)

                        print("Celkovy objem odpadu", vyvezeny_objem, "m3")
                        print("Celkovy pocet vyvezenych nadob", nadob_vyvezenych)
                        print("Cena svozu", nadob_vyvezenych*cena_vysypu)

            #PLOTING
            # try:
            #     graf_nazvy
            # except NameError:
            #     graf_nazvy = np.linspace(3, 62, num=60)
            # plt.cla()
            # plt.title(str(den))
            # plt.ylim([0, 1])
            # plt.bar(graf_nazvy,zaplneni[2:], align='center', color='red')
            # plt.pause(0.01)


print("============================================    Shrnuti    za    celou    dobu    projektu    varianty
0==============================================")
print("Celkovy pocet kilometru", "%.2f" % round(celkovy_pocet_kilometru, 2), "km")
print("Celkovy provozni cas", "%.2f" % round(celkovy_cas/60, 2), "hodin")
    print("Celkove naklady na svoz", int(celkove_naklady_svoz), "Kc")
    print("Celkovy pocet svozu", celkovy_pocet_svozu)
    print("Celkovy pocet vysypu", celkovy_pocet_vysypu)
    print("Celkovy pocet vyvezenych nadob", nadob_vyvezenych_celkem)
    print("Prumerne zaplneni vyvazenych nadob", "%.2f" % round(100/1.1*celkove_naplneni/nadob_vyvezenych_celkem, 2),
    "%")
    print("Cena svozu mesto", int(celkova_cena_svozu), "Kc")
    # print(celkovy_cas)
```

Příloha 2 – Zdrojový kód simulace svozu varianta 1

```
1   #varianta 1 - svoz v pravidelnem intervalu jednou za [frekvence_svozu] dni, ale pouze nadoby nad 0.5 m3 (45 %)
 2 import matplotlib.pyplot as plt
 3 import pandas as pd
 4 import numpy as np
 5 from functions import tsp,get_distance_matrix
 6
 7 #import dat
 8 df = pd.read_excel('gps_coordinates.xls') # stahnuti seznamu nadob, jejich dynamice zaplnovani a pozic
 9 produkce = np.array(df['zaplnovani_absolut'].tolist())
10 vzdalenosti, casy = get_distance_matrix() #ziskani aktualni vzdalenostni a casove matice pomoci google API
11
12 #informace o svoze
13 cena_vysypu = 250 #Kc
14 frekvence_svozu = 28 #jak casto (ve dnech) se odpad svazi
15 zivotnost_projektu = 1 #v letech
16 limit_vozu = 19 #v m3 - limit vozu 20 m3
17 cas_vysypu = 1
18 cas_tridicka = 20
19 id_depa = 0
20 id_tridicky = 1
21 limit_zaplneni = 0.5 #jak plne popelnice se maji vyvazet
22 priprava_svoz_cas = 40 #min zahrnuje pripravu a taky ukonceni svozu
23
24 #svozove naklady
25 hodinovka_posadka = 800 #Kc/
26 km_naklady = 15 #Kc/km
27
28 #ekonomicke informace
29 eskalace = 0.03 #rocni eskalace cen - zhruba kopiruje prumernou inflaci
30
31 #pomocne promenne
32 zaplneni = np.zeros(produkce.size)
```

```
pocet_nadob = produkce.size-2
celkova_cena_svozu = 0
zivotnost_dni = zivotnost_projektu*365
eskalace_denni = eskalace/365
celkovy_pocet_vysypu = 0
celkovy_pocet_svozu = 0
celkovy_pocet_kilometru = 0
celkovy_cas = 0
nadob_vyvezenych_celkem = 0
celkove_naklady_svoz = 0
celkove_naplneni = 0

#zde zacina simulace
for den in range(0, zivotnost_dni):
            zaplneni = zaplneni + produkce #zaplnovani jednotlivych popelnic - scitani dvou listu
            cena_vysypu = cena_vysypu*(1+eskalace_denni)
            hodinovka_posadka = hodinovka_posadka*(1+eskalace_denni)
            km_naklady = km_naklady*(1+eskalace_denni)

            # print(zaplneni.tolist())

            #DEN SVOZU
            if(den % frekvence_svozu == 0 and den != 0):
                        print("-----------------------------------------------Svozovy den, den:", den, "-------
----------------------------------")
                        celkovy_pocet_svozu = celkovy_pocet_svozu + 1

                        #pomocne promenne k pocitani tras
                        path = tsp()
                        predchozi_nadoba = 0
                        ujete_kilometry = 0
                        cas_na_trase = 0
                        vyvezeny_objem = 0
                        naplneni_auta = 0
                        nadob_vyvezenych = 0
                        pocet_vysypu = 0
                        vyvezene = []

                        #priprava na svoz
                        cas_na_trase = cas_na_trase + priprava_svoz_cas

                        #OBJIZDENI jednotlivych kontejneru
                        for popelnice in path:

                                    #JIZDA - zde se scitaji celkem najete kilometry
                                    ujete_kilometry = ujete_kilometry + vzdalenosti[popelnice][predchozi_nadoba]
                                    cas_na_trase = cas_na_trase + casy[popelnice][predchozi_nadoba]
                                    predchozi_nadoba = popelnice

                                    #VYSYPANI
                                    if(popelnice != 0 and popelnice != 1): #0, 1 jsou depo a tridicka - nevysypavat

                                                #vysypavat jenom zaplnene nadoby
                                                if(zaplneni[popelnice] > limit_zaplneni):
                                                            vyvezeny_objem = vyvezeny_objem + zaplneni[popelnice]
                                                            naplneni_auta = naplneni_auta + zaplneni[popelnice]
                                                            celkove_naplneni      =      celkove_naplneni      +
zaplneni[popelnice]
                                                            zaplneni[popelnice] = 0
                                                            vyvezene.append(popelnice)
                                                            nadob_vyvezenych = nadob_vyvezenych + 1
                                                            cas_na_trase = cas_na_trase + cas_vysypu

                                                            #Limit auta - pokud je auto plne, musi na tridicku
                                                            if(naplneni_auta > limit_vozu):
                                                                        print("Vozidlo  naplneno",  naplneni_auta,
"jedu se vysypat")


```

```
                                                                    #odjedeme vozidlem na tridicku a vratime
se k nynejsi popelnici
                                                                    ujete_kilometry   =   ujete_kilometry   +
vzdalenosti[id_tridicky][popelnice]*2
                                                                    cas_na_trase     =     cas_na_trase     +
casy[id_tridicky][popelnice]*2
                                                                    naplneni_auta = 0
                                                                    pocet_vysypu = pocet_vysypu + 1
                                                                    cas_na_trase     =     cas_na_trase     +
cas_tridicka
                                                                    print("Ujete kilometry", ujete_kilometry)
                                                                    print("-------------")

