TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta strojní

Optimalizace procesů v modelu chytré továrny

Diplomová práce

Studijní program: N0788A270004 Inovační a průmyslové inženýrství
Autor práce: **Bc. Marek Lukášek**
Vedoucí práce: Ing. Radek Votrubec, Ph.D.
Katedra výrobních systémů a automatizace

Liberec 2022

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechanical Engineering

Optimalization of Processes in Smart Factory Model

Master thesis

Study programme: N0788A270004 Innovation and Industrial Engineering
Author: **Bc. Marek Lukášek**
Supervisor: Ing. Radek Votrubec, Ph.D.
Department of Manufacturing Systems and Automation

Liberec 2022

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta strojní

Zadání diplomové práce

Optimalizace procesů v modelu chytré továrny

Jméno a příjmení: **Bc. Marek Lukášek**
Osobní číslo: S20000256
Studijní program: N0788A270004 Inovační a průmyslové inženýrství
Zadávající katedra: Katedra výrobních systémů a automatizace
Akademický rok: **2021/2022**

Zásady pro vypracování:

Cílem diplomové práce je zlepšit řídicí strategie v modelu chytré továrny.
1. Seznamte se se všemi komponenty modelu chytré továrny (zásobníky kuliček, vozíky, server a řídicí aplikace), Arduiny a wifi komunikací.
2. Navrhněte nové algoritmy pro více vozíků, nové trasy a více barev korálků. Optimalizujte trasy vozíků.
3. Navrhněte komponent Internet of Services v modelu továrny.
4. Otestujte nové algoritmy na reálném modelu továrny.

Rozsah grafických prací:	dle potřeby
Rozsah pracovní zprávy:	50
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

[1] BEQUETTE, B. Process control: modeling, design, and simulation. Upper Saddle River, N.J.: Prentice Hall PTR, 2003. ISBN 0133536408.
[2] Arduino Learning: Getting Started with Arduino. In: Arduino [online]. 2014 [cit. 2015-01-09]. available from: http://arduino.cc/en/Guide/HomePage

Vedoucí práce:	Ing. Radek Votrubec, Ph.D. Katedra výrobních systémů a automatizace
Datum zadání práce:	15. listopadu 2021
Předpokládaný termín odevzdání:	15. května 2023

L.S.

prof. Dr. Ing. Petr Lenfeld
děkan

Ing. Petr Zelený, Ph.D.
vedoucí katedry

V Liberci dne 15. listopadu 2021

Prohlášení

Prohlašuji, že svou diplomovou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Jsem si vědom toho, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má diplomová práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

11. ledna 2022 Bc. Marek Lukášek

PODĚKOVÁNÍ

Úvodem bych rád poděkoval panu Ing. Radkovi Votrubcovi, Ph.D. za jeho odborné vedení, cenné rady, odborné připomínky a ochotu při řešení daného tématu diplomové práce. Dále bych také velmi rád poděkoval mé rodině za jejich podporu, kterou mi během studia i mimo něj neustále poskytují.

ABSTRAKT

Tato diplomová práce se zabývá optimalizací procesů v modelu chytré továrny. Hlavním cílem je vytvoření vylepšeného řídícího systému, který komunikuje s jednotlivými komponenty pomocí Wi-Fi. Chytrá továrna je tvořena pomocí vozidla, čtyřmi brankami se zásobníky barevných korálků, serveru a řídící aplikace na mobilním telefonu. Vozidlo se pohybuje po předem vytvořené trajektorii pomocí senzorů, které sledují černou čáru za účelem sesbírání požadovaných barev korálků, které si zákazník definoval při tvorbě objednávky. Přede mnou se komunikačním systémem zabývali dva studenti. Celá struktura je řešena pomocí stavových automatů, pro každý z komponentů je vytvořen automat zvlášť pro testování jednotlivých podmínek postupně. Došlo k vylepšení dosavadních principů v modelu chytré továrny implementací prvků Internet of Services, konkrétně možností simulací servisu vozidla. Byla přidána další barva korálku, zvýšila se maximální délka objednávky na šest položek a zároveň došlo k navržení nových tvarů drah, do které byla vozidlu přiřazena možnost odbočení. Zároveň byl navržen princip rozdělování objednávek mezi dvě vozidla. Na modelu chytré továrny jsou testovány řídící strategie.

KLÍČOVÁ SLOVA

Průmysl 4.0, chytrá továrna, automatizace, Arduino, IoT, Blynk

ABSTRACT

The thesis deals with process optimization in a smart factory model. The main goal is to create an improved control system that communicates with individual components via Wi-Fi. The smart factory consists of a vehicle, four gates with containers filled with colored beads, a server and a control application on a mobile phone. The vehicle moves along a pre-created trajectory using sensors that follow the black line in order to collect the requiered bead colors, that the customer defined while creating the order. Two students were working on the communication system before my own research. The whole structure is solved by use of state machines, for each of the components a machine is created separately, especially for testing individual conditions gradually. The existing principles in the smart factory model have been improved by implementing Internet of Services elements, specifically the option of vehicle service simulations. Another bead color was added, the maximum order length was increased to six items, and at the same time, a new path shapes was designed, to which a turn option was assigned to the vehicle. At the same time, the principle of dividing orders between two vehicles is designed. Control strategies are tested on the smart factory model.

KEYWORDS

Industry 4.0, smart factory, automation, Arduino, IoT, Blynk

OBSAH

SEZNAM POUŽITÝCH ZKRATEK

Zkratka	Význam	Poznámka, originální znění
AI	umělá inteligence	artifical intelligence
AR	rozšířená realita	argumented reality
CPS	kyberneticko-fyzikální systém	cyber-physical system
IDE	vývojové prostředí	integrated development environment
IoT	internet věcí	internet of Things
IIoT	průmyslový internet věcí	industrial internet of things
PLC	programovatelný logický automat	programmable logic controler
USB	univerzální sériová sběrnice	universal serial bus
5G	pátá generace bezdrátových systémů	

1 ÚVOD

Svět se rychle a dynamicky rozvíjí pomocí technologií ve všech různých oblastech. Tento přístup nám umožňuje komunikaci a sdílení našich znalostí z různých oborů mezi sebou. V dnešní době dochází ke zkoumání komunikace, která je aplikovaná na virtuální svět strojů za účelem nastavení přenosu informací mezi samostatnými součástmi systému. To vše se děje za účelem vytvoření dostatečně chytrého systému, který by mohl dělat svou práci sám. V současné době je však snaha o vrácení lidského prvku do výrobního procesu, tedy o vytvoření spolupráce lidí a automatizovaných strojů. Jde jen o vytvoření správné rozvahy nad tím, kde je lidský faktor potřebný a kde nikoliv.

Hlavní myšlenkou této práce je snaha o vytvoření modelu chytré továrny s komunikačním systémem. Komunikační systém pana Jansy obsahoval problém, který spočíval ve zpomalování Arduina v průběhu svého kódu. To bylo způsobeno díky velkému počtu kontrolovaných podmínek obsahujících několik logických funkcí v každé smyčce. To způsobovalo podmínky značně obtížné, tudíž časově náročné. Překonání tohoto problému je řešeno pomocí grafů automatů, které všechny kontrolované podmínky v hlavní smyčce rozdělí do mnoha stavů. To vytváří lepší řešení komunikačního modelu, který eliminuje nevýhody v řízení modelu pana Jansy [21].

Hlavním cílem je vylepšení řídícího systému v modelu chytré továrny. Objednávka je přijata od uživatele a následně odeslána na server modelu. Server umožňuje přenesení vytvořené objednávky do samotného vozidla. Vozidlo pomocí jednotlivých komponentů dokáže objednávku převzít ze serveru a pohybovat se po vymezené dráze tak, aby sbíralo korálky ze zásobníků pomocí komunikace s branami.

V práci pana Sivy Vallinayagama docházelo k optimalizaci komunikačního systému mezi vozidlem a zásobníkem pomocí různých způsobů tak, aby docházelo k jednodušší výměně informací a snížení pohybu nebo rotace vozidla. Současně byl systém vytvořen pro pohyb vozidla po jednoduché dráze ve tvaru oválu a pouze pro jedno vozidlo s definovaným pořadím barevných zásobníků [22].

Model chytré továrny jsem dostal ve stádiu, kdy dokázal jezdit po dráze ve tvaru oválu s přesně definovaným pořadím zásobníků s barevnými korálky. Na začátku diplomové práce naleznete vymezení základních pojmů, představení Průmyslu 4.0 a Arduina. V dalších částech se zabývám popsáním jednotlivých komponentů modelu chytré továrny a především vysvětlením principu nových navržených strategií. Model

chytré továrny je navržen tak, aby na něj bylo možné navázat dalšími diplomovými pracemi, které budou implementovat další nové komponenty průmyslu 4.0 a nové řídící strategie. Všechny programy jsou popsány v anglickém jazyce z toho důvodu, aby byla umožněna spolupráce se studenty ze zahraničí.

Takový model chytré továrny může být jednou z variant řešení pro automatickou manipulaci se zásobníky, manipulaci v továrnách a taktéž by se dal využít ve skladech. Výrobní podniky mohou takový typ automatizované přepravy využít především k usnadnění práce, která je spojena s manipulací materiálu.

2 ZÁKLADNÍ POJMY

Nejdříve je potřebné definovat základními pojmy, které jsou úzce spjaty s tématem této diplomové práce, čímž se zároveň umožňuje pochopit problematiku optimalizace řídících procesů v modelu chytré továrny.

2.1 Systém

Prvním pojmem, se kterým se potřebujeme seznámit v této diplomové práci je **systém**. Systém se obecně vysvětluje jako množina prvků, který má vzájemné vazby, a současně takový systém jako komplex má jistou spojitost s okolím. [1]

Jakýkoliv systém je zpravidla specifikován dvěma elementárními vlastnostmi:

1. Spojitost mezi vnějšími podněty působící na vstup a zároveň výstupními reakcemi, které se vyskytují, charakterizují chování systému a jeho vztah k okolí.
2. Interní spojitosti (vztahy) mezi všemi komponenty systému se popisují strukturou, zejména tedy její organizace a chování.

Tyto dvě elementární vlastnosti mají mezi sebou reciproční spojitost. V tomto případě tedy platí, že specifické chování náleží jisté struktuře a naopak.

2.2 Řízení – jeho charakteristika a druhy

Termínem **řízení** se vysvětluje aktivita, při které se zpracovávají a hodnotí informace o průběhu. Díky tomu se zařízení řídí takovým způsobem, aby se spolehlivě dosáhlo určeného cíle. V tomto případě je implementována zpětná vazba. Jestliže k implementaci zpětné vazby nedojde, nazývá se tento stav pouze ovládání, tedy řízení, kdy není porovnáván uvažovaný výsledek s vlivem okamžitého řízení. [2]

Na vstupní veličinu je možné mít vliv pomocí takzvané **zpětné vazby**. K tomu dochází takovým způsobem, že výsledek předešlého postupu řízení se bere v úvahu v následném kroku řízení celého systému. [1]

Řízení se tedy rozděluje na dva základní typy:

1. Ruční – v tomto případě se jedná o řízení s absencí zpětné vazby (kontroly), což odpovídá pojmu ovládání.
2. Automatické – zde se jedná o řízení, které je uskutečněno prostřednictvím automatického technického přístroje (regulátor). [3]

V případě, že je prováděna takzvaná zpětná vazba, jedná se o takzvané automatické řízení. Ve zkratce dochází ke vzájemnému spojení (výměně dat) mezi systémem a objektem, které jsou řízeny.

Rozlišujeme vyšší a nižší formy automatického řízení:

a. **Regulace** – tímto termínem je označováno řízení s využitím zpětné vazby. Při regulaci dochází ke srovnávání veličiny, která je regulována s žádanou hodnotou. Pomocí velikosti rozdílu dochází k úpravě celého procesu regulace.
b. **Optimální řízení** – pomocí tohoto typu řízení jsme schopni najít a určit vzájemné působení takovým způsobem, že jsme schopni dosáhnout prvotřídního chování systému za definovaných podmínek (například dosažení požadovaných vlastností v nejkratším časovém intervalu s relativně nízkou energií).
c. **Adaptivní řízení** – tato forma nám zabezpečuje optimální postup procesu i v takové situaci, že jsou měněny vlastnosti řízeného objektu. Tímto způsobem je zapříčiněna změna struktury systému.
d. **Učící se systémy** – jedná se o třídu adaptivního systému. Umožňuje nám přijímat a zároveň ukládat do paměti informace. Ty pak je v dalších specifických situacích (totožných nebo podobných) schopen využít.
e. **Umělá inteligence** – jedná se o nejvyšší a nejsofistikovanější formu řízení. Hlavní funkcionalitou je možnost rozpoznání předmětů. Dále umožňuje taktéž identifikovat vzájemné spojitosti a pomocí toho si sám umí vymodelovat okolí. Pomocí těchto možností umí umělá inteligence konat náležité rozhodování, pomocí kterého stále zlepšuje svoji funkci.

V dalších případech jsme schopni rozlišovat automatické řízení na přímé řízení (bez přísunu energie) a nepřímé řízení (s přísunem energie, z toho důvodu dochází k vedení špatného signálu od snímače). Automatické řízení je možné uskutečnit různými způsoby:

a. **Logické řízení** – které pro samotné řízení používá dvouhodnotové veličiny. V tomto případě mohou nastat jen dva stavy (hodnoty 0 a 1, vypnuto/zapnuto). Pro zapisování vztahů mezi veličinami je využívána výroková logika. Výroková logika je používána k záznamu spojitosti mezi jednotlivými veličinami.
b. **Spojité řízení** – nabízí nepřetržitou vazbu řízení mezi vstupem a výstupem. To je zapříčiněno pomocí měření údajů specifikující řídící systém jako pouhé

veličiny, které se neustále mění v čase.

c. **Diskrétní řídící systém** – díky sérii impulzů zaznamenávaných v určitém časovém rozmezí je schopný vytvořit reciproční spojitost mezi vstupem a výstupem. Časové rozmezí, ve kterém je tvořena série impulzů se nazývá taktéž termínem perioda vzorkování.
d. **Nespojité regulátory** – jejich hlavním znakem je ten, že signály vstupu nebo výstupu na sobě navzájem kontinuálně nezáleží.
e. **Fuzzy řízení** – k jeho aplikaci dochází v případě, že známe specifickou hodnotu výstupu i v případě, že neznáme spojitosti mezi vstupy a výstupy. Nejčastěji se tento typ řízení používá u systémů, které nejsme schopni popsat natož upřesnit.

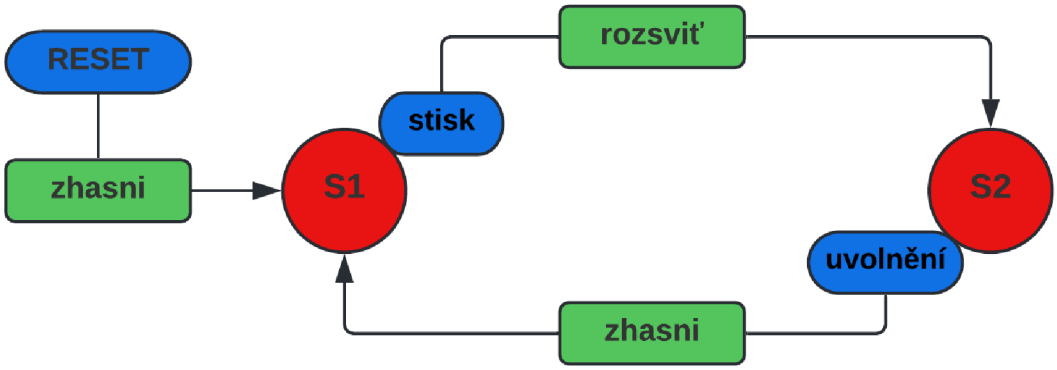
2.3 Stavový automat

Stavovým automatem se označuje model, který je určený pro programování a jeho hlavní výhodou je jeho vysoká sofistikovanost a přehlednost. Takový automat je následně navržen tak, aby došlo k rozdělení akcí programu (zařízení) do stavů ve kterých se může vyskytovat. Stanovením podmínek můžete definovat kdy a jak dochází k přechodu z jednoho stavu do druhého. Při psaní stavového automatu je tedy docílena přesná specifikace toho, co zařízení má dělat a jak reagovat. [4]

Určité chování stavového automatu můžeme zobrazit diagramem, jehož stavba se skládá ze tří částí:

1. **Stav** – okamžik, při kterém konkrétní automat vyčkává na podnět, pomocí kterého je schopen skočit do následujícího stavu (značka kruh).
2. **Přechod** – je uskutečněn pouze v případě, že jsou splněny stanovené podmínky (značka ovál u konkrétního stavu). Podle významu jde o případ, který je mezi jednotlivými stavy a pro přehlednost v postupu je tvořen pomocí orientované šipky.
3. **Akce** – jsou provedeny v okamžiku, kdy dochází k samotnému skoku mezi konkrétními stavy (značka obdélník u orientovaných šipek).

Každý automat musí mít vždy konkrétní počet stavů, které při zahájení chodu fungují v nekončící periodě. Při opětovném zapnutí dochází k resetování automatu, tím pádem začíná opět od prvního stavu.



Obr. 1: Automat pro ovládání světla [5]

3 PRŮMYSL 4.0

Jestliže přemýšlíme o průmyslové revoluci, někteří z nás si vybaví změny ve výrobě zapříčiněné počátkem využívání strojů v osmnáctém století. Ostatní se mohou domnívat, že průmyslová revoluce je pokračující realita, ve které žijeme do současnosti. Obojí tvrzení je správné. Konkrétně termínem průmysl 4.0 se označuje současný trend digitalizace, se kterou je úzce spjata automatizace výroby. Pro tento termín neexistuje žádná norma ani jednoznačná definice, ale i přesto je tento pojem používán nejenom odbornou veřejností. [6]

Současný pokrok v technologiích reprezentuje právě čtvrtá průmyslová revoluce, a proto je dobré si ucelit celý soubor technických pokroků (průmysl 1.0 až 4.0).

3.1 Rozvoj průmyslu

První průmyslová revoluce (taktéž průmysl 1.0) zapříčinila přesun ekonomiky z dosavadního světa řemesel a zemědělství do světa strojů. Prvopočátky sahají zhruba do šedesátých let osmnáctého století se vznikem ve Velké Británii a následnou expanzí do dalších zemí světa. Tato revoluce zavádí parní stroj do výrobního procesu, což způsobuje industrializaci zemědělství, dochází ke vzniku větších továren a významně zjednodušuje výrobu. Zde jsou uvedeny některé pokroky ve výrobě:

- nové zdroje energie (uhlí)
- nové suroviny (ocel)
- vznik dělby práce (dělnická specializace)
- zlepšení dopravy a komunikace

K druhé průmyslové revoluci (průmyslu 2.0) došlo koncem devatenáctého a počátkem dvacátého století, která vedla k významným vynálezům. Stroje hrají stále důležitější roli v průmyslu, rozšiřuje se výroba a využití elektřiny, ropy a oceli. Díky první světové válce dochází k revoluci ve výrobě způsobenou zaváděním takzvané organizované výroby pomocí montážních linek, který pomáhá snadno identifikovat jakýkoliv problém při výrobě a snadno jej vyřešit. Takový způsob výroby se v pozdějších obdobích značně modernizuje.

Třetí průmyslová revoluce (průmysl 3.0) se odehrávala od sedmdesátých let dvacátého století a v podstatě se jedná o digitální revoluci. Mezi hlavní znaky bezpochyby patří elektronické zařízení a systémy informačních technologií což umožnilo rychlý přechod na digitálnější systémy ve výrobě. Dochází k nasazení počítačů přímo na výrobní linku pro zvýšení výkonu a přesnosti ve všech fázích. Mnoho úkolů, které dříve dokončovali lidé přebírá automatizační software. V tomto období je vyroben první programovatelný logický automat (PLC), dochází ke vzniku prvních robotů a umělé inteligence.

Čtvrtá průmyslová revoluce (průmysl 4.0) a inteligentní výroba s ní spjatá se rozvinula v posledních několika desetiletích. Vyznačuje se především kybernetico-fyzikálními systémy (CPS), rozvojem automatizace v průmyslových odvětvích díky strojového učení a analýzy dat v reálném čase samostatnými stroji. Dále také umělou inteligencí (AI), autonomně řídících vozidlech a internetu věcí (IoT), který výrazně urychlil a rozšířil většinu změn po celém světě. Dále v průmyslu 4.0 dochází mezi stroji k přenosu a komunikaci dat bez lidského zásahu, ale také se zaměřuje na vztahy mezi dodavateli, výrobou a samotnými zákazníky. Tím tedy dochází k úpravě technik řízení dodavatelského řetězce v průmyslu, které se realizují pomocí nástrojů, které to prakticky umožňují díky této revoluci.

Obr. 2: Průmyslové revoluce ve zkratce [7]

3.2 Vznik průmyslu 4.0

Průmysl 4.0 vzniká v roce 2011, ale samotná koncepce byla představena až v roce 2013. Samotný projekt je iniciován společností Siemens za podpory německé vlády pod názvem Industry 4.0 a jednou z její hlavních myšlenek je vytvoření takzvané chytré továrny („Smart Factory“), které budou využívat kyberneticko-fyzikální systémy. Ty umožní převzetí jednoduchých a opakujících se činností, které do té doby museli vykonávat lidé. [8]

V dalším případě je nutné přijmout to, že průmysl 4.0 není pouze změna stylu výroby v průmyslu, ale bude se týkat taktéž elektrotechniky, zdravotnictví a dalších oborů.

3.3 Koncepce průmyslu 4.0

Chytrá továrna v průmyslu 4.0 nemá žádné jasné formulace o tom, jak by v dnešní době měla vypadat, ale můžeme se setkat pouze s výčtem procesů, které by v továrně měly být a taktéž zmínky o tom, na jaké úrovni by měly pracovat. Pro usnadnění aplikovatelnosti by nastavení všech procesů v chytré továrně mělo být ve standartní formě. Jako hlavní proces se označuje propojení dodavatelského řetězce se servisním systémem pomocí sítě.

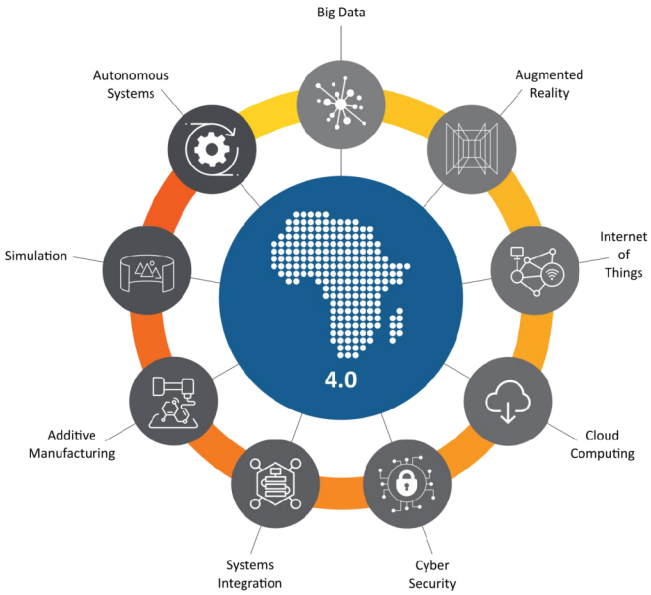
Mezi hlavní cíle koncepce průmyslu 4.0 patří: [9]

a. **interoperabilita** – jedná se o schopnost komunikace všech prvků výrobního systému
b. **decentralizace** – jde o schopnost strojů činit jednoduchá a rutinní rozhodnutí autonomně za účelem docílení co největší optimalizace výroby
c. **virtualizace** – umožňuje vytvářet virtuální modely (dvojče) všech fyzických objektů (využívání informací z čidel strojů na modelu chytré továrny), výroba pak probíhá zároveň ve fyzickém i virtuálním modelu
d. **reálný čas** – odezvy zařízení pohybující se v desítkách milisekund
e. **orientace na služby** – nakupované i poskytované
f. **modularita** – umožní reakci na změny výroby v chytré továrně tím, že dojde k nahrazení nebo rozšíření jednotlivé moduly

3.4 Hlavní pilíře průmyslu 4.0

Průmysl 4.0 zahrnuje širokou škálu technologií. Pro pochopení toho, v čem je průmysl 4.0 unikátní, je potřeba si uvědomit kvality, které tyto různé koncepty a technologie přináší do výroby, a zejména také zcela měnící se způsob výroby zboží a jeho vybírání. [10]

Mezi nástroje, které průmysl 4.0 využívá, spadá devět hlavních technologií. Konkrétně se tedy jedná o internet věcí (IoT), rozšířenou realitu (AR), aditivní výrobu, autonomní roboty, simulace, systémová integrace, big data, kybernetická bezpečnost a cloud computing. Všechny tyto jmenované technologie zajišťují chytrou výrobu, bezdrátovou Wi-Fi komunikaci ve skladech pro manipulaci s produkty a průmyslových odvětvích. [10]



Obr. 3: Hlavní pilíře průmyslu 4.0 [11]

3.4.1 Internet věcí (IoT)

Základní princip IoT spočívá ve výměně dat mezi fyzickými zařízeními vybavenými senzory, elektronikou a hlavně připojením k síti, což umožňuje vzájemnou výměnu dat mezi všemi propojenými zařízeními. [12]

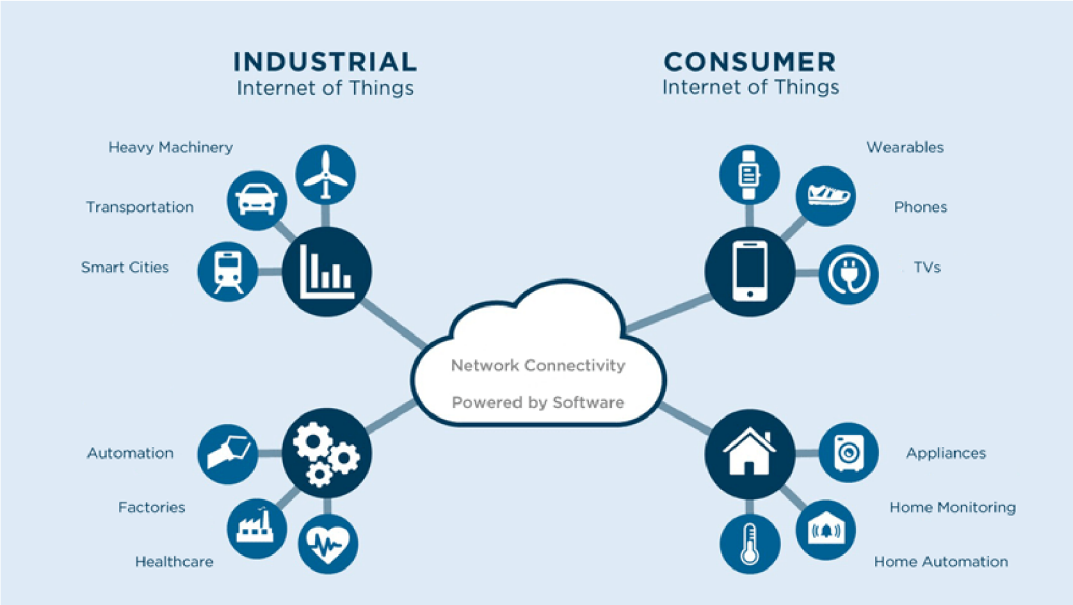
Základem je tedy shromažďování, přenášení a analýza dat, které je realizováno pomocí internetového připojení. Díky pokrokům v celulárních tegnologiích (5G[1]) je umožněno efektivnější využití rádiového frekvenčního spektra. Můžeme tedy vytvářet aplikace, které řeší složitější problémy. [10]

[1] telekomunikační standard mobilní sítě (pátá generace bezdrátových systémů)

3.4.2 Průmyslový internet věcí (IIoT)

Průmyslový internet věcí rozšiřuje použití IoT v průmyslových odvětvích a aplikacích. Označuje propojené stroje, zařízení nebo procesy, propojené díky datovým komunikačním systémům, pro zjednodušení výměny a využívání dat mezi lidmi a stroji. Mezi tyto přístroje obvykle patří senzory, které snímají a shromažďují data v off – line nebo v cloudové databázi. To umožňuje podnikům lépe identifikovat možnosti, jak zlepšit produktivitu, spolehlivost výrobního procesu. IIoT a IoT jsou velmi podobné, ale taktéž velmi odlišné z hlediska služeb a požadavků.

Hlavním prvkem, který odlišuje IIoT je především propojení informačních a provozních technologií. Termínem provozní technologie je označováno propojení průmyslových řídících systémů a provozních procesů. Díky konvergenci těchto dvou technologií dochází k lepší integraci zejména v automatizaci a optimalizaci. [13]



Obr. 4: Rozdíly IIoT a IoT [14]

3.4.3 Kyberneticko-fyzikální systémy (CPS)

Kyberneticko-fyzikální systém je jakýkoliv mechanický proces, který je řízen automaticky pomocí softwaru. Senzory z mechanických komponentů spouští programy, určující principy řízení zařízení a strojů. CPS tedy spojují dohromady strojové učení, robotické systémy a IoT. Umožňují reakci na změny v prostředí, práci v různých prostorech, čímž nabízí snadnou adaptaci na různé potřeby výrobce. [10]

Kyberneticko-fyzikální systémy mají zpětnou vazbu, díky které je umožněna vlastní korekce. V CPS jsou také různé akční členy a senzory, které hrají hlavní roli

v rozvoji mnoha oblastí včetně zdravotnictví, dopravy a životního prostředí.

3.4.4 Big data

„Big data“ jsou v podstatě velké nebo složité datové sady, takže je velmi složité je včas zpracovat i pomocí specializovaného softwaru. Je tedy potřebné najít nejlepší způsoby, jak tyto data ukládat, analyzovat a zejména používat. Díky tomu jsou cenným nástrojem pro pochopení chování jednotlivých uživatelů, a proto výrobci vyvíjí systémy pro účinné vyhledání cenných informací. [10]

3.4.5 Cloud computing

Termínem „cloud computing“ se rozumí zpřístupnění zdrojů, mezi kterými jsou ukládání dat a výpočetní výkon. Jinými slovy se jedná o propůjčení výpočetního výkonu serverů danému uživateli. Tato technologie umožní výrobcům platit za zdroje které potřebují v době potřeby. Firmy, které poskytují cloudové služby nabízejí tři modely cloud computingu: [10]

- **software jako služba** – zákazník platí poskytovateli za využívání cloudového softwaru
- **platforma jako služba** – zákazník platí poskytovateli za výpočetní výkon a infrastrukturu, které potřebuje pro vlastní aplikace
- **infrastruktura jako služba** – zákazník platí poskytovateli za servery, uložiště a datová centra, která potřebuje k usnadnění platformy

3.5 Průmysl 5.0 – když lidé pracují se stroji

Vývojový stupeň v automatizaci průmyslové výroby zatím nemá zcela jasnou definici, ale jednoduše řečeno se jedná o vizi návratu člověka do výrobního procesu. Průmysl 4.0 reprezentuje proces plné automatizace výroby prostřednictvím technologií, které jsou řízeny umělou inteligencí s vlastnostmi umožňující samodiagnostiku, průmysl 5.0 vrací do výrobního procesu lidský prvek. Dá se to také shrnout slovy, kdy se jedná o stav, kde společně ve výrobním procesu spolupracují lidé a automatizované stroje (kolaborativní roboti).

Pokud bude dodržena správná rozvaha nad tím, ve kterých částech výroby je lidský prvek žádoucí nebo nezbytný, se koncepty průmyslu 4.0 a 5.0 mohou ideálně doplňovat. O tom vypovídá existence tří základních směrů o řešení vývoje spolupráce lidí se stroji [15].