                        #konec svozu -> nutno vysypat a z tridicky dostat zpet do depa
                        print("naplneni vozidla", naplneni_auta)

                        pocet_vysypu = pocet_vysypu + 1
                        naplneni_auta = 0
                        print("pocet_vysypu", pocet_vysypu)
                        cas_na_trase = cas_na_trase + cas_tridicka

                        #cesta zpet do depa
                        ujete_kilometry = (ujete_kilometry + vzdalenosti[id_depa][popelnice])
                        cas_na_trase = (cas_na_trase + casy[id_depa][popelnice])
                        print("Ujete kilometry za cely svozovy den", ujete_kilometry)
                        print("Cas na trase za cely svozovy den", cas_na_trase)

                        celkovy_pocet_vysypu = celkovy_pocet_vysypu + pocet_vysypu

                        celkova_cena_svozu = celkova_cena_svozu + nadob_vyvezenych * cena_vysypu
                        celkovy_pocet_kilometru = celkovy_pocet_kilometru + ujete_kilometry
                        celkovy_cas = celkovy_cas + cas_na_trase
                        nadob_vyvezenych_celkem = nadob_vyvezenych_celkem + nadob_vyvezenych
                        celkove_naklady_svoz    =    celkove_naklady_svoz    +    (ujete_kilometry*km_naklady    +
cas_na_trase/60*hodinovka_posadka)

                        print("Celkovy objem odpadu", vyvezeny_objem, "m3")
                        print("Celkovy pocet vyvezenych nadob", nadob_vyvezenych)
                        print("Cena svozu", nadob_vyvezenych*cena_vysypu)

            # # #PLOTING
            # try:
            #     graf_nazvy
            # except NameError:
            #     graf_nazvy = np.linspace(3, 62, num=60)
            # plt.cla()
            # plt.title(str(den))
            # plt.ylim([0, 1])
            # plt.bar(graf_nazvy,zaplneni[2:], align='center', color='red')
            # plt.pause(0.01)


    print("==========================================    Shrnuti    za    celou    dobu    projektu    varianty
    1==========================================")
    print("Celkovy pocet kilometru", "%.2f" % round(celkovy_pocet_kilometru, 2), "km")
    print("Celkovy provozni cas", "%.2f" % round(celkovy_cas/60, 2), "hodin")
    print("Celkove naklady na svoz", int(celkove_naklady_svoz), "Kc")
    print("Celkovy pocet svozu", celkovy_pocet_svozu)
    print("Celkovy pocet vysypu", celkovy_pocet_vysypu)
    print("Celkovy pocet vyvezenych nadob", nadob_vyvezenych_celkem)
    print("Prumerne zaplneni vyvazenych nadob", "%.2f" % round(100/1.1*celkove_naplneni/nadob_vyvezenych_celkem, 2),
    "%")
    print("Cena svozu mesto", int(celkova_cena_svozu), "Kc")
```

Příloha 3 – Zdrojový kód simulace svozu varianta 2