3.5.1 Hlavní směry průmyslu 5.0

Jedním ze tří směrů je již v dnešní době široce využíván, a to tam, kde lidská práce nelze s přiměřenými náklady a složitostí nahradit. Hlavním příkladem je především automobilová produkce, kdy je možné strojově zajistit velkou část úkonů ve výrobě i montáži. V případě zapojování kabelových svazků a při podobných případech je nutností určitá motorika a šikovnost, která zůstává lidskou doménou.

Druhým ze tří směrů je výroba s vysokým počtem individuálních úprav na přání zákazníka, které bez lidské kreativity nelze provést. Příkladem může být výroba zakázkových produktů, při kterých se využívají polotovary, ale finální zpracování vyžaduje práci kvalifikovaných a řemeslně zručných lidí.

Třetí směr představuje technologický posun, který spočívá ve spolupráci lidí a automatizovaných strojů na jednom pracovišti. Spolupráce spočívá v tom, že stroje provádí fyzicky náročné úkony, opakované pracovní činnosti nebo případně nebezpečnou práci. Zároveň budou průmysloví roboti pracovat s lidmi bez jakýchkoliv ochranných prvků ve formě klecí, které zabezpečují operační prostor robotu [15].

3.5.2 Těžká práce pro roboty

Pro docílení spolupráce lidí a robotů, musí samotné technologické řešení splňovat především vysoké bezpečnostní nároky. Celý pracovní proces, tedy rychlost, reakční doby a jiné parametry musí být přizpůsobeny lidským možnostem, jelikož je jasné, že reakční doba člověka a robota jsou zcela odlišné. Z tohoto důvodu nelze zvyšovat rychlost práce nad samotné limity lidí.

Pro zabezpečení efektivní spolupráce, je potřeba myslet na způsob ovládání, respektive interakci lidí s kolaborativními roboty. Prvním způsobem bude nejspíše kognitivní způsob řízení robotů, který spočívá v učení strojů od lidí, tedy ukázané postupy si zapamatují a budou je schopné opakovat. Dalším a komfortnějším způsobem by byla hlasová interakce se stroji. K zajištění toho, aby robot rozuměl naším hlasovým povelům a pochopil, jaký úkon chceme udělat, vede dlouhá cesta vývoje umělé inteligence a strojového učení. Bezpečná spolupráce s roboty na každodenní bázi vyžaduje lepší úroveň kvality interakce, než jsou současné hlasové asistenční služby.

3.5.3 Automatizace přívětivá k lidem

Samotná vize Průmyslu 5.0 se nikterak nevylučuje s koncepty plné automatizace

v rámci Průmyslu 4.0. Hlavní výhodou je zde doplnění o lidský prvek na vhodných místech výrobního procesu a taktéž může zahnat obavy o pracovní místa zaměstnanců, jelikož je postavena na kreativitě a zručnosti lidí, kterým stroje pouze uleví od těžké práce. Pro plnou automatizaci pak existuje mnoho výrobních odvětví, kde je lidský faktor zcela nežádoucí (typicky ve farmacii, chemickém průmyslu atd.). Kolaborativní roboti mohou spolupracovat taktéž i s dalšími stroji a je jen otázkou času, kdy se rozšíří do dalších oblastí průmyslu a podnikání. Záleží pouze na tom, co budeme schopni tyto stoje naučit [15].

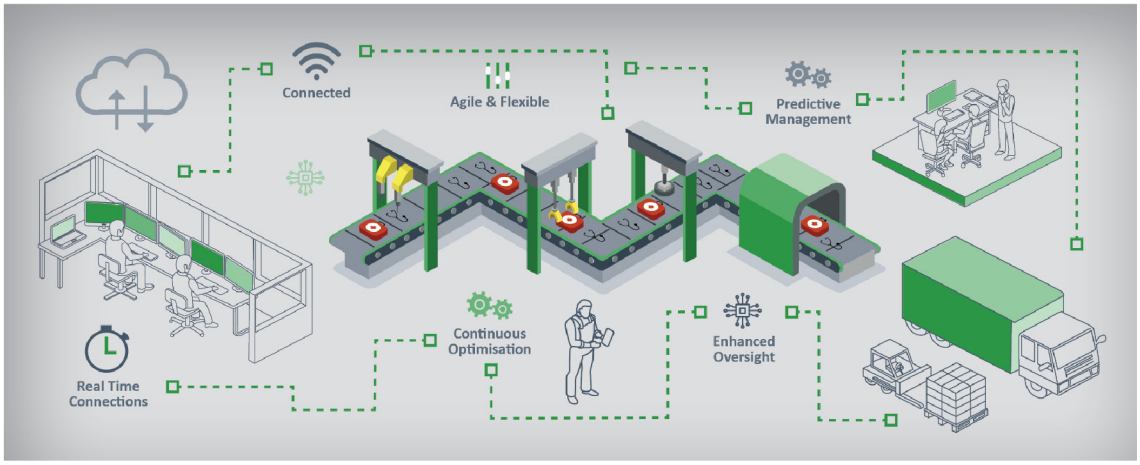
3.6 Chytrá továrna (Smart Factory)

Chytrá továrna lze označit za koncept, který je odvozený z IIoT[2], a vyjadřuje konečný cíl digitalizace ve výrobě. Způsobem, kterým se tento termín nejčastěji využívá, je „Smart Factory“ tedy vysoce digitalizovaná továrna. Hlavním předpokladem je realizace výrobního prostředí jako plně automatizovanou a inteligentní síť systémů, která řídí stroje, zařízení a logistiku ve výrobním závodě bez lidského zásahu. Všechny tyto věci mohou být realizovány díky výměně dat mezi výrobními stroji a nástroji, ale taktéž mezi všemi složkami v řetězci výrobních technologií. Díky tomu je kladen velký důraz na strojové učení, které efektivněji provádí operace a umožní více úspor, než kdyby výrobní procesy zůstaly pod lidským dohledem.

Smart Factory tedy nepřetržitě shromažďuje a sdílí data pomocí připojených strojů, zařízení a výrobních systémů. Následně tato data mohou být využita samooptimalizačními zařízeními nebo k aktivnímu řešení problémů, reakci na nové požadavky a optimalizaci výrobních procesů v rámci celé organizace. Inteligentní výrobní postupy jsou komplexní právě díky technologiím jako AI[3], Big data, Cloud computing a IoT[4]. Vytvořením vazby mezi fyzickým a digitálním světem může Smart Factory monitorovat celý výrobní proces včetně dodavatelského řetězce a jednotlivých operátorů v továrně. [17]

Jestliže je Smart factory plně realizovaná, využívá právě plně integrovaných a spolupracujících výrobních systémů z důvodu zaručení flexibilních, přizpůsobivých a optimalizovaných operací.

[2] IIoT – Industrial Internet of Things (viz kapitola 3.4.2)
[3] AI – Artifical Intelligence (obor, který se zabývá stroji s inteligentním chováním)
[4] IoT – Internet of Things (viz kapitola 3.4.1)



Obr. 5: Smart Factory [16]

3.6.1 Výhody Smart Factory

Hlavní výhodou Smart Factory je bezesporu optimalizace efektivity a produktivity díky rozšíření možností výrobních zařízení a lidí. Vytvořením interaktivního výrobního procesu prostřednictvím shromažďování dat, pomáhají rozhodovacím procesům. Pomocí neustálého zlepšování produktivity mohou chytré továrny snížit náklady, zkrátit prostoje a minimalizovat odpad.

3.6.2 Úrovně Smart Factory

Úrovně Smart Factory se dělí do základních čtyř datových struktur. Ty pomáhají identifikovat kroky na cestě tvorby chytré továrny a taktéž určovat rozhraní, abyste byly schopni pokročit na další úroveň. [17]

1. **úroveň: dostupná data**

   Tato úroveň je současný stav většiny továren, data jsou k dispozici, ale nejsou přístupná. Analýza a třídění dat se provádí ručně a je velmi časově náročná, což je pro zlepšení výroby neefektivní.

2. **úroveň: přístupná data**

   V tomto stavu jsou údaje a data strukturovaně uspořádány a seřazeny na jednom místě. Další systémy pomáhají data správně vizualizovat a zobrazovat řídící panely. Chytrá továrna může provádět analýzy i za cenu vyžadování určitého času.

3. **úroveň: aktivní data**

   Aktivní data jsou taková data, která mohou provádět analýzu využitím umělé inteligence. Systém dokáže identifikovat klíčové problémy a odchylky, díky kterým dokáže předpovědět selhání a informovat lidi ve správný čas.

4.úroveň: data orientovaná na akci

V této fázi díky strojovému učení a umělé inteligenci dojde ke generování možných řešení problémů, které byly zachyceny v předchozích fázích. Všechny výrobní stoje jsou připojeny k takovému systému, díky kterému mohou změny provádět bez lidského zásahu. Sbírání dat, identifikace problémů a vytváření řešení probíhá s minimálním nebo žádným lidským vstupem.

4 ARDUINO

Arduino je podle oficiálních stránek [18] open-source[5] elektronická platforma, která je založená na snadno použitelném hardwaru a softwaru. Mezi portfolio výrobků společnosti arduino patří široká škála jednodeskových počítačů, které jsou založené na mikroprocesorech s různými výkony. Dále také nabízí komponenty, které umožňují vytvářet všestranné zařízení.

Desky arduino jsou tedy schopné číst vstupy a přeměňovat je na výstupy. Svojí desce tedy může říct, co má dělat pomocí instrukcí, které pošlete na mikrokontrolér umístěný na desce. Používá se programovací jazyk Arduino Wiring, který lze rozdělit na tři části: funkce, hodnoty (proměnné a konstanty) a struktura. Vše se provádí v dostupném softwaru Arduino (IDE).

Postupem času se stalo Arduino mozkem tisíce projektů, ať už se jedná o každodenní předměty, ale i vědecké přístroje. Okolo Arduina se shromáždila celosvětová komunita tvůrců, studentů i nadšenců hlavně z toho důvodu, protože se jedná o platformu s otevřeným zdrojovým kódem. Příspěvky pomohly k velkému množství dostupných znalostí, které slouží jako pomoc nováčkům, ale taktéž odborníkům.

Arduino vzniklo v italském Ivrea Interaction Design Institute jako nástroj na prototypování, využívaný zejména studenty bez znalostí v elektronice a programování. Postupem času se Arduino dostalo do širší komunity, tím se začala přizpůsobovat novým potřebám a výzvám. Společnost Arduino začalo svoje portfolio produktů směřovat k výrobkům pro aplikace IoT, 3D tisk a vestavěná prostředí.

Díky celosvětovému rozšíření se bohužel tyto výrobky začaly masivně napodobovat. Což tedy znamená, že by při výběru a nákupu daných komponentů měl zákazník ověřit pravost součástky.

[5] S volně přístupným kódem, který je možné upravovat a redistribuovat.

Nejlepší a nejsnadnější cesta je nakupovat komponenty na oficiálních nebo ověřených stránkách. Důležité je taktéž to, že produkty Arduino jsou určené pouze pro podporu vzdělání a výzkumu, nesmějí tedy být využívány v průmyslové sféře.

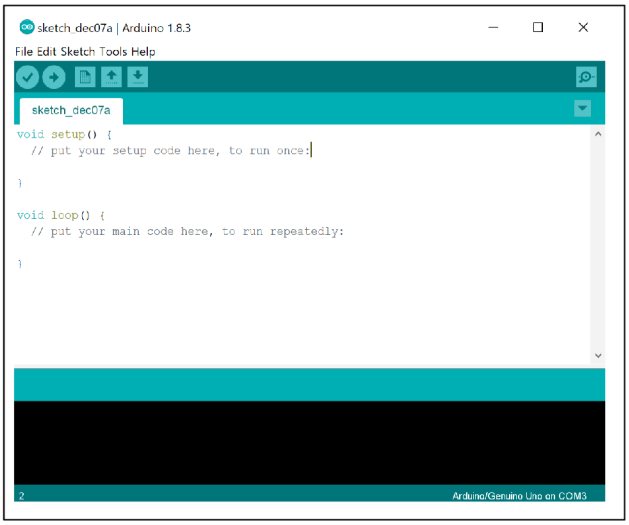
4.1 Programování Arduina

Pro vytváření programů, které chceme, aby mikrokontrolér vykonával lze použít vývojové prostředí Arduino software IDE[6]. Po nainstalování programu se musí v nastavení vybrat použitá vývojová deska a vybrat port, pomocí kterého je deska připojena. Dále už jen stačí napsat program, přeložit do kódu pro daný mikrokontrolér a pomocí USB konektoru nahrát a spustit. Uložené soubory mají koncovku „.ino“.

Programovací jazyk pro mikrokontroléry Arduino vznikl z jazyka Wiring a je založený na jazyku C++. Dále byl tento jazyk pro Arduino zjednodušen díky absenci objektového programování. Pro úspěšné napsání programu není třeba ani obsáhlé deklarace datových typů, inicializace objektů nebo definice grafického prostředí.

Základní části programu pro Arduino:

- deklarace globárních proměnných, načtení extérních knihoven a definice konstant
- **funkce setup ()** – část kódu, která se má proběhnout pouze jednou
- **funkce loop ()** – vykonává hlavní nekonečnou smyčku programu (například odečítá nebo odesílá data ze senzorů)
- případně definice zbývajících funkcí programu



Obr. 6: Prostředí Arduino IDE [zdroj: vlastní]

[6] IDE je anglická zkratka pro vývojové prostředí (integrated development environment).

5 KOMPONENTY SMART FACTORY

Výukový model chytré továrny se skládá ze čtyř nebo více bran, které slouží jako zásobníky korálků různých barev a dvou automaticky naváděných vozidel. Komunikaci mezi prvky celé chytré továrny zabezpečuje server. Tento model chytré továrny představuje jednoduchou výrobní linku. Ve standartní formě, tj. v první verzi dráhy, jsou brány seřazeny za sebou s předem definovanou posloupností barev. Vozidlo se pohybuje po dráze mezi jednotlivými bránami za účelem sesbírání korálků podle vytvořené objednávky, která se vytváří v aplikaci Blynk na mobilním telefonu. Navrhovat lze různé strategie pohybu mezi zásobníky a další verze dráhy.

5.1 Části systému Smart Factory

Chytrá továrna má několik částí, které mezi sebou musí vzájemně komunikovat, čímž tvoří automatizovaný pracovní postup přepravy. Automatizované vozidlo, které se pohybuje po předem definovaném tvaru dráhy, čtyři brány se zásobníky jsou na trase tohoto vozidla, kde vozidlo sbírá korálky dle vytvořené objednávky. Server zde hraje důležitou roli v komunikaci mezi jednotlivými komponenty chytré továrny, tedy při přijímání a distribuci informací mezi zařízeními. Jak jsem se zmiňoval už dříve, uživatelské rozhraní tohoto systému zabezpečuje aplikace Blynk v chytrém telefonu, která je společně propojena pomocí Wi-Fi[7].

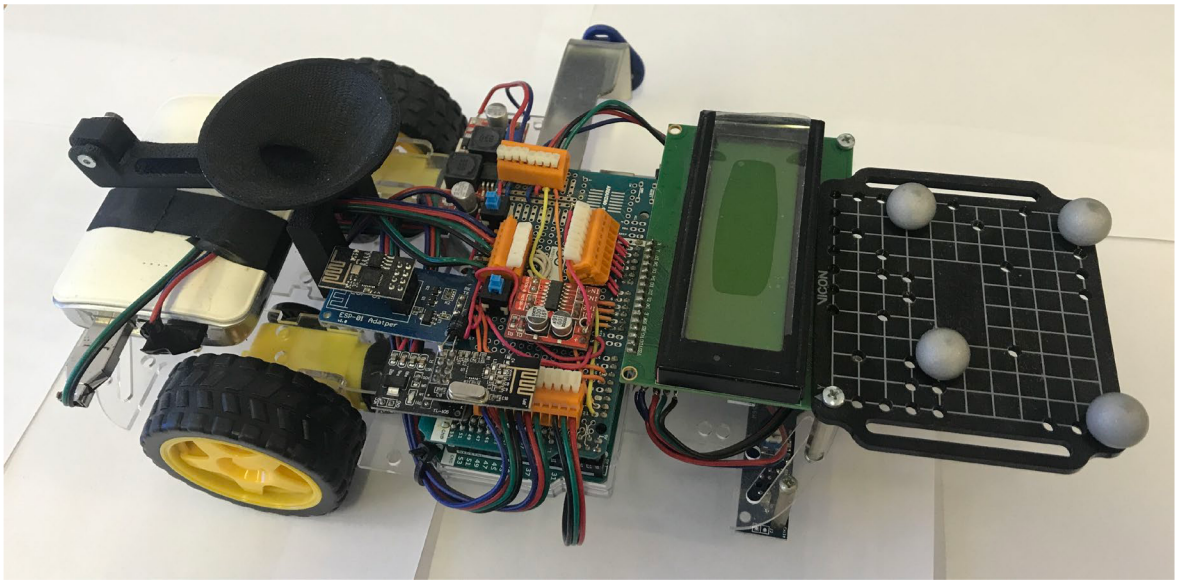
5.2 Automaticky naváděné vozidlo

Jedním z hlavních prvků modelu chytré továrny je vozidlo (obr.7), které se používá pro sběr korálků dané barvy. Celá konstrukce je navržena tak, aby jezdila po předem definovaném tvaru dráhy a to tak, že sleduje černou čáru. Vozidlo používá ke sledování černé čáry IR senzory, o pohyb se stará dvojice pohonů kol. Vozidlo je vyrobeno tak, aby bylo funkční na základě obvodů, nainstalovaných a naprogramovaných na desce plošných spojů Arduino Mega 2560.

Každé vozidlo má ovladač dvojitého H-můstku, který se stará o poháněnní dvou DC motorů, tedy každý pohání jedno kolo. Vozidlo je také osazeno pětikanálovým modulem senzoru sledování linky, který je určen pro sledování černé čáry na podlaze a tím je zabezpečena jízda po předem definované dráze.

[7] Wi-Fi je označení pro bezdrátovou komunikaci v počítačových sítích (wireless ethernet compatibility aliance).

Baterie je zde ve formě powerbanky a slouží jako zdroj energie pro provoz vozidla. Na desce plošných spojů jsou přidělány dva Wi-Fi moduly ESP8266 a nRF24L01. Transceiver[8] ESP8266 je určen pro vzájemnou komunikaci aplikace Blynk s vozidlem ve speciálním manuálním režimu, druhý nRF24L01 slouží pro komunikaci mezi vozidlem, bránami se zásobníky a serverem. Pro sledování informací při komunikaci se serverem a zásobníky na bránách je vozidlo vybaveno LCD obrazovkou. Vozidlo disponuje také zásobníkem s násypkou, pro sběr a uložení korálků.



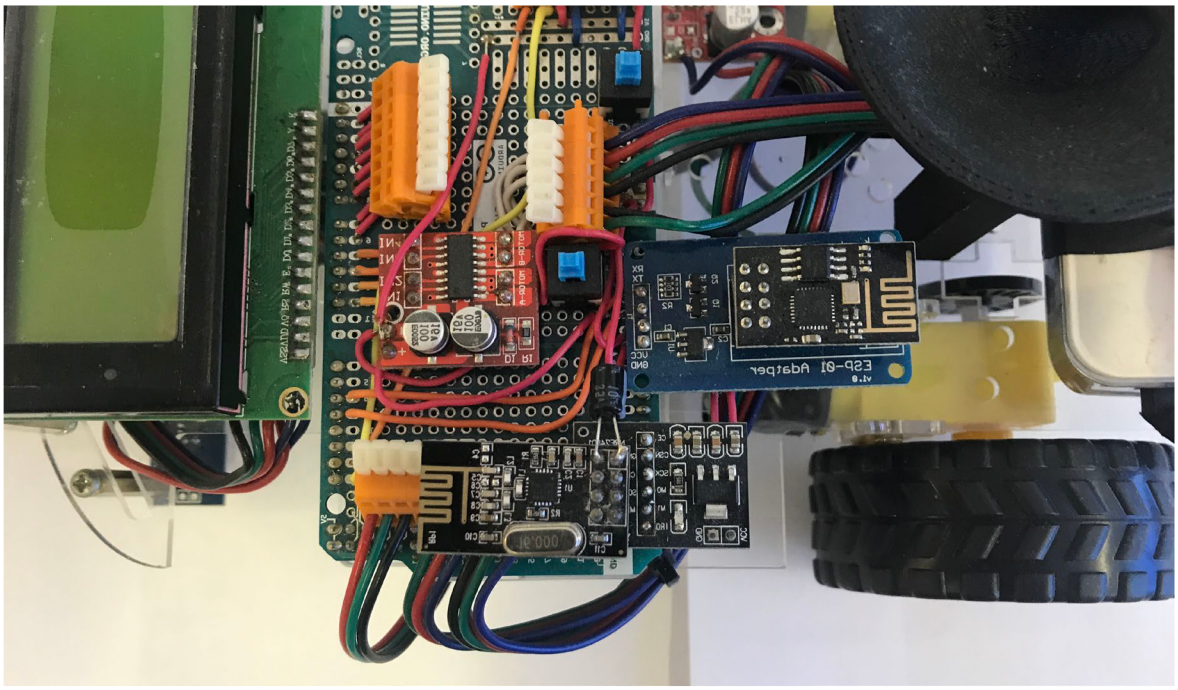
Obr. 7: Automaticky naváděné vozidlo [zdroj: vlastní]

5.2.1 Konstrukce vozidla

Celé vozidlo je zkonstruováno pomocí polymerního rámu, na kterém jsou umístěny všechny důležité komponenty potřebné pro fungování vozidla. Rám vozidla je osazen dvěma koly, které pohání dvojice DC motorů. Samotné řízení pohybu vozidla je zabezpečeno pomocí ovladače dvojitého H-můstku s napětím 2 x 16V. Takový H-můstek umožňuje stejnosměrným motorům běžet dopředu nebo dozadu. Úhlové natočení je vypočítáváno pomocí mikrokontroléru. Jestliže vozidlo potřebuje odbočit na levou stranu, musí pravé kolo vozidla udělat větší úhlové natočení, aby bylo vozidlo schopné zatočit. Tímto způsobem je umožněno vozidlu jezdit po vymezené dráze. Ve středu rámu je umístěno podpěrné otáčecí kolečko, které dodává celému rámu stabilitu při jízdě a zatáčení.

[8] Transciever je označení pro obvod, který v sobě sdružuje funkci vysílače a přijímače signálů

Na vrchní části rámu je umístěna deska Arduino propojená společně s deskou plošných spojů, na které jsou zapojeny všechny jednotlivé komponenty vozidla. Osazena je zejména z důvodu snížení počtu využitých vodičů při stavbě vozidla a díky nepřetržitému kontaktu mezi piny je tak celé vozidlo elektronicky stabilní a bez volných spojů.



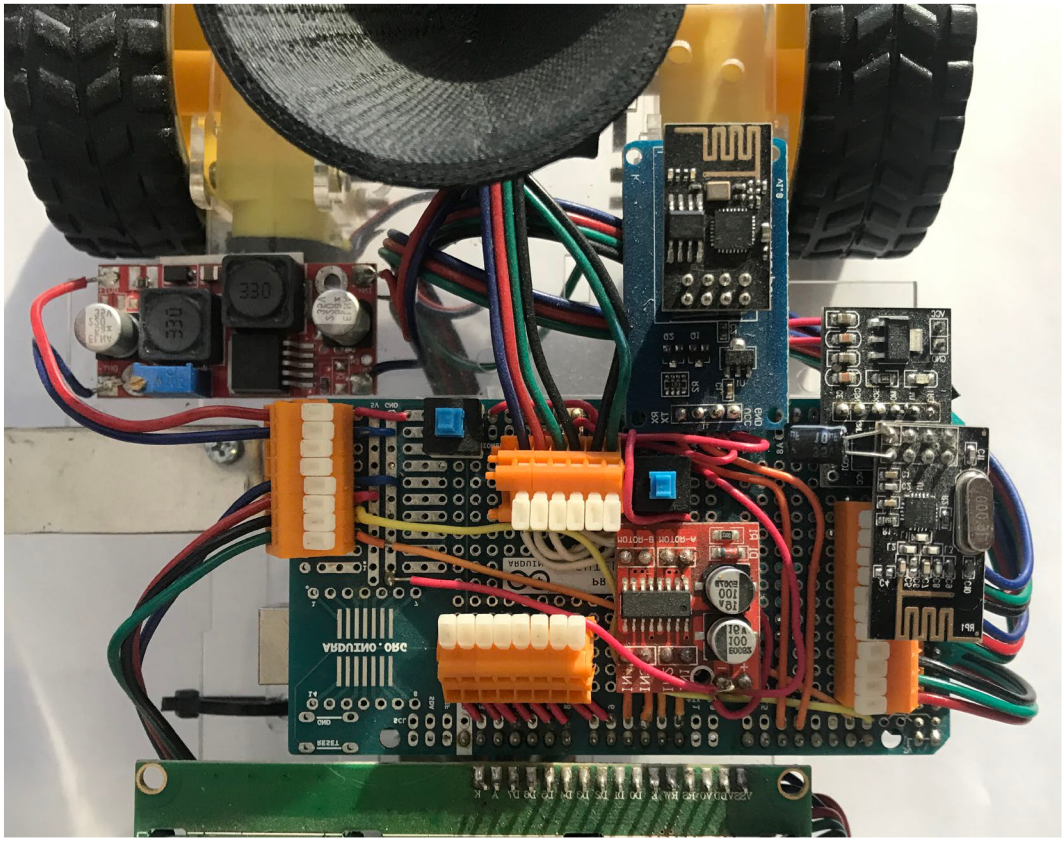
Obr. 8: Transceiver vozidla [zdroj: vlastní]

Z oblasti síťových prvků jsou dva transceivery (obr.8) umístěny ve vozidle pro komunikaci. Jedná se o komponenty, které umožňují vozidlu přijímat i vysílat zprávy. Prvním hlediskem, ze kterého jsou ve vozidle osazeny, je umožnění komunikace mezi vozidlem a serverem. Druhé hledisko je takové, aby vozidlo bylo schopné komunikovat s bránami, ve kterých jsou barevné korálky. Vozidlo obdrží objednávku ze serveru a potvrdí její příchod, dále dochází ke komunikaci s bránami, aby vozidlo bylo schopné sbírat barvu korálků, které jsou uvedené v objednávce.

Na vozidle je umístěn LCD displej, který je určen pro zobrazování komunikace mezi jednotlivými komponenty a je naprogramovaný tak, aby zobrazoval potřebnou komunikaci. V přední části vozidla je osazen senzor sledování linky, který umožňuje vozidlu jezdit po definované dráze. V horní části vozidla je umístěn zásobník s násypkou, do kterého se ukládají korálky z bran.

5.2.2 Deska plošných spojů vozidla

Všechny potřebné komponenty pro provoz vozidla (obr.9) jsou osazeny v desce plošných spojů, která je propojena s mikrokontrolerem Arduino Mega. Tištěný spoj poskytuje stabilní a spolehlivé propojení elektronických součástí, na rozdíl od předchozího propojení pomocí kabílků s konektory, kde docházelo ke špatnému kontaktu.



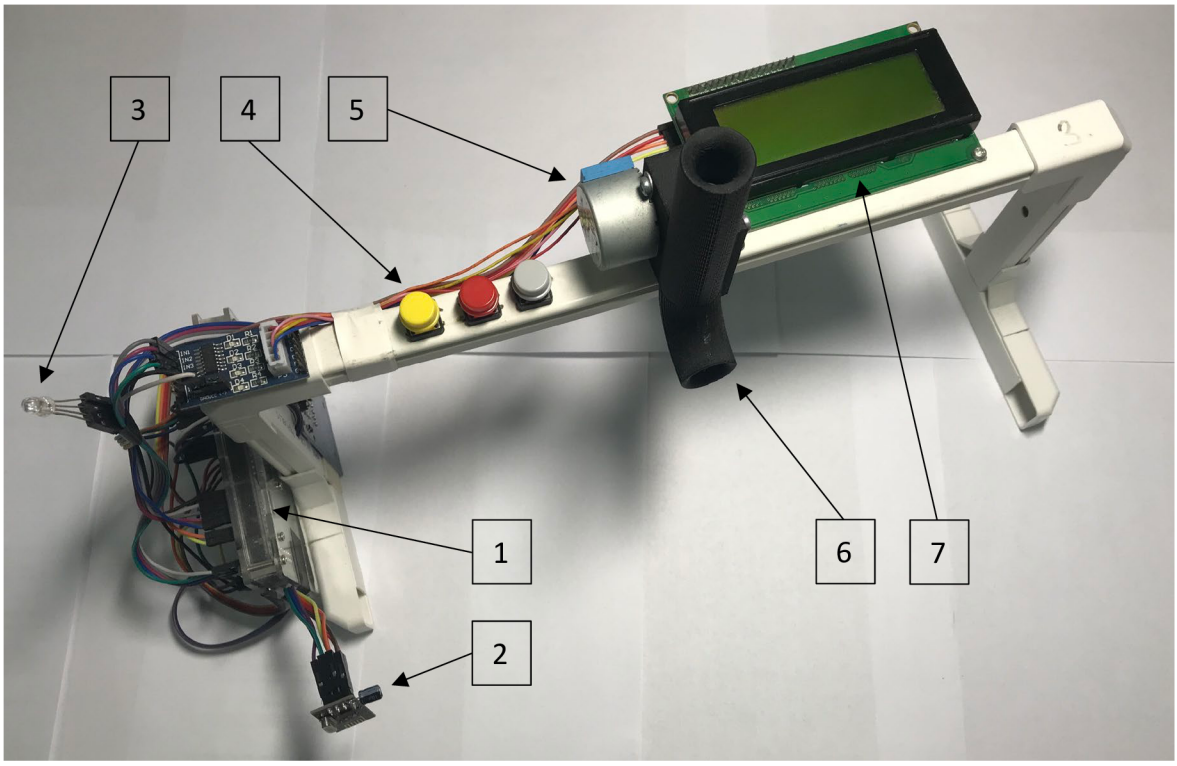
Obr. 9: Deska plošných spojů vozidla [zdroj: vlastní]

Ovladač dvojitého H-můstku je opatřen čtyřmi vstupy pro připojení dvou motorů. Pomocí těchto čtyř pinů je propojeno s digitálními vstupy desky Arduino Mega. První dva vstupní piny jsou určeny k zapojení motoru A, další dva jsou určené pro motor B. Pro připojení ovladače dvojitého H-můstku s motory A a B jsou použity dvě řady konektorů, které spojují piny z ovladače na jedné straně a pohon na straně druhé.

Třetím konektorem jsou piny SDA (Serial Data) a SCL (Serial Clock) modulu I2C propojeného s LCD displejem připojeny na digitální vstup mikrokontroleru. Tento modul s čipem PCF8574 usnadňuje připojení LCD displeje k Arduino desce. Dalším konektorem je propojen modul senzoru sledování linky s deskou Arduino Mega. Resetovací tlačítko je implementováno pomocí digitálních pinů mikrokontroleru a je určeno pro nastartování běhu programu od začátku.

5.3 Brány se zásobníky korálků

V celém modelu chytré továrny se vyskytují čtyři brány, v jehož zásobnících se vyskytují odlišné barvy korálků (červená, modrá, zelená a fialová).