```
1 #varianta 2 - plne dynamicky svoz
```

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from functions import tsp,get_distance_matrix

#import dat
df = pd.read_excel('gps_coordinates.xls') # stahnuti seznamu nadob, jejich dynamice zaplnovani a pozic
produkce = np.array(df['zaplnovani_absolut'].tolist())
vzdalenosti_original, casy_original = get_distance_matrix() #ziskani aktualni vzdalenostni a casove matice pomoci
google API

#informace o svoze
cena_vysypu = 250 #Kc
frekvence_svozu = 28 #jak casto (ve dnech) se odpad svazi
zivotnost_projektu = 1 #v letech
limit_vozu = 19 #v m3 - limit vozu 20 m3
cas_vysypu = 1
cas_tridicka = 20
id_depa = 0
id_tridicky = 1
limit_zaplneni = 0.5 #jak plne popelnice se maji vyvazet
priprava_svoz_cas = 40 #min zahrnuje pripravu a taky ukonceni svozu


#svozove naklady
hodinovka_posadka = 800 #Kc/
km_naklady = 15 #Kc/km

#ekonomicke informace
eskalace = 0.03 #rocni eskalace cen - zhruba kopiruje prumernou inflaci

#pomocne promenne
zaplneni = np.zeros(produkce.size)
pocet_nadob = produkce.size-2
celkova_cena_svozu = 0
zivotnost_dni = zivotnost_projektu*365
eskalace_denni = eskalace/365
celkovy_pocet_vysypu = 0
celkovy_pocet_svozu = 0
celkovy_pocet_kilometru = 0
celkovy_cas = 0
nadob_vyvezenych_celkem = 0
celkove_naklady_svoz = 0
celkove_naplneni = 0

#zde zacina simulace
for den in range(0, zivotnost_dni):
            vzdalenosti, casy = vzdalenosti_original, casy_original
            zaplneni = zaplneni + produkce #zaplnovani jednotlivych popelnic - scitani dvou listu
            cena_vysypu = cena_vysypu*(1+eskalace_denni)
            hodinovka_posadka = hodinovka_posadka*(1+eskalace_denni)
            km_naklady = km_naklady*(1+eskalace_denni)


            #KONTROLA ZAPLNENOSTI
            slovnik_zaplnenosti = { i : zaplneni[i] for i in range(2, len(zaplneni) ) }
            serazena_zaplnenost = {}
            seznam_zaplnenosti = sorted(slovnik_zaplnenosti, key=slovnik_zaplnenosti.get)
            seznam_zaplnenosti.reverse()
            for w in seznam_zaplnenosti:
                serazena_zaplnenost[w] = slovnik_zaplnenosti[w]

            nadoby_k_vyvezeni = []
            naplneni_auta = 0

            #DEN SVOZU
            #nejaky kontejner je zaplnen - bude se vyvazet
            if(slovnik_zaplnenosti[seznam_zaplnenosti[0]] > 1):
                        celkovy_pocet_svozu = celkovy_pocet_svozu + 1
```

```
71                      print("----------------------DEN", den,"--------------------------")
72                      #PRIPRAVA SVOZU
73                      #IDENTIFIKACE nadob k vyvezeni
74                      for nadoba in seznam_zaplnenosti:
75                              naplneni_auta = naplneni_auta + slovnik_zaplnenosti[nadoba]
76                              nadoby_k_vyvezeni.append(nadoba)
77
78                              #vsechny nadoby k naplneni auta jsou identifikovane v listu nadoby_k_vyvezeni
79                              if(naplneni_auta > limit_vozu):
80                                      break
81
82                      #VYPOCET DISTANCE MATRIX PRO DNESNI SVOZ
83                      #do listu nadoby_k_vyvezeni jsou pridany depo a tridicka
84                      nadoby_k_vyvezeni.append(id_depa)
85                      nadoby_k_vyvezeni.append(id_tridicky)
86                      smazat_nadoby = (list(set(seznam_zaplnenosti) - set(nadoby_k_vyvezeni)))
87
88                      vzdalenosti = np.delete(vzdalenosti, smazat_nadoby, 0)
89                      vzdalenosti = np.delete(vzdalenosti, smazat_nadoby, 1)
90                      casy = np.delete(casy, smazat_nadoby, 0)
91                      casy = np.delete(casy, smazat_nadoby, 1)
92                      #v promennych vzdalenosti, casy jsou distrance matrix svozu
93                      #KONEC PRIPRAVY SVOZU
94
95                      #ZDE ZACINA
96                      #vypocet nejlepsi trasy
97
98                      path = tsp(vzdalenosti,casy) #problem jsou indexy! tsp vrati jine indexy, protoze distance
99  matrix je mensi. Jak resit?
100                     nadoby_k_vyvezeni = sorted(nadoby_k_vyvezeni)
101                     slovnik_k_vyvezeni = { i : nadoby_k_vyvezeni[i] for i in range(0,
102 len(nadoby_k_vyvezeni) ) }
103
104                     preindexovani_trasy = []
105                     for i in range(0, len(path)):
106                             preindexovani_trasy.append(slovnik_k_vyvezeni[path[i]])
107
108                     path = preindexovani_trasy
109
110                     predchozi_nadoba = 0
111                     ujete_kilometry = 0
112                     cas_na_trase = 0
113                     vyvezeny_objem = 0
114                     naplneni_auta = 0
115                     nadob_vyvezenych = 0
116                     pocet_vysypu = 0
117                     vyvezene = []
118
119                     #priprava na svoz
120                     cas_na_trase = cas_na_trase + priprava_svoz_cas
121
122                     #OBJIZDENI jednotlivych kontejneru
123
124                     for popelnice in path:
125
126                             #zde se scitaji celkem najete kilometry
127                             ujete_kilometry = ujete_kilometry +
128 vzdalenosti_original[popelnice][predchozi_nadoba]
129                             cas_na_trase = cas_na_trase + casy_original[popelnice][predchozi_nadoba]
130                             predchozi_nadoba = popelnice
131                             vyvezene.append(popelnice)
132
133                             #VYSYPANI
134                             if(popelnice != 0 and popelnice != 1): #0, 1 jsou depo a tridicka - nevysypavat
135
136                                     vyvezeny_objem = vyvezeny_objem + zaplneni[popelnice]
137                                     naplneni_auta = naplneni_auta + zaplneni[popelnice]
138                                     celkove_naplneni = celkove_naplneni + zaplneni[popelnice]
139                                     zaplneni[popelnice] = 0
```

```
                                            nadob_vyvezenych = nadob_vyvezenych + 1
                                            cas_na_trase = cas_na_trase + cas_vysypu

                        #konec svozu -> cely svoz byl od depa do tridicky - jeste nutno pricist cas v tridicce
                        print("Naplneni vozidla", naplneni_auta)

                        pocet_vysypu = pocet_vysypu + 1
                        naplneni_auta = 0
                        print("pocet_vysypu", pocet_vysypu)
                        cas_na_trase = cas_na_trase + cas_tridicka

                        #cesta zpet do depa
                        ujete_kilometry = ujete_kilometry + vzdalenosti_original[id_depa][id_tridicky]
                        cas_na_trase = cas_na_trase + casy_original[id_depa][id_tridicky]
                        print("Ujete kilometry za cely svozovy den", ujete_kilometry)
                        print("Cas na trase za cely svozovy den", cas_na_trase)

                        celkovy_pocet_vysypu = celkovy_pocet_vysypu + pocet_vysypu

                        celkova_cena_svozu = celkova_cena_svozu + nadob_vyvezenych * cena_vysypu
                        celkovy_pocet_kilometru = celkovy_pocet_kilometru + ujete_kilometry
                        celkovy_cas = celkovy_cas + cas_na_trase
                        nadob_vyvezenych_celkem = nadob_vyvezenych_celkem + nadob_vyvezenych
                        celkove_naklady_svoz    =    celkove_naklady_svoz    +    (ujete_kilometry*km_naklady    +
cas_na_trase/60*hodinovka_posadka)

                        print("Celkovy objem odpadu", vyvezeny_objem, "m3")
                        print("Celkovy pocet vyvezenych nadob", nadob_vyvezenych)
                        print("Cena svozu", nadob_vyvezenych*cena_vysypu)
                        print("------------------Konec svozu----------------------")

            # #PLOTING
            # try:
            #     graf_nazvy
            # except NameError:
            #     graf_nazvy = np.linspace(3, 62, num=60)
            # plt.cla()
            # plt.title(str(den))
            # plt.ylim([0, 1])
            # plt.bar(graf_nazvy,zaplneni[2:], align='center', color='red')
            # plt.pause(0.001)

print("==========================================    Shrnuti    za    celou    dobu    projektu    varianty
2==========================================")
print("Celkovy pocet kilometru", "%.2f" % round(celkovy_pocet_kilometru, 2), "km")
print("Celkovy provozni cas", "%.2f" % round(celkovy_cas/60, 2), "hodin")
print("Celkove naklady na svoz", int(celkove_naklady_svoz), "Kc")
print("Celkovy pocet svozu", celkovy_pocet_svozu)
print("Celkovy pocet vysypu", celkovy_pocet_vysypu)
print("Celkovy pocet vyvezenych nadob", nadob_vyvezenych_celkem)
print("Prumerne zaplneni vyvazenych nadob", "%.2f" % round(100/1.1*celkove_naplneni/nadob_vyvezenych_celkem, 2),
"%")
print("Cena svozu mesto", int(celkova_cena_svozu), "Kc")
```

Příloha 4 – Zdrojový kód simulace svozu varianta 3

```
#varianta 3 - dynamicky svoz po fixni trase
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from functions import tsp,get_distance_matrix

#import dat
df = pd.read_excel('gps_coordinates.xls') # stahnuti seznamu nadob, jejich dynamice zaplnovani a pozic
produkce = np.array(df['zaplnovani_absolut'].tolist())
vzdalenosti_original, casy_original = get_distance_matrix() #ziskani aktualni vzdalenostni a casove matice pomoci
google API
```

```
12
13 #informace o svoze
14 cena_vysypu = 250 #Kc
15 frekvence_svozu = 28 #jak casto (ve dnech) se odpad svazi
16 zivotnost_projektu = 1 #v letech
17 limit_vozu = 19 #v m3 - limit vozu 20 m3
18 cas_vysypu = 1
19 cas_tridicka = 20
20 id_depa = 0
21 id_tridicky = 1
22 limit_zaplneni = 0.5 #jak plne popelnice se maji vyvazet
23 priprava_svoz_cas = 40 #min zahrnuje pripravu a taky ukonceni svozu
24
25 #svozove naklady
26 hodinovka_posadka = 800 #Kc/
27 km_naklady = 15 #Kc/km
28
29 #ekonomicke informace
30 eskalace = 0.03 #rocni eskalace cen - zhruba kopiruje prumernou inflaci
31
32 #pomocne promenne
33 zaplneni = np.zeros(produkce.size)
34 pocet_nadob = produkce.size-2
35 celkova_cena_svozu = 0
36 zivotnost_dni = zivotnost_projektu*365
37 eskalace_denni = eskalace/365
38 celkovy_pocet_vysypu = 0
39 celkovy_pocet_svozu = 0
40 celkovy_pocet_kilometru = 0
41 celkovy_cas = 0
42 nadob_vyvezenych_celkem = 0
43 celkove_naklady_svoz = 0
44 celkove_naplneni = 0
45
46 #zde zacina simulace
47 for den in range(0, zivotnost_dni):
48             vzdalenosti, casy = vzdalenosti_original, casy_original
49             zaplneni = zaplneni + produkce #zaplnovani jednotlivych popelnic - scitani dvou listu
50             cena_vysypu = cena_vysypu*(1+eskalace_denni)
51             hodinovka_posadka = hodinovka_posadka*(1+eskalace_denni)
52             km_naklady = km_naklady*(1+eskalace_denni)
53
54
55             #KONTROLA ZAPLNENOSTI
56             slovnik_zaplnenosti = { i : zaplneni[i] for i in range(2, len(zaplneni) ) }
57             serazena_zaplnenost = {}
58             seznam_zaplnenosti = sorted(slovnik_zaplnenosti, key=slovnik_zaplnenosti.get)
59             seznam_zaplnenosti.reverse()
60             for w in seznam_zaplnenosti:
61                 serazena_zaplnenost[w] = slovnik_zaplnenosti[w]
62
63             nadoby_k_vyvezeni = []
64             naplneni_auta = 0
65
66             #DEN SVOZU
67             #nejaky kontejner je zaplnen - bude se vyvazet
68             if(slovnik_zaplnenosti[seznam_zaplnenosti[0]] > 1):
69                         print("----------------------DEN", den,"--------------------------")
70                         #PRIPRAVA SVOZU
71                         #IDENTIFIKACE nadob k vyvezeni
72                         for nadoba in seznam_zaplnenosti:
73                                     naplneni_auta = naplneni_auta + slovnik_zaplnenosti[nadoba]
74                                     nadoby_k_vyvezeni.append(nadoba)
75
76                                     #vsechny nadoby k naplneni auta jsou identifikovane v listu nadoby_k_vyvezeni
77                                     if(naplneni_auta > limit_vozu):
78                                                 break
79
80                         #VYPOCET DISTANCE MATRIX PRO DNESNI SVOZ
```

```
                        #do listu nadoby_k_vyvezeni jsou pridany depo a tridicka

                        #ZDE ZACINA
                        #vypocet nejlepsi trasy

                        path = tsp()
                        predchozi_nadoba = 0
                        ujete_kilometry = 0
                        cas_na_trase = 0
                        vyvezeny_objem = 0
                        naplneni_auta = 0
                        nadob_vyvezenych = 0
                        pocet_vysypu = 0
                        vyvezene = []

                        #priprava na svoz
                        cas_na_trase = cas_na_trase + priprava_svoz_cas


                        #OBJIZDENI jednotlivych kontejneru
                        for popelnice in path:
                                    #pokud list nadoby_k_vyvezeni je prazdny, je konec svozu

                                    if not nadoby_k_vyvezeni:
                                                break

                                    #zde se scitaji celkem najete kilometry
                                    ujete_kilometry              =              ujete_kilometry              +
vzdalenosti_original[popelnice][predchozi_nadoba]
                                    cas_na_trase = cas_na_trase + casy_original[popelnice][predchozi_nadoba]
                                    predchozi_nadoba = popelnice
                                    vyvezene.append(popelnice)

                                    #VYSYPANI
                                    if(popelnice != 0 and popelnice != 1): #0, 1 jsou depo a tridicka - nevysypavat
                                                #vysypavat jenom zaplnene nadoby
                                                if(popelnice in nadoby_k_vyvezeni):
                                                            vyvezeny_objem = vyvezeny_objem + zaplneni[popelnice]
                                                            naplneni_auta = naplneni_auta + zaplneni[popelnice]
                                                            celkove_naplneni      =      celkove_naplneni      +
zaplneni[popelnice]
                                                            zaplneni[popelnice] = 0
                                                            vyvezene.append(popelnice)
                                                            nadob_vyvezenych = nadob_vyvezenych + 1
                                                            cas_na_trase = cas_na_trase + cas_vysypu
                                                            nadoby_k_vyvezeni.remove(popelnice)


                        #konec svozu -> nutno zajet do tridicky, vysypat a zpet do depa
                        print("naplneni vozidla", naplneni_auta)

                        #cesta do tridicky
                        ujete_kilometry = (ujete_kilometry + vzdalenosti[id_tridicky][popelnice])
                        cas_na_trase = (cas_na_trase + casy[id_tridicky][popelnice])

                        #v tridicce odpadu
                        pocet_vysypu = pocet_vysypu + 1
                        naplneni_auta = 0
                        print("pocet_vysypu", pocet_vysypu)
                        cas_na_trase = cas_na_trase + cas_tridicka

                        #cesta zpet do depa
                        ujete_kilometry = (ujete_kilometry + vzdalenosti[id_depa][popelnice])
                        cas_na_trase = (cas_na_trase + casy[id_depa][popelnice])
                        print("Ujete kilometry za cely svozovy den", ujete_kilometry)
                        print("Cas na trase za cely svozovy den", cas_na_trase)

                        celkovy_pocet_vysypu = celkovy_pocet_vysypu + pocet_vysypu