Obr. 10: Brány se zásobníky korálků [zdroj: vlastní]

1 - Arduino Mega 2560, 2 – Wi-Fi modul NRF24L01, 3 - LED dioda, 4 - tlačítka, 5 - krokový motor, 6 - zásobník s korálky, 7 - LCD displej

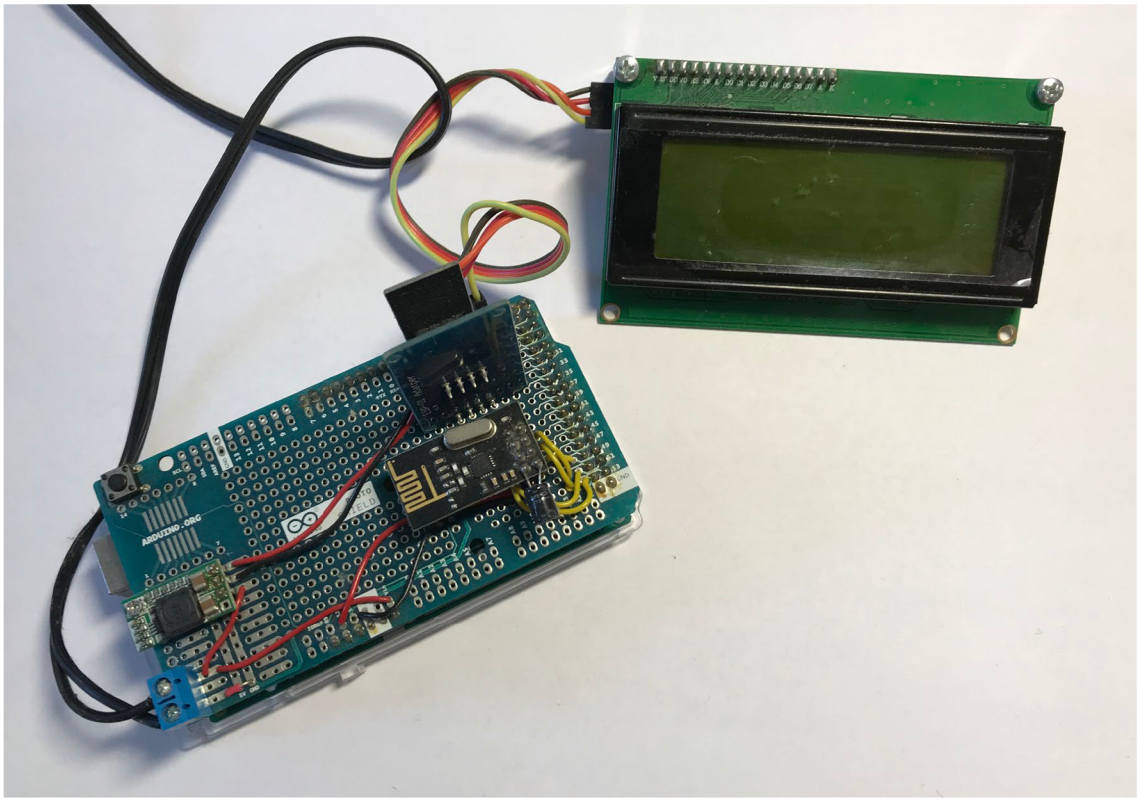
Všechny brány (obr.10) v modelu chytré továrny jsou osazeny mikrokontrolérem Arduino Mega 2560, Wi-Fi modulem nRF24L01, krokovým motorem pro výdej korálků, tlačítky a LED diodou. Zmíněný Wi-Fi modul umožňuje komunikaci brány s vozidlem i se serverem. Krokový motor je zde využíván z důvodu uvolnění korálků do zásobníku připevněného na vozidle. Samotná deska Arduino řídí a zpracovává všechny funkce, které mají být bránou provedeny.

5.4 Server

O řízení a monitorování procesní smyčky se stará server (obr.11), který zabezpečuje fungování mezi jednotlivými komponenty modelu chytré továrny a uživatelem. Server funguje jako zprostředkovatel, jelikož přijímá objednávku od uživatele na mobilním telefonu a dále ji odesílá na vozidlo.

Server pro model chytré továrny je realizován pomocí Arduino Mega 2560 s

deskou plošných spojů s wifi moduly a LCD displejem, který zobrazuje komunikaci s aplikací Blynk a zároveň komunikaci s vozidlem.



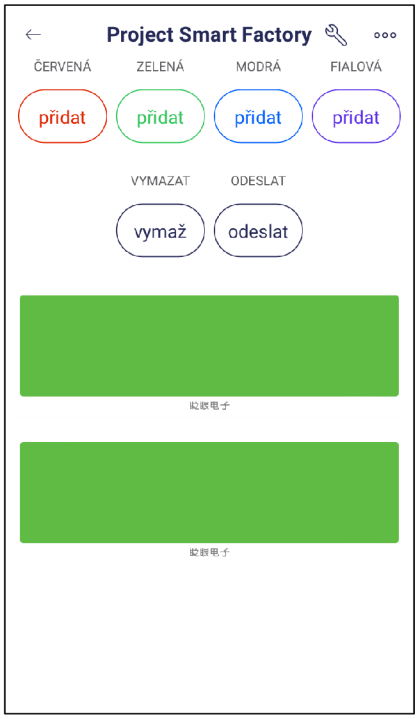
Obr. 11: Server modelu chytré továrny [zdroj: vlastní]

Server používá dva Wi-Fi moduly. První provádí komunikaci s mobilní aplikací Blynk pro definování objednávky korálků uživatelem. Druhý modul zabezpečuje přenos dané objednávky do vozidel.

5.5 Mobilní aplikace Blynk

Blynk je otevřená platforma IoT, která poskytuje bezplatnou cloudovou síť pro připojení zařízení a jeho ovládání. Jde také o platformu, která umožňuje propojení aplikací, lidí, věcí a dat s cloudovou sítí.

Propojení s cloudovou sítí je možné pomocí Ethernetu, Wi-Fi nebo 2G – 5G. Taková aplikace využívá knihovnu Blynk k připojení hardwaru a jelikož se jedná o cloud s otevřeným zdrojovým kódem, umožňuje funkčnost i v našem lokálním prostředí, a tedy usnadňuje celý proces vytváření objednávky korálků a odeslání do vozidla. Pomocí této aplikace je umožněno na dálku ovládat a spravovat více zařízení a funguje na všech chytrých telefonech.



Obr. 12: Mobilní aplikace Blynk [zdroj: vlastní]

V této aplikaci jsem vytvořil projekt Smart Factory (obr.12), který funguje s Arduino Mega 2560. Součástí je šest tlačítek pro přidání červené, zelené, modré a fialové barvy korálku. Dále je zde tlačítko pro vymazání objednávky v případě zadání špatné barvy a zároveň tlačítko pro odeslání objednávky na server. Všechna tlačítka jsou vytvořeny v módu „push“ a jejich příslušné piny jsou nakonfigurovány v projektu aplikace. Dále se zde vyskytují displeje pro sledování konfigurace objednávky a stavu odeslání, které se zobrazí vždy při jejím vytvořením.

5.5.1 Postup při objednávání

V samotné mobilní aplikaci Blynk jednotlivá čísla odpovídají vždy dané barvě korálků. Jmenovitě tedy číslo 1 odpovídá červené, číslo 2 zelené, číslo 3 modré a číslo 4 fialové barvě korálků. Po stisknutí tlačítka odeslat se do objednávky zapíše číslo 0, což znázorňuje konec objednávky. Celý systém objednávky je omezen na celkový počet šesti korálků, jelikož je tak nastaven i požadovaný limit pole objednávek.

Pro konfiguraci objednávky zákazníka je pouze nutné připojení k místnímu Wi-Fi připojení, a otevření mobilní aplikace Blynk. Dále už jen pouze stačí navolit požadované barvy korálků, které se řadí v pořadí kliknutí na tlačítka a zapisují se pouze jejich odpovídající čísla. Po zadání celé objednávky do mobilní aplikace stačí stisknout tlačítko odeslat a objednávka může být poslána na server pro další zpracování.

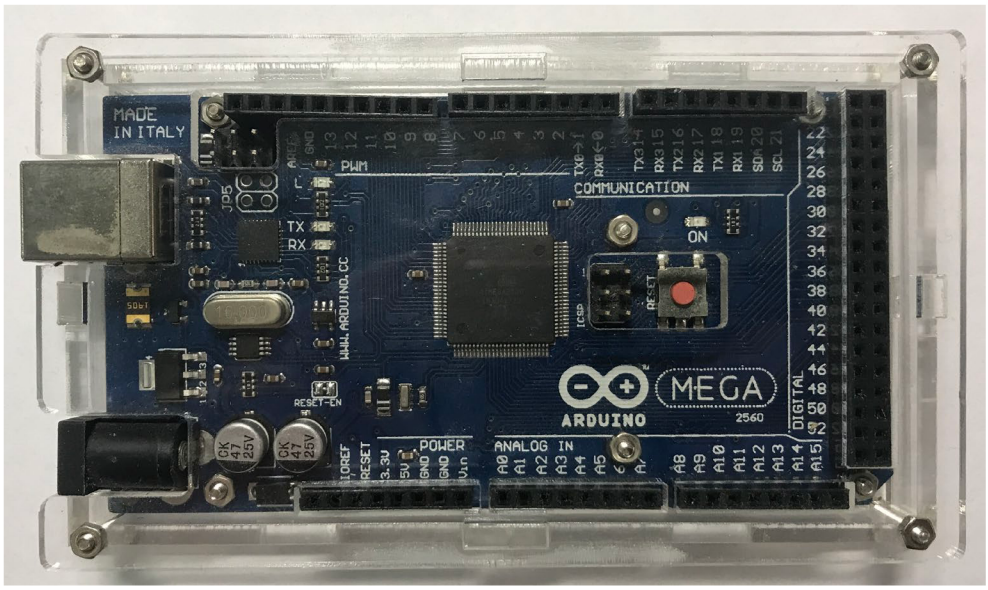
5.6 Součástky v systému Smart Factory

Model chytré továrny je sestavený z vysvětlených systémů a komponentů v předchozích kapitolách. Skládají se z mnoha elektronických součástek, které jsou určeny pro plnění speciálních funkcí v systému. Dohromady tak vytváří ucelený model chytré továrny.

5.6.1 Arduino Mega 2560

Jedná se o elektronickou desku s mikrokontrolerem, díky které je možno číst a zpracovávat vstupy a reagovat na něj pomocí výstupu. Mezi vstupy se řadí jednak data ze senzorů, ale taktéž i zprávy při tvorbě objednávky v aplikaci Blynk. Výstupem pak je pohánění servomotorů pro pohyb vozidla, krokového motoru pro výdej korálků, ale taktéž přenášení informací pomocí modulů do systému určeného pro zpracovávání. Práce mikrokontroleru závisí na podmínkách, které jsou naprogramované přímo na deskách pomocí příslušného programovacího jazyka a nahrané pomocí kabelu USB do desky Arduino.

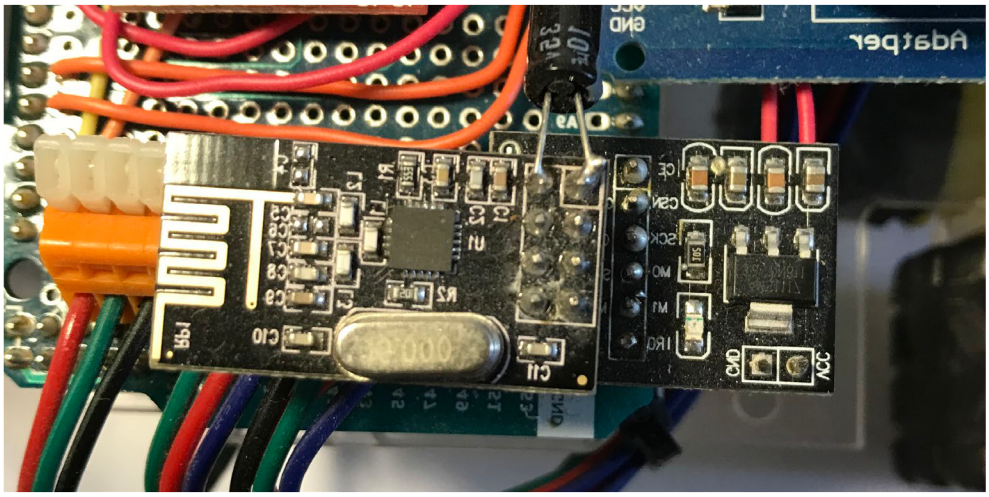
Deska Arduino Mega 2560 (obr.13) je jednou z mnoha z portfolia společnosti Arduino. Deska disponuje 54 digitálními piny určených pro vstup nebo výstup a zároveň 16 analogovými vstupními piny. Obsahuje čtyři univerzální synchronní přijímače / vysílače, které jsou určené pro přenos dat. V modelu chytré továrny jsou využívány dva transceivery, konkrétně pak pro server a vozidlo.



Obr. 13: Deska Arduino Mega 2560 [zdroj: vlastní]

5.6.2 Transceiver (nRF24L01) a Wi-Fi modul (ESP8266)

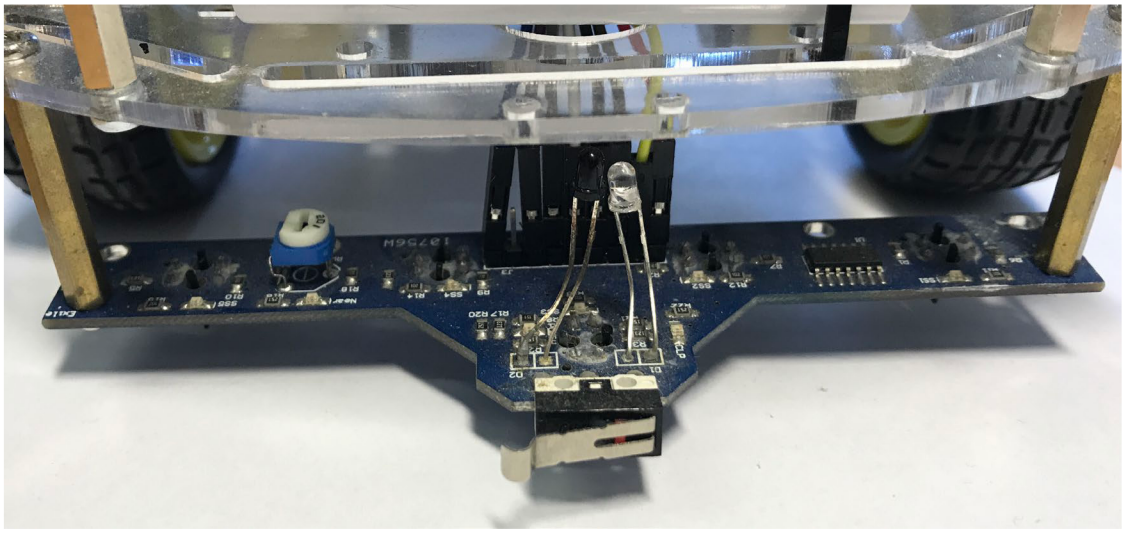
Tento Wi-Fi modul ESP8266 (obr.14) v modelu chytré továrny funguje jako vstupní modul pro desku Arduino. Jeho pracovní napětí je 3,3V, což znamená, že k 5V desce Arduina je připojen přes adaptér, který sníží napětí právě na tuto hranici. Využívá se pro Wi-Fi komunikaci mezi zařízeními, tedy pro využití IoT v praxi. Je osazen 32bitovým RISC procesorem od společnosti Tensillica a mezi hlavní výhody se řadí jeho malá spotřeba energie.



Obr. 14: Transceiver nRF24L01 a Wi-Fi modul ESP8266 [zdroj: vlastní]

5.6.3 Modul senzoru sledování linky TCRT5000

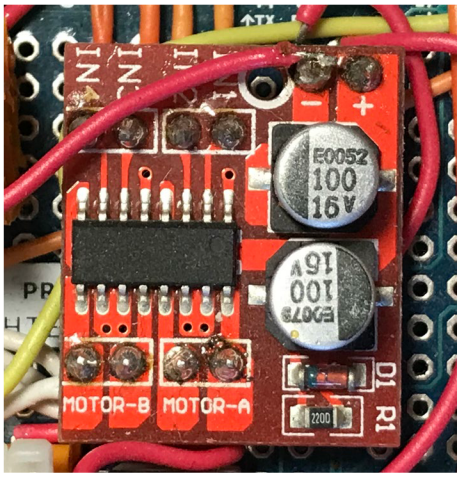
Na vozidlech, která jsou součástí modelu chytré továrny jsou v přední části vozu osazeny pětikanálové senzory sledování linky (obr.15). Má vysokou citlivost a jeho pracovní napětí je 5V, zároveň disponuje pěti výstupy (SS1 – SS5). Společně s ostatními piny jsou propojeny pomocí konektorů s deskou Arduino na vozidle.



Obr. 15: Modul senzoru sledování linky TCRT5000 [zdroj: vlastní]

5.6.4 Ovladač dvojitého H-můstku

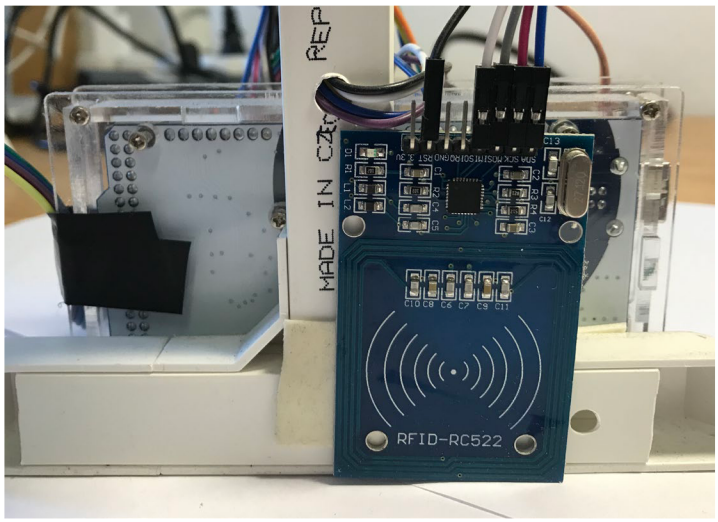
Ovladač dvojitého H-můstku je v konstrukci vozidla využíván k ovládání otáčení obou servomotorů (obr.16). Piny IN1 a IN2 jsou vstupy pro servomotor A, IN3 a IN4 pro servomotor B. PWM[9] signál pro oba motory je propojen pomocí pinů ENA a ENB.



Obr. 16: Ovladač dvojitého H-můstku [zdroj: vlastní]

5.6.5 RFID senzor

V celém modelu jsou umístěny dohromady 4 RFID senzory na všech bránách se zásobníky korálků. Tyto senzory poskytují informace vozidlu z toho důvodu, aby rozpoznalo, o jakou barvu korálků v bráně se jedná. Tento senzor disponuje SDA a SCK vstupními piny, které jsou propojeny s deskou Arduina. Pin RST slouží jako resetovací a piny s označením MISO a MOSI jsou určeny k předávání informací vozidlu.



Obr. 17: RFID senzor [zdroj: vlastní]

[9] PWM je označení pro pulzně šířkovou modulaci (pulse width modulation), je určena pro přenos analogového signálu pomocí dvouhodnotového signálu

6 PRINCIPY MODELU SMART FACTORY

Funkcionalita modelu chytré továrny je založena na principu stavových automatů. Znázorňují se pomocí grafického diagramu, který je založen na podmínkách pro přechody do dalších stavů, při kterých může plnit libovolné akce. Stavové automaty jsou zpravidla vymyšleny jako uzavřené smyčky, které mohou běžet nekonečně dlouho do doby restartování systému. Pomocí podmínek dochází k rozhodování o větvení směru chodu funkčního automatu. Jednotlivé akce se dějí vždy po splnění podmínky a jsou zároveň rozděleny do mnoha stavů, díky kterým nalezneme snadné řešení funkčnosti modelu chytré továrny.

6.1 Princip automatu tvorby objednávky

Server řídí systém chytré továrny a umožňuje přenesení objednávky z mobilu na vozidlo. Pomocí aplikace Blynk je vytvořena aplikace pro definování specifikované objednávky zákazníkem v libovolném pořadí barevných korálků. Pomocí stavového diagramu (obr.18) jsem vytvořil funkční model procesu pro zadávání objednávky pomocí tlačítek červené, zelené, modré a fialové barvy. Dále se v diagramu vyskytuje tlačítko pro smazání a odeslání objednávky.

Pomocí jednoduchého grafického znázornění automatu je vysvětlena funkčnost během tvorby objednávky uživatelem. Jsou znázorněny podmínky, které automat musí zkontrolovat v každém stavu. Dále jsou znázorněny akce, které mají být provedeny v případě, že jsou dané podmínky splněny. Zmíněný graf automatu (obr.18) je vytvořen pro tlačítka, které korespondují s tlačítky v mobilní aplikaci Blynk (obr.12).

Při resetu celého automatu je pole objednávek „i“ nastaveno na první pozici a zároveň jsou všechny tlačítka nastaveny na stav „uvolnit“. Tímto způsobem je ošetřen případ, který by mohl při spuštění programu způsobit samovolné splnění podmínky bez toho, aniž by uživatel stiskl libovolné tlačítko.

V prvním stavu „S1“ program čeká na zákazníka, aby si definoval objednávku korálků na základě potřebných barev. Jestliže zákazník stiskne tlačítko pro červenou barvu korálku, provedou se dvě akce:

1. První buňka pole objednávky je nastavena na číslo odpovídající barvě korálku. V případě červené barvy se jedná o číslo „1“.
2. Dále je kurzor pro vstup nastaven na další buňku pole objednávky pro zadání další barvy korálku.

V tomto okamžiku dojde k přechodu do dalšího stavu „S2“, kde automat čeká na uvolnění stejného tlačítka. Je-li tato podmínka splněna, dojde na sériovém monitoru Arduino k zobrazení pole objednávky, současnému stavu tlačítek a aktuálnímu místu na cestě.

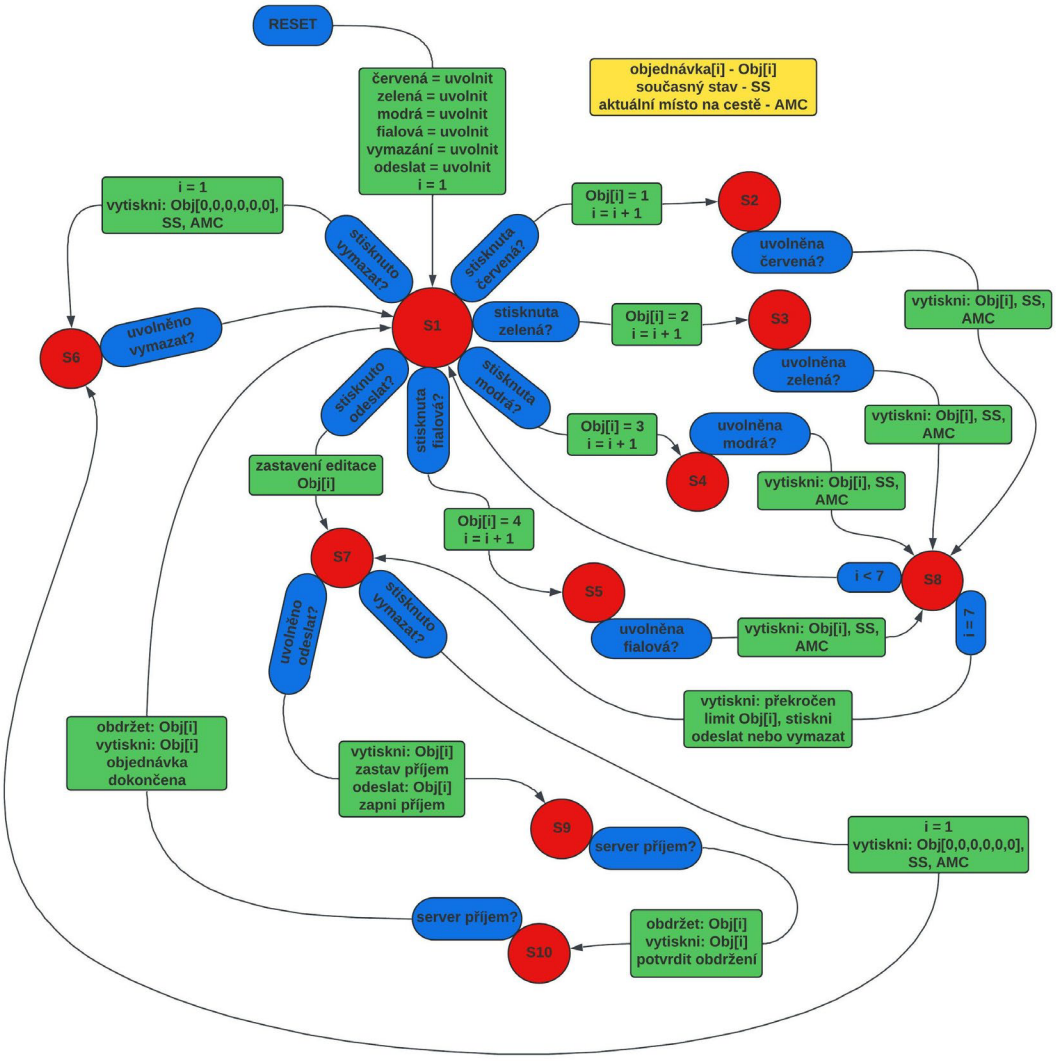
Po vykonání těchto akcí automat přechází do stavu „S8“, kde testuje podmínku o počtu objednaných korálků. Jestliže je pole objednávek „i“ menší než číslo „7“, tak automat přechází zpět do výchozího stavu „S1“, kde čeká na další stisknutí tlačítek pro další korálek v další buňce pole objednávky. Princip objednání ostatních barev má identický postup, jen se v případě ostatních barev zapíšou do buňky pole objednávky odpovídající čísla (zelená = „2“, modrá = „3“, fialová = „4“).

Jestliže dojde v předchozím stavu „S8“ ke splnění podmínky, že pole objednávek je rovno číslu „7“, pak je naplněn limit pro maximální počet korálků v objednávce. Zákazníkovi se na monitoru objeví informace o překročení limitu a vyzve ho buď ke stisku tlačítka odeslat nebo smazat objednávku.

Celý proces definování objednávky pokračuje tak dlouho, dokud uživatel nestiskne tlačítko pro odeslání objednávky. V případě, kdy uživatel stiskne tlačítko „vymazat“, dojde k vyprázdnění pole objednávek tím způsobem, že se všechny buňky pole nahradí číslem „0“. Zároveň se kurzor pole objednávek „i“ nastaví na pozici „1“ a na monitoru se zobrazí pole objednávky, současný stav tlačítek a aktuální místo na cestě. Po uvolnění tlačítka „vymazat“ automat přechází zpět do výchozího stavu „S1“.

V případě, kdy zákazník stiskne tlačítko „odeslat“, dojde k zastavení smyčky v pořadí příjmu od zákazníka a zbývající nevyplněné buňky v poli jsou nataveny na „0“. Automat se přesune do stavu „S7“, kde čeká na uvolnění tlačítka „odeslat“ nebo stisknutí tlačítka „vymazat“. V případě splnění podmínky pro stisk tlačítka pro vymazání objednávky, které pramení z cesty automatu ze stavu „S8“ při naplnění limitu objednávky, automat přechází do stavu „S6“ a následně po uvolnění tlačítka zpět do výchozího stavu „S1“. Po uvolnění tlačítka „odeslat“, aplikace Blynk přestane přijímat objednávku, zobrazí objednávku na sériovém monitoru a odešle objednávku.

Ve stavu „S9“ a „S10“ dochází k testování podmínky o příjmu serveru, v případě splnění je obdržena a vytisknuta objednávka. Potvrzení obdržení objednávky je realizováno ve stavu „S9“. Následně automat ze stavu „S10“ přechází zpět do stavu „S1“, čímž je objednávka dokončena.



Obr. 18: Automat tvorby objednávky [zdroj: vlastní]

6.1.1 Posloupnost příkazů pro tvorbu objednávky

Pro názornost funkčnosti celého automatu pro tvorbu objednávky jsou sepsány příkazy, které slouží pro jeho implementaci a zároveň na sebe chronologicky navazují:

Reset.

Tlačítka červená, zelená, modrá, fialová, vymazání, odeslat nastav na „uvolnit“, i = 1.

1. Jestliže stisknuto tlačítko:
   a. Červená, Obj[i] = 1, i = i + 1. Přejdi do stavu S2.
   b. Zelená, Obj[i] = 2, i = i + 1. Přejdi do stavu S3.
   c. Modrá, Obj[i] = 3, i = i + 1. Přejdi do stavu S4.
   d. Fialová, Obj[i] = 4, i = i + 1. Přejdi do stavu S5.
   e. Vymazat, i = 1, vytiskni Obj[0,0,0,0,0,0], SS, AMC. Přejdi do stavu S6.

f. Odeslat, zastav editaci Obj[i]. Přejdi do stavu S7.

2. Uvolněno tlačítko červená, přejdi do stavu S8.
3. Uvolněno tlačítko zelená, přejdi do stavu S8.
4. Uvolněno tlačítko modrá, přejdi do stavu S8.
5. Uvolněno tlačítko fialová, přejdi do stavu S8.
6. Uvolněno tlačítko vymazat, přejdi do stavu S1.
7. Jestliže je splněna podmínka:
   a. Uvolněno tlačítko odeslat, vytiskni Obj[i], zastav příjem, odešli Obj[i], zapni příjem. Přejdi do stavu S9.
   b. Stisknuto tlačítko vymazat, i = 1, vytiskni Obj[0,0,0,0,0,0], SS, AMC. Přejdi do bodu S6.
8. Jestliže pole objednávek:
   a. i < 7, přejdi do stavu S1.
   b. i = 7, vytiskni překročen limit objednávky, stiskni odeslat nebo vymazat. Přejdi do stavu S7.
9. Pokud je splněna podmínka server příjem, obdržet Obj[i], vytiski Obj[i], potvrď obdržení. Přejdi do stavu S10.
10. Pokud je splněna podmínka server příjem, obdržet Obj[i], vytiskni Obj[i], objednávka dokončena. Přejdi do stavu S1.

6.2 Princip automatu pro distribuci objednávek

V případě, kdy v modelu chytré továrny bude implementováno více vozidel, server musí být schopen rozřadit objednávky mezi vozidly na základě podmínek. Po kontrole a potvrzení daných podmínek dojde k výběru vozidla, kterému má být vydán příkaz. Konkrétně se jedná o podmínky o kontrole dostupnosti vozidla, stavu napájení. Tímto způsobem pak vydá příkaz o přiřazení objednávky danému vozidlu.

6.2.1 Popis pracovního postupu distribuce objednávek

Server obdrží objednávku definovanou zákazníkem a musí úkol pro sběr korálků přiřadit některému z vozidel. Pro rozdělování práce musí server kontrolovat nezbytné podmínky. Mezi kritéria pro přiřazení objednávky vozidlu patří:

a. Vozidlo musí být bez jakékoliv objednávky.
b. Stav kapacity baterie musí být vyšší než 25%.

Vozidla, která se kontrolují pro splnění výše uvedených kritérií, se vždy začínají

kontrolovat v pořadí od prvního vozidla. Takovým způsobem jsou objednávky přiřazovány daným vozidlům jedna po druhé. Pro různý počet vozidel v modelu chytré továrny pak stačí pouze přepsat podmínku kontroly hodnoty „p“ na odpovídající počet vozidel v modelu.

Na obrázku 19 je znázorněný automat serveru, který přiřazuje objednávku[i] vozidlu, které splňuje dané podmínky. Celý proces je možný díky proměnné „APV[p]“, pomocí které dochází k ukládání adresy jednotlivých vozidel. V případě, že jsou u daného vozidla splněny dané podmínky, je k objednávce přiřazeno vozidlo s adresou uloženou právě v proměnné „APV[p]“. Dochází tedy k distribuci objednávky serverem.

Obr. 19: Automat distribuce objednávek [zdroj: vlastní]

6.2.2 Posloupnost příkazů pro distribuci objednávek

Na základě kritérií jsou vytvořeny jednotlivé příkazy, které jsou určeny pro implementaci tohoto procesu na serveru. Pro ukládání adresy vozidla je zavedena proměnná „APV[p]“. Jsou-li obě podmínky splněny, objednávka je odeslána na vozidlo s příslušnou adresou.