```

```
                        celkova_cena_svozu = celkova_cena_svozu + nadob_vyvezenych * cena_vysypu
                        celkovy_pocet_kilometru = celkovy_pocet_kilometru + ujete_kilometry
                        celkovy_cas = celkovy_cas + cas_na_trase
                        nadob_vyvezenych_celkem = nadob_vyvezenych_celkem + nadob_vyvezenych
                        celkove_naklady_svoz    =    celkove_naklady_svoz    +    (ujete_kilometry*km_naklady    +
cas_na_trase/60*hodinovka_posadka)

                        print("Celkovy objem odpadu", vyvezeny_objem, "m3")
                        print("Celkovy pocet vyvezenych nadob", nadob_vyvezenych)
                        print("Cena svozu", nadob_vyvezenych*cena_vysypu)

            #PLOTING
            # try:
            #     graf_nazvy
            # except NameError:
            #     graf_nazvy = np.linspace(3, 62, num=60)
            # plt.cla()
            # plt.title(str(den))
            # plt.ylim([0, 1])
            # plt.bar(graf_nazvy,zaplneni[2:], align='center', color='red')
            # plt.pause(0.001)



print("============================================    Shrnuti    za    celou    dobu    projektu    varianty
3============================================")
print("Celkovy pocet kilometru", "%.2f" % round(celkovy_pocet_kilometru, 2), "km")
print("Celkovy provozni cas", "%.2f" % round(celkovy_cas/60, 2), "hodin")
print("Celkove naklady na svoz", int(celkove_naklady_svoz), "Kc")
print("Celkovy pocet svozu", celkovy_pocet_svozu)
print("Celkovy pocet vysypu", celkovy_pocet_vysypu)
print("Celkovy pocet vyvezenych nadob", nadob_vyvezenych_celkem)
print("Prumerne zaplneni vyvazenych nadob", "%.2f" % round(100/1.1*celkove_naplneni/nadob_vyvezenych_celkem, 2),
"%")
print("Cena svozu mesto", int(celkova_cena_svozu), "Kc")
```

Příloha 5 – Zdrojový kód simulace svozu varianta 4

```
#varianta 4 - svoz v pravidelnem intervalu nadob nad 45% nejkratsi trasou
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from functions import tsp,get_distance_matrix

#import dat
df = pd.read_excel('gps_coordinates.xls') # stahnuti seznamu nadob, jejich dynamice zaplnovani a pozic
produkce = np.array(df['zaplnovani_absolut'].tolist())
vzdalenosti_original, casy_original = get_distance_matrix() #ziskani aktualni vzdalenostni a casove matice pomoci
google API

#informace o svoze
cena_vysypu = 250 #Kc
frekvence_svozu = 28 #jak casto (ve dnech) se odpad svazi
zivotnost_projektu = 1 #v letech
limit_vozu = 19 #v m3 - limit vozu 20 m3
cas_vysypu = 1
cas_tridicka = 20
id_depa = 0
id_tridicky = 1
limit_zaplneni = 0.5 #jak plne popelnice se maji vyvazet
priprava_svoz_cas = 40 #min zahrnuje pripravu a taky ukonceni svozu

#svozove naklady
hodinovka_posadka = 800 #Kc/
km_naklady = 15 #Kc/km

#ekonomicke informace
```

```
eskalace = 0.03 #rocni eskalace cen - zhruba kopiruje prumernou inflaci

#pomocne promenne
zaplneni = np.zeros(produkce.size)
pocet_nadob = produkce.size-2
celkova_cena_svozu = 0
zivotnost_dni = zivotnost_projektu*365
eskalace_denni = eskalace/365
celkovy_pocet_vysypu = 0
celkovy_pocet_svozu = 0
celkovy_pocet_kilometru = 0
celkovy_cas = 0
nadob_vyvezenych_celkem = 0
celkove_naklady_svoz = 0
celkove_naplneni = 0

#zde zacina simulace
for den in range(0, zivotnost_dni):
            vzdalenosti, casy = vzdalenosti_original, casy_original
            zaplneni = zaplneni + produkce #zaplnovani jednotlivych popelnic - scitani dvou listu
            cena_vysypu = cena_vysypu*(1+eskalace_denni)
            hodinovka_posadka = hodinovka_posadka*(1+eskalace_denni)
            km_naklady = km_naklady*(1+eskalace_denni)


            #KONTROLA ZAPLNENOSTI
            slovnik_zaplnenosti = { i : zaplneni[i] for i in range(2, len(zaplneni) ) }
            serazena_zaplnenost = {}
            seznam_zaplnenosti = sorted(slovnik_zaplnenosti, key=slovnik_zaplnenosti.get)
            seznam_zaplnenosti.reverse()
            for w in seznam_zaplnenosti:
                serazena_zaplnenost[w] = slovnik_zaplnenosti[w]

            nadoby_k_vyvezeni = []
            naplneni_auta = 0

            #DEN SVOZU
            #nejaky kontejner je zaplnen - bude se vyvazet
            if(den % frekvence_svozu == 0 and den != 0):
                        # plt.pause(3)
                        celkovy_pocet_svozu = celkovy_pocet_svozu + 1
                        print("----------------------DEN", den,"--------------------------")
                        #PRIPRAVA SVOZU
                        #IDENTIFIKACE nadob k vyvezeni
                        for nadoba in seznam_zaplnenosti:
                                    nadoby_k_vyvezeni.append(nadoba)

                                    #vsechny nadoby k naplneni auta jsou identifikovane v listu nadoby_k_vyvezeni
                                    if(slovnik_zaplnenosti[nadoba] < limit_zaplneni):
                                                break

                        #VYPOCET DISTANCE MATRIX PRO DNESNI SVOZ
                        #do listu nadoby_k_vyvezeni jsou pridany depo a tridicka
                        nadoby_k_vyvezeni.append(id_depa)
                        nadoby_k_vyvezeni.append(id_tridicky)
                        smazat_nadoby = (list(set(seznam_zaplnenosti) - set(nadoby_k_vyvezeni)))

                        vzdalenosti = np.delete(vzdalenosti, smazat_nadoby, 0)
                        vzdalenosti = np.delete(vzdalenosti, smazat_nadoby, 1)
                        casy = np.delete(casy, smazat_nadoby, 0)
                        casy = np.delete(casy, smazat_nadoby, 1)
                        #v promennych vzdalenosti, casy jsou distrance matrix svozu
                        #KONEC PRIPRAVY SVOZU

                        #ZDE ZACINA SAMOTNY SVOZ
                        #vypocet nejlepsi trasy
                        path = tsp(vzdalenosti,casy) #problem jsou indexy! tsp vrati jine indexy, protoze distance
matrix je mensi. Jak resit?
                        nadoby_k_vyvezeni = sorted(nadoby_k_vyvezeni)
```

```
 99                         slovnik_k_vyvezeni    =    {    i    :    nadoby_k_vyvezeni[i]    for    i    in    range(0,
100 len(nadoby_k_vyvezeni) ) }
101
102                         preindexovani_trasy = []
103                         for i in range(0, len(path)):
104                                     preindexovani_trasy.append(slovnik_k_vyvezeni[path[i]])
105
106                         path = preindexovani_trasy
107
108                         predchozi_nadoba = 0
109                         ujete_kilometry = 0
110                         cas_na_trase = 0
111                         vyvezeny_objem = 0
112                         naplneni_auta = 0
113                         nadob_vyvezenych = 0
114                         pocet_vysypu = 0
115                         vyvezene = []
116
117                         #priprava na svoz
118                         cas_na_trase = cas_na_trase + priprava_svoz_cas
119
120                         #OBJIZDENI jednotlivych kontejneru
121                         for popelnice in path:
122
123                                     #zde se scitaji celkem najete kilometry
124                                     ujete_kilometry               =               ujete_kilometry               +
125 vzdalenosti_original[popelnice][predchozi_nadoba]
126                                     cas_na_trase = cas_na_trase + casy_original[popelnice][predchozi_nadoba]
127                                     predchozi_nadoba = popelnice
128                                     vyvezene.append(popelnice)
129
130                                     #VYSYPANI
131                                     if(popelnice != 0 and popelnice != 1): #0, 1 jsou depo a tridicka - nevysypavat
132
133                                                 vyvezeny_objem = vyvezeny_objem + zaplneni[popelnice]
134                                                 naplneni_auta = naplneni_auta + zaplneni[popelnice]
135                                                 celkove_naplneni = celkove_naplneni + zaplneni[popelnice]
136                                                 zaplneni[popelnice] = 0
137                                                 nadob_vyvezenych = nadob_vyvezenych + 1
138                                                 cas_na_trase = cas_na_trase + cas_vysypu
139
140                                                 #Limit auta - pokud je auto plne, musi na tridicku
141                                                 if(naplneni_auta > limit_vozu):
142                                                             print("Vozidlo  naplneno",  naplneni_auta,  "jedu  se
143 vysypat")
144
145                                                             #odjedeme vozidlem na tridicku a vratime se k nynejsi
146 popelnici
147                                                             ujete_kilometry       =       ujete_kilometry       +
148 vzdalenosti_original[id_tridicky][popelnice]*2
149                                                             cas_na_trase         =         cas_na_trase         +
150 casy_original[id_tridicky][popelnice]*2
151                                                             naplneni_auta = 0
152                                                             pocet_vysypu = pocet_vysypu + 1
153                                                             cas_na_trase = cas_na_trase + cas_tridicka
154                                                             print("Ujete kilometry", ujete_kilometry)
155                                                             print("-------------")
156
157                         #konec svozu -> cely svoz byl od depa do tridicky - jeste nutno pricist cas v tridicce
158                         print("Naplneni vozidla", naplneni_auta)
159
160                         pocet_vysypu = pocet_vysypu + 1
161                         naplneni_auta = 0
162                         print("pocet_vysypu", pocet_vysypu)
163                         cas_na_trase = cas_na_trase + cas_tridicka
164
165                         #cesta zpet do depa
166                         ujete_kilometry = ujete_kilometry + vzdalenosti_original[id_depa][id_tridicky]
167                         cas_na_trase = cas_na_trase + casy_original[id_depa][id_tridicky]
```

```
                        print("Ujete kilometry za cely svozovy den", ujete_kilometry)
                        print("Cas na trase za cely svozovy den", cas_na_trase)

                        celkovy_pocet_vysypu = celkovy_pocet_vysypu + pocet_vysypu

                        celkova_cena_svozu = celkova_cena_svozu + nadob_vyvezenych * cena_vysypu
                        celkovy_pocet_kilometru = celkovy_pocet_kilometru + ujete_kilometry
                        celkovy_cas = celkovy_cas + cas_na_trase
                        nadob_vyvezenych_celkem = nadob_vyvezenych_celkem + nadob_vyvezenych
                        celkove_naklady_svoz    =    celkove_naklady_svoz    +    (ujete_kilometry*km_naklady    +
cas_na_trase/60*hodinovka_posadka)

                        print("Celkovy objem odpadu", vyvezeny_objem, "m3")
                        print("Celkovy pocet vyvezenych nadob", nadob_vyvezenych)
                        print("Cena svozu", nadob_vyvezenych*cena_vysypu)
                        print("-----------------Konec svozu--------------------")

            # #PLOTING
            # try:
            #     graf_nazvy
            # except NameError:
            #     graf_nazvy = np.linspace(3, 62, num=60)
            # plt.cla()
            # plt.title(str(den))
            # plt.ylim([0, 1])
            # plt.bar(graf_nazvy,zaplneni[2:], align='center', color='red')
            # plt.pause(0.001)
            # if(den % frekvence_svozu == 0 and den != 0):
            #           plt.pause(3)

    print("============================================    Shrnuti    za    celou    dobu    projektu    varianty
    2=============================================")
    print("Celkovy pocet kilometru", "%.2f" % round(celkovy_pocet_kilometru, 2), "km")
    print("Celkovy provozni cas", "%.2f" % round(celkovy_cas/60, 2), "hodin")
    print("Celkove naklady na svoz", int(celkove_naklady_svoz), "Kc")
    print("Celkovy pocet svozu", celkovy_pocet_svozu)
    print("Celkovy pocet vysypu", celkovy_pocet_vysypu)
    print("Celkovy pocet vyvezenych nadob", nadob_vyvezenych_celkem)
    print("Prumerne zaplneni vyvazenych nadob", "%.2f" % round(100/1.1*celkove_naplneni/nadob_vyvezenych_celkem, 2),
    "%")
    print("Cena svozu mesto", int(celkova_cena_svozu), "Kc")
```

Příloha 6 – Zdrojový kód modulu functions simulace svozu

```
from csv import reader
import numpy as np
from python_tsp.exact import solve_tsp_dynamic_programming
from tsp_solver.greedy import solve_tsp
from python_tsp.heuristics import solve_tsp_simulated_annealing
from csv import reader


def get_distance_matrix():
            with open('vzdalenosti.csv', 'r') as read_obj:
                # pass the file object to reader() to get the reader object
                csv_reader = reader(read_obj)
                # Pass reader object to list() to get a list of lists
                vzdalenosti = list(csv_reader)

            for i in range(0, len(vzdalenosti)):
                        for l in range(0, len(vzdalenosti[i])):
                                    vzdalenosti[i][l] = float(vzdalenosti[i][l])

            with open('casy.csv', 'r') as read_obj:
                # pass the file object to reader() to get the reader object
                csv_reader = reader(read_obj)
                # Pass reader object to list() to get a list of lists
                casy = list(csv_reader)

            for i in range(0, len(casy)):
                        for l in range(0, len(casy[i])):
                                    casy[i][l] = float(casy[i][l])

            return np.array(vzdalenosti), np.array(casy)