Reset.

Server příjem.

1. Pokud je splněna podmínka Obj[i] přijata, p = 0, odeslat zprávu na APV[p], server příjem. Přejdi do stavu S2.
2. Pokud je splněna podmínka Obj[i] přijata. Přejdi do stavu S3.
3. Zkontroluj dostupnost vozidla:
    a. Volné vozidlo = 1. Přejdi do stavu S4.
    b. Volné vozidlo = 0, vytiskni vozidlo v provozu, p = p + 1. Přejdi do stavu S5.
4. Zkontroluj kapacitu baterie:
    a. BT > 25%, odeslat Obj[i] na vozidlo s adresou APV[p], server příjem. Přejdi do stavu S1.
    b. BT ≤ 25%, p = p + 1. Přejdi do stavu S5.
5. Zkontroluj hodnotu „p“:
    a. p = 2, nastav hodnotu p = 0. Přejdi do stavu S6.
    b. p < 2. Přejdi do stavu S6.
6. Pokud je hodnota p < 2, odeslat zprávu na adresu APV[p], server příjem. Přejdi do stavu S2.

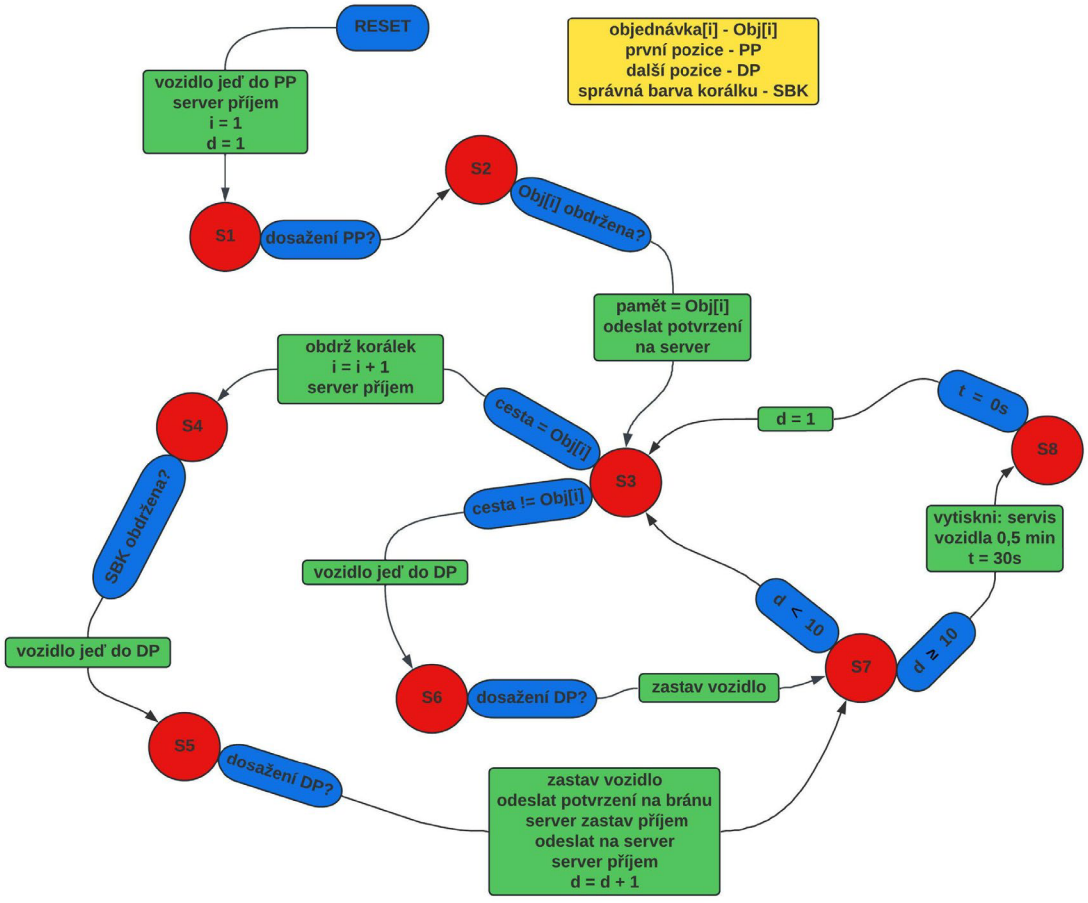
6.3 Princip automatu pro fungování vozidla

Vozidlo je klíčovou součástí modelu chytré továrny, jelikož umožňuje sesbírání správných barevných korálků ve správném pořadí. Pro znázornění je vytvořen stavový diagram (obr.20), který tento proces usnadňuje a zvyšuje jeho efektivitu.

Při resetu celého automatu je pole objednávek „i“ nastaveno na první pozici, proměnná „d“ nastavena na hodnotu „1“ a zároveň je nastaven pohyb vozidla do první pozice. Ve stavu „S1“ se pouze čeká na dosažení první polohy vozidlem, je-li podmínka splněna, dojde k přechodu do stavu „S2“. V tomto stavu dochází ke kontrole podmínky obdržení objednávky ze serveru na vozidlo. Při splnění dojde k uložení objednávky do paměti, odeslání potvrzení na server a přechází do dalšího stavu. Ve

stavu „S3“ dochází k testování dvou podmínek. Jestliže je na cestě brána s požadovanou barvou korálků dle první buňky v objednávce, tak vozidlo čeká na výdej a zapíná příjem serveru. V nadcházejícím stavu „S4“ se pouze čeká na splnění podmínky o výdeji správné barvy korálku a vozidlo pak jede do další pozice. Ve stavu „S5“ dochází k testování dosažení polohy další brány pro výdej korálků a zastaví vozidlo. Zároveň dochází k odesílání potvrzení bráně, odeslání na server a zvýšení proměnné „d“.

V případě, že ve stavu „S3“ dojde ke splnění podmínky nesprávné barvy brány vůči objednávce, tak vozidlo jede do další pozice. Následný stav „S6“ pouze kontroluje dosažení další pozice. Proměnná „d“ je tu z toho důvodu, jelikož v modelu chytré továrny dochází k simulaci opravy vozidla. Ve stavu „S7“ dochází tedy k testování velikosti proměnné „d“. Jestliže je její hodnota „d < 10“, tak automat přechází zpět do stavu „S3“ a opět porovnává objednávku s barvou brány u které stojí. V případě že hodnota „d ≥ 10“, tak se na sériovém monitoru zobrazí servis vozidla a dojde k nastavení časovače na 30 sekund. Po uplynutí této doby se proměnná „d“ nastaví zpět na hodnotu jedna. Automat zároveň přechází zpět do stavu „S3“.



Obr. 20: Automat fungování vozidla [zdroj: vlastní]

6.3.1 Posloupnost příkazů pro fungování vozidla

Pro názornost funkčnosti celého automatu fungování vozidla jsou sepsány příkazy poskytující informace o činnostech, na které se vozidlo potřebuje soustředit v každé fázi. Vozidlo tyto příkazy plní krok po kroku.

Reset.

Vozidlo jeď do PP, server příjem, i = 1, d = 1.

1. Pokud je splněna podmínka dosažení PP. Přejdi do stavu S2.
2. Pokud je splněna podmínka Obj[i] přijata, paměť = Obj[i], odeslat potvrzení na server. Přejdi do stavu S3.
3. Zkontroluj bránu s požadovanou barvou na cestě:
    a. Cesta = Obj[i], obdrž korálek, i = i + 1, server příjem. Přejdi do stavu S4.
    b. Cesta != Obj[i], vozidlo jeď do DP. Přejdi do stavu S6.
4. Pokud je obdržena SBK, vozidlo jeď do DP. Přejdi do stavu S5.
5. Pokud dosažena DP, zastav vozidlo, odeslat potvrzení na bránu, server zastav příjem, odeslat na server, zapni příjem, d = d + 1. Přejdi do stavu S7.
6. Pokud je dosažena DP, zastav vozidlo. Přejdi do stavu S7.
7. Zkontroluj hodnotu proměnné „d“:
    a. d < 10. Přejdi do stavu S3.
    b. d ≥ 10, vytiskni servis vozidla 0,5 min, t = 30s. Přejdi do stavu S8.
8. Pokud je t = 0s, d = 1. Přejdi do stavu S3.

6.4 Optimalizace funkčnosti pohybu vozidla

V sytému chytré továrny jsou celkem čtyři brány, kterými vozidlo projíždí. Každá z těchto bran má zásobník s korálky příslušné barvy. Vozidlo je navrženo tak, aby se pohybovalo po určité dráze pomocí snímače polohy a cestou sbíralo korálky ze zásobníků na bránách, které odpovídají objednávce.

Během tohoto procesu se vozidlo chová podle automatu, díky kterému je umožněno zkontrolovat korálky v objednávce a zároveň je díky tomu posbírat z jednotlivých bran. Takový automat se musí různými způsoby optimalizovat, aby se zlepšil způsob, jakým dané vozidlo korálky sbírá.

V optimalizačním procesu jsou zahrnuty dohromady tři strategie (A,B,C), které zvyšují efektivitu pohybu vozidla při sbírání objednaných korálků. Každá strategie obsahuje specifické vylepšení. Další dvě varianty obsahují implementaci mapování trasy vozidlem, díky které nemusí být definováno pořadí barevných bran, a zároveň je

umožněno odbočování na definované dráze. Poslední varianta přiřazuje vozidlu schopnost neustále aktualizovat pořadí bran, které umožňuje jet na přímo stanovenou pozici zásobníku s danou barvou korálků. Hlavním cílem implementace uvedených strategií je optimalizace pohybu vozidla při sbírání korálků.

6.4.1 Princip fungování vozidla – strategie A

Tato základní strategie A, která se používá pro pohyb vozidla je jednoduchá. Vozidlo v chytré továrny zkontroluje každou buňku v poli objednávek pro výdej barevných korálků ze zásobníků, pak se přesune k další bráně. V této strategii se bere v úvahu první buňka pole, podle které se vozidlo pohybuje mezi jednotlivými pozicemi zásobníků pro sběr jednotlivých korálků objednávky v požadovaném pořadí. V okamžiku, kdy se sebere korálek z první buňky pole, začne se plnit druhá buňka pole objednávky. Takto celý proces pokračuje až k poslední buňce pole, kdy je objednávka dokončena.

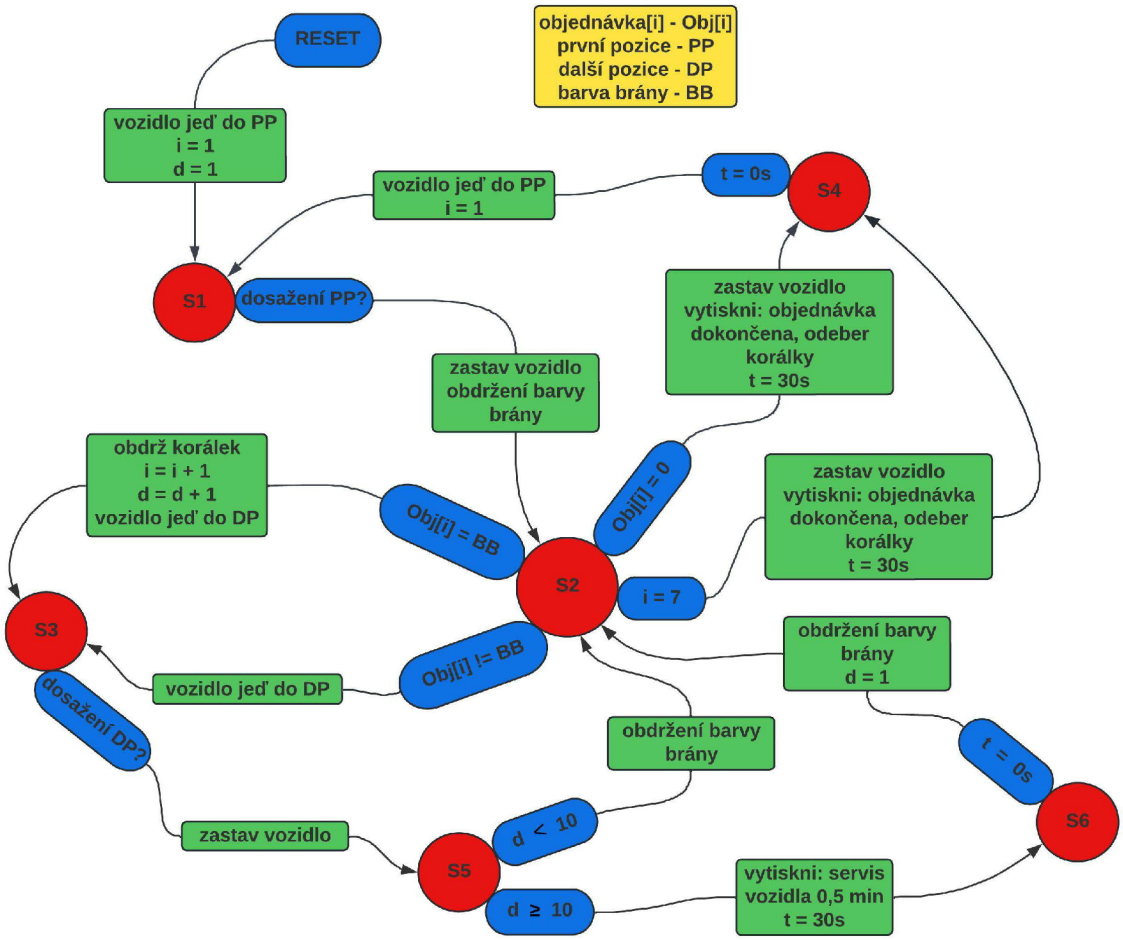
6.4.1.1 Popis pracovního postupu

Vozidlo se nejprve přesune na první pozici, kde požádá o korálek v první buňce pole objednávky z prvního zásobníku. Jestliže se vyskytuje v daném zásobníku požadovaná barva korálku, dojde k jeho sebrání. Poté se vozidlo přesune k další pozici zásobníku s korálky. V případě, že se nejedná o požadovanou barvu, vozidlo se přesune do další pozice a požádá o stejnou barvu korálku v první buňce z dalšího zásobníku. Takovým způsobem vozidlo pokračuje dál.

Teprve v okamžiku, kdy získá korálek v první buňce objednávky z některého zásobníku na bráně, postoupí k druhé buňce. Vozidlo pak žádá o další výdej požadované barvy ze zásobníků, tedy sbírá korálky jeden po druhém. Tímto způsobem se pokračuje do doby dokončení objednávky, která je omezena na maximální počet šesti korálků. Vozidlo poté přejíždí do počáteční polohy.

Na tomto základě je graf automatu (obr.21) navržen tak, aby byl snadno naprogramovaný. Dané akce mají být provedeny v případě, kdy jsou splněny podmínky testované v jednotlivých stavech. Při resetu celého automatu je pole objednávek „i“ nastaveno na první pozici a proměnná „d“ je nastavena na hodnotu „1“. Pohyb vozidla je nastaven do první pozice. Ve stavu „S1“ je testována podmínka o dosažení první pozice. Pak se vozidlo zastaví, žádá informaci o obdržení barvy brány. V následujícím stavu „S2“ se testují podmínky, díky kterým vozidlo získá představu o shodnosti barvy

v zásobníku a objednávky. Dojde buď k výdeji korálku, nastavení na následující buňku pole objednávky a přesun do další pozice nebo k přesunu dojde ihned v případě nesprávné barvy zásobníku s porovnáním v poli objednávky. Dále jsou tam obsaženy další dvě podmínky pro případ přesunutí do posledního pole objednávky nebo naleznutím konce objednávky symbolizované hodnotou „0“. V obou těchto případech dojde k zastavení vozidla na 30 sekund a zobrazením zprávy o dokončení objednávky. Po uplynutí tohoto časového intervalu se vozidlo přesouvá do první pozice, pole objednávek „i“ je nastaveno na první pozici. Ve stavu „S5“ dochází pouze ke kontrole hodnoty proměnné „d“, která slouží k simulování případu servisu vozidla.



Obr. 21: Automat fungování vozidla – strategie A [zdroj: vlastní]

6.4.1.2 Posloupnost příkazů – strategie A

Pro názornost funkčnosti celého automatu fungování vozidla se strategií A jsou sepsány příkazy poskytující informace o činnostech, na které se vozidlo potřebuje soustředit v každé fázi. Vozidlo tyto příkazy plní krok po kroku a pomáhá při sestavení programu s rozdělenými kontrolami podmínek. Příkazy slouží k implementaci a jsou

následující:

Reset.

Vozidlo jeď do PP, i = 1, d = 1.

1. Pokud je dosažena PP, zastav vozidlo, obdržení barvy brány. Přejdi do stavu S2.
2. Zkontroluj požadované podmínky:
    a. Obj[i] = BB, obdrž korálek, i = i + 1, d = d + 1, vozidlo jeď do DP. Přejdi do stavu S3.
    b. Obj[i] != BB, vozidlo jeď do DP. Přejdi do stavu S3.
    c. i = 7, zastav vozidlo, vytiskni objednávka dokončena, t = 30s. Přejdi do stavu S4.
    d. Obj[i] = 0, zastav vozidlo, vytiskni objednávka dokončena, t = 30s. Přejdi do stavu S4.
3. Pokud je dosaženo DP, zastav vozidlo, obdržení barvy brány. Přejdi do stavu S5.
4. Pokud je t = 0s, vozidlo jeď do PP, i = 0. Přejdi do stavu S1.
5. Zkontroluj hodnotu proměnné „d“:
    a. d < 10, obdržení barvy brány. Přejdi do stavu S2.
    b. d ≥ 10, vytiskni servis vozidla 0,5 min, t = 30s. Přejdi do stavu S6.
6. Pokud je t = 0s, obdržení barvy brány, d = 1. Přejdi do stavu S2.

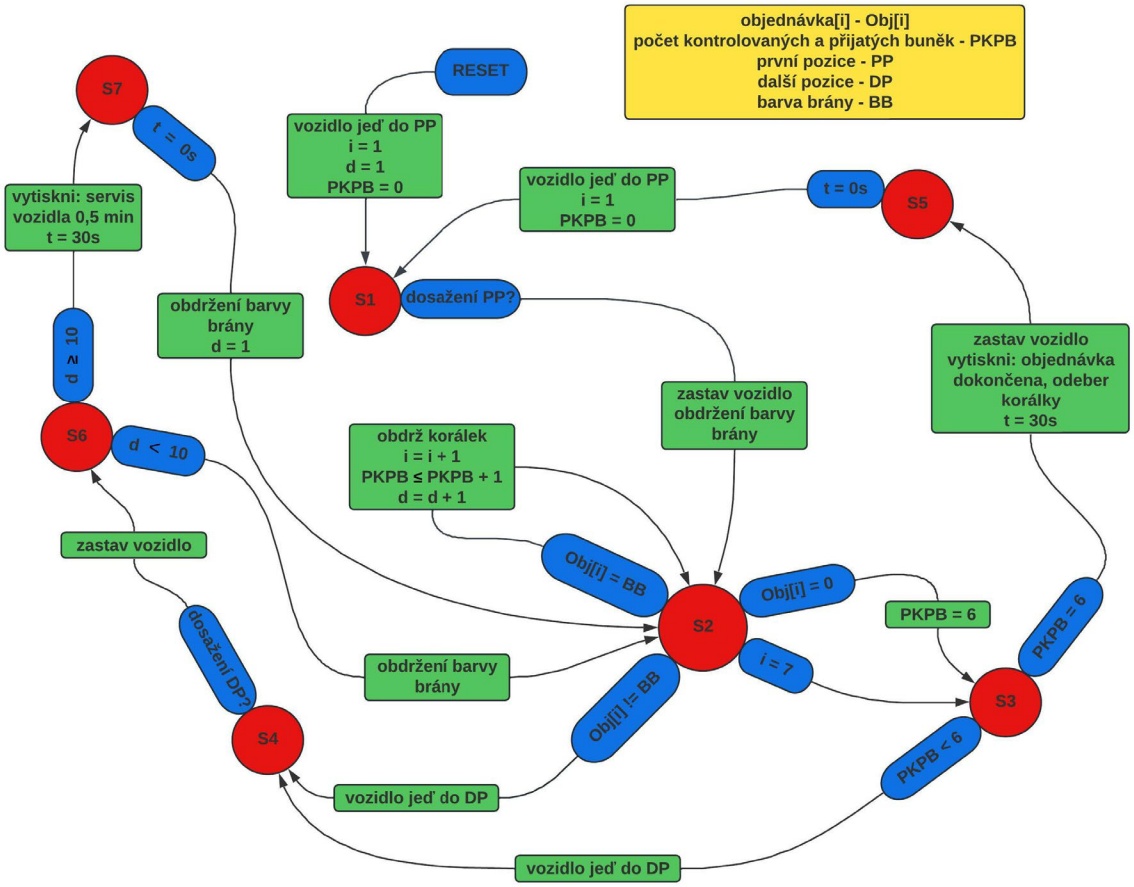
6.4.2 Princip fungování vozidla – strategie B

Strategie B přináší zlepšení přístupu kontroly objednávky pomocí zásobníků na bránách v jednotlivých pozicích. Vozidlo tedy zkontroluje další buňku pole objednávky, aby obdrželo korálky z téhož zásobníku v případě, že jsou objednány. V tomto případě vozidlo nejprve kontroluje první buňku pole se zásobníkem, jestliže dojde ke shodě tak čeká na výdej korálku. Pak kontroluje jednotlivé buňky pole jednu po druhé. V případě, že se barva korálku neshoduje se zásobníkem, vozidlo jede do další pozice. Z toho vyplývá, že v každé pozici zásobníku vozidlo kontroluje korálky v pořadí pole po sobě jdoucích buněk pro stejně barevné korálky v objednávce. To umocňuje situaci, že vozidlo kontroluje pořadí mnoha po sobě jdoucích buněk u každého ze zásobníků v případě, je-li to možné.

6.4.2.1 Popis pracovního postupu

Vše začíná přesunutím vozidla do první polohy a požádá o korálek, jehož barva je definovaná v první buňce pole objednávky. Po kontrole první buňky najde odpovídající zásobník s požadovanou barvou a získá korálek. Pak se v té samé pozici kontrolují po sobě jdoucí buňky pole objednávky do doby shodnosti odpovídající barvy korálku objednávky se zásobníkem. V případě, že barva v objednávce nesouhlasí s barvou zásobníku, vozidlo jede k další pozici a pokračuje stejným způsobem prohledáváním a kontroly pole objednávky. Tímto způsobem se pokračuje až do doby, než vozidlo získá všechny objednané korálky.

Na tomto základě je graf automatu (obr.22) navržen tak, aby byl do jisté míry co nejjednodušší. Při resetu celého automatu je pole objednávek „i“ nastaveno na první pozici, hodnota počtu kontrolovaných a přijatých buněk „PKPB“ je nastavena na hodnotu „0“ a proměnná „d“ je na „1“. Pohyb vozidla je nastaven do první pozice. Ve stavu „S1“ je testována podmínka o dosažení první pozice. Pak se vozidlo zastaví, žádá informaci o obdržení barvy brány. V následujícím stavu „S2“ se testují podmínky, které dávají přehled o shodnosti barvy v zásobníku a objednávky. Ve správném případě dojde buď k výdeji korálku, nastavení na následující buňku pole objednávky a zvýšení hodnoty proměnné „PKPB“ o hodnotu „1“ nebo k přesunu dojde ihned v případě nesprávné barvy zásobníku s porovnáním v poli objednávky. Dále ve stavu „S2“ jsou implementovány podmínky pro případ, kdy se objednávka přesune do posledního pole nebo nalezne konec objednávky symbolizovaného hodnotou „0“. V obou těchto případech se stav přesouvá do stavu „S3“, kde testuje velikost hodnoty proměnné „PKPB“. V případě, že je její hodnota rovna šesti, dojde k zastavení vozidla na 30 sekund a zobrazením zprávy o dokončení objednávky. Po uplynutí tohoto časového intervalu se vozidlo přesouvá do první pozice, pole objednávek „i“ je nastaveno na první pozici a hodnota proměnné „PKPB“ je nastavena na „0“. Ve stavu „S6“ dochází pouze ke kontrole hodnoty proměnné „d“, která slouží k simulování případu servisu vozidla.



Obr. 22: Automat fungování vozidla – strategie B [zdroj: vlastní]

6.4.2.2 Posloupnost příkazů – strategie B

Pro funkčnost celého automatu vozidla se strategií B jsou sepsány příkazy poskytující informace o činnostech, na které se vozidlo potřebuje soustředit v každé fázi. Vozidlo tyto příkazy plní krok po kroku a pomáhá při sestavení programu.

Reset.

Vozidlo jeď do PP, i = 1, d = 1, PKPB = 0.

1. Pokud je dosažena PP, zastav vozidlo, obdržení barvy brány. Přejdi do stavu S2.
2. Zkontroluj požadované podmínky:
   a. Obj[i] = BB, obdrž korálek, i = i + 1, PKPB ≤ PKPB + 1, d = d + 1. Přejdi do stavu S2.
   b. Obj[i] != BB, vozidlo jeď do DP. Přejdi do stavu S4.
   c. i = 7. Přejdi do stavu S3.
   d. Obj[i] = 0, PKPB = 6. Přejdi do stavu S3.

3. Zkontroluj hodnotu proměnné „PKPB“:
    a. PKPB < 6, vozidlo jeď do další pozice. Přejdi do stavu S4.
    b. PKPB = 6, zastav vozidlo, vytiskni objednávka dokončena, t = 30s. Přejdi do stavu S5.
4. Pokud je dosaženo DP, zastav vozidlo. Přejdi do stavu S6.
5. Pokud je t = 0s, vozidlo jeď do PP, i = 0, PKPB = 0. Přejdi do stavu S1.
6. Zkontroluj hodnotu proměnné „d“:
    a. d < 10, obdržení barvy brány. Přejdi do stavu S2.
    b. d ≥ 10, vytiskni servis vozidla 0,5 min, t = 30s. Přejdi do stavu S7.
7. Pokud je t = 0s, obdržení barvy brány, d = 1. Přejdi do stavu S2.

6.4.3 Princip fungování vozidla – strategie C

Tato strategie využívá mnohem lepší optimalizaci pohybu vozidla, než je tomu u předchozích dvou strategií. V případě, že vozidlo dosáhne prvního ze zásobníků na dráze chytré továrny, dojde ke zkontrolování všech buněk pole objednávky. To znamená, že na jednu zastávku u daného zásobníku získá všechny korálky dané barvy, které se vyskytují v objednávce. Výsledkem této strategie je, že u každého ze zásobníků zastaví jen jednou.

6.4.3.1 Popis pracovního postupu

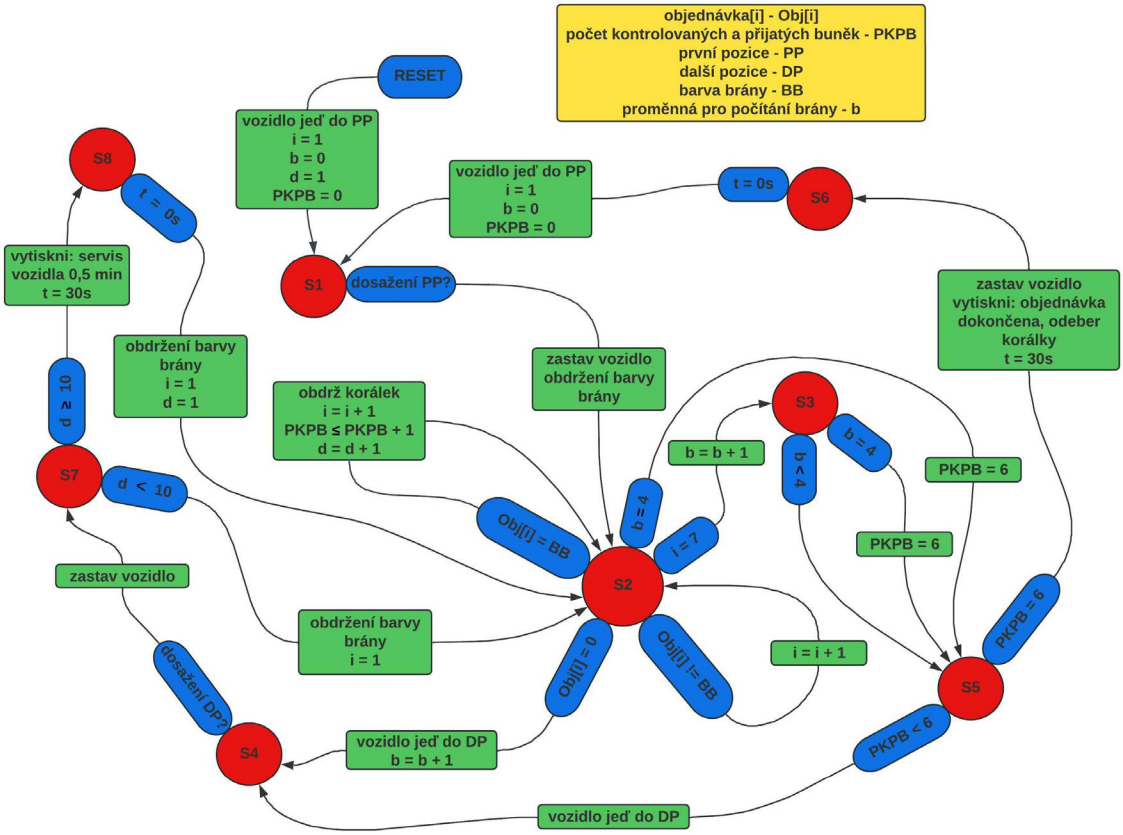
V první řadě je vozidlo nuceno se přesunout do první polohy, kde vozidlo zkontroluje barvu zásobníku na bráně se všemi buňkami v poli objednávek. V případě, že se barva korálku v bráně shoduje s požadovanou barvou v objednávce, korálek je vydán vozidlu a dále pokračuje s kontrolou další buňky pole. Tímto způsobem dochází ke kontrole všech buněk pole objednávky s prvním zásobníkem brány. Jestliže jsou zkontrolovány všechny buňky pole, dochází ke kontrole proměnné počtu kontrolovaných a přijatých buněk „PKPB“. V případě, že je těchto korálků méně než šest, vozidlo se přesune na další pozici zásobníku. Celá objednávka je tedy zkontrolovaná s barvou korálků v zásobníku a v případě potřeby je proveden výdej korálků. Takto celý cyklus pokračuje do doby, než vozidlo neposbírá všechny potřebné korálky definované v objednávce.

V případě, že se v objednávce vyskytuje hodnota „0“, je vozidlo automaticky odesláno na další pozici a zároveň dojde ke zvýšení proměnné „b“ o hodnotu „1“. Tím je ošetřen problém dokončení celé objednávky v případě, že není využita maximální

délka objednávky. Vozidlo tedy je nuceno projet všechny zásobníky s barevnými korálky. Objednávka je tedy dokončena tehdy, když proměnná „b“ je rovna hodnotě „4“.

Na tomto základě je navržen graf automatu (obr.23). Při resetu celého automatu je pole objednávek „i“ nastaveno na první pozici, hodnota počtu kontrolovaných a přijatých buněk „PKPB“ je vynulována, proměnná „b“ je nastavena na hodnotu „0“ a pohyb vozidla je nastaven do první pozice. Ve stavu „S1“ je testována podmínka o dosažení první pozice, poté se vozidlo zastaví, žádá informaci o obdržení barvy brány. Ve stavu „S2“ se testují podmínky, které dávají přehled o shodnosti barvy v zásobníku a objednávky. Ve správném případě dojde buď k obdržení korálku, nastavení na následující buňku pole objednávky a zvýšení hodnoty proměnné „PKPB“ o hodnotu „1“. V případě nesprávné barvy zásobníku s porovnáním v poli objednávky dojde ke přesunutí na následující buňku. Ve stavu „S2“ jsou implementovány podmínky pro přesunutí objednávky posledního pole, nalezení konce objednávky (hodnota „0“) a projetí všech zásobníků s korálky (hodnota „b = 4“).

V případě nalezení konce objednávky se vozidlo přesouvá do další pozice a dochází ke zvýšení proměnné „b“ o hodnotu „1“ (do stavu „S4“). Pokud jsou prohledány všechny buňky pole objednávky, automat přechází do stavu „S3“, kde se testuje velikost proměnné „b“. Jestliže je její hodnota menší než „4“, přechází automat do stavu „S5“, kde je pomocí podmínky o velikosti proměnné „PKPB“ menší než hodnota „6“ delegováno do další pozice se zásobníkem korálků. V případě, že je ve stavu „S3“ splněna podmínka velikosti proměnné „b = 4“, automat přiřadí proměnné počtu kontrolovaných a přijatých buněk „PKPB“ hodnotu „6“. V následujícím stavu je pak tedy splněna podmínka pro dokončení objednávky („PKPB = 6“). Dojde tedy k zastavení vozidla na 30 sekund a zobrazení zprávy o dokončení. Po uplynutí toho časového intervalu se vozidlo přesouvá do první pozice, pole objednávek „i“ je nastaveno na první pozici, hodnota proměnné „PKPB“ a „b“ je nastavena na „0“. Ve stavu „S7“ dochází pouze ke kontrole hodnoty proměnné „d“, která slouží k simulování případu servisu vozidla.



Obr. 23: Automat fungování vozidla – strategie C [zdroj: vlastní]

6.4.3.2 Posloupnost příkazů – strategie C

Pro funkčnost celého automatu vozidla se strategií C jsou sepsány příkazy poskytující informace, na které se vozidlo potřebuje soustředit v každé fázi. Vozidlo tyto příkazy plní krok po kroku a slouží při sestavení programu.

Reset.

Vozidlo jeď do PP, i = 1, b = 0, d = 1, PKPB = 0.

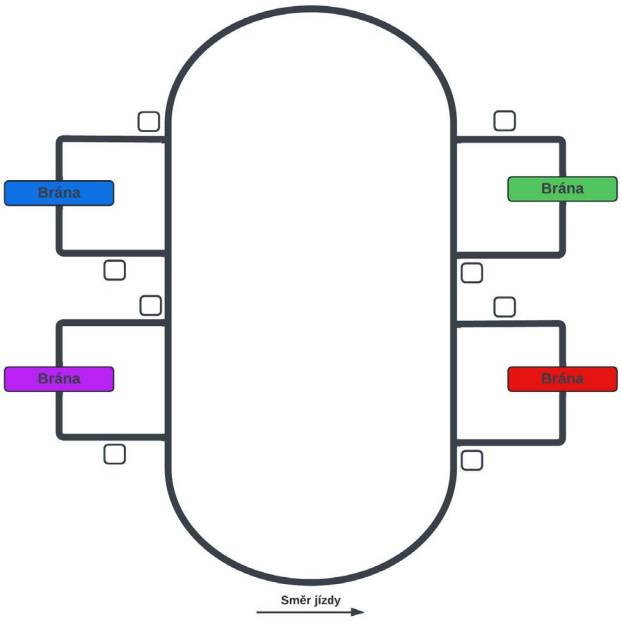
1. Pokud je dosažena PP, zastav vozidlo, obdržení barvy brány. Přejdi do stavu S2.
2. Zkontroluj požadované podmínky:
   a. Obj[i] = BB, obdrž korálek, i = i + 1, PKPB ≤ PKPB + 1, d = d + 1. Přejdi do stavu S2.
   b. Obj[i] != BB, i = i + 1. Přejdi do stavu S2.
   c. i = 7, b = b + 1. Přejdi do stavu S3.
   d. Obj[i] = 0, vozidlo jeď do DP, b = b + 1. Přejdi do stavu S4.
   e. b = 4, PKPB = 6. Přejdi do stavu S5.

3. Zkontroluj hodnotu proměnné „b“:
    a. b < 4, Přejdi do stavu S5.
    b. b = 4, PKPB = 6. Přejdi do stavu S5.
4. Pokud je dosaženo DP, zastav vozidlo. Přejdi do stavu S7.
5. Zkontroluj hodnotu proměnné „PKPB“:
    a. PKPB < 6, vozidlo jeď do DP. Přejdi do stavu S4.
    b. PKPB = 6, zastav vozidlo, vytiskni objednávka dokončena, t = 30s. Přejdi do stavu S6.
6. Pokud je t = 0s, vozidlo jeď do PP, i = 1, PKPB = 0, b = 0. Přejdi do stavu S1.
7. Zkontroluj hodnotu proměnné „d“:
    a. d < 10, obdržení barvy brány, i = 1. Přejdi do stavu S2.
    b. d ≥ 10, vytiskni servis vozidla 0,5 min, t = 30s. Přejdi do stavu S8.
8. Pokud je t = 0s, obdržení barvy brány, i = 1, d = 1. Přejdi do stavu S2.

6.5 Princip fungování vozidla s odbočením na trase

Při návrhu této varianty jsem vycházel z myšlenky o vytvoření odlišného tvaru dráhy v modelu chytré továrny. V předchozích případech totiž tvarem byl pouhý ovál, na kterém byly za sebou seřazeny jednotlivé brány se zásobníky barevných korálků.

Pro tento případ jsem navrhnul nový tvar trajektorie dráhy (obr.24). Na tomto obrázku je patrné, že je složen z oválu, na který jsou připojeny jednotlivé prvky pro možnost odbočení k zásobníku korálků dané barvy.



Obr. 24: Tvar dráhy pro princip vozidla s odbočením [zdroj: vlastní]

Odbočení je umožněno díky nalepenému čtverečku u každé odbočky. Díky senzorům, které jsou osazené na každém vozidle, je umožněno rozhodnutí pro zatočení vozidla nebo průjezdu rovně po hlavní trajektorii chytré továrny. Je nutné ovšem dodržet směr jízdy proti směru hodinových ručiček, jelikož v automatu jsou definovány akce, podle kterých vozidlo následuje buď pravý nebo levý okraj černé čáry trajektorie.

6.5.1 Princip fungování vozidla

Celý proces možnosti odbočení je založen na jednoduchém principu. Ten spočívá v pohybu vozidla při první jízdě po dané trajektorii v tom, že při prvním okruhu vozidlo zatočí na každé z možných výhybek. V okamžiku, kdy přijede k bráně se zásobníky, získá informaci o barvě brány a uloží si jí do paměti s proměnnou pro počítání odboček „z“. V případě shody požadované barvy korálku se zásobníkem na bráně se korálek obdrží a přejde na druhou buňku v poli objednávky. Tímto způsobem vozidlo projede celou dráhu a uloží si vždy požadovanou barvu zásobníku s příslušnou hodnotou proměnné „z“.

Po tomto prvním okruhu vozidlo zmapuje, na které odbočce se nachází daná barva brány se zásobníkem korálků. Vozidlo je tedy potom schopno při začátku druhého okruhu samo rozhodnout podle postupného prohledávání buněk pole objednávky, zdali odbočit nebo ne. Tímto způsobem vozidlo pokračuje do doby, než je objednávka dokončena. Tento stav nastane v případě, že hodnota pole objednávek je rovna „7“ nebo při postupném prohledávání objednávky narazí na konec objednávky symbolizovanou hodnotou „0“.

Dále je v tomto procesu implementováno simulování opravy vozidla, která je umožněna díky proměnné „d“. Ta se vždy zvýší o hodnotu „1“ pokaždé, kdy je obdržen korálek vozidlem. V případě, že je proměnná „d ≥ 10“, vozidlo zastaví a na displej zobrazí zprávu servis vozidla. Po uplynutí časového intervalu 30 sekund, nastaví proměnnou „d“ na hodnotu „1“ a vozidlo pokračuje dál v plnění objednávky.

6.5.1.1 Popis pracovního postupu

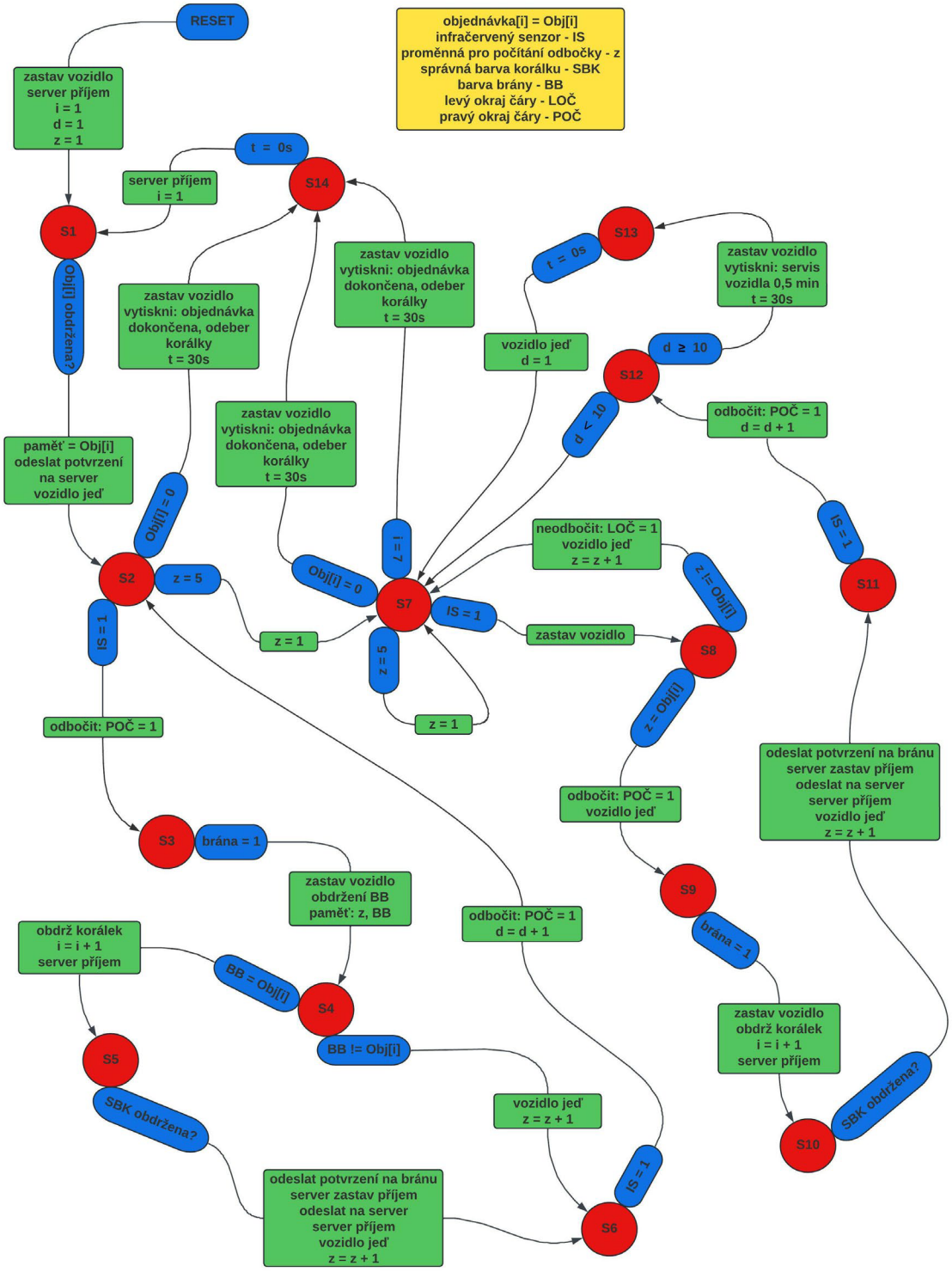
Pro znázornění je vytvořen stavový diagram (obr.25), který tento proces usnadňuje a zvyšuje jeho efektivitu. V první řadě při resetu celého automatu je vozidlo zastaveno, pole objednávek „i“ nastaveno na první pozici a proměnná „d“ a „z“ je nastavena na hodnotu „1“. Ve stavu „S1“ dochází ke kontrole podmínky obdržení

objednávky ze serveru na vozidlo. Při splnění dojde k uložení objednávky do paměti, odeslání potvrzení na server a vozidlo je uvedeno do pohybu. Poté automat přechází do dalšího stavu „S2“, kde kontroluje trojici podmínek. V případě, kdy hodnota infračerveného senzoru „IS = 1“, znamená že vozidlo přijelo na místo odbočení. V tom případě je vozidlu přikázáno sledovat pravý okraj čáry, který umožní vozidlu odbočit k bráně s prvním zásobníkem na cestě. Ve stavu „S3“ se kontroluje pouze příjezd vozidla k zásobníku. Po splnění přikáže vozidlu zastavit, obdrží barvu brány a následně si ji uloží do paměti společně s hodnotou proměnné „z“ pro číslování odbočky. Dále se ve stavu „S2“ vyskytuje podmínka pro symbolizování konce objednávky pomocí hodnoty „0“ a zároveň je kontrolovaná hodnota proměnné „z“, podle které vozidlo pozná, zda dokončilo celý cyklus určený k zjištění pořadí barevných bran se zásobníky korálků.

Ve stavu „S4“ se testují dvě podmínky a to, zdali se barva brány shoduje s požadovanou barvou v poli objednávky či nikoliv. V případě shody, vozidlo čeká na obdržení korálku a nastaví pole objednávek na další pozici. Ve stavu „S5“ čeká na potvrzení o získání správné barvy korálku a poté vozidlo odešle potvrzení na danou bránu, odešle potvrzení na server a zvýší hodnotu proměnné „z“ o hodnotu „1“. V případě, že se barva brány neshoduje s požadovanou barvou v poli objednávky, vozidlo se dá do pohybu a taktéž zvýší hodnotu proměnné „z“, která udává, že první brána na cestě s první odbočkou byla dosažena. Ve stavu „S6“ se testuje hodnota „IS = 1“, to umožní vozidlu odbočit díky sledování pravého okraje čáry. Automat se tímto způsobem opět vrací do stavu „S2“.

Když je dokončen první mapovací okruh, tak automat přejde do stavu „S7“ a zároveň proměnnou „z“ vrátí na hodnotu „1“. V tomto stavu už vozidlo ví, na kterých odbočkách se nachází požadovaná barva zásobníku. Kontroluje tedy hodnotu proměnné „i“ a výskyt hodnoty „0“ v objednávce. Tím je značeno, že vozidlo dokončilo svoji objednávku. Dále kontroluje hodnotu proměnné „z = 5“, v případě naplnění dojde k vrácení na hodnotu „1“ a automat se dostane zpět do stejného stavu. Poslední podmínka, kterou kontroluje v tomto stavu, je hodnota infračerveného senzoru „IS = 1“. V případě jejího naplnění se vozidlo zastaví a ve stavu „S8“ porovnává, zdali hodnota „z“ je v souladu s číslem v objednávce či nikoliv. V případě, že je potřeba odbočit z toho důvodu, že na dané odbočce se vyskytuje požadována barva korálku v objednávce, vozidlo následuje pravý okraj čáry a další postup je totožný jako v průjezdu prvního mapovacího okruhu. Jestliže se v objednávce nevyskytuje

požadovaná barva korálku, u odboček, ve které se nachází, dojde k rozhodnutí pokračovat rovně způsobem sledování levého okraje čáry. Proměnná „z“ je zvýšena o hodnotu „1“ aby vozidlo zjistilo, že bude následovat možnost odbočení k další barvě brány. Ve stavu „S12“ dochází pouze ke kontrole hodnoty proměnné „d“, která slouží k simulování případu servisu vozidla.



Obr. 25: Princip fungování vozidla s odbočením na trase [zdroj: vlastní]

6.5.1.2 Posloupnost příkazů

Pro funkčnost celého automatu vozidla jsou sepsány příkazy poskytující informace o činnostech, na které se vozidlo potřebuje soustředit v každé fázi. Vozidlo tyto příkazy plní krok po kroku a pomáhá při sestavení programu a jeho následném testování na modelu chytré továrny.

Reset.

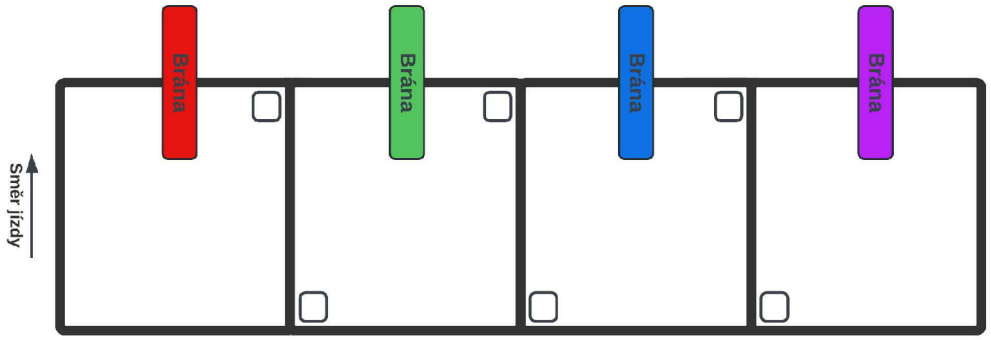
Zastav vozidlo, server příjem, i = 1, d = 1, z = 1.

1. Pokud je Obj[i] obdržena, paměť: Obj[i], odeslat potvrzení na server, vozidlo jeď. Přejdi do stavu S2.
2. Zkontroluj požadované podmínky:
    a. IS = 1, odbočit POČ = 1. Přejdi do stavu S3.
    b. z = 5, nastav z = 1. Přejdi do stavu S7.
    c. Obj[i] = 0, zastav vozidlo, vytiskni objednávka dokončena, t = 30s. Přejdi do stavu S14.
3. Pokud brána = 1, zastav vozidlo, obdržení BB, paměť: z, BB. Přejdi do stavu S4.
4. Zkontroluj požadované podmínky:
    a. BB = Obj[i], obdrž korálek, i = i + 1, server příjem. Přejdi do stavu S5.
    b. BB != Obj[i], vozidlo jeď, z = z + 1. Přejdi do stavu S6.
5. Pokud je SBK obdržena, odeslat potvrzení na bránu, server zastav příjem, odeslat na server, zapni příjem, vozidlo jeď, z = z + 1. Přejdi do stavu S6.
6. Pokud je IS = 1, odbočit POČ = 1, d = d + 1. Přejdi do stavu S2.
7. Zkontroluj požadované podmínky:
    a. IS = 1, zastav vozidlo. Přejdi do stavu S8.
    b. z = 5, nastav z = 1. Přejdi do stavu S7.
    c. Obj[i] = 0, zastav vozidlo, vytiskni objednávka dokončena, t = 30s. Přejdi do stavu S14.
    d. i = 7, zastav vozidlo, vytiskni objednávka dokončena, t = 30s. Přejdi do stavu S14.
8. Zkontroluj požadované podmínky:
    a. z = Obj[i], odbočit POČ = 1, vozidlo jeď. Přejdi do stavu S9.
    b. z != Obj[i], neodbočit LOČ = 1, vozidlo jeď, z = z + 1. Přejdi do stavu S7.

9. Pokud brána = 1, zastav vozidlo, obdrž korálek, i = i + 1, server příjem. Přejdi do stavu S10.
10. Pokud je SBK obdržena, odeslat potvrzení na bránu, server zastav příjem, odeslat na server, zapni příjem, vozidlo jeď, z = z + 1. Přejdi do stavu S11.
11. Pokud je IS = 1, odbočit POČ = 1, d = d + 1. Přejdi do stavu S12.
12. Zkontroluj hodnotu proměnné „d“:
    a. d < 10. Přejdi do stavu S7.
    b. d ≥ 10, vytiskni servis vozidla 0,5 min, t = 30s. Přejdi do stavu S13.
13. Pokud je t = 0s, vozidlo jeď, d = 1. Přejdi do stavu S7.
14. Pokud je t = 0s, server příjem, i = 1. Přejdi do stavu S1.

6.6 Princip fungování vozidla s možností vrácení se na trase

Při návrhu tohoto typu automatu jsem chtěl vytvořit možnost vrácení vozidla na trase po projetí jakékoliv brány se zásobníky barevných korálků. Pro tento případ jsem navrhnul tvar trajektorie dráhy (obr.26). Na tomto obrázku je patrné, že dráha obsahuje prvky pro zkrácení trasy vozidla do počáteční pozice.



Obr. 26: Dráha pro princip vozidla s možností vrácení na trase [zdroj: vlastní]

Možnost vrácení se na trase je jako u předchozí varianty umožněno díky nalepenému čtverečku u každé odbočky. Díky senzorům, které jsou osazené na každém vozidle, je umožněno rozhodnutí pro zatočení vozidla nebo průjezdu rovně. Je nutné ovšem dodržet směr jízdy po směru hodinových ručiček, jelikož v automatu jsou definovány akce, podle kterých vozidlo následuje buď pravý nebo levý okraj černé čáry trajektorie.

6.6.1 Princip fungování vozidla

Celý proces možnosti vrácení se na trase je založen na principu, který spočívá v tom, že vozidlo při první jízdě po dané trajektorii projede všechny výhybky rovně.
V okamžiku, kdy přijede k bráně se zásobníky, získá informaci o barvě brány a uloží si jí do paměti s proměnnou pro počítání odboček „z“. V případě shody požadované barvy korálku se zásobníkem na bráně se korálek obdrží a posune se na druhou buňku v poli objednávky. Tímto způsobem vozidlo projede celou dráhu a uloží si vždy požadovanou barvu zásobníku s příslušnou hodnotou proměnné „z“.

Po tomto prvním okruhu vozidlo zmapuje, na které odbočce se nachází daná barva brány se zásobníkem korálků. Vozidlo je tedy potom schopno při začátku druhého okruhu samo rozhodnout podle postupného prohledávání buněk pole objednávky, zdali odbočit nebo ne. Samotné vrácení na trase je rozhodnuto podle toho, jestli se na další pozici v objednávce nachází stejná barva korálku. Pokud ano odbočí, jinak pokračuje v jízdě rovně k dalšímu zásobníku. Vozidlo se vždy po odbočení vrací na úplný začátek dráhy, tedy před první pozici zásobníku. Tímto způsobem vozidlo pokračuje do doby, než je objednávka dokončena. Dále je v tomto procesu implementováno simulování opravy vozidla.

6.6.1.1 Popis pracovního postupu

Pro znázornění je vytvořen stavový diagram (obr.27). V první řadě při resetu celého automatu je vozidlo zastaveno, pole objednávek „i“ nastaveno na první pozici a proměnná „d“ a „z“ je nastavena na hodnotu „1“. Ve stavu „S1“ dochází ke kontrole podmínky obdržení objednávky ze serveru na vozidlo. Při splnění dojde k uložení objednávky do paměti, odeslání potvrzení na server a vozidlo je uvedeno do pohybu. Poté automat přechází do dalšího stavu, kde kontroluje podmínku, jestli vozidlo dorazilo k bráně se zásobníkem korálků. Po splnění přikáže vozidlu zastavit, obdrží barvu brány a následně si ji uloží do paměti společně s hodnotou proměnné „z“ pro číslování odbočky.

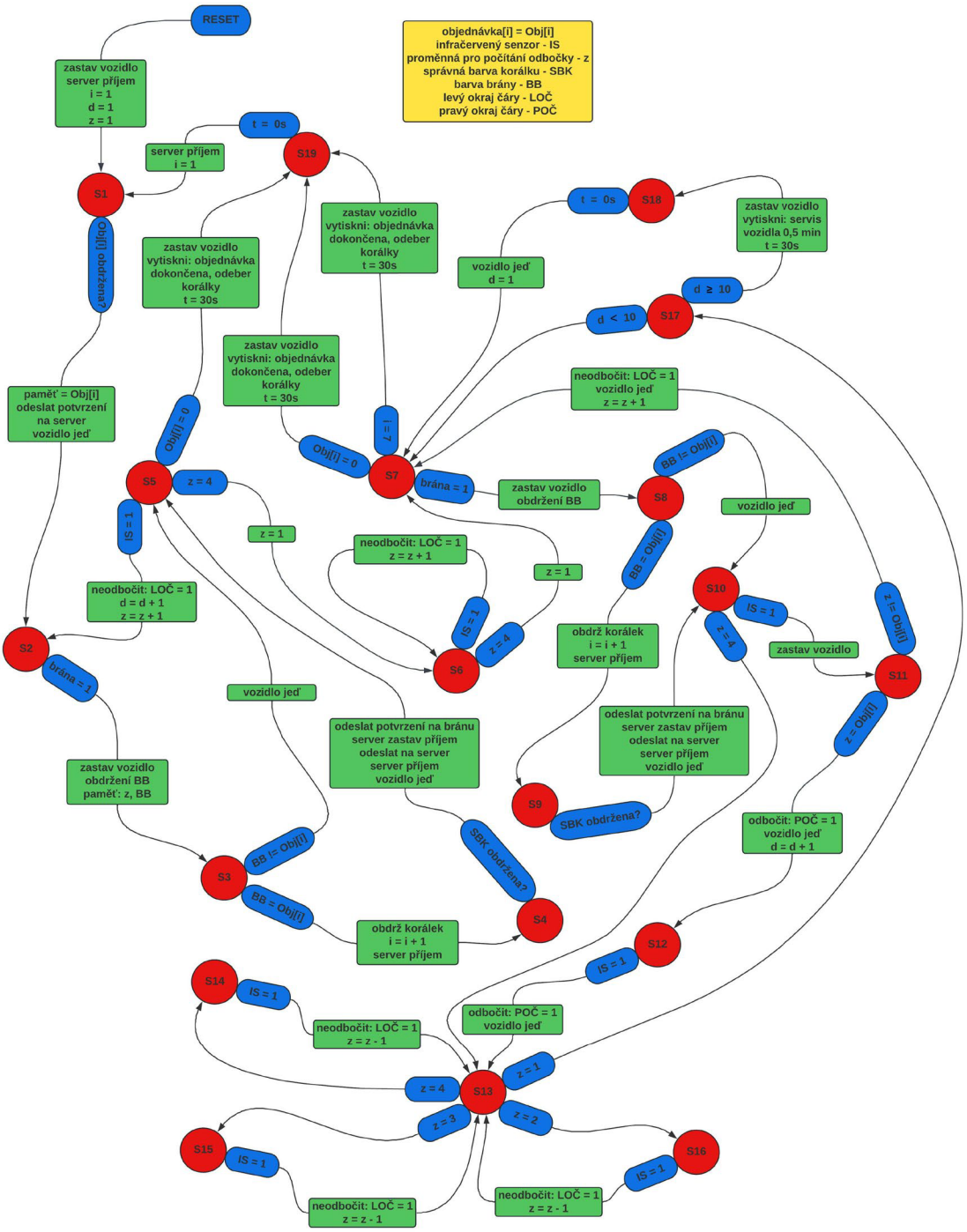
Ve stavu „S3“ se testují dvě podmínky a to, zdali se barva brány shoduje s požadovanou barvou v poli objednávky či nikoliv. V případě shody, vozidlo čeká na obdržení korálku a nastaví pole objednávek na další pozici. Ve stavu „S4“ čeká na potvrzení o získání správné barvy korálku a poté vozidlo odešle potvrzení na danou bránu a na server. V případě, že se barva brány neshoduje s požadovanou barvou v poli objednávky, vozidlo se dá do pohybu. Poté automat přechází do dalšího stavu

„S5“, kde kontroluje trojici podmínek. V případě, kdy hodnota infračerveného senzoru „IS = 1“, znamená že vozidlo přijelo na místo odbočení. V tom případě je vozidlu přikázáno sledovat levý okraj čáry, který umožní vozidlu projet rovně k bráně s dalším zásobníkem na cestě a zároveň dojde ke zvýšení proměnných „z“ a „d“ o hodnotu „1“. Automat se tímto způsobem opět vrací do stavu „S2“. Dále se ve stavu „S5“ vyskytuje podmínka pro symbolizování konce objednávky pomocí hodnoty „0“ a zároveň je kontrolovaná hodnota proměnné „z“, podle které vozidlo pozná, zda dokončilo celý cyklus určený k zjištění pořadí barevných bran se zásobníky korálků.

Když je dokončen první okruh, tak automat přejde do stavu „S6“ a zároveň proměnnou „z“ vrátí na hodnotu „1“. V tomto stavu dochází pouze ke kontrole podmínek „IS = 1“ a „z = 4“. Ty jsou tam z toho důvodu, aby vozidlo bylo schopné dojet na pozici před první barevnou bránu na dráze. V případě splnění podmínky „z = 4“ je nastavena hodnota opět na „1“ a automat přechází do stavu „S7“.

V tomto stavu už vozidlo ví, na kterých pozicích se nachází požadovaná barva zásobníku a je schopné se pro stejnou barvu korálku vrátit. Ve stavu „S7“ se kontroluje tedy hodnota proměnné „i“ a výskyt hodnoty „0“ v objednávce. Tím je značeno, že vozidlo dokončilo svoji objednávku. Dále také kontroluje podmínku, jestli vozidlo dorazilo k bráně se zásobníkem korálků. Po splnění přikáže vozidlu zastavit, vyžádá si barvu brány a přechází do stavu „S8“, kde porovnává, zdali je barva brány v souladu s číslem v objednávce či nikoliv. Pokud se barva korálku v objednávce neshoduje s barvou zásobníku, tak vozidlo odjíždí. V opačném případě je korálek obdržen a automat přechází do stavu „S9“, kde vyčkává na potvrzení o sebrání správné barvy korálku. Po splnění podmínky vozidlo odešle potvrzení na danou bránu a na server.

Ve stavu „S10“ dochází k testování podmínky „IS = 1“, po splnění dochází k zastavení vozidla. Zároveň se ve stejném stavu testuje i hodnota proměnné „z = 4“. V následujícím stavu dochází k porovnání hodnoty „z“ s číslem v objednávce. V případě, pokud hodnota není totožná tak vozidlo jede rovně, hodnota proměnné „z“ je zvýšena o hodnotu „1“ a automat přechází opět do stavu „S7“. V případě, že hodnota „z“ je totožná s číslem v objednávce, tak vozidlo odbočí a hodnota proměnné „d“ je zvýšena o hodnotu „1“. Následně odbočí doprava ještě jednou a automat se dostává do stavu „S13“, ve kterém se kontrolují hodnoty proměnné „z“. Tento stav je tu z toho důvodu, aby vozidlo bylo schopné přejet k prvnímu zásobníku s barevnými korálky. Ve stavu „S17“ dochází pouze ke kontrole hodnoty proměnné „d“, která slouží k simulování případu servisu vozidla.



Obr. 27: Princip fungování vozidla s možností vrácení se na trase [zdroj: vlastní]

6.6.1.2 Posloupnost příkazů

Pro funkčnost automatu vozidla jsou sepsány příkazy poskytující informace o činnostech, na které se vozidlo potřebuje soustředit v každé fázi. Vozidlo tyto příkazy plní krok po kroku a pomáhá při sestavení programu a jeho následném testování na modelu chytré továrny.

Reset.

Zastav vozidlo, server příjem, i = 1, d = 1, z = 1.