```

```
def tsp(vzdalenosti = get_distance_matrix()[0], casy = get_distance_matrix()[1]):


            depo_id = 0
            tridicka_id = 1

            # print("Trasa")
            path = solve_tsp(vzdalenosti, endpoints = (depo_id,tridicka_id))

            # print(path)

            vzdalenost = 0
            cas = 0
            predtim = 0
            for popelnice in path:
                        vzdalenost = vzdalenost + vzdalenosti[popelnice][predtim]
                        cas = cas + casy[popelnice][predtim]
                        predtim = popelnice

            # print()
            # print("Celkova vzdalenost:", "%.2f" % round(vzdalenost, 2), "km")
            # print("Celkovy cas:", "%.2f" % round(cas, 2), "minut")

            return path
```

Příloha 7 – Zdrojový kód modulu get_distance_matrix

```
import pandas as pd

from itertools import permutations
import numpy as np
import googlemaps
import os
from datetime import datetime
import csv
from csv import reader

#Google API key
gmaps = googlemaps.Client(key='xxxxxxxxxxxxxxxxx')

#---------------------------------
#funkce na vybudovani matice casu a vzdalenosti mezi misty
def distance_matrix():

            df = pd.read_excel('gps_coordinates.xls') # can also index sheet by name or fetch all sheets
            mesta = df['GPS'].tolist()
            # print(mylist)

            for i in range (0, len(mesta)):
                        # print(item.replace(" ", ""))
                        mesta[i] = mesta[i].replace(" ", "")
                        mesta[i] = mesta[i].replace("N", "")
                        mesta[i] = mesta[i].replace("E", "")

            print('----Ziskavam data od Google Maps-----')
            seznam = mesta.copy()
            # seznam.insert(0,tridicka)

            vzdalenosti = np.zeros(shape=(len(seznam), len(seznam)))
            casy = np.zeros(shape=(len(seznam), len(seznam)))

            for i in range(len(seznam)):
                        for j in range(len(seznam)):
                                    my_dist = gmaps.distance_matrix(seznam[i],seznam[j])
                                    # print(my_dist)
                                    vzdalenosti[i][j]                                                                                    =
float(my_dist['rows'][0]['elements'][0]['distance']['value']/1000)
                                    casy[i][j] = float(my_dist['rows'][0]['elements'][0]['duration']['value']/60)

                        # exit()
                        print(len(seznam)-i, end=' ')
                        print()
            print('----Distance_matrix hotova-----')
```

```
            print(vzdalenosti)

            return vzdalenosti, casy

vzdalenosti, casy = distance_matrix()

vzdalenosti_list = vzdalenosti.tolist()
casy_list = casy.tolist()

with open("vzdalenosti.csv", "w") as f:
    writer = csv.writer(f)
    writer.writerows(vzdalenosti_list)

with open("casy.csv", "w") as f:
    writer = csv.writer(f)
    writer.writerows(casy_list)
```

Příloha 8 – Zdrojový kód simulace svozu varianty 5

```
#varianta 5 - dynamický svoz nádob s naplněním nad xxx

import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from functions import tsp,get_distance_matrix

#import dat
df = pd.read_excel('gps_coordinates.xls') # stahnuti seznamu nadob, jejich dynamice zaplnovani a pozic
produkce = np.array(df['zaplnovani_absolut'].tolist())
vzdalenosti_original, casy_original = get_distance_matrix() #ziskani aktualni vzdalenostni a casove matice pomoci
google API

#informace o svoze
cena_vysypu = 250 #Kc
frekvence_svozu = 28 #jak casto (ve dnech) se odpad svazi
zivotnost_projektu = 1 #v letech
limit_vozu = 19 #v m3 - limit vozu 20 m3
cas_vysypu = 1
cas_tridicka = 20
id_depa = 0
id_tridicky = 1
limit_zaplneni = 0.9 #jak plne popelnice se maji vyvazet
priprava_svoz_cas = 20 #min zahrnuje pripravu a taky ukonceni svozu


#svozove naklady
hodinovka_posadka = 800 #Kc/
km_naklady = 15 #Kc/km

#ekonomicke informace
eskalace = 0.03 #rocni eskalace cen - zhruba kopiruje prumernou inflaci

#pomocne promenne
zaplneni = np.zeros(produkce.size)
pocet_nadob = produkce.size-2
celkova_cena_svozu = 0
zivotnost_dni = zivotnost_projektu*365
eskalace_denni = eskalace/365
celkovy_pocet_vysypu = 0
celkovy_pocet_svozu = 0
celkovy_pocet_kilometru = 0
celkovy_cas = 0
nadob_vyvezenych_celkem = 0
celkove_naklady_svoz = 0
celkove_naplneni = 0

#zde zacina simulace
for den in range(0, zivotnost_dni):
            vzdalenosti, casy = vzdalenosti_original, casy_original
```

```
            zaplneni = zaplneni + produkce #zaplnovani jednotlivych popelnic - scitani dvou listu
            cena_vysypu = cena_vysypu*(1+eskalace_denni)
            hodinovka_posadka = hodinovka_posadka*(1+eskalace_denni)
            km_naklady = km_naklady*(1+eskalace_denni)


            #KONTROLA ZAPLNENOSTI
            slovnik_zaplnenosti = { i : zaplneni[i] for i in range(2, len(zaplneni) ) }
            serazena_zaplnenost = {}
            seznam_zaplnenosti = sorted(slovnik_zaplnenosti, key=slovnik_zaplnenosti.get)
            seznam_zaplnenosti.reverse()
            for w in seznam_zaplnenosti:
                serazena_zaplnenost[w] = slovnik_zaplnenosti[w]

            nadoby_k_vyvezeni = []
            naplneni_auta = 0

            #DEN SVOZU
            #nejaky kontejner je zaplnen - bude se vyvazet
            if(slovnik_zaplnenosti[seznam_zaplnenosti[0]] > 1):
                        celkovy_pocet_svozu = celkovy_pocet_svozu + 1
                        print("----------------------DEN", den,"--------------------------")
                        #PRIPRAVA SVOZU
                        #IDENTIFIKACE nadob k vyvezeni
                        for nadoba in seznam_zaplnenosti:
                                    naplneni_auta = naplneni_auta + slovnik_zaplnenosti[nadoba]
                                    nadoby_k_vyvezeni.append(nadoba)

                                    #vsechny nadoby k naplneni auta jsou identifikovane v listu nadoby_k_vyvezeni
                                    if(slovnik_zaplnenosti[nadoba] < limit_zaplneni):
                                                break

                        #VYPOCET DISTANCE MATRIX PRO DNESNI SVOZ
                        #do listu nadoby_k_vyvezeni jsou pridany depo a tridicka
                        nadoby_k_vyvezeni.append(id_depa)
                        nadoby_k_vyvezeni.append(id_tridicky)
                        smazat_nadoby = (list(set(seznam_zaplnenosti) - set(nadoby_k_vyvezeni)))

                        vzdalenosti = np.delete(vzdalenosti, smazat_nadoby, 0)
                        vzdalenosti = np.delete(vzdalenosti, smazat_nadoby, 1)
                        casy = np.delete(casy, smazat_nadoby, 0)
                        casy = np.delete(casy, smazat_nadoby, 1)
                        #v promennych vzdalenosti, casy jsou distrance matrix svozu
                        #KONEC PRIPRAVY SVOZU

                        #ZDE ZACINA
                        #vypocet nejlepsi trasy

                        path = tsp(vzdalenosti,casy) #problem jsou indexy! tsp vrati jine indexy, protoze distance
matrix je mensi. Jak resit?
                        nadoby_k_vyvezeni = sorted(nadoby_k_vyvezeni)
                        slovnik_k_vyvezeni = { i : nadoby_k_vyvezeni[i] for i in range(0,
len(nadoby_k_vyvezeni) ) }

                        preindexovani_trasy = []
                        for i in range(0, len(path)):
                                    preindexovani_trasy.append(slovnik_k_vyvezeni[path[i]])

                        path = preindexovani_trasy

                        predchozi_nadoba = 0
                        ujete_kilometry = 0
                        cas_na_trase = 0
                        vyvezeny_objem = 0
                        naplneni_auta = 0
                        nadob_vyvezenych = 0
                        pocet_vysypu = 0
                        vyvezene = []