1. Pokud je Obj[i] obdržena, paměť = Obj[i], odeslat potvrzení na server, vozidlo jeď. Přejdi do stavu S2.
2. Pokud brána = 1, zastav vozidlo, obdržení BB, paměť: z, BB. Přejdi do stavu S3.
3. Zkontroluj požadované podmínky:
    a. BB = Obj[i], obdrž korálek, i = i + 1, server příjem. Přejdi do stavu S4.
    b. BB != Obj[i], vozidlo jeď. Přejdi do stavu S5.
4. Pokud je SBK obdržena, odeslat potvrzení na bránu, server zastav příjem, odeslat na server, zapni příjem, vozidlo jeď. Přejdi do stavu S5.
5. Zkontroluj požadované podmínky:
    a. IS = 1, neodbočit LOČ = 1, d = d + 1, z = z + 1. Přejdi do stavu S2.
    b. z = 4, nastav z = 1. Přejdi do stavu S6.
    c. Obj[i] = 0, zastav vozidlo, vytiskni objednávka dokončena, t = 30s. Přejdi do stavu S19.
6. Zkontroluj požadované podmínky:
    a. IS = 1, neodbočit LOČ = 1, z = z + 1. Přejdi do stavu S6.
    b. z = 4, nastav z = 1. Přejdi do stavu S7.
7. Zkontroluj požadované podmínky:
    a. brána = 1, zastav vozidlo, obdržení barvy brány. Přejdi do stavu S8.
    b. Obj[i] = 0, zastav vozidlo, vytiskni objednávka dokončena, t = 30s. Přejdi do stavu S19.
    c. i = 7, zastav vozidlo, vytiskni objednávka dokončena, t = 30s. Přejdi do stavu S19.
8. Zkontroluj požadované podmínky:
    a. BB = Obj[i], obdrž korálek, i = i + 1, server příjem. Přejdi do stavu S9.
    b. BB != Obj[i], vozidlo jeď. Přejdi do stavu S10.
9. Pokud je SBK obdržena, odeslat potvrzení na bránu, server zastav příjem, odeslat na server, zapni příjem, vozidlo jeď. Přejdi do stavu S10.
10. Zkontroluj požadované podmínky:
    a. IS = 1, zastav vozidlo. Přejdi do stavu S11.
    b. z = 4. Přejdi do stavu S13.

11. Zkontroluj požadované podmínky:
    a. z = Obj[i], odbočit POČ = 1, vozidlo jeď, d = d + 1. Přejdi do stavu S12.
    b. z != Obj[i], neodbočit LOČ = 1, vozidlo jeď, z = z + 1. Přejdi do stavu S7.
12. Pokud IS = 1, odbočit POČ = 1, vozidlo jeď. Přejdi do stavu S13.
13. Zkontroluj požadované podmínky:
    a. z = 1. Přejdi do stavu S17.
    b. z = 2. Přejdi do stavu S16.
    c. z = 3. Přejdi do stavu S15.
    d. z = 4. Přejdi do stavu S14.
14. Pokud je IS = 1, neodbočit LOČ = 1, z = z - 1. Přejdi do stavu S13.
15. Pokud je IS = 1, neodbočit LOČ = 1, z = z - 1. Přejdi do stavu S13.
16. Pokud je IS = 1, neodbočit LOČ = 1, z = z - 1. Přejdi do stavu S13.
17. Zkontroluj hodnotu proměnné „d“:
    a. d < 10. Přejdi do stavu S7.
    b. d ≥ 10, vytiskni servis vozidla 0,5 min, t = 30s. Přejdi do stavu S18.
18. Pokud je t = 0s, vozidlo jeď, d = 1. Přejdi do stavu S7.
19. Pokud je t = 0s, server příjem, i = 1. Přejdi do stavu S1.

6.7 Princip fungování vozidla s určením barev na dráze

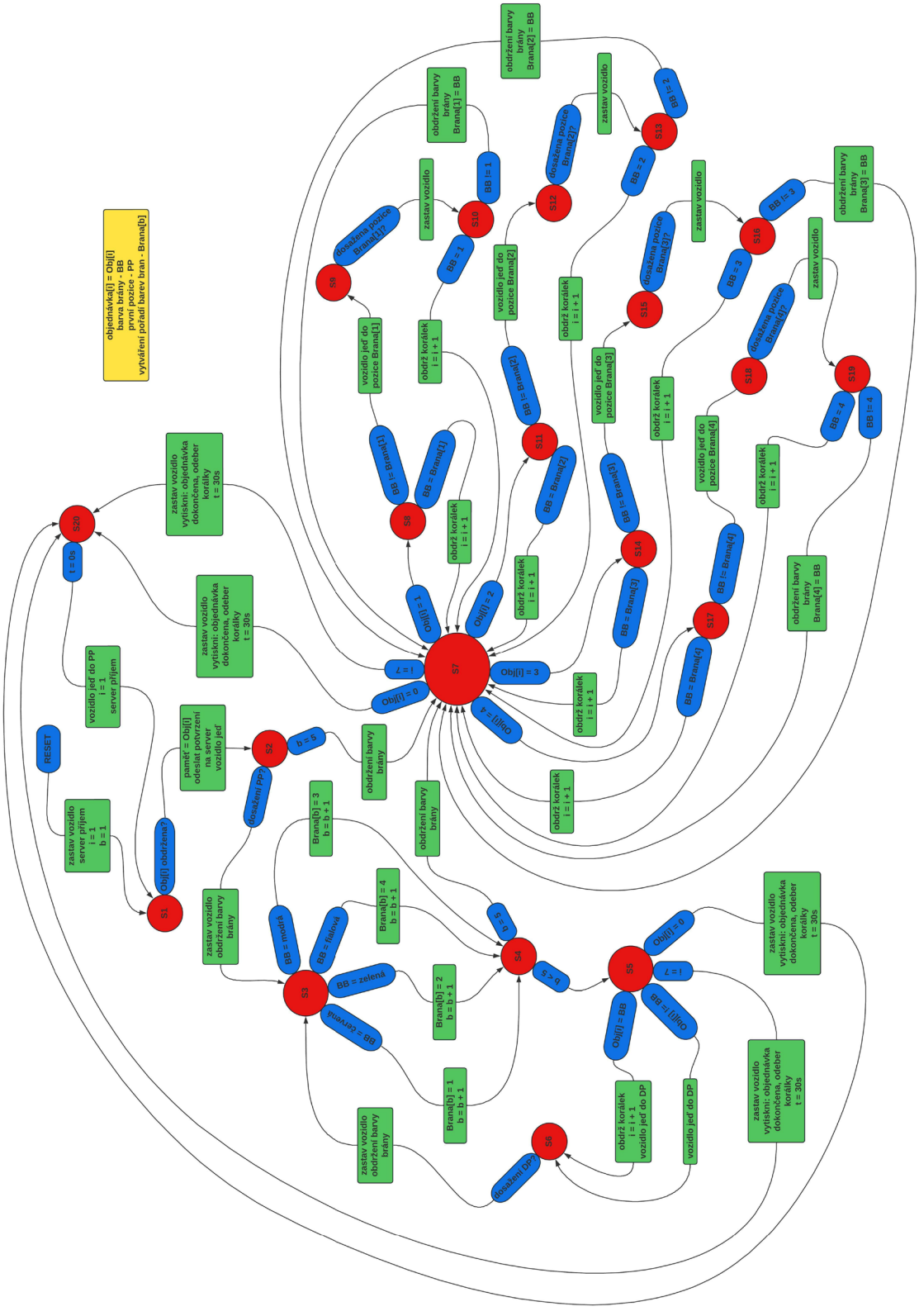
Při návrhu tohoto typu automatu jsem chtěl docílit zapisování barev zásobníku s korálky jako do mapy. Princip je navržen tak, aby dokázal při změně pořadí barevných zásobníků v průběhu pohybu vozidla opravit mapu. Po jejím vytvoření, vozidlo přesně ví, které barevné brány má před sebou a v jakém pořadí. Vozidlo je tedy schopné projet o několik bran dál a zastavit přímo u konkrétní barvy zásobníku s korálky podle definované objednávky. Daný princip je testován na trajektorii ve tvaru oválu.

6.7.1 Popis pracovního postupu

Pro znázornění je vytvořen stavový diagram (obr.28). V první řadě při resetu celého automatu je vozidlo zastaveno, pole objednávek „i“ a pole brány „b“ je nastaveno na první pozici. Ve stavu „S1“ dochází ke kontrole podmínky obdržení objednávky ze serveru na vozidlo. Při splnění dojde k uložení objednávky do paměti, odeslání potvrzení na server a vozidlo je uvedeno do pohybu. Poté automat přechází do dalšího stavu, kde kontroluje podmínku, jestli vozidlo dorazilo do první pozice. Po

splnění přikáže vozidlu zastavit a obdrží barvu brány. Ve stavu „S3“ dochází ke kontrole čtveřice podmínek, které porovnávají, zdali se nachází u červené, zelené, modré nebo fialové barvy brány. Po splnění jakékoliv z podmínek dojde k uložení příslušné barvy zásobníku s korálky do pole brány (červená = 1, zelená = 2, modrá = 3, fialová = 4) a přesune se na další pozici v poli. Ve stavu „S4“ dochází k porovnávání proměnné „b“. Pokud je menší než „5“ tak přechází do dalšího stavu „S5“, kde dochází k porovnávání barvy brány s objednávkou. V případě shody je vydán korálek, pozice v poli objednávky přesunuta na další pozici a vozidlo jede do další pozice. Jestliže se barva brány neshoduje s barvou v objednávce, vozidlo odjíždí do další pozice. Ve stavu „S6“ se testuje pouze dosažení další pozice, poté vozidlo zastaví a čeká na obdržení barvy brány. Automat potom přechází zpět do stavu „S3“.

Pokud je ve stavu „S2“ a „S4“ splněna podmínka, že se proměnná „b“ rovná hodnotě „5“, přechází se do stavu „S7“. Hodnota „5“ signalizuje, že je vytvořena mapa s pořadím barevných bran s korálky. Ve stavu „S7“ dochází k porovnávání hodnot v objednávce. Pokud je hodnota rovna „1“, přechází do dalšího stavu, kde dochází k porovnávání barvy brány s hodnotou v poli brány. Jestliže se shoduje, tak obdrží korálek a vrací se zpět do stavu „S7“. V případě, že se barva brány neshoduje s hodnotou v poli brány, tak vozidlo je vysláno do pozice s požadovanou barvou zásobníku korálků. Ve stavu „S10“ se testují dvě podmínky, které v případě rovnosti barvy brány s hodnotou „1“, obdrží korálek a vrací se zpět do stavu „S7“ s tím, že se pole objednávky přesune na další pozici. V opačném případě dojde k přepsání pořadí barevných zásobníků v poli brány. U ostatních barev korálků je postupováno podle stejného principu. Automat končí v případě, že je hodnota proměnné „i“ rovna hodnotě „7“ nebo je v objednávce číslo „0“, které symbolizuje konec objednávky.



Obr. 28: Princip fungování vozidla s určením barev na dráze [zdroj: vlastní]

6.7.1.1 Posloupnost příkazů

Pro funkčnost automatu vozidla jsou sepsány příkazy poskytující informace o činnostech, na které se vozidlo potřebuje soustředit v každé fázi. Vozidlo tyto příkazy

plní krok po kroku.

Reset.

Zastav vozidlo, server příjem, i = 1, b = 1.

1. Pokud je Obj[i] obdržena, paměť: Obj[i], odeslat potvrzení na server, vozidlo jeď. Přejdi do stavu S2.
2. Zkontroluj požadované podmínky:
    a. Dosažena PP, zastav vozidlo, obdržení barvy brány. Přejdi do stavu S3.
    b. b = 5, obdržení barvy brány. Přejdi do stavu S7.
3. Zkontroluj požadované podmínky:
    a. BB = červená, Brana[b] = 1, b = b + 1. Přejdi do stavu S4.
    b. BB = zelená, Brana[b] = 2, b = b + 1. Přejdi do stavu S4.
    c. BB = modrá, Brana[b] = 3, b = b + 1. Přejdi do stavu S4.
    d. BB = fialová, Brana[b] = 4, b = b + 1. Přejdi do stavu S4.
4. Zkontroluj hodnotu proměnné „b“:
    a. b = 5, obdržení barvy brány. Přejdi do stavu S7.
    b. b < 5. Přejdi do stavu S5.
5. Zkontroluj požadované podmínky:
    a. Obj[i] = BB, obdrž korálek, i = i + 1, vozidlo jeď do další pozice. Přejdi do stavu S6.
    b. Obj[i] != BB, vozidlo jeď do další pozice. Přejdi do stavu S6.
    c. Obj[i] = 0, zastav vozidlo, vytiskni objednávka dokončena, t = 30s. Přejdi do stavu S20.
    d. i = 7, zastav vozidlo, vytiskni objednávka dokončena, t = 30s. Přejdi do stavu S20.
6. Pokud je dosažena DP, zastav vozidlo, obdržení barvy brány. Přejdi do stavu S3.
7. Zkontroluj požadované podmínky:
    a. Obj[i] = 1. Přejdi do stavu S8.
    b. Obj[i] = 2. Přejdi do stavu S11.
    c. Obj[i] = 3. Přejdi do stavu S14.
    d. Obj[i] = 4. Přejdi do stavu S17.
    e. Obj[i] = 0, zastav vozidlo, vytiskni objednávka dokončena, t = 30s. Přejdi do stavu S20.
    f. i = 7, zastav vozidlo, vytiskni objednávka dokončena, t = 30s. Přejdi

do stavu S20.

8. Zkontroluj požadované podmínky:
    a. BB = Brana[1], obdrž korálek, i = i + 1. Přejdi do stavu S7.
    b. BB != Brana[1], vozidlo jeď do pozice Brana[1]. Přejdi do stavu S9.
9. Pokud je dosažena pozice Brana[1], zastav vozidlo. Přejdi do stavu S10.
10. Zkontroluj požadované podmínky:
    a. BB = 1, obdrž korálek, i = i + 1. Přejdi do stavu S7.
    b. BB != 1, obdržení barvy brány, Brana[1] = BB. Přejdi do stavu S7.
11. Zkontroluj požadované podmínky:
    a. BB = Brana[2], obdrž korálek, i = i + 1. Přejdi do stavu S7.
    b. BB != Brana[2], vozidlo jeď do pozice Brana[2]. Přejdi do stavu S12.
12. Pokud je dosažena pozice Brana[2], zastav vozidlo. Přejdi do stavu S13.
13. Zkontroluj požadované podmínky:
    a. BB = 2, obdrž korálek, i = i + 1. Přejdi do stavu S7.
    b. BB != 2, obdržení barvy brány, Brana[2] = BB. Přejdi do stavu S7.
14. Zkontroluj požadované podmínky:
    a. BB = Brana[3], obdrž korálek, i = i + 1. Přejdi do stavu S7.
    b. BB != Brana[3], vozidlo jeď do pozice Brana[3]. Přejdi do stavu S15.
15. Pokud je dosažena pozice Brana[3], zastav vozidlo. Přejdi do stavu S16.
16. Zkontroluj požadované podmínky:
    a. BB = 3, obdrž korálek, i = i + 1. Přejdi do stavu S7.
    b. BB != 3, obdržení barvy brány, Brana[3] = BB. Přejdi do stavu S7.
17. Zkontroluj požadované podmínky:
    a. BB = Brana[4], obdrž korálek, i = i + 1. Přejdi do stavu S7.
    b. BB != Brana[4], vozidlo jeď do pozice Brana[4]. Přejdi do stavu S18.
18. Pokud je dosažena pozice Brana[4], zastav vozidlo. Přejdi do stavu S19.
19. Zkontroluj požadované podmínky:
    a. BB = 4, obdrž korálek, i = i + 1. Přejdi do stavu S7.
    b. BB != 4, obdržení barvy brány, Brana[4] = BB. Přejdi do stavu S7.
20. Pokud je t = 0s, vozidlo jeď do PP, i = 1, server příjem. Přejdi do stavu S1.

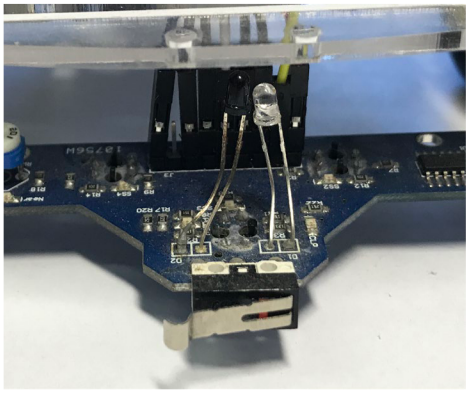
6.8 Bezpečnostní prvky provozu vozidel

Každé vozidlo je vybaveno pomocí komponentů, které zabraňují střetu více vozidel při provozu modelu chytré továrny. Konkrétně se jedná a dva komponenty, kterými jsou infračervený senzor (IS) a mechanický přední spínač (MPS). Ty jsou upevněny před vozidlem tak, aby byly schopny identifikovat jiné vozidlo nebo jakoukoliv překážku před ním. V případě, že některý z těchto komponentů nalezne překážku, vozidlo se automaticky zastaví a tím tedy dojde k zabránění nehody.

Vozidlo je také iniciováno pomocí proměnné „bool free“, která slouží pro registraci vozidla bez pracovního vytížení a je k dispozici pro přijetí objednávky. Proměnná „int battery“ slouží při kontrole úrovně kapacity baterie, která je přiřazena v automatu distribuce objednávky v předchozí kapitole (6.2).

6.8.1 Komponenty pro ochranu proti překážkám

U těchto dvou komponentů (obr. 29) jsou kontrolovány jejich odezvy, v případě, že je vozidlo v pohybu. Jestliže tyto dva komponenty hlásí hodnotu „0“, tak servomotor může být poháněn a tím pádem se vozidlo může volně pohybovat.



Obr. 29: IS senzor a mechanický přední spínač [zdroj: vlastní]

V případě, že některá z těchto součástí hlásí hodnotu „1“, vozidlo se automaticky zastaví a tím pádem dojde k zabránění nehody s jakoukoliv překážkou. Tyto úpravy se vkládají do systému s implementací těchto komponentů do vozidla a zápisem seznamu příkazů, které vozidlo musí plnit, když se pohybuje.

6.8.1.1 Popis pracovního postupu

Jedná se o jednoduchý princip, kdy v případě, že je vozidlu dán příkaz k pohybu, dochází ke kontrole obou hodnot vrácených z jednotlivých komponentů. Tímto způsobem dochází k rozhodnutí o možnosti pohybu vozidla či nikoliv. Vše je umožněno

pomocí proměnné „mohu_jet“, která rozhoduje o pohybu nebo zastavení. V případě, že je proměnná „mohu_jet = 1“, vozidlo se může pohybovat a v okamžiku, kdy hodnota „mohu_jet = 0“, vozidlo zastaví.

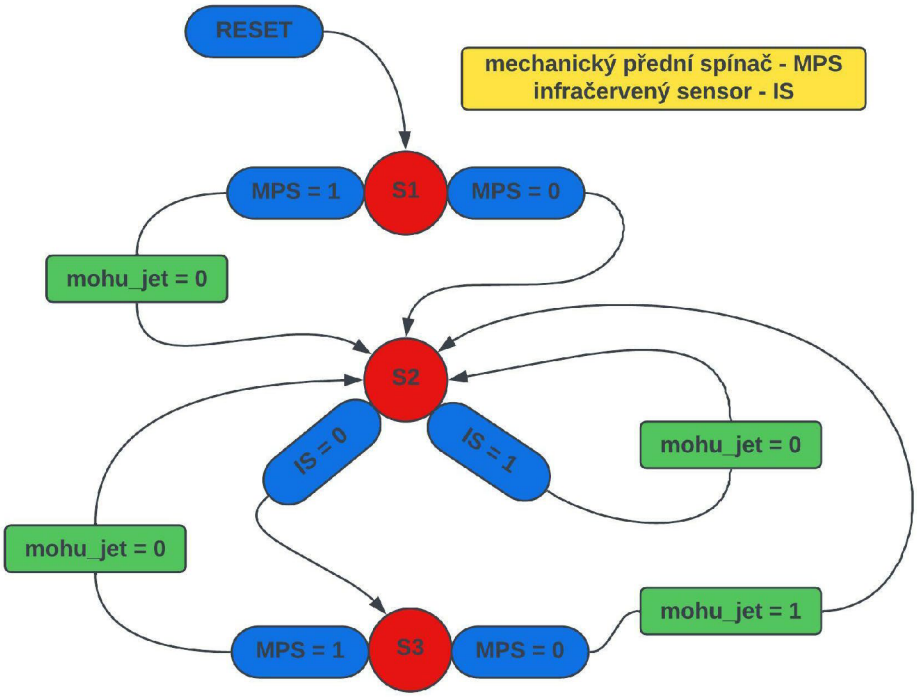
Tab. 1: Rozhodování proměnné mohu_jet

IS	MPS	Proměnná mohu_jet
0	0	1
0	1	0
1	0	0
1	1	0

Komponenty mechanického předního spínače a infračerveného senzoru jsou kontrolovány, díky tomu dochází k rozhodnutí o proměnné „mohu_jet, která umožňuje vozidlu pohyb nebo zastavení. Tabulka 1 ukazuje jasnou představu o odezvě nastavení proměnné „mohu_jet“ na hodnotu „0“ nebo „1“.

Pro znázornění pracovního postupu je vytvořen stavový diagram (obr. 30). Automat je navržen tak, že pouze v případě, kdy oba komponenty hlásí hodnotu „0“, je proměnná „mohu_jet“ nastavena na „1“. Jedná se o uzavřený automat a běží, když vozidlo spustí funkci „going()“. To pomáhá vozidlu vyhnout se srážce.

Ve stavu „S1“ je kontrolována hodnota proměnné „MPS“. V případě, že přepínač vrátí hodnotu „1“ tak je proměnná „mohu_jet“ nastavena na „0“. Ve stavu „S2“ dochází ke kontrole odezvy infračerveného senzoru „IS“. Jestliže senzor vrátí hodnotu „1“, tak proměnná „mohu_jet“ je nastavena na „0“. V okamžiku, kdy senzor vrátí hodnotu „0“, automat přechází do stavu „S3“. V tomto stavu dochází opět ke kontrole proměnné „MPS“. V případě, že spínač vrátí hodnotu „0“, tak proměnná „mohu_jet“ je nastavena na „1“, v opačném případě je hodnota proměnné „mohu_jet“ nastavena na „0“. Automat se poté vrací zpět do druhého stavu, kde opět kontroluje vrácenou hodnotu senzoru.



Obr. 30: Automat pro ochranu proti kolizím vozidla [zdroj: vlastní]

6.8.1.2 Posloupnost příkazů

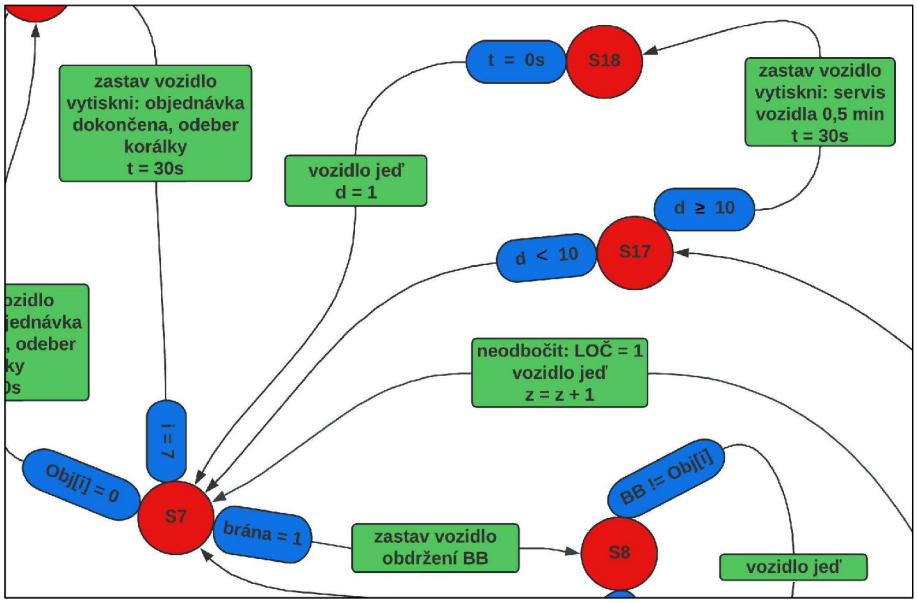
Na základě popisu a obrázku 30 je jasná představa o podmínkách, které je třeba dodržet v každém kroku. Pro tento případ jsou napsány následující příkazy:

Reset.

1. Zkontroluj vrácenou hodnotu „MPS“:
    a. MPS = 1, mohu_jet = 0. Přejdi do stavu S2.
    b. MPS = 0. Přejdi do stavu S2.
2. Zkontroluj vrácenou hodnotu „IS“:
    a. IS = 1, mohu_jet = 0. Přejdi do stavu S2.
    b. IS = 0. Přejdi do stavu S3.
3. Zkontroluj vrácenou hodnotu „MPS“:
    a. MPS = 1, mohu_jet = 0. Přejdi do stavu S2.
    b. MPS = 0, mohu_jet = 1. Přejdi do stavu S2.

7 KOMPONENTY INTERNET OF SERVICES

V zadání diplomové práce jsem měl za úkol navrhnout komponenty Internet of Services. V každém vytvořeném grafu automatu pro fungování vozidla je tedy vytvořen princip, kterým se dané vozidlo dostane do stavu, kdy je potřebné vozidlu udělat údržbu. Vše je založeno na jednoduchém principu, kdy se porovnává hodnota proměnné „d“ (obr.31). V případě hodnoty proměnné „d ≥ 10“, vozidlo se zastaví a zobrazí zprávu o servisu vozidla. Tato zpráva se ukáže na LCD displeji příslušného vozidla, ale taktéž v mobilní aplikaci Blynk. Tímto způsobem byl navržený mechanismus Internet of Services, kde žádost o servis je generována na základě počtu sesbíraných korálků.



Obr. 31: Mechanismus IoS pro generování žádosti o servis [zdroj: vlastní]

Dále byl do systému přidán monitoring průběhu procesu na mobilní telefon. Vozidlo může požádat o servis po deseti obdržených korálcích nebo pěti uskutečněných objednávkách. Kvůli tomu je vytvořena proměnná „IoScount“, která je na počátku nastavena na hodnotu „0“ a dále se zvyšuje. V případě dosažení stanovené hranice počtu objednávek, vozidlo zašle zprávu serveru. Samotný server posílá zprávu na mobilní telefon.

To samé funguje pro bránu se zásobníky korálků. V případě, že brána požádá o servis vozidla po vydání deseti korálků, posílá zprávu serveru a ten přeposílá zprávu do mobilního telefonu.

Program pro server

V sekci před funkcí void setup ():

Pro propojení s widgetem v mobilní aplikaci Blynk (obr. 32) je potřebné definovat virtuální pin pomocí příkazu WidgetLCD lcdPhoneIoS(V16).

```
LiquidCrystal_I2C lcd(0x27, 20, 4);      // LCD 20x4
ESP8266 wifi(&EspSerial);                // blynk
BlynkTimer timer;
WidgetLCD lcdPhone(V15);
WidgetLCD lcdPhoneIoS(V16);               //IoS
RF24 myRadio(49, 53);                     //WIFI

byte addresses[][6] = {"0"};              //WIFI
struct package {
  //char text[20]="000000";
  int Order[6]={0,0,0,0,0,0};
  int c=1;
```

Obr. 32: Propojení s widgetem na mobilním telefonu [zdroj: vlastní]

V rámci funkce void setup ():

Bezprostředně po začátku napíše hlášení „No service“ jelikož není naplněna ani jedna podmínka pro vyžádání servisu vozidla (obr.33).

```
// Wifi settings and listening
myRadio.begin();                          //WIFI NRF
myRadio.setChannel(0x60);
myRadio.setPALevel(RF24_PA_MAX);
//myRadio.setDataRate( RF24_250KBPS );
printLcd(0,"Server-order beads");
printLcd3();
sendMessageToPhone3();
lcdPhoneIoS.clear();                      //IoS
lcdPhoneIoS.print(0, 0, "No service");    //IoS
}
```

Obr. 33: Hlášení při začátku první objednávky vozidla [zdroj: vlastní]

V rámci funkce void loop ():

V případě, že přijde Wi-Fi zpráva od libovolného zařízení s adresou „tx“, přeposílá zprávu na server a ten pošle zprávu na telefon (obr.34).

```
lcdPhoneIoS.clear();                          //IoS
lcdPhoneIoS.print(0, 0, "Service request");   //IoS
lcdPhoneIoS.print(1, 0, "device");            //IoS
lcdPhoneIoS.print(0, 7, dataReceive.tx);      //IoS
Sb=1;
```

Obr. 34: Hlášení v případě příchozí Wi-Fi zprávy [zdroj: vlastní]

Program pro vozidlo

V sekci před funkcí void setup ():

V programu je potřebné vytvořit proměnnou pro počítání objednávek „IoScount“ (obr.35).

```
int S=0;                        //state of vehicle´s main automaton and arrow to S0
int lastcl = 0;                 //aux var for wifi comm
int i=1;                        // actual position of needed bead in order
int Obj[6]={0,0,0,0,0,0};       //order
//int Obj[6]={1,3,2,3,1,2};     //demo order
int p=0;
//int Path[5]={0,1,2,3,2};      //demo map of stacks
int Path[5]={0,0,0,0,0};        //map of stacks

int IoScount = 0;               //IoS
```

Obr. 35: Vytvoření proměnné IoScount [zdroj: vlastní]

V rámci funkce void loop ():

Jestliže je proměnná „IoScount“ větší nebo rovna počtu pěti dokončených objednávek, nastaví proměnnou zpět na hodnotu „0“. Tato proměnná je obsažena ve funkci „OrderFinished()“ (obr. 36).

```
void OrderFinished()
   {                    IoScount=Ioscount+1;
                        if (IoScount>=5) {IoScount=0;
                                          myRadio.stopListening();              //WIFI NRF TRANSMIT
                                          dataTransmit.c=dataTransmit.c+1;
                                          dataTransmit.tx=2;
                                          dataTransmit.rx=1;
                                          dataTransmit.val=5;
                                          myRadio.openWritingPipe(addresses[0]);
                                          myRadio.write(&dataTransmit, sizeof(dataTransmit));
                                          };
                        LedOrder();
                        printLcd(0,"Vehicle-finished");
```

Obr. 36: Funkce OrderFinished() s proměnnou IoScount [zdroj: vlastní]

8 REALIZACE ŘÍDÍCÍCH STRATEGIÍ

Testovány byly tři řídící strategie (A, B, C), které jezdí na oválné trase s jedním vozidlem. Principy těchto strategií jsou detailně popsány v předchozích kapitolách (viz 6.4.1, 6.4.2, 6.4.3). Dané strategie spočívají v odlišných způsobech sbírání korálků. Strategie A sebere vždy jeden korálek požadované barvy a jede k další bráně kde porovná barvu s objednávkou. Podle toho se rozhodne o tom, zdali korálek sebere nebo jede do další pozice.

Strategie B nabízí vylepšený postup při kontrole požadované barvy. Při sebrání prvního korálku následně zkontroluje barvu na dalším poli objednávky. Pokud je stejná jako barva zásobníku u které se vozidlo nachází, dojde k vydání korálku. Tímto způsobem vozidlo pokračuje do doby, kdy je objednávka dokončena.

Obr. 37: Testování řídících strategií na modelu chytré továrny [zdroj: vlastní]

V případě strategie C vozidlo zkontroluje celou objednávku u každého ze zásobníků s barevnými korálky. Objednávka je tedy dokončena na jedno projetí každé brány.

8.1 Programování serveru

Server je naprogramován tak, aby přijímal objednávku od uživatele a přenášel ji do vozidla. V modelu chytré továrny tedy funguje jako prostředník mezi mobilní aplikací a vozidlem. V programu serveru běží dva automaty zároveň (pro tlačítka a automat serveru).

V sekci před funkcí void setup ():

Import knihoven

V první řadě je potřebné do programu naimportovat potřebné knihovny pro funkčnost LCD displejů a Wi-Fi připojení pomocí ESP modulů pro propojení aplikace Blynk s Arduinem. Z oficiálních stránek aplikace Blynk je zapotřebí stáhnout dvě knihovny a implementovat je do kódu (ESP8266_Lib.h, BlynkSimpleShieldEsp8266.h).

```
#define BLYNK_PRINT Serial                  // blynk
#include <ESP8266_Lib.h>                    // blynk
#include <BlynkSimpleShieldEsp8266.h>       // blynk
#include <SPI.h>                            // WIFI
#include "RF24.h"                           // WIFI
#include <Wire.h>                           // LCD
#include <LiquidCrystal_I2C.h>              // LCD

#define EspSerial Serial2                   // blynk
#define ESP8266_BAUD 115200                 // blynk
```

Obr. 38: Import knihoven [zdroj: vlastní]

Inicializace proměnných pro server

Všechny potřebné vstupní proměnné tlačítek jsou inicializovány na nulu. Zmíněná barevná tlačítka pro červenou, zelenou, modrou a fialovou jsou inicializovány jako „tR“, „tG“, „tB“, „tP“. Tlačítka vymazat a odeslat jsou inicializována jako „tClear“ a „tFinish“. Hlavní stav automatu serveru jako „Ss“ a stav tlačítek automatu je „Sb“.

```
int tR=0;  //proměnné pro ovládání tlačítek Blynk
int tG=0;
int tB=0;
int tP=0;
int tClear=0;
int tFinish=0;
```

Obr. 39: Inicializace počátečních proměnných [zdroj: vlastní]

V rámci funkce void setup ():

Funkce „void setup()“ v programu proběhne pouze jednou (obr.40). V této funkci je naprogramováno nastavení LCD displeje a zároveň modulu ESP8266 pro fungování Wi-Fi komunikace. Příkaz "EspSerial.begin" spouští komponenty samotného ESP modulu. Při propojování aplikace Blynk s Arduinem je zapotřebí znát autorizační token (WIFI_AUTH), název sítě (WIFI_SSID) a odpovídající heslo pro přístup. Poté je server nastavený tak, aby poslouchal ovládácí aplikaci Blynk.

```
void setup() {
  pinMode(12, INPUT_PULLUP);    // button for stepping
  Serial.begin(9600);
// LCD settings
  lcd.init();//  lcd.begin();   //begin or init depends on LCD library
  lcd.backlight();
  lcd.clear();
  printLcd(0,"Server-initializing");
  printLcd(1,"TX:");
  printLcd(2,"RX:");
// Wifi settings
  EspSerial.begin(ESP8266_BAUD);
  delay(1000);
  Blynk.begin(WIFI_AUTH, wifi, WIFI_SSID, WIFI_PASSWORD);
  //timer.setInterval(1000L, sendSensor);     //timer of blynk

  // Wifi settings and listening
  myRadio.begin();              //WIFI NRF
  myRadio.setChannel(0x60);
  myRadio.setPALevel(RF24_PA_MAX);
  //myRadio.setDataRate( RF24_250KBPS );
  printLcd(0,"Server-order beads");
  printLcd3();
  sendMessageToPhone3();
}
```

Obr. 40: Funkce serveru "void Setup()" [zdroj: vlastní]

V rámci funkce void loop ():

```
void loop() {
  Blynk.run();
  //timer.run();                  //timer of blynk
  if (Ss==1){                     //Ss1-creating order on the phone
            //SMofSb();           //SerialMonitor: debugging blynk buttons and Sb
            SbAutomaton();
            }
  }
```

Obr. 41: Funkce serveru "void loop()" [zdroj: vlastní]

Funkce „void loop ()“ vykonává hlavní nekonečnou smyčku programu. Dochází zde k spouštění aplikace Blynk a funkce "SbAutomaton()" spouští program automatu pro tlačítka.

Funkce v serverovém programu

Funkcí „readCodeFromWifi()“ dochází ke čtení kódu při přijetí dat. Pro odeslání zpráv a vytisknutí objednávky do telefonu jsou dvě funkce „sendMessageToPhone“ a „sendMessageToPhone3“. Dále je potřeba pro načítání a aktualizaci hodnot virtuálních pinů uvést makro „BLYNK_WRITE“, které je určeno pouze pro virtuální piny (obr.42).

```
int readCodeFromWifi() {          // listens wifi and puts recieved code to dataRecieve
  while (myRadio.available()) {
    myRadio.read(&dataRecieve, sizeof(dataRecieve));
  }
}

void sendMessageToPhone(String message) {  // sends message to phone
  lcdPhone.clear();
  lcdPhone.print(0, 0, message);
}

void sendMessageToPhone3() {  // sends message line3 to phone
  lcdPhone.clear();
  lcdPhone.print(0, 0, "Obj:"+String(Obj[1])+String(Obj[2])+String(Obj[3])+String(Obj[4])+String(Obj[5])+String(Obj[6]));
}

void printLcd(int line, String message) {   //prints message at N line
  lcd.setCursor(0, line);
  lcd.print("                ");
  lcd.setCursor(0, line);
  lcd.print(message);
}

void printLcd3() {
  printLcd(3,"Ss"+String(Ss)+" Order:"+String(Obj[1])+String(Obj[2])+
  String(Obj[3])+String(Obj[4])+String(Obj[5])+String(Obj[6])+" o"+String(o)+"p"+String(p));
}

//    sync with Blynk
BLYNK_WRITE(V0) {bR = param.asInt();}
BLYNK_WRITE(V1) {bG = param.asInt();}
BLYNK_WRITE(V2) {bB = param.asInt();}
BLYNK_WRITE(V3) {bP = param.asInt();}
BLYNK_WRITE(V4) {bClear = param.asInt();}
BLYNK_WRITE(V5) {bFinish  = param.asInt();}
```

Obr. 42: Funkce v serverovém programu [zdroj: vlastní]

Stav Sb1 ve funkci „void SbAutomaton()“

Na obrázku 43 se nachází první stav tlačítek automatu pro tvorbu objednávky. V případě stisknutí jednoho z tlačítek, přiřadí se odpovídající následující stav, dojde k zapsání hodnoty označující daný barevný korálek a pořadí v objednávce se posune na další pozici.

```
void SbAutomaton(){
  lcdPhoneIoS.clear();
  lcdPhoneIoS.print(0, 0, Sb);
  if (Sb==1) {if (tR==1) {Sb=2;
                          Obj[i]=1;
                          if (i<7) i=i+1;
                          printLcd3();
                          sendMessageToPhone3();
                         }
             if (tG==1) {Sb=3;
                         Obj[i]=2;
                         if (i<7) i=i+1;
                         printLcd3();
                         sendMessageToPhone3();
                         }
             if (tB==1) {S=4;
                         Obj[i]=3;
                         if (i<7) i=i+1;
                         printLcd3();
                         sendMessageToPhone3();
                        }
             if (tP==1) {Sb=5;
                         Obj[i]=4;
                         if (i<7) i=i+1;
                         printLcd3();
                         sendMessageToPhone3();
                        }
             if (tClear==1) {Sb=6;
                             Obj[1]=0;
                             Obj[2]=0;
                             Obj[3]=0;
                             Obj[4]=0;
                             Obj[5]=0;
                             Obj[6]=0;
                             i=1;
                             printLcd3();
                             sendMessageToPhone3();
                            }
             if (tFinish==1) {Sb=7;
                             }
  }
```

Obr. 43: Stav Sb1 v automatu pro tvorbu objednávky [zdroj: vlastní]

Tímto způsobem se provádí pro všechna čtyři tlačítka pro červenou, zelenou, modrou a fialovou barvu. V případě splnění podmínky pro stisknutí tlačítka vymazat, všechny buňky pole objednávky jsou naplněny hodnotou „0“. Pro všechny tyto podmínky je spuštěna funkce „printLcd3()“ pro zobrazení na LCD displej a zároveň funkcí „sendMessageToPhone3()“ pro odeslání zprávy do mobilního telefonu. V případě stisknutí tlačítka pro odeslání, přechází automat do stavu „Sb7“.

Stav Sb7 v „void SbAutomaton()“

Ve stavu „Sb7“ je dvojice podmínek. Buďto může uživatel stisknout tlačítko vymazat „tClear“ (obr.44) nebo tlačítko odeslat „tFinish“. Stavy „Sb2“, „Sb3“, „Sb4“, „Sb5“ a „Sb6“ kontrolují pouze uvolnění tlačítek.

```
if (Sb==7) {if (tClear==1) {Sb=6;
                            Obj[1]=0;
                            Obj[2]=0;
                            Obj[3]=0;
                            Obj[4]=0;
                            Obj[5]=0;
                            Obj[6]=0;
                            i=1;
                            printLcd3();
                            sendMessageToPhone3();;
                          }
```

Obr. 44: Stav Sb7 po stisknutí vymazání [zdroj: vlastní]

V případě uvolnění tlačítka pro odeslání (obr. 45) posílá se hotová objednávka vozidlu a vždy dochází k aktualizaci displeje. Čeká na potvrzení obdržení zprávy od vozidla a přechází dál, kde čeká zprávu o dokončení. Objednávka je tedy zobrazena na obrazovce mobilního telefonu pomocí příkazu „lcdPhone.print()“.

```
if (tFinish==0) {
                printLcd(0,"Server-order sent");
                lcdPhone.clear();
                lcdPhone.print(0, 0, "Obj:"+String(Obj[1])+String(Obj[2])+
                String(Obj[3])+String(Obj[4])+String(Obj[5])+String(Obj[6])+"-Sent");
                myRadio.stopListening();
                //WIFI NRF TRANSMIT
                dataTransmit.Obj[1]=Obj[1];
                dataTransmit.Obj[2]=Obj[2];
                dataTransmit.Obj[3]=Obj[3];
                dataTransmit.Obj[4]=Obj[4];
                dataTransmit.Obj[5]=Obj[5];
                dataTransmit.Obj[6]=Obj[6];
                dataTransmit.c=dataTransmit.c+1;
                dataTransmit.tx=1;
                dataTransmit.rx=2;
                dataTransmit.val=2;
                myRadio.openWritingPipe(addresses[0]);
                myRadio.write(&dataTransmit, sizeof(dataTransmit));
                lcd.setCursor (3,1);
                lcd.print("                ");
                lcd.setCursor (3,1);
                //lcd.print(dataTransmit.text);
                lcd.print(String(dataTransmit.Obj[1]));
                lcd.print(String(dataTransmit.Obj[2]));
                lcd.print(String(dataTransmit.Obj[3]));
                lcd.print(String(dataTransmit.Obj[4]));
                lcd.print(String(dataTransmit.Obj[5]));
                lcd.print(String(dataTransmit.Obj[6]));
                lcd.print(" c=");
                lcd.print(dataTransmit.c);
                lcd.print(" ");
                lcd.print(dataTransmit.tx);
                lcd.print(">");
                lcd.print(dataTransmit.rx);
                lcd.print(" ");
                lcd.print(dataTransmit.val);
                myRadio.openReadingPipe(1, addresses[0]); //automaton-action listen after RESET
                myRadio.startListening();
                printLcd(0,"Server-prijem");
                Sb=9;
```

Obr. 45: Stav Sb7 po uvolnění tlačítka odeslat [zdroj: vlastní]

Server přestane naslouchat příkazem „myRadio.stopListening()“ a předá pole objednávek do vozidla pomocí „dataTransmit“. Následně server otevře kanál pro zápis a zapíše data k přenosu příkazem „myRadio.openWritingPipe(adresses[o])“. Přenášený příkaz je zobrazen na LCD displeji funkcí „lcd.print“. Dále server otevře kanál pro čtení pomocí příkazu „myRadio.openReadingPipe(1, addresses [o])“ a začíná poslouchat mobilní telefon.

Stav Sb8 ve funkci „void SbAutomaton()“

Ve stavu „Sb8“ dochází ke kontrole dvojice podmínek. V případě že hodnota proměnné je menší než hodnota 7, automat přechází zpět do stavu „Sb1“. V případě rovnosti dochází k zobrazení zprávy na obrazovku mobilního telefonu a zároveň na LCD displej (obr. 46).

```
if (Sb==8) {if (i<7) {Sb=1;
                      }
            if (i==7) {Sb=7;
                       lcdPhone.clear();
                       lcdPhone.print(0, 0, "Over limit Obj[i]");
                       lcdPhone.print(1, 0, "press Finish");
                       lcdPhone.print(2, 0, "or Clear");
                       printLcd(0,"Over limit Obj[i]");
                       printLcd(1,"Press Finish");
                       printLcd(2,"or Clear");
                      }
}
```

Obr. 46: Stav "Sb8" v automatu pro tvorbu objednávky [zdroj: vlastní]

Stav Sb9 a Sb10 v „voidSbAutomaton()“

Po stavu „Sb7“ přechází do stavu „Sb9“ (obr. 47). V případě, že je síť k dispozici, server přijímá přenášené zprávy a zobrazí je na LCD displej. Dále potvrdí příjem zpráv a nastaví stav na „Sb10“.

```
if (Sb==9) {while (myRadio.available()) {
                myRadio.read(&dataRecieve, sizeof(dataRecieve));}
            if (lastc2<dataRecieve.c&&dataRecieve.rx==1) {
            lastc2=dataRecieve.c;
            lcd.setCursor (3,2);
            lcd.print("                ");
            lcd.setCursor (3,2);
            lcd.print(String(dataRecieve.Obj[1]));
            lcd.print(String(dataRecieve.Obj[2]));
            lcd.print(String(dataRecieve.Obj[3]));
            lcd.print(String(dataRecieve.Obj[4]));
            lcd.print(String(dataRecieve.Obj[5]));
            lcd.print(String(dataRecieve.Obj[6]));
            lcd.print(" c=");
            lcd.print(dataRecieve.c);
            lcd.print(" ");
            lcd.print(dataRecieve.tx);
            lcd.print(">");
            lcd.print(dataRecieve.rx);
            lcd.print(" ");
            lcd.print(dataRecieve.val);
            printLcd(0,"Server-conf received");
            delay(3000);
            printLcd(0,"Server-waiting end");
            Sb=10;              } }
```

Obr. 47: Stav "Sb9" v automatu pro tvorbu objednávky [zdroj: vlastní]

Ve stavu „Sb10“ je objednávka přijata z telefonu a následně ji zobrazí a dokončí. Automat přechází zpět do stavu „Sb1“.

```
if (Sb==10) {while (myRadio.available()) {
                myRadio.read(&dataRecieve, sizeof(dataRecieve));}
            if (lastc2<dataRecieve.c&&dataRecieve.rx==1) {
            lastc2=dataRecieve.c;
            lcd.setCursor (3,2);
            lcd.print("                ");
            lcd.setCursor (3,2);
            lcd.print(String(dataRecieve.Order[1]));
            lcd.print(String(dataRecieve.Order[2]));
            lcd.print(String(dataRecieve.Order[3]));
            lcd.print(String(dataRecieve.Order[4]));
            lcd.print(String(dataRecieve.Order[5]));
            lcd.print(String(dataRecieve.Order[6]));
            lcd.print(" c=");
            lcd.print(dataRecieve.c);
            lcd.print(" ");
            lcd.print(dataRecieve.tx);
            lcd.print(">");
            lcd.print(dataRecieve.rx);
            lcd.print(" ");
            lcd.print(dataRecieve.val);
            printLcd(0,"Server-order finished");
            delay(3000);
            printLcd(0,"Server-order beads");

            lcdPhoneIoS.clear();                        //IoS
            lcdPhoneIoS.print(0, 0, "Service request"); //IoS
            lcdPhoneIoS.print(1, 0, "device");          //IoS
            lcdPhoneIoS.print(0, 7, dataReceive.tx);    //IoS
            Sb=1;

           }
}
```

Obr. 48: Stav "Sb10" v automatu pro tvorbu objednávky [zdroj: vlastní]

8.2 Programování vozidla

Vozidlo je naprogramováno na základě automatů pro fungování vozidla (obr.20) se strategií A (obr.21). Vozidlo tedy přijede na start a vydá se k první bráně. Jestliže se barva zásobníku shoduje s barvou v objednávce, vysílá žádost o korálek. V opačném případě jede k dalšímu zásobníku.

V sekci před funkcí void setup ():

Inicializace proměnných pro vozidlo

Proměnné jsou uvedeny jako vstupy (obr.49), „FSpin“, „CSpin“, „LSpin“ atd. jsou inicializovány jako celočíselný datový typ „int“ a jsou uvedeny s piny, se kterými jsou spojeny. Jedná se o piny snímače sledování linky, který je připevněný na přední části vozidla.

Dále jsou zde uvedeny proměnné „LMF“, „RMF“, „LMB“, „RMB“ inicializované datovým typem „bool“, jelikož dávají pokyny pro levý a pravý motor vozidla pro pohyb dopředu nebo dozadu. Rychlost vozidla je možná od 0 do 255, a její hodnota je nastavena na hodnotu 100. Proměnné „LMFpin“, „RMFpin“, „LMBpin“, „RMBpin“ jsou uvedeny s propojenými piny.

```
int FSpin  = 34; //LINE front
int CSpin  = 40; //LINE central
int LSpin  = 42; //LINE left
int LLSpin = 44; //LINE left left
int RSpin  = 38; //LINE right
int RRSpin = 36; //LINE right right
int FSSpin = 46; //LINE front switch

bool FS;         //LINE front
bool CS;         //LINE central
bool LS;         //LINE left
bool LLS;        //LINE left left
bool RS;         //LINE right
bool RRS;        //LINE right right
bool FSS;        //LINE front switch

int LMFpin = 5;  //left motor forward
int RMFpin = 3;  //right motor forward
int LMBpin = 4;  //left motor backward
int RMBpin = 2;  //right motor backward
int LMpwm = 0;   //pwm left motor
int PMpwm = 0;   //pwm right motor
bool LMF ;       //left motor forward
bool RMF ;       //right motor forward
bool LMB ;       //left motor backward
bool RMB ;       //right motor backward
int vel = 100;   //velocity of the vehicle 0-255
```

Obr. 49: Inicializace proměnných v programu vozidla [zdroj: vlastní]

Funkce „printLcd“

Daná funkce „printLcd“ (obr.50) se využívá pro zobrazování dat na LCD displeji. Jsou dány dvě vstupní proměnné „line“ a „message“. Dojde k nastavení kurzoru na určené místo na obrazovce a pak zobrazí prázdné místo. Potom se na příslušném řádku zobrazí požadovaná zpráva.

```
void printLcd(int line, String message) {   //prints message at N line
  lcd.setCursor(0, line);
  lcd.print("                ");
  lcd.setCursor(0, line);
  lcd.print(message);
}

void ReadLine() {
  FS =   digitalRead(FSpin);  //LINE front
  CS =  !digitalRead(CSpin);  //LINE central
  LS =  !digitalRead(LSpin);  //LINE left
  LLS = !digitalRead(LLSpin); //LINE left left
  RS =  !digitalRead(RSpin);  //LINE right
  RRS = !digitalRead(RRSpin); //LINE right right
  FSS =  digitalRead(FSSpin); //LINE front switch
}
```

Obr. 50: Funkce "printLcd" a "ReadLine" [zdroj: vlastní]

Funkce „ReadLine()“ čte hodnoty pinů „FSpin“, „CSpin“, „LSpin“, „LLSpin“, „RSpin“, „RRSpin“, a „FSSpin“. Všechny hodnoty jsou znegovány kromě pinu „FSpin“ a předního spínače „FSSpin“.

Funkce „Going()“

Funkce „Going()“ (obr.51) plní zásadní roli pro pohyb vozidla. Nejprve přečte hodnoty pinů funkcí „ReadLine()“. Proměnné „LS“ (levý), „CS“ (centrální) a „RS“ (pravý)

jsou kontrolovány na možnost různých vstupů. Pomocí kontroly všech možností může vozidlo nastavit kola díky své rychlosti „vel“ a v potřebných místech se rychlost snižuje o hodnotu „30“.

```
void Going() {
  ReadLine();
  if ((!LS)&&(!CS)&&(!RS))          {analogWrite(LMFpin,0);
                                     analogWrite(LMBpin,0);
                                     analogWrite(RMFpin,0);
                                     analogWrite(RMBpin,0);
                                    }
  else if (( LS)&&( CS)&&( RS))     {analogWrite(LMFpin,vel);
                                     analogWrite(LMBpin,0);
                                     analogWrite(RMFpin,vel);
                                     analogWrite(RMBpin,0);
                                    }
  else if ((!LS)&&( CS)&&(!RS))     {analogWrite(LMFpin,vel);
                                     analogWrite(LMBpin,0);
                                     analogWrite(RMFpin,vel);
                                     analogWrite(RMBpin,0);
                                    }
  else if ((!LS)&&( CS)&&( RS))     {analogWrite(LMFpin,vel);
                                     analogWrite(LMBpin,0);
                                     analogWrite(RMFpin,vel-30);
                                     analogWrite(RMBpin,0);
                                    }
  else if ((!LS)&&(!CS)&&( RS))     {analogWrite(LMFpin,vel-30);
                                     analogWrite(LMBpin,0);
                                     analogWrite(RMFpin,0);
                                     analogWrite(RMBpin,0);
                                    }
  else if (( LS)&&( CS)&&(!RS))     {analogWrite(LMFpin,vel-30);
                                     analogWrite(LMBpin,0);
                                     analogWrite(RMFpin,vel);
                                     analogWrite(RMBpin,0);
                                    }
  else if (( LS)&&(!CS)&&(!RS))     {analogWrite(LMFpin,0);
                                     analogWrite(LMBpin,0);
                                     analogWrite(RMFpin,vel-30);
                                     analogWrite(RMBpin,0);
                                     }
  }
```

Obr. 51: Funkce "Going()" [zdroj: vlastní]

Funkce „Stopping()“

Funkce „Stopping()“ (obr.52) slouží pro zastavení vozidla. Nejprve přečte hodnoty pinů funkcí „ReadLine()“ a následně na piny „LMFpin“, „RMFpin“, „LMBpin“, „RMBpin“ jsou zapsány hodnoty „0“ pro ukončení otáčení kol tedy zastavení.

Vozidlo musí znát čísla pro odpovídající barevné korálky, funkcí „LedOrder()“ rozsvítí RGB LED diodu. V případě splnění podmínky „Obj[i]==1“ tak „LEDRpin“ je nastaven na hodnotu „1“ a ostatní piny na „0“. V případě že „Obj[i]==0“ tak jsou na hodnotu „1“ nastaveny všechny čtyři piny.

```
void Stopping() {
  ReadLine();
  analogWrite(LMFpin, 0);
  analogWrite(RMFpin, 0);
  analogWrite(LMBpin, 0);
  analogWrite(RMBpin, 0);
  }

void LedOrder() {
  if (Obj[i]==1){digitalWrite(LEDRpin,1);digitalWrite(LEDGpin,0);digitalWrite(LEDBpin,0);digitalWrite(LEDPpin,0);}
  if (Obj[i]==2){digitalWrite(LEDRpin,0);digitalWrite(LEDGpin,1);digitalWrite(LEDBpin,0);digitalWrite(LEDPpin,0);}
  if (Obj[i]==3){digitalWrite(LEDRpin,0);digitalWrite(LEDGpin,0);digitalWrite(LEDBpin,1);digitalWrite(LEDPpin,0);}
  if (Obj[i]==4){digitalWrite(LEDRpin,0);digitalWrite(LEDGpin,0);digitalWrite(LEDBpin,0);digitalWrite(LEDPpin,1);}
  if (Obj[i]==0){digitalWrite(LEDRpin,1);digitalWrite(LEDGpin,1);digitalWrite(LEDBpin,1);digitalWrite(LEDBpin,1);}
  }
```

Obr. 52: Funkce "Stopping()" a "LedOrder()" [zdroj: vlastní]

Funkce „Order Finished()“

Po ukončení objednávky se spustí funkce „OrderFinished()“, která volá funkci „LedOrder()“. Dále zobrazí na LCD displej hlášení o dokončení vozidla. Dále vozidlo zastaví poslouchání a vysílá proměnné „tx“, „rx“, „val“ a „c“ zvýšenou o hodnotu „1“. Probíhá také zobrazení těchto proměnných na displej a vozidlo je zastaveno pomocí funkce „Stopping()“.

```
void OrderFinished()
  {                   IoScount=Ioscount+1;
                      if (IoScount>=5) {IoScount=0;
                                        myRadio.stopListening();            //WIFI NRF TRAN
                                        dataTransmit.c=dataTransmit.c+1;
                                        dataTransmit.tx=2;
                                        dataTransmit.rx=1;
                                        dataTransmit.val=5;
                                        myRadio.openWritingPipe(addresses[0]);
                                        myRadio.write(&dataTransmit, sizeof(dataTransmit));
                                        };
                      LedOrder();
                      printLcd(0,"Vehicle-finished");
                      while(p==0){
                                  while (!((LLS)&&(RRS))) {Going();}
                                  p=p+1;if (p==5) {p=0;}
                                 }
                      myRadio.stopListening();            //WIFI NRF TRANSMIT
                      dataTransmit.c=dataTransmit.c+1;
                      dataTransmit.tx=2;
                      dataTransmit.rx=1;
                      dataTransmit.val=6;
                      myRadio.openWritingPipe(addresses[0]);
                      myRadio.write(&dataTransmit, sizeof(dataTransmit));
                      lcd.setCursor (3,1);
                      lcd.print("                ");
                      lcd.setCursor (3,1);
                      //lcd.print(dataTransmit.text);
                      lcd.print(" c=");
                      lcd.print(dataTransmit.c);
                      lcd.print(" ");
                      lcd.print(dataTransmit.tx);
                      lcd.print(">");
                      lcd.print(dataTransmit.rx);
                      lcd.print(" ");
                      lcd.print(dataTransmit.val);
                      Stopping();
                      delay(10000000);
  }
```

Obr. 53: Funkce "OrderFinished()" [zdroj: vlastní]

V rámci funkce void setup ():

Konfigurace vstupních a výstupních pinů

```
void setup() {
  pinMode(FSpin,  INPUT);
  pinMode(CSpin,  INPUT);
  pinMode(LSpin,  INPUT);
  pinMode(LLSpin, INPUT);
  pinMode(RSpin,  INPUT);
  pinMode(RRSpin, INPUT);
  pinMode(FSSpin, INPUT);

  pinMode(LMFpin, OUTPUT);
  pinMode(RMFpin, OUTPUT);
  pinMode(LMBpin, OUTPUT);
  pinMode(RMBpin, OUTPUT);
  pinMode(LEDRpin, OUTPUT);
  pinMode(LEDGpin, OUTPUT);
  pinMode(LEDBpin, OUTPUT);
```

Obr. 54: Konfigurace vstupních a výstupních pinů [zdroj: vlastní]

Piny jsou nastaveny jako vstupy a výstupy pomocí příkazu „pinMode“ (obr.54). Vstupy slouží jako zdroj informací vozidla a výstupy z programu na komponenty, které mají fungovat.

Funkce „void setup()“ v programu (obr.55) po konfiguraci všech vstupních a výstupních pinů je sériový monitor spuštěný příkazem „Serial.Begin(9600)“. Následně se spustí LCD displej a rozsvítí se jeho podsvícení. Na první tři řádky displeje se zobrazí „Vehicle-initializing“ a „TX:“, „RX:“. Poslední řádek je naplněn polem objednávky odeslaným do vozidla.

Kontrolují se hodnoty „LLS“ a „RRS“, dále se vozidlo rozjede a zastaví na první pozici. Zahájí se komunikace Wi-Fi a začne poslouchat pomocí funkce „myRadio.startListenig()“. Stav automatu je nastaven na „1“.

```
Serial.begin(9600);
lcd.init();              //LCD init or begin
lcd.backlight();         //LCD
printLcd(0,"Vehicle-initializing");
printLcd(1,"TX:");
printLcd(2,"RX:");
printLcd(3,"S"+String(S)+" Obj:"+String(Obj[1])+String(Obj[2])+
String(Obj[3])+String(Obj[4])+String(Obj[5])+String(Obj[6])+" o"+String(i)+"p"+String(p));
/* // demo*/  while (!((LLS)&&(RRS))) {Going();}     //the vehicle goes to first position
Stopping();           //automaton - action STOP
p=0;
myRadio.begin();                                    //WIFI NRF
myRadio.setChannel(0x60);
myRadio.setPALevel(RF24_PA_MAX);
//myRadio.setDataRate( RF24_250KBPS );
myRadio.openReadingPipe(1, addresses[0]);      //automaton - action listen
myRadio.startListening();
S=1;                                                //automaton - arrow S0 to S1
// S=2;LedOrder();  //demo
printLcd(0,"Vehicle-listening");
printLcd(3,"S"+String(S)+" Obj:"+String(Obj[1])+String(Obj[2])+
String(Obj[3])+String(Obj[4])+String(Obj[5])+String(Obj[6])+" o"+String(i)+"p"+String(p));
```

Obr. 55: Funkce "void setup()" [zdroj: vlastní]

V rámci funkce void loop ():

Nejprve dochází k testování komponentů (obr.56). Pomocí funkce „ReadLine()“ jsou proměnné čteny vozidlem a následně zobrazovány na sériovém monitoru pro kontrolu vstupních proměnných. LED diody jsou kontrolovány svícením jedna po druhé příkazem „digitalWrite(LEDRpin,1)“ se zpožděním jedné sekundy. Dále je proveden test obou motorů a test na sledování a zastavení na čáře.

```
void loop() {
  if (0==1) { ReadLine();                                          //LINE test
              Serial.print(FSS);Serial.print(FS);Serial.print("-");
              Serial.print(LLS);Serial.print(LS);Serial.print(CS);Serial.print(RS);Serial.println(RRS);
            }
  if (0==1) { digitalWrite(LEDRpin,1);digitalWrite(LEDGpin,0);digitalWrite(LEDBpin,0);delay(1000);digitalWrite(LEDPpin,0);delay(1000) //LED test
              digitalWrite(LEDRpin,0);digitalWrite(LEDGpin,1);digitalWrite(LEDBpin,0);delay(1000);digitalWrite(LEDPpin,0);delay(1000)
              digitalWrite(LEDRpin,0);digitalWrite(LEDGpin,0);digitalWrite(LEDBpin,1);delay(1000);digitalWrite(LEDPpin,0);delay(1000)
              digitalWrite(LEDRpin,0);digitalWrite(LEDGpin,0);digitalWrite(LEDBpin,0);delay(1000);digitalWrite(LEDPpin,1);delay(1000)
            }
  if (0==1) { Stopping();analogWrite(LMFpin,150);delay(500);    //MOTOR test
              Stopping();analogWrite(RMFpin,150);delay(500);
              Stopping();analogWrite(LMBpin,150);delay(500);
              Stopping();analogWrite(RMBpin,150);delay(500);
            }
  if (0==1) { Serial.println(millis()-lasttime);   //time of last loop
              lasttime=millis();
            }
  if (0==1) { Going();
              if ((LLS)&&(RRS)) {Stopping();delay(1000); }        //simple following test with stops on lines
            }
```

Obr. 56: Testování komponentů systému [zdroj: vlastní]

Stav S1 strategie A ve funkci „void loop()“

Vozidlo poslouchá Wi-Fi a podmínka pro přechod do dalšího stavu je kontrolována podmínkou „if“. V případě splněné podmínky dojde k zapsání do paměti pro všechny buňky pole objednávky. Dále je pořadí nastaveno na první pozici. Na prvním řádku LCD displeje se zobrazí „Vehicle-order obtained“ a po třech sekundách zobrazí „Vehicle-conf sending“. Pak vozidlo přestane poslouchat a začne vysílat. Hodnota „c“ se zvýší a přenese se pomocí příkazu „myRadio.Write(&dataTransmit, sizeof(dataTransmit))“. Na druhém řádku displeje se zobrazí všechny přenášené hodnoty a stav je nastaven na „2“. Získaná objednávka se zobrazí na čtvrtém řádku a je zavolána funkce „LedOrder()“ (obr.57).

```
if (S==1) { //initial state - listening
  while (myRadio.available()) {myRadio.read(&dataRecieve, sizeof(dataRecieve));}              //WIFI NRF listening
  if (/*lastcl<dataRecieve.c&&*/dataRecieve.rx==2&&dataRecieve.tx==1&&dataRecieve.val==2) {    //condition to S2 - if order recieved
    lastcl=dataRecieve.c;
    Obj[1]=dataRecieve.Obj[1];
    Obj[2]=dataRecieve.Obj[2];
    Obj[3]=dataRecieve.Obj[3];
    Obj[4]=dataRecieve.Obj[4];
    Obj[5]=dataRecieve.Obj[5];
    Obj[6]=dataRecieve.Obj[6];                                  //action to S2 - write in the order to memory
    i=1;
    printLcd(2,"RX: c="+String(dataRecieve.c)+" "+String(dataRecieve.tx)+">"+String(dataRecieve.rx)+
    " "+String(dataRecieve.val)+" "+String(Obj[1])+String(Obj[2])+String(Obj[3])+String(Obj[4])+String(Obj[5])+String(Obj[6]));
    printLcd(3,"S"+String(S)+" Obj:"+String(Obj[1])+String(Obj[2])+String(Obj[3])+String(Obj[4])+String(Obj[5])+String(Obj[6])+
    " o"+String(i)+"p"+String(p));
    printLcd(0,"Vehicle-order obted");  delay(3000);
    printLcd(0,"Vehicle-conf sending");             //action to S2 - send confirmation to server
    myRadio.stopListening();                        //WIFI NRF TRANSMIT
    dataTransmit.c=dataTransmit.c+1;
    dataTransmit.tx=2;
    dataTransmit.rx=1;
    dataTransmit.val=2;
    myRadio.openWritingPipe(addresses[0]);
    myRadio.write(&dataTransmit, sizeof(dataTransmit));
    printLcd(1,"TX: c="+String(dataTransmit.c)+" "+String(dataTransmit.tx)+">"+String(dataTransmit.rx)+" "+String(dataTransmit.val));
    S=2;                                            //arrow to S2
    printLcd(3,"S"+String(S)+" Obj:"+String(Obj[1])+String(Obj[2])+String(Obj[3])+String(Obj[4])+String(Obj[5])+String(Obj[6])+
    " o"+String(i)+"p"+String(p));
    LedOrder();
    printLcd(0,"Vehicle-going");
    } //end of condition
} //end of S==1
```

Obr. 57: Stav S1 [zdroj: vlastní]

Stav S2 strategie A ve funkci „void loop()“

Ve druhém stavu (obr.58) se vozidlo začne pohybovat funkcí „Going()“ a potom se zastaví díky funkci „Stopping()“ kontrolou podmínky „(LLS)&&(RRS)“ pro hledání značky pro zastavení (tj. oba dva boční senzory vjedou na černou). V této podmínce „if“ se proměnná pro pozici na cestě „p“ zvyšuje a kontroluje se, zdali dosáhla hodnoty „5“. V případě splnění „p==5“, opět dojde k vrácení na hodnotu „0“.