```

```
120                         #priprava na svoz
121                         cas_na_trase = cas_na_trase + priprava_svoz_cas
122
123                         #OBJIZDENI jednotlivych kontejneru
124
125                         for popelnice in path:
126
127                                     #zde se scitaji celkem najete kilometry
128                                     ujete_kilometry              =              ujete_kilometry              +
129 vzdalenosti_original[popelnice][predchozi_nadoba]
130                                     cas_na_trase = cas_na_trase + casy_original[popelnice][predchozi_nadoba]
131                                     predchozi_nadoba = popelnice
132                                     vyvezene.append(popelnice)
133
134                                     #VYSYPANI
135                                     if(popelnice != 0 and popelnice != 1): #0, 1 jsou depo a tridicka - nevysypavat
136
137                                                 vyvezeny_objem = vyvezeny_objem + zaplneni[popelnice]
138                                                 naplneni_auta = naplneni_auta + zaplneni[popelnice]
139                                                 celkove_naplneni = celkove_naplneni + zaplneni[popelnice]
140                                                 zaplneni[popelnice] = 0
141                                                 nadob_vyvezenych = nadob_vyvezenych + 1
142                                                 cas_na_trase = cas_na_trase + cas_vysypu
143
144                                                 #Limit auta - pokud je auto plne, musi na tridicku
145                                                 if(naplneni_auta > limit_vozu):
146                                                             print("Vozidlo  naplneno",  naplneni_auta,  "jedu  se
147 vysypat")
148
149                                                             #odjedeme vozidlem na tridicku a vratime se k nynejsi
150 popelnici
151                                                             ujete_kilometry       =       ujete_kilometry       +
152 vzdalenosti_original[id_tridicky][popelnice]*2
153                                                             cas_na_trase         =         cas_na_trase         +
154 casy_original[id_tridicky][popelnice]*2
155                                                             naplneni_auta = 0
156                                                             pocet_vysypu = pocet_vysypu + 1
157                                                             cas_na_trase = cas_na_trase + cas_tridicka
158                                                             print("Ujete kilometry", ujete_kilometry)
159                                                             print("-------------")
160
161                         #konec svozu -> cely svoz byl od depa do tridicky - jeste nutno pricist cas v tridicce
162                         print("Naplneni vozidla", naplneni_auta)
163
164                         pocet_vysypu = pocet_vysypu + 1
165                         naplneni_auta = 0
166                         print("pocet_vysypu", pocet_vysypu)
167                         cas_na_trase = cas_na_trase + cas_tridicka
168
169                         #cesta zpet do depa
170                         ujete_kilometry = ujete_kilometry + vzdalenosti_original[id_depa][id_tridicky]
171                         cas_na_trase = cas_na_trase + casy_original[id_depa][id_tridicky]
172                         print("Ujete kilometry za cely svozovy den", ujete_kilometry)
173                         print("Cas na trase za cely svozovy den", cas_na_trase)
174
175                         celkovy_pocet_vysypu = celkovy_pocet_vysypu + pocet_vysypu
176
177                         celkova_cena_svozu = celkova_cena_svozu + nadob_vyvezenych * cena_vysypu
178                         celkovy_pocet_kilometru = celkovy_pocet_kilometru + ujete_kilometry
179                         celkovy_cas = celkovy_cas + cas_na_trase
180                         nadob_vyvezenych_celkem = nadob_vyvezenych_celkem + nadob_vyvezenych
181                         celkove_naklady_svoz    =    celkove_naklady_svoz    +    (ujete_kilometry*km_naklady    +
182 cas_na_trase/60*hodinovka_posadka)
183
184                         print("Celkovy objem odpadu", vyvezeny_objem, "m3")
185                         print("Celkovy pocet vyvezenych nadob", nadob_vyvezenych)
186                         print("Cena svozu", nadob_vyvezenych*cena_vysypu)
187                         print("------------------Konec svozu----------------------")
188
```

```
189             # #PLOTING
190             # try:
191             #     graf_nazvy
192             # except NameError:
193             #     graf_nazvy = np.linspace(3, 62, num=60)
194             # plt.cla()
195             # plt.title(str(den))
196             # plt.ylim([0, 1])
197             # plt.bar(graf_nazvy,zaplneni[2:], align='center', color='red')
198             # plt.pause(0.001)
199
    print("==========================================    Shrnuti    za    celou    dobu    projektu    varianty
    5==========================================")
    print("Celkovy pocet kilometru", "%.2f" % round(celkovy_pocet_kilometru, 2), "km")
    print("Celkovy provozni cas", "%.2f" % round(celkovy_cas/60, 2), "hodin")
    print("Celkove naklady na svoz", int(celkove_naklady_svoz), "Kc")
    print("Celkovy pocet svozu", celkovy_pocet_svozu)
    print("Celkovy pocet vysypu", celkovy_pocet_vysypu)
    print("Celkovy pocet vyvezenych nadob", nadob_vyvezenych_celkem)
    print("Prumerne zaplneni vyvazenych nadob", "%.2f" % round(100/1.1*celkove_naplneni/nadob_vyvezenych_celkem, 2),
    "%")
    print("Cena svozu mesto", int(celkova_cena_svozu), "Kc")
```