Dále dochází ke kontrole podmínky „Path[p]==Obj[i]“. Vozidlo zastaví a následně vysílá zprávu zásobníku o barevný korálek pouze v případě splnění podmínky. V poli „Path[p]“ jsou definované barvy zásobníků. Vozidlo zobrazí na displeji „Vehicle-asking bead“. Dále předá adresu vozidla (odesílatele) a adresu brány N (přijímače).

V neposlední řadě jsou přenášená data zobrazena na obrazovce, vozidlo začne opět poslouchat a stav je nastaven na „3“.

Pokud platí podmínka „Path[p]!==Obj[i]“, vozidlo zobrazí na displeji „Vehicle-wrong bead“ stav je nastaven zpět na „2“ a vozidlo jede dál.

```
if (S==2) {Going();                    //state S2 - following the line
  if ((LLS)&&(RRS)) {Stopping();
      p=p+1;
  if (p==5) {p=0;}
  if (Path[p]==Obj[i]){ printLcd(0,"Vehicle-asking bead");
      myRadio.stopListening();              //WIFI NRF TRANSMIT
      dataTransmit.c=dataTransmit.c+1;
      dataTransmit.tx=2;                    //adress of vehicle - sender
      dataTransmit.rx=Obj[i]+2;             //adress of GateN - receiver
      dataTransmit.val=3;
      myRadio.openWritingPipe(addresses[0]);
      myRadio.write(&dataTransmit, sizeof(dataTransmit));
      printLcd(1,"TX: c="+String(dataTransmit.c)+" "+String(dataTransmit.tx)+">"+
      String(dataTransmit.rx)+" "+String(dataTransmit.val));
      myRadio.startListening();
      S=3;                                   //automaton - arrow to S3 - waiting for the bead
      printLcd(3,"S"+String(S)+" Obj:"+String(Obj[1])+String(Obj[2])+String(Obj[3])+
      String(Obj[4])+String(Obj[5])+String(Obj[6])+" o"+String(i)+"p"+String(p));
      }
  else {printLcd(0,"Vehicle-wrong bead");
      delay(2000);
      S=2;                                   //automaton - arrow to S2 - continue to next stack
      printLcd(3,"S"+String(S)+" Obj:"+String(Obj[1])+String(Obj[2])+String(Obj[3])+
      String(Obj[4])+String(Obj[5])+String(Obj[6])+" o"+String(i)+"p"+String(p));
      printLcd(0,"Vehicle-going");
      }
    } //end of condition
 } //end of S==2
```

Obr. 58: Stav S2 [zdroj: vlastní]

Stav S3 strategie A ve funkci „void loop()“

Jestliže se automat nachází ve třetím stavu, tak dochází k poslouchání vozidla (obr.59). V případě, že je splněna pravdivost trojice podmínek „dataRecieve.rx==2 && dataRecieve.tx==Obj[i]+2 && dataRecieve.val==5“, vozidlo může pokračovat v akcích do dalšího stavu. Hodnoty s proměnnými „c“, „tx“, „rx“, „val“, a všechny buňky pole objednávky jsou zobrazeny na třetím řádku LCD displeje.

```
if (S==3) { //state - listening and waiting for obtain the right bead
  while (myRadio.available()) {myRadio.read(&dataRecieve, sizeof(dataRecieve));}   //WIFI NRF listening
  if (/*lastcl<dataRecieve.c&&*/dataRecieve.rx==2&&dataRecieve.tx==Obj[i]+2&&dataRecieve.val==5) {    //condition of S3 - confirmation that the right bead was obtained
     lastcl=dataRecieve.c;
     printLcd(2,"RX: c="+String(dataRecieve.c)+" "+String(dataRecieve.tx)+">"+String(dataRecieve.rx)+" "+String(dataRecieve.val)+" "+
     String(Obj[1])+String(Obj[2])+String(Obj[3])+String(Obj[4])+String(Obj[5])+String(Obj[6]));
     printLcd(0,"Bead obtained");  delay(3000);
     printLcd(0,"Vehicle-conf sending");                  //action to S2 - send confirmation to GateN
     myRadio.stopListening();                             //WIFI NRF TRANSMIT
     dataTransmit.c=dataTransmit.c+1;
     dataTransmit.tx=2;
     dataTransmit.rx=Obj[i]+2;
     dataTransmit.val=2;
     myRadio.openWritingPipe(addresses[0]);
     myRadio.write(&dataTransmit, sizeof(dataTransmit));
     printLcd(1,"TX: c="+String(dataTransmit.c)+" "+String(dataTransmit.tx)+">"+String(dataTransmit.rx)+" "+String(dataTransmit.val));
                                                          // end of actions on arrow to S2

   S=5;                                                   //arrow to S5
   Obj[i]=0;
   printLcd(3,"S"+String(S)+" Obj:"+String(Obj[1])+String(Obj[2])+String(Obj[3])+String(Obj[4])+String(Obj[5])+String(Obj[6])+" o"+String(o)+"p"+String(p));
   i=i+1;
   LedOrder();
   printLcd(0,"Vehicle-going");

   if ((Obj[1]==0)&&(Obj[2]==0)&&(Obj[3]==0)&&(Obj[4]==0)&&(Order[5]==0)&&(Order[6]==0)) {OrderFinished();}
   if ((i==7) {OrderFinished();}
   } //end of condition
} //end of S==3
```

Obr. 59: Stav S3 [zdroj: vlastní]

Dále se na prvním řádku zobrazí získaný korálek a po časové prodlevě tří sekund se zobrazí „vehicle-conf sending“. Potvrzení je provedeno zastavením poslouchání

vozidla, přenášením „tx=2“, „rx=Obj[i]+2“, „val=2“ a zvýšením proměnné „c“ pomocí „ myRadio.write(&dataTransmit, sizeof(dataTransmit))“.

Stav je nastaven na „5“ a „Obj[i]=0“. Na LCD displeji se zobrazí objednávka a pořadí v poli je zvýšeno. Následně se už pouze volá funkce „LedOrder()“. Objednávka je dokončena v případě pravdivosti podmínky všech polí objednávky „Obj[i]=0“.

Stav S4, S5, S6 strategie A ve funkci „void loop()“

Ve stavu „S4“ (obr.60) dochází pouze k aktivaci pohybu vozidla, nastavení hodnoty „i=1“ a přechodu do prvního stavu za předpokladu uplynutí předepsaného časového intervalu. To je realizováno podmínkou „if (stoptime<=millis())“.

Stav „S5“ kontroluje hodnotu proměnné „d<10“. V případě shody přechází stav do stavu „S2“, jinak je zobrazena zpráva „vehicle-service“ a je nastaven časový interval příkazem „stoptime=millis()+30000“ na třicet sekund. Stav je nastaven na „6“.

Stav „S6“ pouze kontroluje podmínku uplynutí časového intervalu pomocí příkazu „stoptime<=millis()“. Proměnná „d“ je nastavena na „1“ a stav na hodnotu „2“. Tímto způsobem je naprogramováno vozidlo tak, aby fungovalo podle strategie A.

```
if (S==4) {
  if (stoptime<=millis()) {
    Going();
    i=1;
    S=1;                                          //arrow to S1
  }
} //end of S==4
if (S==5) {
  if (d<10) {
      S=2;                                        //arrow to S2
  }
  }
  else {
    printLcd(0,"Vehicle-service");
    stoptime=millis()+30000;
    S=6;                                          //arrow to S6
  }
} //end of S==5
if (S==6) {
  if (stoptime<=millis()) {
    d=1;
    S=2;                                          //arrow to S2
  } //end of condition
} //end of S==6

} //end of loop
```

Obr. 60: Stav S4, S5, S6 [zdroj: vlastní]

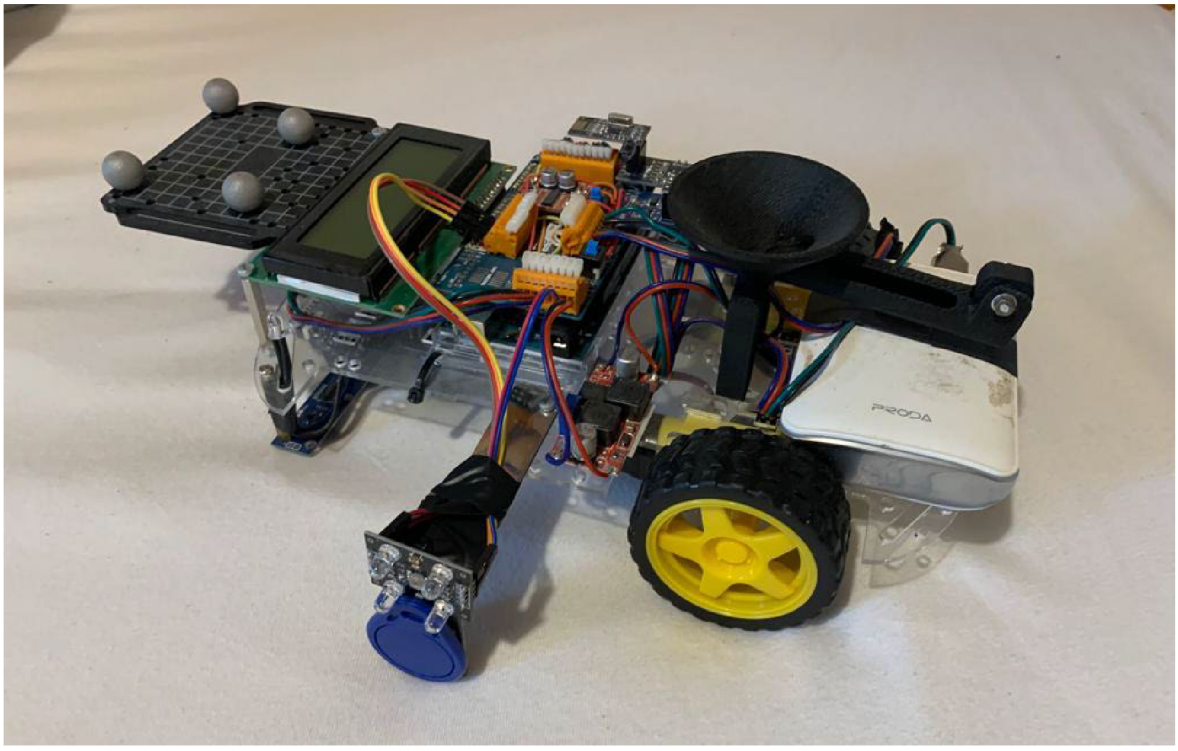
8.3 Vytváření mapy barevných zásobníků

Pro zapisování barevných zásobníků na dráze jako do mapy je potřebné vybavit vozidlo komponentem pro detekci barvy (TCS230). Princip zapisování pořadí bran s barevnými zásobníky je uveden v předchozí kapitole (6.7).

Detektor barvy TCS230

Tento modul pro detekci barvy umožňuje vozidlu rozpoznat barvy v jeho těsné blízkosti. Je vybavený senzorem TCS230, který se skládá z velkého počtu fotodiod s odlišnými barevnými filtry a čtveřicí bílých LED diod pro osvícení (obr.61).

Princip fungování spočívá v osvícení bílými LED diodami a následně po odrazu světla zpět do senzoru je schopný na svém výstupu vytvořit frekvenci výskytu barevných složek RGB. V případě, že například přiložíme modrý předmět, odrazové světlo obsahuje v největší míře modrou složku. Ta má za důsledek vytvoření největšího proudu skrze fotodiodu s modrým filtrem. Proud je dále převeden na frekvenci. Pro senzor platí že, čím větší výskyt určité barvy, tím je její frekvence na výstupu menší [24].



Obr. 61: Detektor barvy TCS230 na vozidle [zdroj: vlastní]

Implementace v programu vozidla

V sekci před funkcí void setup ():

Nastavení propojovacích pinů

Nejdříve je za potřebí provést nastavení propojovacích pinů modulu pomocí „#define“, který umožňuje pojmenovat konstanty před kompilací programu. Takže konstanty nezabírají v Arduinu žádný programový paměťový prostor na čipu. Propojené piny detektoru „So-S3“ a „OUT“ s piny Arduina jsou vidět na obrázku 62.

```
#define pinS0 6                          //color sensor
#define pinS1 7
#define pinS2 8
#define pinS3 9
#define pinOut 10
```

Obr. 62: Nastavení propojovacích pinů [zdroj: vlastní]

Deklarace a inicializace mapy

Vytvoření pole pro zapisování pořadí barevných zásobníků je uskutečněno deklarací mapy „Path[5] = {0,0,0,0,0}“. Ta slouží pro ukládání pořadí barevných zásobníků (obr. 63).

```
int S=0;                        //state of vehicle´s main automaton and arrow to S0
int lastcl = 0;                 //aux var for wifi comm
int i=1;                        // actual position of needed bead in order
int Obj[6]={0,0,0,0,0,0};       //order
//int Obj[6]={1,3,2,3,1,2};     //demo order
int p=0;
//int Path[5]={0,1,2,3,2};      //demo map of stacks
int Path[5]={0,0,0,0,0};        //map of stacks

int IoScount = 0;               //IoS
```

Obr. 63: Deklarace proměnné pro ukládací [zdroj: vlastní]

V rámci funkce void setup ():

Konfigurace vstupních a výstupních pinů

V dalším kroku je potřebné nastavit propojovací piny jako vstupní a výstupní pomocí příkazu „pinMode“ (obr.64). Dále je zapotřebí inicializovat detektor barev pomocí „digitalWrite(pinSo, HIGH)“ a „digitalWrite(pinS1, LOW)“.

```
pinMode(pinS0, OUTPUT);     //color sensor pins
pinMode(pinS1, OUTPUT);
pinMode(pinS2, OUTPUT);
pinMode(pinS3, OUTPUT);
pinMode(pinOut,INPUT);
digitalWrite(pinS0, HIGH);  //color sensor init
digitalWrite(pinS1, LOW);
```

Obr. 64: Konfigurace vstupních a výstupních pinů [zdroj: vlastní]

V rámci funkce void loop ():

Detekce barvy

Nejdříve si deklarujeme proměnné pro uložení frekvencí barev „int frekR, frekG, frekB“. Dále přejdeme k nastavení měření červené barvy (obr.65) pomocí „digitalWrite(pinS2, LOW)“ a „digitalWrite(pinS3, LOW)“. Načtení frekvence barvy se provádí pomocí „frekR = pulseIn(pinOut, LOW)“. Obdobným způsobem nastavíme i měření všech ostatních barev.

```
int frekR, frekG, frekB;                    // proměnné pro uložení frekvencí barev

digitalWrite(pinS2,LOW);                    // nastavení měření červené barvy
digitalWrite(pinS3,LOW);
delay(50);                                  // krátká pauza pro přesné měření
frekR = pulseIn(pinOut, LOW);               // načtení frekvence červené barvy
digitalWrite(pinS2,HIGH);                   // nastavení měření zelené barvy
digitalWrite(pinS3,HIGH);
delay(50);                                  // krátká pauza pro přesné měření
frekG = pulseIn(pinOut, LOW);               // načtení frekvence zelené barvy
digitalWrite(pinS2,LOW);                    // nastavení měření modré barvy
digitalWrite(pinS3,HIGH);
delay(50);                                  // krátká pauza pro přesné měření
frekB = pulseIn(pinOut, LOW);               // načtení frekvence modré barvy
```

Obr. 65: Nastavení měření barev [zdroj: vlastní]

Následně dojde k vytištění načtených frekvencí po sériové lince příkazem „SerialPrint(frekR)“. Dále už se pouze kontroluje čtveřice podmínek pro detekci barvy (obr.66). V případě, že je hodnota „frekR < 60“ dojde k vytisknutí informace po sériové lince o detekci červené, a do pole „Path[p]“ se uloží hodnota „1“. Následně je proměnná „p“ zvýšena o hodnotu „1“ pro přesunutí na další pole. Obdobným způsobem se kontrolují podmínky pro ostatní barvy jen s tím rozdílem, že se do pole „Path[p]“ zapíše hodnota, která odpovídá požadované barvě.

```
Serial.print("R: ");                        // vytištění načtených frekvencí po sériové lince
Serial.print(frekR);
Serial.print(" | G: ");
Serial.print(frekG);
Serial.print(" | B: ");
Serial.print(frekB);
// pokud je detekována nějaká z barev, vytiskneme
// informaci po sériové lince
if (frekR < 60) {Serial.print(" | Detekce R. ");
                 Path[p]=1;
                 p=p+1;
                 }
if (frekG < 60) {Serial.print(" | Detekce G. ");
                 Path[p]=2;
                 p=p+1;
                 }
if (frekB < 60) {Serial.print(" | Detekce B. ");
                 Path[p]=3;
                 p=p+1;
}
if (frekR < 60 & frekG < 60 ) {Serial.print(" | Detekce P. ");
                 Path[p]=4;
                 p=p+1;
}
Serial.println();                           // ukončení řádku
delay(850);                                 // volitelná pauza pro přehledné čtení výsledků
```

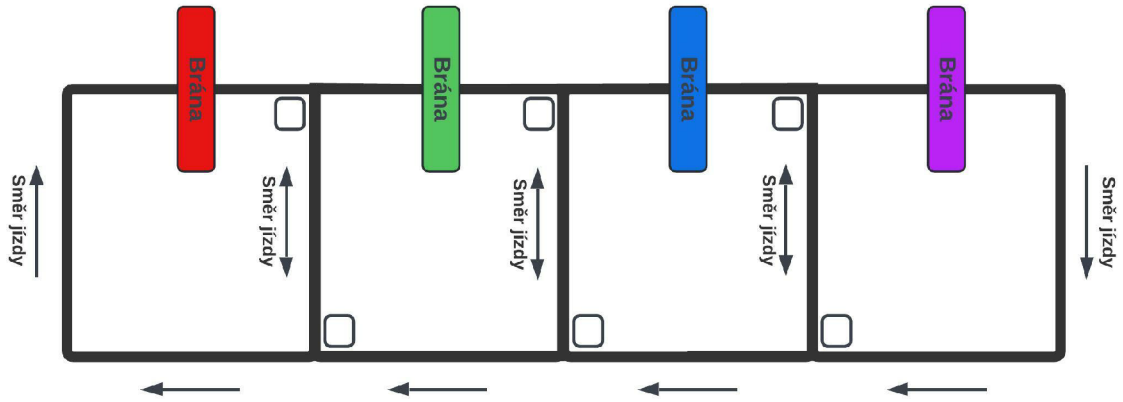
Obr. 66: Podmínky pro detekci barev [zdroj: vlastní]

9 DALŠÍ PRÁCE NA MODELU CHYTRÉ TOVÁRNY

Dosavadní model chytré továrny má zcela jistě spoustu dalších možností, jak by se dal ještě vylepšit. Je navržen tak, aby se dal co nejjednodušším způsobem dále rozšiřovat a rozvíjet. Záleží tedy pouze na dalších studentech, kteří rozhodnou o směru vylepšení modelu chytré továrny.

Dalším zajímavým vylepšením, které by zvýšilo efektivnost sbírání definované objednávky spočívá v úpravě strategie na dráze obsahující prvky pro zkrácení trasy vozidla (obr. 67). Při mém návrhu jsem vytvořil strategii, která rozhoduje o vrácení se na první pozici v případě, že se na další pozici v objednávce nachází stejná barva korálku. Tento způsob rozhodování sice do jisté míry vylepšuje efektivitu sbírání barevných korálků, ale naráží na limity, které spočívají v pohybu vozidla pouze jedním směrem na každém prvku pro zkrácení dráhy.

Ideálním řešením by bylo tedy navrhnout strategii, která bude schopna po prvcích jezdit obousměrně. Snížila by se především najetá vzdálenost vozidla a tím pádem i doba, za kterou vozidlo dokončí objednávku.



Obr. 67: Dráha pro princip vozidla s obousměrnými zkracujícím prvky [zdroj: vlastní]

Další možností úpravy strategie na tomto typu dráhy by bylo možné v případě, že by nebylo vyžadováno sesbírání barevných korálků v pořadí, které si uživatel stanoví v objednávce. Tento případ by umožnil implementaci metody umělé inteligence pro nalezení nejkratší cesty pro sesbírání barevných korálků. Dráhu je schopné převést do orientovaného grafu s ohodnocenými hranami podle vzdálenosti. Vozidlo by po stanovení pořadí barevných zásobníků v prvním okruhu následně rozhodovalo, pro který korálek jet nejkratší cestou podle aktuální polohy na dráze.

10 ZÁVĚR

Hlavním cílem této práce je vylepšení a rozšíření řídícího systému modelu chytré továrny. Jednotlivé komponenty jsou vybaveny různými moduly a senzory propojenými s mikrokontrolerem Arduino Mega 2560. Vzájemně spolu komunikují pomocí wifi. Všechny prvky systému jsou naprogramovány na bázi struktury grafů automatů.

Pro zabezpečení komunikace v rámci tohoto systému slouží server, který byl vytvořen především pro příjem objednávek od uživatele pomocí mobilní aplikaci Blynk a odesílání do jednotlivých vozidel. Vozidlo s aktivní objednávkou jezdí po trase mezi zásobníky korálků a pomocí různých strategií vybírá barvy dle objednávky.

Výchozím stavem bylo použití jednoho vozidla a prosté oválné trasy. Pro tuto konfiguraci byly navrženy a otestovány tři různé řídící strategie. Do systému byl přidán monitoring průběhu procesu na mobilní telefon. Dále byl navržený mechanismus Internet of Services, kde žádost o servis je generována na základě počtu sesbíraných korálků.

Dále byla vytvořena strategie pro výběr vhodného vozidla z více současně provozovaných vozidel. Byly provedeny další návrhy dráhy s rozvětvenými cestami umožňující optimalizaci pohybu. V jednom případě jsou zásobníky korálků umístěny mimo hlavní ovál do odboček. Vozidlo se může rozhodnout, jestli zajede k zásobníku nebo pokračuje kratší cestou dál. Ve druhé variantě dráha obsahuje prvky pro zkrácení trasy vozidla, kdy se vozidlo vždy vrací na první pozici.

Dále byla vytvořena strategie, díky které vozidlo umí vytvářet mapu barevných zásobníků, kterou je schopno přepisovat v případě, že dojde k promíchání pořadí v průběhu pohybu vozidla. Dané vozidlo dokáže zastavit přímo u požadované barvy zásobníku s korálky, aniž by muselo zastavovat u každého zásobníku. Pro tuto strategii bylo vozidlo vybaveno modulem pro detekci barev.

Takový řídící systém má mnoho aplikací v továrnách a může být implementován v průmyslových odvětvích pro přepravu různých předmětů vozidlem bez přítomnosti člověka.

SEZNAM POUŽITÉ LITERATURY

[1] BALÁTĚ, Jaroslav. *Automatické řízení*. 2., přeprac. vyd. Praha: BEN - technická literatura, 2004. ISBN 80-7300-148-9.

[2] OLEHLA, Miroslav, Slavomír NĚMEČEK a Ivan ŠVARC. *Automatické řízení*. Liberec: Technická univerzita v Liberci, 2013. ISBN 978-80-7372-972-1.

[3] *Základy automatického řízení: teorie* [online]. [cit. 09.02.2022]. Dostupné z: https://adoc.pub/zaklady-automatick-ho-rzen60cea5ec952c399041c5c6df24aaf90571246.html

[4] *Stavový automat*. Elektromyš [online]. [cit. 10.02.2022] Dostupné z: http://www.elektromys.eu/clanky/stm8_12_automat/clanek.html

[5] *Intelligent Diagramming* | Lucidchart. [online]. Copyright © [cit. 10.02.2022]. Dostupné z: https://www.lucidchart.com/pages/

[6] *The Industrial Revolution: From Industry 1.0 to Industry 4.0. Momentum - Industrial Marketing* [online]. Copyright © 2022 [cit. 10.02.2022]. Dostupné z: https://www.seekmomentum.com/the-evolution-of-industry-from-1-to-4/

[7] *Průmysl 4.0* | Lean Industry. [online]. Copyright © 2020 leanindustry.cz. Vytvořilo reklamní a grafické studio [cit. 10.02.2022]. Dostupné z: https://www.leanindustry.cz/prumysl-4-0/

[8] KORBEL, Petr. *Průmyslová revoluce 4.0: Za 10 let se továrny budou řídit samy a produktivita vzroste o třetinu*. Hospodářské noviny [online]. 2015-05-17 [cit. 2015-09-20]. https://byznys.hn.cz/c1-64009970-prumyslova-revoluce-4-0-za-10-let-se-tovarny-budou-ridit-samy-a-produktivita-vzroste-o-tretinu

[9] *Jsme připraveni na Průmysl 4.0 - EZÚ*. [online]. Copyright © 2021 [cit. 10.02.2022]. Dostupné z: https://ezu.cz/aktuality/prumysl-4-0-2/

[10] EMnify | *IoT Connectivity Platform*. [online]. Copyright © 2022 All rights reserved. [cit. 11.02.2022]. Dostupné z: https://www.emnify.com/

[11] *9 Pillars Of Industry 4.0* [online]. Copyright © Shenzhen BestAI Internet Co., Ltd . 2019 All Rights Reserved [cit. 11.02.2022]. Dostupné z: https://www.kindpng.com/imgv/TxwJoh_9-pillars-of-industry-4-0-hd-png-download/

[12] *Co je to IoT?* | *IoTPort*. [online]. Copyright © 2022 České Radiokomunikace a.s. [cit. 11.02.2022]. Dostupné z: https://www.iotport.cz/iot-novinky/ostatni-clanky-o-iot/co-to-je-iot

[13] *Industrial Internet of Things (IIoT) - Definition*. [online]. Copyright © 2022 Trend

Micro Incorporated. All rights reserved. [cit. 11.02.2022]. Dostupné z: https://www.trendmicro.com/vinfo/us/security/definition/industrial-internet-of-things-iiot

[14] *Smart PPE for Connected Worker Safety in Industry 4.0 - Intellinium* [online]. Copyright © 2020 Intellinium [cit. 11.02.2022]. Dostupné z: https://intellinium.io/why-iiot-is-different-from-iot/

[15] Průmysl 5.0 – když lidé pracují se stroji | Rockwell Automation. *Select a Region* [online]. Dostupné z: https://www.rockwellautomation.com/cs-cz/company/news/blogs/prumysl-5-0-kdyz-lide-pracuji-se-stroji.html

[16] *Home - SL Controls* [online]. Copyright © [cit. 16.02.2022]. Dostupné z: https://slcontrols.com/wp-content/uploads/2019/12/Smart-Factory-Evolution.jpg

[17] *Tulip | The Industry's Leading Frontline Operations Platform* [online]. Dostupné z: https://tulip.co/glossary/what-is-a-smart-factory-and-what-it-means-for-you/

[18] *Arduino - Home* [online]. Dostupné z: https://www.arduino.cc/en/Guide/Introduction

[19] *Vývojové desky Arduino – actrl.cz.* [online]. Dostupné z: http://actrl.cz/blog/index.php/2016/vyvojove-desky-arduino/

[20] *Arduino store* [online]. Dostupné z: https://store.arduino.cc/

[21] JANSA, Martin. Výukový model řízení výroby s principy Průmyslu 4.0: The educational model of production control with principles of Industry 4.0. Liberec: Technická univerzita v Liberci, 2019. Diplomové práce. Technická univerzita v Liberci. Vedoucí práce Radek Votrubec.

[22] VALLINAYAGAM, Backia Siva. Control System of Smart Factory Model. Liberec: Technická univerzita v Liberci, 2020. Diplomové práce. Technická univerzita v Liberci. Vedoucí práce Radek Votrubec.

[23] Osvěžte si základy automatizace. FactoryAutomation.cz. FactoryAutomation.cz • Časopis o automatizaci a robotice [online]. Copyright © 2014 [cit. 15.05.2022]. Dostupné z: https://factoryautomation.cz/osvezte-si-zaklady-automatizace-znate-definici-regulace/

[24] Arduino detektor barvy TCS230 | Návody Drátek. *Webový magazín o ARDUINU | Návody Drátek* [online]. Dostupné z: https://navody.dratek.cz/navody-k-produktum/arduino-detektor-barvy.html

SEZNAM OBRÁZKŮ

SEZNAM PŘLOH

Příloha 1 – Obsah přiloženého CD

- **zdrojový kód programů**

  server.ino

  car.ino

  stacks.ino

- **schémata automatů**

  automat_distribuce_objednávek.jpeg

  automat_fungování_vozidla.jpeg

  automat_fungování_vozidla_odbočení_na_trase.jpeg

  automat_fungování_vozidla_strategie_A.jpeg

  automat_fungování_vozidla_strategie_B.jpeg

  automat_fungování_vozidla_strategie_C.jpeg

  automat_fungování_vozidla_určení_barev_na_dráze.jpeg

  automat_fungování_vozidla_vrácení_na_trase.jpeg

  automat_kontrolních_senzorů.jpeg

  automat_tvorba_objednávky.jpeg